POLITECNICO DI MILANO

MASTER THESIS

Interest rate derivative pricing and calibration in multicurve extension models

Author:
Stefano GEROSA

Supervisors:
Prof. Roberto BAVIERA

A thesis submitted in fulfillment of the requirements for the degree of Mathematical Engeneering

in the

Industrial and Information Engineering
Department of Mathematics

3 October 2018

POLITECNICO DI MILANO

Abstract

Industrial and Information Engineering
Department of Mathematics

Mathematical Engeneering

Interest rate derivative pricing and calibration in multicurve extension models

by Stefano GEROSA

The recent financial crisis has driven practitioners to elaborate multicurve models for term structure. Here we study two multicurve extensions on interest rate model: a parsimonious multicurve three parameters extension of the well-known Hull & White model and a multicurve seven parameters extension of a three Gaussian factors exponentially quadratic short rate model. These models allow obtaining closed exact formulas for main vanilla derivatives on interest rate, in particular here we concentrate on linear swap and non-linear swaption interest rate derivatives. Calibration for both models involves initial discount and pseudo-discount curves construction, about that Mr. Crab Bootstrap methodology is described. Calibration issues are discussed in details, moreover Matlab codes are provided and performance issues are addressed.

Acknowledgements

I would like to thank Professor Baviera for giving me the opportunity to do this thesis work and for granting me part of his time supervising the work. In particular i want to dedicate this thesis to my family, friends and colleagues whose support has been crucial during these years of studies.
Thank you.

Contents

Abstract **i**

Acknowledgements **ii**

Contents **iii**

Introduction **1**

1 Dual Curve Construction **3**
- 1.1 Discounting curve . . . 3
- 1.2 Pseudo-discounting curve . . . 4
- 1.3 Numerical Results . . . 6

2 A Multicurve Gaussian HJM Model **8**
- 2.1 Basic Financial Quantities . . . 8
- 2.2 The model . . . 9
- 2.3 Interest Rate Swap pricing . . . 10
- 2.4 European Swaption pricing . . . 10

3 An Exponentially Quadratic Model **12**
- 3.1 Martingale Measure . . . 12
- 3.2 The exponentially quadratic short rate model . . . 13
- 3.3 Bond prices (discount and pseudo-discount) . . . 13
- 3.4 Forward measure . . . 16
- 3.5 Interest rate swap pricing . . . 17
- 3.6 European swaption pricing . . . 21
- 3.7 Monte Carlo formula . . . 25
- 3.8 Numerical Results . . . 26

4 Calibration **29**
- 4.1 Methodology . . . 29
- 4.2 Numerical Results . . . 30
- 4.3 Alternative EQM model parameters calibration . . . 32

5 Matlab Code's Critical Issues **34**
- 5.1 Swaption pricing through Jamshidian approach . . . 34
- 5.2 Swaption pricing through triple integration . . . 37
- 5.3 Calibration . . . 38

Conclusion **40**

A ODE solutions **42**
- A.1 Section 3.3 . . . 42
- A.2 Section 3.5 . . . 43

B Matlab Code **45**
B.1 Crab's Bootstrap 45
B.2 Calibration and swaption pricing via MHM model 50
B.3 Calibration and swaption pricing via EQM model 54
B.4 EQM swaption pricer via Monte Carlo 62

Introduction

In European markets underlying rates of interest rate derivatives are typically Euribor or Libor, these rates are determined by a group of banks and thus they reflect counterparty and liquidity risk in the interbank market.
The recent crisis has pointed out this particular sort of risks and has shown that the standard (pre-crisis) non-arbitrage relation between Euribor/Libor rates of different maturities does not hold anymore while significant spreads between rates with different tenor have been observed, in particular between 'risky' rates and risk-free ones such as Overnight Indexed Swap rates.
This scenario has led to the extension of the pre-crisis single curve framework to a multicurve setting where future cash flows are no more generated via discounting curves, but thought curves associated to underlying rates (as many curves for each tenor structure considered) while the discounting curves sees its role reduced precisely to discount future cash flows.
In this thesis two multicurve extensions of existing single curve interest rate models are considered, for both of them we give closed exact pricing formulas of two of the main derivatives on interest rates: swap and swaption.
The former is a contract that actually allows swapping at certain predefined dates a floating rate with a fixed one while the latter gives the opportunity at a predefined maturity date to enter into a swap contract.
Besides, we propose a calibration algorithm, a fundamental practice that allows these theoretical model frameworks to be adapted to the market conditions in order to be then properly used by practitioners.
The most important contributions in this thesis are:

- corrections of the theoretical pricing methodology for swap and swaption contracts under the Exponentially Quadratic short rate model described in K. Glau et al. (2016);
- extension of the pricing formulas to contracts whose fixed and floating legs' frequencies are no longer equal;
- addition of a fundamental hypothesis to obtain swaption closed formulas;
- proposal of a cascade calibration methodology to calibrate this model;
- implementation of a Matlab library to price swaptions with closed formula and Monte Carlo algorithm and to calibrate the model.

More in details the thesis is structured as follows. In *Chapter 1*, it is presented the bootstrap extension to a multicurve framework, a necessary first step to the calibration cascade. The entire methodology is borrowed from previous work in the field by Baviera and Cassaro (2015) and it is implemented on Matlab and the results are shown in the last section of the chapter, the dataset used is available and can be accessed on *www.mate.polimi.it/qfinlab/baviera/data/MarketData_Crab.xls*.
In *Chapter 2* it is addressed the multicurve extension of the Hull and White model, a parsimonious three parameters model, we present swap and swaptions pricing

formula following the derivation explained in in the work of Baviera (2017), the formulas are then implemented in Matlab for later use during calibration algorithm.
The thesis *Chapter 3* represents the core of the work: starting from the multicurve Exponentially Quadratic Model suggested by K. Glau et al. (2016) swap and swaption closed exact formulas are corrected and then generalized relaxing hypothesis that gives the formulas a better look despite their applicability in practice. A Monte Carlo relative formula for swaptions is derived in order to validate the new closed exact formula obtained, the whole is implemented and tested in Matlab. The model in question is a seven parameters three Gaussian factors short rate model belonging to the Exponentially Quadratic class.
In *Chapter 4* the calibration cascade issue is addressed and its methodology explained, given a set of data on swaptions volatilities, results of calibration on both models are shown and discussed. Besides it will be only presented an alternative methodology calibration for the Exponentially Quadratic multicurve model.
Finally, *Chapter 5* discusses upon issues met during code writing on Matlab about code time performance, improvements are given and described, some of them involving the Parallel Computing Toolbox of Matlab.
In *Appendix A* we report some math deductions while the most relevant Matlab scripts and functions are reported in *Appendix B*.

Chapter 1

Dual Curve Construction

In this first chapter, the goal is to describe the algorithm, known as *Curve Construction*, used in order to obtain discount factors starting from the most liquid quoted instruments in the interbank market, such as:

- cash deposits (from here on depos);
- Forward Rate Agreement (FRA);
- Interest Rate Swap (swap).

Before the 2007 crisis, the algorithm used was a forward-looking iterative one, in the sense that in order to obtain the next curve's knot only information from previous knots was used, besides just one curve was constructed and it was used for both discounting and computing forward rates. Instead, it is now commonly accepted by practitioners to consider a dual curve framework, involving a *discounting curve*, used as the name suggests to discount future cash flows and a *pseudo-discounting curve* used to compute forward rates. It is important to underline that a different pseudo-discounting curve is needed for each tenor considered due to credit risk related to borrowing/lending at various time horizons.
The Bootstrap algorithm is not unique, the one here described is a backward-forward-looking iterative algorithm, hence the name *Crab's Bootstrap*, in fact like the crab on the sand moves forward, backward and then forward again we need to do some backward step when constructing the pseudo-discounting curve due to the fact that we must consider only instruments with the same tenor.
In the context of this thesis, we consider only a six-month tenor Euribor rate, that's the reason why our bootstrap construction does not involve Short-Term Interest Rate Futures that does not exist with six months Euribor as underlying.

1.1 Discounting curve

First of all we indicate with $B(t_0, t_i)$ the discounting curve with start date t_0 (settlement date) and end date t_i, the curve knots are the first three weeks, every month till one year, every quarter form first to second year, every year form two up to thirty years. As it is standard in the market (see e.g. Henrard (2010)) we choose the EONIA curve as the discounting curve, the bootstrap technique is standard and takes into consideration Overnight Indexed Swap (from here on OIS) rates (indicated with $R_{OIS}(t_0, t_i)$) with increasing maturities $(t_i - t_0)$.
The procedure is simple and can be summarized in two steps, in the first one we consider OIS with maturities shorter than one year:

$$B(t_0, t_i) = \frac{1}{1 + \delta(t_0, t_i) R_{OIS}(t_0, t_i)}$$

where $\delta(t_0, t_i)$ represents the year fraction corrisponding to the lag (t_0, t_i) evaluated with ACT/360 convention.
The second step formula, for maturities greater than one year, is obtained imposing Net Present Value (hereinafter NPV) of the future swap cash flows to zero and then inverting the formula, getting:

$$B(t_0, t_i) = \frac{1 - R_{OIS}(t_0, t_i) \sum_{k=1}^{i-1} \delta_i B(t_0, t_k)}{1 + \delta_i R_{OIS}(t_0, t_i)}$$

where $\delta_i = \delta(t_0, t_i)$.

1.2 Pseudo-discounting curve

This part of the algorithm is the one that truly defines and gives the name to the procedure, we consider the six months Euribor curve as pseudo-discounting curve, indicated with $\bar{B}(t_0, t_i)$, and we compute the curve on the following curve knots: each month up to six months, each semester up to two years, annually up to thirty years.
The forward-backward iterative methodology can be split into five steps:

1. Compute the six months pseudo-discount using six months depos rates ($L(t_0, t_6)$):

$$\bar{B}(t_0, t_6) = \frac{1}{1 + \delta(t_0, t_6) L(t_0, t_6)}$$

   where the year fraction convention is ACT/360.

2. Consider the 6x12 FRA ($F(t_0; t_6, t_{12})$) and compute the forward pseudo discount:

$$\bar{B}(t_0; t_6, t_{12}) = \frac{1}{1 + \delta(t_6, t_{12}) F(t_0; t_6, t_{12})} \tag{1.1}$$

   where for FRA the year fraction convention is always ACT/360.
   Then inverting the definition of forward pseudo discount, moving forward and computing the one year pseudo discount:

$$\bar{B}(t_0, t_{12}) = \bar{B}(t_0, t_6) \bar{B}(t_0; t_6, t_{12}) \tag{1.2}$$

3. Now it comes the backward step, in fact we have to conveniently use equation (1.1) to obtain $\bar{B}(t_0; t_i, t_{i+6})$ with $i = 1, 2, , .., 5$ from 1x7, 2x8,.., 5x11 FRA rates. Then we must obtain $\bar{B}(t_0, t_{i+6})$ with $i = 1, 2, , .., 5$ via linear interpolation on zero rates using six months and one year pseudo discount factors and finally use:

$$\bar{B}(t_0, t_i) = \frac{\bar{B}(t_0, t_{i+6})}{\bar{B}(t_0; t_i, t_{i+6})}$$

   with $i = 1, 2, , .., 5$.

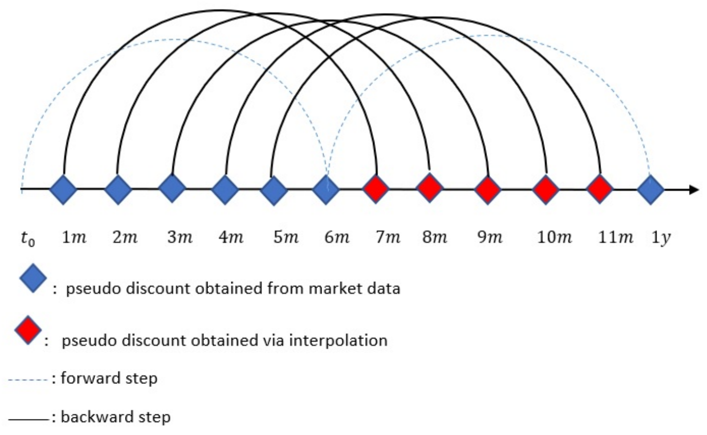
t_0
$1m$
$2m$
$3m$
$4m$
$5m$
$6m$
$7m$
$8m$
$9m$
$10m$
$11m$
$1y$
: pseudo discount obtained from market data
: pseudo discount obtained via interpolation
: forward step
: backward step

FIGURE 1.1: Forward and backward steps in *Mr Crab's Bootstrap*

4. Compute the first two values of $F(t_0; t_0, t_6)$ and $F(t_0; t_6, t_{12})$ forward rates interpolating the pseudo-discounts on curve knots, while $F(t_0; t_{12}, t_{18})$ is found imposing NPV of the 18 months swap to zero:

$$F(t_0; t_{12}, t_{18}) = \frac{I(1) - \delta(t_0, t_6)B(t_0, t_6)F(t_0; t_0, t_6) - \delta(t_6, t_{12})B(t_6, t_{12})F(t_0; t_6, t_{12})}{\delta(t_{12}, t_{18})B(t_{12}, t_{18})}$$

   where the year fraction convention is ACT/360 and

$$I(1) = S(t_0, t_{18})(\delta(t_0, t_6)B(t_0, t_6) + \delta(t_6, t_{18})B(t_6, t_{18}))$$

   with 30/360 annual convention and $S(t_0, t_{18})$ represents the quoted 18m swap rate.
   Furhtermore we have:

$$F(t_0; t_{18}, t_{24}) = \frac{I(2) - I(1)}{\delta(t_{18}, t_{24})B(t_{18}, t_{24})}$$

   where $I(2) = S(t_0, t_2) \sum_{k=1}^{2} \delta_k B(t_0, t_k)$ with $S(t_0, t_2)$ the two years swap rate.

5. finally we must take into consideration quoted swap rates starting from the third year, we want to use coherently formula (1.2) where $\bar{B}(t_0, t_{i-1})$ has already been obtained in previous steps while the forward pseudo discount factors have to be computed in this way:

$$\bar{B}(t_0; t_{i-1}, t_i) = \prod_{k \in Y(i)} \frac{1}{1 + \delta_k F(t_0; t_{k-1}, t_k)} \quad i > 2$$

   where $k \in Y(i)$ indicates six months forward rates in the i-th year, these two values have to be found via interpolation, in this case, the interpolation rule we choose to select is a log-linear one on pseudo discounts.
   This final step is the only one involving a numerical solver method due to the fact that we do not have the final pseudo discount value in order to perform the interpolation, so we need to find it imposing the corresponding swap NPV to zero.

Before showing the numerical results we must point out that whatever interpolation rule is chosen (linear on zero rates or log-linear on pseudo discounts) the year fraction has to be taken with ACT/365 convention.

1.3 Numerical Results

In this section, our aim is to show numerical results of the methodology described in the previous sections using EURO interbank market data on 13 September 2012. In the following tables are shown the actual data we used in the implementation of the dual curve Bootstrap algorithm described.

	OIS rate (%)	**Swap rate vs 6m (%)**
1w	0,105	-
2w	0,106	-
3w	0,105	-
1m	0,102	-
2m	0,096	-
3m	0,089	-
4m	0,082	-
5m	0,076	-
6m	0,074	-
7m	0,073	-
8m	0,073	-
9m	0,074	-
10m	0,076	-
11m	0,078	-
12m	0,079	0,465
15m	0,088	-
18m	0,103	0,473
21m	0,122	-
2y	0,146	0,513
3y	0,277	0,632
4y	0,463	
5y	0,676	0,813
6y	0,883	1,021
7y	1,074	1,225
8y	1,242	1,410
9y	1,390	1,573
10y	1,520	1,714
11y	1,638	1,838
12y	1,741	2,050
15y	1,963	2,253
20y	2,089	2,356
25y	2,141	2,390
30y	2,177	2,412

TABLE 1.1: OIS rates and swap rates vs EURIBOR 6m end of the day mid quotes on 13/9/12.
In this table only real rates are inserted, in order to find annually rates we choose to use a cubic spline interpolation rule.

	rate (%)
6m Depo	0,493
1x7 FRA	0,459
2x8 FRA	0,444
3x9 FRA	0,431
4x10 FRA	0,425
5x11 FRA	0,425
6x12 FRA	0,424

TABLE 1.2: Depo's and FRAs' rates end of the day mid quotes on 13/9/12.

In Figure 1.2 below we plot the zero rates curves corresponding to discounting and pseudo-discounting curves when consider EONIA as discounting curve and EURIBOR 6m as underlying for the pseudo-discounting one. As we expected due to credit and counterparty risk the pseudo discounting curve's zero rates are higher than the respective discounting rates.

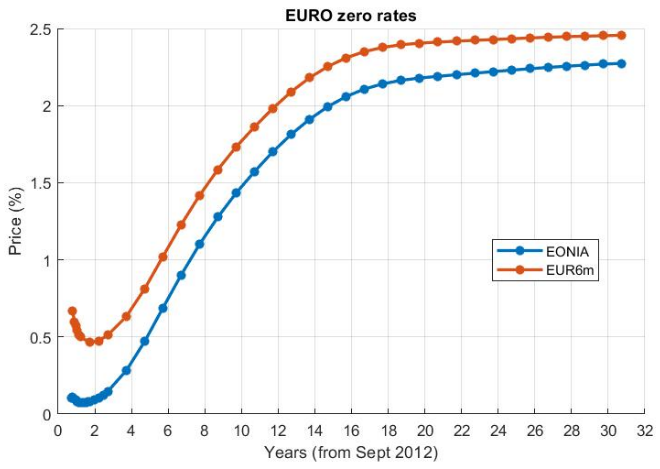

FIGURE 1.2: Zero rates discount and pseudo-discount curves on 13 September 2012

Chapter 2

A Multicurve Gaussian HJM Model

In this chapter, we aim to introduce and discuss a Multicurve Gaussian HJM Model (hereinafter MHJM model), a three-parameters multicurve extension of the well-known Hull and White model (1990).
The multicurve perspective of this model is justified by the 2007 crisis, that has shown large spreads in interbank markets like EUR or USD, moreover, the parameters' parsimony is a crucial feature considering that in today markets volumes on exotic derivatives and liquidity have decreased even on plain vanilla instruments.
In the following section we will introduce and discuss the financial quantities on which the model is based, the model itself and the pricing of some of the main vanilla Interest Rate (from here on IR) options; all the results are borrowed from previous literature work by Baviera (2017), hence we remind to this specific article for main results' proofs.

2.1 Basic Financial Quantities

Since our main goal is to price swaptions in this section we recall some key relations between main quantities on which our model and swaptions' pricing are constructed.
Let's start introducing $(\Omega, \mathscr{F}, \mathbb{P})$ a complete filtered probability space under which our framework will work, in particular given a value date t_0 $\mathscr{F}$ represents a collection of filtration $\mathscr{F}_t$ with $t \geq t_0$ and $\mathbb{P}$ is the objective probabilistic measure.
As introduced in the previous chapter let $B(t,T)$ and $\bar{B}(t,T)$ be the discounting and pseudo-discounting curve with $t_0 \leq t \leq T$ while $D(t,T)$ will represent the stochastic discount factor. From here on with $\mathbb{E}_t$ we will indicate the expected value given $\mathscr{F}_t$ under the martingale measure $\mathbb{Q}$. In line with the first chapter let $L(T, T+\Delta)$ be the Libor rate between lag Δ (in our contest we will consider EURIBOR 6 months, i.e. $\Delta = 6m$).
Finally we can define the discounting curves forward spread as

$$\beta(t; T, T+\Delta) = \frac{B(t; T, T+\Delta)}{\bar{B}(t; T, T+\Delta)} \tag{2.1}$$

The only property curves' spread has to satisfied is the T-forward martingality property:

$$B(t,T)\beta(t; T, T+\Delta) = \mathbb{E}_t[D(t,T)\beta(T, T+\Delta)] \tag{2.2}$$

2.2 The model

MHJM model is constructed upon forward discount and spread curves dynamics taking into consideration that the first one must be a martingale under t_α-forward measure while the second under t_i-forward measure:

$$\begin{cases} dB(t;t_\alpha,t_i) = -B(t;t_\alpha,t_i)[\boldsymbol{\sigma}(t,t_i)-\boldsymbol{\sigma}(t,t_\alpha)]\cdot[d\mathbf{W}_t+\boldsymbol{\rho}\boldsymbol{\sigma}(t,t_\alpha)dt],\ t\in[t_0,t_\alpha] \\ d\beta(t;t_i,t_{i+1}) = \beta(t;t_i,t_{i+1})[\boldsymbol{\eta}(t,t_{i+1})-\boldsymbol{\eta}(t,t_i)]\cdot[d\mathbf{W}_t+\boldsymbol{\rho}\boldsymbol{\eta}(t,t_i)dt],\ t\in[t_0,t_i] \end{cases} \tag{2.3}$$

where $\boldsymbol{\sigma}(t,T)$ and $\boldsymbol{\eta}(t,T)$ are d-dimensional vectors of deterministic functions of time such that $\boldsymbol{\sigma}(t,t)=\boldsymbol{\eta}(t,t)=0$, $\mathbf{W}_t$ is a d-dimensional Brownian motion under Q with covariance matrix $\boldsymbol{\rho}$.
Once observe that $d\mathbf{W}^i_t = d\mathbf{W}_t+\boldsymbol{\rho}\boldsymbol{\sigma}(t,t_i)dt$ is a Brownian motion under t_i-forward measure the martingale properties are trivial. The dynamics of the forward pseudo-discounting curve can be derived applying Ito's Lemma to the equation in (2.1) and using dynamics in (2.3) (here the dynamic is expressed under the t_i-forward measure):

$$d\bar{B}(t;t_i,t_{i+1}) = -\bar{B}(t;t_i,t_{i+1})[\boldsymbol{\sigma}_i(t)+\boldsymbol{\eta}_i(t)]\cdot[d\mathbf{W}^i_t-\boldsymbol{\rho}\boldsymbol{\eta}_i(t)dt],\ t\in[t_0,t_i]$$

where $\boldsymbol{\sigma}_i(t)=\boldsymbol{\sigma}(t,t_{i+1})-\boldsymbol{\sigma}(t,t_i)$ and $\boldsymbol{\eta}_i(t)=\boldsymbol{\eta}(t,t_{i+1})-\boldsymbol{\eta}(t,t_i)$. Furthermore it can be observed that the pseudo discounts volatilities are the sum between discounts and spread volatilities.
In the following we will consider a one dimensional version of this framework model with Hull-White-like volatilities:

$$\begin{cases} \sigma(t,T) = (1-\gamma)v(t,T) \\ \eta(t,T) = \gamma v(t,T) \\ v(t,T) = \sigma\frac{1-e^{-a(T-t)}}{a}\mathbb{I}(a>0)+\sigma(T-t)\mathbb{I}(a=0) \end{cases} \tag{2.4}$$

where $a,\sigma\in\mathbb{R}^+$ and $\gamma\in[0,1]$ are the three model parameters while $\mathbb{I}(x)$ represents an indicator function that returns one if x is true otherwise zero.
In view of the following lemma it occours to introduce some shorthands to lighten the notations otherwise too heavy:

$$\begin{cases} v_{\alpha,i} = v(t_\alpha,t_i) \\ \lambda_{\alpha,i} = (1-\gamma)v_{\alpha,i} \\ \nu_{\alpha,i} = \lambda_{\alpha,i}-(\eta(t_\alpha,t_{i+1})-\eta(t_\alpha,t_i)) \end{cases} \tag{2.5}$$

Using (2.4) and the other relations in (2.5) one can write $\nu_{\alpha,i}=v_{\alpha,i+1}(\gamma_i-\gamma)$ where $\gamma_i=\frac{v_{\alpha,i}}{v_{\alpha,i+1}}\in(0,1)$, then it is easy to observe that while $v_{\alpha,i}$ and $\lambda_{\alpha,i}$ are strictly positive and t_i time increasing and so actually volatilities, $\nu_{\alpha,i}$ could potentially be negative depending on γ, we call them extended volatilities.
We are now ready to state the following lemma that is simply an application of Ito's formula to dynamics (2.3) using definitions in (2.4) and (2.5):

Lemma 1 *Let $\xi=\int_{t_0}^{t_\alpha}e^{-a(t_\alpha-u)}dW^a_u$ a Gaussian random variable with zero mean and variance $V^2=\frac{1-e^{-2a(t_\alpha-t_0)}}{2a}\mathbb{I}(a>0)+(t_\alpha-t_0)\mathbb{I}(a=0)$ then, according to MHJM model (2.3)* *and curves' initial condition in t_0, discount and spread curve in t_α could be expressed as:*

$$\begin{cases} B(t_\alpha, t_i) = B(t_0; t_\alpha, t_i) e^{-\lambda_{\alpha,i}\xi - \frac{\lambda^2_{\alpha,i} V^2}{2}} \\ \beta(t_\alpha; t_i, t_{i+1}) B(t_\alpha, t_i) = \beta(t_0, t_i, t_{i+1}) B(t_0; t_\alpha, t_i) e^{-\nu_{\alpha,i}\xi - \frac{\nu^2_{\alpha,i} V^2}{2}} \end{cases}$$

2.3 Interest Rate Swap pricing

First of all we give the definition of a receiver IR swap: given two collections of dates $\{t'_i\}_{i=\alpha+1,..,\omega}$ and $\{t_i\}_{k=\alpha+1,..,\omega}$, a swap starting in t_α is a financial contract where a stream of interest payments on the notional is received at a fixed rate R at the dates t_i in exchange for paying, at the dates t'_i, an analogous stream corresponding to the Libor rate. A payer swap is a perfectly symmetric contract where the fixed rate is paid but in this chapter, we concentrate only on the receiver one.
Let us introduce additional shorthands notation:

$$\begin{cases} B_{\alpha,i}(t) = B(t; t_\alpha, t_i) \\ B_{\alpha',i}(t) = B(t; t_\alpha, t'_i) \\ \beta_i(t) = \beta(t; t'_i, t'_{i+1}) \\ \delta_i = \delta(t_i, t_{i+1}) \\ \delta'_i = \delta(t'_i, t'_{i+1}) \\ c_i = \delta_{i-1} R + \mathbb{I}(i = \omega) \end{cases}$$

where the year fractions convention with respect to fix leg is 30/360 annual European while with respect to floating leg is ACT/360.
The price of a receiver swap in $t \in (t_0, t_\alpha)$ starting in t_α with strike rate R is the difference between the NPV of the fixed leg and the NPV of the floating leg.
Since the price derivation is dealt with in details in the work by Baviera (2017), here we only report the final formula

$$P^{rsw}(t; t_\alpha, \{t'_i\}, \{t_i\}, R)) = R \sum_{i=\alpha}^{\omega-1} \delta_i B_{\alpha,i+1}(t) - B(t, t_\alpha) + B(t, t_\omega) - \sum_{i=\alpha}^{\omega-1} B(t, t'_i)[\beta_i(t) - 1] \tag{2.6}$$

2.4 European Swaption pricing

We start recalling that an european swaption is an option to enter at a time t'_α into a swap starting in t_α, we consider here the receiver swap described in previous section and we will assume that $t'_\alpha \equiv t_\alpha$.
By definition the payoff of such contract must be $max(P^{rsw}(t_\alpha; t_\alpha, \{t'_i\}, \{t_i\}, R), 0)$ and then its price has to be with respect to t_α-forward measure:

$$\begin{aligned} P^{rswn}(t_0; t_\alpha, \{t'_i\}, \{t_i\}, R) &= \mathbb{E}_{t_0}[D(t_0, t_\alpha) max(P^{rsw}(t_\alpha; t_\alpha, \{t'_i\}, \{t_i\}, R), 0)] \\ &= B(t_0, t_\alpha) \mathbb{E}^\alpha_{t_0}[max(P^{rsw}(t_\alpha; t_\alpha, \{t'_i\}, \{t_i\}, R), 0)] \\ &= B(t_0, t_\alpha) \mathbb{E}^\alpha_{t_0}[PO^{rswn}(t_\alpha)] \end{aligned} \tag{2.7}$$

where in the last equality $PO^{rswn}(t_\alpha)$ stands for receiver swaption payoff evaluated at option's expiry.

