TÉCNICO LISBOA

Writing a pricer in a recombining tree for CDS Options using an HJM model (Cheyette)

Daniela Caeiro Miranda

Thesis to obtain the Master of Science Degree in

Mathematics and Applications

Supervisor(s): Prof. Cláudia Rita Ribeiro Coelho Nunes Philippart
Dr. Simon Moreau

November 2017

Para o meu pai...

Acknowledgments

I would first like to thank my teacher Cláudia for all the support and very useful advice for my future. The topic of this thesis was proposed by Simon from BNP Paribas. I'm very thankful to him indeed, because without his help it wouldn't be possible to do this study. Also, I would like to thank Claude for all the advice, like to program in C++, and his passionate participation. Along this thesis when I had doubts or when I was a little bit lost, they helped my to find the correct direction.

I must express my profound gratitude to my mom and to João for the continuous encouragement along my academic journey and for all the unconditional love. This would not have been possible without them. Thank you (Obrigada por tudo).

PS: Although I know that you cannot see this, I dedicate this thesis to you my father, I know that you would be proud of me. (Embora eu saiba que não consigas ver o que consegui alcançar, eu sei que ficarias orgulhoso. Por isso dedico esta tese a ti Pai. Da sempre tua Daniela.)

Resumo

Tendo em conta os derivados de crédito, os Credit Default Swaps (CDSs) e Credit Default Swap Options (CDS options ou CDSwaptions) estão entre os produtos de crédito mais populares. Com o crescimento dos mercados financeiros, surge a necessidade de pesquisar novos métodos e modelos que nos permitam efectuar o *pricing* destes instrumentos financeiros complexos.

O principal objetivo desta tese é o cálculo do preço de uma opção de crédito de nome único, numa árvore recombinante baseada num modelo Cheyette; sendo estruturada da seguinte forma:

No capítulo 1 descrevemos brevemente as nossas motivações e objetivos, também mencionamos alguns trabalhos e modelos que nos inspiraram e a abordagem apresentada ao longo da tese.

No capítulo 2 fornecemos os conceitos e metodologia necessários para o desenvolvimento do nosso trabalho. Em particular, descrevemos um modelo Cheyette proposto por Krekel et al. [2009].

No capítulo 3, adaptamos o método numérico Li et al. [1995] para o nosso produto financeiro de interesse (opções Credit Swap padrão) e desenvolvemos a implementação do nosso algoritmo, que chamamos de algoritmo Cheyette.

No capítulo 4 fazemos um estudo de sensibilidade exaustiva no algoritmo Cheyette.

Finalmente, no capítulo 5 descrevemos em breve o nosso estudo, ou seja, as realizações e algumas sugestões para trabalhos futuros.

Palavras-chave: intensidade padrão, probabilidades de sobrevivência direta, volatilidade implícita, taxas de juro, Credit Default Swaps, Credit Default Swap options.

Abstract

Concerning credit derivatives, Credit Default Swaps (CDSs) and Credit Default Swap Options (CDS options or CDSwaptions) are amongst of the most popular credit products. With the constantly growing of the financial markets, arises the need of researching for new methods and models that allow us to price these complex financial instruments.

The main goal of this thesis is the computation of the price of a credit single-name option, in a recombining tree based on a Cheyette model. The structure of the thesis is the following:

In Chapter 1 we briefly describe our motivations and objectives, we also mention some works and models that inspired us and the approach presented along the thesis.

In Chapter 2 we provide the necessary concepts and methodology for the development of our work. In particular, we describe a Cheyette model proposed by Krekel et al. [2009].

In Chapter 3, we write an adaptation of the Li et al. [1995] numerical method for our financial product of interest (Credit Default Swap options), and we develop the implementation of our algorithm, which we call Cheyette algorithm.

In Chapter 4 we do an exhaustive sensitivity study in the Cheyette algorithm.

Finally, in Chapter 5 we describe in short our study, namely, the achievements and some suggestions for future works.

Keywords: default intensity, forward survival probabilities, implied volatility, Credit Default Swaps, Credit Default Swap options.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation and Objectives

In the financial market world, credit derivatives have had an increasingly relevant role. Credit Default Swaps (CDSs) and Credit Default Swap Options (CDS options or CDSwaptions) are among the most popular products. With the constantly growing of the financial markets, arises the need of researching for new methods and models that would allow us to price these complex financial instruments in a more efficient, parsimonious and accurate way.

In this direction, the Quantitative Research team from BNP Paribas Corporate & Institutional Banking (CIB) proposed the topic of this thesis, which is to write a pricer in a recombining tree for a CDS option based on a Markov representation of a Heath, Jarrow and Morton (HJM) model (i.e, Cheyette [1996] model). In other words, based on a default intensity model (derived in Krekel et al. [2009]), our main goal is to create an algorithm that computes the price for a CDS option at the valuation date. In order to do that, we will adapt the numerical procedure of Li et al. [1995] to our particular credit derivative. Then, we will perform a sensitivity study where we intend to study how a change in the input parameters impacts important features of CDS options.

Since this dissertation was made in collaboration with BNP Paribas CIB, it has a huge financial component. Along this thesis, we chose not to be too formal, as we usually are in the field of Mathematics. Therefore, we decide to provide in Chapter 2 some basic financial concepts and methodology that are fundamental for the comprehension of our study.

1.2 Topic Overview

In this section, we provide a brief explanation of the main models and methods that will serve as an inspiration and guidance in our study.

- Heath et al. [1992]: given the (default-free) zero coupon bond prices, Heath, Jarrow and Morton (HJM) developed a general framework for modelling the interest rate dynamics, in continuous time. Their stochastic structure is directly related with the evolution of the forward rate curve where the only requirements are the initial forward rate curve, also known as initial yield curve, and the volatility structure for all forward rates. They choose all the term structure of the forward rates as a state variable (i.e, an economic fundamental) and showed that their evolution could depend on the entire path taken by this term structure. Because of this, their model does not ensure that the term structure evolution is Markovian with respect to a finite number of state variables (Krekel et al. [2009]). Since the forward and short rate processes may not have a Markovian character, it implies some problems in numerical implementations, for instance, non-recombining trees.

- Ritchken and Sankarasubramanian [1995]: in order to find a solution for the numerical issues of the HJM framework, they provide some conditions on the volatility structure of forward rates that allows the evolution of the term structure to be represented by a two-dimensional state variable Markov process, in the sense that, given these two variables, the term structure can be recovered.

- Cheyette [1996]: given an arbitrary initial term structure, he derived a class of non-arbitrage term structure models with a Markovian character for a finite number of state variables by restricting the forward rate volatility function. Therefore, his framework avoids some numerical problems and guarantees a Markovian-character that does not hold, in general, for an HJM model.

- Li et al. [1995]: in this paper, Li, Ritchken and Sankarasubramanian built efficient algorithms for pricing European and American claims based on the HJM model with the same class of volatility structures presented in Ritchken and Sankarasubramanian [1995]. Moreover, this seems to be the first paper that develops efficient algorithms for this kind of claims.

- Krekel et al. [2009]: based on the work of Heath et al. [1992], Cheyette [1996] and Schönbucher [2000], in this paper, they used the model developed by Cheyette in order to model the default intensity and apply their study to real data using two different approaches involving the finite-difference method.

1.3 Thesis Outline

In this section we briefly explain the contents of the different chapters of this thesis:

- In Chapter 2, at first place we will provide some fundamental concepts and conventions about credit derivatives. Also, some important financial concepts that will be essential for a better understanding of our study. Then, we will present a default intensity model derived in Krekel et al. [2009], and finally we will describe and provide important information regarding the credit derivatives used in this thesis (CDSs and CDS options);

- In Chapter 3, we will describe, in detail, how it is possible to adapt the numerical method from Li et al. [1995] for CDS options and how the algorithm developed for this particular product was implemented;

- In Chapter 4, we will present some numerical results using the algorithm described in the previous Chapter. In particular, we will perform a sensitivity study where we aim to analyse how a change in the input parameters impacts important features of CDS options;
- In Chapter 5, we will provide some final remarks like what we consider to be the major contributions of this thesis and suggest some future works.

Chapter 2

Concepts and Methodology

In this Chapter, we start by providing in sections 2.1 and 2.2, some basic financial concepts, results, definitions and notations, that will be essential in the derivation of the models that we will use to accomplish our goal. We recall briefly, that the main goal of our thesis is to compute the price of a particular single-name credit derivative in a recombining tree. Next, in section 2.3, we describe the approach used in order to model the default intensity. Finally, in section 2.4, we explain the mechanism of a CDS contract and the valuation of CDS options. Also, we introduce a key concept that will play a important role on the realization/interpretation of our results in the next Chapter. This concept is the implied volatility (for CDS options).

2.1 Credit Derivatives and Conventions

In general, a derivative is a *"financial instrument whose value depends on (or derives from) the values of other, more basic, underlying variables"* (Hull [2012]). There are many types of derivatives, such as, options, forward contracts, swaps, etc. Since we pretend to price a credit product, we are interested in credit derivatives which consist of *"privately held negotiable bilateral contracts that allow users to manage their exposure to credit risk."* [1]. Credit risk is also called default risk, measures the risk that a contract will not be honoured (Hull [2012]).

Following Schönbucher [2000], associated with the credit derivative, we have its payoff, that depends on the occurrence of a credit event, namely:

- Payment default;
- Bankruptcy or insolvency;
- Protection filling;
- Ratings downgrade below a given threshold;
- Changes in the default intensity;
- Payment moratorium.

Concerning the types of credit derivatives we may have [14]:

- Credit single-name derivatives (payoff depends on the default of a single underlying credit);
- Credit multi-name derivatives (payoff depends on the default of several underlying credits).

In this thesis we deal only with the first type of contracts (single-name credit derivatives), in particular the so-called Credit Default Swap (CDS) contracts and Credit Default Swaptions (CDS options or CDSwaptions). Actually, CDSs are the most popular single-name credit derivatives (Hull [2012]).

Along this thesis we assume that the interest rates and forward rates are continuously compounded and we use preferably the time to maturity information, instead of maturity date, usually denoted by T, and current time, t, where $t < T$. In particular, $t = 0$ can be seen as the valuation date, i.e, when the value of the financial product is determined. Trivially, the time to maturity is $T - t$, which is the interval of time, in years, until the maturity of the contract, as we can observe in figure 2.2.

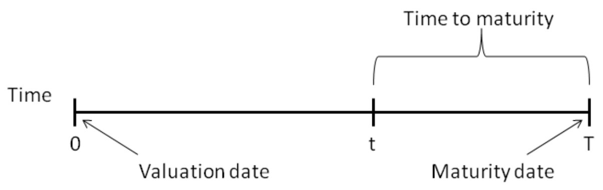


Figure 2.1: Timeline of a financial instrument.

Moreover, following the **Actual/365** convention [1], we convert the dates t and T (usually in the format `day-month-year`) to year fraction. So the time to maturity is calculated as the fraction of the number of days between the two dates [2] divided by 365 (Brigo and Mercurio [2006], O'Kane [2008]).

2.2 Some Fundamental Financial Concepts

We start by explaining the notion of term-structure. A term-structure relates a certain financial variable or parameter in function of its time to maturity (Filipović [2009]). Besides other classification terms, the term-structure of financial variables can be default-free or defaultable, in the sense that for the first ones (default-free term-structure) there is no default risk associated and for the second ones (defaultable term structure) there is some default risk associated with these quantities.

Next, we let τ denote the default time [3] and we consider the following indicator function:

$$\mathbf{1}_{\{\tau>t\}} = \begin{cases} 1, & \text{if } \tau > t \\ 0, & \text{if } \tau \leq t \end{cases} \tag{2.1}$$

[1] In Finance, there are several day-count conventions.

[2] Please note that the dates were transformed in *"numerical format"*, with the help of software *Excel*.

[3] Please, see Krekel et al. [2009] for more details about τ.

This indicator function represents a survival indicator, i.e, is equal to one if a default did not occur before time t and zero otherwise. Obviously, we assume that if a default happens it will never occur again, that is, default is a one time event where $0 < \tau < +\infty$ (O'Kane [2008]).

Following Krekel et al. [2009], we will use the *"overline"* notation for the defaultable (financial) instruments (i.e, that are subject to default risk) and all the definitions of this defaultable quantities, that we will present next, are only valid for times before default ($\tau > t$). Also, we assume zero recovery in case of default before time t, in the sense that, for $\tau \leq t$ the payoff at time t is given by the survival indicator (2.1).

At this point, we are able to provide the basic information concerning each term structure. Firstly, we introduce the concept of zero-coupon bond which can be default-free or defaultable. A default-free zero-coupon bond has a face value of one unit of currency and a defaultable zero-coupon bond (i.e, subject to default risk) has a payoff at maturity T of $\mathbf{1}_{\{\tau>T\}}$. For both cases, zero-coupon bonds have no intermediate payments. We let,

- $P(t,T)$ denote the price of a default-free zero-coupon bond at current time t, when its maturity is T, where $t < T$. In particular, when $t = 0$, we can interpret $P(0,T)$ as the present value of one monetary unit to be paid at maturity.
- $\bar{P}(t,T)$ denote the price of a defaultable zero-coupon bond at current time t, when its maturity is T, where $t < T$, given that default did not occur before time t (i.e., given that $\mathbf{1}_{\{\tau>t\}} = 1$, which we omit in the text, to ease the notation).

For each maturity T, concerning the first case ($P(t,T)$) the holder of a zero-coupon bond has the guarantee of a payment of one unit of currency at maturity T. On the other hand, for the second case ($\bar{P}(t,T)$), if a default did not occur before time t the price of a defaultable zero-coupon bond will be given by $\bar{P}(t,T)$, otherwise the indicator function assumes the value zero and therefore $\bar{P}(t,T)$ does not need to be zero.

Obviously, we have that, $P(T,T) = 1, \forall T$ and $\bar{P}(T,T) = 1, \forall T$ (if $\tau > T$), such that [4]

$$P(t,T_1) \geq P(t,T_2) \text{ and } \bar{P}(t,T_1) \geq \bar{P}(t,T_2), \forall t < T_1 < T_2 \tag{2.2}$$

This means that the zero-coupon bond prices are decreasing as a function of time to maturity (Schönbucher [2003]).

Moreover, the following assumptions hold (Filipović [2009], Schönbucher [2003]):

Assumptions 2.1. *1. The transactions of zero-coupon bonds are cost free (i.e, the market is frictionless);*

2. $P(t,T)$ is differentiable w.r.t. T, that is, $\dfrac{\partial P(t,T)}{\partial T}$ exists, $\forall T$.

3. $\bar{P}(t,T)$ is differentiable w.r.t. T, that is, $\dfrac{\partial \bar{P}(t,T)}{\partial T}$ exists, $\forall T$.

[4] However, sometimes in the financial markets are observed negative interest rates, which makes the equation wrong. So here, we consider that we have non-negative interest rates.

4. *In order to avoid arbitrage opportunities, the following holds:*

$$P(t,T) > \bar{P}(t,T) \geq 0, \forall t < T \tag{2.3}$$

Concerning the absence of arbitrage assumption, we note that the zero-coupon bond prices for the default-free and defaultable case are non-negative functions of maturity and the prices of the default-free zero-coupon bonds are always greater than the defaultable zero-coupon bonds.

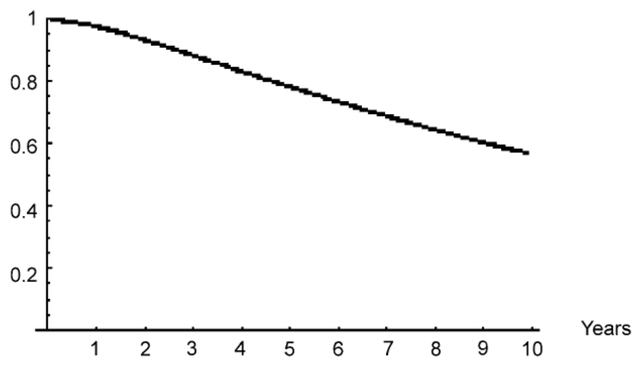


Figure 2.2: Term-structure of default-free zero-coupon bond prices of US Treasury Bonds, March 2002 (Figure 2.2 from Filipović [2009]).

Regarding the default-free zero-coupon bonds, in figure 2.2 we plot the price of a zero-coupon bond, $P(0,T)$, at time $t = 0$ (valuation date) as a function of the time to maturity (T). This curve is called zero-bond curve or term structure of discount factors (Brigo and Mercurio [2006]). We can see that if the time to maturity increases, the values of the zero-coupon bond decreases, and according to a decreasing smooth curve that starts at $P(0,0) = 1$. This last comment, is on the line with the second item of assumptions 2.1, in the sense that being smooth, it is differentiable w.r.t. T. However, in practice, the zero-coupon bonds may not be traded at all maturities.

Another key concept is the definition of interest rates. Interest rates can be seen as the amount of money a borrower promises to pay to a lender (Hull [2012]), that is, there is of common sense that receive a given amount of money tomorrow is not the same that receive the same amount today (Brigo and Mercurio [2006]). Interest rates can be fixed along time, time changing in a deterministic way or stochastic. Also, they can be compound in different ways, depending also if they behave in discrete or continuous time. There are many types of interest rates, for instance, forward rates are interest rates that can be established in current time t for an investment in a future time period. They are closely linked with the current term structure of discount factors (Brigo and Mercurio [2006]).

Next we present some useful definitions about forward rates [5].

Definition 2.1. *Let $f(t,T_1,T_2)$ denote the continuously compounded default-free forward rate over the*

[5]The definitions and notations are inspired by Schönbucher [2000].

period $[T_1, T_2]$ at time t, computed as follows (for $t \leq T_1 < T_2$):

$$f(t, T_1, T_2) = \frac{\ln P(t, T_1) - \ln P(t, T_2)}{T_2 - T_1} \tag{2.4}$$

Definition 2.2. *Let $\bar{f}(t, T_1, T_2)$ denote the continuously compounded defaultable forward rate over the period $[T_1, T_2]$ at time t, computed as follows (for $t \leq T_1 < T_2$):*

$$\bar{f}(t, T_1, T_2) = \frac{\ln \bar{P}(t, T_1) - \ln \bar{P}(t, T_2)}{T_2 - T_1} \tag{2.5}$$

Concerning the default-free case, we can write ,equivalently, (2.4) as,

$$f(t, T_1, T_2) = \frac{1}{T_2 - T_1} \ln \left(\frac{P(t, T_1)}{P(t, T_2)} \right) \Leftrightarrow \frac{P(t, T_1)}{P(t, T_2)} = e^{f(t,T_1,T_2)(T_2 - T_1)}$$

which means that, considering the future time period $[T_1, T_2]$ and two default-free zero-coupon bonds with maturities T_1 and T_2 where $T_1 < T_2$, the price proportion of $P(t, T_1)$ with respect to $P(t, T_2)$ grows exponentially according to the forward rate $f(t, T_1, T_2)$.

In particular, if T_2 tends to T_1 we obtain,

$$\lim_{T_2 \to T_1^+} f(t, T_1, T_2) = - \lim_{T_2 \to T_1^+} \frac{\ln P(t, T_1) - \ln P(t, T_2)}{T_1 - T_2} = - \frac{\partial \ln P(t, T_1)}{\partial T_1}$$

In the defaultable case, the definitions and properties are similar, with the corresponding change in the notation (namely, the *"overline"* notation) and taking into account that they are only valid for times t before default.

Therefore we have the following definitions,

Definition 2.3. *The continuously compounded instantaneous default-free forward rate at time t with maturity T is defined as:*

$$f(t, T) = - \frac{\partial \ln P(t, T)}{\partial T} \tag{2.6}$$

where $t < T$ and in case the derivative of $P(t, T)$ w.r.t. T exists.

Definition 2.4. *The continuously compounded instantaneous defaultable forward rate at time t with maturity T, is defined as:*

$$\bar{f}(t, T) = - \frac{\partial \ln \bar{P}(t, T)}{\partial T} \tag{2.7}$$

where $t < T$ and in case the derivative of $\bar{P}(t, T)$ w.r.t. T exists.

Regarding the default-free case, we can interpret $f(t, T)$ as a forward rate at time t whose maturity is almost equal to T, that is, when $\Delta T \to 0$ then $f(t, T) \approx f(t, T, T + \Delta T)$ (Brigo and Mercurio [2]). Similarly for the defaultable case, for times before default.

It follows trivially by inverting, respectively, (2.6) and (2.7), that the price of zero-coupon bonds can be written as:

$$P(t, T) = e^{-\int_t^T f(t,s)ds} \text{ and } \bar{P}(t, T) = e^{-\int_t^T \bar{f}(t,s)ds} \tag{2.8}$$

Considering (2.6) and (2.7) in the particular case of $T = t$, we define, respectively, the instantaneous default-free short rate and instantaneous defaultable short rate at time t, as follows:

$$r(t) = f(t,t) \text{ and } \bar{r}(t) = \bar{f}(t,t) \tag{2.9}$$

Moreover, we introduce the concepts of bank account and discount factor(Krekel et al. [2009], Brigo and Mercurio [2006]).

Definition 2.5. *Let $B(t)$ denote the default-free bank account and $\bar{B}(t)$ denote the defaultable bank account. Given an investment of one monetary unit at time 0 (and therefore $B(0) = 1$ and $\bar{B}(0) = 1$), then:*

$$B(t) = \exp\left(\int_0^t r(s)ds\right) \textit{ and } \mathbf{1}_{\{\tau>t\}}\bar{B}(t) = \mathbf{1}_{\{\tau>t\}} \exp\left(\int_0^t \bar{r}(s)ds\right) \tag{2.10}$$

For the default-free case (resp. defaultable case, if $\tau > t$) the rate at which the bank account increases, r (resp. $\bar{r}$), is the instantaneous default-free (resp. defaultable) short rate. Here, r and $\bar{r}$ may be deterministic or stochastic.

Unlike the zero-coupon bond which has a payoff of one unit of currency at maturity, the bank account accumulates according to rate r given an investment of one unit of currency at the time 0. So the first one decreases and the second one increases as the time to maturity increases.

Conversely, we define the default-free discount factor and the defaultable discount factor, respectively, as follows:

$$D(t) = \exp\left(-\int_0^t r(s)ds\right) \text{ and } \mathbf{1}_{\{\tau>t\}}\bar{D}(t) = \mathbf{1}_{\{\tau>t\}} \exp\left(-\int_0^t \bar{r}(s)ds\right) \tag{2.11}$$

In general, the discount factor between times t and T, where $t < T$ (and for the defaultable case $\tau > t$) is given by,

$$D(t,T) = \frac{B(t)}{B(T)} = \exp\left(-\int_t^T r(s)ds\right) \text{ and } \bar{D}(t,T) = \frac{\bar{B}(t)}{\bar{B}(T)} = \exp\left(-\int_t^T \bar{r}(s)ds\right) \tag{2.12}$$

which may be interpreted as the value of one monetary unit at present time t, to be received at time T.

