A Comprehensive View of Markov-Functional Models and Their Application

Michael Lapere

2006

Supervised by Professor R. Becker

Presented to the University of Cape Town in the partial fulfilment of a Masters in Science - Mathematical Finance

Abstract

Markov-Functional models are a very powerful class of market models which calibrate and compute prices and Greeks quickly. This dissertation explains, in detail, how Markov-Functional models work as well as discussing all of the specific models developed in the literature. It contains the key points that can be found in the present literature. We explain, in detail, all of the concepts, from the theoretical framework down to the numerical implementation of the specific models. This involves explaining the framework for Markov-Functional models, describing specific models, obtaining a deeper understanding of how the model parameters affect the results, discussing the issues involved in the implementation, implementing various models and investigating the effect of numerical and market parameters on the outcome. Various concepts, not discussed in the present literature, such as considerations for selecting a discretization grid for the numerical implementation, are developed. The practical application of Markov-Functional models is considered as well as alternative fields, such as Actuarial science, where the model can be applied. In summary, this dissertation embodies a complete discussion of the current class of Markov-Functional models.

Acknowledgements

I am most grateful to my supervisor, Professor R. Becker, for suggesting this topic, his excellent guidance, inspiration and support during the course of this research. I also thank my family for their patience and support. Thanks are also due to James Jardine and David Acott for their many helpful suggestions and discussions. Finally, I thank my peers for their patience and support.

Contents

1 Introduction

Interest rate models form an important part of the field of quantitative finance. The simplest models are short rate models. These models, although easy to understand and quick to calibrate, suffer from the serious flaw that they do not model the yield curve accurately enough under all market conditions. Short rate models model the short rate, which is not a directly observable market rate. In order to obtain an arbitrage free model and the observable market rates requires some complex calculations.

The next innovation was the Heath-Jarrow-Morton (HJM) framework which ensured no arbitrage by setting restrictions on the dynamics of the of the interest rate process. This framework requires all the forward volatilities, which is a significant increase in the number of parameters from the short rate models. Short rate models fall under the HJM framework[1] although simplifying assumptions about the forward rate volatilities are required[2] e.g. the forward volatilities are constant. This still leaves the flaws that simple models do not provide an accurate enough model of the yield curve and that they do not model the market quoted rates directly.

Market models were introduced to model the yield curve more accurately and to model market quoted rates directly. They model market quoted rates and calibrate by ensuring that the model produces the market prices for a selection of derivatives based on these rates. For example, instead of modelling the underlying short rate and using that rate to model the Libor rate, Market models model the Libor rates directly. This improvement came at a price: Market models tend to be very complex, have many factors (sources of noise) and are difficult to calibrate. They can be computationally intensive resulting in the time taken to calculate the Greeks and price derivatives being significant. This can be problematic in practice. There has been much work in reducing the number of factors in Market models and applying restrictions to make these models driven by a low dimension Markov process[3]. The reason that we want the driving process to be a low dimensional Markov process is in order to reduce computational time and complexity.

Markov-Functional models ensure that the driving process is a low dimensional process Markovian process. Markov-Functional models are Market

[1] In fact, most interest rate models do.

[2] This then 'reduces' the number of actual parameters to the number of parameters in the short rate model.

[3] A comprehensive discussion of this can be found in Chiarella and Kwon [6].

models, in the sense that they model market quoted rates, but are much simpler in structure than traditional Market models. They start by using only a single Markovian factor and factors are added if necessary[4].

It turns out that single factor Markov-Functional models price derivatives and calculate Greeks with results that are comparable, for practical purposes, with those of more complex Market models. In addition, there are some tricks that can be used to calibrate to the correlation of co-terminal options, even though there is only a single factor. They calculate the prices and Greeks extremely quickly making them very suitable for practical purposes.

The absence of an extensive literature does pose the question of whether or not these models are used in practice. After questioning practitioners at the MIF conference[5] it emerged that the models are being used in practice[6].

During the course of this dissertation, it should become clear that Markov-Functional models are powerful models with a wide range of practical applications and provide a simple framework for solving complex problems. They calibrate and calculate their output much quicker than other Market models and their results are comparable. Their simple structure leads to an easy understanding of the model and thus allows one (e.g. trader) to obtain an intuitive understanding as to how the price was calculated. This could be used to adjust for market imperfections in a coherent way, making the model suitable for practical applications.

The format of the dissertation is as follows. In section 2 some preliminary theory is recalled. In sections 3 and 4 the framework for Markov-Functional models is introduced along with a heuristic explanation. In section 5 Libor Markov-Functional models are explained along with formulae to calculate the prices and Greeks of options. The swap Markov-Functional model is presented in section 6. Volatility smiles are considered in section 7. In section 8 the spot measure models are introduced. A multi-factor model is given in section 9, along with the relevant formulae.

The next part of the dissertation delves into the deeper meaning of the Markov process driving the model. The impact of the volatility of the Markov

[4]Single factor Markov-Functional models dominate the literature.

[5]Kruger Park 2005.

[6]Notably, Professor Lane Hughston mentioned that the models were being used in practice.

process on the model is examined in section 10. This leads to an approximation allowing one to calibrate to the terminal correlation of a Bermudan swaption. The relation between the models under different measures is examined.

We now move on to the uses of the model. In sections 12 and 13 we present pricing formulae along with some Actuarial applications. These present the framework for pricing any derivative in our economy.

The numerical implementation is not straightforward. In section 14 we explain, in detail, exactly how one goes about implementing the model. Issues such as numerical integration, calculating the cumulative Normal distribution, selecting the correct discretization points and the general procedure are considered. Many of these issues are not discussed in the present literature.

Sections 15 to 17 deal with the results. First we explain how to obtain relevant market data. We then have a literature review of the empirical results. In addition to presenting the standard results, we present results for different numerical and market parameters in order to determine how the Libor and swap model behave in different situations. We also present results for a Bermudan swaption priced using the Libor model in order to determine how consistent the numerical results of the swap and Libor Markov-Functional models are.

The C++ code forms a substantial part of this dissertation. There is a brief discussion of this code in section 18. The code can be found on the attached CD. It is written in a library type form in order to allow a user to quickly price any type of derivative. We thus explain how one goes about pricing a new derivative using the code. The header files which should explain some more of the technical details about the workings can be found in the appendix.

Finally, we conclude in section 19, commenting on possible further research that could be done.

2 Preliminary theory

This dissertation requires knowledge of finance and the mathematics of finance. Here we recall some important theory used in the derivation, interpretation and application of the Markov-Functional models.

2.1 Pure Discount Bonds (PDB's)

A PDB is a bond that pays out a single unit of currency at the maturity time. It is essentially a discount factor. For $t < s$, the value of a PDB at time t, which expires at time s is denoted by D_{ts} . D_{ts} is calculated by using

$$D_{ts} = \frac{1}{1+r} \tag{1}$$

where r is the effective rate of interest at time t for the interval $[t, s]$. In other words, r is the (simple annual) rate of interest at time t multiplied by $s - t$ (and time is measured in years).