Furtheremore using (2.6) in the payoff expression one gets:

$$PO^{rswn}(t_\alpha) = max\left(\sum_{i=\alpha+1}^{\omega} c_i B_{\alpha,i}(t_\alpha) + \sum_{i=\alpha+1}^{\omega-1} B_{\alpha',i}(t_\alpha) - \sum_{i=\alpha}^{\omega-1} \beta_i(t_\alpha) B_{\alpha',i}(t_\alpha), 0 \right) \tag{2.8}$$

Lemma 1 from section *The model* will help us rewrite the payoff (2.8) as a function of Gaussian random variable ξ, even if this function in not always monotonic for every possible choice of the three model parameters it always has one unique zero, a property that turns out to be fondamental in order to find an easy closed formula for swaptions. Here we only report the statement:

Lemma 2 *Receiver swaption payoff (2.8) under MHJM model (2.3)-(2.4)-(2.5) can be written as a function $f(\xi)$ with an unique zero in ξ^* for every parameters' choice, moreover the function is positive $\forall \xi < \xi^*$.*

$$\begin{aligned} f(\xi) = &\sum_{i=\alpha+1}^{\omega} c_i B_{\alpha,i}(t_0) e^{-\lambda_{\alpha,i}\xi - \frac{\lambda^2_{\alpha,i} V^2}{2}} + \sum_{i=\alpha+1}^{\omega-1} B_{\alpha',i}(t_0) e^{-\lambda_{\alpha,i}\xi - \frac{\lambda^2_{\alpha,i} V^2}{2}} \\ &- \sum_{i=\alpha}^{\omega-1} \beta_i(t_0) B_{\alpha',i}(t_0) e^{-\nu_{\alpha,i}\xi - \frac{\nu^2_{\alpha,i} V^2}{2}} \end{aligned}$$

Applying *Lemma 2* to (2.8) and substituting the resulting expression in (2.7) allow to obtain the closed swaption pricing formula:

$$\begin{aligned} P^{rswn}(t_0; t_\alpha, \{t'_i\}, \{t_i\}, R) = &\sum_{i=\alpha+1}^{\omega} c_i B_{\alpha,i}(t_0) \Phi\left(\frac{\xi^*}{V} + V\lambda_{\alpha,i} \right) + \sum_{i=\alpha+1}^{\omega-1} B_{\alpha',i}(t_0) \Phi\left(\frac{\xi^*}{V} + V\lambda_{\alpha',i} \right) \\ &- \sum_{i=\alpha}^{\omega-1} \beta_i(t_0) B_{\alpha',i}(t_0) \Phi\left(\frac{\xi^*}{V} + V\nu_{\alpha',i} \right) \end{aligned} \tag{2.9}$$

where with Φ we indicate the standard normal cumulative distribution function.
We stress again that the reader interested in the exhaustive pricing methodology can find it in Baviera (2017).

Chapter 3

An Exponentially Quadratic Model

In the previous chapter we have considered a multicurve model extension of an Heath-Jarrow-Morton (HJM) setup, now we want to introduce, discuss and develop a multicurve extension of another typical pre-crisis single-curve model class. In particular we want to extend to a multicurve setting the less well-known exponentially quadratic short rate model, we stress again the importance to extend every model to a multicurve setting, for this particular model the transition between the two settings is made adding a short rate spread to the short rate itself for every possible tenor structure considered.
More in details we will consider a dual curve model, a discounting and a pseudo-discounting one (the first to discount future cash flow, the latter to generate future cash flow), thus considering a single tenor structure (in particular it will be a six-month tenor structure). We should then introduce a factor model for the short rate and its spread, usually in the IR literature the most common factor class is the exponentially affine one, for which in order to guarantee rates' and spreads' positivity one has to consider square root models that eventually lead to involved distribution such as noncentral chi-squared distribution. Here instead we will consider an Exponentially Quadratic model (EQM) whose factor class has a more convenient Gaussian distribution.

3.1 Martingale Measure

Before starting we must say that we want to keep where possible the same notation already introduced in previous chapters in order to naturally link the two different model, nevertheless where necessary some already discussed financial quantities will be recalled therefore avoiding a boring backward research.
As usual we indicate with $B(t,T)$ the discount factors and with $\bar{B}(t,T)$ the pseudo discount factors while $f(t,T) = -\frac{\partial}{\partial T} ln\ B(t,T)$ represents the instantaneous forward rate, $r_t = \lim_{T \to t} f(t,T)$ the short rate and $B_t^{OIS} = e^{\int_0^t r_u du}$ the OIS bank account, that represents the numeraire for the standard martingale measure $\mathbb{Q}$ under which our model is build.
We have already introduce the stochastic discount factor but here we want to express its dependence with the short rate $D(t,T) = e^{-\int_t^T r_u du}$, then substituting we get $B(t,T) = \mathbb{E}_t\left[e^{-\int_t^T r_u du}\right]$. In the following we label with s_t the short rate spread

related to the six months tenor, we can then replicate these financial quantities considering the short rate plus spread:

$$\begin{cases} \bar{f}(t,T) = -\frac{\partial}{\partial T} ln\ \bar{B}(t,T) \\ \bar{r}_t = \lim_{T\to t} \bar{f}(t,T) \\ \bar{D}(t,T) = e^{-\int_t^T \bar{r}_u du} \\ \bar{B}(t,T) = \mathbb{E}_t\big[e^{-\int_t^T \bar{r}_u du}\big] \end{cases}$$

where $\bar{r}_u = r_u + s_u$.
Our goal is now to define a model for Libor rate: before the crisis it was consider equal to the discrete compounding forward rate ($F(t,T,T+\Delta) = \frac{1}{\Delta}(\frac{B(t,T)}{B(t,T+\Delta)} - 1)$) while now it is commonly accepted to consider the Libor-OIS spread and so to define the forward Libor rate starting from the pseudo discounting curve:

$$L(t;T,T+\Delta) = \frac{1}{\delta(T,T+\Delta)}\left(\frac{\bar{B}(t,T)}{\bar{B}(t,T+\Delta)} - 1\right) \tag{3.1}$$

3.2 The exponentially quadratic short rate model

We need a dynamical model for both the short rate and its spread under Q martingale measure, as already said the two basic factor model classes are the exponentially affine and the exponentially quadratic, both of them allow for flexibility and analytical tractability and this, in turn, allows for closed formulas for linear and optional IR derivatives.
We have chosen the latter class in order to deal with more comfortable distribution of the factors with the agreement to introduce a quadratic term in both short rate and spread. For the model factors, we have selected a mean reversion Gaussian distribution lead by an Ornstein-Uhlenbeck (OU) dynamic with zero-mean reversion level and a positive mean reversion parameter. We want to stress that the linear term of our model can potentially lead to negative values of rates and spreads, although only with small probability due to the mean reversion parameter's positiveness.
We chose the minimal number of Gaussian factors that guaranteed anyway the possibility to introduce correlation between rate and spread:

$$d\Psi_t^i = -b_i\Psi_t^i dt + \sigma_i dW_t^i,\ \Psi_{t_0}^i = \psi_i\ \ i = 1,2,3 \tag{3.2}$$

with $b_i, \sigma_i > 0$, $\psi_i = const$ and W_t^i three independent Q-Brownian motions that ensure factors' independence. The model we choose for short rate and its spread takes into consideration an independent idiosyncratic risk factor for both short rate and spread:

$$\begin{cases} r_t = \Psi_t^1 + (\Psi_t^2)^2 \\ s_t = \kappa\Psi_t^1 + (\Psi_t^3)^2 \end{cases} \tag{3.3}$$

where $\kappa \in [-1,1]$ is the correlation coefficient.

3.3 Bond prices (discount and pseudo-discount)

The main goal of this section is to find a theoretical explicit formula for $B(t,T)$ and $\bar{B}(t,T)$ given the EQM model defined by (3.2)-(3.3).

Let us start rewriting in a vectorial form (3.2):

$$d\mathbf{\Upsilon}_t = \mathbf{F}\mathbf{\Upsilon}_t dt + \mathbf{D}d\mathbf{W}_t$$

where $\mathbf{\Upsilon}_t = (\Upsilon_t^1\ \Upsilon_t^2\ \Upsilon_t^3)'$, $\mathbf{F} = -diag(b_1, b_2, b_3)$, $\mathbf{D} = diag(\sigma_1, \sigma_2, \sigma_3)$ and $'$ denotes the transposition operator.
Besides let the instantenous forward rates be:

$$\begin{cases} f(t,T) = a(t,T) + \mathbf{b}'(t,T)\mathbf{\Upsilon}_t + \mathbf{\Upsilon}_t'\mathbf{c}(t,T)\mathbf{\Upsilon}_t \\ \bar{f}(t,T) = \bar{a}(t,T) + \bar{\mathbf{b}}'(t,T)\mathbf{\Upsilon}_t + \mathbf{\Upsilon}_t'\bar{\mathbf{c}}(t,T)\mathbf{\Upsilon}_t \end{cases} \tag{3.4}$$

where $a(t,T)$, $\mathbf{b}(t,T)$, $\mathbf{c}(t,T)$ and their respective bar versions are respectively scalar, vector and symmetric-matrix-valued functions , differentiable with respect to t. Recalling definitions of short rates and substituting (3.4) we get:

$$\begin{cases} r_t = a(t) + \mathbf{b}'(t)\mathbf{\Upsilon}_t + \mathbf{\Upsilon}_t'\mathbf{c}(t)\mathbf{\Upsilon}_t \\ \bar{r}_t = \bar{a}(t) + \bar{\mathbf{b}}'(t)\mathbf{\Upsilon}_t + \mathbf{\Upsilon}_t'\bar{\mathbf{c}}(t)\mathbf{\Upsilon}_t \end{cases} \tag{3.5}$$

where $a(t) = a(t,t)$, $\mathbf{b}(t) = \mathbf{b}(t,t)$, $\mathbf{c}(t) = \mathbf{c}(t,t)$ and the same relations still hold for quantities' bar version. Comparing (3.3) and (3.5) we obtain the following condition

$$\begin{cases} a(t) = \bar{a}(t) = 0 \\ \mathbf{b}(t) = (1\,0\,0)',\ \bar{\mathbf{b}}(t) = (1+\kappa\,0\,0)' \\ \mathbf{c}(t) = diag(0,1,0),\ \bar{\mathbf{c}}(t) = diag(0,1,1) \end{cases} \tag{3.6}$$

Now if we invert relations holding for the instantaneous forward rate and bond prices and we substitute (3.6) we have:

$$\begin{cases} B(t,T) = e^{-A(t,T) - \mathbf{B}'(t,T)\mathbf{\Upsilon}_t - \mathbf{\Upsilon}_t'\mathbf{C}(t,T)\mathbf{\Upsilon}_t} \\ \bar{B}(t,T) = e^{-\bar{A}(t,T) - \bar{\mathbf{B}}'(t,T)\mathbf{\Upsilon}_t - \mathbf{\Upsilon}_t'\bar{\mathbf{C}}(t,T)\mathbf{\Upsilon}_t} \end{cases} \tag{3.7}$$

where $A(t,T) = \int_t^T a(t,u)du$, $\mathbf{B}(t,T) = \int_t^T \mathbf{b}(t,u)du$, $\mathbf{C}(t,T) = \int_t^T \mathbf{c}(t,u)du$.
Our objective is therefore to obtain the expressions of the time dependent functions $A(t,T)$, $\mathbf{B}(t,T)$, $\mathbf{C}(t,T)$ and their respective bar versions, we have included them into the following proposition, in whose proof it is shown the procedure needed in order to obtain them.

Proposition 1 *Assume the EQM model introduced in the previous section (see (3.2) and (3.3)), then bond prices are given by*

$$B(t,T) = e^{-A(t,T) - \mathcal{B}(t,T)\Upsilon_t^1 - C(t,T)(\Upsilon_t^2)^2} \tag{3.8}$$

$$\bar{B}(t,T) = e^{-\bar{A}(t,T) - (\kappa+1)\mathcal{B}(t,T)\Upsilon_t^1 - C(t,T)(\Upsilon_t^2)^2 - \bar{C}(t,T)(\Upsilon_t^3)^2} \tag{3.9}$$

where

$$\begin{cases} A(t,T) = \sigma_2^2 \int_t^T C(u,T)du - \frac{\sigma_1^2}{2} \int_t^T \mathcal{B}^2(u,T)du \\ \bar{A}(t,T) = \sigma_2^2 \int_t^T C(u,T)du + \sigma_3^2 \int_t^T \bar{C}(u,T)du - \frac{(\sigma_1(1+\kappa))^2}{2} \int_t^T \mathcal{B}^2(u,T)du \\ \mathcal{B}(t,T) = \frac{1-e^{-b_1(T-t)}}{b_1} \\ C(t,T) = \frac{2(e^{h^2(T-t)}-1)}{2h^2+(2b_2+h^2)(e^{h^2(T-t)}-1)}, \ h^2 = \sqrt{4b_2^2+8\sigma_2^2} \\ \bar{C}(t,T) = \frac{2(e^{h^3(T-t)}-1)}{2h^3+(2b_3+h^3)(e^{h^3(T-t)}-1)}, \ h^3 = \sqrt{4b_3^2+8\sigma_3^2} \end{cases} \tag{3.10}$$

Proof 1 *We want to find Ordinary Differential Equations (ODE) for the time-dependent coefficients applying the well-known HJM drift condition. In the following (in particular in Appendix A) we show how to obtain ODEs for not bar quantities, obviously, for the bar quantities the procedure is exactly analogous.*
Applying Ito's lemma to $f(t,T)$ in (3.4) one gets:

$$\begin{aligned} df(t,T) = {} & [a_t(t,T) + \mathbf{b}_t'(t,T)\boldsymbol{\Psi}_t + \boldsymbol{\Psi}_t'\mathbf{c}_t(t,T)\boldsymbol{\Psi}_t + 2\boldsymbol{\Psi}_t'\mathbf{F}\mathbf{c}(t,T)\boldsymbol{\Psi}_t + tr(\mathbf{D}c(t,T)\mathbf{D})]dt \\ & + [\mathbf{b}'(t,T)\mathbf{D} + 2\boldsymbol{\Psi}_t'\mathbf{c}(t,T)\mathbf{D}]d\mathbf{W}_t \end{aligned} \tag{3.11}$$

Let us call the drift $\mu(t,T)$ while $\boldsymbol{\sigma}(t,T)$ the volatility, then we can applay the well-known HJM dirft condition $\mu(t,T) = \boldsymbol{\sigma}'(t,T)\int_t^t \boldsymbol{\sigma}(t,u)du$ that leads to:

$$\begin{aligned} \mu(t,T) = {} & 4\boldsymbol{\Psi}_t'\mathbf{c}(t,T)\mathbf{D}^2 \int_t^T \mathbf{c}(t,u)du\boldsymbol{\Psi}_t + 2\boldsymbol{\Psi}_t'\mathbf{c}(t,T)\mathbf{D}^2 \int_t^T \mathbf{b}(t,u)du \\ & + 2\mathbf{b}'(t,u)\mathbf{D}^2 \int_t^T \mathbf{c}(t,u)du\boldsymbol{\Psi}_t + \mathbf{b}'(t,u)\mathbf{D}^2 \int_t^T \mathbf{b}(t,u)du \end{aligned} \tag{3.12}$$

Now we have to equate the quadratic, linear and constant term of (3.11) and (3.12), then observe that the equations must be valid for every $\boldsymbol{\Psi}_t$, integrating with respect to T variable and so get the following ODEs:

$$\begin{cases} \mathbf{C}_t(t,T) + 2\mathbf{F}\mathbf{C}(t,T) - 2\mathbf{C}(t,T)\mathbf{D}^2\mathbf{C}(t,T) + \mathbf{c}_t(t) = 0, \mathbf{C}(T,T) = 0 \\ \bar{\mathbf{C}}_t(t,T) + 2\mathbf{F}\bar{\mathbf{C}}(t,T) - 2\bar{\mathbf{C}}(t,T)\mathbf{D}^2\bar{\mathbf{C}}(t,T) + \bar{\mathbf{c}}_t(t) = 0, \bar{\mathbf{C}}(T,T) = 0 \\ \mathbf{B}_t(t,T) + \mathbf{F}\mathbf{B}(t,T) - 2\mathbf{C}\mathbf{D}^2\mathbf{B}(t,T) + \mathbf{b}(t) = 0, \mathbf{B}(T,T) = 0 \\ \bar{\mathbf{B}}_t(t,T) + \mathbf{F}\bar{\mathbf{B}}(t,T) - 2\bar{\mathbf{C}}\mathbf{D}^2\bar{\mathbf{B}}(t,T) + \bar{\mathbf{b}}(t) = 0, \bar{\mathbf{B}}(T,T) = 0 \\ A_t(t,T) + tr(\mathbf{D}\mathbf{C}(t,T)\mathbf{D}) - \frac{1}{2}\mathbf{B}'(t,T)\mathbf{D}^2\mathbf{B}(t,T) + a(t) = 0, A(T,T) = 0 \\ \bar{A}_t(t,T) + tr(\mathbf{D}\bar{\mathbf{C}}(t,T)\mathbf{D}) - \frac{1}{2}\bar{\mathbf{B}}'(t,T)\mathbf{D}^2\bar{\mathbf{B}}(t,T) + \bar{a}(t) = 0, \bar{A}(T,T) = 0 \end{cases}$$

The final condition are found imposing that $B(T,T) = \bar{B}(T,T) = 1$.
Now observe that the first ODE has only one non trivial solution, the second two with one coinciding with the first ODE solution:

$$\begin{cases} C_t(t,T) - 2b_2C(t,T) - 2\sigma_2^2C^2(t,T) + 1 = 0, \ C(T,T) = 0 \\ \bar{C}_t(t,T) - 2b_3\bar{C}(t,T) - 2\sigma_3^2\bar{C}^2(t,T) + 1 = 0, \ \bar{C}(T,T) = 0 \end{cases} \tag{3.13}$$

These are both Bernoulli ODEs after a change of variable with final condition whose solution is indicated in (3.10).
Between the six ODEs representing by the third and fourth vectorized ODEs only two are

non zero, they are:

$$\begin{cases} \mathcal{B}_t(t,T) - b_1\mathcal{B}(t,T) + 1 = 0, \ \mathcal{B}(T,T) = 0 \\ \bar{\mathcal{B}}_t(t,T) - b_1\bar{\mathcal{B}}(t,T) + 1 + \kappa = 0, \ \bar{\mathcal{B}}(T,T) = 0 \end{cases} \tag{3.14}$$

whose solution can simply be obtained.
To close the proof we sould rewrite the last two scalar ODEs of the system and observe that the expression for $A(t,T)$ and $\bar{A}(t,T)$ in (3.10) are exactly solutions of these ODEs.

$$\begin{cases} A_t(t,T) + \sigma_2^2 C(t,T) - \frac{1}{2}\sigma_1^2\mathcal{B}^2(t,T) = 0, \ A(T,T) = 0 \\ \bar{A}_t(t,T) + \sigma_2^2 C(t,T) + \sigma_3^2\bar{C}(t,T) - \frac{1}{2}\sigma_1^2\bar{\mathcal{B}}^2(t,T) = 0, \ A(T,T) = 0 \end{cases} \tag{3.15}$$

We remind to Appendix A for all the details. Finally, (3.8) and (3.9) are easily obtained from (3.7) and the discussion above about the non-trivial solution of ODEs.

Remark 1

- *For the time functions coefficient $\mathcal{B}$ we have chosen an italic format only to avoiding confusion with discount factor $B(t,T)$.*
- *In the case of h^2 and h^3 the apexes represents indices and not powers, they are put at apex because later on we will introduce h_k and again we want to distinguish them.*
- *In the process of implementing the code, when possible, we always try to find an analytic formula for the integral left unsolved like the ones on the previous statement, here some of their closed formula:*

$$\begin{cases} \int_t^T C(u,T)du = 2\frac{2ln\left[\frac{2h^2}{2b_2(e^{h^2(T-t)}-1)+h^2(e^{h^2(T-t)}+1)}\right]+(2b_2+h^2)(T-t)}{(2b_2+h^2)(2b_2-h^2)} \\ \int_t^T \bar{C}(u,T)du = 2\frac{2ln\left[\frac{2h^3}{2b_3\left(e^{h^3(T-t)}-1\right)+h^3\left(e^{h^3(T-t)}+1\right)}\right]+(2b_3+h^3)(T-t)}{(2b_3+h^3)(2b_3-h^3)} \\ \int_t^T \mathcal{B}^2(u,T)du = \frac{2b_1(T-t)-e^{-2b_1(T-t)}+4e^{-b_1(T-t)}-3}{2b_1^3} \end{cases}$$

3.4 Forward measure

In light of the following sections where the goal will be to price some IR derivatives, it is now necessary to introduce the martingale forward measure. Let us indicate with $\mathbb{Q}^T$ the T-forward measure with $B(t,T)$ as numeraire, then the Radon-Nikodym process will be defined as

$$\mathcal{L}_t = \left.\frac{d\mathbb{Q}^T}{d\mathbb{Q}}\right|_{\mathscr{F}_t} = \frac{B(t,T)}{B(0,T)B_t^{OIS}}$$

from which using *Proposition 1* and applying Ito formula (remembering that $\mathcal{L}_t$ is a Q-martingale by definition) is easy to obtain its dynamics:

$$d\mathcal{L}_t = \mathcal{L}_t(-\sigma_1\mathcal{B}(t,T) \quad -2\sigma_2 C(t,T)\Psi_t^2 \quad 0)\cdot d\mathbf{W}_t^{\mathbb{Q}}$$

where $\mathbf{W}_t^{\mathbb{Q}} = (W_t^{1,\mathbb{Q}} \ W_t^{2,\mathbb{Q}} \ W_t^{3,\mathbb{Q}})'$ while from now on $W_t^{i,T}$ with $i = 1,2,3$ represent Q^T independent Brownian motions.