Concerning the default-free case, in case of the short rates be deterministic, $r(s) = r$, $\forall\, t \leq s \leq T$, then the formulas bank account and discount factor simplify, respectively, to (Brigo and Mercurio [2]):

$$B(t) = e^{rt} \text{ and } D(t,T) = e^{-r(T-t)} \tag{2.13}$$

It follows from the definitions of discount factor (2.12) and zero-coupon bond, that the following holds (Brigo and Mercurio [2006]):

- If the interest rates are deterministic, $r(s) = r$, $\forall\, t \leq s \leq T$, then $D(t,T) = P(t,T)$, $\forall (t,T)$;
- If the interest rates are stochastic then, we will see later on that $P(t,T)$ can be seen as the expectation of $D(t,T)$ under a certain probability measure.

Now we introduce a crucial concept in our thesis, the default intensity, denoted by $\lambda(t)$, which is also known as a hazard rate. The default intensity accounts for the conditional likelihood that default will occur in the next small interval of time, given that it did not occur yet and all other available information. A default occurs with probability $\lambda(t)dt$, which corresponds to the probability of a default occurring in the interval $[t, t + dt]$, given that it survives at least up to time t. Mathematically, one may define λ as follows,

$$\lambda(t) = \lim_{dt \to 0^+} \frac{P[t \leq \tau < t + dt | \tau \geq t]}{dt} \tag{2.14}$$

where τ is, as previously, the time of default.

In practice, $\lambda(t)$ is not deterministic because it can changes over time randomly with arrival of new information in the market. So, it is necessary to assume that the default intensity follows a stochastic process. Then, the default can be seen as a Poisson process where the time-dependent intensity $\lambda(t)$ (stochastic process), given the information of the path $\{\lambda(s) : 0 \leq s \leq t\}$ (Duffie and Singleton [2003]).

Moreover, it is necessary to consider that the interest rates are also stochastic.

Following Duffie and Singleton [2003], we let $S(t, T)$ denote the forward survival probability at time t for maturity T, that is, is the likelihood that a default will not occur (at least) until time T, given that it survives until time t. We assume that $S(t, T)$ is strictly positive and is differentiable w.r.t. T, $\forall T$ (Duffie and Singleton [2003]). Also we assume that the interest rates and time to default are independent in order to simplify our framework (Schönbucher [2003]).

So we let $s(t, T)$ denote the instantaneous forward default intensity at time t for maturity T, given as follows:

$$s(t, T) = -\frac{1}{S(t, T)} \frac{\partial S(t, T)}{\partial T} = -\frac{\partial}{\partial T} \ln(S(t, T)) \tag{2.15}$$

Conversely, if we invert the last equality of equation (2.15) we obtain,

$$S(t, T) = e^{-\int_t^T s(t,u)du} \tag{2.16}$$

Clearly, $s(t, T)$ can be seen as an hazard rate and $S(t, T) \in [0, 1]$ (which justifies the name of (forward) survival probability).

Following Krekel et al. [2009], we can go further and define forward survival probability as a function of the zero-coupon bonds prices.

Definition 2.6. *The forward survival probability at time t, for maturity T, is defined in the following way:*

$$S(t, T) = \frac{\bar{P}(t, T)}{P(t, T)} \tag{2.17}$$

Moreover, from (2.8), the forward survival probability (2.17) can be written as:

$$S(t, T) = e^{-\int_t^T [\bar{f}(t,u) - f(t,u)]du} \tag{2.18}$$

Then, from (2.16) and (2.18), we can rewrite the instantaneous forward default intensity at time t for maturity T as follows:

$$s(t,T) = \bar{f}(t,T) - f(t,T) \tag{2.19}$$

In particular, if $T = t$ we define the default intensity at time t in the following way:

$$\lambda(t) = s(t,t) = \bar{r}(t) - r(t) \tag{2.20}$$

So besides other interpretations, the default intensity, can also be interpreted as the difference between the instantaneous defaultable short rate and the default-free short rate.

Thus, we can infer that there is an analogy between forward survival probabilities and zero-coupon bond prices, since both are non-negative, differentiable for all maturities and decreasing as function of T. Also, from (2.17) and given that $\tau > T$, we can easily see that $S(T,T) = 1$, $\forall T$ and if we have all the information relative to zero-coupon bond prices for all maturities we will also have all the information of survival probabilities for all maturities. Then, similarly to the zero-bond curve we can build a term structure of (forward) survival probabilities, which we will call, in the rest of the thesis, survival curve.

Moving forward, we know that the price of every instrument, under a so-called *risk-neutral* probability measure (some martingale measure), is the expected value of its discounted expected payoff, where the (continuously compounded) default-free short rate is used for discounting. So we can write zero-coupon bond price at time t with maturity T, as the discounted expectation of its payoff at time T, under some risk-neutral measure (O'Kane [2008]). Thus, recalling the assumption that no default occurred until time t ($\tau > t$), we have that the zero-coupon bond prices are given by (Schönbucher [2003]):

$$P(t,T) = \mathbf{E}\left[\exp\left(-\int_t^T r(s)ds\right) \times 1\right] \text{ and } \bar{P}(t,T) = \mathbf{E}\left[\exp\left(-\int_t^T r(s)ds\right) \mathbf{1}_{\{\tau > T\}}\right] \tag{2.21}$$

By the assumption of independence of the interest rate and the default time we have that,

$$\bar{P}(t,T) = \mathbf{E}\left[\exp\left(-\int_t^T r(s)ds\right)\right] \mathbf{E}[\mathbf{1}_{\{\tau>T\}}] = P(t,T)\mathbf{E}[\mathbf{1}_{\{\tau>T\}}] = P(t,T)S(t,T) \tag{2.22}$$

Since we assumed that the default intensity follows a stochastic process, default arrives according to a Cox Process. Then, we can write the forward survival probability as, $S(t,T) = \mathbf{E}[\mathbf{1}_{\{\tau>T\}}]$ where,

$$\begin{aligned} \mathbf{E}\left[\mathbf{1}_{\{\tau>T\}}\right] =& \mathbf{E}[\mathbf{E}[\mathbf{1}_{\{\tau>T\}} | \{\lambda(s), t < s < T\}]] = \\ =& \mathbf{E}[P(\tau > T | \{\lambda(s), t \leq s \leq T\})] = \mathbf{E}\left[\exp\left(-\int_t^T \lambda(s)ds\right)\right] \end{aligned}$$

We note that the first equality follows from the tower property (Mikosch [1999]) and the last expectation is w.r.t. the distribution of the default intensity. And as before, given all the information until time t, $S(t,T)$ is the probability of survival until the (future) time T.

Finally from the last equalities we van write the defaultable zero-coupon bond price as:

$$\bar{P}(t,T) = \mathbf{E}\left[exp\left(-\int_t^T (r(s)+\lambda(s))ds\right)\right] \tag{2.23}$$

To summarize, we can see that there is a parallel between the default-free and the survival cases. Both have a term structure (P and S, respectively) that is decreasing as a function of time to maturity, with initial value equal to one. The default-free zero-coupon bond (P) can be written as a function of the forward interest rates (f), whereas in the survival case, the forward survival probability (S) can be seen as function of the forward default intensities (s). When $T = t$, in the default-free case the instantaneous interest rate is r and in the survival case is λ, both assumed to be positive in order to ensure non-arbitrage. In table 2.1 we present schematically this analogy.

Default-free	Survival
$T \to P(t,T)$	$T \to S(t,T)$
$f(t,T)$	$s(t,T)$
$r(t)$	$\lambda(t)$
No arbitrage $r(t) > 0$	No arbitrage $\lambda(t) > 0$

Table 2.1: Parallel between the default-free and survival framework, inspired by O'Kane [2008].

2.3 Modelling the default intensity

In this section we follow essentially the approach, notation and results from Krekel et al. [2009]. We have decided not to include in this thesis some stochastic calculus that Krekel et al. [2009] present (namely measure changing), as this is beyond the scope of our thesis.

Firstly, we start by describing a default intensity model from Krekel et al. [2009]. The default intensity model is based on the Cheyette Markov representation of the HJM model (see the previous Chapter for a short description of these models).

Given the initial term structure of the instantaneous forward default intensities, $s(0,T)$, $\forall T$, we assume that for each t and T, s is such that:

$$ds(t,T) = \mu_s(t,T)dt + \sigma_s(t,T)dW(t), \tag{2.24}$$

where $\mu_s(t,T)$ and $\sigma_s(t,T)$ are the drift and the volatility parameters, respectively, and $\{W(t),t\}$ is a standard Brownian motion under some martingale measure. Therefore s is a diffusion process, and we assume that both μ_s and σ_s are (eventually) stochastic processes, adapted to the same filtration as the forward default intensities. Moreover, it follows from the definition of diffusion, that s is a Markov process (Mikosch [1999]).

Following Schönbucher [2000], the non-arbitrage drift condition of the forward default intensity is given by,

$$\mu_s(t,T) = \sigma_s(t,T)\int_t^T \sigma_f(t,u)du + \sigma_f(t,T)\int_t^T \sigma_s(t,u)du + \sigma_s(t,T)\int_t^T \sigma_s(t,u)du \tag{2.25}$$

where $\sigma_f(t,u)$ is the volatility of the forward rate $f(t,u)$.

In order to get a simpler formula, Schönbucher [2000] assumed that the Brownian motion $\{W(t),t\}$ depends either on the forward interest rates f or the forward default intensity s, but not on both. That is if,

$$\sigma_s(t,T) \neq 0 \Rightarrow \int_t^T \sigma_f(t,u)du = 0 \text{ and } \sigma_f(t,T) \neq 0 \Rightarrow \int_t^T \sigma_s(t,u)du = 0 \tag{2.26}$$

So, if we assume the independence of $s(t,T_1)$ and $f(t,T_2)$ for all $t \leq \min(T_1,T_2)$, the non-arbitrage drift condition for the forward default intensity (2.25) can be rewritten as:

$$\mu_s(t,T) = \sigma_s(t,T)\int_t^T \sigma_s(t,u)du \tag{2.27}$$

Moreover, if we substitute this last equation in (2.24), it follows that s is such that:

$$ds(t,T) = \left(\sigma_s(t,T)\int_t^T \sigma_s(t,u)du\right)dt + \sigma_s(t,T)dW(t). \tag{2.28}$$

Therefore, under the independence assumption, the Itô's process of the forward default intensity can be written only in terms of the volatility.

Furthermore, we assume that:

$$\sigma_s(t,T) = \sigma_\lambda(t)e^{-k_\lambda(T-t)} \tag{2.29}$$

where k_λ is a constant such that $k_\lambda \neq 0$ and $\{\sigma_\lambda(t) = \sigma_s(t,t), t\}$ is some adapted stochastic process, then from appendix D of Krekel et al. [2009], the evolution of the default intensity, $\{\lambda(t),t\}$, is given by the following two dimensional Markov diffusion processes:

$$d\lambda(t) = \mu(t,\lambda(t),\phi_\lambda(t))dt + \sigma_\lambda(t)dW(t),\ \lambda(0) = \lambda_0 \tag{2.30}$$

$$d\phi_\lambda(t) = (\sigma_\lambda^2(t) - 2k_\lambda\phi_\lambda(t))dt,\ \phi_\lambda(0) = 0 \tag{2.31}$$

where λ_0 is the initial value of the default intensity and the drift term $\mu(t,\lambda(t),\phi_\lambda)$ is given by:

$$\mu(t,\lambda(t),\phi_\lambda(t)) = k_\lambda[s(0,t) - \lambda(t)] + \phi_\lambda(t) + \frac{d}{dt}s(0,t) \tag{2.32}$$

and $\phi_\lambda(t)$ is the accumulated variance of the forward default intensity up to time t, that is,

$$\phi_\lambda(t) = \int_0^t \sigma_s^2(u,t)du = \int_0^t \sigma_\lambda^2(u)e^{-2k_\lambda(t-u)}du \tag{2.33}$$

where the second equality was obtained from (2.29).

Also, it follows (in view of the previous equations) that we may define the forward survival probability at time t with maturity T as (please see Krekel et al. [2009] and derivation in Cheyette [1996]):

$$S(t,T) = \frac{S(0,T)}{S(0,t)} e^{\beta_s(t,T)(s(0,t)-\lambda(t))-\frac{1}{2}\phi_\lambda(t)\beta_s^2(t,T)} \tag{2.34}$$

where

$$\beta_s(t,T) = \int_t^T e^{-k_\lambda(u-t)}du = \frac{1-e^{-k_\lambda(T-t)}}{k_\lambda} \tag{2.35}$$

From (2.34), we can see that the forward survival probability, $S(t,T)$, is written in terms of the values of the forward survival probabilities at time 0, $S(0,T)$ and $S(0,t)$, the default intensity at time t, $\lambda(t)$, and the accumulated variance of the forward default intensity up to time t. We recall that, by definition, $s(0,0) = \lambda(0) = \lambda_0$.

From now on, we will consider that the volatility of the default intensity, $\sigma_\lambda(t)$, is stochastic. In particular, we will follow the same approach as Li et al. [1995] used for interest rates, namely:

$$\sigma_\lambda(t) = \sigma[\lambda(t)]^\gamma \tag{2.36}$$

where $\gamma > 0$ (usually called the elasticity parameter) and σ are constant.

We recall that in order to assure non-arbitrage, one needs to have strictly positive $\lambda(.)$F, which in some cases may be difficult to check. In fact, only in the case of $\gamma = 1$ there exists a sufficient condition (see Krekel et al. [9]). For this reason, in the rest of the work (unless otherwise stated) we assume that $\gamma = 1$.

2.4 CDS and CDS Options

When we were studying Krekel et al. [2009], we found that there was a mistake or a typo in equation (21) of this paper. For that reason, in this section we derive the correct formula, that will be presented in equation (2.47).

However, before we present the mathematical derivations, we still need to provide some (financial) definitions related with CDS contracts, which may be found in the standard literature of credit derivatives. According to O'Kane [2008], a Credit Default Swap (CDS) is an over-the-counter contract [6] between two parties, the protection buyer and the protection seller, where the purpose is to protect one of them, the protection buyer, against default of a specific company or sovereign entity (issuer). The company or issuer is known as the **reference entity** and the default by the company is the **credit/default event**. Usually a CDS has maturity of five years and we assume that the notional value (or face value) of the CDS contract is one unit of currency.

Moreover, this contract is characterized by two payment legs: the premium (or fixed) leg and the protection (or floating) leg, which will be paid at specific times defined in the contract, depending on the occurrence of a default event, and that we denote by $T_i, i = 1, \dots, n$, such that T_n is equal to the maturity of the CDS.

[6] An over-the-counter contract *" is a telephone and computer-linked of dealers. Trades are done over the phone and are usually between two financial institutions or between a financial institution and one of its clients. "* (Hull [2012])

So, before explaining, in more detail, these payment legs, we need to define the some quantities, that is, we let,

- $\delta_i = T_{i+1} - T_i$ be the time interval between payments of a CDS contract, that is, the time (in years) between the dates T_i and T_{i+1}, for $i = 0, ..., n-1$.
- ξ be the credit swap rate or also called the premium payment rate. Usually it is represented as basis points (bp) in the market quotes (i,e. 1 bp = 0.01%).

Then, regarding that n^* can be equal to $0, 1, ..., n-1$, the payments legs are defined as follows:

- **Premium/Fixed leg**: the protection seller receives regular payments of $\xi\delta_i$, at times $T_1, T_2, \ldots, T_{n^*}$ from the protection buyer, where $n^* = n$ if default did not occur until maturity (T_n); otherwise ($0 < n^* < n$), the regular payments end as soon as default happens and, in this case, n^* is such that $T_{n^*} < \tau < T_{n^*+1}$. Usually, these payments are made quarterly, that is, in intervals of 3 months.
- **Protection/Floating leg**: if a default occurs before the contract maturity date (T_n) then the protection seller has to compensate the protection buyer in $(1-R)$ times the notional value at time T_{n^*+1}, where R is the recovery rate [7]. Usually for CDS contracts the recovery rate is 40%.

Since we have already defined the mechanism of a CDS contract, we are now able to define forward CDS contracts and CDS options.

Following Krekel et al. [2009], a forward CDS contract is a CDS contract starting on a future date, T_k, say (with $T_k < T_n$). Then the above definitions hold, except that the payments start only from time T_{k+1} onwards.

Next we define the meaning of a CDS option or CDSwaption, which is an option on a underlying forward CDS contract. Usually, it is of European style, which means that the holder of the option can only exercise the option at the expiry date (T_k) [8] (O'Kane [2008]). There are two kinds of CDSwaptions. The first one (usually called a payer option) is an option that allows the holder to buy protection from default, and the second one (called receiver option) allows the holder to sell protection. So we can see the payer and receiver CDSwaptions as call and put options, respectively.

Formally, we have the following definitions,

Definition 2.7. *A payer (receiver) CDSwaption with expiry (or exercise) date T_k and default swap rate ξ is a call (put) option on a forward CDS starting at time T_k. So the holder of the option has the right, not the obligation, to buy (sell) a CDS at the predefined value ξ, if there has been no credit event until time T_k.*

Therefore, ξ can be seen as the fixed amount paid in the fixed leg and also the strike price of the CDS option.

[7] *"The recovery rate for a bond is normally defined as the bond's market value a few days after a default, as a percent of its face value"*, that is, the recovery rate (R) can be seen as a fraction of the notional value of the CDS contract (Hull [2012]).

[8] In Finance, instead of denoting T_k as the maturity date of the CDS option we denote as expiry or exercise date in order to distinguish from the maturity of the underlying forward CDS contract.

Thus, if there has been no default until the maturity of the option, the holder of a payer CDSwaption has the right to enter into a long position on a forward CDS contract and, on the other hand, the holder of a receiver CDSwaption has the right to enter into a short position on a forward CDS contract.

2.4.1 Valuation of CDS Options

Under the survival measure $\bar{Q}$ derived in Krekel et al. [2009] [9], for all maturities $T > 0$, we have that:

- The default-free term structure of zero-coupon bonds is given by:

$$P(0,T) = \mathrm{E}_{\bar{Q}}\left[\exp\left(-\int_0^T r(u)du\right)\right] \tag{2.37}$$

- The defaultable term structure of zero-coupon bonds is given by:

$$\bar{P}(0,T) = \mathrm{E}_{\bar{Q}}\left[\exp\left(-\int_0^T (r(u)+\lambda(u))du\right)\right] = \mathrm{E}_{\bar{Q}}\left[\exp\left(-\int_0^T \bar{r}(u)du\right)\right] \tag{2.38}$$

Now following the valuation of CDS options in Krekel et al. [2009], the value of the fixed leg of a forward CDS at time 0, which we denote by $V_{fixed}(0)$, is given by:

$$V_{fixed}(0) = \xi\sum_{i=k}^{n-1}\delta_i \mathrm{E}_{\bar{Q}}\left[\exp\left(-\int_0^{T_{i+1}} \bar{r}(u)du\right)\right] = \xi\sum_{i=k}^{n-1}\delta_i\bar{P}(0,T_{i+1}) \tag{2.39}$$

where the last equality follows from (2.38).

On the other hand, the value of the floating leg of a forward CDS at time 0, hereby denoted by $V_{floating}(0)$, is given by:

$$V_{floating}(0) = (1-R)\sum_{i=k}^{n-1}\mathrm{E}_{\bar{Q}}\left[\exp\left(-\int_0^{T_i}\bar{r}(u)du\right)\times\left(P(T_i,T_{i+1}) - \exp\left(-\int_{T_i}^{T_{i+1}}\bar{r}(u)du\right)\right)\right] \tag{2.40}$$

By linearity of expectation and by (2.38), we have,

$$V_{floating}(0) = (1-R)\sum_{i=k}^{n-1}\left\{\mathrm{E}_{\bar{Q}}\left[exp\left(-\int_0^{T_i}\bar{r}(u)du\right)P(T_i,T_{i+1})\right] - \bar{P}(0,T_{i+1})\right\} \tag{2.41}$$

Next, we denote by $\xi_{k,n}(0)$ the value for ξ such that the value of the floating leg is equal to the value of the fixed leg of a forward CDS, both at time 0, i.e.:

$$\xi_{k,n}(0) = (1-R)\frac{\sum_{i=k}^{n-1}[\mathrm{E}_{\bar{Q}}(\exp(-\int_0^{T_i}\bar{r}(u)du)P(T_i,T_{i+1})) - \bar{P}(0,T_{i+1})]}{\sum_{i=k}^{n-1}\delta_i\bar{P}(0,T_{i+1})} \tag{2.42}$$

usually called in the literature as the forward credit swap rate or forward spread. That is, is the credit swap rate between the expiry date of the option and the maturity date of the CDS contract, which in this notation are, respectively, represented by k and n.

[9] for future details about the change of measure, please see Krekel et al. [2009].

This definition can be easily extended for any time $t \leq T_k$, as follows:

$$\xi_{k,n}(t) = (1-R)\frac{\sum_{i=k}^{n-1}[\mathrm{E}_{\bar{Q}}(\exp(-\int_t^{T_i}\bar{r}(u)du)P(T_i,T_{i+1})) - \bar{P}(t,T_{i+1})]}{\sum_{i=k}^{n-1}\delta_i\bar{P}(t,T_{i+1})} \tag{2.43}$$

where in the previous definition we assume that the value of the defaultable short rate process is known until time t.

We note that equations (2.39) and (2.41) can be extended to an arbitrary time t, as follows:

$$V_{fixed}(t) = \xi\sum_{i=k}^{n-1}\delta_i\bar{P}(t,T_{i+1}) \tag{2.44}$$

$$V_{floating}(t) = \xi_{k,n}(t)\sum_{i=k}^{n-1}\delta_i\bar{P}(t,T_{i+1}) \tag{2.45}$$

assuming that all information regarding the defaultable interest rate is known until time t.

If we consider the default-free short rate process $\{r(t),t\}$ deterministic, the formula of $\xi_{k,n}(t)$ can be rewritten as:

$$\xi_{k,n}(t) = (1-R)\frac{\sum_{i=k}^{n-1}[S(t,T_i)P(t,T_{i+1}) - \bar{P}(t,T_{i+1})]}{\sum_{i=k}^{n-1}\delta_i\bar{P}(t,T_{i+1})} \tag{2.46}$$

Then, by (2.44) and (2.45), the payoff of the payer CDSwaption at the maturity T_k, hereby denoted by $V_{payer}(T_k)$, is given by:

$$V_{payer}(T_k) = (V_{floating}(T_k) - V_{fixed}(T_k))^+ = (\xi_{k,n}(T_k)-\xi)^+\sum_{i=k}^{n-1}\delta_i\bar{P}(T_k,T_{i+1}) \tag{2.47}$$

Conversely, the payoff of the receiver CDSwaption at maturity T_k is given by [10]

$$V_{receiver}(T_k) = (V_{fixed}(T_k) - V_{floating}(T_k))^+ = (\xi-\xi_{k,n}(T_k))^+\sum_{i=k}^{n-1}\delta_i\bar{P}(T_k,T_{i+1}) \tag{2.48}$$

Similarly as in Tankov and Touzi [2015], we define the intrinsic value of a payer (resp. receiver) CDS option as $(\xi_{k,n}(t)-\xi)^+$ (resp. $(\xi-\xi_{k,n}(t))^+$), that is, the value immediately received after exercising the option at a certain date t. In this context, for a payer (resp. receiver) CDS option, we have the following cases :

- If the intrinsic value is positive, $\xi_{k,n}(t) > \xi$ (resp. $\xi_{k,n}(t) < \xi$), we say that the option is *in-the-money* (ITM);
- If the intrinsic value is negative, $\xi_{k,n}(t) < \xi$ (resp. $\xi_{k,n}(t) > \xi$), we say that the option is *out-of-the-money* (OTM);
- If the option as intrinsic value equals to zero (i.e, $\xi_{k,n}(t) = \xi$ for payer and receiver CDS options) then we say that the option is *at-the-money* (ATM).