2.2 Forward rates

A forward rate is a rate that is available now, but only applies for a period in the future. In other words, for $t < s < T$, the interest rate available at time t that applies over the interval $[s, T]$ is the forward rate at time t for time s that ends at time T . This rate is called the $s \times (T - s)$ forward rate at time t. Forward bonds emerge from forward rates as follows: we have $t < s < T$. Let i_{xy} denote the effective rate of interest at time t on the interval $[x, y]$ for any times $t < x < y$. The effective rate of interest on the interval $[t, T]$ at time t is

$$i_{tT} = (1 + i_{ts})(1 + i_{sT}) - 1 \tag{2}$$

Writing this in terms of bonds we have

$$\begin{aligned} D_{tT} &= \frac{1}{[(1 + i_{ts})(1 + i_{sT}) - 1 + 1]} \\ &= \frac{1}{(1 + i_{ts})} \frac{1}{(1 + i_{sT})} \\ &= D_{ts} D_{sT,t} \end{aligned} \tag{3}$$

where $D_{sT,t}$ denotes a forward PDB at time t for the period $[s, T]$. A forward PDB is a PDB that is available now but only discounts over a forward period. In other words a forward PDB is a PDB that we can agree now to purchase at a future time.

2.3 Caps

A cap is a derivative (contract), which pays out $\max(0, r_{T_i} - K) \times i$, at set time periods, $T_i, i = 0, 1, ..., n$, where K is a predetermined value, called the strike, r_{T_i} is the underlying rate of interest for the period $[T_i, T_{i+1}]$, and $i = (T_{i+1} - T_i)$. The set time periods, denoted T_i, $i = 0, 1, ..., n$ are called tenors. Generally, tenors are times that refer to some event in the market, such as the times that a cap pays out. A cap could pay out at the times T_i relating to the interest rates r_{T_i} or it could pay out in *arrears* which means that the payment $\max(0, r_{T_i} - K)$ is paid at time T_{i+1}.

A digital (or binary) cap pays out a single unit if $r_{T_i} > K$ or zero otherwise at time T_i (or T_{i+1} if it is in arrears).

2.4 Swaps

A swap is a contract to swap a particular (function[7] of a) variable market quoted rate for a fixed set rate, where the rate is charged on some agreed nominal amount. The nominal amount is never exchanged. The payments occur at specific tenors. An example would be to swap the Libor rate for a fixed rate, say 5%, on a nominal amount of 100. The person receiving the fixed rate is said to hold a receive fixed position. The payout at each tenor for the receive fixed position is $[K - r_{T_i}]\alpha_i$. The payout for a receive floating (pay fixed) position is the opposite of that, $[r_{T_i} - K]\alpha_i$. The fixed rate is set so that the value of the swap is zero to both parties, although in practice the fixed rate is usually set to benefit the bank providing the swap[8].

The fixed rate that sets the value of the swap to zero for both parties is called the *par swap rate*. The par swap rate can be calculated as follows: consider a swap at time T_0 with tenors T_i, $i = 0, 1, ..., n$. The PDB's are calculated using the underlying interest rate. The underlying interest rate is the rate on which the swap is based. The swap can be seen as a contract where the two parties lend each other a single unit and pay each other interest on the loan until time T_n when the capital is repaid. The loan is repaid at a fixed and a variable interest rate respectively. This is still a swap as the payments of the capital at the start and end of the loan cancel each other out. The fixed rate must be selected so that the value of these two loans is

[7]This is usually a simple function such as the rate plus some amount. This is done to take factors such as credit risk and profit margins into account.

[8]Seen differently, the floating rate is a function of some market quoted rate where the function is set to benefit the bank.

the same.

Consider the loan with the variable interest payments. This is a loan whose interest is repaid at the market rate of interest at regular intervals and the capital is repaid at the end. At outset, this loan has a value of zero as the rate of interest paid and the rate at which these payments are discounted are the same; they are the market rate of interest.

The value of the fixed interest loan with the fixed rate K, at time T_0 is the discounted sum of the interest payments and the capital at the end. The value is thus

$$1 - K\sum_{i=0}^{n-1} \alpha_i D_{T_0T_i} - D_{T_0T_n} \tag{4}$$

We require

$$0 = 1 - K\sum_{i=0}^{n-1} \alpha_i D_{T_0T_i} - D_{T_0T_n} \tag{5}$$

which can be solved for the fixed rate K

$$K = \frac{1 - D_{T_0T_n}}{\sum_{i=0}^{n-1} D_{T_0T_{i+1}}} \tag{6}$$

K, which is often denoted by y_t, is the par swap rate.

The *forward par swap rate* is the swap rate available now for a swap over some interval in the future. The forward par swap rate at time t, for a swap over the interval $[s,T]$, where $t \leq s \leq T$, is called the $t \times (T-s)$ forward par swap rate at time t. This is denoted by $y_t^{(s,T)}$.

2.5 Swaptions

A swaption is an option to enter a swap on predetermined terms. For example, there could be a swaption that allows one to enter a receive fixed position on a two year swap at a swap rate of 5% with quarterly tenors. The position can only be entered into on the expiry of the option, which is in one year.

A Bermudan swaption is a swaption that can be exercised at a fixed set of tenors. The holder can only exercise once. The swap which is entered into could depend on the date on which the swap is exercised. An example of

this is a *co-terminal* Bermudan swaption. In this swaption, the holder can exercise on a fixed set of tenors but the swap that is entered into will expire on a fixed date. Thus the later one exercises the option, the shorter the time span of the swap will be.

A typical use of a Bermudan swaption would be for a lender to hedge against the early repayment of a fixed interest loan that can be fully repaid by the borrower at a fixed set of times. This occurs in housing loans where the borrower has the option to pay back the full outstanding amount at any time. The lender giving out the loan would like to ensure that, regardless of when the borrower pays back the loan, the lender earns the fixed amount of interest for the entire duration of the loan. For example, if the loan is a 20 year loan on 100 at 5%, with the capital being repaid at the end and annual interest payments, the borrower may repay the 100 outstanding on the loan after 10 years. The lender would need to invest that 100 for the remaining 10 years at 5% interest in order to earn the interest that was expected. In 10 years time, if the market only offers 4% for the remaining 10 years, the lender will not earn his/her required 5% interest.

By entering a co-terminal Bermudan swaption, a lender could hedge itself. The Bermudan swaption would require the following terms

- The exercise dates of the Bermudan swaption would have to co-incide with the dates that the loan can be repaid.
- The swap that is entered into would be a receive fixed swap. The strike rate needs to be the same as the rate at which the money was lent (which is 5% in the above example).
- The floating rate (underlying) would need to be a market rate that can be obtained at any time in the future.
- The co-terminal expiry date of the swaps would need to co-incide with the date that the loan would be repaid if it was not paid back early.