In this section we want to find expected values and variances of the three factor processes under the Q^T forward measure, in order to achieve this objective we need first their dynamics under the appropriate probabilistic measure and then we must apply Girsanov's theorem ($d\mathbf{W}_t^Q = (-\sigma_1 B(t,T) \quad -2\sigma_2 C(t,T)\Psi_t^2 \quad 0)'dt + d\mathbf{W}_t^T$) to the $\mathbb{Q}$ dynamics expressed in (3.2):

$$\begin{cases} d\Psi_t^1 = -[b_1\Psi_t^1 + \sigma_1^2 \mathcal{B}(t,T)]dt + \sigma_1 dW_t^{1,T} \\ d\Psi_t^2 = -[b_2 + 2\sigma_2^2 C(t,T)]\Psi_t^2 dt + \sigma_2 dW_t^{2,T} \\ d\Psi_t^3 = -b_3\Psi_t^3 dt + \sigma_3 dW_t^{3,T} \end{cases} \tag{3.16}$$

As it can be easly observed the third factor is independent from the change of measure while, for what concerned the first gaussian factor, now the mean-reversion level is no more equal to zero.
In order to compute the two moments of the distributions it is helpfull to introduce the following result which gives us the analytic solution of a SDE of the OU type.

Lemma 3 *Given an appropriate probabilistic measure and a process with dynamic* $dX_t = (a(t)X_t + b(t))dt + \sigma dW_t$ *and deterministic initial condition* X_{t_0} *the analytic solution of the SDE is*

$$X_t = \phi(t)\left(X_{t_0} + \int_{t_0}^t \frac{b(u)}{\phi(u)}du + \int_{t_0}^t \frac{\sigma}{\phi(u)}dW_u\right)$$

where $\phi(t) = e^{\int_{t_0}^t a(u)du}$

After being defined $\alpha_t^{i,T} = \mathbb{E}^T[\Psi_t^i]$ and $\beta_t^{i,T} = Var^T(\Psi_t^i)$ where with $\mathbb{E}^T$ we represents the expected value under T-forward measure, we only have to apply *Lemma 3* to our three factors and then simply evalutate the expected value and variance of the stochastic processes to find the following results under the T-forward measure:

$$\begin{cases} \alpha_t^{1,T} = e^{-b_1(t-t_0)}\left[\Psi_{t_0}^1 - \frac{\sigma_1^2}{b_1^2}(e^{b_1(t-t_0)} - 1) + \frac{\sigma_1^2}{2b_1^2}e^{-b_1T}(e^{b_1(2t-t_0)} - e^{b_1t_0})\right] \\ \beta_t^{1,T} = \frac{\sigma_1^2}{2b_1}(1 - e^{-2b_1(t-t_0)}) \\ \alpha_t^{2,T} = \Psi_{t_0}^2 e^{-b_2(t-t_0)-2\sigma_2^2 C_{int}(t;t_0,T)} \\ \beta_t^{2,T} = \sigma_2^2 e^{-2b_2(t-t_0)-4\sigma_2^2 C_{int}(t;t_0,T)} \int_{t_0}^t e^{2b_2(u-t_0)+4\sigma_2^2 C_{int}(u;t_0,T)}du \\ \alpha_t^{2,T} = \Psi_{t_0}^3 e^{-b_3(t-t_0)} \\ \beta_t^{3,T} = \frac{\sigma_3^2}{2b_3}(1 - e^{-2b_3(t-t_0)}) \end{cases} \tag{3.17}$$

where $C_{int}(t;t_0,T) = \int_{t_0}^t C(u,T)du$.

3.5 Interest rate swap pricing

Definitions and notations are very similar to the ones reported in *Chapter 3- Section 2.3*, in this chapter we change slightly the notation and we consider a payer swap. Let $\{T_k^f\}_{k=1,..,n}$ be a collection of dates corresponding to the floating leg (receiving dates) and $\{T_k^F\}_{k=1,..,N}$ corresponding to fixed leg (payment dates), T_0 swap's starting date, R strike rate and $\delta_k^f = \delta(T_{k-1}^f, T_k^f)$ the year fraction between two receiving dates with Act/360 convention while with $\delta_k^F = \delta(T_{k-1}^F, T_k^F)$ the year fraction between two payment dates with 30/360 European convention.
As usual the arbitrage-free price at time $t < T_0$ of such a contract is found imposing

to zero the NPV in t (in order to easier the notation we impose to one the notional):

$$\begin{aligned}
P^{psw}(t;T_0,\{T_k^f\},\{T_k^F\},R) &= NPV^f(t,T_0,\{T_k^f\}) - NPV^F(t,T_0,\{T_k^F\},R)\\
&= \sum_{k=1}^{n} B(t,T_k^f)\delta_k^f \mathbb{E}_t^{T_k^f}[L(T_{k-1}^f,T_k^f)] - \sum_{k=1}^{N} B(t,T_k^F)\delta_k^F R\\
&= \sum_{k=1}^{n} B(t,T_k^f)\mathbb{E}_t^{T_k^f}\left[\frac{1}{\bar{B}(T_{k-1}^f,T_k^f)}\right] - \left[\sum_{k=1}^{n} B(t,T_k^f) + R\sum_{k=1}^{N} B(t,T_k^F)\delta_k^F\right]
\end{aligned} \tag{3.18}$$

where in the last equality it was used (3.1).
In the expression (3.18) the key component to be calculated is the expected value which can be rewritten using (3.9) from *Proposition 1* and the independence of the gaussian factors in the following way:

$$\mathbb{E}_t^{T_k^f}\left[\frac{1}{\bar{B}(T_{k-1}^f,T_k^f)}\right] = e^{A_k}\mathbb{E}_t^{T_k^f}\left[e^{(\kappa+1)B_k\Psi^1_{T_{k-1}^f}}\right]\mathbb{E}_t^{T_k^f}\left[e^{C_k\left(\Psi^2_{T_{k-1}^f}\right)^2}\right]\mathbb{E}_t^{T_k^f}\left[e^{\bar{C}_k\left(\Psi^3_{T_{k-1}^f}\right)^2}\right] \tag{3.19}$$

where in order to easier the notation we imposed:

$$\begin{cases}
A_k = \bar{A}(T_{k-1}^f,T_k^f)\\
B_k = \mathcal{B}(T_{k-1}^f,T_k^f)\\
C_k = C(T_{k-1}^f,T_k^f)\\
\bar{C}_k = \bar{C}^{(}T_{k-1}^f,T_k^f)
\end{cases}$$

Given the fact that our final goal is to price a swaption, we want to write the expectation conditional on the filtration in t as random variable extrapolated from a stochastic process at time t, here's why we now introduce two lemmas that will help us achieve our goal.

Lemma 4 *If $K \in \mathbb{R}$ and a stochastic process has an Hull-White dynamic of the type $dX_t = (a(t) - bX_t)dt + \sigma dW_t$ then $\mathbb{E}_t[e^{-KX_T}] = e^{\alpha(t,T)-\beta(t,T)X_t}$ with $\alpha(t,T)$ and $\beta(t,T)$ solutions of the following ODEs*

$$\begin{cases}\beta_t(t,T) - b\beta(t,T) = 0\\ \beta(T,T) = K\end{cases} \begin{cases}\alpha_t(t,T) = a(t)\beta(t,T) - \frac{\sigma^2}{2}\beta^2(t,T)\\ \alpha(T,T) = 0\end{cases}$$

Lemma 5 *Let X_t be a stochastic process with dynamic $dX_t = b(t)X_tdt + \sigma dW_t$.
Then $\forall C \in \mathbb{R}$ such that $\mathbb{E}_t[e^{CX_T^2}] < \infty$ we have $\mathbb{E}_t[e^{CX_T^2}] = e^{\alpha(t,T)-\beta(t,T)X_t^2}$ with $\alpha(t,T)$ and $\beta(t,T)$ solutions of the following ODEs*

$$\begin{cases}\beta_t(t,T) + 2b(t)\beta(t,T) - 2\sigma^2\beta^2(t,T) = 0\\ \beta(T,T) = -C\end{cases} \begin{cases}\alpha_t(t,T) = \sigma^2\beta(t,T)\\ \alpha(T,T) = 0\end{cases}$$

We highlight that, when applying *Lemma 4* to $\Psi^1_{T_{k-1}^f}$ and *Lemma 5* to $\Psi^2_{T_{k-1}^f}$ and $\Psi^3_{T_{k-1}^f}$, we must consider, conditional to the actual receiving date, the proper Q^{T_k} forward measure in order to be sure to use the right process dynamic; in the following we report the forward dynamics of the three gaussian processes under the Q^{T_k} measure (c.f. system (3.16)):

$$\begin{cases} d\Psi^1_t = -[b_1\Psi^1_t + \sigma_1^2\mathcal{B}(t,T^f_k)]dt + \sigma_1 dW^{1,T_k}_t \\ d\Psi^2_t = -[b_2 + 2\sigma_2^2 C(t,T^f_k)]\Psi^2_t dt + \sigma_2 dW^{2,T_k}_t \\ d\Psi^3_t = -b_3\Psi^3_t dt + \sigma_3 dW^{3,T_k}_t \end{cases}$$

We list the results (for details calculations see *Appendix A*):
First factor

$$\begin{cases} \mathbb{E}^{T^f_k}_t\left[e^{(\kappa+1)B_k\Psi^1_{T^f_{k-1}}}\right] = e^{\Gamma^1(t,T^f_{k-1})-\rho^1(t,T^f_{k-1})\Psi^1_t} \\ \rho^1(t,T^f_{k-1}) = -(\kappa+1)B_k e^{-b_1(T^f_{k-1}-t)} \\ \Gamma^1(t,T^f_{k-1}) = \frac{\sigma_1^2}{2}\int_t^{T^f_{k-1}}(\rho^1(u,T^f_{k-1}))^2 du + \sigma_1^2\int_t^{T^f_{k-1}}\mathcal{B}(u,T^f_k)\rho^1(u,T^f_{k-1})du \end{cases} \tag{3.20}$$

Second factor

$$\begin{cases} \mathbb{E}^{T^f_k}_t\left[e^{C_k\left(\Psi^2_{T^f_{k-1}}\right)^2}\right] = e^{\Gamma^2(t,T^f_{k-1})-\rho^2(t,T^f_{k-1})(\Psi^2_t)^2} \\ \rho^2(t,T^f_{k-1}) = \frac{C_k exp\{-2b_2(T_{k-1}-t)-4\sigma_2^2\int_t^{T^f_{k-1}}C(u,T^f_k)du\}}{2\sigma_2^2 C_k\int_t^{T^f_{k-1}}exp\{-2b_2(T^f_{k-1}-w)-4\sigma_2^2\int_w^{T^f_{k-1}}C(u,T^f_k)du\}dw-1} \\ \Gamma^2(t,T^f_{k-1}) = -\sigma_2^2\int_t^{T^f_{k-1}}\rho^2(u,T^f_{k-1})du \end{cases} \tag{3.21}$$

Third factor

$$\begin{cases} \mathbb{E}^{T^f_k}_t\left[e^{\bar{C}_k\left(\Psi^3_{T^f_{k-1}}\right)^2}\right] = e^{\Gamma^3(t,T^f_{k-1})-\rho^3(t,T^f_{k-1})(\Psi^3_t)^2} \\ \rho^3(t,T^f_{k-1}) = \frac{-4b_3h_k e^{-2b_3(T^f_{k-1}-t)}}{4\sigma_3^2 h_k e^{-2b_3(T^f_{k-1}-t)}-1}, \text{with } h_k = \frac{\bar{C}_k}{4\sigma_3^2\bar{C}_k-4b_3} \\ \Gamma^3(t,T^f_{k-1}) = -\sigma_3^2\int_t^{T^f_{k-1}}\rho^3(u,T^f_{k-1})du \end{cases} \tag{3.22}$$

Remark 2

- *Obviously apexes for ρ^i and Γ^i are indices and not powers.*
- *Lemma 3 hypothesis is naturally satisfied.*
- *The followings are the analytic solution of the integrals that appear in the formulas (the ones that can be analytically solved of course):*

$$\begin{cases} \int_t^{T^f_{k-1}}(\rho^1(u,T^f_{k-1}))^2 = \frac{[(\kappa+1)B_k]^2}{2b_1}\left(1-e^{-2b_1(T^f_{k-1}-t)}\right) \\ \int_t^{T^f_{k-1}}\mathcal{B}(u,T^f_k)\rho^1(u,T^f_{k-1})du = \frac{(\kappa+1)B_k}{2b_1}\left(e^{-b_1(T^f_k-T^f_{k-1})} - e^{-b_1(T^f_k+T^f_{k-1}-2t)} + 2e^{-b_1(T^f_{k-1}-t)} - 2\right) \\ \int_t^{T^f_{k-1}}C(u,T^f_k)du = 2\frac{2ln\left[\frac{2b_2\left(e^{h^2(T^f_k-T^f_{k-1})}-1\right)+h^2\left(e^{h^2(T^f_k-T^f_{k-1})}+1\right)}{2b_2\left(e^{h^2(T^f_{k-1}-t)}-1\right)+h^2\left(e^{h^2(T-t)}+1\right)}\right]+(2b_2+h^2)(T-t)}{(2b_2+h^2)(2b_2-h^2)} \\ \int_t^{T^f_{k-1}}\rho^3(u,T^f_{k-1})du = \frac{ln\left[\frac{b_3e^{2b_3(T^f_{k-1}-t)}-\sigma_3^2\bar{C}_k\left(e^{2b_3(T^f_{k-1}-t)}\right)}{b_3}\right]-2b_3(T^f_{k-1}-t)}{2\sigma_3^2} \end{cases}$$

It is of foundamental importance to observe that, while $\rho^1(t, T^f_{k-1})$ and $\rho^2(t, T^f_{k-1})$ are continous and at least one time t-differentiable functions $\forall k = 1, .., n$, $\rho^3(t, T^f_{k-1})$ could potentially lead to a singularity of the second type given the fact that the denominator could became null for a certain value of time t and a possible choice of the model's parameters. If this singularity falls within the interval (t, T^f_{k-1}), the corresponding Cauchy problem formed by the ODE and the final condition in T^f_{k-1} is extensible backward from T^f_{k-1} to the singularity plus an ϵ that goes to 0^+.
It is clear that a solution at a time u such that $t < u < \widetilde{T}_{k-1}$, i.e. prior the second order discountinuity ($\widetilde{T}_{k-1}$) could theoretically be found, but it would be required to have a initial/final condition prior the singularity, a condition that we clearly can't found in any way, that is the reason why we must restric our model's parameter in a way that ensures that for every k the eventual singularities fall out of the interval (t, T^f_{k-1}). Due to the complexity of the function $\bar{C}_k$ involved this condition will be given in a implicit way:

$$\mathbf{H0} : (b_3, \sigma_3 \text{ st } (4\sigma_3^2 h_k < 0) \vee (\widetilde{T}_{k-1} < t \vee \widetilde{T}_{k-1} > T^f_{k-1})) \quad \forall k = 1, .., n$$
$$\text{with } \widetilde{T}_{k-1} = T^f_{k-1} - \frac{ln(4\sigma_3^2 h_k)}{2b_3}$$

Finally substituting (3.19)-(3.20)-(3.21)-(3.22) into (3.18) we are only left some calculation to do and we get the following result.

Proposition 2 *Under the model (3.2)-(3.3) and hypothesis* ***H0*** *the price of a payer swap is:*

$$\begin{aligned} P^{psw}(t; T_0, \{T^f_k\}, \{T^F_k\}, R) &= \sum_{k=1}^{n} D_{t,k} e^{-A^f_{t,k} - \widetilde{B}^f_{t,k}\Psi^1_t - \widetilde{C}^{2,f}_{t,k}(\Psi^2_t)^2 - \widetilde{C}^{3,f}_{t,k}(\Psi^3_t)^2} \\ &- \left[\sum_{k=1}^{n} e^{-A^f_{t,k} - B^f_{t,k}\Psi^1_t - C^f_{t,k}(\Psi^2_t)^2} + R \sum_{k=1}^{N} \delta^F_k e^{-A^F_{t,k} - B^F_{t,k}\Psi^1_t - C^F_{t,k}(\Psi^2_t)^2} \right] \end{aligned} \tag{3.23}$$

where for $i \in \{f, F\}$ we have:

$$\begin{cases} A^i_{t,k} = A(t, T^i_k),\ B^i_{t,k} = \mathcal{B}(t, T^i_k),\ C^i_{t,k} = C(t, T^i_k) \\ \widetilde{B}^f_{t,k} = B^f_{t,k} + \rho^1(t, T^f_{k-1}),\ \widetilde{C}^{2,f}_{t,k} = C^f_{t,k} + \rho^2(t, T^f_{k-1}),\ \widetilde{C}^{3,f}_{t,k} = \rho^3(t, T^f_{k-1}) \\ D_{t,k} = e^{A^f_k + \Gamma^1(t, T^f_{k-1}) + \Gamma^2(t, T^f_{k-1}) + \Gamma^3(t, T^f_{k-1})} \end{cases} \tag{3.24}$$

Proof 2 *First of all, we have to substitute (3.19)-(3.20)-(3.21)-(3.22) into (3.18) in order to obtain:*

$$\begin{aligned} &P^{psw}(t; T_0, \{T^f_k\}, \{T^F_k\}, R) = \\ &\sum_{k=1}^{n} B(t, T^f_k) e^{\Gamma^1(t, T^f_{k-1}) + \Gamma^2(t, T^f_{k-1}) + \Gamma^3(t, T^f_{k-1}) - \rho^1(t, T^f_{k-1})\Psi^1_t - \rho^2(t, T^f_{k-1})(\Psi^2_t)^2 - \rho^3(t, T^f_{k-1})(\Psi^3_t)^2} \\ &- \left[\sum_{k=1}^{n} B(t, T^f_k) + R \sum_{k=1}^{N} B(t, T^F_k)\delta^F_k \right] \end{aligned}$$

Then using (3.8) from Proposition 1 we get

$$
\begin{aligned}
&P^{psw}(t;T_0,\{T_k^f\},\{T_k^F\},R) = \\
&\sum_{k=1}^{n} e^{-A(t,T_k^f)+\Gamma^1(t,T_{k-1}^f)+\Gamma^2(t,T_{k-1}^f)+\Gamma^3(t,T_{k-1}^f)-(B(t,T_k^f)+\rho^1(t,T_{k-1}^f))\Psi_t^1-(C(t,T_k^f)+\rho^2(t,T_{k-1}^f))(\Psi_t^2)^2-\rho^3(t,T_{k-1}^f)(\Psi_t^3)^2} \\
&-\left[\sum_{k=1}^{n} e^{-A(t,T_k^f)-B(t,T_k^f)\Psi_t^1-C(t,T_k^f)(\Psi_t^2)^2} + R\sum_{k=1}^{N} e^{-A(t,T_k^F)-B(t,T_k^F)\Psi_t^1-C(t,T_k^F)(\Psi_t^2)^2}\delta_k^F\right]
\end{aligned}
$$

that ends the proof.

3.6 European swaption pricing

In this section, we price a European IR swaption on a payer swap whose starting date T_0 coincide with swaption's expiry T_m. The definition of such a contract was already given in the previous chapter.
The arbitrage-free price is then given by:

$$
\begin{aligned}
P^{pswn}(t;T_m,\{T_k^f\},\{T_k^F\},R) &= B(t,T_m)\mathbb{E}_t^{T_m}[(P^{psw}(T_m;T_0,\{T_k^f\},\{T_k^F\},R))^+] \\
&= B(t,T_m)\int_{\mathbb{R}^3}(g(x,y,z)-h(x,y))^+ f_{(\Psi_{T_m}^1,\Psi_{T_m}^2,\Psi_{T_m}^3)}(x,y,z)dxdydz
\end{aligned}
\tag{3.25}
$$

where g and h are found thanks to *Proposition 2*

$$
\begin{cases}
g(x,y,z) = \sum_{k=1}^{n} D_{T_m,k}e^{-A_{T_m,k}^f-\widetilde{B}_{T_m,k}^f x-\widetilde{C}_{T_m,k}^{2,f}y^2-\widetilde{C}_{T_m,k}^{3,f}z^2} \\
h(x,y) = \sum_{k=1}^{n} e^{-A_{T_m,k}^f-B_{T_m,k}^f x-C_{T_m,k}^f y^2} + R\sum_{k=1}^{N}\delta_k^F e^{-A_{T_m,k}^F-B_{T_m,k}^F x-C_{T_m,k}^F y^2}
\end{cases}
\tag{3.26}
$$

The density function of the random vector $(\Psi_{T_m}^1,\Psi_{T_m}^2,\Psi_{T_m}^3)$ can be written as the product of $f_i = f_{\mathcal{N}(\alpha_{T_m}^{i,T_m},\beta_{T_m}^{i,T_m})}$ due to the independency of the gaussian factors.
For certain values of the model's parameters, g- for fixed $(x,y)\in\mathbb{R}^2$- has particular forms that allow to trasform after some math the triple integral into a double integral, these particular forms are the *Exponential Parabola* and the *Gaussian bell*. The reason behind is that these particular functional forms ensure the monotony of $g(x,y,z)$ with respect to z in the regions $z<0$ and $z>0$ and so the uniqueness of solutions (if they exist) to the equation $g(x,y,z)=h(x,y)$ in the regions $z<0$ and $z>0$.
These forms depend on the sign of $\widetilde{C}_{T_m,k}^{3,f}$ coefficients and the following lemma whose derivation is a mere calcolation left to the readers will help us distinguish between three scenarios in which our parameters can fall.

Lemma 6

$$
\widetilde{C}_{T_m,k}^{3,f} > 0 \Leftrightarrow h_k \in \left(0,\frac{e^{2b_3(T_{k-1}^f-T_m)}}{4\sigma_3^2}\right)
$$

We now introduce the set $\mathcal{M}=\{(x,y)\in\mathbb{R}^2:\ g(x,y,0)\le h(x,y)\}$.
Thanks to *Lemma 6* we know that if $h_k\notin\left(0,\frac{e^{2b_3(T_{k-1}^f-T_m)}}{4\sigma_3^2}\right)$ $\forall k=1,..,n$ we fall into the *Exponential Parabola* case. In this case if $(x,y)\in\mathcal{M}$ then swaption's payoff $(g(x,y,z)-h(x,y))^+ = g(x,y,z)-h(x,y)$ if and only if $z\notin(z_1(x,y),z_2(x,y))$

where $z_1(x,y) = -z_2(x,y)$ are solution to the equation $g(x,y,z) = h(x,y)$. While if $(x,y) \notin \mathcal{M}$ then $(g(x,y,z) - h(x,y))^+ = g(x,y,z) - h(x,y)\ \forall z \in \mathbb{R}$. These results are due to the fact that if *Lemma 6* does not hold $\forall k = 1,..,n$ then $g(x,y,0)$ is the absolute minimum of $g(x,y,z)$ with respect to z variable.
Instead if $h_k \in \left(0, \frac{e^{2b_3(T^f_{k-1}-T_m)}}{4\sigma_3^2}\right)\ \forall k = 1,..,n$ we fall into the *Gaussian bell* case, where $g(x,y,0)$ is the functional abolute maximum with respect to z. Here if $(x,y) \notin \mathcal{M}$ then swaption's payoff reads $(g(x,y,z) - h(x,y))^+ = g(x,y,z) - h(x,y)$ if and only if $z \in (z_1(x,y), z_2(x,y))$ while if $(x,y) \in \mathcal{M}$ then $(g(x,y,z) - h(x,y))^+ = 0\ \forall z \in \mathbb{R}$.
It is important to point out that it may exists sets of parameters such that we do not fall in neither cases, in that eventuality we cannot reduce the integral dimension from three to two.
Before giving the proposition about the swaption price we need to state another lemma that will be used in the proposition.