[10] For instance, if we consider $x, y \in \mathrm{R}$ then $(x-y)^+$ represents the maximum function, i.e, $(x-y)^+ = max(x-y,0)$.

So, for all option expiry dates, T_k say, and knowing that we are dealing with European options, we have the following cases:

- The holder of a payer(call) CDS option exercises at T_k only if $\xi_{k,n}(T_k) > \xi$;
- The holder of a receiver(put) CDS option exercise at T_k only if $\xi_{k,n}(T_k) < \xi$.

This means that, in both cases, it is only worth to exercise the CDS option at T_k when its intrinsic value is positive, i.e, when the option is ITM. Thereby, if the holder of the payer (resp. receiver) CDS option decides to exercise the option at the expiry date then he enters in a forward CDS contract at time T_k with maturity T_n as a protection buyer (resp. seller). Clearly, the expiry date of the CDS option coincides with the starting date of the forward CDS contract.

Therefore, the price of a payer CDSwaption at time 0 is given by,

$$V_{payer}(0) = \mathrm{E}_{\bar{Q}}\left[\exp\left(-\int_0^{T_k} \bar{r}(s)ds\right)\sum_{i=k}^{n-1} \delta_i \bar{P}(T_k, T_{i+1})(\xi_{k,n}(T_k) - \xi)^+\right] \tag{2.49}$$

The price of the payer CDSwaption at time 0, under the survival measure $\bar{Q}$, is the discounted expected payoff of the CDSswaption at expiry T_k where the discount factor is $\bar{r}(.)$.

2.4.2 Implied Volatility for CDS options

In this subsection we address the question on implied volatilities. In order to motivate the problem and associated definition, we first present the Black's formula, which is a standard result from any finance text book (for instance, Tankov and Touzi [2015]).

In the sequel, we let Φ denote the cumulative distribution function of a standard normal distribution, $\mathcal{N}(0,1)$. According to Black's formula , the price of a European call option when the discount factor is D, strike price is K, time to maturity is $T-t$, forward price of the underlying asset is F and its volatility is σ, is given by:

$$C(F, T-t) = D \times Black(F, K, \sigma, T-t) \tag{2.50}$$

where

$$Black(F, K, \sigma, T-t) = F\Phi(d_1) - K\Phi(d_2) \tag{2.51}$$

with

$$\begin{aligned} d_1 &= \frac{\ln(\frac{F}{K}) + \frac{1}{2}\sigma^2(T-t)}{\sigma\sqrt{T-t}} \\ d_2 &= d_1 - \sigma\sqrt{T-t} \end{aligned}$$

We note that in the previous formula almost all the information and data is observed or provided by the market, except the volatility. So if we could just simply invert the formula (2.50) in order to derive σ, then the process would be trivial. But that is not the case, and we need to use numerical methods. In fact the term implied volatility means that is the volatility calibrated from the option price observed in the market (Hull [2012]).

Latter, still in this subsection, we will comment on numerical methods used in order to compute the implied volatility. But for the time being, we return to the question of the standard market formula for payer CDS options at the valuation date. Although we have been using in the above sections the value 0 to denote the valuation time, for now onwards we use preferentially the value t_0 [11], with the obvious changes in the formulas. The reason for this change will become clear when we present the results, in Chapter 4.

According to Brigo and Morini [2005], under some technical conditions, the standard market formula for payer CDS options with maturity T_k at the valuation date t_0, is given by:

$$CDSOptionBS = Black(\xi_{k,n}(t_0), \xi, \sigma, T_k - t_0) \sum_{i=k}^{n-1} \delta_i \bar{P}(t_0, T_{i+1}) \tag{2.52}$$

where the forward spread $\xi_{k,n}(t_0)$ is calculated according (2.42).

In this context, the discount factor, $D = \sum_{i=k}^{n-1} \delta_i \bar{P}(t_0, T_{i+1})$, is called the forward risky level or forward duration and is obtained by dividing the value of the fixed leg at time t_0 ($V_{fixed}(t_0)$) by the strike price of the option (ξ) where the first term is given by (2.39).

As previously stated, we need to use numerical approaches to derive estimates for the implied volatility according to the previous equations. In this thesis we consider two numerical methods, the bisection method and the Newton's method. In appendix A.1 we present the codes developed in the software R that were used in the numerical illustration that we will present later, for the Newton's and for the bisection methods.

Before ending this Chapter, we make the following remark: one of the goals of our thesis is to compute the prices at the valuation date of CDS options in a recombining tree, based on a default intensity model described in the previous Chapter. For that we will use a numerical method that will be described in the following Chapter. But then it means that in order to find the implied volatility, we will not use market quotes of (payer) CDS options, but instead we will use the price at the valuation date obtained from the algorithm described later on.

[11] We remind that the time to maturity is calculated using the Actual/365 convention as described at the beginning of this Chapter.

Chapter 3

Numerical Procedure for CDS Options

In this Chapter, we accomplished our main goal: write a pricer, in a recombining tree, for a CDS option, using a Cheyette model (i.e, based in the default intensity model described in the previous Chapter in section 2.3).

With this in view, we derive an adaptation of the numerical method derived in Li et al. [1995], for the particular case of CDS options. In particular, in section 3.1, we derive the relevant results and build the structure (i.e, the recombining tree) that will allow us to compute the CDS option price at the valuation date. Next, in section 3.2, we explain how did we implemented the numerical method, described in the previous section, and we also mention the main simplifications that we have used in our algorithm. Lastly, we present a (short and simple) illustrative example.

3.1 Pricer in a Recombining Tree

In order to construct the lattice used to compute the approximation of the two-state variable process (2.30, 2.31) described in the previous Chapter, we adapt the method described in Li et al. [1995], using, in our case, the default intensity $\lambda(t)$, instead of the short rate $r(t)$ process (used in the original paper).

Therefore, we transform the default intensity process (2.30) in a process that has a constant volatility parameter, and we build a recombining tree for this new process. For non-arbitrage arguments, we need to consider values of the elasticity parameter (γ, with $\gamma > 0$) such that the default intensity is always positive.

From the default intensity process $\{\lambda(t), t\}$ we consider the following transformation,

$$Y(t, \lambda(t)) = \int_{\lambda_0}^{\lambda(t)} \frac{1}{\sigma l^{\gamma}} dl \tag{3.1}$$

where σ and λ_0 (such that, $\lambda_0 > 0$) are constants.

Then, we apply the Itô's lemma to (3.1) and we obtain the following differential equation,

$$dY(t, \lambda(t)) = \Big[\frac{\partial Y}{\partial t} + \mu(t, \lambda(t), \phi_\lambda(t))\frac{\partial Y}{\partial \lambda} + \frac{1}{2}\sigma_\lambda^2(t)\frac{\partial^2 Y}{\partial \lambda^2}\Big]dt + \frac{\partial Y}{\partial \lambda}\sigma_\lambda(t)dW(t) \tag{3.2}$$

where the drift term of the default intensity process, $\mu(t, \lambda(t), \phi_\lambda(t))$, is given by (2.32). For simplicity of notation, we denote the drift term of the transformed process as follows,

$$m(Y, \phi_\lambda, t) = \frac{\partial Y}{\partial t} + \mu(t, \lambda(t), \phi_\lambda(t))\frac{\partial Y}{\partial \lambda} + \frac{1}{2}\sigma_\lambda^2(t)\frac{\partial^2 Y}{\partial \lambda^2} \tag{3.3}$$

Trivially, we have that the relationship (3.1) is equivalent to,

$$Y(t, \lambda(t)) = \frac{[\lambda(t)]^{1-\gamma}}{\sigma(1-\gamma)} - \frac{\lambda_0^{1-\gamma}}{\sigma(1-\gamma)} \tag{3.4}$$

So, in particular, the partial terms of (3.2) are equal to,

$$\frac{\partial Y}{\partial t} = 0 \quad , \quad \frac{\partial Y}{\partial \lambda} = \frac{1}{\sigma_\lambda(t)} \quad \text{and} \quad \frac{\partial^2 Y}{\partial \lambda^2} = -\frac{\sigma\gamma[\lambda(t)]^{\gamma-1}}{\sigma_\lambda^2(t)}$$

Substituting these partial terms on (3.2), we have that,

$$dY(t, \lambda(t)) = m(Y, \phi_\lambda, t)dt + dW(t) \tag{3.5}$$

where the drift term of the transformed process Y, (3.3), can be written as follows,

$$m(Y, \phi_\lambda, t) = \frac{\mu(t, \lambda(t), \phi_\lambda(t))}{\sigma_\lambda(t)} - \frac{\sigma\gamma\lambda(t)^{\gamma-1}}{2} = \frac{\mu(t, \lambda(t), \phi_\lambda(t))}{\sigma[\lambda(t)]^\gamma} - \frac{\gamma\sigma[\lambda(t)]^\gamma}{2\lambda(t)} \tag{3.6}$$

Therefore, from now on, instead of working with the two-state variable process (2.30) and (2.31) we consider the following,

$$\begin{aligned} dY(t, \lambda(t)) &= m(Y, \phi_\lambda, t)dt + dW(t) \\ d\phi_\lambda(t) &= (\sigma_\lambda^2(t) - 2k_\lambda\phi_\lambda(t))dt \end{aligned} \tag{3.7}$$

where the drift term of Y is given by (3.6) and $\sigma_\lambda^2(t) = \sigma^2[\lambda(t)]^{2\gamma}$ from (2.36).

As previously stated, we can only guarantee non-arbitrage when $\gamma = 1$ (Krekel et al. [2009]); in this case the resulting model is usually called proportional model (Li et al. [1995]). Therefore, the volatility of the default intensity process (2.36) is written as follows,

$$\sigma_\lambda(t) = \sigma\lambda(t) \tag{3.8}$$

Then it follows, by straightforward calculations, that the process Y is given by,

$$Y(t, \lambda(t)) = \frac{1}{\sigma}\int_{\lambda_0}^{\lambda(t)} \frac{1}{l}dl \approx \frac{ln(\lambda(t))}{\sigma} \tag{3.9}$$

Conversely, the default intensity can be written as,

$$\lambda(t) = e^{\sigma Y(t)} \tag{3.10}$$

Therefore, in this case, we have the following processes,

$$\begin{aligned} dY(t,\lambda(t)) &= m(Y,\phi_\lambda,t)dt + dW(t) && (3.11)\\ d\phi_\lambda(t) &= (\sigma^2\lambda(t)^2 - 2k_\lambda\phi_\lambda(t))dt && (3.12)\end{aligned}$$

where, considering the formulas (3.6) and (2.32), the drift term of the transformed process can be written as,

$$m(Y,\phi_\lambda,t) = \frac{1}{\sigma}\left\{\frac{k_\lambda[s(0,t)-\lambda(t)] + \phi_\lambda(t) + \frac{d}{dt}s(0,t)}{\lambda(t)}\right\} - \frac{\sigma}{2} \qquad (3.13)$$

From now on, all the calculations and implementations will be for the proportional model.

3.1.1 Lattice Approximation

Considering the previous results, for the case of the proportional model ($\gamma = 1$), we are now able to build the lattice approximation of the processes (3.11) and (3.12) for pricing a CDS option at the valuation date.

Similarly to Li et al. [1995] approach, we start by partitioning the interval between the valuation date and the expiry date into subintervals of equal length, Δt, and we let N denote the number of such subintervals, and also the number of (time) steps considered in the recombining tree, that we will be describe next.

Before explaining in detail the algorithm, we explain briefly the structure of the tree. The tree is a recombining one, and we need two indexes to describe the state of the tree, namely: the first index (i, say) is related with the number of time steps since the valuation date until the expiry date of the CDS option, so $i \in \{0, 1, \ldots, N\}$. The second index ($j$, say) is the number of nodes of the tree at each time step, where $j \in \{0, 1, \ldots, i\}$. As the tree is a recombining one, at time step i we have $(i + 1)$ nodes, and therefore at the expiry date ($i = N$) we have $N + 1$ nodes. So, for instance, (i, j) represents the jth node of the ith time step of the tree.

Following the terminology of Li et al. [1995], we call the values of the join process (Y, ϕ_λ) of the tree as the *approximation variables*, and we use the following notation: if at a certain node ((i, j), say) we assume that the values of the join process are (y^a, ϕ^a_λ), then in the next time step we will have the values $(y^{a+}, \phi^{a+}_\lambda)$ and $(y^{a-}, \phi^{a-}_\lambda)$ (respectively at nodes $(i + 1, j)$ and $(i + 1, j + 1)$) where, y^{a+} and y^{a-} are, respectively, given by,

$$y^{a+} = y^a + (J + 1)\sqrt{\Delta t} \quad \text{and} \quad y^{a-} = y^a + (J - 1)\sqrt{\Delta t} \qquad (3.14)$$

such that,

$$y^{a-} \leq y^a + m(y^a, \phi^a_\lambda, t)\Delta t \leq y^{a+} \qquad (3.15)$$

and J is an integer defined as follows,

$$J = \begin{cases} |Z|, & \text{if } Z \text{ is even} \\ Z+1, & \text{otherwise} \end{cases} \tag{3.16}$$

with

$$Z = \left\lfloor m(y^a, \phi_\lambda^a, t)\sqrt{\Delta t} \right\rfloor \tag{3.17}$$

Note that trivial calculations show that:

$$m(y^a, \phi_\lambda^a, t)\sqrt{\Delta t} - 1 \leq J \leq m(y^a, \phi_\lambda^a, t)\sqrt{\Delta t} + 1 \tag{3.18}$$

where $m(y^a, \phi_\lambda^a, t)$ is given by (3.13).

Regarding ϕ_λ^a, from (3.12) one can see that $\{\phi_\lambda(t), t\}$ is a deterministic process, because only depends on t, then the values ϕ_λ^{a+} and ϕ_λ^{a-} are equal. So, we denote ϕ_λ^{a*} as their common value, which can be obtained by integrating the process (3.12) until time Δt,

$$\phi_\lambda^{a*} = \phi_\lambda^a + [(\sigma\lambda^a)^2 - 2k_\lambda\phi_\lambda^a]\Delta t \tag{3.19}$$

where $\lambda^a = e^{\sigma y^a}$ from (3.8).

We can easily see that the total number of distinct values of ϕ_λ^a at each node will be equal to the total number of unique paths that reach that node. Following the simplification proposed by Li et al. [1995], we chose to keep at each node the interval $[\underline{\phi}_\lambda^a, \bar{\phi}_\lambda^a]$ where $\underline{\phi}_\lambda^a$ (resp. $\bar{\phi}_\lambda^a$) is the minimum (resp. maximum) value of ϕ_λ^a from the valuation date until the respective node.

Therefore, we note that at the beginning (at node $(0,0)$), by construction, the value of this state variable is zero and at the edge nodes [1] of the tree we only have one (unique) value of ϕ_λ^a because, we can only reach these nodes from their preceding node. Otherwise, considering the middle nodes, that is, a node that can be generated from two distinct nodes at the previous time step, we need to calculate the corresponding intervals of the form $[\underline{\phi}_\lambda^a, \bar{\phi}_\lambda^a]$. The way to derive these lower and upper bounds is the following:

We assume that at a certain node ((i,j), say) we have the interval $[\underline{\phi}_\lambda^a, \bar{\phi}_\lambda^a]$. So, we let $\mathtt{m}$ be the number of elements in this interval and then we partition $[\underline{\phi}_\lambda^a, \bar{\phi}_\lambda^a]$ into subintervals of equal length $\Delta\phi$, i.e,

$$\underline{\phi}_\lambda^a = \phi_\lambda^a(1) < \phi_\lambda^a(2) < ... < \phi_\lambda^a(\mathtt{m}) = \bar{\phi}_\lambda^a$$

where, $\phi_\lambda^a(u), u = 1, ..., \mathtt{m}$ denotes the uth point of the partition and the partition step is given by
$\Delta\phi = (\phi_\lambda^a(\mathtt{m}) - \phi^a(1))/(\mathtt{m} - 1)$.

Then for each value of this partition ($\phi_\lambda^a(u), u = 1, ..., \mathtt{m}$) we calculate the respective successor value ($\phi_\lambda^{a*}(u), u = 1, ..., \mathtt{m}$) and we proceed in the same way for all the middle nodes at the ith step. Since

[1] Clearly, at a certain time step i, when $j = 0$ or $j = i$ we denote these nodes as edge nodes.

each one of the middle nodes at the $(i+1)$th step can be obtained from two nodes at the previous step, we need to compare the values of ϕ_λ^{a*} and select the, respective, minimum and maximum value for the corresponding interval of the accumulated variance in each one of the middle node at $(i+1)$th step. In particular, if we consider the nodes, (i,j) and $(i,j+1)$ then, for each node, we partition the interval of the form $[\underline{\phi}_\lambda^a, \bar{\phi}_\lambda^a]$ into subintervals of equal length and calculate the corresponding next values $(\phi_\lambda^{a*}(u), u = 1, ..., \mathtt{m})$. So since node $(i+1, j+1)$ can be reached from these two preceding nodes, we need to select the lower and upper bounds, respectively, the minimum and maximum values, from the values of ϕ_λ^{a*} which were obtained from the nodes (i,j) and $(i,j+1)$.

The tree that we are building is a binomial tree, whose nodes we have already described. So now we need to describe the probability structure of the tree, i.e, we need to compute the probabilities amongst the nodes. We let $p = p(y^a, \phi_\lambda^a)$ be the probability of moving from y^a to y^{a+} in the next time increment., where p depends on the value of the approximate variables as follows:

$$p(y^{a+} - y^a) + (1-p)(y^{a-} - y^a) = m(y^a, \phi_\lambda, t)\Delta t \tag{3.20}$$

This means that the expected value of the magnitude of the jumps between the two nodes (which is equal to $(y^{a+} - y^a)$ in case of an up, and $(y^{a-} - y^a)$), in case of a down) is equal to the drift parameter of the transformed process $Y(t, \lambda(t))$ times the time increment.

Equivalently, we can write the probability of occurring an up movement (3.20) as follows,

$$p = \frac{m(y^a, \phi_\lambda^a, t)\Delta t + (y^a - y^{a-})}{(y^{a+} - y^{a-})} \tag{3.21}$$

We note that the particular choice of J is such that p in (3.21) is indeed a probability.

Moreover, from (3.14), we have that,

$$y^{a+} - y^{a-} = 2\sqrt{\Delta t} \quad \text{and} \quad y^a - y^{a-} = (1-J)\sqrt{\Delta t}$$

Then, the probability (3.21) can be rewritten in function of the integer J,

$$p = \frac{m(y^a, \phi_\lambda^a, t)\Delta t + (1-J)\sqrt{\Delta t}}{2\sqrt{\Delta t}} \tag{3.22}$$

We note that on the implementation of the method we will use this formula for calculating this probability.

3.1.2 Backward Recursion and Interpolation

After building the lattice approximation or the recombining tree we present a general procedure to price an option where we use a general backward recursion approach and interpolation to calculate the option price at the valuation date.

We let, $g_i(y^a, \phi_\lambda^a)$ be the option value at the i^{th} step of the recombining tree conditional on the state variables (y^a, ϕ_λ^a). Also, we assume that all the option values at $(i+1)^{th}$ step are available, then the option prices at i^{th} step are as follows,

$$g_i(y^a, \phi_\lambda^a) = [pg_{i+1}(y^{a+}, \phi_\lambda^{a*}) + (1-p)g_{i+1}(y^{a-}, \phi_\lambda^{a*})]e^{-\bar{r}_i^a \Delta t} \tag{3.23}$$

where the discount factor follows from (2.49) for the case of a payer CDS option.

However, the value of the accumulated variance at the $(i+1)^{th}$ (ϕ_λ^{a*}) is not always available, because we have chosen to only keep the minimum and the maximum values of the successor values of ϕ_λ^a, that is, it is completely determined by the current values of (y^a, ϕ_λ^a). Therefore, in the case that we do not have the exact values of $g_{i+1}(y^{a+}, \phi_\lambda^a)$ or $g_{i+1}(y^{a-}, \phi_\lambda^a)$, we apply linear interpolation in order to determine these option prices. In particular, if ϕ_λ^{a*} at the up node does not exist then, the option price at (y^{a+}, ϕ^{a*}) is given by,

$$g_{i+1}(y^{a+}, \phi_\lambda^{a*}) = g_{i+1}(y^{a+}, \phi_-^{a*}) + \Big(\frac{\phi^{a*} - \phi_-^{a*}}{\phi_+^{a*} - \phi_-^{a*}}\Big)(g_{i+1}(y^{a+}, \phi_+^{a*}) - g_{i+1}(y^{a+}, \phi_-^{a*})) \tag{3.24}$$

On the other hand, if ϕ_λ^{a*} at the down node does not exist then, the option price at (y^{a-}, ϕ^{a*}) is given by,

$$g_{i+1}(y^{a-}, \phi_\lambda^{a*}) = g_{i+1}(y^{a-}, \phi_-^{a*}) + \Big(\frac{\phi^{a*} - \phi_-^{a*}}{\phi_+^{a*} - \phi_-^{a*}}\Big)(g_{i+1}(y^{a-}, \phi_+^{a*}) - g_{i+1}(y^{a-}, \phi_-^{a*})) \tag{3.25}$$

Then, we apply the formula (3.23) in order to obtain the option price at i^{th} step [2].

3.2 Cheyette Algorithm

The procedure explained at the last section, and whose formulas are displayed in section 3.1, is called the Cheyette algorithm, that we have implemented, using the programming language C++. In this section we explain in more detail the implementation issues. The method will be derived for the particular case of a CDS option on a underlying forward CDS contract. Along the rest of the thesis, we assume the case of the proportional model ($\gamma = 1$) for the reasons previously mentioned. Furthermore we assume that the initial term structure of the forward default intensity $s(0,t)$ is flat initially, in the sense that $s(0,t) = \lambda_0$, and we assume that the default-free interest rates are deterministic, i.e, $r(t) = r_0 \; \forall t$.