The lender would hedge as follows: if the loan is paid back early the lender would invest the capital sum in the underlying market rate of interest, exercise the co-terminal Bermudan swaption and swap the floating rate of interest which is earned in the market for the fixed rate for the remaining duration of the swap.

The price paid for the co-terminal Bermudan swaption is not the fair theoretical price that the borrower should be charged for the option to pay the loan

back early. This is because if the loan is paid back early, the terms of the swap, should the option be exercised, may be less favourable than the market conditions. Thus the cost of the optionality of being allowed to exercise the swaption or not should not be charged to the borrower.

Note that the co-terminal Bermudan swaption essentially removes the downside reinvestment risk associated with obtaining early payments as well as adding optionality, so that one is not necessarily locked into entering a swap on unfavourable terms. In practice, things are not so simple and the capital is not all repaid at the end. Rather, the repayments include some of the capital, so the use of a co-terminal Bermudan swaption is not so simple.

A co-terminal Bermudan swaption leads to one considering forward rates for swaps that all expire on the same date, say T_n. The forward swap rate at time t for the period $[T_i, T_n]$ is denoted as $y_t^{(i)}$. We can obtain a set of co-terminal forward swap rates at each time t, namely $\{y_t^{(i)},\ i = 0, 1, ..., n\}$. The correlation between these forward co-terminal swap rates at time t is called the *terminal correlation* at time t.

A co-terminal Bermudan swaption is implemented and the results can be found in section 17.2.1.

2.6 Calculating Greeks

The Greeks are the sensitivities of a derivative to changes in the market variables that affect the price of the derivative. When we have a closed form solution for the price of a derivative we can calculate the Greeks by calculating the partial derivatives of the formula for the price of the derivative. In Markov-Functional models, we do not have a simple closed-form solution for the price of our derivatives. The Greeks are therefore calculated using the so called *bump-and-revalue* technique.

As the name suggests, the bump-and-revalue technique works by changing (bumping) a market variable and the revaluing the option. The sensitivity of the price to the market variable is calculated as

$$\text{Sensitivity} = \frac{\text{Price} - \text{Price after bumping}}{\text{Size of the bump}} \tag{7}$$

the size of the bump is the amount by which we changed the market parameter of interest. It is called the *perturbation* interval. Different size perturbation intervals usually lead to different sensitivities due to the second

order effect. We can calculate the second order effect in a similar fashion by comparing the sensitivities of two different perturbation intervals. In other words we calculate

$$\text{Second order effect} = \frac{\text{Sensitivity(1)-Sensitivity(2)}}{\text{Difference}} \tag{8}$$

where Sensitivity(1) refers to the sensitivity calculated using a large perturbation interval, Sensitivity(2) refers to the sensitivity calculated using a small perturbation interval and Difference refers to the difference between the large and the small perturbation interval. Note that we have two different perturbation intervals when calculating the second order effect: the perturbation intervals to calculate the sensitivity (using the small perturbation interval) and the difference between the two perturbation intervals. When we calculate the second order effect, the perturbation interval used to calculate the first order effects, Sensitivity(1) and Sensitivity (2), should be as small as possible as the second order effect should not be included in the calculation of the first order effect. Changing the size of the difference between the perturbation intervals used to calculate the second order effect will show a third order effect. Usually we are not interested in the third order effect, but this sensitivity can be calculated in a similar fashion.

2.7 The fundamental pricing theorem

The following theorem is derived by Harrison and Pliska [9] and taken from Bjork [5]. Consider a market model with asset price processes $S_0, ..., S_N$ on the time interval $[0, T]$. The numeraire process, S_0 is assumed to be strictly positive. The first fundamental theorem states that the market is arbitrage free iff there exists a martingale measure $\mathbb{Q} \sim \mathbb{P}$ such that the processes

$$\frac{S_i}{S_0}, \ i = 1, ..., N \tag{9}$$

are all (local) martingales under $\mathbb{Q}$.

The second fundamental theorem states that, assuming the absence of arbitrage, the market model is complete iff the martingale measure $\mathbb{Q}$ is unique.

These theorems imply that in a complete and arbitrage free market, with numeraire N and associated equivalent martingale measure $\mathbb{Q}$, the value of a traded asset at time $t < T$, V_t, is given by

$$V_t = N_t E^{\mathbb{Q}} \left[\frac{V_T}{N_T} | F_t \right] \tag{10}$$

as $\frac{V_t}{N_t}$ is a $\mathbb{Q}$-martingale.

2.8 Girsanov's theorem

The following theorem can be found in Bjork [5]. Let $W^{\mathbb{P}}$ be a d-dimensional standard $\mathbb{P}$-Wiener process on $(\Omega, \mathcal{F}, \mathbb{P}, \{F\}_t)$ and let φ be any d-dimensional adapted column process. Choose a fixed T and define the process L on $[0, T]$ by

$$\begin{aligned} dL_t &= \varphi_t' L_t dW_t^{\mathbb{P}} \\ L_0 &= 1 \end{aligned} \tag{11}$$

where the prime denotes the transpose. In other words, we have

$$L_t = \exp\left(\int_0^t \varphi_s' dW_s^{\mathbb{P}} - \frac{1}{2}\int_0^t ||\varphi_s||^2 ds\right) \tag{12}$$

where $||\cdot||$ denotes the Euclidean norm. Assume that

$$E^{\mathbb{P}}[L_T] = 1 \tag{13}$$

and define a new probability measure $\mathbb{Q}$ on $\mathcal{F}_t$ by

$$L_t = \frac{d\mathbb{Q}}{d\mathbb{P}}, \text{ on } \mathcal{F}_t \tag{14}$$

Then

$$dW_t^{\mathbb{P}} = \varphi_t dt + dW_t^{\mathbb{Q}} \tag{15}$$

where $W^{\mathbb{Q}}$ is a $\mathbb{Q}$-Wiener process. This essentially states how, in a Brownian world, the drift of a process, in this case process L_t (with zero drift under $\mathbb{P}$), changes as we changes our measure. Note that our volatility of the process remains unchanged. φ is known as the *Girsanov kernel.*

2.9 Martingale measure pricing

This theorem was derived by Geman, El Karoui and Rochet [8]. Suppose that we have a process A_t and that $\hat{A}_t$ is a process with the property that $\frac{\hat{A}_t}{A_t}$ is a strictly positive $\mathbb{Q}$-martingale. Define

$$\begin{aligned} L(t) &= \frac{\hat{A}_t/A_t}{\hat{A}_0/A_0} \\ L(T) &= \frac{d\hat{\mathbb{Q}}}{d\mathbb{Q}} \end{aligned} \tag{16}$$

If $\frac{M_t}{A_t}$ is a $\mathbb{Q}$-martingale, then $\frac{M_t}{\hat{A}_t}$ is a $\hat{\mathbb{Q}}$-martingale. In particular, if X is an attainable contingent claim, then

$$X_t = \hat{A}_t E^{\hat{\mathbb{Q}}} \left[\frac{X(T)}{\hat{A}(T)} | \mathcal{F}_t \right] \tag{17}$$

This theorem shows how we can change the measure and still have a martingale process by changing the numeraire. Here, the numeraire associated with $\mathbb{Q}$ is A_t and the numeraire associated with $\hat{\mathbb{Q}}$ is $\hat{A}_t$. A numeraire is a unit of measurement, like a currency. It is therefore a process and represents a tradeable asset. We divide our process of interest by the numeraire to obtain its value in terms of the numeraire. This theorem shows how measures and their associated numeraires are linked.