Lemma 7

$$1 + 2\beta^{3,T_m}_{T_m}\widetilde{C}^{3,f}_{T_m,k} \geq 0 \Leftrightarrow h_k \notin \left(\frac{e^{2b_3(T^f_{k-1}-T_m)}}{4\sigma_3^2}, \frac{e^{2b_3(T^f_{k-1}-t_0)}}{4\sigma_3^2}\right)$$

Proposition 3 *Under the model (3.2)-(3.3) and hypothesis* **H0** *the price of a payer swaption can be computed in three different way depending on the scenario the model parameters are fallen into:*
Case (1): Exponential Parabola+Additional Hp Lemma 7

$$\begin{aligned}
&P^{pswn}(t; T_m, \{T^f_k\}, \{T^F_k\}, R) = B(t,T_m)(P_1 + P_2) \\
&P_1 = \sum_{k=1}^{n} e^{-A^f_{T_m,k}} \int_{\mathcal{M}} \left[D_{T_m,k}\gamma_k e^{-\widetilde{B}^f_{T_m,k}x - \widetilde{C}^{2,f}_{T_m,k}y^2}\left(\Phi(d^1_k(x,y)) + \Phi(-d^2_k(x,y))\right)\right. \\
&\left. - e^{-B^f_{T_m,k}x - C^f_{T_m,k}y^2}\left(\Phi(d^3(x,y)) + \Phi(-d^4(x,y))\right)\right] f_1(x)f_2(y)dxdy \\
&- R\sum_{k=1}^{N} \delta^F_k e^{-A^F_{T_m,k}} \int_{\mathcal{M}} e^{-B^F_{T_m,k}x - C^F_{T_m,k}y^2}\left(\Phi(d^3(x,y)) + \Phi(-d^4(x,y))\right) f_1(x)f_2(y)dxdy \\
&P_2 = \sum_{k=1}^{n} e^{-A^f_{T_m,k}} \int_{\mathcal{M}^C} \left[D_{T_m,k}\gamma_k e^{-\widetilde{B}^f_{T_m,k}x - \widetilde{C}^{2,f}_{T_m,k}y^2} - e^{-B^f_{T_m,k}x - C^f_{T_m,k}y^2}\right] f_1(x)f_2(y)dxdy \\
&- R\sum_{k=1}^{N} \delta^F_k e^{-A^F_{T_m,k}} \int_{\mathcal{M}^C} e^{-B^F_{T_m,k}x - C^F_{T_m,k}y^2} f_1(x)f_2(y)dxdy
\end{aligned} \tag{3.27}$$

Case (2): Gaussian Bell

$$\begin{aligned}
&P^{pswn}(t; T_m, \{T^f_k\}, \{T^F_k\}, R) = B(t,T_m)(P_1 + P_2) \\
&P_1 = 0 \\
&P_2 = \sum_{k=1}^{n} e^{-A^f_{T_m,k}} \int_{\mathcal{M}^C} \left[D_{T_m,k}\gamma_k e^{-\widetilde{B}^f_{T_m,k}x - \widetilde{C}^{2,f}_{T_m,k}y^2}\left(\Phi(d^2_k(x,y)) - \Phi(d^1_k(x,y))\right)\right. \\
&\left. - e^{-B^f_{T_m,k}x - C^f_{T_m,k}y^2}\left(\Phi(d^4(x,y)) - \Phi(d^3(x,y))\right)\right] f_1(x)f_2(y)dxdy \\
&- R\sum_{k=1}^{N} \delta^F_k e^{-A^F_{T_m,k}} \int_{\mathcal{M}^C} e^{-B^F_{T_m,k}x - C^F_{T_m,k}y^2}\left(\Phi(d^4(x,y)) - \Phi(d^3(x,y))\right) f_1(x)f_2(y)dxdy
\end{aligned} \tag{3.28}$$

Other acceptable set of model parameters

$$P^{pswn}(t;T_m,\{T_k^f\},\{T_k^F\},R) = B(t,T_m)\int_{\mathbb{R}^3}(g(x,y,z)-h(x,y))^+ f_1(x)f_2(y)f_3(z)dxdydz \tag{3.29}$$

where Φ is the cumulative distribution function of a standard normal random variable, f_i $\forall i = 1,2,3$ are Gaussian density function with mean $\alpha_{T_m}^{i,T_m}$ and variance $\beta_{T_m}^{i,T_m}$ (see (3.17)), $z_1(x,y)$, $z_2(x,y)$ are solutions to the equation $g(x,y,z) = h(x,y)$, time dependent coefficients are defined in (3.24) from Proposition 2, $g(x,y,z)$, $h(x,y)$ are defined in (3.26) and finally $\forall k = 1,..,n$ we have:

$$\begin{cases} \theta_k = \frac{\alpha_{T_m}^{3,T_m}}{\beta_{T_m}^{3,T_m}}\left(1-\frac{1}{\sqrt{1+2\beta_{T_m}^{3,T_m}\widetilde{C}_{T_m,k}^{3,f}}}\right) \\ \gamma_k = \frac{exp\{\frac{\theta_k^2}{2}\beta_{T_m}^{3,T_m}-\theta_k\alpha_{T_m}^{3,T_m}\}}{\sqrt{1+2\beta_{T_m}^{3,T_m}\widetilde{C}_{T_m,k}^{3,f}}} \\ d_k^1(x,y) = \frac{\sqrt{1+2\beta_{T_m}^{3,T_m}\widetilde{C}_{T_m,k}^{3,f}}z_1(x,y)-(\alpha_{T_m}^{3,T_m}-\theta_k\beta_{T_m}^{3,T_m})}{\sqrt{\beta_{T_m}^{3,T_m}}} \\ d_k^2(x,y) = \frac{\sqrt{1+2\beta_{T_m}^{3,T_m}\widetilde{C}_{T_m,k}^{3,f}}z_2(x,y)-(\alpha_{T_m}^{3,T_m}-\theta_k\beta_{T_m}^{3,T_m})}{\sqrt{\beta_{T_m}^{3,T_m}}} \\ d^3(x,y) = \frac{z_1(x,y)-\alpha_{T_m}^{3,T_m}}{\sqrt{\beta_{T_m}^{3,T_m}}} \\ d^4(x,y) = \frac{z_2(x,y)-\alpha_{T_m}^{3,T_m}}{\sqrt{\beta_{T_m}^{3,T_m}}} \end{cases} \tag{3.30}$$

Proof 3
The scenario where Jamishidian approach cannot be applied it was already proven in the section (c.f. equation (3.25)), so we are left to prove the other two cases.
Let's start observing that $\mathcal{M}$ and its complementary form a partition of $\mathbb{R}^2$, so using Fubini-Tonelli theorem and the measured property of the Lebesgue's integral we can write starting from (3.25):

$$\begin{aligned} P^{swn}(t;T_m,\{T_k^f\},\{T_k^F\},R) &= B(t,T_m)\bigg(\int_{\mathcal{M}}\int_{\mathbb{R}}(g(x,y,z)-h(x,y))^+ f_3(z)dzf_1(x)f_2(y)dxdy \\ &+ \int_{\mathcal{M}^C}\int_{\mathbb{R}}(g(x,y,z)-h(x,y))^+ f_3(z)dzf_1(x)f_2(y)dxdy\bigg) \\ &= B(t,T_m)(P_1+P_2) \end{aligned} \tag{3.31}$$

where we have called the triple integral with integration domain $\mathcal{M}\times\mathbb{R}$ P_1 while the one with domain $\mathcal{M}^C\times\mathbb{R}$ P_2.
In light of the discussion made about the extistence of z-solutions to the equation g=h we can write for Case (1):

$$\begin{cases} P_1 = \int_{\mathcal{M}}\bigg(\int_{-\infty}^{z_1(x,y)}(g(x,y,z)-h(x,y))f_3(z)dz \\ \qquad + \int_{z_2(x,y)}^{+\infty}(g(x,y,z)-h(x,y))f_3(z)dz\bigg)f_1(x)f_2(y)dxdy \\ P_2 = \int_{\mathcal{M}^C}\int_{\mathbb{R}}(g(x,y,z)-h(x,y))f_3(z)dzf_1(x)f_2(y)dxdy \end{cases} \tag{3.32}$$

After substituting (3.26) into (3.32) and exploiting the additivity property of Lebesgue's integral one gets:

$$\begin{cases} P_1 = \displaystyle\sum_{k=1}^{n} e^{-A^f_{T_m,k}} \int_{\mathcal{M}} \Bigg[D_{T_m,k} e^{-\widetilde{B}^f_{T_m,k}x - \widetilde{C}^{2,f}_{T_m,k}y^2} \Bigg(\int_{-\infty}^{z_1(x,y)} e^{-\widetilde{C}^{3,f}_{T_m,k}z^2} f_3(z)dz + \int_{z_2(x,y)}^{+\infty} e^{-\widetilde{C}^{3,f}_{T_m,k}z^2} f_3(z)dz \Bigg) \\ \quad - e^{-B^f_{T_m,k}x - C^f_{T_m,k}y^2} \Bigg(\displaystyle\int_{-\infty}^{z_1(x,y)} f_3(z)dz + \int_{z_2(x,y)}^{+\infty} f_3(z)dz \Bigg) \Bigg] f_1(x) f_2(y) dx dy \\ \quad - R \displaystyle\sum_{k=1}^{N} \delta^F_k e^{-A^F_{T_m,k}} \int_{\mathcal{M}} e^{-B^F_{T_m,k}x - C^F_{T_m,k}y^2} \Bigg(\int_{-\infty}^{z_1(x,y)} f_3(z)dz + \int_{z_2(x,y)}^{+\infty} f_3(z)dz \Bigg) f_1(x) f_2(y) dx dy \\ P_2 = \displaystyle\sum_{k=1}^{n} e^{-A^f_{T_m,k}} \int_{\mathcal{M}^C} \Bigg[D_{T_m,k} e^{-\widetilde{B}^f_{T_m,k}x - \widetilde{C}^{2,f}_{T_m,k}y^2} \int_{\mathbb{R}} e^{-\widetilde{C}^{3,f}_{T_m,k}z^2} f_3(z)dz \\ \quad - e^{-B^f_{T_m,k}x - C^f_{T_m,k}y^2} \displaystyle\int_{\mathbb{R}} f_3(z)dz \Bigg] f_1(x) f_2(y) dx dy \\ \quad - R \displaystyle\sum_{k=1}^{N} \delta^F_k e^{-A^F_{T_m,k}} \int_{\mathcal{M}^C} e^{-B^F_{T_m,k}x - C^F_{T_m,k}y^2} \int_{\mathbb{R}} f_3(z)dz \, f_1(x) f_2(y) dx dy \end{cases} \tag{3.33}$$

In (3.33) two classes of uni-dimensional integral on variable z appear, we aim to rewrite both of them as cdf of a standard normal variable.
For the first class we have to made two change of variable in order to reach our goal: before $\xi = \sqrt{1 + 2\beta^{3,T_m}_{T_m}\widetilde{C}^{3,f}_{T_m,k}}z$ *and then* $u = \frac{\xi - (\alpha^{3,T_m}_{T_m} - \theta_k \beta^{3,T_m}_{T_m})}{\sqrt{\beta^{3,T_m}_{T_m}}}$. *The required additional hypothesis needed in Case (1) actually derived from the first change of variable.*
For the second class of integrals only one change of variable is needed, that is $\xi = \frac{z - \alpha^{3,T_m}_{T_m}}{\sqrt{\beta^{3,T_m}_{T_m}}}$.
Through this methodology we can get:

$$\begin{cases} \int_{-\infty}^{z_1(x,y)} e^{-\widetilde{C}^{3,f}_{T_m,k}z^2} f_3(z)dz = \gamma_k \Phi(d^1_k(x,y)) \\ \int_{z_2(x,y)}^{+\infty} e^{-\widetilde{C}^{3,f}_{T_m,k}z^2} f_3(z)dz = \gamma_k \Phi(-d^2_k(x,y)) \\ \int_{\mathbb{R}} e^{-\widetilde{C}^{3,f}_{T_m,k}z^2} f_3(z)dz = \gamma_k \\ \int_{-\infty}^{z_1(x,y)} f_3(z)dz = \Phi(d^3(x,y)) \\ \int_{z_2(x,y)}^{+\infty} f_3(z)dz = \Phi(-d^4(x,y)) \\ \int_{\mathbb{R}} f_3(z)dz = 1 \end{cases} \tag{3.34}$$

Summing up, backward substituting (3.34)-(3.33)-(3.32)-(3.31) we get (3.27).
To conclude the proof we have to do the same for Case (2) where:

$$\begin{cases} P_1 = \int_{\mathcal{M}} \int_{\mathbb{R}} 0 f_3(z)dz f_1(x) f_2(y) dx dy \\ P_2 = \int_{\mathcal{M}^C} \int_{z_1(x,y)}^{z_2(x,y)} (g(x,y,z) - h(x,y)) f_3(z)dz f_1(x) f_2(y) dx dy \end{cases} \tag{3.35}$$

As it can be seen due to the discussion above over domain $\mathcal{M}$ in Case (2) we immediately obtain $P_1 = 0$, so we are only left to exploit expression of P_2, that becomes:

$$\begin{aligned} P_2 = \sum_{k=1}^{n} e^{-A^f_{T_m,k}} \int_{\mathcal{M}^C} \Bigg[D_{T_m,k} e^{-\widetilde{B}^f_{T_m,k}x - \widetilde{C}^{2,f}_{T_m,k}y^2} \int_{z_1(x,y)}^{z_2(x,y)} e^{-\widetilde{C}^{3,f}_{T_m,k}z^2} f_3(z)dz \\ - e^{-B^f_{T_m,k}x - C^f_{T_m,k}y^2} \int_{z_1(x,y)}^{z_2(x,y)} f_3(z)dz \Bigg] f_1(x)f_2(y)dxdy \\ - R\sum_{k=1}^{N} \delta^F_k e^{-A^F_{T_m,k}} \int_{\mathcal{M}^C} e^{-B^F_{T_m,k}x - C^F_{T_m,k}y^2} \int_{z_1(x,y)}^{z_2(x,y)} f_3(z)dz\, f_1(x)f_2(y)dxdy \end{aligned} \tag{3.36}$$

And applying the same change of variable as before:

$$\begin{cases} \int_{z_1(x,y)}^{z_2(x,y)} e^{-\widetilde{C}^{3,f}_{T_m,k}z^2} f_3(z)dz = \gamma_k(\Phi(d^2_k(x,y) - \Phi(d^1_k(x,y))) \\ \int_{z_1(x,y)}^{z_2(x,y)} f_3(z)dz = \Phi(d^4_k(x,y) - \Phi(d^3_k(x,y)) \end{cases} \tag{3.37}$$

We finally end the proof substituting (3.37)-(3.36)-(3.35) into (3.31) getting (3.28).

Remark 3
In the process of implementing this pricing model, every integral (double or triple) was normalized by the trasformation

$$x = \alpha^{1,T_m}_{T_m} + \sqrt{\beta^{1,T_m}_{T_m}}X,\ y = \alpha^{2,T_m}_{T_m} + \sqrt{\beta^{2,T_m}_{T_m}}Y,\ z = \alpha^{3,T_m}_{T_m} + \sqrt{\beta^{3,T_m}_{T_m}}Z,$$

where X, Y, Z are iid $\mathcal{N}(0,1)$.

- *A numerical solver is needed to find the solution $z_1(x,y)$ and $z_2(x,y)$ of the equation $g(x,y,z) = h(x,y)$ for fixed values of x and y.*
- *The integrals in Proposition 3, Case (1) and Case (2) have a non-trivial integration domain $\mathcal{M}$, $\mathcal{M}^C$. Fortunately $\forall y \in \mathbb{R}$ $g(x,y,0)$ and $h(x,y)$ have a unique intersection, so, in order to compute an appropiate grid, we can use a numerical solver to find the solution $\bar{x}$ st $g(\bar{x},y,0) = h(\bar{x},y)$.*
- *We need to specify that when in the contract the swaption maturity date coincides with the swap start date it is matematically impossible for the parameters to fall into Case (2). In fact the first condition h_k has to satisfy when $k = 1$ is $h_1 \in \left(0, \frac{1}{4\sigma_3^2}\right)$, but considering the functional form of h_1 in (3.22) and the fact that $\bar{C}_1$ is always positive (c.f. equation (3.10)) it follows that $h_1 \notin \left(0, \frac{1}{4\sigma_3^2}\right)$ for every T_m, T_1.*

3.7 Monte Carlo formula

In this section, we present the Monte Carlo formula for a payer swaption in the context of EQM model framework we have discussed in this chapter.
In particular, we aim to estimate the expected value in the first equality of (3.25)

where thanks to *Proposition 1* we have

$$P^{psw}(T_m;T_0,\{T_k^f\},\{T_k^F\},R) = \sum_{k=1}^{n} D_{T_m,k} e^{-A^f_{T_m,k}-\widetilde{B}^f_{T_m,k}\Psi^1_{T_m}-\widetilde{C}^{2,f}_{T_m,k}(\Psi^2_{T_m})^2-\widetilde{C}^{3,f}_{T_m,k}(\Psi^3_{T_m})^2}$$
$$-\left[\sum_{k=1}^{n} e^{-A^f_{T_m,k}-B^f_{T_m,k}\Psi^1_{T_m}-C^f_{T_m,k}(\Psi^2_{T_m})^2}+R\sum_{k=1}^{N}\delta^F_k e^{-A^F_{T_m,k}-B^F_{T_m,k}\Psi^1_{T_m}-C^F_{T_m,k}(\Psi^2_{T_m})^2}\right] \tag{3.38}$$

The key point of the Monte Carlo algorithm is to simulate, under the Q^{T_m}-forward measure, the random value of $\Psi^i_{T_m}$, $i=1,2,3$ stochastic processes starting from their initial condition in t_0 $\boldsymbol{\psi}$.
We recall here their dynamics under the Q^{T_m}-forward measure:

$$\begin{cases} d\Psi^1_t = -[b_1\Psi^1_t+\sigma_1^2 B(t,T_m)]dt+\sigma_1 dW_t^{1,T_m} \\ d\Psi^2_t = -[b_2+2\sigma_2^2 C(t,T_m)]\Psi^2_t dt+\sigma_2 dW_t^{2,T_m} \\ d\Psi^3_t = -b_3\Psi^3_t dt+\sigma_3 dW_t^{3,T_m} \end{cases}$$

Given the fact that under this forward measure Ψ^i_t are time-inhomogeneous OU processes we can use *Lemma 3* in order to find their analytic solution at time T_m:

$$\begin{cases} \Psi^1_{T_m} = e^{-b_1(T_m-t_0)}\left[\Psi^1_{t_0}-\frac{\sigma_1^2}{b_1^2}(e^{b_1(T_m-t_0)}-1)+\frac{\sigma_1^2}{2b_1^2}e^{-b_1T_m}(e^{b_1(2T_m-t_0)}-e^{b_1t_0})+\sigma_1\int_{t_0}^{T_m}e^{b_1(t-t_0)}dW_t^{1,T_m}\right] \\ \Psi^2_{T_m} = e^{-b_2(T_m-t_0)-2\sigma_2^2C_{int}(T_m;t_0,T_m)}\left[\Psi^2_{t_0}+\sigma_2\int_{t_0}^{T_m}e^{b_2(t-t_0)+2\sigma_2^2C_{int}(t;t_0,T_m)}dW_t^{2,T_m}\right] \\ \Psi^3_{T_m} = e^{-b_3(T_m-t_0)}\left[\Psi^3_{t_0}+\sigma_3\int_{t_0}^{T_m}e^{b_3(t-t_0)}dW_t^{3,T_m}\right] \end{cases} \tag{3.39}$$

Now using a well-known property of the stochastic integral we can rewrite

$$\begin{cases} \int_{t_0}^{T_m}e^{b_1(t-t_0)}dW_t^{1,T_m} = \sqrt{\frac{e^{2b_1(T_m-t_0)}-1}{2b_1}}Z_1 \\ \int_{t_0}^{T_m}e^{b_2(t-t_0)+2\sigma_2^2C_{int}(t;t_0,T_m)}dW_t^{2,T_m} = \sqrt{\int_{t_0}^{T_m}e^{2b_2(t-t_0)+4\sigma_2^2C_{int}(t;t_0,T_m)}dt}\,Z_2 \\ \int_{t_0}^{T_m}e^{b_3(t-t_0)}dW_t^{3,T_m} = \sqrt{\frac{e^{2b_3(T_m-t_0)}-1}{2b_3}}Z_3 \end{cases} \tag{3.40}$$

where Z_i are iid standard normal random variable.
Now we have all the basic bricks to build our Monte Carlo swaption's pricing formula, infact we only have to backward substitute (3.40)-(3.38) into the first equality in (3.25) in order to obtain it.
Finally we have to mention that in the following section it will be implemented a Monte Carlo method with variance reduction through antithetic variable: $\forall Z_i$ we have considered an antithetic version $\widetilde{Z}_i=-Z_i$.

3.8 Numerical Results

In this chapter final section, we aim to verify numerically the closed swaption's formula stated in *Proposition 3*. In order to do this, we chose a 5x5 payer swaption contract with valuation date 13 September 2012, five years expiry and tenor whose underlying is a payer swap on Euribor six months with strike rate 2.7485%. We price this contract once using *Proposition 3* closed formula for an arbitrary parameter's choice and then with the same set of parameters we price it multiple times with Monte Carlo method with antithetic variables and an increasing simulation number. We choose a high number of simulations (10^8) in order to get close results. We have

computed a confidence interval of 95% level in order to give an idea of the increasing accuracy of Monte Carlo method, besides we have evaluated the absolute error and the relative error respectively as $|CFprice - MCprice|$ and $\frac{|CFprice - MCprice|}{CFprice}$.
Parameters' choice: $b_i = 10\%$, $\sigma_i = 1\% \ \forall i = 1, 2, 3$, $\kappa = 0.5$
Factors' initial condition: $\boldsymbol{\psi} = (2.5\% \ 0\% \ 0\%)'$
In *Figure 3.1* and *Figure 3.2*, we can observe that the confidence interval's width de-

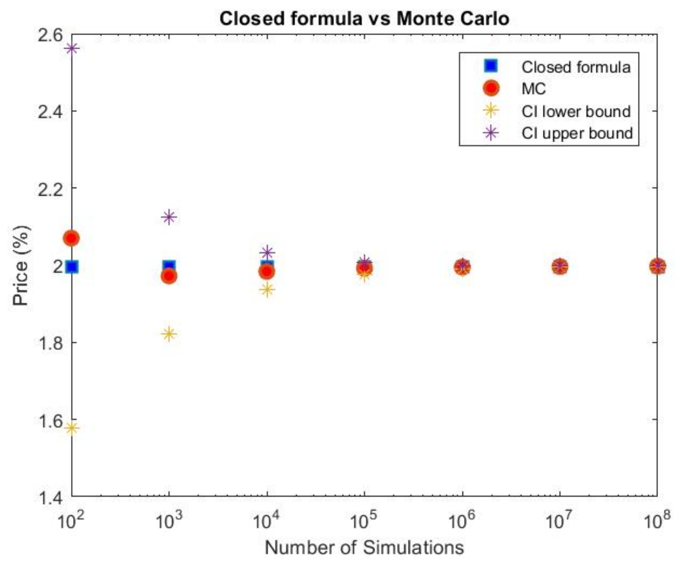

FIGURE 3.1: In the semilog plot with respect to X-axis blue squares represents closed formula prices, red circles Monte Carlo prices (increasingly closer to closed formula prices as the number of simulations grows) while violet and yellow stars respectively represent 95% confidence interval upper and lower limits (confidence interval's width decreases as the number of simulations grows). The number of simulations is $10^2, 10^3, .., 10^8$.

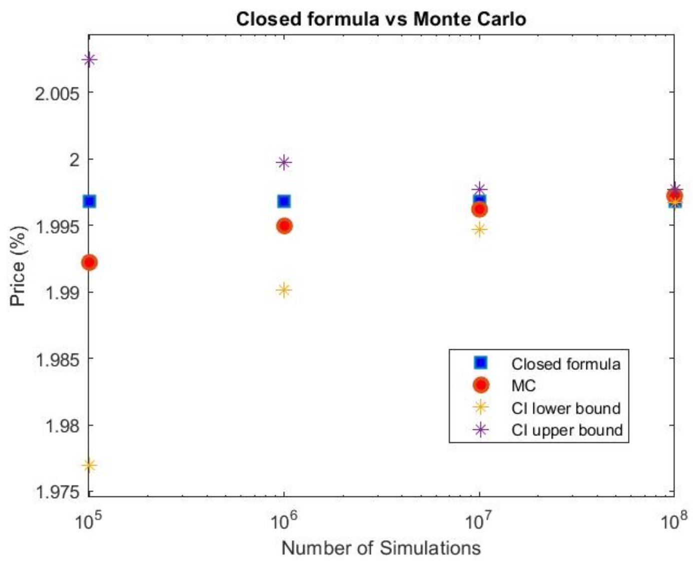

FIGURE 3.2: This chart is a zoom on the higher numbers of simulations of the *Figure 3.1* chart, in particular on $10^5, 10^6, 10^7, 10^8$ number of simulations.

creases as the simulation number grows meaning the increasing accuracy of Monte Carlo algorithm, closed formula price always falls into Monte Carlo confidence interval and the distance between the two prices drops as simulations grow. These signs validate and verify our closed formula, moreover, *Figure 3.3* and *Figure 3.4* show that prices' distance with one hundred million simulations is between 0.1 and 0.01 Basis Point (BP, where 1 BP= 0.01%) while the power scale factor of the absolute error is between 0.25 and 0.5.

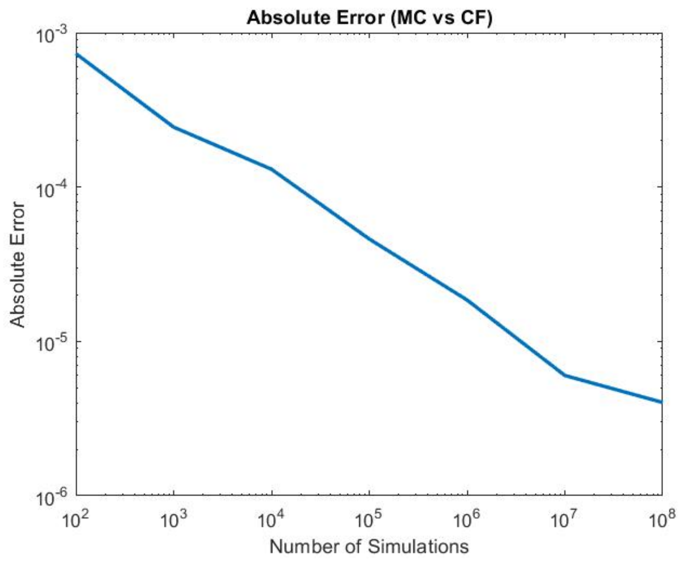

FIGURE 3.3: Logarithmic scale graphs of absolute error versus simulation steps. Number of simulations: $10^2, 10^3, .., 10^8$.