As (input) model parameters we consider the following ones[3]

- `vol`: volatility (σ);
- `ks`: a constant called mean reversion term (k_λ);
- `m`: (integer) number of elements in the partition of the interval of the accumulated variance $[\underline{\phi}_\lambda^a, \bar{\phi}_\lambda^a]$;
- `N`: (integer) number of steps of the recombining tree;
- `strike`: strike price of the CDS option (ξ).

[2] Note that by the way that the lattice approximation was built, the values at nodes the successor nodes (y^{a+}, ϕ_+^{a*}) and (y^{a+}, ϕ_-^{a*}) will be available, where, $\phi_-^{a*} \leq \phi^{a*} \leq \phi_+^{a*}$ Li et al. [10].

[3] We use the same name for the variables as the ones that we have used in the code, provided at the end of the text.

- `R`: recovery rate of the CDS option (R);
- `dateT`: expiry date of the CDS option (T_k);
- `typeoption`: type of the CDS option, that is, there are only allowed the characters 'C' and 'P' representing, respectively, a payer (call) or receiver (put) CDS option;
- `r0`: (deterministic) default-free interest rate (r_0);
- `lambda0`: initial value of the default intensity (λ_0);
- `phi0`: initial value of the accumulated variance which, by construction, is such that $\phi_\lambda(0) = 0$;
- *Dates.txt*, text file containing the following information:
  1. `dates`: contains the valuation and expiry date of the CDS option, the payment dates and the maturity date of the underlying forward CDS (in the format day-month-year);
  2. `numdate`: `dates` in numerical format.

On the other hand, we consider the CDS option price at the valuation date as the output of our algorithm.

Moreover, we have chosen that each node of the recombining tree is characterized by the following variables:

1. `lambda`: default intensity (λ);
2. `phi`: interval $[\underline{\phi}^a_\lambda, \bar{\phi}^a_\lambda]$;
3. `probUp`: probabilities of an up movement (3.22);
4. `optionPrice`: CDS option prices (3.23);
5. `visit`: takes the value 0 if the node wasn't visit yet and 1 otherwise.

Also, as auxiliary functions we define the following function:

- `Z`: integer of equation (3.17);
- `calcJ`: integer J calculated according the formula (3.16).

We recall that we have denoted i as the i^{th} step of the recombining tree, with $i = 0, 1, \ldots, N$, where N is the number of steps of the tree. And also, j as the j^{th} node of the i^{th} step, where $j = 0, 1, \ldots, i$. Clearly, since we are constructing a recombining tree and at each node we only have up or down movements with respect to a probability, at the i^{th} step we have $i + 1$ nodes.

So, we divide the construction of the Cheyette algorithm in three parts, namely,

- Lattice approximation (where we construct the tree, for the whole steps and nodes, i.e, for $i = 0, 1, \ldots, N - 1$ and $j = 0, 1, \ldots, i$);

- Terminal nodes (where, as the name suggest, we compute for the last step of the tree, $i = N$, for the whole nodes, $j = 0, 1, \dots, N$);
- Backward recursion and interpolation (once computed the value for the terminal nodes, we proceed backwards, or the steps $i = N-1, N-2, \dots, 0$ and for the respective nodes $j = 0, 1, \dots, i$).

We let T_k be the expiry date and t_0 be the valuation date of the CDS option. Then, we start by computing the time to expiry (in years) of this option at the valuation date, [4], as follows,

$$\tau^* = \frac{T_k - t_0}{365} \tag{3.26}$$

So, having our interval of interest, we now divide the time to expiry τ^*, in intervals of equal length, that is,

$$\Delta t = \frac{\tau^*}{N} \tag{3.27}$$

where N corresponds to the number of steps of the recombining tree.

We next explain in more detail each one of the three parts of the Cheyette algorithm.

3.2.1 Lattice Approximation

If we consider the (i, j) node (recall that we have defined already this notation, and that ith denotes the step and j the corresponding node) and given that the value $\mathtt{m}$ is the number of subintervals of the partition of $[\underline{\phi}^a_\lambda, \bar{\phi}^a_\lambda]$, we implement the lattice approximation in the following way:

1. Regarding the initial conditions that we have taken, in particular, that $\gamma = 1$ and $s(0,t) = \lambda_0$ (i.e, $\dfrac{d}{dt}s(0,t) = 0$), we can rewrite the equation (3.13) as follows:

$$m(y^a, \phi^a_\lambda(u), t) = \frac{k_\lambda[\lambda_0 - \lambda^a] + \phi^a_\lambda(u)}{\sigma\lambda^a} - \frac{\sigma}{2} \tag{3.28}$$

   where $u = 1, 2, \dots, \mathtt{m}$ and λ^a denotes the default intensity at node (i, j).

2. For each value of $\phi^a_\lambda(u)$, we compute the value of the drift term of the transformed process, according to (3.28), and then the value of the integer J according to (3.16). So, we are now able to calculate the probability p of an up movement according to (3.22) [5].

   Taking into account that most of the time the value of J is equal to zero, we decide to send an error message and stop the procedure, when J is such that $J \neq 0$, to avoid larger jumps in the value of the default intensity, .

3. Next, given that we know the value of the default intensity (λ^a) at node (i, j), then the values at the nodes $(i+1, j)$ and $(i+1, j+1)$ are computed, respectively, as follows,

$$\lambda^{a+} = exp\Big\{\lambda^a + (J+1)\sqrt{\Delta t}\Big\} \text{ and } \lambda^{a-} = exp\Big\{\lambda^a + (J-1)\sqrt{\Delta t}\Big\} \tag{3.29}$$

[4] In order to calculate the time to expiry, it is necessary to consider the numerical format of these two dates.

[5] We note that, on the edge nodes of the tree we only have one (distinct) value for ϕ^a_λ, because $\underline{\phi}^a_\lambda = \bar{\phi}^a_\lambda$.

where these formulas follow from equations (3.10) and (3.14).

We note that, since this is a recombining tree, we always verify if the nodes $(i+1,j)$ and $(i+1,j+1)$ have not been visited yet (i.e, `visit=0`). Then, in case they have not been visited yet, we insert the (next) corresponding value of the default intensity, otherwise we do nothing.

4. Finally, given that we have at node (i,j) the interval $[\underline{\phi}^a_\lambda, \bar{\phi}^a_\lambda]$, we calculate the corresponding intervals at nodes $(i+1,j)$ and $(i+1,j+1)$, according to the procedure described previously.

3.2.2 Terminal nodes

After building the lattice approximation as described in the previous item, we are now able to calculate the CDS option price at the expiry date (that is, at the terminal nodes of the tree).

At the terminal nodes we have all the information that we need to calculate the CDS option price. In particular, we proceed as follows:

1. Firstly, we compute the intervals of time between payments of the forward CDS:

$$\delta_{i^*} = \frac{T_{i^*+1} - T_{i^*}}{365}, \ \forall i^* = k, k+1, ..., n-1 \tag{3.30}$$

where the times $T_{k+1}, T_{k+2}, ..., T_n$ are the payment dates of the forward CDS contract;

2. Since we consider that the default-free interest rates are deterministic (r_0), then the zero-coupon bond price at expiry time T_k, for each date of the forward CDS, is calculated as follows,

$$P(T_k, T_{i^*+1}) = \exp\left(-r_0 \frac{(T_{i^*+1} - T_k)}{365}\right), \ \forall i^* = k, k+1, ..., n-1 \tag{3.31}$$

3. For each value of $\phi^a_\lambda(u)$ with $u = 1, \ldots, \mathtt{m}$, we compute, respectively, the forward survival probability at time T_k for each date of the forward CDS (i.e, $S(T_k, T_{i^*+1})$, $\forall i^* = k, k+1, \ldots, n-1$) according to formula (2.34) and considering that at the valuation date the forward survival probability is calculated as follows,

$$S(t_0, T_{i^*}) = \exp\left(-\lambda_0 \frac{(T_{i^*} - t_0)}{365}\right), \ \forall i^* = k, k+1, ..., n-1, n \tag{3.32}$$

Also, we calculate the defaultable zero-coupon bond price at time T_k for each date of the forward CDS (i.e, $\bar{P}(T_k, T_{i^*+1})$, $\forall i^* = k, k+1, \ldots, n-1$), according to equation (2.17). In particular,

$$\bar{P}(T_k, T_{i^*+1}) = S(T_k, T_{i^*+1}) \times P(T_k, T_{i^*+1}), \ \forall i^* = k, k+1, ..., n-1 \tag{3.33}$$

4. Next, we calculate the value of the forward credit swap rate at time T_k ($\xi_{k,n}(T_k)$) according to equation (2.46), because here the default-free short rates are deterministic.

Therefore, after calculating these quantities, we can now calculate the payoff at the terminal nodes of the payer (resp. receiver) CDS option according to formula 2.47 (resp. 2.48).

3.2.3 Backward Recursion and Interpolation

After computing the CDS option prices at the expiry date, we use backward recursion and linear interpolation from $i = N - 1$ to $i = 0$, to calculate the option prices in each one of the nodes of the tree. In order to do that, we proceed as described in the previous section 3.1.

3.2.4 Example

In order to illustrate the Cheyette algorithm, we consider a payer CDS option with the following dates:

- Option valuation date: 17-10-2008;
- Option expiry date: 20-12-2008;
- Maturity date of the underlying CDS: 20-12-2013.

Also we consider the following input values:

σ	k_λ	ϕ_0	λ_0	r_0	R	ξ(bp)	m	N
0.94	-0.10	0.0	0.02	0.01	0.4	100	3	3

Table 3.1: Cheyette parameters (for example in Chapter 3).

Concerning the set of parameters represented in table 3.1, the mean reversion and the volatility parameters were chosen to be equal to the optimal parameters obtained in Krekel et al. [2009] for an iTRAXX S10 IG five year CDS.

We start by partitioning the time to expiry ($\tau^* = 0.1753425$ years, which is approximately 64 days), in subintervals of equal length ($\Delta t = 0.0584475$). Then, we perform the lattice approximation as described previously and we obtain the values of the default intensity and the intervals of the accumulated variance shown in table 3.2 [6].

$\lambda_{00} = 0.02$ $\phi_{00} = 0$			
$\lambda_{10} = 0.025103$ $\phi_{10} = 2.065577\text{e}{-5}$	$\lambda_{11} = 0.0159344$ $\phi_{11} = 2.065577\text{e}{-5}$		
$\lambda_{20} = 0.0315079$ $\phi_{20} = 5.34432\text{e}{-5}$	$\lambda_{21} = 0.02$ $\phi_{21min} = 3.40118\text{e}{-5}$ $\phi_{21max} = 5.34432\text{e}{-5}$	$\lambda_{22} = 0.0126952$ $\phi_{22} = 3.40118\text{e}{-5}$	
$\lambda_{30} = 0.0395471$ $\phi_{30} = 0.000105338$	$\lambda_{31} = 0.025103$ $\phi_{31min} = 5.50671\text{e}{-5}$ $\phi_{31max} = 0.000105338$	$\lambda_{32} = 0.0159344$ $\phi_{32min} = 4.27328\text{e}{-5}$ $\phi_{32max} = 7.47256\text{e}{-5}$	$\lambda_{33} = 0.0101145$ $\phi_{33} = 4.27328\text{e}{-5}$

Table 3.2: Default intensities and intervals of the accumulated variance, at each step of the Cheyette algorithm.

Next, we calculate at the terminal nodes the payoff of this payer CDS option, using, in this case, the information in table 3.4 and following the procedure described previously. Then, we use backward recursion and linear interpolation to calculate the price of this option, at the valuation date.

[6] In this example, we denote the default intensity at node (i, j) as λ_{ij} (instead of λ^a) and the interval of the accumulated variance as $[\phi_{ijmin}, \phi_{ijmax}]$ (instead of $[\underline{\phi}^a, \bar{\phi}^a]$).

Finally, we obtain that the price of this option at the valuation date is 158.486 (bp) [7], as we can see in table 3.3. Also, we note that, in this case, at the valuation date, the forward risky level (D) is 4.60374 and the forward credit swap rate ($\xi_{k,n}(t_0)$) is 120.301 (bp).

$price_{00} = 158.486$			
$price_{10} = 289.994$	$price_{11} = 54.3127$		
$price_{20} = 135.501$	$price_{21}(1) = 123.605$ $price_{21}(2) = 124.099$ $price_{21}(3) = 124.593$	$price_{22} = 0$	
$price_{30} = 760.793$	$price_{31}(1) = 279.252$ $price_{31}(2) = 282.005$ $price_{31}(3) = 284.756$	$price_{32}(1) = 0$ $price_{32}(2) = 0$ $price_{32}(3) = 0$	$price_{33} = 0$

Table 3.3: Payer CDS option prices (in bp) at each step of the Cheyette algorithm.

Dates	Numerical Dates	$P(0,T)$	$S(0,T)$	$\bar{P}(0,T)$	δ
17-10-2008	39738	1.00000	1.00000	1.00000	——-
20-12-2008	39802	0.99825	0.99650	0.99475	0.17534
20-03-2009	39892	0.99579	0.99160	0.98742	0.24658
20-06-2009	39984	0.99328	0.98661	0.97998	0.25206
20-09-2009	40076	0.99078	0.98165	0.97260	0.25206
20-12-2009	40167	0.98832	0.97677	0.96535	0.24932
20-03-2010	40257	0.98588	0.97196	0.95824	0.24658
20-06-2010	40349	0.98340	0.96708	0.95102	0.25206
20-09-2010	40441	0.98092	0.96221	0.94386	0.25206
20-12-2010	40532	0.97848	0.95743	0.93682	0.24932
20-03-2011	40622	0.97607	0.95272	0.92992	0.24658
20-06-2011	40714	0.97362	0.94793	0.92291	0.25206
20-09-2011	40806	0.97116	0.94316	0.91596	0.25206
20-12-2011	40897	0.96875	0.93847	0.90914	0.24932
20-03-2012	40988	0.96633	0.93380	0.90236	0.24932
20-06-2012	41080	0.96390	0.92910	0.89556	0.25206
20-09-2012	41172	0.96147	0.92443	0.88882	0.25206
20-12-2012	41263	0.95908	0.91983	0.88220	0.24932
20-03-2013	41353	0.95672	0.91531	0.87569	0.24658
20-06-2013	41445	0.95431	0.91071	0.86910	0.25206
20-09-2013	41537	0.95191	0.90613	0.86255	0.25206
20-12-2013	41628	0.94954	0.90162	0.85612	0.24932

Table 3.4: Default-free and defaultable zero coupon bonds (respectively, $P(0,T)$ and $\bar{P}(0,T)$)) and survival probabilities ($S(0,T)$), at the valuation date 17-10-2008 with $\lambda_0 = 2\%$.

[7] 1% = 100 (bp) where bp denotes basis points.

Chapter 4

Results

Regarding the Cheyette algorithm explained previously, in this Chapter, our goal is to study/analyse, in detail, how a change in the input parameters (or also called Cheyette parameters) impacts the computation of the price and calibration of the implied volatility of a payer CDS option. We chose only to perform this analysis for payer CDS options because the case of receiver CDS options will produce the obvious changes. Therefore it will not add any significant information to this study.

In order to perform this sensitivity study, we present three examples that show different numerical features/properties of this algorithm. In the following examples we analyse,

- The evolution of the payer CDS option price with the number of steps of the tree (N), for different number of partitions ($\mathtt{m}$) of the accumulated variance (ϕ) interval;
- The convergence and the calibration of the implied volatility using two different root-finding methods, namely, the bisection and Newton's method;
- The effect of a change of the mean reversion parameter (k_λ) and also of the volatility parameter (σ) in the computation of the price and implied volatility of a payer CDS option, as a function of the strike price (ξ);
- The implied volatility for several values of the initial default intensity (λ_0), as a function of the strike price (ξ);
- For an *at-the-money* (ATM) payer CDS option, how a change in the volatility parameter (σ) impacts the implied volatility, for several values of the mean reversion parameter (k_λ).

Furthermore, we consider payer CDS options from 17-10-2008, whose expiry date can be either 20-12-2008 or 20-03-2009, on an underlying five year CDS contract with maturity date 20-12-2013 and quarterly payment dates [1]. In the following examples, we start by choosing k_λ and σ, such that, they have the same values as the ones obtained in Krekel et al. [2009] for the optimal values of these parameters, for an iTRAXX S10 IG five year CDS or for an iTRAXX S10 Xover five year CDS. Moreover, depending on the choice of the values of k_λ and σ, we consider different values for the strike prices (ξ). In particular,

[1]These dates are the same used in Krekel et al. [2009].

when k_λ and σ are equal to the optimal parameters for an iTRAXX S10 IG five year CDS, the values, in basis points (bp), of the strike price (ξ) are: 80, 90, 100, 110, 120, 130, 140, 150, 160 and 170. On the other hand, when k_λ and σ are equal to the optimal parameters for an iTRAXX S10 Xover five year CDS, the values, in basis points (bp), of the strike price (ξ) are: 300, 325, 350, 375, 400, 425, 450, 475, 500, 525, 550, 575, 600 and 625.

Before moving forward to the examples, we make two important remarks:

1. Here, we always try to compute the payer CDS option price, at least, one time per day, from its valuation date until its expiry date, taking into account the computational time and the numerical stability of the option price;

2. We only show the effect of k_λ and σ on the pricing for the example 4.1, because for the other examples the evolution of the price has the same behaviour.

4.1 Example 1

In this example, we consider a payer CDS option for a five year CDS, with the following dates:

- Option valuation date: 17-10-2008;
- Option expiry date: 20-12-2008;
- Maturity date of the underlying CDS: 20-12-2013.

We start by considering that the value of the initial default intensity (λ_0) is equal, say, to 2%. In this case, when $\lambda_0 = 2\%$, we use the information in table 3.4 to compute the payer CDS option prices. Then at the valuation date, we have the following values [2]:

- Forward risky level: $D = 4.60374$;
- Forward credit swap rate: $\xi_{k,n}(t_0) = 120.301$ (bp);
- Time to expiry: $\tau^* = 0.1753425$ years, which is approximately 64 days.

4.1.1 Effect of N and $\mathtt{m}$ on price

Regarding the Cheyette parameters in table 4.1, where σ and k_λ were chosen to be equal to the optimal values obtained in Krekel et al.[2009] for an iTRAXX S10 IG five year CDS, we start by analysing how the price of this payer CDS option changes as function of the number of steps of the tree (N), for several number of partitions ($\mathtt{m}$) of the interval of ϕ.

As we can observe in figure 4.1, as the number of steps (N) and the number of partitions ($\mathtt{m}$) of the interval of the accumulated variance (ϕ) increases, keeping the other parameters constant (table 4.1), the price of this payer CDS option for a five year CDS contract tends to get more stable. In

[2]We recall that $D = \sum_{i=k}^{n-1} \delta_i \bar{P}(t_0, T_{i+1})$ and $\xi_{k,n}(t_0) = (1-R)\dfrac{\sum_{i=k}^{n-1}[S(t_0,T_i)P(t_0,T_{i+1}) - \bar{P}(t_0,T_{i+1})]}{\sum_{i=k}^{n-1}\delta_i \bar{P}(t_0,T_{i+1})}$.

σ	k_λ	ϕ_0	λ_0	r_0	R	ξ(bp)
0.94	-0.10	0.0	0.02	0.01	0.4	100

Table 4.1: Cheyette parameters for the price convergence study of a payer CDS option (example 1).

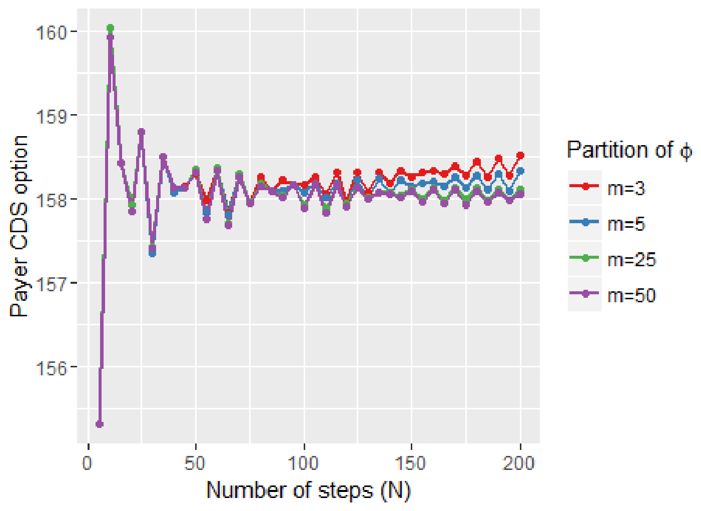


Figure 4.1: Payer CDS option price as a function of the number of steps of the tree (N), for different number of partitions (m) of the interval of ϕ (example 1).

particular, we note that for $5 \leq N \leq 75$, the value of m appears to have no effect on this price. However, for $75 < N \leq 200$, the value of m produces some variations in the pricing of this payer CDS option, although the values are close, approximately between 158 and 159 (bp). Thus, for $N = 200$, although the difference of prices between m=3 and m=50 its around 0.5 (bp), we can observe that the price starts to converge as the value of m increases and also that there is no significant difference between m=25 and m=50 on the calculation of the price of this option. Moreover, it is important to mention that, as expected the computational time increases as the values of N and m increase.

Therefore, in order to get a more stable price and to save some computational time, we continue this example with $N = 200$ and m=25.

4.1.2 Bisection vs Newton's method

Next, we calibrate the implied volatility as function of the strike price (ξ), for a payer CDS option with a constant set of parameters shown in table 4.2. So, in order to calibrate the implied volatility, we consider root-finding methods, namely, bisection method and Newton's method.

σ	k_λ	ϕ_0	λ_0	r_0	R	m	N
0.94	-0.10	0.0	0.02	0.01	0.4	25	200

Table 4.2: Cheyette parameters for the root-finding methods (example 1).

According to table 4.3, we can see, as expected, that the price of this payer CDS option decreases as the strike price increases and the values of the implied volatility, calculated using the bisection and Newton's method, are approximately equal. Thus, in order to get a more clear visualization of these

ξ (bp)	Cheyette Price (bp)	Implied Volatility	
		Bisection	Newton
80	216.782	129.36	129.36
90	185.628	127.14	127.14
100	158.090	125.48	125.48
110	133.856	123.97	123.98
120	112.729	122.58	122.58
130	94.597	121.41	121.41
140	79.151	120.42	120.42
150	66.024	119.51	119.51
160	54.925	118.66	118.66
170	45.629	117.91	117.91

Table 4.3: Cheyette price and implied volatility (calibrated from bisection method and Newton's method), for each value of the strike price (ξ) and for a payer CDS option with parameters 4.2.

results, we decide to plot the implied volatility as function of the strike price (ξ) for both methods (see 4.2). In this plot, we highlight the point that represents the value of the implied volatility for the ATM option (i.e, when the strike price is, such that, $\xi = \xi_{k,n}(t_0) = 120.301$ (bp)), as we can observe in figure 4.2. In this case, for values of $\xi < 120.301$(bp) the option is ITM which means that is worth exercising but for values of $\xi > 120.301$(bp) the option has negative intrinsic value and so it does not worth exercising at the expiry date.