3 A heuristic explanation of Markov-Functional Models

A Markov process is one where the current position of the process summarizes the entire information available from the path of the process. In other words, the process increments are path-independent and its next position is dependent only on its current position. Markov-Functional models model the parameter of interest[9] using a function whose inputs are a Markov process e.g. if we are modelling the Libor rate, we model $L_t(x_t) \equiv$ Libor rate at time t, where x is a Markov process.

A simple example would be a Markov process defined by

$$dx_t = \mu dt + \sigma dW_t \tag{18}$$

where dW_t is Brownian motion under the risk-neutral measure, $\mathbb{Q}$, and the (identity) function mapping the Markov process to the short rate is

$$r_t(x_t) = x_t \tag{19}$$

where r_t is the short rate at time t. Here, the Markov process is the driving process as well as the short rate itself. This is the Merton short rate model. This example allows limited calibration to the market but demonstrates what *Markov-Functional* means.

[9] Usually a market quoted interest rate.

Markov-Functional models, as introduced by Hunt, Kennedy and Pelsser [10], take a slightly different and more technical approach. The following explanation leaves out many technical considerations which will be discussed later. We have

- An economy which consists only of pure discount bonds (PDB).
- The assumed existence of an equivalent martingale measure (EMM), $\mathbb{Q}$.
- A Markov process, x_t, with a known law under $\mathbb{Q}$ that is the only driving factor in our economy.
- A numeraire, $N_t(x_t)$, corresponding with the measure $\mathbb{Q}$. The numeraire is a function of only our Markov process.

We want to price derivatives using the fundamental pricing theorem

$$V_0 = N_0 E^{\mathbb{Q}} \left[\frac{V_t(x_t)}{N_t(x_t)} | \mathcal{F}_0 \right] \tag{20}$$

where V_t denotes the value of the derivative at time t. In order to do this we only require the functional form of $N_t(x_t)$ and $V_t(x_t)$. Usually, we calculate the functional form of the interest rates. We let $N_t(x_t)$ be a PDB which is a known function of the interest rates. The payoff of the derivative $V_t(x_t)$ is a known function of the interest rates. Thus by calculating the functional form of the interest rates we can obtain the functional forms of $N_t(x_t)$ and $V_t(x_t)$ which is what we require. Once we have these functional forms, we can calculate the expectation as we know the law of x_t under $\mathbb{Q}$. The functions are calculated (numerically) in such a way that the model is calibrated to the market i.e. we calibrate to the market by selecting the correct functional form. In higher dimensional cases calibration also includes calculating drift parameters. We usually model the market rates such as the Libor and the swap rate.

A Markov-Functional model does not necessarily imply that all the assets in the economy are Markovian. The numeraire of the spot measure model discussed in section 8 is not Markovian. We will only obtain a Markovian asset price process if we assume that an asset price process is not path dependent. The asset price process will then be Markovian in any EMM where the associated numeraire is of the form $N_t(x_t)$.

4 The Framework for Markov-Functional Models

In this section we explain and describe the Markov-Functional framework that was presented in Hunt et al. [10]. Figure 1 outlines the model. Formally, we are working in a single currency economy consisting only of pure discount bonds (PDB). Define D_{vt} as the value at time v of a PDB, with a nominal value of one, expiring at time t. We assume the existence of a unique EMM and hence conclude that the market is arbitrage free and complete. We work in a discrete time frame with times $T_0, ..., T_{n+1}$[10]. The filtration generated by this economy is defined as

$$\mathcal{F}_t = \sigma(D_{vt} : \ v \leq t, t \in \{T_0, ..., T_{n+1}\}) \tag{21}$$

where $\sigma(\cdot)$ denotes the generated sigma-algebra. Any type of trading is allowed but all strategies must be self financing. Let V_{T_m} denote the value of a derivative, expiring at T_{n+1}, at time $T_m \leq T_{n+1}$. This value can only be determined by the evolution of price processes until time T_m. This means that we only need to consider the evolution of prices until time T_m in order to price this derivative. Due to market completeness and the no arbitrage assumption we can replicate the derivative and have a numeraire N with associated EMM $\mathbb{Q}$, where equivalence is with respect to the 'real world' measure, $\mathbb{P}$. The probability space we use is $(\Omega, \mathcal{F}, \mathbb{Q})$. The value of a derivative at time $t < T_m$ is

$$V_t = N_t E^{\mathbb{Q}} \left(\frac{V_{T_m}}{N_{T_m}} | \mathcal{F}_t \right) \tag{22}$$

by the fundamental pricing theorem.

The assumptions are

1. x_t is a Markov process under the EMM $\mathbb{Q}$ associated with numeraire N_t.

2. The PDB prices are of the form: $D_{tS} = D_{tS}(x_t), \ \ 0 \leq t \leq \partial_B \leq S$ for some boundary curve $\partial_B : [0, S] \rightarrow [0, S]$

3. The numeraire N is of the form $N_t = N_t(x_t), \ \ 0 \leq t \leq S$

Note that the notation used in the definition above differs from that used in Hunt et al [10]. We refer to B as the *terminal time*. S corresponds to the