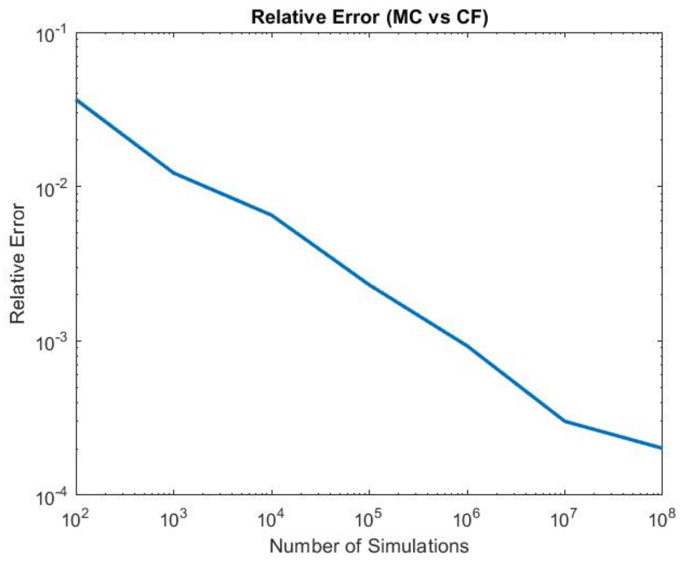

FIGURE 3.4: Logarithmic scale graphs of relative error versus simulation steps. Number of simulations: $10^2, 10^3, .., 10^8$.

Chapter 4

Calibration

In this chapter, we describe the models' calibration method in order to find the models' parameters that better reflect market condition, in particular, we calibrate both models on market conditions of 13 September 2012. For both models, we choose the same calibration cascade methodology, nevertheless in the final section of this chapter it will be presented, even if only introduced, a possible alternative algorithm to calibrate the EQM model.

4.1 Methodology

Cascade calibration algorithm consists of two steps:

- firstly we calibrate IR curves via dual-curve Crab's Bootstrap technique whose methodology was already explained in *Chapter 1*;

- then calibrate models' set of parameters on European At The Money (ATM) swaptions vs Euribor six months on the 10y-diagonal, i.e. considering the nine ATM swaption 1y9y, 2y8y, 3y7y,.., 7y3y, 8y2y, 9y1y, where the first number represents swaption's expiry while the second swap's tenor.

Market's provider such as Bloomberg does not provide directly swaptions' prices but their volatilities, so we must choose a benchmark model to obtain market swaptions' prices from market volatilities. Swaptions' volatilities could be lognormal and expressed in percentage or normal and expressed in BP, for lognormal volatilities the benchmark model is the *Black Model* while for the latter is the *Black Normal Model*.
In the dataset we consider we have lognormal volatilities. The swaption Black formula we consider for the payer and the receiver swaptions are respectively:

$$\begin{cases} P^{pswn}_{mkt}(t_0) = B(t_0,t_\alpha)BPV_{\alpha,\omega}(t_0)[S_{\alpha,\omega}(t_0)\Phi(d_1) - R\Phi(d_2)] \\ P^{rswn}_{mkt}(t_0) = B(t_0,t_\alpha)BPV_{\alpha,\omega}(t_0)[R\Phi(-d_2) - S_{\alpha,\omega}(t_0)\Phi(-d_1)] \end{cases} \tag{4.1}$$

where t_α represents option's expiry, R is the strike rate, $S_{\alpha,\omega}$ is the forward swap rate, $\sigma_{\alpha,\omega}$ are market's volatilities, $BPV_{\alpha,\omega}(t_0) = \sum_{i=\alpha}^{\omega} \delta(t_i,t_{i+1})B(t_0;t_\alpha,t_{i+1})$ is the forward Basis Point Value and:

$$\begin{cases} d_1 = \frac{ln\frac{S_{\alpha,\omega}(t_0)}{R} + \frac{\sigma^2_{\alpha,\omega}(t_\alpha - t_0)}{2}}{\sigma_{\alpha,\omega}\sqrt{t_\alpha - t_0}} \\ d_2 = d_1 - \sigma_{\alpha,\omega}\sqrt{t_\alpha - t_0} \end{cases} \tag{4.2}$$

Since we are dealing with ATM contract (4.1) and (4.2) simplifies to

$$\begin{cases} P^{swn}_{mkt}(t_0) = P^{pswn}_{mkt}(t_0) = P^{rswn}_{mkt}(t_0) = B(t_0, t_\alpha) BPV_{\alpha,\omega}(t_0) R[2\Phi(d) - 1] \\ d = \frac{\sigma_{\alpha,\omega}\sqrt{t_\alpha - t_0}}{2} \end{cases} \tag{4.3}$$

We minimize the squadre distance between swaption model and market prices in order to find the optimal choice of model set of parameters:

$$\mathbf{p}^{opt} = argmin_{\mathbf{p}\in\mathcal{P}} \sum_{i=1}^{9} (P^{swn,i}_{mod}(t_0; \mathbf{p}) - P^{swn,i}_{mkt}(t_0))^2 \tag{4.4}$$

where $\mathcal{P}$ represents the set of admissible parameters for the selected model (for example **H0** for EQM model) while **p** represents the vector of selected model parameters (i.e $\mathbf{p} = (a\ \sigma\ \gamma)'$ for MHJM model and $\mathbf{p} = (b_1\ b_2\ b_3\ \sigma_1\ \sigma_2\ \sigma_3\ \kappa)'$ for EQM model). Depending on the model we are calibrating for $P^{swn,i}_{mod}(t_0; \mathbf{p})$ it shall be used formula (2.9) or depending on the parameters cases (3.27), (3.28) or (3.29) from *Proposition 3*.

4.2 Numerical Results

In this section we show the results of the calibration cascade on MHJM and EQM model described in *Chapter 2* and *Chapter 3*.
For what concerns the first step in the cascade algorithm we refer to *Section 1.3* of *Chapter 1* for results (see *Table 1.1*, *Table 1.2* and *Figure 1.2*).
Regarding instead the second step in *Table 4.1* we present market volatilities of the 10y-diagonal swaptions considered while as Gaussian factors' initial condition we put $\boldsymbol{\psi} = (2.5\%\ 0\%\ 0\%)'$.
If a needed volatility datum was missing, we linearly interpolated between closer data in order to find it.

expiry	tenor	volatility (%)	strike rate (%)
1y	9y	40,4	2.0000
2y	8y	37.6	2.1989
3y	7y	35.1	2.4050
4y	6y	32.8	2.5952
5y	5y	30.8	2.7485
6y	4y	29.3	2.8471
7y	3y	27.7	2.9583
8y	2y	26.8	3.0251
9y	1y	26.3	3.1086

TABLE 4.1: 10y-diagonal quoted swaption market lognormal volatilities and strike rates on 13 September 2012, end of the day quote.

Due to the high number of parameters to calibrate for EQM model we decide to parallelized the code in order to improve the computational time and its performance, in fact as it can be seen in *Figure 4.1* and *Figure 4.2* the minimization algorithm required many iterations before finding the optimal parameters' set for EQM model while the MHJM model required very fewer iterations in order to find the functional minimum. Practical details about calibration will be addressed in the final chapter.
Both calibrations are stable for large classes of admissible starting points.

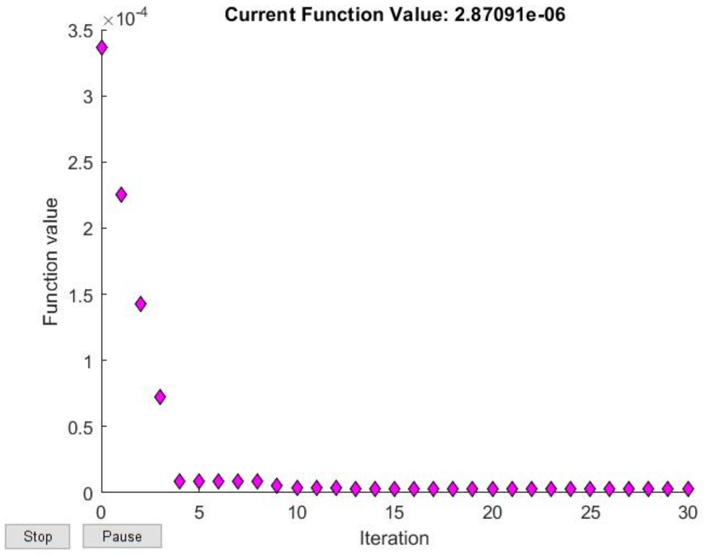

FIGURE 4.1: The chart show minimization steps implemented by the Matlab minimization function *fminsearch* for MHJM model

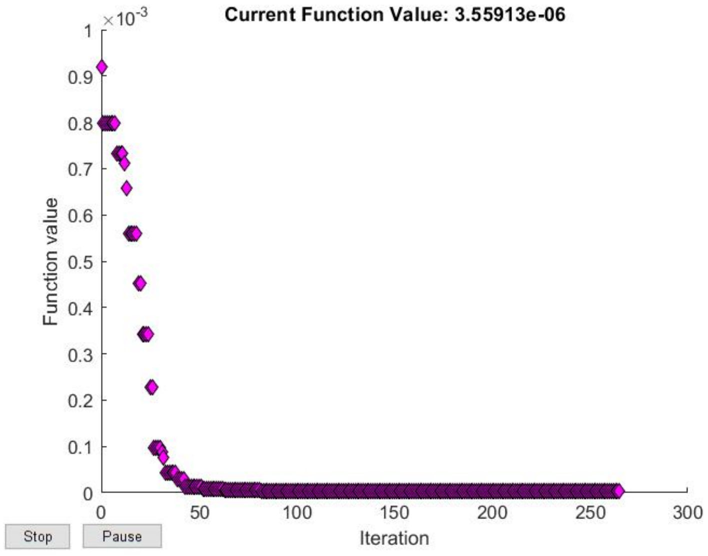

FIGURE 4.2: The chart show minimization steps implemented by the Matlab minimization function *fminsearch* for EQM model

In the following we list the optimal parameters we have found for MHJM (to the left) and EQM (to the right) models:

- $a = 7,9876\%$
- $\sigma = 1,1664\%$
- $\gamma = 0,0416\%$

- $b_1 = 10,01\%$
- $b_2 = 5,4069\%$
- $b_3 = 9,6461\%$
- $\sigma_1 = 2,9890\%$
- $\sigma_2 = 1,2062\%$
- $\sigma_3 = 0,0002\%$
- $\kappa = 0,0012\%$

To end this section we show in *Figure 4.3* and *Figure 4.4* the 10y-diagonal swaptions' market and model prices with optimal calibrated parameters' set $\mathbf{p}^{opt}$, as it can be seen the fitting between the curves results good in both cases.

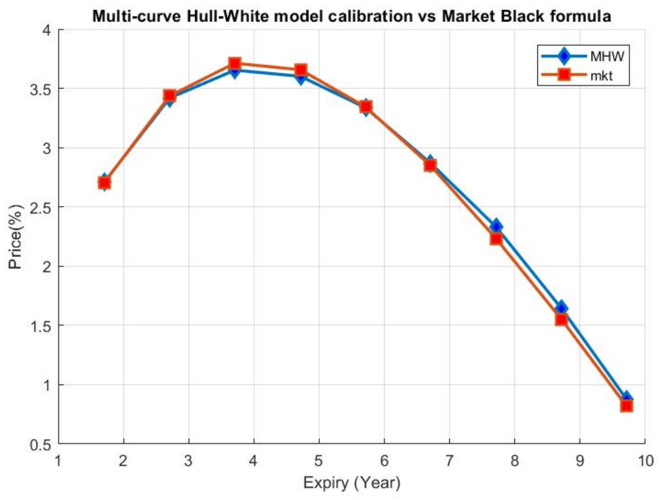

FIGURE 4.3: Market prices are represented as squares while the corrisponding MHJM model prices on the 10y diagonal are represented as diamonds

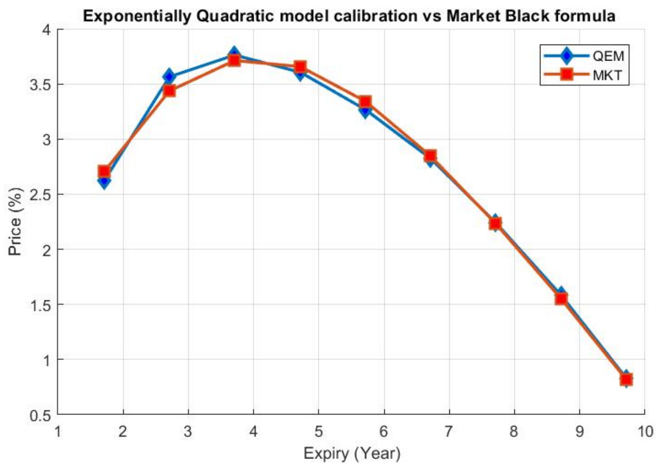

FIGURE 4.4: Market prices are represented as squares while the corrisponding EQM model prices on the 10y diagonal are represented as diamonds

4.3 Alternative EQM model parameters calibration

For the EQM model this kind of calibration we adopted could be problematic because model parameters are calibrated only through swaptions but they influence even bond prices $B(t,T)$ and swap prices (c.f. equations (3.8) and (3.23)), moreover we did not use the initial term structure $B(t_0,t)$ and $\bar{B}(t_0,t)$ given by Bootstrap. Consequently, our calibrated could not reflect the initial term structure obtained via the most liquid instruments of the market.
Hence we propose in this section an alternative way to calibrate this model, this

methodology follows a suggestion by K. Glau et al. (2016).
Since OIS swaps and FRAs are several orders of magnitude more liquid than swaptions in markets we could build the second step in the cascade calibration upon them. OIS swap prices in our model depend exclusively on b_i, σ_i with $i = 1, 2$ and via Bootstrap technique we have $B(t_0, t)$, so using (3.8) we should be able to calibrate four of the seven model parameters. For the remaining parameters b_3, σ_3 and κ we could calibrate them starting from FRA rates observable in the market, but we must firstly price FRA in our model framework. We start recalling the contract definition: the Forward Rate Agreement is a contract that allows the holder to lock at a generic date $t \leq T$ the interest rate between T and $T + \Delta$ at a fixed value R^{FRA} called FRA rate. Considering a unitary notional the FRA price under the $\mathbb{Q}^{T+\Delta}$ forward measure is:

$$\begin{aligned} P^{FRA}(t; T, T+\Delta, R^{FRA}) &= \delta(T, T+\Delta) B(t, T+\Delta) \mathbb{E}_t^{T+\Delta}[L(T, T+\Delta) - R^{FRA}] \\ &= B(t, T+\Delta) \mathbb{E}_t^{T+\Delta}\left[\frac{1}{\bar{B}(T, T+\Delta)} - (1 + \delta(T, T+\Delta) R^{FRA})\right] \\ &= B(t, T+\Delta)[\bar{v}_{t,T} - (1 + \delta(T, T+\Delta) R^{FRA})] \end{aligned} \tag{4.5}$$

where in the second equality we have used (3.1) and in the third one we have defined

$$\bar{v}_{t,T} = \mathbb{E}_t^{T+\Delta}\left[\frac{1}{\bar{B}(T, T+\Delta)}\right]$$

The fair FRA rate is found imposing the corresponding FRA price to zero getting from (4.5)

$$R^{FRA} = \frac{\bar{v}_{t,T} - 1}{\delta(T, T+\Delta)} \tag{4.6}$$

We are only left to find a closed formula for $\bar{v}_{t,T}$, so in analogy with what it was already done in *Chapter 3, Section 5* we write using (3.9), Gaussian factor independence and *Lemma 4 and 5*

$$\begin{aligned} \bar{v}_{t,T} &= e^{A(T,T+\Delta)} \mathbb{E}_t^{T+\Delta}\left[e^{(\kappa+1)B(T,T+\Delta)\Psi_T^1}\right] \mathbb{E}_t^{T+\Delta}\left[e^{C(T,T+\Delta)\left(\Psi_T^2\right)^2}\right] \mathbb{E}_t^{T+\Delta}\left[e^{\bar{C}(T,T+\Delta)\left(\Psi_T^3\right)^2}\right] \\ &= e^{A(T,T+\Delta)+\Gamma^1_{FRA}(t,T)+\Gamma^2_{FRA}(t,T)+\Gamma^3_{FRA}(t,T)-\rho^1_{FRA}(t,T)\Psi^1_t-\rho^2_{FRA}(t,T)(\Psi^2_t)^2-\rho^3_{FRA}(t,T)(\Psi^3_t)^2} \end{aligned} \tag{4.7}$$

where

$$\begin{cases} \rho^1_{FRA}(t,T) = -(\kappa+1)\mathcal{B}(T, T+\Delta)e^{-b_1(T-t)} \\ \Gamma^1_{FRA}(t,T) = \frac{\sigma_1^2}{2}\int_t^T (\rho^1_{FRA}(u,T))^2 du + \sigma_1^2 \int_t^T \mathcal{B}(u, T+\Delta)\rho^1(u,T)du \\ \rho^2_{FRA}(t,T) = \frac{C(T,T+\Delta)exp\{-2b_2(T-t)-4\sigma_2^2\int_t^T C(u,T+\Delta)du\}}{2\sigma_2^2 C(t,T+\Delta)\int_t^T exp\{-2b_2(T-w)-4\sigma_2^2\int_w^T C(u,T+\Delta)du\}dw-1} \\ \Gamma^2_{FRA}(t,T) = -\sigma_2^2 \int_t^T \rho^2_{FRA}(u,T)du \\ \rho^3_{FRA}(t,T) = \frac{-4b_3 h_k^{FRA} e^{-2b_3(T-t)}}{4\sigma_3^2 h_k^{FRA} e^{-2b_3(T-t)}-1}, \text{with } h_k^{FRA} = \frac{\bar{C}(T,T+\Delta)}{4\sigma_3^2\bar{C}(T,T+\Delta)-4b_3} \\ \Gamma^3_{FRA}(t,T) = -\sigma_3^2 \int_t^T \rho^3_{FRA}(u,T)du \end{cases}$$

Summing up the last three parameters of EQM model could be calibrated using (4.6) and (4.7).

Chapter 5

Matlab Code's Critical Issues

In this final chapter, we present the main criticality we met during code writing: computational time and code performance, in fact, while Crab's Bootstrap and MHJM model codes did not involve such problem we certainly cannot tell the same for EQM code. Hence we improve code performance using several techniques (e.g. vectorizing everything where feasible, minimizing the number of cycles and parallelizing the most time expensive cycles).
In the following, we present the three best improvements we apply to the code in order to save computational execution time.

5.1 Swaption pricing through Jamshidian approach

Most of the times **p** falls into *Case (1) or (2)*, so it is of fundamental importance that the relative code performs well, unfortunately the Jamshidian approach is also very heavy from a computational point of view since its execution involves the computation of several bidimensional integrals over a non-rectangular domain and the usage of numerical solver in order to find $z_i(x, y)$ and grids of this non-rectangular domains. In this section we take as an example *Case (1)* but everything will be said still hold for *Case (2)* and the improvements here described has been applied even in *Case (2)*.
Observing formula (3.27) in *Proposition 3* and considering a Euribor 6m swaption contract over a Y years tenor swap one can deduce that $6Y$ integrals need to be calculated in order to obtain the price.
The more advanced and updated function in Matlab for double integration is *integral2* that better performs in our framework with the tiled method, that requires the integration domain limits to be finite. Firstly we need to change our infinite integration domain limits, luckily these Lebesgue's integrals are expected values so in the integrand function always appear a Gaussian density function that quickly goes to zero departing from the distributional mean. We shall consider basically zero the integrand function when $|x| > 6$ or $|y| > 6$ with $x, y \sim \mathcal{N}(0, 1)$ and $\mathbb{P}(|x| > 6) \approx 0$. We remind that every integral variable is normalized (c.f. *Remark* 3).
The main problem of the Matlab integration function is the impossibility to preselect the integration grid outside the function itself, taken as an example *integral2*, for non-rectangular shape integration domain it wants finite quantities as the grid limits of the independent variable and vectorized function handle as limits for the dependent variable. As a direct consequence of that for every $6Y$ double integrals, we should call every time a numerical solver for computing the bidimensional grid and for every integrand function call recompute $z_i(x, y)$ although grids and z solutions are always the same!
We solve this issue by slightly modifying formula (3.27): using integrals additivity

property we bring the summation under the integral sign gathering everything under a unique double integral. In this way we are able to express the price as a sum of only two integrals, one with $\mathcal{M}$ and the latter with $\mathcal{M}^C$ domain, reducing considerably the number of calls for grid construction and z solution computation.
Practically speaking with this improvement on our machine, considering a nine years tenor, the computational time pass about from 90 seconds to only 3, a 3000% time improvement. Obviously, the computational time depend on swap tenor and approximately goes from some tenth of second with short swap tenor to $3 - 3.5$ seconds when we have a ten-year swap tenor.
In the following we attach the Matlab code concerning *Case (1)* swaption pricing where this issue is addressed while *Case (2)* pricing will be attached in *Appendix B* with the remaining parts of the code not already insert in the following section of this chapter.
P_1 integrand function

```
function f=fun1(x,y,Af,AF,Bf,BF,Cf,CF,Bt,Ct2,Ct3,D,alpha,beta,R,yF1y,gamma
    ,theta,g,h)

P0=-.25;
opt=optimset('Display','off','Algorithm','Levenberg-Marquardt');
z1=fsolve(@(z)g(x,y,z)-h(x,y),P0*ones(size(x)),opt);
z2=-z1;
d3=(z1-alpha(3))/sqrt(beta(3));
d4=(z2-alpha(3))/sqrt(beta(3));

freq=2;
f=0;
for i=1:length(Af)
    d1=(sqrt(1+2*beta(3)*Ct3(i))*z1-(alpha(3)-theta(i)*beta(3)))/sqrt(beta
    (3));
    d2=(sqrt(1+2*beta(3)*Ct3(i))*z2-(alpha(3)-theta(i)*beta(3)))/sqrt(beta
    (3));

    f=f+(D(i)*gamma(i)*exp(-Af(i)-Bt(i)*(alpha(1)+sqrt(beta(1))*x) -...
        Ct2(i)*(alpha(2)+sqrt(beta(2))*y).^2-x.^2/2-y.^2/2).*(normcdf(d1)+
    normcdf(-d2)) ...
        -exp(-Af(i)-Bf(i)*(alpha(1)+sqrt(beta(1))*x)-Cf(i)*(alpha(2)+sqrt(
    beta(2))*y).^2 -...
        x.^2/2-y.^2/2).*(normcdf(d3)+normcdf(-d4)))/(2*pi);
    if mod(i,freq)==0
        j=i/freq;
        f=f-R*yF1y(j)*exp(-AF(j)-BF(j)*(alpha(1)+sqrt(beta(1))*x) -...
            CF(j)*(alpha(2)+sqrt(beta(2))*y).^2-x.^2/2-y.^2/2).*(normcdf(
    d3)+normcdf(-d4)) ...
            /(2*pi);
    end
end

end
```

Appendices/fun1.m

Jamishidian Case (1) approach

```
function price=Jamshidian1(Af,AF,Bf,BF,Cf,CF,Bt,Ct2,Ct3,D,alpha,beta,R,Tm,
    Tf,TF,PD)

%% g and h definition
yF1y=yearfrac([Tm;TF(1:end-1)],TF,6);
freq=2;

g=@(x,y,z)0;
h=@(x,y)0;
for i=1:length(Tf)
    g=@(x,y,z)g(x,y,z)+D(i)*exp(-Af(i)-Bt(i)*(alpha(1)+sqrt(beta(1))*x)...
        -Ct2(i)*(alpha(2)+sqrt(beta(2))*y).^2-Ct3(i)*z.^2);
    h=@(x,y)h(x,y)+exp(-Af(i)-Bf(i)*(alpha(1)+sqrt(beta(1))*x) -...
        Cf(i)*(alpha(2)+sqrt(beta(2))*y).^2);
    if mod(i,freq)==0
        j=i/freq;
        h=@(x,y)h(x,y)+R*yF1y(j)*exp(-AF(j)-BF(j)*(alpha(1)+sqrt(beta(1))*
    x)...
            -CF(j)*(alpha(2)+sqrt(beta(2))*y).^2);
    end
end

%% grid construction
opt=optimset('Display','off','Algorithm','Levenberg-Marquardt');
x0=0;
c=6;

yExI=-c;
yExS=c;

MxI=-c;
MxS=@(y)max(MxI,min(c,fsolve(@(x)g(x,y,zeros(size(y)))-h(x,y),x0*ones(size
    (y)),opt)));

cMxS=c;
cMxI=@(y)min(cMxS,max(-c,fsolve(@(x)g(x,y,zeros(size(y)))-h(x,y),x0*ones(
    size(y)),opt)));

%% integral computation
theta=alpha(3)/beta(3)*(1-1./sqrt(1+2*beta(3)*Ct3));
gamma=exp(theta.^2*beta(3)/2-theta*alpha(3))./sqrt(1+2*beta(3)*Ct3);

f1=@(x,y)fun1(x,y,Af,AF,Bf,BF,Cf,CF,Bt,Ct2,Ct3,D,alpha,beta,R,yF1y,gamma,
    theta,g,h);
P1=integral2(@(y,x)f1(x,y),yExI,yExS,MxI,MxS,'method','tiled');

f2=@(x,y)fun2(x,y,Af,AF,Bf,BF,Cf,CF,Bt,Ct2,D,alpha,beta,R,yF1y,gamma);
P2=integral2(@(y,x)f2(x,y),yExI,yExS,cMxI,cMxS,'method','tiled');

%% price computation
price=PD*(P1+P2);

end
```