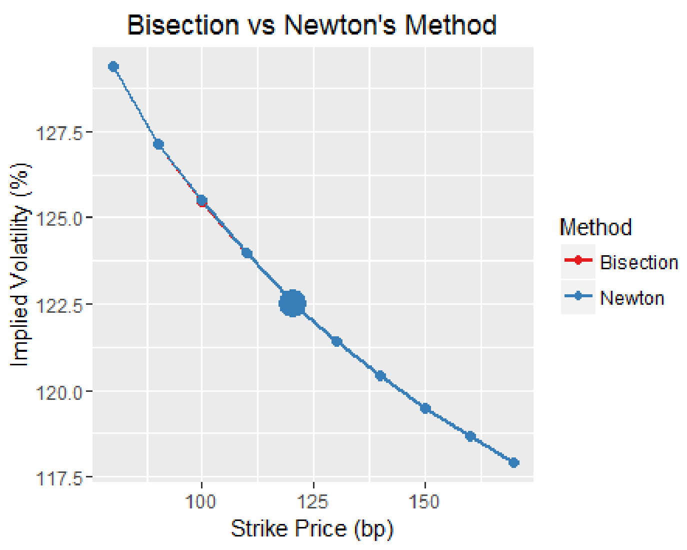


Figure 4.2: Implied volatility as function of the strike price (ξ), calibrated from bisection method (red) and the Newton's method (blue), for a payer CDS option with parameters 4.2.

Therefore, as we can see, there is no significant differences between both methods. So from now on we proceed this study presenting the results from the application of the Newton's method, because it converges faster than the bisection method.

4.1.3 Effect of k_λ and σ in the price and in the implied volatility

Now we analyse the impact of a change in the mean reversion parameter (k_λ) and in the volatility parameter (σ) of the Cheyette algorithm in the pricing and calibration of the implied volatility of a payer CDS option.

Keeping the set of parameters (as shown in table 4.4) constant, we consider the following two cases:

- Effect of k_λ in the price and implied volatility, with k_λ taking the values -0.1, -0.01, 0.01 and 0.1, and $\sigma = 0.94$;
- Effect of σ in the price and implied volatility, with σ taking the values 0.5, 0.75 and 0.9, and $k_\lambda = -0.1$.[3]

ϕ_0	λ_0	r_0	R	m	N
0.0	0.02	0.01	0.4	25	200

Table 4.4: Cheyette parameters for the study of the effect of k_λ and σ (example 1).

In figure 4.3 we plot the price (graph (a)) and the implied volatility (graph (b)) of a payer CDS option, as a function of the strike price (ξ), for several values of k_λ. In graph (a), we can observe that the payer CDS option price decreases as the value of ξ increases and the prices, for a fixed strike price, decrease with increasing k_λ. In graph (b) we can observe one of the three following behaviours:

- For values of k_λ nearer zero (-0.01 and 0.01) the implied volatility is almost constant as a function of ξ. In particular, when $k_\lambda = -0.01$ (resp. $k_\lambda = 0.01$) the values of the implied volatility are approximately 0.96 (resp. 0.91);
- For $k_\lambda = -0.1$, the implied volatility decreases with the strike price (the values of the implied volatility, in this case, vary from 1.29362 to 1.17917);
- Contrary to the previous case, when $k_\lambda = 0.1$ the implied volatility increases with the strike price increases (the values of the implied volatility vary between 0.68497 and 0.76796);

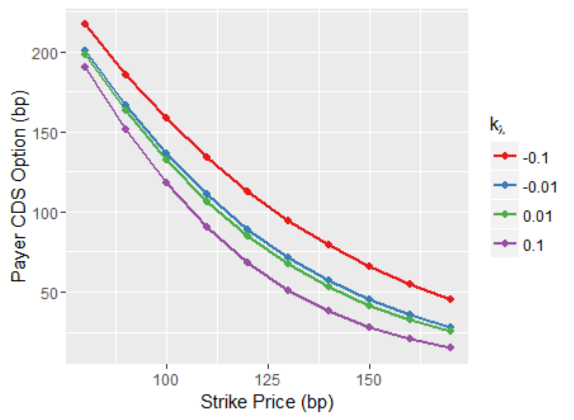
Moreover, this example suggests that, for a fixed strike price, the implied volatility tends to increase when k_λ decreases.

We perform the same kind of analysis but now for different values of σ, as a function of the strike price ξ. The numerical illustration is presented in figure 4.4. In this figure we can observe that both price and implied volatility decrease as the strike price (ξ) increases and, for a fixed strike price, both increase with increasing σ.

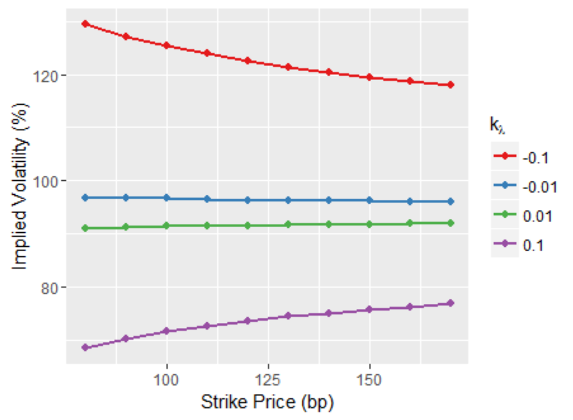
4.1.4 Impact of σ and k_λ in the implied volatility for an ATM option

It is also important to study the implied volatility when the option is *at-the-money* (ATM) because, in financial markets, the implied volatilities used are, usually, calibrated from an ATM option price.

[3]With these values, we can assure that the integer J is equal to zero, the only case that we address in this thesis.

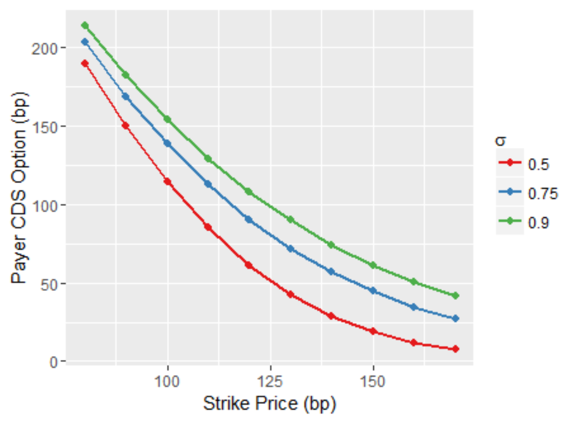


(a) Payer CDS option price as a function of the strike price (ξ), for several values of k_λ.

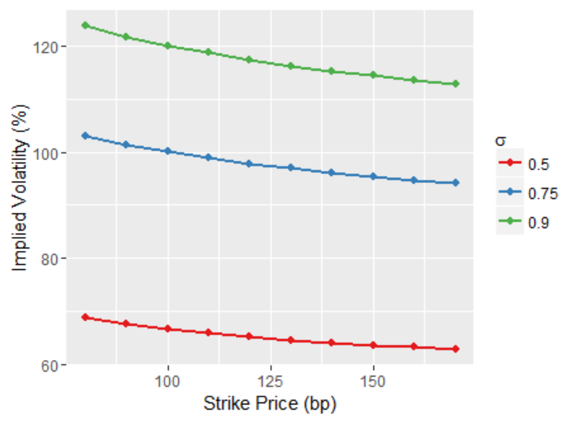


(b) Implied volatility as a function of the strike price (ξ), for several values of k_λ.

Figure 4.3: Effect of k_λ in the price and implied volatility, for a payer CDS option with parameters 4.4 and $\sigma = 0.94$ (example 1).



(a) Payer CDS option price as a function of the strike price (ξ), for several values of σ.



(b) Implied volatility as a function of the strike price (ξ), for several values of σ.

Figure 4.4: Effect of σ in the price and implied volatility, for a payer CDS option with parameters 4.4 and $k_\lambda = -0.1$ (example 1).

Thus, our goal now is to determine how a change in the volatility parameter (σ) impacts the implied volatility, for several values of the mean reversion parameter (k_λ), for the set of values considered in table 4.4, setting the strike price of the ATM option equal to $\xi = \xi_{k,n}(t_0) = 120.301$ (bp).

As we can observe in figure 4.5, the implied volatility increases with σ, and it appears to change almost linearly. Moreover, for fixed σ, the implied volatility is larger with smaller k_λ and it seems that the slope of the line (assuming that indeed there is a linear dependence between the implied volatility and σ) increases when k_λ decreases.

4.1.5 Evolution of the price and the implied volatility for different values of λ_0

Here, we pretend to investigate how a change in the initial default intensity (λ_0) impacts the implied volatility.

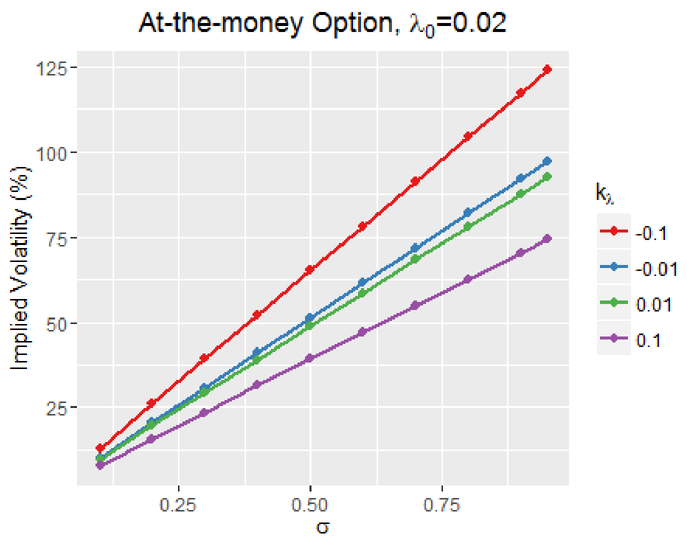


Figure 4.5: Implied volatility as a function of the volatility parameter (σ) for several values of the mean reversion parameter (k_λ), for an ATM payer CDS option (example 1).

We start by considering, the Cheyette parameters shown in table 4.5 and the initial value of the default intensity (λ_0) taking values in 0.5%, 1%, 1.5%, 2% and 4%. It is important to mention that, for each value of λ_0, we have different values of forward risky level (D) and forward credit swap rate ($\xi_{k,n}(t_0)$), at the valuation date, as we can see in table 4.6.

σ	k_s	ϕ_0	r_0	R	m	N
0.94	-0.10	0.0	0.01	0.4	25	200

Table 4.5: Cheyette parameters for the impact of λ_0 (example 1).

λ_0 (%)	D	$\xi_{k,n}(t_0)$(bp)
0.5	4.797	30.018
1	4.732	60.075
1.5	4.667	90.169
2	4.603	120.301
4	4.360	241.205

Table 4.6: Forward risky level (D) and forward credit swap rate ($\xi_{k,n}(t_0)$), for several values of λ_0 (example 1).

According to table 4.6, we can extract the following remarks:

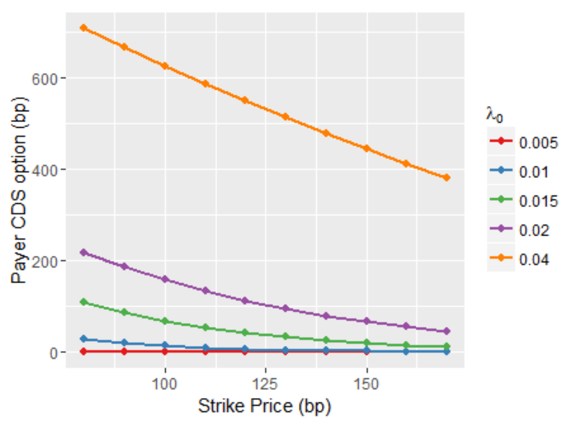
- We can observe that as the value of λ_0 increases, D tends to decrease but $\xi_{k,n}(t_0)$ increases significantly, as indeed expected from their respective formulas;
- For λ_0 equal to 0.5% and 1% the option is OTM, for all the values of ξ shown in table 4.7, because for both cases $\xi_{k,n}(t_0) < 80(bp)$;
- For λ_0 equal to 1.5% (resp. 2%) the option is ITM when $\xi < 90.169$(bp) (resp.$\xi < 120.301$ (bp)) and OTM when $\xi > 90.169$(bp) (resp. $\xi > 120.301$(bp)).

- For λ_0 equals to 4%, the option is ITM, for all the values of ξ shown in table 4.7, because $\xi_{k,n}(t_0) > 170(bp)$.

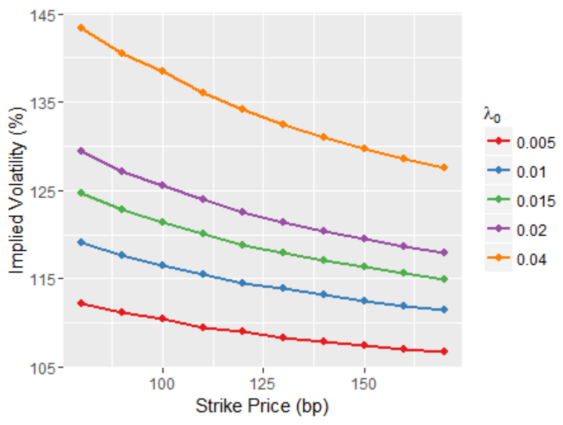
ξ	$\lambda_0 = 0.5\%$	$\lambda_0 = 1\%$	$\lambda_0 = 1.5\%$	$\lambda_0 = 2\%$	$\lambda_0 = 4\%$
80	7.22E-05	28.260	107.560	216.782	707.413
90	3.50E-05	19.537	85.672	185.628	666.353
100	1.72E-05	13.4405	67.869	158.090	626.352
110	8.39E-06	9.193	53.4089	133.856	587.172
120	4.34E-06	6.299	41.788	112.729	549.284
130	2.20E-06	4.337	32.653	94.597	512.683
140	1.14E-06	2.980	25.474	79.151	477.604
150	6.10E-07	2.048	19.827	66.024	444.029
160	3.30E-07	1.412	15.4087	54.9251	412.080
170	1.80E-07	9.79E-05	11.961	45.629	381.674

Table 4.7: Payer CDS option prices for different values of initial default intensity λ_0, where all the information is in basis points (bp) (example 1).

Then we plot the price (graph(a)) and the implied volatility (graph(b)) as a function of the strike price (ξ), for several values of the initial default intensity (λ_0), as shown in figure 4.6.



(a) Payer CDS option price as a function of the strike price (ξ), for several values of λ_0.



(b) Implied volatility as a function of the strike price (ξ), for several values of λ_0.

Figure 4.6: Effect of λ_0 in the price and in the implied volatility for payer CDS options (example 1).

According to figure 4.6 and table 4.7, when the initial default intensity (λ_0) is equal to 0.5%, 1%, 1.5%, 2% and 4%, we can observe that in both graphs, for a fixed strike price (ξ), the price and the implied volatility increases with λ_0. And, for each value of λ_0, the price and implied volatility decreases with ξ. Moreover, we note that the variation of values of the price and implied volatility is grater as λ_0 increases, but in graph (a) this variation is more significant than in graph(b).

4.2 Example 2

In this example, we consider a payer CDS option for a five year CDS with the following dates:

- Option valuation date: 17-10-2008;
- Option expiry date: 20-03-2009;
- Maturity date of the underlying CDS: 20-12-2013;

As in example 4.1, we consider $\lambda_0 = 2\%$ and then, at the valuation date, we have the following values:

- Forward risky level: $D = 4.36026$;
- Forward credit swap rate: $\xi_{k,n}(t_0) = 120.301$(bp);
- Time to expiry: $\tau^* = 0.421918$ years which is approximately 154 days.

4.2.1 Effect of N and $\mathtt{m}$ on price and the effect of k_λ in the implied volatility

As in the previous example, we keep the same set of parameters in table 4.1 constant, and we investigate if the behaviours found in the previous example still hold or, on the contrary, we see major differences. We note that in this example, the time to expiry is significantly larger than the one considered in the first example. Therefore we have a contract with a larger time horizon, and we want to check its impact in the numerical results.

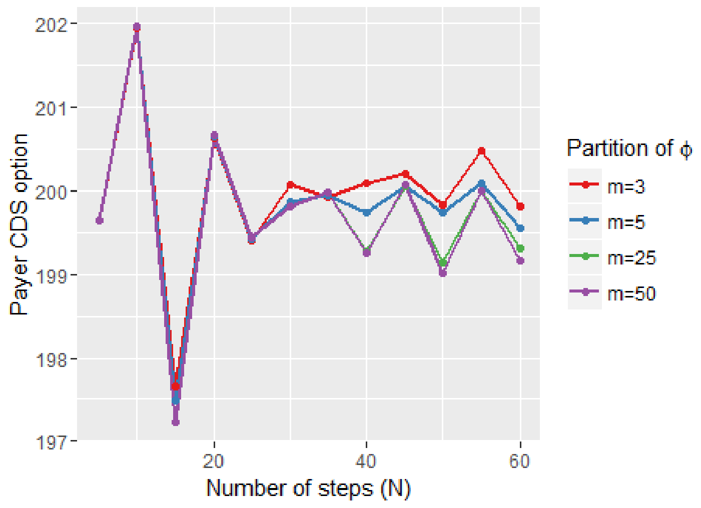


Figure 4.7: Payer CDS option price as a function of the number of steps (N), for different number of partitions $(\mathtt{m})$ of partitions of the interval of ϕ (example 2).

We plot, for $\mathtt{m}$ equal to 3, 5, 25 and 50, the option price as a function of the number of the steps, but only until $N = 60$, because, when calculating the price for $N > 60$, the value of the integer J starts to change (i.e, $J \neq 0$). Thus, this set of parameters does not seems to be a proper choice for this payer CDS option, since we want to calculate the option price, at least, one time per day. With the expiry date that we have, we need to be able to compute the price of the option when $N = 154$. So in order to overcome this problem, we need to tune some values of the parameters. In fact, we just need to change

k_λ, as for values of k_λ close as possible to zero, the implied volatility does not change much (see figure 4.8). Therefore we change, with respect to table 4.1, only the value of k_λ, which now we assume to be equal to -0.001 (see table 4.8)

σ	ϕ_0	λ_0	r_0	R	$\mathtt{m}$	N
0.94	0.0	0.02	0.01	0.4	25	60

Table 4.8: Cheyette parameters for the effect of k_λ (example 2).

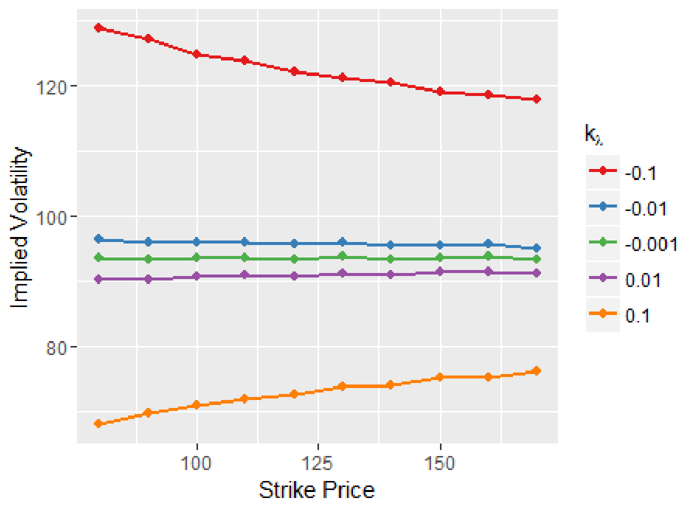


Figure 4.8: Implied volatility as a function of the strike price (ξ), for several values of k_λ, with parameters 4.8 (example 2).

Therefore, we change the value of the mean reversion parameter (k_λ) to, say, -0.001 and according to our expectations, if we increase N until, say 200, the integer J stays equal to zero.

Regarding the set of parameters in table 4.9, we plot again the price of this payer CDS option as a function of N, for $\mathtt{m}$ equal to 3, 5, 25 and 50, as shown in figure 4.9.

σ	k_λ	ϕ_0	λ_0	r_0	R	ξ(bp)
0.94	-0.001	0.0	0.02	0.01	0.4	100

Table 4.9: Cheyette parameters for pricing the payer CDS option as a function of N, for several values of $\mathtt{m}$ (example 2).

As we can observe in figure 4.9, as the number of steps (N) and the number of partitions of ϕ ($\mathtt{m}$) increases, the price of this payer CDS option tends to get more stable. However, we notice that, in this case, for $m = 3$ we observe a greater numerical instability in the calculation of the price, as N increases. For $\mathtt{m}$ equals to 5, 25 and 50, the prices are, approximately, between 165 and 164 (bp) when $100 \leq N \leq 200$, and the difference between $m = 25$ and $m = 50$ is even smaller.

Therefore, in order to save computational time and get a more stable price, we decide to continue this analysis with $m = 25$ and $N = 154$.

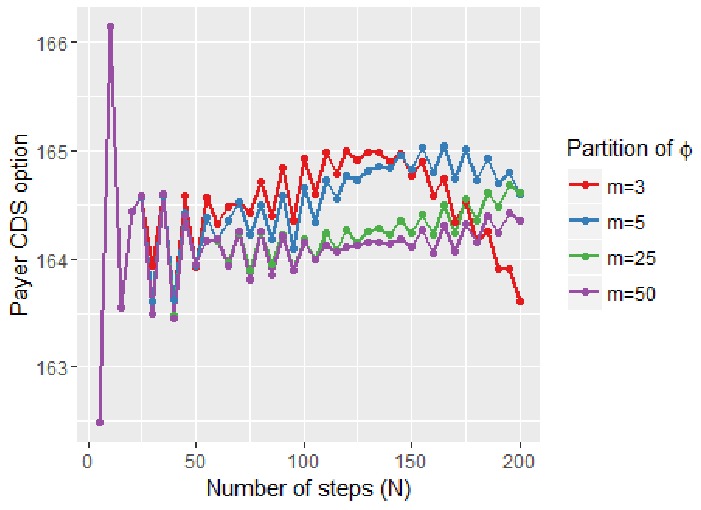


Figure 4.9: Payer CDS option price as a function of the number of steps of the tree (N), for several values (m) of partitions of the interval of ϕ (example 2).

4.2.2 Effect of σ in the implied volatility

Next, we analyse the impact of a change in the volatility parameter (σ) in the implied volatility of a payer CDS option, keeping the set of parameters in table 4.13 constant.

We plot the implied volatility as a function of the strike price, for σ equals to 0.5, 0.75 and 0.9 with k_λ equal to -0.001, as we can see in figure 4.10.

ϕ_0	λ_0	k_λ	r_0	R	m	N
0.0	0.02	-0.001	0.01	0.4	25	154

Table 4.10: Cheyette parameters for the study of the effect of σ (example 2).