[10]The results can be extended to a continuum

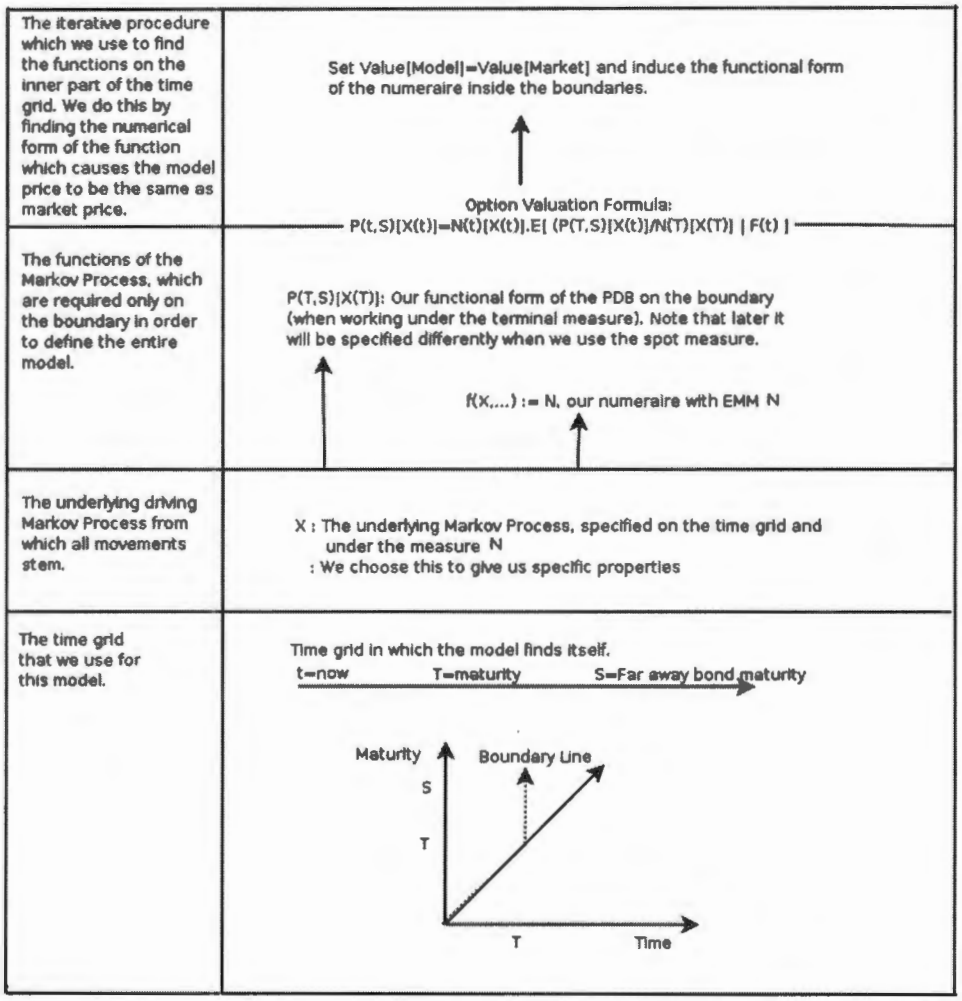

Figure 1: Outline of the Markov-Functional framework

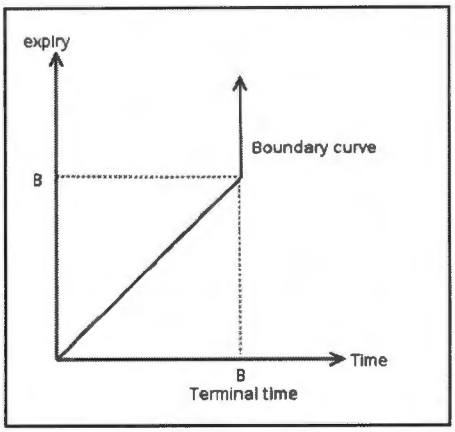

Figure 2: Boundary of the Libor Markov-Functional models with terminal time B.

time of the last payment of interest. The boundary curve ∂_B is

$$\partial_B = \begin{cases} t, & \text{if} \quad t \leq B \\ B, & \text{if} \quad t > B \end{cases}$$

where t refers to time. Figure 2 illustrates this boundary curve. In the implementation we let $T_{n+1} = S = B$ as we price products with the same terminal time, T_{n+1}.

Assumption (2) states that the value of the PDB, expiring on, and valued at all times from zero till the terminal time B, is a function of the Markov process, x_t. In addition, at any time before or equal to B the price of the PDB bond before, or on, time S is also a function of the Markov process. Assumption (1) and (3) ensure that we have a numeraire from time zero till time B which is a function of the Markov process. Thus if we specify D_{Bj}, $B \leq j \leq S$ we can obtain the functional form of the PDB inside the boundary curve by using the fundamental pricing theorem.

$$D_{tj} = N_t(x_t) E^{\mathbb{Q}} \left[\frac{D_{\partial_B j}(x_j)}{N_{\partial_B}(x_{\partial_B})} | \mathcal{F}_t \right], \quad t \leq B \leq j \leq S \tag{23}$$

The reason that we need to specify the functional form of the PDB outside the boundary is that we do not calibrate the model outside of this boundary. The numeraire used outside the boundary is the rolling unit numeraire and thus does not require a functional form. i.e. our model and its calibration

falls within the boundary. Outside the boundary we must specify all the calibrated functional forms ourselves.

To fully specify the model, we require

1. The law of x under the measure $\mathbb{Q}$.
2. The deterministic functional form of the numeraire $N_t(x_t) \; \forall \; 0 \leq t \leq B$.
3. The PDB prices as a function of x on the boundary curve, ∂_B. We only need the PDB prices on the boundary as we can calculate the prices inside the boundary by using equation 23 and $N_t(x_t)$[11].

This specifies the model as we now have a numeraire with a known law and the EMM. We can price any payoff function in this economy with this information by taking expectations.

The law of the Markov process is specified in the specific models. The PDB prices on the boundary are often taken to be the trivial $D_{t,t}(x_t) \equiv 1$ by selecting the appropriate boundary[12]. The functional form of the numeraire needs to be calculated numerically. This is done in the specific models.

An example of a model specification is

1. $dx = \sigma(t)dW_t$
2. $D_{T_nT_{n+1}} = \dfrac{1}{1 + \alpha_n L_0^{(n)} \exp(-\frac{1}{2}\int_0^{T_n} (\sigma_u^{(n)})^2 du + x_{T_n})}$
3. $N_t(x_t) = D_t(x_t)$

where $S = T_{n+1}$ and $B = T_n$. This example is the Libor Markov-Functional model under the terminal measure and is discussed in section 5 where all the terms are defined.

We have been working in a discrete time framework. We would like to ensure that our model is arbitrage free at all the times that fall between our discrete times i.e. we want to ensure that there is no arbitrage for times t, $T_i \leq t \leq T_{i+1}$, where $i = 0, 1, ..., n$ and $T_{n+1} \leq t$. The next section shows that we have no arbitrage at the intermediate times by developing an arbitrage free model for all the intermediate times that is consistent with the Markov-Functional framework.

[11] Which we have from the second requirement

[12] Let the last payment of interest coincide with the terminal time.