Appendices/Jamshidian1.m

P_2 integrand function

```
function    f=fun2(x,y,Af,AF,Bf,BF,Cf,CF,Bt,Ct2,D,alpha,beta,R,yF1y,gamma)

freq=2;
f=0;
for  i=1:length(Af)
     f=f+(D(i)*gamma(i)*exp(-Af(i)-Bt(i)*(alpha(1)+sqrt(beta(1))*x) -...
         Ct2(i)*(alpha(2)+sqrt(beta(2))*y).^2-x.^2/2-y.^2/2) -...
         exp(-Af(i)-Bf(i)*(alpha(1)+sqrt(beta(1))*x)-Cf(i)*(alpha(2)+sqrt(
     beta(2))*y).^2 -...
         x.^2/2-y.^2/2))/(2*pi);
     if mod(i,freq)==0
         j=i/freq;
         f=f-R*yF1y(j)*exp(-AF(j)-BF(j)*(alpha(1)+sqrt(beta(1))*x) -...
              CF(j)*(alpha(2)+sqrt(beta(2))*y).^2-x.^2/2-y.^2/2)/(2*pi);
     end
end

end
```

Appendices/fun2.m

5.2 Swaption pricing through triple integration

If the set of parameters don't fall into one of the two Jamishidian cases we must solve directly a triple integral (c.f. (3.29)).
The most recommended Matlab function is *integral3* that exactly like *integral2* gives the opportunity to choose between two different integration methods: tiled or iterated depending on integration extrema. But unlike before it is not empirically evident what method better performs, instead it seems to have a dependency upon the set of parameters. Therefore to make sure to always use the fastest integration method we rely on Matlab parallel computing toolbox, in particular on *parfeval*, *fetchNext* and *cancel* parallel functions. In fact thanks to *parfeval* we can execute in parallel both integration methods while *fetchNext* functions collects the first output of the two operations running in parallel and finally with *cancel* we eliminate the execution of the slower operation still running. In this way for every parameters' set for which triple integration is needed we are sure to get the result from the fastest integration method available in *integral3*. In the following, we attach the code we implemented.
Integrand function

```
function  f=fun4(g,h,x,y,z)

f=zeros(size(x));
iZ=isfinite(g(x,y,z)) & isfinite(h(x,y));

f(iZ)=max(g(x(iZ),y(iZ),z(iZ))-h(x(iZ),y(iZ)),0).*normpdf(x(iZ)).*normpdf(
    y(iZ))...
     .*normpdf(z(iZ));

end
```

Appendices/fun4.m

Integration methods

```
function price=parallelInt(f,PD,i)

if i==1
    c=6;
    price=PD*integral3(f,-c,c,-c,c,-c,c);
else
    price=PD*integral3(f,-Inf,Inf,-Inf,Inf,-Inf,Inf);
end

end
```

Appendices/parallelInt.m

Option pricing

```
function price=tripleIntegration(Af,AF,Bf,BF,Cf,CF,Bt,Ct2,Ct3,D,alpha,beta
    ,R,Tm,Tf,TF,PD)

%% g and h definition
yF1y=yearfrac([Tm;TF(1:end-1)],TF,6);
freq=2;
g=@(x,y,z)0;
h=@(x,y)0;
for i=1:length(Tf)
    g=@(x,y,z)g(x,y,z)+D(i)*exp(-Af(i)-Bt(i)*(alpha(1)+sqrt(beta(1))*x)...
        -Ct2(i)*(alpha(2)+sqrt(beta(2))*y).^2-Ct3(i)*(alpha(3)+sqrt(beta
    (3))*z).^2);
    h=@(x,y)h(x,y)+exp(-Af(i)-Bf(i)*(alpha(1)+sqrt(beta(1))*x) -...
        Cf(i)*(alpha(2)+sqrt(beta(2))*y).^2);
    if mod(i,freq)==0
        j=i/freq;
        h=@(x,y)h(x,y)+R*yF1y(j)*exp(-AF(j)-BF(j)*(alpha(1)+sqrt(beta(1))*
    x)...
            -CF(j)*(alpha(2)+sqrt(beta(2))*y).^2);
    end
end

f=@(x,y,z)fun4(g,h,x,y,z);

%%
p=gcp();
nGrid=2;
for i=1:nGrid
    v(i)=parfeval(p,@(i)parallelInt(f,PD,i),1,i);
end
[~,price]=fetchNext(v);
cancel(v)

end
```

Appendices/tripleIntegration.m

5.3 Calibration

Finally, we have improved the time performance of the most time spending process of all the code: calibration. In particular, we have parallelized the *for* cycle represented by the summation in (4.4) using Matlab *parfor*.
parfor allows to execute *for* in parallel, obviously, it requires the iterations to be independent of each other since they will be executed in a nondeterministic order. Loop

iterations are split between Matlab's workers, every worker is identified by a CPU physical core, so the more cores there are the faster the code is. We were working on a modest machine whose number of physical cores was two so we manage to approximately halve the computational time but our code if run on a machine with a higher number of cores would be even much faster. The best results in term of time would be reached on a PC with nine workers, in this case for a set of parameters the time spent to compute a square distance would be equal to the time spent to price the swaptions with the greater tenor.
The following is the code that computes the square distance between model and market prices, functions that appear in it are reported in *Appendix B*.
Square distance function

```
function errQ=objfun(b1,b2,b3,sigma1,sigma2,sigma3,k,IC,swnData,discCurve,
    MKTswnPrice)

%%
pFlag=1;
i=1;
while pFlag && i<=length(swnData.Tm)
    pFlag=modelParamDomain(b1,b2,b3,sigma1,sigma2,sigma3,k,swnData.Tm(i)
    ,...
        swnData.nSwn(i).Tf);
    i=i+1;
end

%%
if pFlag==0
    errQ=1e5;
else
    dist=zeros(size(swnData.Tm));
    t=swnData.t;
    Tm=swnData.Tm;
    nSwn=swnData.nSwn;
    R=swnData.R;
    parfor i=1:length(swnData.Tm)
        dist(i)=QEMswnPricer(b1,b2,b3,sigma1,sigma2,sigma3,k,t,...
            Tm(i),nSwn(i).Tf,nSwn(i).TF,R(i),discCurve,IC,1)-MKTswnPrice(i
    );
    end
    errQ=sum(dist.^2);
end

end
```

Appendices/objfun.m

Conclusion

The main purpose of this thesis has been the analysis of interest rate models finalized to find closed exact pricing formula for swaps and swaptions and in order for this model to be practically useful and not only a mere theoretical construction we proposed and applied a cascade calibration on both models.
The cascade calibration required as a first step an initial curves construction, we have then chosen the Mr. Crab's Bootstrap as dual curves construction algorithm. The second step required instead the calibration of the models parameters upon swaption instruments, so using the Black model as benchmark market model, through which obtain prices from market volatilities, and the two models' swaption pricing formulas we obtained for both model the optimal and calibrated sets of parameters by minimizing the prices' square distance.
For EQM model we have met time performance problem during the writing of the code, problem that we have overcome by a vectorization of the code and with the help of parallel computing tools.
Summing up, in the following bulleted list we would like to itemize the most important contributions of this thesis by referring for each of them to the relevant results:

- the revision of the EQM model framework introduced in K. Glau et al. (2016) by corrections of formulas, the addition of **H0** hypothesis and the relaxation of simplified hypotheses has led to find a closed exact formula for swaption contract (c.f. equations (3.27), (3.28) and (3.29) from *Proposition 3* in *Chapter 3 Section 3.6*);

- the validation of EQM closed formula by comparison with a Monte Carlo version of the formula (c.f. *Figure 3.1, 3.2, 3.3 and 3.4* in *Chapter 3 Section 3.8*);

- proposal and practical application of a cascade calibration methodology to both the EQM and MHJM models on market conditions of 13 September 2012 (c.f. *Figure 1.2* in *Chapter 1 Section 1.3* and *Figure 4.3, 4.4* in *Chapter 4 Section 4.2*);

- implementation of a Matlab library, optimized with parallel computing tools, to price swaptions for both EQM and MHJM models (closed formula and Monte Carlo for EQM) and to calibrate both models (c.f. *Chapter 4* and *Appendix B*).

Looking at calibration results the difference in term of computation time is evident between the MHJM three parameters model and the seven parameters EQM model, this is due to the higher dimensionality of the minimization problem and moreover to a greater computational cost of the swaption exact closed formula for the Exponentially Quadratic short rate model. Nevertheless, we must specify that this difference would be much thinner, the larger the number of PC cores would be, since the calibration code is written in parallel allowing greater performance on advanced PC.
Finally, we want to spend some last words speaking about the pros and cons of the model choice, in particular between parsimonious parameters model against

parameters-rich model. In fact, while obviously the latter is quite flexible and better adaptable to market conditions it requires more instruments on which to be calibrated, but postcrisis today markets are very often less liquid, hence the necessity to deal with parsimonious models that are easier to calibrate.

Appendix A

ODE solutions

In this appendix we will show in greater details how to find solution functions in (3.10), (3.20), (3.21) and (3.22) starting from their respective ODEs.

A.1 Section 3.3

Let us start finding the solution to the first ODE in (3.14). First of all, we have to change variables and rename the solution functions via

$$\mathcal{B}(t,T) = y(x),\ x = T - t \tag{A.1}$$

In this way we can rewrite (3.14) as

$$\begin{cases} y'(x) + b_1 y(x) - 1 = 0,\ x \in [0,T] \\ y(0) = 0 \end{cases} \tag{A.2}$$

where in this appendix with $'$ we indicate the derivative operator with respect to x. (A.2) is an nonhomogeneous first order ODE with constant particular solution and general solution easily obtained with separation of variables method whose solution is

$$y(x) = \frac{1 - e^{-b_1 x}}{b_1} \tag{A.3}$$

Switching back to the original set of variables from (A.3) one gets $\mathcal{B}(t,T)$ as it is shown in (3.10).
For what concerns $\bar{\mathcal{B}}(t,T)$ the general solution is exactly the same while the particular solution differs a little bit due to the different constant in the ODE, in fact using (A.1) one gets $\bar{\mathcal{B}}(t,T) = (1+\kappa)\mathcal{B}(t,T)$.
Before solving (3.13) observe that the two ODEs are almost the same since only constant coefficients vary, then we will only show how to find the solution to the first one. Applying the same change of variable shown in (A.1) we obtain

$$\begin{cases} y'(x) = 1 - 2b_2 y(x) - 2\sigma_2^2 y^2(x),\ x \in [0,T] \\ y(0) = 0 \end{cases}$$

that is a nonhomogeneous Riccati differential equation. To solve it we need to introduce another change of variable

$$y(x) = \frac{u'(x)}{2\sigma_2^2 u(x)}$$

obtaining a second order ODE on $u(x)$

$$\begin{cases} u''(x) + 2b_2 u'(x) - 2\sigma_2^2 u(x) = 0, \; x \in [0, T] \\ u(0) = 1 \\ u'(0) = 0 \end{cases}$$

whose solution can be easily computed

$$u(x) = \left(\frac{1}{2} - \frac{b_2}{h^2}\right) e^{(-b_2 - \frac{h^2}{2})x} + \left(\frac{1}{2} + \frac{b_2}{h^2}\right) e^{(-b_2 + \frac{h^2}{2})x}$$

where h^2 is defined in (3.10). Rolling back throught changes of variables one gets

$$\begin{cases} y(x) = \dfrac{\left(\frac{1}{2} - \frac{b_2}{h^2}\right)\left(-b_2 - \frac{h^2}{2}\right) + \left(\frac{1}{2} + \frac{b_2}{h^2}\right)\left(-b_2 + \frac{h^2}{2}\right) e^{h^2 x}}{2\sigma_2^2 \left[\left(\frac{1}{2} - \frac{b_2}{h^2}\right) + \left(\frac{1}{2} + \frac{b_2}{h^2}\right) e^{h^2 x}\right]} \\ C(t, T) = \dfrac{\left(\frac{1}{2} - \frac{b_2}{h^2}\right)\left(-b_2 - \frac{h^2}{2}\right) + \left(\frac{1}{2} + \frac{b_2}{h^2}\right)\left(-b_2 + \frac{h^2}{2}\right) e^{h^2 (T-t)}}{2\sigma_2^2 \left[\left(\frac{1}{2} - \frac{b_2}{h^2}\right) + \left(\frac{1}{2} + \frac{b_2}{h^2}\right) e^{h^2 (T-t)}\right]} \end{cases} \tag{A.4}$$

After some algebric operations from $C(t, T)$ in (A.4) it can be found $C(t, T)$ in (3.10). Regarding $A(t, T)$ and $\bar{A}(t, T)$ they can simply be found integrating (3.15) with respect to the first input over the interval $[t, T]$.

A.2 Section 3.5

Applying *Lemma 4* to the expected value in the first row of (3.20) one gets

$$\begin{cases} \rho_t^1(t, T_{k-1}^f) - b_1 \rho_t^1(t, T_{k-1}^f) = 0, \; \rho^1(T_{k-1}^f, T_{k-1}^f) = -(\kappa + 1) B_k \\ \Gamma_t^1(t, T_{k-1}^f) = -\sigma_1^2 \mathcal{B}(t, T_k^f) \rho^1(t, T_{k-1}^f) - \frac{\sigma_1^2}{2} (\rho^1(t, T_{k-1}^f))^2, \; \Gamma^1(T_{k-1}^f, T_{k-1}^f) = 0 \end{cases} \tag{A.5}$$

The solution in (3.20) of the second ODE in (A.5) is trivial so let us focus on the ODE on ρ^1. As before we must begin applying the usual change of variables

$$\rho^1(t, T_{k-1}^f) = y(x), \; x = T_{k-1}^f - t \tag{A.6}$$

obtaining

$$\begin{cases} y'(x) + b_1 y(x) = 0, \; x \in [0, T] \\ y(0) = -(\kappa + 1) B_k \end{cases} \tag{A.7}$$

(A.7) is an honogeneous first order differential equation with constant coefficients whose solution is

$$y(x) = -(\kappa + 1) B_k e^{-b_1 (T_{k-1}^f - t)} \tag{A.8}$$

resubstituting (A.6) from (A.8) we can obtain the solution to the first ODE in (A.5) that is the second member of (3.20).
If we instead apply *Lemma 5* to the expected value in (3.21) we would get

$$\begin{cases} \rho_t^2(t, T_{k-1}^f) - 2[b_2 + 2\sigma_2^2 C(t, T_k^f)] \rho_t^2(t, T_{k-1}^f) - 2\sigma_2^2 (\rho^2(t, T_{k-1}^f))^2 = 0, \; \rho^2(T_{k-1}^f, T_{k-1}^f) = -C_k \\ \Gamma_t^2(t, T_{k-1}^f) = \sigma_2^2 \rho^2(t, T_{k-1}^f), \; \Gamma^2(T_{k-1}^f, T_{k-1}^f) = 0 \end{cases} \tag{A.9}$$

As before let us focus on the first ODE in (A.9), applying (A.6) it becomes

$$\begin{cases} y'(x) + 2[b_2 + 2\sigma_2^2 C(x)]y(x) + 2\sigma_2^2 y^2(x) = 0, \ x \in [0, T] \\ y(0) = -C_k \end{cases}$$

a Bernoulli equation with non constant coefficients, it can be solved similarly to what has been done in previous section with $C(t, T)$. First of all we change variables according to $y(x) = \frac{u'(x)}{2\sigma_2^2 u(x)}$ obtaining

$$\begin{cases} u''(x) + 2[b_2 + 2\sigma_2^2 C(x)]u'(x) = 0, \ x \in [0, T] \\ u(0) = 1 \\ u'(0) = -2\sigma_2^2 C_k \end{cases}$$

This second order differential equation can be reducted to a first order one via $u'(x) = v(x)$

$$\begin{cases} v'(x) + 2[b_2 + 2\sigma_2^2 C(x)]v(x) = 0, \ x \in [0, T] \\ v(0) = -2\sigma_2^2 C_k \end{cases}$$

Solving this ODE and rolling back the changes of variable we get

$$\begin{cases} v(x) = -2\sigma_2^2 C_k e^{-2b_2 x - 4\sigma_2^2 \int_0^x C(s)ds} \\ u(x) = 1 - 2\sigma_2^2 C_k \int_0^x e^{-2b_2 w - 4\sigma_2^2 \int_0^w C(s)ds} dw \\ y(x) = \frac{C_k e^{-2b_2 x - 4\sigma_2^2 \int_0^x C(s)ds}}{2\sigma_2^2 C_k \int_0^x e^{-2b_2 w - 4\sigma_2^2 \int_0^w C(s)ds} dw - 1} \end{cases} \tag{A.10}$$

Finally applying back (A.6) change of variable form the last solution in (A.10) we can get the solution in (3.21).
We are only left to prove solutions in (3.22), the corresponding ODE are obtained via *Lemma 5* and are the followings

$$\begin{cases} \rho_t^3(t, T_{k-1}^f) - 2b_3\rho_t^3(t, T_{k-1}^f) - 2\sigma_3^2(\rho^2(t, T_{k-1}^f))^2 = 0, \ \rho^2(T_{k-1}^f, T_{k-1}^f) = -\bar{C}_k \\ \Gamma_t^3(t, T_{k-1}^f) = \sigma_3^2 \rho^3(t, T_{k-1}^f), \ \Gamma^3(T_{k-1}^f, T_{k-1}^f) = 0 \end{cases} \tag{A.11}$$

The second one does not deserve comments, while for the ODE in $\rho^3(t, T_{k-1}^f)$ following the same method as for $\rho^2(t, T_{k-1}^f)$ one obtains

$$\rho^3(t, T_{k-1}^f) = \frac{\bar{C}_k e^{-2b_3(T_{k-1}^f - t)}}{\frac{\sigma_3^2 \bar{C}_k}{b_3}\left(1 - e^{-2b_3(T_{k-1}^f - t)}\right) - 1} \tag{A.12}$$

The solution expressed in (3.22) is derived directly from (A.12) after some algebric calculation and the definition of h_k.

Appendix B

Matlab Code

This appendix shows the most relevant parts of the Matlab code made for this thesis. Parts of code already attached in *Chapter 5* won't be present in this appendix.

B.1 Crab's Bootstrap

With the aim of facilitating the reading of the codes, we create for every section a flowchart that intuitively explains the reading order of the script/functions.
In *crabBootstrap* script there is the main body of the boostrap technique.

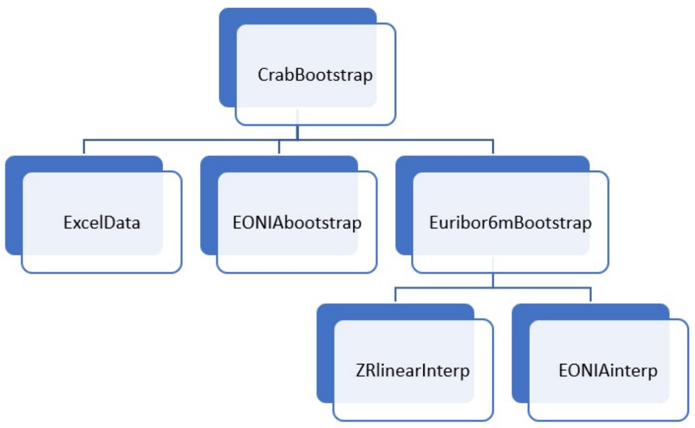

FIGURE B.1: Each block represents a script or a function, if two blocks are connected it means that in the script or function represented by the block at the top it exists a call of the script/function represented by the block below.

```
%% reading data
ExcelData;

%% curves construction
discCurve=EONIAbootstrap(dates,R);
pseudoDcurve=Euribor6mBootstrap(dates,R,discCurve);

%% results
hold all
plot(discCurve.knots(2:end),discCurve.rates(2:end),'-*')
plot(pseudoDcurve.knots(2:end),pseudoDcurve.rates(2:end),'-*')
grid on
dateFormat = 11;
datetick('x',dateFormat);
legend('EONIA','EU6m')
```

Appendices/crabBootstrap.m

This is the script used to read market data from Excel file.

```
%% file data
fileName='MarketData_Crab20120913.xls';
formatData='dd/mm/yyyy';

%% today date
[~,settlement]=xlsread(fileName,1,'H4');
dates.settlement=datenum(settlement,formatData);

%% EONIA data
[~,oisDates]=xlsread(fileName,1,'J12:J44');
oisDates=datenum(oisDates,formatData);
% spline interpolation to obtain yearly rates
dates.OIS=zeros(length(oisDates)+14,1);
j=29;
dates.OIS(1:j)=oisDates(1:j);
for i=1:2
    newDate=addtodate(dates.OIS(j+i-1),1,'year');
    dates.OIS(j+i)=newDate.*isbusday(newDate)+(1-isbusday(newDate)).*
   busdate(newDate,'modifiedfollow');
end
for k=1:3
    dates.OIS(j+i+1)=oisDates(29+k);
    j=j+i+1;
    for i=1:4
        newDate=addtodate(dates.OIS(j+i-1),1,'year');
        dates.OIS(j+i)=newDate.*isbusday(newDate)+(1-isbusday(newDate)).*
   busdate(newDate,'modifiedfollow');
    end
end
dates.OIS(end)=oisDates(end);

rates=xlsread(fileName,1,'F12:F44');
R.OIS=interp1(oisDates,rates,dates.OIS,'spline');

%% Euribor6m data
% deposits
[~,deposDates]=xlsread(fileName,3,'K20');
dates.depos=datenum(deposDates,formatData);
R.depos=xlsread(fileName,3,'F20');

% FRA
[~,FRAstartDates]=xlsread(fileName,3,'J21:J26');
[~,FRAdates]=xlsread(fileName,3,'K21:K26');
dates.startFRA=datenum(FRAstartDates,formatData);
dates.FRA=datenum(FRAdates,formatData);
R.FRA=xlsread(fileName,3,'F21:F26');

% swap
[~,swapDates]=xlsread(fileName,3,'K34:K48');
swapDates=datenum(swapDates,formatData);
% spline interpolation to obtain yearly rates
dates.swap=zeros(length(swapDates)+15,1);
j=10;
dates.swap(1:j)=swapDates(1:j);
newDate=addtodate(dates.swap(j),1,'year');
dates.swap(j+1)=newDate.*isbusday(newDate)+(1-isbusday(newDate)).*busdate(
    newDate,'modifiedfollow');
dates.swap(j+2)=swapDates(j+1);
```

```
j=j+2;
for i=1:2
    newDate=addtodate(dates.swap(j+i-1),1,'year');
    dates.swap(j+i)=newDate.*isbusday(newDate)+(1-isbusday(newDate)).*
    busdate(newDate,'modifiedfollow');
end
for k=1:3
    dates.swap(j+i+1)=swapDates(11+k);
    j=j+i+1;
    for i=1:4
        newDate=addtodate(dates.swap(j+i-1),1,'year');
        dates.swap(j+i)=newDate.*isbusday(newDate)+(1-isbusday(newDate)).*
    busdate(newDate,'modifiedfollow');
    end
end
dates.swap(end)=swapDates(end);

rates=xlsread(fileName,3,'F34:F48');
R.swap=interp1(swapDates,rates,dates.swap,'spline');

clear settlement oisDates deposDates FRAstartDates FRAdates swapDates k j
    i
```

Appendices/ExcelData.m

This is the script where discounting curve is built.

```
function discCurve=EONIAbootstrap(dates,R)

%% initialization
PD=zeros(size(dates.OIS));
maturities=yearfrac(dates.settlement,dates.OIS,3);
idx=find(maturities<=1,1,'last');

%% EONIA swap with expiry within 1y
yearF=yearfrac(dates.settlement,dates.OIS(1:idx),2); % EONIA SWAP act/360
PD(1:idx)=1./(1+yearF.*R.OIS(1:idx));

%% EONIA swap with expiry greater than 1y
periodYF=[yearF(end);yearfrac(dates.OIS(idx:end-1),dates.OIS(idx+1:end),2)
    ];
for i=idx+1:length(dates.OIS)
    PD(i)=(1-R.OIS(i)*sum(periodYF(1:i-idx).*PD(idx:i-1)))/(1+periodYF(i
    +1-idx)*R.OIS(i));
end

%% EONIA curve
discCurve.knots=[dates.settlement;dates.OIS];
discCurve.discounts=[1;PD];
discCurve.rates=[0;-log(PD)./maturities];
end
```