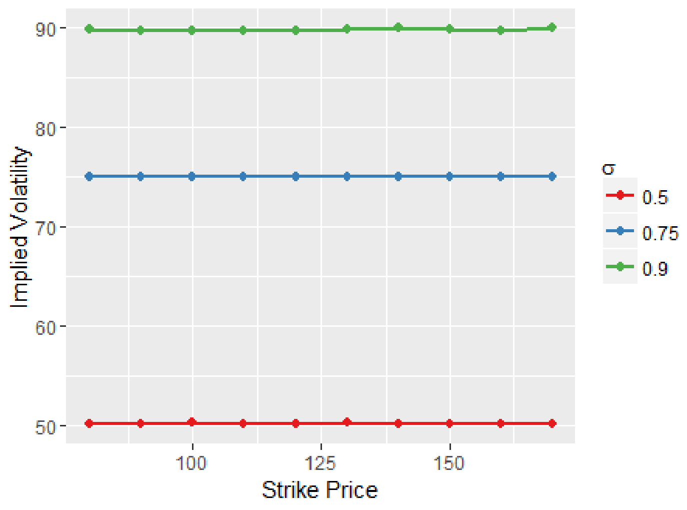


Figure 4.10: Implied volatility as a function of the strike price (ξ), for several values of σ (example 2).

In figure 4.14 we plot the implied volatility as a function of the strike price (ξ), for several values of

σ. Also, as previously, for a fixed strike price, the implied volatility increases as the value of σ increases. However, the implied volatility seems relatively constant for a fixed value of σ and changing the strike price ξ, and this behaviour is different from the one that we have observed in example 1. We will come back to this question latter on, in example 3.

4.2.3 Impact of σ and k_λ in the implied volatility for an ATM option

As in the previous example, for an ATM option the strike price is such that $\xi = \xi_{k,n}(t_0)$ and also, in this case, is equal to 120.301 (bp). So given a change in the Cheyette volatility (σ), we pretend to determine what is the implied volatility for an ATM payer CDS option for k_λ equals to -0.01, 0.001 and 0.01, keeping the set of parameters in table 4.11 constant.

ϕ_0	λ_0	r_0	R	ξ(bp)	m	N
0.0	0.02	0.01	0.4	120.301	25	154

Table 4.11: Cheyette parameters for an ATM payer CDS option (example 2).

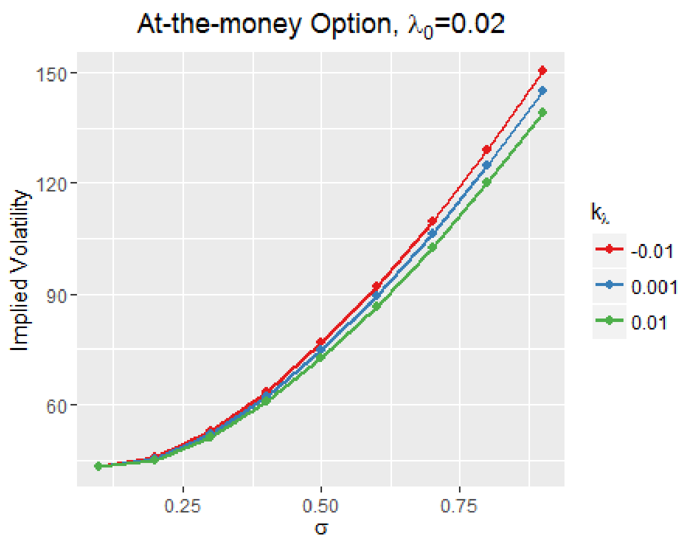


Figure 4.11: Implied volatility as a function of σ, for an ATM payer CDS option, for several values of k_λ, with parameters 4.11.

According to figure 4.11, we can observe that for this ATM option, there is no significant change on the values of the implied volatility when k_λ changes. Moreover, we observe the same kind of behaviours that we mentioned before, in example 1.

4.3 Example 3

We consider a payer CDS option for a five year CDS with the same dates as in the example 4.1, in particular:

- Option valuation date: 17-10-2008;

- Option expiry date: 20-12-2008;
- Maturity date of the underlying CDS: 20-12-2013.

But now σ and k_λ are different from the previous examples, as we choose exactly the same values as Krekel et al. [2009], for an iTRAXX S10 Xover five year CDS, shown in table 4.12.

We note that we choose $\lambda_0 = 8\%$ because for smaller values of this parameter, the obtained prices are almost zero and therefore not interesting to study.

At the valuation date, we have the following values:

- Forward risky level: $D = 3.92057$;
- Forward credit swap rate: $\xi_{k,n}(t_0) = 484.835$(bp);
- Time to expiry: $\tau^* = 0.1753425$ years, which is approximately 64 days.

4.3.1 Effect of N and $\mathtt{m}$ on price

Firstly we study how the price of this payer CDS option changes as a function of the number of steps of the tree (N), for several number of partitions ($\mathtt{m}$) of the interval of ϕ.

σ	k_λ	ϕ_0	λ_0	r_0	R	ξ(bp)
1.39	0.39	0.0	0.08	0.01	0.4	600

Table 4.12: Cheyette parameters for the price convergence study of a payer CDS option (example 3).

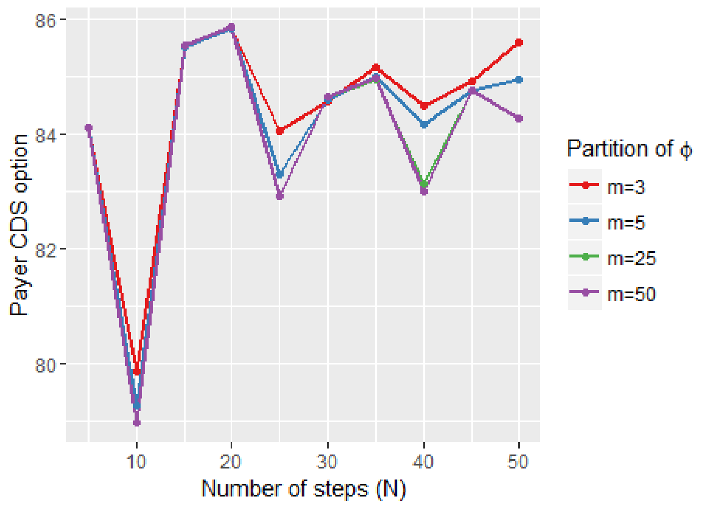


Figure 4.12: Payer CDS option price as a function of the number of steps (N), with N until 50, for different number of partitions ($\mathtt{m}$) of the interval of ϕ (example 3).

As we can observe in figure 4.12 and regarding the set of parameters that we have chosen, we notice that we cannot compute the price of this payer CDS option, at least, one time per day because, the values of J starts to change (i.e, $J \neq 0$) (we can only compute for N up to 50). Thus, as in example

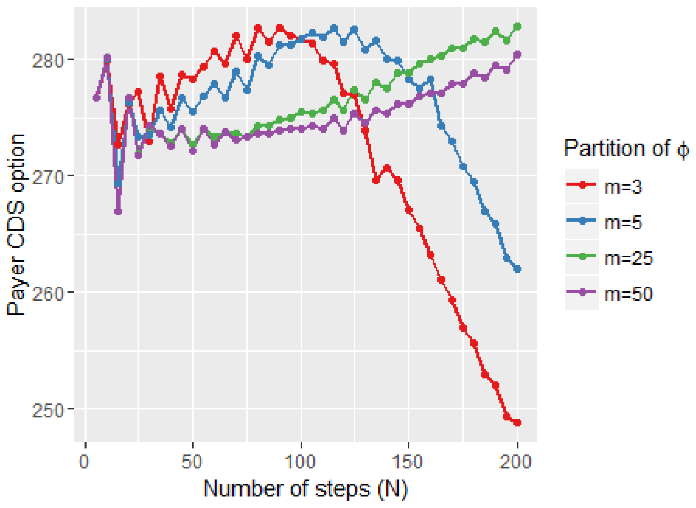


Figure 4.13: Payer CDS option price as a function of the number of steps (N), for different number of partitions ($\mathtt{m}$) of the interval of ϕ (example 3).

4.2, we change the value of k_λ, for example, to 0.01; the results for this new set of values are presented in figure 4.13.

As we can observe in figure 4.13, we have more instability on the computation of the price of this payer CDS option as the number of steps (N) increases, when $\mathtt{m}$ is equal to 3 and 5. Moreover, if we consider $N \leq 100$ and $\mathtt{m}$ equals to 25 and 50, we see that doesn't seems to exist a significant difference on the computation of the price and it tends to get more stable as N increases. However, for $N > 100$ and $m = 25$ and $m = 50$, we observe a significant difference between the prices.

We note that the prices in figure 4.12 are around 83 (bp) and the prices in figure 4.13 are around 275 (bp). Thus, although we only have changed the value of k_λ, it has a significant impact in the computation of the price.

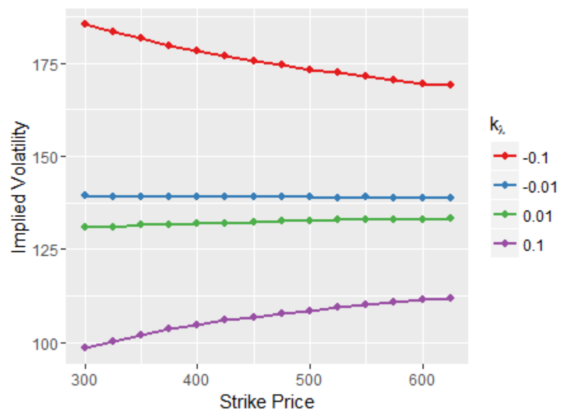
Therefore, we chose to continue this analysis with $N = 64$ and $m = 25$, in order to save computational time and to price this option one time per day.

4.3.2 Effect of k_λ and σ in the implied volatility

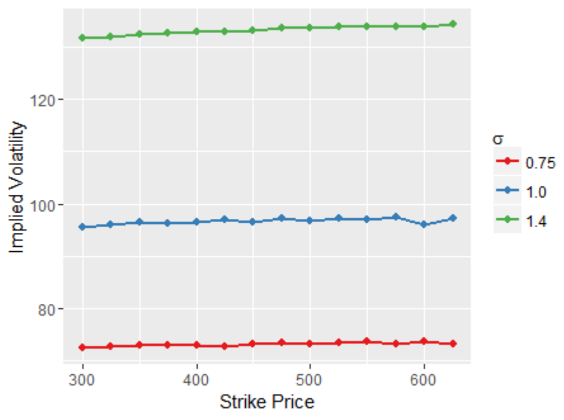
Next, we analyse the impact of a change in the mean reversion term (k_λ) and the volatility parameter (σ) in the implied volatility of a payer CDS option, keeping the set of parameters in table 4.13 constant. For the first case, we chose k_λ equal to -0.1, -0.01, 0.01 and 0.1 with σ equal to 1.39 (see graph (a) from figure 4.14). And, for the second case, we chose σ equal to 0.75, 1.0 and 1.4 with k_λ equal to 0.01 (see graph (b) from figure 4.14).

ϕ_0	λ_0	r_0	R	m	N
0.0	0.08	0.01	0.4	25	64

Table 4.13: Cheyette parameters for the study of the effect of k_λ and σ in the implied volatility (example 3).

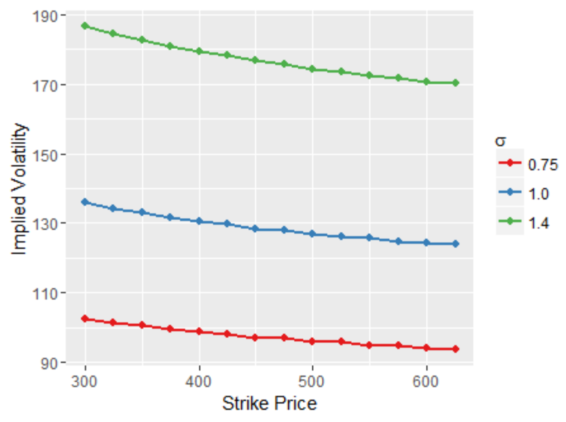


(a) Implied volatility as a function of the strike price (ξ), for several values of k_λ.

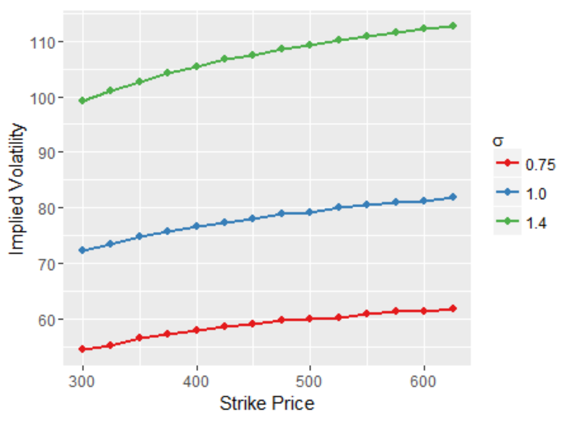


(b) Implied volatility as a function of the strike price (ξ), for several values of σ.

Figure 4.14: Effect of k_λ and σ in the implied volatility for a payer CDS option with parameters 4.13.



(a) Implied volatility as a function of the strike price (ξ), for several values of σ, with $k_\lambda = -0.1$.



(b) Implied volatility as a function of the strike price (ξ), for several values of σ, with $k_\lambda = 0.1$.

Figure 4.15: Effect of σ in the implied volatility for a payer CDS option with parameters 4.13.

In figure 4.14 we plot the implied volatility as a function of the strike price (ξ), for different values of k_λ in graph (a), and for different values of σ in graph (b). For graph (a), we observe the same patterns as in the previous examples. Also, for graph (b), for a fixed strike price, the implied volatility increases with σ, as we have noticed, for example, in example 1. But now we remark other kind of behaviours that we didn't notice previously. In particular, for this choice of parameters, the implied volatility seems constant (as a function of the strike price), with a sightly tendency to increase. We suspect that the shape of the curve of graph (b) its related with the value of k_λ. So we decide to plot the implied volatility as a function of ξ, for different values of σ, with k_λ equal to -0.1 and 0.1 (respectively plot a) and b)), as represented in figure 4.15.

Therefore, according to figures 4.14 and 4.15, the relationship between the implied volatility and ξ, for several values of σ, appears to evolve on the same way as k_λ in graph (a) from figure 4.14 (for each choice of this parameter).

4.3.3 Evolution of the price and the implied volatility for different values of λ_0

In order to study the evolution of the price and the implied volatility for different values of λ_0, we consider the following set of parameters:

σ	k_λ	ϕ_0	r_0	R	$\mathtt{m}$	N
1.39	0.01	0.0	0.01	0.4	25	64

Table 4.14: Cheyette parameters for the impact of λ_0 (example 3).

In this case, we chose λ_0 equal to 2%, 4%, 6% and 8% and we used the corresponding information in tables: 3.4, B.2 and B.2 (the last two can be seen in Appendix B), to compute the relevant quantities. As in example 1, we have different values of forward risky level (D) and forward credit swap rate ($\xi_{k,n}(t_0)$), at the valuation date, as we can see in table 4.15.

λ_0 (%)	D	$\xi_{k,n}(t_0)$(bp)
2	4.60374	120.301
4	4.3601	241.205
6	4.13279	362.715
8	3.92057	484.835

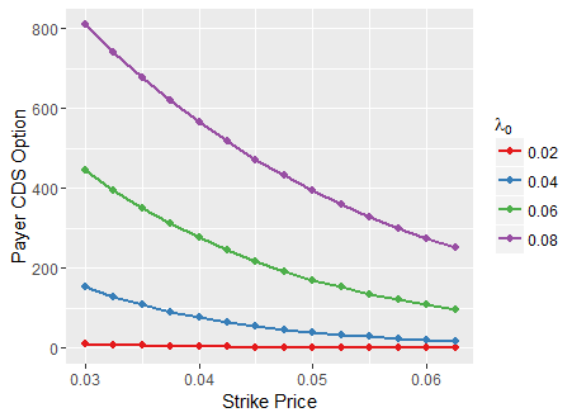
Table 4.15: Forward risky level (D) and forward credit swap rate ($\xi_{k,n}(t_0)$) for several values of λ_0 (example 3).

From table 4.15, when λ_0 is equal to 2% and 4%, for all the values of ξ in figure 4.16, the option is OTM. But, when λ_0 is equal to 6% (resp. 8%), the option is ITM for $\xi < 362.715$(bp) (resp. $\xi < 484.835$(bp)) and is OTM for $\xi > 362.715$(bp) (resp. $\xi > 484.835$(bp)).

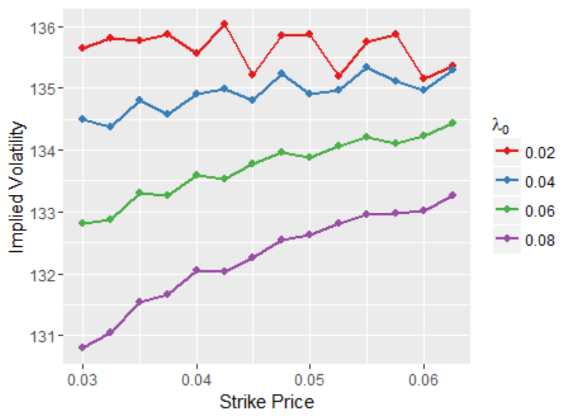
ξ	$\lambda_0 = 2\%$	$\lambda_0 = 4\%$	$\lambda_0 = 6\%$	$\lambda_0 = 8\%$
300	10.979	152.240	446.344	808.585
325	8.168	126.996	395.253	740.344
350	6.087	106.988	350.671	677.672
375	4.595	89.421	310.21	619.207
400	3.430	75.613	275.325	566.069
425	2.681	63.842	243.597	516.395
450	1.968	53.693	216.352	471.548
475	1.577	45.951	192.257	430.894
500	1.229	38.660	170.366	393.210
525	9.187E-05	32.938	151.720	359.165
550	7.506E-05	28.416	135.163	328.124
575	5.985E-05	24.114	120.03	299.459
600	4.509E-05	20.537	107.138	273.502
625	3.654E-05	17.843	95.9134	250.482

Table 4.16: Payer CDS option prices for different values of initial default intensity (λ_0), where all the prices and strike prices are represented in basis points (bp)(example 3).

From figure 4.16 and table 4.16, and considering the initial value of the default intensity (λ_0) taking values in 2%, 4%, 6% and 8%, we can do the following comments:



(a) Payer CDS option price as a function of the strike price (ξ), for several values of λ_0 .



(b) Implied volatility as a function of the strike price (ξ), for several values of λ_0.

Figure 4.16: Effect of λ_0 in the price and implied volatility for a payer CDS options (example 3).

- Like in example 4.1, the payer CDS option price decreases as λ_0 and/or strike price (ξ) increase. And the variation of prices tends to increase as the values of λ_0 increases, as we can see more clearly according to table 4.16;
- Regarding the implied volatility, the behaviour of it as a function of the strike price ξ is non-monotonic, although for values of λ equal to 4, 6 and 8 %, it shows a tendency to increase. When $\lambda = 2\%$, this is no longer apparent, and we remark a quite unstable behaviour of the implied volatility as a function of the strike price.

4.3.4 Impact of k_λ and σ in the implied volatility of an ATM option

We consider the same kind of analysis, but now for the ATM payer CDS option. We consider the implied volatility for σ equals to 0.1,0.2,...,1.5, and for k_λ equals to -0.1,0.01,0.1 and 0.2. In this case, an ATM option has a strike price (ξ) of 484.835 (bp) with the set of parameters in table 4.17.

λ	ϕ_0	r_0	ξ(bp)	R	$\mathtt{m}$	N
0.08	0.0	0.01	484.835	0.4	25	64

Table 4.17: Cheyette price parameters for an ATM payer CDS option (example 3).

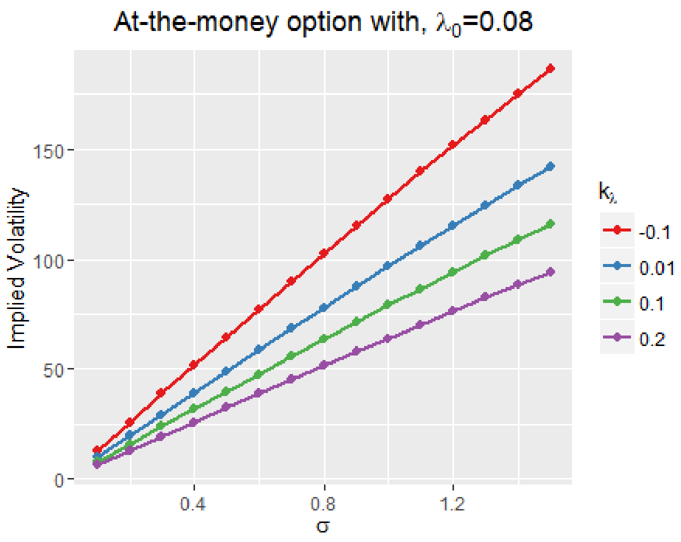


Figure 4.17: Implied volatility for an ATM payer CDS option as a function of the volatility parameter (σ), for several values of k_λ (example 3).

In figure 4.17 we observe the same patterns as the corresponding ones observed in the previous examples.

In order to assess the impact of the strike price, we choose different combinations of k_λ and σ with the same implied volatility. Here, for the illustration, we have chosen the implied volatility approximately equal to 75%, and the corresponding pairs (k_λ, σ) equal to (-0.1,0.58), (0.01,0.77), (0.1,0.95) and (0.2,1.18). For these choices, the ATM payer CDS option price is, approximately, 238 (bp).

In figure 4.18 we plot, for these particular pairs, the evolution of the implied volatility with the strike price. For all the pairs, and for the initial value of the strike price 484.835 (bp), the corresponding implied volatility is 75%, as previously fixed.

According to figure 4.18, although we have considered the same implied volatility for the ATM option (75%) for the different sets of parameters, (k_s, σ), we can observe that, for each pair (k_s, σ), the implied volatility exhibits different patterns as a function of the strike price (ξ). In particular,

- For $(k_\lambda = -0.1, \sigma = 0.58)$, the implied volatility tends to decrease (approximately, from 75.1% to 72.5%) as ξ increases;
- For $(k_\lambda = 0.01, \sigma = 0.77)$, we notice the same pattern, but now the range of values is lower, and exhibits a more constant behaviour (around 75.3%);

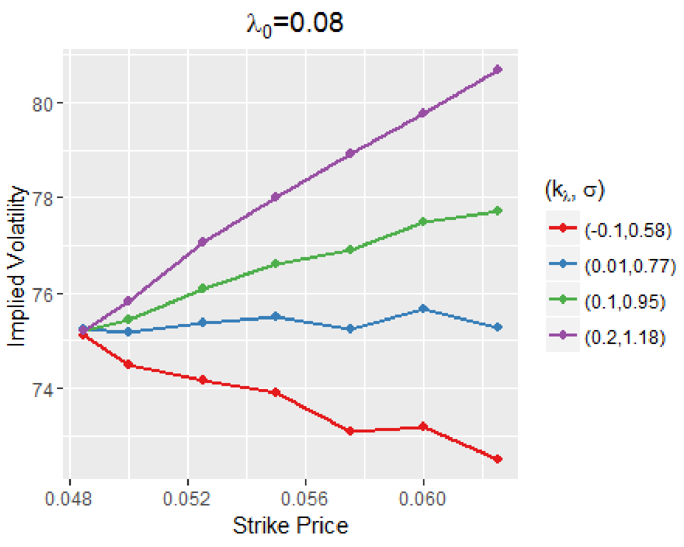


Figure 4.18: Implied volatility as a function of the strike price (ξ), for several pairs (k_λ, σ) (example3).