4.1 A consistent arbitrage-free term structure model for Markov-Functional models

This section draws on Hunt and Kennedy [11][13]. This section extends the functional form of the numeraire to all $t \leq B$. Due to the presence of an EMM and associated numeraire every discount bond divided by the numeraire is a martingale. Thus

$$\frac{D_{tT_i}}{N_t} \equiv E^{\mathbb{Q}}\left[\frac{D_{T_iT_i}}{N_{T_i}}|\mathcal{F}_t\right] = E^{\mathbb{Q}}\left[\frac{1}{N_{T_i}}|\mathcal{F}_t\right] \tag{24}$$

To show that this holds for points outside our discretization points define

$$N_t(x_t) = \frac{T_{i+1}-t}{T_{i+1}-T_i}N_{T_i}(x_{T_i}) + \frac{t-T_i}{T_{i+1}-T_i}N_{T_{i+1}}(x_{T_{i+1}}) \tag{25}$$

for $T_i \leq t \leq T_{i+1}$ and $T_0 \equiv 0$. This is a linear interpolation between the numeraire values. We could also use the interpolation

$$\frac{1}{N_t(x_t)} = \frac{D_{0t}-D_{0T_{i+1}}}{D_{0T_i}-D_{0T_{i+1}}}\cdot\frac{1}{N_{T_i}(x_{T_i})} + \frac{D_{0T_i}-D_{0t}}{D_{0T_i}-D_{0T_{i+1}}}\cdot\frac{1}{N_{T_{i+1}}(x_{T_{i+1}})} \tag{26}$$

This provides us with a continuous semi-martingale with a known law. The law is known as we have the functional form of $N_{T_i}(x_{T_i})$ for all i under the measure $\mathbb{Q}$. We now define a numeraire model. Let S denote the time of the last relevant payment and T_{n+1} denote the terminal time in our model. Define

$$D_{tS}(x_t) \equiv N_t(x_t)E^{\mathbb{Q}}\left(\frac{1}{N_S(x_S)}|\mathcal{F}_t\right) \tag{27}$$

for $0 \leq t \leq S \leq T_{n+1}$ and

$$D_{tS}(x_t) \equiv N_t(x_t)E^{\mathbb{Q}}\left(\frac{D_{T_nS}(x_{T_n})}{N_{T_n}(x_{T_n})}|\mathcal{F}_t\right) \tag{28}$$

for $0 \leq t \leq T_{n+1} \leq S$. Extending the model deterministically for $T_{n+1} \leq t \leq S$ using

$$D_{tS}(x_t) = \frac{D_{T_nt}(x_{T_n})}{D_{T_nS}(x_{T_n})} \tag{29}$$

completes the model. Note that we use the numeraire, $N_t(x_t)$ as defined until the terminal time and then use the unit numeraire thereafter.

[13]The equations are exactly as in the reference except for a few typographical errors that were corrected.

$N_t(x_t)$ is not adapted to the filtration generated by the process x, and is therefore never Markov. This is not a problem as this section describes and arbitrage-free term structure that is consistent with the Markov-Functional model, and not a Markov-Functional arbitrage-free term structure model.

5 The Libor Markov-Functional model under the terminal measure

5.1 A formal definition of the model

This section draws from Hunt et al [10] and Fries and Rott [7]. For notational convenience let ξ be a numerical parameter denoting a state of the Markov process i.e. $\xi \equiv x_{T_i}(\omega)$ at time T_i in state ω.

The model framework does not specify what the functional forms of the numeraire are in the interior of the boundary curve. These functional forms must be obtained by calibrating to the market. Here we calibrate to the Libor market.

The i^{th} forward Libor rate, where i is the time index, counting from time T_0, available at time T_i, in state ξ is defined as

$$1 + L_{T_i}^{(i)}(\xi)(T_{i+1} - T_i) \equiv \frac{D_{T_iT_i}(\xi)}{D_{T_iT_{i+1}}(\xi)} = \frac{1}{D_{T_iT_{i+1}}(\xi)} \tag{30}$$

Our numeraire at time T_i, corresponding to the terminal measure $\mathbb{Q}$, and state ξ is defined as

$$N_{T_i}(\xi) \equiv D_{T_iT_{n+1}}(\xi) \tag{31}$$

In the model framework we specified that the numeraire is a deterministic function of the Markov process, x_{T_i}. The Libor rate is thus also a deterministic function of the Markov process due to the above equation. The Libor rate is related to the numeraire by

$$\begin{aligned} 1 + L_{T_i}^{(i)}(\xi)(T_{i+1} - T_i) &= \frac{1}{D_{T_iT_{i+1}}(\xi)} \\ &= \frac{1}{N_{T_i}(\xi) E^{\mathbb{Q}}\left(\frac{1}{N_{T_{i+1}}(x_{T_{i+1}})} | (T_i, \xi)\right)} \end{aligned} \tag{32}$$

Note the expectation contains a numeraire as a function of $x_{T_{i+1}}$ instead of ξ. This is because ξ is the known state that the process is currently in. In the expectation, $x_{T_{i+1}}$ denotes the Markov process which is the relevant random variable. Also note that our conditioning is on (T_i, ξ), which is a state of the Markov process as defined at the beginning of this section.

Solving for the numeraire we obtain

$$N(T_i,\xi) = \frac{1}{E^{\mathbf{Q}}\left(\frac{1}{N_{T_{i+1}}(x_{T_{i+1}})}|(T_i,\xi)\right)\cdot(1+L^{(i)}_{T_i}(\xi)(T_{i+1}-T_i))} \tag{33}$$

If we had the function $\xi \mapsto L^{(i)}_{T_i}(\xi)$ we would be able to calculate the functional form of $N_{T_i}(\xi)$ from $N_{T_{i+1}}(\xi)$. This is because we can calculate the expectation in the denominator as we know the law of the Markov process. We start the iterative procedure with $N_{T_{n+1}}(x_{T_{n+1}}) \equiv 1$.

Thus all that remains is to specify the law of the Markov process and the function $\xi \mapsto L^{(i)}_{T_i}(\xi)$. The law of the Markov process is specified using volatility considerations discussed in section 10. The function is calculated numerically.

5.2 The theory behind the numerical calculation of the functional form of the numeraire

This section draws from Hunt et al [10], Fries and Rott [7] and Pelsser [13]. The single payoff, and hence the market price at the time that the payoff occurs, of a digital caplet in arrears on $L_{T_{i-1}}$, with strike K and fixing date T_i, is given by

$$V^{\text{market}}_{K,T_{i-1}}(T_i) = 1_{L^{(i-1)}_{T_{i-1}} \geq K} \text{ paid at time } T_i \tag{34}$$

where $1_{x \geq y} = 1$ if $x \geq y$ and zero otherwise. We now make the *assumption* that the function $\xi \mapsto L(T_{i-1},\xi)$ is monotone and increasing in ξ. This ensures that

$$\{x_{T_i} > x^*\} = \{L^{(i)}_{T_i} > K^{(i)}(x^*)\} \tag{35}$$

for a unique $K^{(i)}(x^*)$. For a fixed $x^* \in \mathbb{R}$, the model payoff of a digital caplet in arrears on $L^{(i-1)}_{T_{i-1}}$, settled at T_i with strike $L^{(i-1)}_{T_{i-1}}(x^*)$ is given by

$$\xi \mapsto V^{\text{model}}_{L^{(i-1)}_{T_{i-1}}(x^*)}(T_i,\xi) = 1_{\xi \geq x^*} \tag{36}$$