Appendices/EONIAbootstrap.m

In *Euribor6mBootstrap* there is the technique used to build the pseudo-discounting curve.

```
function pseudoDcurve=Euribor6mBootstrap(dates,R,discCurve)

%% 6m deposit
P=zeros(length(dates.FRA)+1,1);
P(end-1)=1/(1+yearfrac(dates.settlement,dates.depos,2)*R.depos);

%% forward pseudo discounts from FRA and 1y pseudo discounts computation
```

```
PF=1./(1+yearfrac(dates.startFRA,dates.FRA,2).*R.FRA);
P(end)=P(end-1)*PF(end);

%% pseudo discounts interpolation between 6m and 1y and pseudo discounts
    backward calculation
Pint=ZRlinearInterp(dates.settlement,[dates.depos;dates.FRA(end)],P(end-1:
    end),...
     dates.FRA(1:end-1));
P(1:end-2)=Pint./PF(1:end-1);

%% Euribor6m curve's update
pseudoDcurve.knots=[dates.startFRA(1:end-1);dates.depos;dates.FRA(end)];
pseudoDcurve.discounts=P;

%% 18m and 2y pseudo discounts computation
% initialization
freq=2;
Nyears=floor(yearfrac(dates.settlement,dates.swap(end),3));
[dCurve6m,dCurve1y]=EONIAinterp(discCurve,Nyears,freq);
yearF6m=yearfrac([discCurve.knots(1);dCurve6m.dates(1:end-1)],dCurve6m.
    dates,2);
yearF1y=yearfrac([discCurve.knots(1);dCurve1y.dates(1:end-1)],dCurve1y.
    dates,6);
yearF18=yearfrac([discCurve.knots(1);dCurve6m.dates(1)],...
     [dCurve6m.dates(1);dCurve6m.dates(3)],6);
% pseudo discounts interpolation and forward rates computation
Pint=ZRlinearInterp(dates.settlement,pseudoDcurve.knots,pseudoDcurve.
    discounts,...
    dCurve6m.dates(1:2));
F=zeros(freq*2,1);
F(1:2)=[(1/Pint(1)-1)/yearF6m(1);(Pint(1)/Pint(2)-1)/yearF6m(2)];
% evalutation of F(t0;t12,t18) and F(t0;t18,t24) strating from 18m and 2y
    swaps
I18m=R.swap(1)*(yearF18(1)*dCurve6m.discounts(1)+yearF18(2)*dCurve6m.
    discounts(3));
F(3)=(I18m-sum(yearF6m(1:2).*dCurve6m.discounts(1:2).*F(1:2)))/...
     (yearF6m(3)*dCurve6m.discounts(3));
I2y=R.swap(2)*sum(yearF1y(1:2).*dCurve1y.discounts(1:2));
F(4)=(I2y-I18m)/(yearF6m(4)*dCurve6m.discounts(4));
% forward pseudo discounts
PF=[1/(1+yearF6m(3)*F(3));prod(1./(1+yearF6m(3:4).*F(3:4)))];

%% Euribor6m curve's update
pseudoDcurve.knots=[pseudoDcurve.knots;dCurve6m.dates(3:4)];
pseudoDcurve.discounts=[pseudoDcurve.discounts;pseudoDcurve.discounts(end)
    *PF];

%% pseudo discounts computation from 2y up to 30y
% initialization
BPV=sum(yearF1y(1:2).*dCurve1y.discounts(1:2));
floatLeg=sum(yearF6m(1:4).*dCurve6m.discounts(1:4).*F);
yDist=floor(yearfrac(dates.settlement,dates.swap(2:end),3)); % years
    between market swap's expiries
P=zeros(length(yDist),1);
P(1)=pseudoDcurve.discounts(end);
opt=optimset('Display','off');
% computation
for i=1:length(yDist)-1
    BPV=BPV+sum(yearF1y(yDist(i)+1:yDist(i+1)).*dCurve1y.discounts(yDist(i
    )+1:yDist(i+1)));
     floatSum=@(p)0;
    % swap's forward rates calculation given unknown i+1-th pseudo
    discount
    for j=1:freq*(yDist(i+1)-yDist(i))
        Pi{j}=@(p)ZRlinearInterp(dates.settlement,dates.swap(i+1:i+2),[P(i
    );p],...
            dCurve6m.dates(freq*yDist(i)+j));
        if j==1
            Fint{j}=@(p)-log(Pi{j}(p)/P(i))/yearF6m(freq*yDist(i)+j);
        else
            Fint{j}=@(p)-log(Pi{j}(p)/Pi{j-1}(p))/yearF6m(freq*yDist(i)+j)
    ;
        end
        floatSum=@(p)floatSum(p)+yearF6m(freq*yDist(i)+j)*...
            dCurve6m.discounts(freq*yDist(i)+j)*Fint{j}(p);
    end
    % numerical evalutation of the unknown pseudo discounts
    NPV=@(p)floatLeg+floatSum(p)-R.swap(i+2)*BPV;
    p=fsolve(NPV,1,opt);
    % forward pseudo discounts and then pseudo discounts computation
    PF=1;
    for j=1:freq*(yDist(i+1)-yDist(i))
        PF=PF*1/(1+yearF6m(freq*yDist(i)+j).*Fint{j}(p));
    end
    P(i+1)=P(i)*PF;
    % cycle variable update
    floatLeg=floatLeg+floatSum(p);
end

%% Euribor6m curve's update
pseudoDcurve.knots=[dates.settlement;pseudoDcurve.knots;dates.swap(3:end)
    ];
pseudoDcurve.discounts=[1;pseudoDcurve.discounts;P(2:end)];
pseudoDcurve.rates=[0;-log(pseudoDcurve.discounts(2:end))./...
    yearfrac(dates.settlement,pseudoDcurve.knots(2:end),3)];
end
```

Appendices/Euribor6mBootstrap.m

The linear interpolation over zero rates is made in this function.

```
function P=ZRlinearInterp(t0,ti,Pi,t)

%% linear interpolation on zero rates
Zi=-log(Pi)./yearfrac(t0,ti,3);
Z=interp1(ti,Zi,t);
P=exp(-yearfrac(t0,t,3).*Z);
end
```

Appendices/ZRlinearInterp.m

EONIAinterp function generates two structs containing dates and discount factors with six months and one year lag.

```
function [dCurve6m,dCurve1y]=EONIAinterp(discCurve,Nyears,freq)

%% adding 6m dates and 6m discounts interpolation
dCurve6m.dates=zeros(freq*Nyears,1);
for i=1:freq*Nyears
    newDate=addtodate(discCurve.knots(1),12/freq*i,'month');
    dCurve6m.dates(i)=newDate.*isbusday(newDate)+(1-isbusday(newDate)).*
    busdate(newDate,'modifiedfollow');
end
dCurve6m.discounts=ZRlinearInterp(discCurve.knots(1),discCurve.knots(2:end
    ),...
    discCurve.discounts(2:end),dCurve6m.dates);
```

```
%% extrapolation of 1y dates and yearly discounts interpolation
dCurve1y.dates=dCurve6m.dates(freq*(1:Nyears));
dCurve1y.discounts=ZRlinearInterp(discCurve.knots(1),discCurve.knots(2:end
    ),...
    discCurve.discounts(2:end),dCurve1y.dates);
end
```

Appendices/EONIAinterp.m

B.2 Calibration and swaption pricing via MHM model

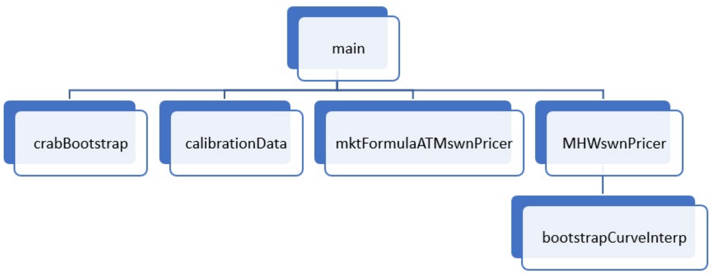

FIGURE B.2: MHJM model flowchart

In *main* script there is the main body of the calibration technique for MHJM model.

```
clear all
close all
clc

crabBootstrap
calibrationData;

%% Objective function definition
mktATMswnPrice=zeros(length(swnData.mktVol),1);
errQ=@(p)0;
for i=1:length(mktATMswnPrice)
    mktATMswnPrice(i)=mktFormulaATMswnPricer(swnData.t,swnData.Tm(i),
    swnData.nSwn(i).TF,...
        discCurve,swnData.mktVol(i),swnData.R(i));
    MHWswnPrice=@(p)MHWswnPricer(p(1),p(2),p(3),swnData.t,swnData.Tm(i)
    ,...
        swnData.nSwn(i).Tf,swnData.nSwn(i).TF,swnData.R(i),discCurve,
    pseudoDcurve);
    errQ=@(p)errQ(p)+(MHWswnPrice(p)-mktATMswnPrice(i)).^2;
end
paramDomain=@(p)(p(1)>=0) & (p(2)>0) & (p(3)>=0 & p(3)<=1);

%% Optimization
opt=optimset('Display','iter','PlotFcns',@optimplotfval);
p0=[.1,.01,0];
tic
optP=fminsearch(@(p)errQ(p).*paramDomain(p)+1e5*(1-paramDomain(p)),p0,opt)
    ;
toc

MHWswnPriceOpt=zeros(size(mktATMswnPrice));
```

```
for i=1:length(mktATMswnPrice)
    MHWswnPriceOpt(i)=MHWswnPricer(optP(1),optP(2),optP(3),swnData.t,
    swnData.Tm(i),...
        swnData.nSwn(i).Tf,swnData.nSwn(i).TF,swnData.R(i),discCurve,
    pseudoDcurve);
end

%% Resuts
hold all
plot(swnData.Tm,100*MHWswnPriceOpt,'-d','Markersize',7.5,'MarkerFaceColor'
    ,'b','Linewidth',2)
plot(swnData.Tm,100*mktATMswnPrice,'-s','Markersize',10,'MarkerFaceColor',
    'r','Linewidth',2)
grid on
dateFormat = 11;
datetick('x',dateFormat);
legend('MHW','mkt')
```

Appendices/mainM.m

This is the script used to read swaption market data from Excel file.

```
%% file data
fileName='EURSwaptionVol_091312.xlsx';
formatData='dd/mm/yyyy';

%% valutation date
[~,valDate]=xlsread(fileName,1,'D1');
valDate=datenum(valDate,formatData);
% settlement day convention
j=0;
swnData.t=valDate;
while j~=2
    swnData.t=swnData.t+1;
    if isbusday(swnData.t)
        j=j+1;
    end
end

%% initialization
N=9;
freq=2;
swnData.mktVol=zeros(N,1);
swnData.R=zeros(N,1);
swnData.Tm=zeros(N,1);
for i=1:N
    swnData.nSwn(i).Tf=zeros(freq*i,1);
    swnData.nSwn(i).TF=zeros(i,1);
end

%% creation of maturities & fix/floating payments date
every6Mdates=zeros(freq*N,1);
newDate=addtodate(swnData.t,1,'year');
swnData.Tm(1)=newDate.*isbusday(newDate)+(1-isbusday(newDate)).*busdate(
    newDate,'modifiedfollow');
for i=1:freq*N
    newDate=addtodate(swnData.Tm(1),12/freq*i,'month');
    every6Mdates(i)=newDate.*isbusday(newDate)+(1-isbusday(newDate)).*
    busdate(newDate,'modifiedfollow');
end

swnData.Tm(2:end)=every6Mdates(2:2:end-1);
for i=1:N
```

```
    swnData.nSwn(end-i+1).Tf=every6Mdates(end-2*i+1:end);
    swnData.nSwn(end-i+1).TF=swnData.nSwn(end-i+1).Tf(2:2:end);
end

%% reading market data
swnData.mktVol(1)=xlsread(fileName,9,'F10')*1e-2;
swnData.R(1)=xlsread(fileName,9,'F28')/100;

swnData.mktVol(2)=xlsread(fileName,8,'F12')*1e-2;
swnData.R(2)=xlsread(fileName,8,'F30')/100;

swnData.mktVol(3)=xlsread(fileName,7,'F13')*1e-2;
swnData.R(3)=xlsread(fileName,7,'F31')/100;

swnData.mktVol(4)=xlsread(fileName,6,'F14')*1e-2;
swnData.R(4)=xlsread(fileName,6,'F32')/100;

swnData.mktVol(5)=xlsread(fileName,5,'F15')*1e-2;
swnData.R(5)=xlsread(fileName,5,'F33')/100;

closeMktVol=xlsread(fileName,4,'F15:F16')*1e-2;
closeStrike=xlsread(fileName,4,'F33:F34')/100;
swnData.mktVol(6)=interp1([swnData.Tm(5);swnData.Tm(7)],closeMktVol,
    swnData.Tm(6));
swnData.R(6)=interp1([swnData.Tm(5);swnData.Tm(7)],closeStrike,swnData.Tm
    (6));

swnData.mktVol(7)=xlsread(fileName,3,'F16')*1e-2;
swnData.R(7)=xlsread(fileName,3,'F34')/100;

closeMktVol=xlsread(fileName,2,'F16:F17')*1e-2;
closeStrike=xlsread(fileName,2,'F35:F36')/100;
swnData.mktVol(8)=interp1([swnData.Tm(7);every6Mdates(end)],closeMktVol,
    swnData.Tm(8));
swnData.R(8)=interp1([swnData.Tm(7);every6Mdates(end)],closeStrike,swnData
    .Tm(8));

closeMktVol=xlsread(fileName,1,'F16:F17')*1e-2;
closeStrike=xlsread(fileName,1,'F34:F35')/100;
swnData.mktVol(9)=interp1([swnData.Tm(7);every6Mdates(end)],closeMktVol,
    swnData.Tm(9));
swnData.R(9)=interp1([swnData.Tm(7);every6Mdates(end)],closeStrike,swnData
    .Tm(9));

%% cleaning workspace
clear valDate j N freq i every6Mdates newDate closeMktVol closeStrike...
    fileName formatData
```

Appendices/calibrationData.m

mktFormulaATMswnPricer function computes ATM swaption Black formula and Black normal formula (we just need the Black formula).

```
function price=mktFormulaATMswnPricer(t,Tm,TF,discCurve,sigma,strike)

PD=ZRlinearInterp(t,discCurve.knots,discCurve.discounts,[Tm;TF]);
yearF1y=yearfrac([Tm;TF(1:end-1)],TF,6);
frwPD=PD(2:end)/PD(1);
BPV=sum(yearF1y.*frwPD);
if nargin==5 % Black normal model
    price=PD(1)*BPV*sigma*sqrt(yearfrac(t,Tm,3)/(2*pi));
else % Black model
    price=PD(1)*BPV*strike*(2*normcdf(sigma*sqrt(yearfrac(t,Tm,3))/2)-1);
```

```
end

end
```

Appendices/mktFormulaATMswnPricer.m

In this function we price a swaption with the methodology explained in *Chapter 2*.

```
function price=MHWswnPricer(a,sigma,gamma,t,Tm,Tf,TF,R,discCurve,
    pseudoDcurve)

[PD,P]=bootstrapCurvesInterp(discCurve,pseudoDcurve,t,Tm,Tf,TF);
frwPD.TmF=PD.TF/PD.Tm;
frwPD.Tmf=PD.Tf/PD.Tm;
frwPD.Tf=PD.Tf./[PD.Tm;PD.Tf(1:end-1)];
frwP=P.Tf./[P.Tm;P.Tf(1:end-1)];
spreadD=frwPD.Tf./frwP;

yearF1y=yearfrac([Tm;TF(1:end-1)],TF,6);
c=yearF1y*R;
c(end)=c(end)+1;

wa=c.*frwPD.TmF;
wb=frwPD.Tmf(1:end-1);
wc=spreadD.*[1;wb];

dt=yearfrac(t,Tm,3);
if a~=0
    V=(1-exp(-2*a*dt))/(2*a);
    v=@(t,T)sigma/a*(1-exp(-a*yearfrac(t,T,3)));
else
    V=dt;
    v=@(t,T)sigma*yearfrac(t,T,3);
end

lambda.TF=(1-gamma)*v(Tm,TF);
lambda.Tf=(1-gamma)*v(Tm,Tf(1:end-1));
nu=v(Tm,[Tm;Tf(1:end-1)])-gamma*v(Tm,Tf);

freq=2;
f=@(x)0;
for i=1:length(Tf)
    if i~=length(Tf)
        f=@(x)f(x)+wb(i)*exp(-lambda.Tf(i)*x-(lambda.Tf(i))^2*V/2);
    end
    f=@(x)f(x)-wc(i)*exp(-nu(i)*x-(nu(i))^2*V/2);
    if mod(i,freq)==0
        f=@(x)f(x)+wa(i/freq)*exp(-lambda.TF(i/freq)*x-(lambda.TF(i/freq))
    ^2*V/2);
    end
end
opt=optimset('Display','off','Tolfun',1e-12,'TolX',1e-8);
xi=fsolve(f,0,opt);

price=PD.Tm*(sum(wa.*normcdf(xi/sqrt(V)+sqrt(V)*lambda.TF))+...
    sum(wb.*normcdf(xi/sqrt(V)+sqrt(V)*lambda.Tf))-sum(wc.*normcdf(xi/sqrt
    (V)+sqrt(V)*nu)));
end
```

Appendices/MHWswnPricer.m

bootstrapCurvesInterp function interpolates along the necessary set of dates the discounting and pseudo-discounting curves obtained via Bootstrap.

```
function [PD,P]=bootstrapCurvesInterp(discCurve,pseudoDcurve,t,Tm,Tf,TF)

PD.TF=ZRlinearInterp(t,discCurve.knots,discCurve.discounts,TF);
PD.Tf=ZRlinearInterp(t,discCurve.knots,discCurve.discounts,Tf);
PD.Tm=ZRlinearInterp(t,discCurve.knots,discCurve.discounts,Tm);

P.Tf=ZRlinearInterp(t,pseudoDcurve.knots,pseudoDcurve.discounts,Tf);
P.Tm=ZRlinearInterp(t,pseudoDcurve.knots,pseudoDcurve.discounts,Tm);
end
```

Appendices/bootstrapCurvesInterp.m

B.3 Calibration and swaption pricing via EQM model

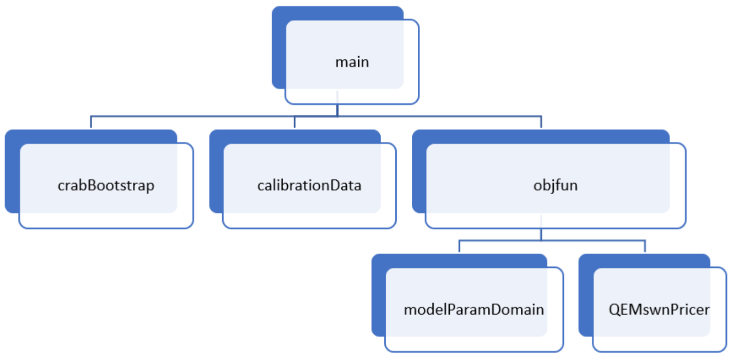

FIGURE B.3: EQM calibration flowchart

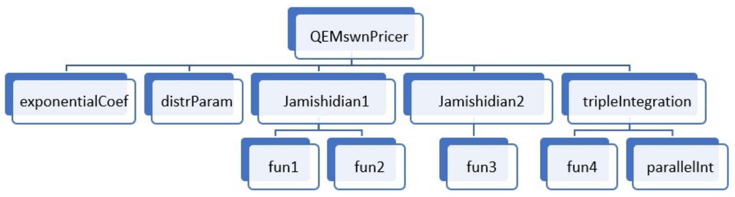

FIGURE B.4: EQM pricer flowchart

In *mainM* script there is the main body of the calibration technique for EQM model.

```
clear all
close all
clc

crabBootstrap
calibrationData;

%% Market prices
MKTswnPrice=zeros(size(swnData.Tm));
for i=1:length(swnData.mktVol)
    MKTswnPrice(i)=mktFormulaATMswnPricer(swnData.t,swnData.Tm(i),...
        swnData.nSwn(i).TF,discCurve,swnData.mktVol(i),swnData.R(i));
end
```

```
IC=[.025;0;0];
p0=[.1;.1;.1;.01;.01;.01;.5];

%% Optimization
opt=optimset('Display','iter','PlotFcns',@optimplotfval);
tic
optP=fminsearch(@(p)objfun(p(1),p(2),p(3),p(4),p(5),p(6),p(7),IC,swnData,
    discCurve,MKTswnPrice),p0,opt);
toc

%% Results
QEMswnPriceOpt=zeros(size(MKTswnPrice));
for i=1:length(MKTswnPrice)
    QEMswnPriceOpt(i)=QEMswnPricer(optP(1),optP(2),optP(3),optP(4),...
        optP(5),optP(6),optP(7),swnData.t,swnData.Tm(i),swnData.nSwn(i).Tf
    ,...
        swnData.nSwn(i).TF,swnData.R(i),discCurve,IC);
end
hold all
plot(swnData.Tm,100*QEMswnPriceOpt,'-d','Markersize',7.5,'MarkerFaceColor'
    ,'b','Linewidth',2)
plot(swnData.Tm,100*MKTswnPrice,'-s','Markersize',10,'MarkerFaceColor','r'
    ,'Linewidth',2)
grid on
dateFormat = 11;
datetick('x',dateFormat);
legend('QEM','MKT')
```

Appendices/main.m

We define the objective function to minimize in this Matlab function (observe that if a given set of parameters does not satisfy model constrains then the value of the square distance is set to 100000).

```
function errQ=objfun(b1,b2,b3,sigma1,sigma2,sigma3,k,IC,swnData,discCurve,
    MKTswnPrice)

%%
pFlag=1;
i=1;
while pFlag && i<=length(swnData.Tm)
    pFlag=modelParamDomain(b1,b2,b3,sigma1,sigma2,sigma3,k,swnData.Tm(i)
    ,...
        swnData.nSwn(i).Tf);
    i=i+1;
end

%%
if pFlag==0
    errQ=1e5;
else
    dist=zeros(size(swnData.Tm));
    t=swnData.t;
    Tm=swnData.Tm;
    nSwn=swnData.nSwn;
    R=swnData.R;
    parfor i=1:length(swnData.Tm)
        dist(i)=QEMswnPricer(b1,b2,b3,sigma1,sigma2,sigma3,k,t,...
            Tm(i),nSwn(i).Tf,nSwn(i).TF,R(i),discCurve,IC,1)-MKTswnPrice(i
    );
    end
    errQ=sum(dist.^2);
end
```

```
end
```

Appendices/objfun.m

In this function we price a swaption with the methodology explained in *Chapter 3*.

```
function price=QEMswnPricer(b1,b2,b3,sigma1,sigma2,sigma3,k,t,Tm,...
     Tf,TF,R,discCurve,IC,calFlag)

if nargin<15
     if modelParamDomain(b1,b2,b3,sigma1,sigma2,sigma3,k,Tm,Tf)

          [Af,AF,Bf,BF,Cf,CF,Bt,Ct2,Ct3,D]=exponentialCoef(b1,b2,b3,sigma1,
     sigma2,...
               sigma3,k,Tm,Tf,TF);

          [alpha,beta]=distrParam(IC,b1,b2,b3,sigma1,sigma2,sigma3,t,Tm,Tm);

          PD=ZRlinearInterp(t,discCurve.knots,discCurve.discounts,Tm);

          sFlag=scenarioDef(b3,sigma3,t,Tm,Tf);

          if sFlag==1
               price=Jamshidian1(Af,AF,Bf,BF,Cf,CF,Bt,Ct2,Ct3,D,alpha,beta,R,
     Tm,Tf,TF,PD);
          elseif sFlag==2
               price=Jamshidian2(Af,AF,Bf,BF,Cf,CF,Bt,Ct2,Ct3,D,alpha,beta,R,
     Tm,Tf,TF,PD);
          else
               price=tripleIntegration(Af,AF,Bf,BF,Cf,CF,Bt,Ct2,Ct3,D,alpha,
     beta,R,Tm,Tf,TF,PD);
          end
     else
          price=-1;
     end
else
     [Af,AF,Bf,BF,Cf,CF,Bt,Ct2,Ct3,D]=exponentialCoef(b1,b2,b3,sigma1,
     sigma2,...
          sigma3,k,Tm,Tf,TF);

     [alpha,beta]=distrParam(IC,b1,b2,b3,sigma1,sigma2,sigma3,t,Tm,Tm);

     PD=ZRlinearInterp(t,discCurve.knots,discCurve.discounts,Tm);

     sFlag=scenarioDef(b3,sigma3,t,Tm,Tf);

     if sFlag==1
          price=Jamshidian1(Af,AF,Bf,BF,Cf,CF,Bt,Ct2,Ct3,D,alpha,beta,R,Tm,
     Tf,TF,PD);
     elseif sFlag==2
          price=Jamshidian2(Af,AF,Bf,BF,Cf,CF,Bt,Ct2,Ct3,D,alpha,beta,R,Tm,
     Tf,TF,PD);
     else
          price=tripleIntegration(Af,AF,Bf,BF,Cf,CF,Bt,Ct2,Ct3,D,alpha,beta,
     R,Tm,Tf,TF,PD);
     end
end

end
```