- For $(k_\lambda = 0.1, \sigma = 0.95)$, the implied volatility tends to increase (approximately, from 75.2% to 77.73%) as ξ increases;
- For $(k_\lambda = 0.2, \sigma = 1.18)$, the implied volatility increases as ξ increases (approximately, from 75.2% to 80.68%), more significantly as the previous case (the slope of the line is higher).

Chapter 5

Conclusion

In this thesis, the main goal was to compute the price of a CDS option in a recombining tree based on a default intensity model derived in Krekel et al. [2009]. Since this is an Markovian model one and has a closed formula for the forward survival probability, we are in conditions to build a recombining tree for the computation of CDS option prices. In this sense, we developed the Cheyette algorithm inspired in the Li et al. [1995] framework and made a sensitivity study in order to analyse the numerical features of this procedure.

Since we approached some concepts that are not usually used in the field of Mathematics, before drawing the algorithm, in Chapter 2, we explained some (basic) financial concepts for a better understanding of the problem. Also, we provide the default intensity model from Krekel et al. [2009]. This model was used to help us write an adaptation of the numerical method from Li et al. [1995].

In Chapter 3, regarding the default intensity model and the relevant CDS and CDS options formulas stated in the previous Chapter, we adapted the Li et al. [1995] approach to our case of interest, CDS options. In this context, we developed the Cheyette algorithm taking into account some specifications and initial conditions, that allowed us to overcome some numerical issues in the implementation of this procedure.

In Chapter 4, we did a very exhaustive analysis in order to find some patterns and investigate the numerical limitations in the Cheyette algorithm. In particular, we made a sensitivity study where we investigated how a modification in the input parameters impacts the pricing and calibration of the implied volatility of payer CDS options. Having the results obtained in this Chapter in mind, we can extract the following remarks:

1. As expected from Li et al. [1995], we observed that the price of a payer CDS option tends to converge with the increasing of the number of steps of the recombining tree (N) and with the number of partitions ($\mathtt{m}$) of the interval of the accumulated variance;

2. As expected from Krekel et al. [2009], we observed that the mean reversion (k_λ) and the volatility (σ) parameters have a reversed effect on the pricing of a payer CDS option. This means that, for a fixed strike price, the price and the implied volatility increase with increasing σ and decreasing k_λ;

3. For the case of ATM payer CDS options, we observed the same patterns among the illustrative examples. These examples suggested that the implied volatility assumes an almost linear behaviour with σ;

4. However, we observed peculiar numerical behaviours when analysing the evolution of the implied volatility for different values of λ_0 in examples 4.1 and 4.3.

Therefore we believe that the major contributions of this dissertation were the formulation of a numerical procedure and the implementation of a parsimonious and efficient algorithm, that enable us to price CDS options.

Nevertheless, we consider that there are some aspects that need to be improved. So, as future work we suggest the following:

1. Concerning the last item of the previous remarks, a careful investigation should be done in order to find a justification for this numerical instability;

2. In the context of this dissertation, we have made an exhaustive sensitivity study (in Chapter 4) using only the same dates of the payer CDS options as Krekel et al. [2009], and changing the input parameters of the Cheyette algorithm. However, we did not apply the developed algorithm to real data. In this direction, we believe that it is extremely important that, in future works, the Cheyette algorithm is applied to recent real data in order to explore the performance and accuracy of this procedure in the financial markets. Due to the fact that the Cheyette algorithm seems to be a parsimonious and efficient method to compute the price of these financial instruments;

3. We consider that it would be also very interesting to develop similar numerical procedures for other kinds of credit derivatives with the corresponding changes in the formulas and then perform a similar analysis. In that analysis, one should look for patterns and also numerical peculiarities as we did in this thesis.

q

Bibliography

[1] Credit derivative. Instopedia. `http://www.investopedia.com/terms/c/creditderivative.asp`.

[2] D. Brigo and F. Mercurio. *Interest Rate Models - Theory and Practice with Smile, Inflation and Credit, Chapter 1. Definitions and Notation*. Springer, 2nd edition, 2006. ISBN:3-540-22l49-2.

[3] D. Brigo and M. Morini. Cds market formulas and models. `http://wwwf.imperial.ac.uk/~dbrigo/cdsmktfor.pdf`, September 2005.

[4] O. Cheyette. Markov representation of the heath-jarrow-morton model. BARRA Inc. 1995 University Ave. Berkeley, CA 94704, August 1996. Presented at the UCLA Workshop of the Future of Fixed Income Financial Theory April 8 and 9, 1994.

[5] D. Duffie and K. J. Singleton. *Credit Risk: Pricing, Measurement and Management*. Princeton University Press, 2003.

[6] D. Filipović. *Term-Structure Models: A Graduate Course, Chapter 2. Interest Rates and Related Contracts*. Springer, 2009. ISBN:978-3-540-09726-6.

[7] D. Heath, R. Jarrow, and A. Morton. Bond pricing and the term structure of interest rates: a new methodology for contingent claims valuation. *Econometria*, 60(1):77–105, 1992.

[8] J. C. Hull. *Options, Futures and other derivatives*. Prentice Hall, 8^{th} edition, 2012. ISBN:978-0-13-216494-8(hbk.).

[9] M. Krekel, K. Natcheva-Acar, and S. K. Acar. Modeling credit spreads with the cheyette model and its application to credit default swaptions. *The Journal of Credit Risk*, 2009.

[10] A. Li, P. Ritchken, and L.Sankarasubramanian. Lattice models for pricing american interest rate claims. *Journal of Finance*, 1995.

[11] T. Mikosch. Elementary stochastic calculus with finance in view. In *Advanced Series on Statistical Science and Applied probability*, volume 6. Ole E. Barndoff-Nielsen, 1999. ISBN: 9810235437 (alk. paper).

[12] D. O'Kane. *Modelling single-name and multi-name Credit Derivatives*. John Wiley and Sons Ltd, 2008. ISBN:978-0-470-51928-8(HB).

[13] P. Ritchken and L. Sankarasubramanian. Volatility structures of forward rates and the dynamics of the term structure. *Mathematical Finance*, 50:55–72, 1995.

[14] S. M. Schaefer. Single name credit derivatives. Website, Summer 2012. Credit Risk Elective: http://dse.univr.it/safe/documents/SSEFCANAZEI2012/04_single_name_credit_derivatives.pdf.

[15] P. J. Schönbucher. Credit risk modeling and credit derivatives. Master's thesis, Rheinischen Friedrich-Wilhelms-Universit´at Bonn, 2000.

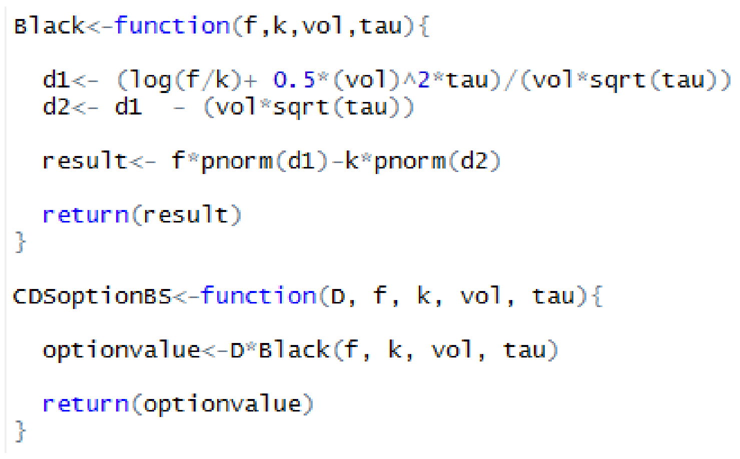
[16] P. J. Schönbucher. *Credit Derivatives Pricing Models: Models, Pricing and Implementation*. Wiley, 2003. ISBN:0-470-84291-1.

[17] P. Tankov and N. Touzi. Calcul stochastique en finance. *Ecole Polytechnique Paris*, September 2015. http://www.cmap.polytechnique.fr/~touzi/Poly-MAP552.pdf.

Appendix A

Implementation

A.1 Implied volatility



```
Black<-function(f,k,vol,tau){

  d1<- (log(f/k)+ 0.5*(vol)^2*tau)/(vol*sqrt(tau))
  d2<- d1  - (vol*sqrt(tau))

  result<- f*pnorm(d1)-k*pnorm(d2)

  return(result)
}

CDSoptionBS<-function(D, f, k, vol, tau){

  optionvalue<-D*Black(f, k, vol, tau)

  return(optionvalue)
}
```

Figure A.1: Black's formula according to equations (2.50) and (2.51).

A.1.1 Bisection method

```
impliedvol<-function(D, fswap, k, tau, price , sigma_up, sigma_down, itmax, precision){
  if((CDSoptionBS(D, fswap, k, sigma_down, tau)-price)*
      (CDSoptionBS(D, fswap, k, sigma_up, tau)-price)>0){
    print("f(sigma_down) and f(sigam_up) must be of opposite sign!")
    }
  else{
    sigma_mid<-(sigma_up+sigma_down)/2 #mid point of the interval [sigma_down , sigma_up]
    miderror<-CDSoptionBS(D, fswap, k, sigma_mid, tau)-price # f(sigma_mid)
    downerror<-CDSoptionBS(D, fswap, k, sigma_down, tau)-price # f(sigma_down)
    it<-1
    while( abs(miderror) > precision  && it<itmax && miderror!=0){
      if(downerror*miderror<0){
        sigma_up<-sigma_id
      }else{
        sigma_down<-sigma_mid
      }
      sigma_mid<-(sigma_up+sigma_down)/2
      miderror<-CDSoptionBS(D, fswap, k, sigma_mid, tau)-price
      downerror<-CDSoptionBS(D, fswap, k, sigma_down, tau)-price
      it<-it+1
    }
    if(it==itmax){
      return(NA)
    }else{
      #if abs(f(sigma_mid)) is less than a thresold we found the implied volatility !
      return(sigma_mid)
    }
  }
}
```

Figure A.2: Bissection method.

A.1.2 Newton's method

```
BSvega<-function(D,f,k,sigma,tau){
  d1 <- (log(f/k)+(sigma^2 * tau)/2)/(sigma*sqrt(tau))
  value<-f *D*dnorm(d1) * sqrt(tau)
  return(value)
}
```

Figure A.3: Derivative of the value of a payer CDS option as function of the volatility (σ).

```
impliedvolNR<-function(D, fswap, k, tau, marketquote, sigma_0, itmax, epsilon){

  it<-0

  d<- (marketquote - CDSoptionBS(D, fswap, k, sigma_0, tau))/ BSvega(D,fswap,k,sigma_0,tau)

  sigma_old<-sigma_0

  while( abs(d) > epsilon && it<itmax ){

    sigma_new<- sigma_old + d

    it<-it+1

    d<- (marketquote - CDSoptionBS(D, fswap, k, sigma_new, tau) )/ BSvega(D,fswap,k,sigma_new,tau)

    sigma_old <-sigma_new

  }

  if(it==itmax){
    print("Do not converge...")
    return(sigma_old)

  }else{
    return(sigma_old)

  }
}
```

Figure A.4: Newton's method.

A.2 Cheyette algorithm

```
/* ASSUMPTIONS:
   1) Proportional model assume elasticity parameter \gamma = 1
   2) Default intensity is defined \lambda(t) = \bar{r}(t)-r(t)
   3) Initial term structure of the forward credit spread is assumed to be flat:
        s(0,t) = lambda_0
   4) The volatility and the mean reversion term represented, respectively, by:
       vol and k_s are constants.

    INPUT: Model parameters

        double vol;  volatitily (sigma)
        double k_s;  exogenous factor or mean reversion term

        int m;  number of elements in the partition of phi
        int N;  number of steps in the tree
        double strike; strike price of the CDSwaption
        double R; recovery rate

        string dateT; maturity date of the CDSwaption

        char typeoption; call 'C' or put option 'P', respectively, payer or
            receiver CDSwaption

        double r0; initial interest rate assumed to be constant over the time
        double lambda0; initial default intensity
        double phi0; initial accumulated variance for the forward credit spread

*/

#include <iostream>
#include <cmath>
#include <vector>
#include <string>
#include<fstream>
#include<algorithm>
#include<assert.h>

using namespace std;


```

```
/* Node Declaration
 */
struct Nodes
{
        int visit; // 0 - not visit , 1 - visit
    double lambda; // default intensity
        double phi[2];// Array with min and max of phi
    double *optionPrice; // option price
    double *probUp; // up probabilities
};

int signal (double Z);

double calcJ(double mean, double sqrtt);

void buildLRStree( double vol,double k_s,int m, int N, double strike, char
    typeoption, double r0, double lambda0, double phi0,double R, const string&
    dateT, const vector<string>& dates, const vector<int>& numdate);



/*********Main Contains Menu *********/

int main()
{
    double vol=0.58, k_s=-0.1, strike=0.0625, r0=0.01, phi0=0.0, lambda0=0.06, R
        =0.4;

        string dateT= "20-12-2008";

    // number of partitions of phi and tree steps
    int m=25, N =64;

    // declare and initialize type of the option (call C ou put P)
    char typeoption='C';

    // extract the values of each column of the text file
    vector<string> dates;
    vector<int> daycount;

    ifstream theFile("dates.txt");

    string reading1;
```

```
        int reading2;


    while(theFile>> reading1>> reading2){
                dates.push_back(reading1);
        daycount.push_back(reading2);
        }

    buildLRStree( vol, k_s, m, N, strike, typeoption,  r0, lambda0, phi0, R,
        dateT, dates, daycount);

    return 0;

}

void buildLRStree( double vol,double k_s, int m, int N, double strike, char
    typeoption, double r0,
                   double lambda0, double phi0, double R,const string& dateT,
                       const vector<string>& dates,
                                    const vector<int>& numdate){


        int posT = -1; // position on the txt file of the maturity date of the
            CDSwaption

    for(unsigned int i=0; i < dates.size();i++){
        if(dates[i]==dateT)
                posT=i;
    }


    assert(posT>-1); // verify if in fact exists the maturity date on the file (
        debug)

    /*Local variables
        dt = time interval
        sqrt_dt = square root of the time interval
        */

        double dt=0.0;
        double sqrt_dt=0.0;

        dt=((numdate[posT]-numdate[0])/365.0)/N; //time steps
        cout<<"tau "<<(numdate[posT]-numdate[0])/365.0 <<endl;

        sqrt_dt= sqrt(dt);

    //Dynamically allocating memory in each node
        Nodes **tree = new Nodes *[N+1];

        for(int i=0; i<N+1 ; i++) // dimension of each line
        tree[i] = new Nodes [i+1];

        //Inicializing the root of the tree
        tree[0][0].lambda = lambda0;
    tree[0][0].phi[0]= phi0;
    tree[0][0].phi[1]= phi0;

    for (int i = 0; i < N+1 ; i++){
        for (int j = 0; j <= i; j++){
                tree[i][j].visit=0;
                        tree[i][j].optionPrice= new double [m];
                        tree[i][j].probUp = new double [m];
        }
    }


 /*** Calculating the values of lambda, phi and probUp for each node of the tree
     , except the probUp on the terminal node (i=N) ***/

for(int i=0; i<N ;i++){
        for(int j=0; j<i+1 ;j++){

                if(tree[i][j].phi[0]==tree[i][j].phi[1]){ //edges of the lattice

                                double mean1=0.0;
                                int J1=0;

                                mean1= (k_s*(lambda0-tree[i][j].lambda) + tree[i
                                    ][j].phi[0])/(vol * tree[i][j].lambda) - (
                                    vol/2);
                            J1=calcJ(mean1, sqrt_dt);

                                if(J1!=0){
```

```
                                cout<< i << " " << j << " " << "J1
                                    change is value for " << J1 <<endl;
                                exit(0);
                        }

                        for(int k=0; k<m ;k++)
                tree[i][j].probUp[k]=(mean1*dt + (1-J1)*sqrt_dt)/(2*sqrt_dt)
                    ;

                // Insert the next values of lambda, if until the moment the
                     next nodes aren't visit

                if(tree[i+1][j].visit==0)
                        tree[i+1][j].lambda= exp(log(tree[i][j].lambda) +
                            vol*(J1+1) * sqrt_dt);

                        if(tree[i+1][j+1].visit==0)
                        tree[i+1][j+1].lambda= exp(log(tree[i][j].lambda) +
                            vol*(J1-1) * sqrt_dt);

            // Calculate the next value of phi
             double phi_next=0.0;
                phi_next= tree[i][j].phi[0] + (pow(vol,2) * pow(tree[i][j].
                    lambda,2)- 2*k_s*tree[i][j].phi[0])*dt;

                if(tree[i+1][j].visit==0){
                    tree[i+1][j].phi[0]= phi_next;
                    tree[i+1][j].phi[1]= phi_next;
                    tree[i+1][j].visit=1;
                    }else{
                        tree[i+1][j].phi[0] = min(phi_next, tree[i+1][j
                            ].phi[0] );
                        tree[i+1][j].phi[1] = max(phi_next,tree[i+1][j].
                            phi[1]);
                    }

                    if(tree[i+1][j+1].visit==0){
                        tree[i+1][j+1].phi[0]= phi_next;
                        tree[i+1][j+1].phi[1]= phi_next;
                        tree[i+1][j+1].visit=1;
                    }else{
                        tree[i+1][j+1].phi[0]= min(phi_next, tree[i+1][j
                                +1].phi[0] );
                        tree[i+1][j+1].phi[1]= max(phi_next,tree[i+1][j
                            +1].phi[1]);
                }


                } else{ //middle nodes

                        double step=0.0; // step of the partition of phi
                        step= (tree[i][j].phi[1]-tree[i][j].phi[0])/(m
                            -1);

                vector<double> partPhi; // vector with m-partition of
                    phi
                        partPhi.push_back(tree[i][j].phi[0]);

            for(int k = 1; k < m-1 ; k++)
            partPhi.push_back(partPhi[k-1] + step);
            partPhi.push_back(tree[i][j].phi[1]);

            vector<double> mean; // Now, the mean has m elements because
                 we hve m values of phi
            for(int k=0; k < m ; k++)
            mean.push_back((k_s * (lambda0 - tree[i][j].lambda) +
                partPhi[k] ) / (vol*tree[i][j].lambda) - (vol/2));

                    vector<int> J;
                    int valueJ=0;
                    int check=0;

                    for( int k=0; k<m ; k++){
                        J.push_back(calcJ(mean[k], sqrt_dt));

                                if(J[k]!=J[0]){
                                check=1;
                cout<< i << " " << j << " " << "Vector J change is value
                     in" << k << " for " << J[k]<<endl;
                                }
                        }

                if(check==0)
                        valueJ=J[0];
```

```
                                else if(check==1)
                                exit(0);


                    for(int k=0; k<m ;k++)
                    tree[i][j].probUp[k]=(mean[k]*dt + (1-J[k])*sqrt_dt)/(2*
                        sqrt_dt);

                // Insert the next values of lambda, if until the moment the
                    next nodes aren't visit
                    if( tree[i+1][j].visit==0)
                    tree[i+1][j].lambda= exp(log(tree[i][j].lambda) + vol*(
                        valueJ+1) * sqrt_dt);

                            if(tree[i+1][j+1].visit==0)
                            tree[i+1][j+1].lambda= exp(log(tree[i][j].lambda) +
                                vol*(valueJ-1)* sqrt_dt);

                                vector <double> sucPhi; // vector with the
                                    successor values of partPhi (vector with dim
                                    =m)
                    for(int k=0;k<m;k++)
                        sucPhi.push_back(partPhi[k]+ (pow(vol,2) * pow(tree[i][j
                            ].lambda,2)- 2 * k_s * partPhi[k] )*dt);

                            /*Update the successor values of phi */
                    if( tree[i+1][j].visit==1){ // By construction, we already
                        visit this node

                        double currentmin=tree[i+1][j].phi[0];
                                double currentmax=tree[i+1][j].phi[1];

                                for(int k=0;k<m;k++){
                                        if(sucPhi[k]<currentmin) //minimum value
                                            of sucPhi
                                        currentmin=sucPhi[k];

                        if(sucPhi[k]>currentmax) //maximum value of sucPhi
                                currentmax=sucPhi[k];
                                    }

                        tree[i+1][j].phi[0]= currentmin;
```

```
                                tree[i+1][j].phi[1]= currentmax;

                            }else{
                                cout<< "You are doing the lattice in a wrong way
                                    -update(i+1,j)"<< endl;
                            }

                            double minsucPhi=sucPhi[0]; //first value of sucPhi
                    double maxsucPhi=sucPhi[m-1]; //last value of sucPhi

                                 /*Calculation of the minimum and maximum values
                                     of the vector sucPhi*/

                        for(int k=0;k<m;k++){
                                         if(sucPhi[k]<minsucPhi) //minimum value
                                             of sucPhi
                        minsucPhi=sucPhi[k];
                        if(sucPhi[k]>maxsucPhi) //maximum value of sucPhi
                        maxsucPhi=sucPhi[k];
                                }

                            if(tree[i+1][j+1].visit==0){ // By construction, we
                                have't already visit this node
                            tree[i+1][j+1].phi[0]= minsucPhi;
                            tree[i+1][j+1].phi[1]= maxsucPhi;
                            tree[i+1][j+1].visit=1;
                            }else{
                        cout<< "You are doing the lattice in a wrong way-update(
                            i+1,j+1)"<< endl;
                    }

                        }

        } //close j

        } //close i


/****** Terminal nodes *****/

/*Remember that posT  is the position of the maturity date T_k of CDSwaption on
    the textfile
```

```
which coincides with the starting time of the forward CDS contract
*/
    vector<double> delta; // interval of time between payments posT=T_k,...,T_n

    for(unsigned int k=0; k < numdate.size()-1;k++){
        delta.push_back((numdate[k+1]-numdate[k])/365.0);
        }


    vector<double> survival; //survival factor S(0,T_i), for i=k,..,n-1,n

    for(unsigned int k=0; k < numdate.size();k++){
    survival.push_back(exp(-lambda0*((numdate[k]-numdate[0])/365.0)));
        }

    //Calculate the forward default swap rate at the valuation date:
    vector<double> bondPrice0; //default-free discount factor P(0,T_i+1), for i=
        k,..,n-1

    for(unsigned int k=0; k < numdate.size();k++){
      bondPrice0.push_back(exp(-r0*((numdate[k]-numdate[0])/365.0)));
        }

    vector<double> defaultBondPrice0; // default zero coupon bond bar(P)(posT,
        T_i+1), i=k,..,n-1

        for( unsigned int k=0;k < numdate.size();k++){
                defaultBondPrice0.push_back(survival[k]*bondPrice0[k]);
        }

        double num = 0.0;
        double den = 0.0;

        for(unsigned int k=posT; k < numdate.size()-1;k++)
        num+= survival[k]*bondPrice0[k+1]-defaultBondPrice0[k+1];
        int count=posT+1;
        defaultBondPrice0.erase(defaultBondPrice0.begin(),
        defaultBondPrice0.begin()+count);

        int countDelta=posT;
        delta.erase(delta.begin(),delta.begin()+countDelta);
        for(unsigned int k=0; k < delta.size();k++)
        den+= delta[k]*defaultBondPrice0[k];

    double fswaprate0=0.0;
        fswaprate0=(1-R)*(num/den);

    vector<double> bondPrice; //default-free discount factor P(posT,T_i+1), for
        i=k,..,n-1

        for(unsigned int k=0; k < numdate.size()-1;k++){
        bondPrice.push_back(exp(-r0*((numdate[k+1]-numdate[posT])/365.0)));
        }

        int count1=0;
        for( unsigned int k=0;k < bondPrice.size();k++){
                if(bondPrice[k]>=1)
                count1+=1;
        }
        bondPrice.erase(bondPrice.begin(),bondPrice.begin()+count1);




/*
***---------Loop on the terminal nodes-------***

Goal: Calculate the values of option Price in each terminal node
*/

        for(int j=0; j < N+1 ;j++){

                if(tree[N][j].phi[0]==tree[N][j].phi[1]){ //edges of the lattice
                vector<double> defaultRisk; // S(t,T) forward survival
                    probability

                    for(unsigned int k=0; k < numdate.size();k++){
                defaultRisk.push_back((survival[k]/survival[posT])*exp( ((1-exp
                    (- k_s*((numdate[k]-numdate[posT])/365.0))) /k_s)*
            (lambda0-tree[N][j].lambda)- 0.5*pow((1-exp(-k_s*((numdate[k]-
                numdate[posT])/365.0))),2)/pow(k_s,2)* tree[N][j].phi[0] ));
                        }

                int count3=0;
```

```

for( unsigned int k=0;k < defaultRisk.size();k++){
        if(defaultRisk[k]>1)
        count3+=1;
}
defaultRisk.erase(defaultRisk.begin(),defaultRisk.begin()+count3);

vector<double> defaultBondPrice; // default zero coupon bond bar(P)(posT,T_i+1), i=k,..,n-1

for( unsigned int k=0;k < bondPrice.size();k++){
        defaultBondPrice.push_back(defaultRisk[k+1]*bondPrice[k]);
}


/* Calculation of the forward swap rate \xi_{k,n} (fswaprate):

auxfswap = numerator of the forward swap rate formula
discount= denominator of the forward swap rate formula and also
          the discount factor of the payoff at T_k of the CDSoption

fswaprate = formula of the forward swap rate on remark 5.2
*/

double auxfswap = 0.0;
double discount = 0.0;

for(unsigned int k=0; k < bondPrice.size();k++){
        auxfswap+= defaultRisk[k]*bondPrice[k]-defaultBondPrice[k];
        discount+= delta[k]*defaultBondPrice[k];
}

double fswaprate=0.0;
fswaprate=(1-R)*(auxfswap/discount);