Note that the model payoff is a function of x^* only and not $L_{T_{i-1}}^{(i-1)}(x^*)$. Using the fundamental pricing theorem we obtain

$$V_{L_{T_{i-1}}^{(i-1)}(x^*)}^{\text{model}}(T_0) = N_{T_0} E^{\mathbb{Q}}\left(\frac{1_{x \geq x^*}}{N_{T_i}(x)} | \mathcal{F}_0\right) \tag{37}$$

$$= N_{T_0} E^{\mathbb{Q}}\left(\frac{1_{x \geq x^*}}{N_{T_i}(x)} | (T_0, x_0)\right) \tag{38}$$

We do not need $L_{T_{i-1}}^{(i-1)}(x^*)$ to calculate the right hand side of equation (38) as our numeraire is a function of only $L_{T_i}^{(i)}(\xi)$. To proceed iteratively backwards through time we will require the values of $L_{T_{i-1}}^{(i-1)}(x)$ for all x in the next iteration i.e. we require the functional form of $L_{T_{i-1}}^{(i-1)}(\xi)$. The left hand side of equation (38) depends on $L_{T_{i-1}}^{(i-1)}(x^*)$. Thus we can calculate the value of $L_{T_{i-1}}^{(i-1)}(x^*)$ in this step by equating the model price, which is the right hand side of the equation, with the market price and solve the left hand side of the equation for $L_{T_{i-1}}^{(i-1)}(x^*)$. Doing this for all $x^* \in \mathbb{R}$ will provide us with the required function.

We set

$$V_{x^*,T_{i-1}}^{\text{model}}(T_0) = V_{K,T_{i-1}}^{\text{market}}(T_0) \tag{39}$$

in order to ensure that the model is calibrated to the market. We calculate the functional form of the Libor rate by solving for the market strike K in this equation. Solving gives us $K = L_{T_{i-1}}(x^*)$ which provides us with

$$\xi \mapsto L_{T_{i-1}}(\xi) \tag{40}$$

due to equation (35). In summary the mapping procedure, depicted in Figure 3, is

- We calculate the model price corresponding to x^* by integrating over the relevant distribution (which will be defined to be the Normal distribution) from x^* to infinity.
- We find the market price that is equal to the model price.
- We find the strike that corresponds to this market price.
- The value of this strike is the value of the Libor rate corresponding to x^*.

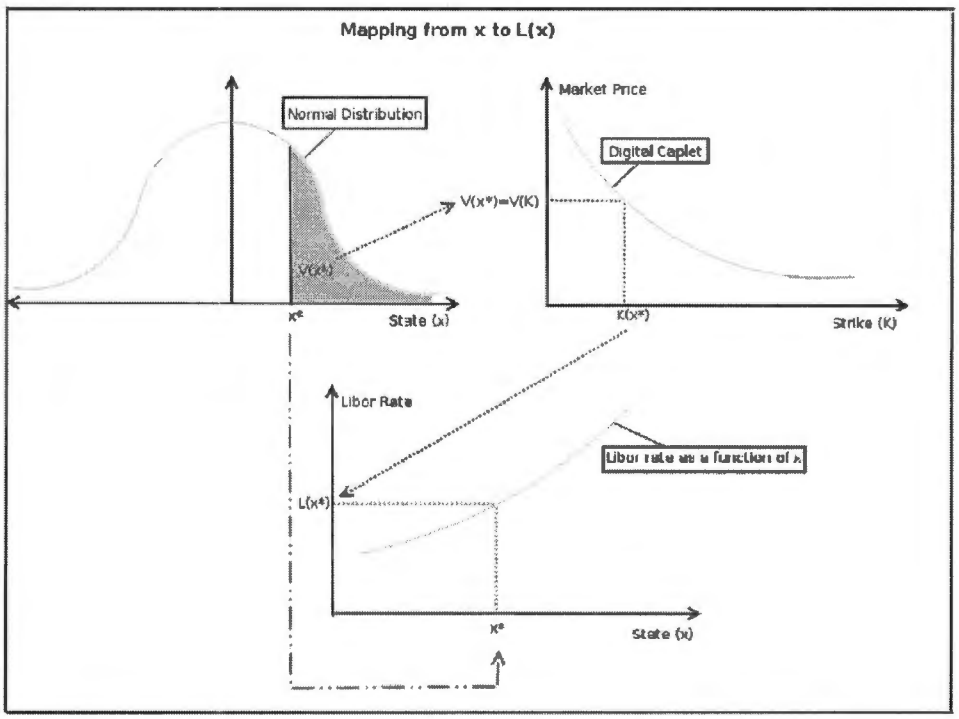

Figure 3: The mapping from x^* to $L(x^*)$. We start on the top left with by calculating the model price as the Gaussian integral above x^*. The mapping procedure is depicted by the dotted blue line. We follow this line to find the corresponding market price of a digital caplet. This provides us with a strike which is the Libor rate that we seek corresponding to x^*. Doing this for all x^* provides us with the functional form. Figure adapted from Fries and Rott [7].

5.3 Formulae for numerically calculating the functional form of the numeraire

The above gave a theoretical description of how one use backward induction to obtain the functional form of the numeraire inside the grid. Here we draw on Hunt et al. [10] and explain, including all the steps and formulae, exactly how one would go about finding the functional form. So far we have

1. A set of forward Libor rates at time T_0, $L^{(i)}_{T_0}$, $i = 1, 2, ..., n$
2. A numeraire, $D_{tT_{n+1}}$, with the corresponding terminal measure, $\mathbb{Q}$.

For consistency with Black's Formula for caplets, we assume that the forward LIBOR rate, $L^{(n)}_t$, defined in equation (30), is a log-normal martingale under the terminal measure $\mathbb{Q}$. So

$$dL^{(n)}_t = \sigma^{(n)}_t L^{(n)}_t dW_t \tag{41}$$

where W is standard Brownian motion under the terminal measure and $\sigma^{(n)}_t$ is the deterministic volatility. Solving gives us

$$\begin{aligned} L^{(n)}_t &= L^{(n)}_0 \exp(-\frac{1}{2}\int_0^t (\sigma^{(n)}_u)^2 du + x_t) \\ &\text{where} \\ dx_t &= \sigma^{(n)}_t dW_t \\ x_0 &= 0 \end{aligned} \tag{42}$$

The boundary curve is trivial and has the terminal time T_{n+1} and the boundary time T_n, which correspond to S and B in the framework respectively. We now define the functional forms of our PDB prices on the boundary. We have that $D_{T_nT_n}(x_{T_n}) = 1$ and

$$\begin{aligned} D_{T_nT_{n+1}} &= \frac{1}{1 + \alpha_n L^{(n)}_{T_n}} \\ &= \frac{1}{1 + \alpha_n L^{(n)}_0 \exp(-\frac{1}{2}\int_0^{T_n} (\sigma^{(n)}_u)^2 du + x_{T_n})} \end{aligned} \tag{43}$$