Appendices/QEMswnPricer.m

In this function we check if the set of model parameters satisfies **H0** hypothesis.

```
function flag=modelParamDomain(b1,b2,b3,sigma1,sigma2,sigma3,k,Tm,Tf)

flag=(b1>0) && (b2>0) && (b3>0) && (sigma1>0) && (sigma2>0) && (sigma3>0)
    &&...
    (k>=0 && k<=1);

h2=sqrt(4*b2^2+8*sigma2^2); % (3.5)
dt=@(t,T)yearfrac(t,T,3);
C2=@(t,T)2*(exp(h2*dt(t,T))-1)./(2*h2+(2*b2+h2)*(exp(h2*dt(t,T))-1)); %
    (3.5)
C2int=@(t,T,TM)2*(2*log((2*b2*(exp(h2*dt(T,TM))-1)+h2*(exp(h2*dt(T,TM))+1)
    )./...
    (2*b2*(exp(h2*dt(t,TM))-1)+h2*(exp(h2*dt(t,TM))+1)))+(2*b2+h2)*dt(t,T)
    )/...
    ((2*b2+h2)*(2*b2-h2));
fSing2=@(t,T,TM)integral(@(u)exp(-2*b2*dt(u,T)-4*sigma2^2*C2int(u,T,TM)),t
    ,T)/365-...
    1/(2*sigma2^2*C2(T,TM));

h3=sqrt(4*b3^2+8*sigma3^2); % (3.5)
C3=@(t,T)2*(exp(h3*dt(t,T))-1)./(2*h3+(2*b3+h3)*(exp(h3*dt(t,T))-1));
hk=@(t,T)C3(t,T)./(4*sigma3^2*C3(t,T)-4*b3); % (4.25)
sing3=@(t,T)t-log(4*sigma3^2*hk(t,T))/(2*b3);

date=[Tm;Tf(1:end-1)];
opt=optimset('Display','off');
i=1;
while flag && i<=length(date)
    sing2=fsolve(@(t)fSing2(t,date(i),Tf(i)),0,opt);
    if (Tm<sing2 && sing2<date(i)) || (hk(date(i),Tf(i))>0 && Tm<sing3(
    date(i),Tf(i))...
            && sing3(date(i),Tf(i))<date(i))
        flag=0;
    end
    i=i+1;
end

end
```

Appendices/modelParamDomain.m

In *exponentialCoef* we compute coefficients in (3.24).

```
function [Af,AF,Bf,BF,Cf,CF,Bt,Ct2,Ct3,D]=exponentialCoef(b1,b2,b3,sigma1,
    sigma2,...
    sigma3,k,Tm,Tf,TF)

%% A
h2=sqrt(4*b2^2+8*sigma2^2); % (3.5)
dt=@(t,T)yearfrac(t,T,3);
C2int=@(t,T,TM)2*(2*log((2*b2*(exp(h2*dt(T,TM))-1)+h2*(exp(h2*dt(T,TM))+1)
    )./...
    (2*b2*(exp(h2*dt(t,TM))-1)+h2*(exp(h2*dt(t,TM))+1)))+(2*b2+h2)*dt(t,T)
    )/...
    ((2*b2+h2)*(2*b2-h2));
BquadInt=@(t,T)(2*b1*dt(t,T)-exp(-2*b1*dt(t,T))+4*exp(-b1*dt(t,T))-3)/(2*
    b1^3);
A=@(t,T)sigma2^2*C2int(t,T,T)-sigma1^2/2*BquadInt(t,T);

Af=A(Tm,Tf);
AF=A(Tm,TF);

%% B
```

```
B=@(t,T)(1-exp(-b1*dt(t,T)))/b1; % (3.6)

Bf=B(Tm,Tf);
BF=B(Tm,TF);

%% C
C2=@(t,T)2*(exp(h2*dt(t,T))-1)./(2*h2+(2*b2+h2)*(exp(h2*dt(t,T))-1)); %
    (3.5)

Cf=C2(Tm,Tf);
CF=C2(Tm,TF);

%% Bt
rho1=@(t,T,TM)-(k+1)*B(T,TM).*exp(-b1*dt(t,T)); % (4.19)

Bt=Bf+rho1(Tm,[Tm;Tf(1:end-1)],Tf);

%% Ct2
date=[Tm;Tf(1:end-1)];
rho2=zeros(size(date));
for i=1:length(date)
    I=integral(@(u)exp(-2*b2*dt(u,date(i))-4*sigma2^2*C2int(u,date(i),Tf(i
    ))),Tm,date(i))...
        /365;
    rho2(i)=C2(date(i),Tf(i))*exp(-2*b2*dt(Tm,date(i))-4*sigma2^2*C2int(Tm
    ,date(i),Tf(i)))/...
        (2*sigma2^2*C2(date(i),Tf(i))*I-1);
end

Ct2=Cf+rho2;

%% Ct3
h3=sqrt(4*b3^2+8*sigma3^2); % (3.5)
C3=@(t,T)2*(exp(h3*dt(t,T))-1)./(2*h3+(2*b3+h3)*(exp(h3*dt(t,T))-1));
hk=@(t,T)C3(t,T)./(4*sigma3^2*C3(t,T)-4*b3); % (4.25)
rho3=@(t,T,TM)-4*b3*hk(T,TM).*exp(-2*b3*dt(t,T))./(4*sigma3^2*hk(T,TM)
    .*...
    exp(-2*b3*dt(t,T))-1); % (4.25)-(5.24)

Ct3=rho3(Tm,[Tm;Tf(1:end-1)],Tf);

%% D
% At
C3int=@(t,T,TM)2*(2*log((2*b3*(exp(h3*dt(T,TM))-1)+h3*(exp(h3*dt(T,TM))+1)
    )./...
    (2*b3*(exp(h3*dt(t,TM))-1)+h3*(exp(h3*dt(t,TM))+1)))+(2*b3+h3)*dt(t,T)
    )/...
    ((2*b3+h3)*(2*b3-h3));
At=@(t,T)sigma2^2*C2int(t,T,T)+sigma3^2*C3int(t,T,T)-sigma1^2/2*(1+k)^2*
    BquadInt(t,T); % (3.7)-(5.24)

% gamma1
rhoQuadInt=@(t,T,TM)(k+1)^2*(B(T,TM)).^2.*(1-exp(-2*b1*dt(t,T)))/(2*b1); %
     general integral of (rho1(t,Tk-1))^2
BrhoInt=@(t,T,TM)(k+1)*B(T,TM).*(exp(-b1*dt(T,TM))-exp(-b1*(dt(t,T)+dt(t,
    TM)))...
    +2*exp(-b1*dt(t,T))-2)/(2*b1^2); % general integral of B1(t,Tk)*rho1(t
    ,Tk-1)
gamma1=@(t,T,TM)sigma1^2/2*rhoQuadInt(t,T,TM)+sigma1^2*BrhoInt(t,T,TM); %
    (4.19)

% gamma2
gamma2=zeros(size(Tf));
```

```
for i=1:length(Tf)
    gamma2(i)=-sigma2^2*integral(@(t)rho2fun(b2,sigma2,t,date(i),Tf(i)),Tm
    ,date(i))/365;
end

% gamma3
rho3int=@(t,T,TM)(log((b3*exp(2*b3*dt(t,T))-sigma3^2*C3(T,TM).*(exp(2*b3*
    dt(t,T))-1))/b3)...
    -2*b3*dt(t,T))/(2*sigma3^2);
gamma3=@(t,T,TM)-sigma3^2*rho3int(t,T,TM);

D=exp(At([Tm;Tf(1:end-1)],Tf)+gamma1(Tm,[Tm;Tf(1:end-1)],Tf)+gamma2+...
    gamma3(Tm,[Tm;Tf(1:end-1)],Tf));

end
```

Appendices/exponentialCoef.m

Since in ρ^2 expression there is an integrand function whose primitive is analitically impossible to find we need to compute the integral numerically, but even Γ^2, that is the integral of ρ^2, has to be computed numerically, so in order to use Matlab *integral* function inside itself we need another function to serve as a function handle representing ρ^2.

```
function f=rho2fun(b2,sigma2,t,Tk,Tkk)

h2=sqrt(4*b2^2+8*sigma2^2); % (3.5)
dt=@(t,T)yearfrac(t,T,3);
C2=@(t,T)2*(exp(h2*dt(t,T))-1)./(2*h2+(2*b2+h2)*(exp(h2*dt(t,T))-1)); %
    (3.5)
C2int=@(t,T,TM)2*(2*log((2*b2*(exp(h2*dt(T,TM))-1)+h2*(exp(h2*dt(T,TM))+1)
    )./...
    (2*b2*(exp(h2*dt(t,TM))-1)+h2*(exp(h2*dt(t,TM))+1)))+(2*b2+h2)*dt(t,T)
    )/...
    ((2*b2+h2)*(2*b2-h2));
I=@(t)integral(@(u)exp(-2*b2*dt(u,Tk)-4*sigma2^2*C2int(u,Tk,Tkk)),t,Tk)
    /365;

f=zeros(size(t));
for i=1:length(t)
    f(i)=C2(Tk,Tkk)*exp(-2*b2*dt(t(i),Tk)-4*sigma2^2*C2int(t(i),Tk,Tkk))
    /...
        (2*sigma2^2*C2(Tk,Tkk)*I(t(i))-1);
end

end
```

Appendices/rho2fun.m

In this function we compute means and variances as in (3.17).

```
function [alpha,beta]=distrParam(IC,b1,b2,b3,sigma1,sigma2,sigma3,t,T,TM)

alpha=zeros(size(IC));
beta=zeros(size(alpha));
dt=@(t,T)yearfrac(t,T,3);

%% factor 1
alpha(1)=exp(-b1*dt(t,T))*(IC(1)-(sigma1/b1)^2*(exp(b1*dt(t,T))-1)+(sigma1
    /b1)^2/2*...
    (exp(b1*(dt(TM,T)+dt(t,T)))-exp(b1*dt(TM,t))));
beta(1)=sigma1^2/(2*b1)*(1-exp(-2*b1*dt(t,T)));

```

```
%% factor 2
h2=sqrt(4*b2^2+8*sigma2^2); % (3.5)
C2int=@(t,T,TM)2*(2*log((2*b2*(exp(h2*dt(T,TM))-1)+h2*(exp(h2*dt(T,TM))+1)
    )./...
    (2*b2*(exp(h2*dt(t,TM))-1)+h2*(exp(h2*dt(t,TM))+1)))+(2*b2+h2)*dt(t,T)
    )/...
    ((2*b2+h2)*(2*b2-h2));
I=integral(@(u)exp(2*b2*dt(t,u)+4*sigma2^2*C2int(t,u,TM)),t,T)/365;

alpha(2)=IC(2)*exp(-b2*dt(t,T)-2*sigma2^2*C2int(t,T,TM));
beta(2)=sigma2^2*exp(-2*b2*dt(t,T)-4*sigma2^2*C2int(t,T,TM))*I;

%% factor 3
alpha(3)=IC(3)*exp(-b3*dt(t,T));
beta(3)=(sigma3)^2/(2*b3)*(1-exp(-2*b3*dt(t,T)));

end
```

Appendices/distrParam.m

This function returns one, two or zero if respectively the set parameters fall into *Case (1)*, *Case (2)* or the triple integral scenario.

```
function flag=scenarioDef(b3,sigma3,t,Tm,Tf)

h3=sqrt(4*b3^2+8*sigma3^2); % (3.5)
dt=@(t,T)yearfrac(t,T,3);
C3=@(t,T)2*(exp(h3*dt(t,T))-1)./(2*h3+(2*b3+h3)*(exp(h3*dt(t,T))-1));
hk=@(t,T)C3(t,T)./(4*sigma3^2*C3(t,T)-4*b3); % (4.25)
hkDis=@(t,T)exp(2*b3*dt(t,T))/(4*sigma3^2);
date=[Tm;Tf(1:end-1)];

if hk(date,Tf)<0 | hk(date,Tf)>hkDis(t,date) % Exponential parabola+Add hp
    met
    flag=1;
elseif hk(date,Tf)>0 & hk(date,Tf)<hkDis(Tm,date) % Gaussian bell
    flag=2;
else
    flag=0;
end

end
```

Appendices/scenarioDef.m

Jamshidian approach in scenario *Case (2)* (*Jamshidian1* function, *tripleIntegration* and the functions called inside them were already attached in *Chapter 5*).

```
function price=Jamshidian2(Af,AF,Bf,BF,Cf,CF,Bt,Ct2,Ct3,D,alpha,beta,R,Tm,
    Tf,TF,PD)

%% g and h definition
yF1y=yearfrac([Tm;TF(1:end-1)],TF,6);
freq=2;

g=@(x,y,z)0;
h=@(x,y)0;
for i=1:length(Tf)
    g=@(x,y,z)g(x,y,z)+D(i)*exp(-Af(i)-Bt(i)*(alpha(1)+sqrt(beta(1))*x)...
        -Ct2(i)*(alpha(2)+sqrt(beta(2))*y).^2-Ct3(i)*z.^2);
    h=@(x,y)h(x,y)+exp(-Af(i)-Bf(i)*(alpha(1)+sqrt(beta(1))*x) -...
        Cf(i)*(alpha(2)+sqrt(beta(2))*y).^2);
    if mod(i,freq)==0
        j=i/freq;
        h=@(x,y)h(x,y)+R*yF1y(j)*exp(-AF(j)-BF(j)*(alpha(1)+sqrt(beta(1))*
    x)...
            -CF(j)*(alpha(2)+sqrt(beta(2))*y).^2);
    end
end

%% grid construction
opt=optimset('Display','off','Algorithm','Levenberg-Marquardt');
x0=0;
c=6;

yExI=-c;
yExS=c;

cMxS=c;
cMxI=@(y)min(cMxS,max(-c,fsolve(@(x)g(x,y,zeros(size(y)))-h(x,y),x0*ones(
    size(y)),opt)));

%% integrals computation
theta=alpha(3)/beta(3)*(1-1./sqrt(1+2*beta(3)*Ct3));
gamma=exp(theta.^2*beta(3)/2-theta*alpha(3))./sqrt(1+2*beta(3)*Ct3);

f2=@(x,y)fun3(x,y,Af,AF,Bf,BF,Cf,CF,Bt,Ct2,Ct3,D,alpha,beta,R,yF1y,gamma,
    theta,g,h);
P2=integral2(@(y,x)f2(x,y),yExI,yExS,cMxI,cMxS,'method','tiled');

%% price computation
price=PD*P2;

end
```

Appendices/Jamshidian2.m

Here we define the function to be integrated for *Case (2)*.

```
function f=fun3(x,y,Af,AF,Bf,BF,Cf,CF,Bt,Ct2,Ct3,D,alpha,beta,R,yF1y,gamma
    ,theta,g,h)

P0=-.25;
opt=optimset('Display','off','Algorithm','Levenberg-Marquardt');
z1=fsolve(@(z)g(x,y,z)-h(x,y),P0*ones(size(x)),opt);
z2=-z1;
d3=(z1-alpha(3))/sqrt(beta(3));
d4=(z2-alpha(3))/sqrt(beta(3));

freq=2;
f=0;
for i=1:length(Af)
    d1=(sqrt(1+2*beta(3)*Ct3(i))*z1-(alpha(3)-theta(i)*beta(3)))/sqrt(beta
    (3));
    d2=(sqrt(1+2*beta(3)*Ct3(i))*z2-(alpha(3)-theta(i)*beta(3)))/sqrt(beta
    (3));

    f=f+(D(i)*gamma(i)*exp(-Af(i)-Bt(i)*(alpha(1)+sqrt(beta(1))*x) -...
        Ct2(i)*(alpha(2)+sqrt(beta(2))*y).^2-x.^2/2-y.^2/2).*(normcdf(d2)-
    normcdf(d1))...
        -exp(-Af(i)-Bf(i)*(alpha(1)+sqrt(beta(1))*x)-Cf(i)*(alpha(2)+sqrt(
    beta(2))*y).^2 -...
        x.^2/2-y.^2/2).*(normcdf(d4)-normcdf(d3)))/(2*pi);
    if mod(i,freq)==0
        j=i/freq;
        f=f-R*yF1y(j)*exp(-AF(j)-BF(j)*(alpha(1)+sqrt(beta(1))*x) -...
            CF(j)*(alpha(2)+sqrt(beta(2))*y).^2-x.^2/2-y.^2/2).*(normcdf(
    d4)-normcdf(d3))...
```

```
                /(2*pi);
        end
end

end
```

Appendices/fun3.m

B.4 EQM swaption pricer via Monte Carlo

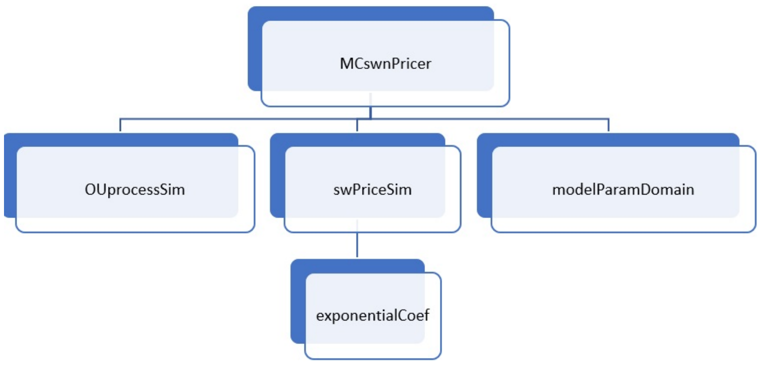

FIGURE B.5: EQM pricer flowchart

Here we price under the EQM model a swaption with Monte Carlo algorithm.

```
function [price,CI]=MCswnPricer(b1,b2,b3,sigma1,sigma2,sigma3,k,t,Tm,...
    Tf,TF,R,discCurve,IC,Nsim)

%% gaussian factors
if modelParamDomain(b1,b2,b3,sigma1,sigma2,sigma3,k,Tm,Tf)
    [psi1Tm,psi2Tm,psi3Tm]=OUprocessSim(IC,b1,b2,b3,sigma1,sigma2,sigma3,t
    ,Tm,Tm,Nsim);

    %% computation
    PD=ZRlinearInterp(t,discCurve.knots,discCurve.discounts,Tm);
    swPrice=swPriceSim(psi1Tm,psi2Tm,psi3Tm,b1,b2,b3,sigma1,sigma2,sigma3,
    k,Tm,Tf,TF,R);
    swnPayoff=(max(swPrice(:,1),0)+max(swPrice(:,2),0))/2;
    [price,~,CI]=normfit(PD*swnPayoff);
else
    price=-1;
end

end
```

Appendices/MCswnPricer.m

In the following function we simulate the threee Gaussian factors.

```
function [psi1,psi2,psi3]=OUprocessSim(IC,b1,b2,b3,sigma1,sigma2,sigma3,t,
    T,TM,Nsim)
% t=initial time, T=evaluation time, TM=forward measure time
psi1=zeros(Nsim,2);
psi2=psi1;
psi3=psi1;
dt=@(t,T)yearfrac(t,T,3);

```

```
%% psi1
Z=randn(Nsim,1);
psi1(:,1)=exp(-b1*dt(t,T))*(IC(1)-(sigma1/b1)^2*(exp(b1*dt(t,T))-1)+...
    (sigma1/b1)^2/2*(exp(b1*(dt(TM,T)+dt(t,T)))-exp(b1*dt(TM,t)))+...
        sigma1*sqrt((exp(2*b1*dt(t,T))-1)/(2*b1))*Z);
psi1(:,2)=exp(-b1*dt(t,T))*(IC(1)-(sigma1/b1)^2*(exp(b1*dt(t,T))-1)+...
    (sigma1/b1)^2/2*(exp(b1*(dt(TM,T)+dt(t,T)))-exp(b1*dt(TM,t)))+...
        sigma1*sqrt((exp(2*b1*dt(t,T))-1)/(2*b1))*(-Z));

%% psi2
h2=sqrt(4*b2^2+8*sigma2^2); % (3.5)
C2int=@(t,T,TM) 2*(2*log((2*b2*(exp(h2*dt(T,TM))-1)+h2*(exp(h2*dt(T,TM))+1)
    )./...
    (2*b2*(exp(h2*dt(t,TM))-1)+h2*(exp(h2*dt(t,TM))+1)))+(2*b2+h2)*dt(t,T)
    )/...
    ((2*b2+h2)*(2*b2-h2));
I=integral(@(u)exp(2*b2*dt(t,u)+4*sigma2^2*C2int(t,u,TM)),t,T)/365;

Z=randn(Nsim,1);
psi2(:,1)=exp(-b2*dt(t,T)-2*sigma2^2*C2int(t,T,TM))*(IC(2)+sigma2*sqrt(I)*
    Z);
psi2(:,2)=exp(-b2*dt(t,T)-2*sigma2^2*C2int(t,T,TM))*(IC(2)+sigma2*sqrt(I)
    *(-Z));

%% psi3
Z=randn(Nsim,1);
psi3(:,1)=exp(-b3*dt(t,T))*(IC(3)+sigma3*sqrt((exp(2*b3*dt(t,T))-1)/(2*b3)
    )*Z);
psi3(:,2)=exp(-b3*dt(t,T))*(IC(3)+sigma3*sqrt((exp(2*b3*dt(t,T))-1)/(2*b3)
    )*(-Z));

end
```

Appendices/OUprocessSim.m

Using *Proposition 2* and the already simulated processes we evaluate swap price in *swPriceSim* function

```
function price=swPriceSim(psi1,psi2,psi3,b1,b2,b3,sigma1,sigma2,sigma3,k,
    Tm,Tf,TF,R)

%% coefficients
[Af,AF,Bf,BF,Cf,CF,Bt,Ct2,Ct3,D]=exponentialCoef(b1,b2,b3,sigma1,sigma2
    ,...
    sigma3,k,Tm,Tf,TF);

%% formula
yF1y=yearfrac([Tm;TF(1:end-1)],TF,6);
freq=2;
price=zeros(size(psi1));
for i=1:length(Tf)
    price=price+...
        D(i)*exp(-Af(i)-Bt(i)*psi1-Ct2(i)*psi2.^2-Ct3(i)*psi3.^2) -...
        exp(-Af(i)-Bf(i)*psi1-Cf(i)*psi2.^2);
    if mod(i,freq)==0
        j=i/freq;
        price=price-R*yF1y(j)*exp(-AF(j)-BF(j)*psi1-CF(j)*psi2.^2);
    end
end

end
```

Appendices/swPriceSim.m

Bibliography

[1] Baviera, R., & Cassaro, A. (2015). A note on dual-curve construction: Mr. Crab's Bootstrap. Applied Mathematical Finance, 22(2), 105-132.

[2] Baviera, R. (2017). Back-of-the-envelope swaptions in a very parsimonious multicurve interest rate model. arXiv preprint arXiv:1712.06466.

[3] Bormetti, G., Brigo, D., Francischello, M., & Pallavicini, A. (2018). Impact of multiple curve dynamics in credit valuation adjustments under collateralization. Quantitative Finance, 18(1), 31-44.

[4] Brigo, D., & Mercurio, F. (2007). Interest rate models-theory and practice: with smile, inflation and credit. Springer Science & Business Media.

[5] Chen, L., Filipović, D., & Poor, H. V. (2004). Quadratic term structure models for risk-free and defaultable rates. Mathematical Finance: An International Journal of Mathematics, Statistics and Financial Economics, 14(4), 515-536.

[6] Filipović, D. (2002). Separable term structures and the maximal degree problem. Mathematical Finance, 12(4), 341-349.

[7] Glau, K., Grbac, Z., Scherer, M., & Zagst, R. (Eds.). (2016). Innovations in Derivatives Markets: Fixed Income Modeling, Valuation Adjustments, Risk Management, and Regulation (Vol. 165). Springer.

[8] Gombani, A., & Runggaldier, W. J. (2001). A filtering approach to pricing in multifactor term structure models. International Journal of Theoretical and Applied Finance, 4(02), 303-320.

[9] Grbac, Z., & Runggaldier, W. J. (2015). Interest rate modeling: post-crisis challenges and approaches. Cham-Heidelberg-New York: Springer.

[10] Henrard, M. (2010). The irony in derivatives discounting part II: The crisis. Wilmott Journal, 2(6), 301-316.

[11] Jamshidian, F. (1989). An exact bond option formula. The journal of Finance, 44(1), 205-209.

[12] Lamberton, D., & Lapeyre, B. (2011). Introduction to stochastic calculus applied to finance. Chapman and Hall/CRC, 149-169.

[13] Meneghello L. (2014). Interest Rate derivative pricing in multicurve factor models. Master Thesis. University of Padova.