```

```

                        if(typeoption=='C'){ // if is a payer CDSwaption

                                for(unsigned int k=0; k<m ;k++)
                                tree[N][j].optionPrice[k]=discount* max(
                                    fswaprate - strike, 0.0);

                }else if(typeoption=='P'){ // if is a receiver CDSwaption

                                for(unsigned int k=0;k<m;k++)
                                tree[N][j].optionPrice[k]=discount* max( strike
                                    - fswaprate , 0.0);
                    }

            } else{ //middle nodes

                        double step=0.0;
            step=(tree[N][j].phi[1]-tree[N][j].phi[0])/(m-1);

            vector<double> partPhi;
                        partPhi.push_back(tree[N][j].phi[0]);

                    for(int k = 1; k < m-1 ; k++)
                    partPhi.push_back(partPhi[k-1] + step);
                    partPhi.push_back(tree[N][j].phi[1]);

            for(int k=0; k< m ;k++){ // for each value of the partition of phi

                                vector<double> defaultRisk;

                                for(int u=0 ; u < numdate.size();u++)
                                defaultRisk.push_back((survival[u]/survival[posT
                                    ])*exp( ((1-exp(- k_s*((numdate[u]-numdate[
                                    posT])/365.0))) /k_s)*
                (lambda0-tree[N][j].lambda)- 0.5*pow((1-exp(-k_s*((numdate[u]-
                    numdate[posT])/365.0)))/k_s,2)* partPhi[k] ));

                int count3=0;

                                for( unsigned int u=0;u < numdate.size();u++){
                                        if(defaultRisk[u]>1)
                                        count3+=1;
                          }

                          defaultRisk.erase(defaultRisk.begin(), defaultRisk.begin()+count3);

                        vector<double> defaultBondPrice;

                                for(unsigned int u=0;u<bondPrice.size();u++)
                    defaultBondPrice.push_back(defaultRisk[u+1]*bondPrice[u]);
          //

                    //Calculation of the forward swap rate:
                    double auxfswap = 0.0;
                    double discount = 0.0;

                                for(int u=0; u< bondPrice.size() ;u++){
                                    auxfswap+= defaultRisk[u]*bondPrice[u]-defaultBondPrice[u];
                                    discount+=delta[u]*defaultBondPrice[u];
                    }

                    double fswaprate=0.0;
                    fswaprate=(1-R)*(auxfswap/discount);


                if(typeoption=='C')
                tree[N][j].optionPrice[k]=discount* max( fswaprate - strike, 0.0);
                                else if (typeoption=='P')
                                tree[N][j].optionPrice[k]=discount* max( strike - fswaprate , 0.0);

                    }
               }
              }//end of loop on the terminal nodes

          /*****------------Backward recursion-----------------*****/

              for( int i=N-1; i>=0;i--){
            for( int j=0; j<=i; j++)
         {

        if(tree[i][j].phi[0]==tree[i][j].phi[1]){ //edge nodes of the lattice

                        // Calculate the next value of phi
            double phi_next=0.0;
            phi_next= tree[i][j].phi[0] + (pow(vol,2) * pow(tree[i][j].lambda,2)
                - 2*k_s*tree[i][j].phi[0])*dt;

                        if(phi_next==tree[i+1][j].phi[0] && phi_next==tree[i+1][
                            j+1].phi[1]){
                                for(int k=0;k<m;k++)
                                tree[i][j].optionPrice[k]= (tree[i][j].probUp[k
                                    ]*tree[i+1][j].optionPrice[0]+(1-tree[i][j].
                                    probUp[k])*
                                            tree[i+1][j+1].optionPrice[m-1]) *exp
                                                (-(tree[i][j].lambda+r0)*dt);
                        }else{
                                cout<< " Doing in a wrong way the Linear
                                    interpolation: edge nodes of the lattice" <<
                                    endl;

                        }


                } else{ //middle nodes

                        /*At node (i,j) we will compare the values of next
                            values of phi
                         with the values of phi in nodes (i+1,j) and (i+1,j+1)*/

                double step=0.0;
                step= (tree[i][j].phi[1]-tree[i][j].phi[0])/(m-1);

            vector<double> partPhi;
                        partPhi.push_back(tree[i][j].phi[0]);

                    for(int k = 1; k < m-1 ; k++)
                    partPhi.push_back(partPhi[k-1] + step);
                    partPhi.push_back(tree[i][j].phi[1]);

                vector <double> sucPhi; //next values of phi on node (i,j)
                for(int k=0;k<m;k++)
                        sucPhi.push_back(partPhi[k]+ (pow(vol,2) * pow(tree[i][j
                            ].lambda,2)- 2 * k_s * partPhi[k] )*dt);

                double step2=0.0;
                        step2= (tree[i+1][j].phi[1]-tree[i+1][j].phi[0])/(m-1);

                        vector<double> partPhi2; //values of the partition of
                            phi in (i+1,j)
                        partPhi2.push_back(tree[i+1][j].phi[0]);

                    for(int k = 1; k < m-1 ; k++)
                    partPhi2.push_back(partPhi2[k-1] + step2);
                    partPhi2.push_back(tree[i+1][j].phi[1]);

                    //In theory the min and max pf partPhi2 are at positions 0
                        and m-1 --> just to check
                    double minpartPhi2=partPhi2[0];
                    double maxpartPhi2=partPhi2[m-1];
                        int minpos2=0;
                        int maxpos2=m-1;

                        for(int k=0;k<m;k++){
                                if(partPhi2[k]<minpartPhi2){
                                        minpartPhi2=partPhi2[k];
                                        minpos2=k;
                                } //minimum value of partPhi2


                if(partPhi2[k]>maxpartPhi2){
                        maxpartPhi2=partPhi2[k];
                        maxpos2=k;
                                } //maximum value of partPhi2

                    }



                double step3=0.0;
                step3= (tree[i+1][j+1].phi[1]-tree[i+1][j+1].phi[0])/(m-1);

                        vector<double> partPhi3; //values of the partition of
                            phi in (i+1,j+1)
                        partPhi3.push_back(tree[i+1][j+1].phi[0]);


                for( int k = 1; k < m -1 ; k++)
                partPhi3.push_back(partPhi3[k-1] + step3);
                partPhi3.push_back(tree[i+1][j+1].phi[1]);

                //In theory, when J=0, the min and max of partPhi3 are at
                    positions 0 and m-1 --> just to check
                    double minpartPhi3=partPhi3[0];
                    double maxpartPhi3=partPhi3[m-1];
                        int minpos3=0;
                        int maxpos3=m-1;

                        for(int k=0;k<m;k++){
                                if(partPhi3[k]<minpartPhi3){
                                        minpartPhi3=partPhi3[k];
                                        minpos3=k;
                                } //minimum value of partPhi3


                if(partPhi3[k]>maxpartPhi3){
                        maxpartPhi3=partPhi3[k];
                        maxpos3=k;
                                } //maximum value of partPhi3

                    }

                /** Doing Backward recursion and Interpolation **/

                for(int k=0;k<m;k++){

                                double gup=0.0;
                double gdown=0.0;

                        if( (sucPhi[k] != minpartPhi2) && (sucPhi[k] !=
                            maxpartPhi3)){

                                if(sucPhi[k] == partPhi2[k])
                                cout<<"error: sucphi(k)=partphi2(k)"<<endl;
                                if(sucPhi[k] == partPhi3[k])
                                cout<<"error: sucphi(k)=partphi3(k)"<<endl;

                                int plus2 =0;
                                int minus2 =0;
                                for(int u=0; u<m; u++){
                                        if((sucPhi[k] > partPhi2[u]) && (sucPhi[
                                            k] < partPhi2[u+1])){
                                                minus2=u;
                                                plus2=minus2+1;
                                                }
                                    }

                                        int plus3 =0;
                                        int minus3 =0;
                                        for(int u=0; u<m; u++){
                                                if((sucPhi[k] > partPhi3[u]) &&
                                                    (sucPhi[k] < partPhi3[u+1]))
                                                    {
                                                        minus3=u;
                                                        plus3=minus3+1;
                                            }
                                    }

                                gup=tree[i+1][j].optionPrice[minus2] + (sucPhi[k
                                    ]-partPhi2[minus2])/(partPhi2[plus2]-
                                    partPhi2[minus2] ) *
                                     (tree[i+1][j].optionPrice[plus2]-tree[i+1][
                                        j].optionPrice[minus2]);

                                gdown=tree[i+1][j+1].optionPrice[minus3] + (
                                    sucPhi[k]-partPhi3[minus3])/(partPhi3[plus3
                                    ]-partPhi3[minus3] ) *
                                     (tree[i+1][j+1].optionPrice[plus3]-tree[i
                                        +1][j+1].optionPrice[minus3]);

                                        tree[i][j].optionPrice[k]=(tree[i][j].
                                            probUp[k] * gup + (1-tree[i][j].
                                            probUp[k])*gdown) * exp(-(tree[i][j
                                            ].lambda+r0)*dt);

                            }


                        if(sucPhi[k] == minpartPhi2 && sucPhi[k] != maxpartPhi3
                            && sucPhi[k] != minpartPhi3){
                            //coincide on the first midle node at i=N-1
                            if(sucPhi[k] == partPhi3[k])
                            cout<<"error: sucphi(k)=partphi3(k)"<<endl;

                                        int plus3 =0;
                                        int minus3 =0;
                                        for(int u=0; u<m; u++){
                                                if((sucPhi[k] > partPhi3[u]) &&
                                                    (sucPhi[k] < partPhi3[u+1]))
                                                    {
                                                        minus3=u;
                                                        plus3=minus3+1;
                                            }
                                    }

              gdown=tree[i+1][j+1].optionPrice[minus3] + (sucPhi[k]-partPhi3[
                  minus3])/(partPhi3[plus3]-partPhi3[minus3] ) *
                                (tree[i+1][j+1].optionPrice[plus3]-tree[i+1][j
                                    +1].optionPrice[minus3]);

                        tree[i][j].optionPrice[k]=(tree[i][j].probUp[k]*tree[i
                            +1][j].optionPrice[minpos2]+
                                 (1-tree[i][j].probUp[k])*gdown)*exp(-(tree[i][
                                     j].lambda+r0)*dt);

                            }

                            if(sucPhi[k] == maxpartPhi3 && sucPhi[k] !=
                                minpartPhi2 && sucPhi[k] != maxpartPhi2){
                                //coincide for the last edge node at i=N-1
                                if(sucPhi[k] == partPhi2[k])
                            cout<<"error: sucphi(k)=partphi2(k)"<<endl;

                            int plus2 =0;
                        int minus2 =0;
                                for(int u=0; u<m; u++){
                                        if((sucPhi[k] > partPhi2[u]) && (sucPhi[
                                            k] < partPhi2[u+1])){
                                                minus2=u;
                                                plus2=minus2+1;
                    }
                            }
```

```


                            gup=tree[i+1][j].optionPrice[minus2] + (sucPhi[k]-
                                partPhi2[minus2])/(partPhi2[plus2]-partPhi2[
                                minus2] ) *
                                (tree[i+1][j].optionPrice[plus2]-tree[i+1][j].
                                    optionPrice[minus2]);

                tree[i][j].optionPrice[k]=(tree[i][j].probUp[k]*gup +
                                 (1-tree[i][j].probUp[k])*tree[i+1][j+1].
                                     optionPrice[maxpos3] ) * exp(-(tree[i][j].
                                     lambda+r0)*dt);

                            }

                            if( (sucPhi[k] == minpartPhi2) && (sucPhi[k] ==
                                maxpartPhi3)){
                                cout<< "erro1"<<endl;
                                }
                                 if( sucPhi[k] == maxpartPhi2){
                                cout<< "erro2"<<endl;
                                }
                                if( sucPhi[k] == minpartPhi3){
                                cout<< "erro3"<<endl;
                                }

                }
          }
        }
        }


        for( int i=0 ; i<=N; i++){
                for( int j=0;j<=i;j++){
//                         cout<< i <<" "<< j << " "<<" dspread "<< tree[i][j].
    lambda<< endl;
//               cout <<i<<" "<< j << " "<<" phi0 "<<  tree[i][j].phi[0]<< endl;
//              cout <<i<<" "<< j << " "<<" phi1 "<< tree[i][j].phi[1]<< endl;
//
               for(int k=0;k<m;k++){
           //   cout <<i<<" "<< j << " prob "<< tree[i][j].probUp[k]<< endl;
                //cout <<i<<" "<< j << " "<< "optionPrice "<<tree[i][j].
```

```
                    optionPrice[k]<< endl;
                  }

          }
      }

         for(int k=0;k<m;k++){
          //cout <<" "<<  " prob1 "<< tree[99][0].probUp[k]<< endl;
          //cout <<" "<< " prob2 "<< tree[99][1].probUp[k]<< endl;
          //cout<< " " << "probability " << tree[2][1].probUp[k];
          //cout << "(2,1) "<< "optionPrice "<<tree[2][1].optionPrice[k]<< endl;
        }
        for(int k=0;k<m;k++ ){
          //cout << "(3,2) "<< "optionPrice "<<tree[3][2].optionPrice[k]<< endl;
          cout << " "<< "optionPrice "<<tree[0][0].optionPrice[k]<< endl;

             }



  //return tree[0][0].optionPrice[0];
}

/* Auxilary Functions: Z and calcJ -----> for calculating J (LRS paper)*/
int signal ( double Z){

        int result;

        if(Z>0){
        result=1;
        } else if(Z<0)
        {
                result=-1;
        } else if( Z==0)
        {
                result=0;
        }

        return result;

        }


```

```

double calcJ(double mean, double sqrtt)
{
        int Z;
        double J;

        Z= floor(mean*sqrtt);

        if ( Z % 2 == 0)
        J=signal(Z)*Z;

        else
        J=signal(Z)*abs(Z) +1;


        return J;

}
```

Appendix B

Relevant Data

B.1 Some tables with initial term-structures

Dates	$P(0,T)$	$\lambda_0 = 0.5\%$		$\lambda_0 = 1\%$		$\lambda_0 = 1.5\%$		$\lambda_0 = 4\%$	
		$S(0,T)$	$\bar{P}(0,T)$	$S(0,T)$	$\bar{P}(0,T)$	$S(0,T)$	$\bar{P}(0,T)$	$S(0,T)$	$\bar{P}(0,T)$
17-10-2008	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
20-12-2008	0.99825	0.99912	0.99475	0.99825	0.99650	0.99737	0.99563	0.99301	0.99127
20-03-2009	0.99579	0.99789	0.98742	0.99579	0.99160	0.99369	0.98951	0.98327	0.97913
20-06-2009	0.99328	0.99664	0.97998	0.99328	0.98661	0.98994	0.98329	0.97340	0.96686
20-09-2009	0.99078	0.99538	0.97260	0.99078	0.98165	0.98621	0.97712	0.96364	0.95475
20-12-2009	0.98832	0.99414	0.96535	0.98832	0.97677	0.98252	0.97104	0.95407	0.94293
20-03-2010	0.98588	0.99292	0.95824	0.98588	0.97196	0.97890	0.96508	0.94471	0.93137
20-06-2010	0.98340	0.99167	0.95102	0.98340	0.96708	0.97520	0.95901	0.93523	0.91971
20-09-2010	0.98092	0.99042	0.94386	0.98092	0.96221	0.97152	0.95299	0.92585	0.90819
20-12-2010	0.97848	0.98918	0.93682	0.97848	0.95743	0.96790	0.94707	0.91667	0.89694
20-03-2011	0.97607	0.98796	0.92992	0.97607	0.95272	0.96432	0.94125	0.90767	0.88595
20-06-2011	0.97362	0.98672	0.92291	0.97362	0.94793	0.96068	0.93534	0.89856	0.87485
20-09-2011	0.97116	0.98548	0.91596	0.97116	0.94316	0.95706	0.92946	0.88955	0.86390
20-12-2011	0.96875	0.98425	0.90914	0.96875	0.93847	0.95349	0.92369	0.88072	0.85320
20-03-2012	0.96633	0.98302	0.90236	0.96633	0.93380	0.94993	0.91795	0.87198	0.84263
20-06-2012	0.96390	0.98178	0.89556	0.96390	0.92910	0.94634	0.91218	0.86324	0.83207
20-09-2012	0.96147	0.98055	0.88882	0.96147	0.92443	0.94277	0.90645	0.85458	0.82165
20-12-2012	0.95908	0.97933	0.88220	0.95908	0.91983	0.93925	0.90082	0.84610	0.81147
20-03-2013	0.95672	0.97812	0.87569	0.95672	0.91531	0.93579	0.89528	0.83779	0.80153
20-06-2013	0.95431	0.97689	0.86910	0.95431	0.91071	0.93225	0.88966	0.82939	0.79149
20-09-2013	0.95191	0.97566	0.86255	0.95191	0.90613	0.92874	0.88407	0.82107	0.78158
20-12-2013	0.94954	0.97444	0.85612	0.94954	0.90162	0.92527	0.87858	0.81292	0.77190

Table B.1: Default-free and defaultable zero coupon bonds (respectively, $P(0,T)$ and $\bar{P}(0,T)$)) and survival probabilities ($S(0,T)$), at the valuation date 17-10-2008 for several future dates with λ_0 equals to 0.5%, 1%, 1.5% and 4%.

Dates	$P(0,T)$	$\lambda_0 = 6\%$		$\lambda_0 = 8\%$	
		$S(0,T)$	$\bar{P}(0,T)$	$S(0,T)$	$\bar{P}(0,T)$
17-10-2008	1.00000	1.00000	1.00000	1.00000	1.00000
20-12-2008	0.99825	0.98954	0.98780	0.98607	0.98434
20-03-2009	0.99579	0.97500	0.97090	0.96681	0.96274
20-06-2009	0.99328	0.96037	0.95392	0.94751	0.94115
20-09-2009	0.99078	0.94595	0.93723	0.92860	0.92004
20-12-2009	0.98832	0.93191	0.92102	0.91026	0.89962
20-03-2010	0.98588	0.91822	0.90526	0.89248	0.87988
20-06-2010	0.98340	0.90444	0.88943	0.87466	0.86014
20-09-2010	0.98092	0.89087	0.87387	0.85720	0.84085
20-12-2010	0.97848	0.87764	0.85875	0.84027	0.82219
20-03-2011	0.97607	0.86475	0.84406	0.82386	0.80415
20-06-2011	0.97362	0.85177	0.82930	0.80742	0.78611
20-09-2011	0.97116	0.83899	0.81479	0.79130	0.76848
20-12-2011	0.96875	0.82653	0.80070	0.77567	0.75143
20-03-2012	0.96633	0.81426	0.78684	0.76035	0.73475
20-06-2012	0.96390	0.80204	0.77308	0.74517	0.71827
20-09-2012	0.96147	0.79000	0.75956	0.73030	0.70216
20-12-2012	0.95908	0.77827	0.74642	0.71588	0.68658
20-03-2013	0.95672	0.76684	0.73365	0.70189	0.67152
20-06-2013	0.95431	0.75533	0.72082	0.68788	0.65645
20-09-2013	0.95191	0.74399	0.70821	0.67415	0.64173
20-12-2013	0.94954	0.73295	0.69596	0.66084	0.62749

Table B.2: Default-free and defaultable zero coupon bonds (respectively, $P(0,T)$ and $\bar{P}(0,T)$)) and survival probabilities ($S(0,T)$), at the valuation date 17-10-2008 for several future dates with λ_0 equals to 6% and 8%.