The numeraire is defined as the PDB price $N_{T_i}(x_{T_i}) = D_{T_iT_{n+1}}(x_{T_i})$ and is thus already specified on the boundary. Next we need to obtain the functional form of the numeraire inside the boundary curve. We obtain this by calibrating to the market prices of caplets which is the same as calibrating to

the market price of digital caplets. Calibrating to digital caplets and caplets is equivalent as we are using the Black formula to obtain the implied volatility. The Black implied volatility for a digital caplet and a caplet is the same, else arbitrage would exist. The model value at time zero of the i^{th} digital caplet is, from equation (38)

$$V_0^{(i)\text{ model}}(K) = D_{T_0T_{n+1}}(x_0)E^{\mathbb{Q}}\left[\frac{D_{T_iT_{i+1}}(x_{T_i})}{D_{T_iT_{n+1}}(x_{T_i})}1_{(L_{T_i}^{(i)}(x_{T_i})>K)}\right] \tag{44}$$

Assume now that the market value is given by Black's formula[14]. The price of a digital caplet in arrears on $L_{T_i}^{(i)}$, settled at T_{i+1} with strike K is

$$V_0^{(i)}(K) = D_{T_0T_{i+1}}(x_0)\Phi(d_2^{(i)}) \tag{45}$$

$$\text{where } d_2^{(i)} = \frac{\log(L_0^{(i)}/K)}{\sigma^{(i)}\sqrt{T_i}} - \frac{1}{2}\sigma^{(i)}\sqrt{T_i} \tag{46}$$

and Φ is the standard Normal cumulative distribution and $\sigma^{(i)}$ is the market quoted volatility of the Libor rate for the i^{th} time period. Now define

$$\begin{aligned} J_0^{(i)}(x^*) &= N_{T_0}(x_0)E^{\mathbb{Q}}\left[\frac{D_{T_iT_{i+1}}(x_{T_i})}{D_{T_iT_{n+1}}(x_{T_i})}1_{x_{T_i}>x^*}\right] \\ &= N_{T_0}(x_0)E^{\mathbb{Q}}\left[E^{\mathbb{Q}}\left[\frac{D_{T_{i+1}T_{i+1}}(x_{T_{i+1}})}{D_{T_{i+1}T_{n+1}}(x_{T_{i+1}})}|\mathcal{F}_{T_i}\right]1_{x_{T_i}>x^*}\right] \\ &= N_{T_0}(x_0)\int_{x^*}^{\infty}\left[\int_{-\infty}^{\infty}\frac{1}{D_{T_{i+1}T_{n+1}}(u)}\phi_{x_{T_{i+1}}|x_{T_i}=v}(u)du\right]\phi_{x_{T_i}}(v)dv \end{aligned} \tag{47}$$

where $\phi_{x_{T_i}}$ is the transition density function of x_{T_i}, which in this case is Gaussian with zero mean and the variance as defined by the equation governing the Markov Process. $\phi_{x_{T_{i+1}}|x_{T_i}=v}$ is the transition density of $x_{T_{i+1}}$ given $x_{T_i} = v$. In this case, $\phi_{x_{T_{i+1}}|x_{T_i}}$ is the Normal density with mean v and variance $\int_{T_i}^{T_{i+1}}(\sigma_s^{(n)})^2ds$.

$J_0^{(i)}(x^*)$ can be interpreted as the model price of a digital caplet settled at T_{i+1} with strike $L_{T_i}^{(i)}(x^*)$. Recall equation (35) which links the Markov process with the Libor rates and strikes. We need to find the strike K so that the values of our model and the market match i.e we must find

$$L_{T_i}^{(i)}(x^*) = K^{\text{market}} = K^{(i)}(x^*) \tag{48}$$

[14]Note that we could have used any monotone continuous price vs strike and volatility curve. The Black formula is a natural choice. Changing this choice will result in different formulae to numerically calculate the functional form of the numeraire.

where the market strike, $K^{(i)}(x^*)$ is found from

$$J_0^{(i)}(x^*) = V_0^{(i)\text{ market}}(K^{(i)}(x^*)) \tag{49}$$

Thus we invert the market price given by the Black formula, equation (45), to solve for the strike, K.

$$\begin{aligned}
& V_0^{(i)}(K) = D_{T_0T_{i+1}}(x_0)\Phi(d_2^{(i)}) \\
\Rightarrow\ & d_2^{(i)} = \Phi^{-1}\left(\frac{V_0^{(i)\text{market}}}{D_{T_0T_{n+1}}}\right) \\
\Rightarrow\ & \frac{\log(L_0^{(i)}/K)}{\sigma^{(i)}\sqrt{T_i}} - \frac{1}{2}\sigma^{(i)}\sqrt{T_i} = \Phi^{-1}\left(\frac{J_0^{(i)}(x^*)}{D_{T_0T_{n+1}}(x_0)}\right) \\
\Rightarrow\ & K = L_0^{(i)}\exp\left[-\frac{1}{2}(\sigma^{(i)})^2T_i - \sigma^{(i)}\sqrt{T_i}\Phi^{-1}\left(\frac{J_0^{(i)}(x^*)}{D_{T_0T_{i+1}}(x_0)}\right)\right] \\
\Rightarrow\ & L_{T_i}^{(i)}(x^*) = L_0^{(i)}\exp\left[-\frac{1}{2}(\sigma^{(i)})^2T_i - \sigma^{(i)}\sqrt{T_i}\Phi^{-1}\left(\frac{J_0^{(i)}(x^*)}{D_{T_0T_{i+1}}(x_0)}\right)\right]
\end{aligned} \tag{50}$$

From this rate we calculate the functional form of $\xi \mapsto D_{T_iT_{n+1}}(\xi)$ using

$$\begin{aligned}
D_{T_iT_{n+1}}(x_{T_i}) &= \left((1+\alpha_i L_{T_i}^{(i)}(x_{T_i}))D_{T_{i+1}T_{n+1}}(x_{T_i})\right)^{-1} \\
&= \left((1+\alpha_i L_{T_i}^{(i)}(x_{T_i}))\frac{D_{T_iT_{i+1}}(x_{T_i})}{D_{T_iT_{n+1}}(x_{T_i})}\right)^{-1}
\end{aligned} \tag{51}$$

which allows us to calculate $J_0^{(j)}(x^*)$ for our next iteration (where $j = i-1$). Note that

$$\frac{D_{T_iT_{i+1}}(x_{T_i})}{D_{T_iT_{n+1}}(x_{T_i})} \tag{52}$$

is the inner integral in $J(x^*)$ which we have calculated already.

5.4 Market completeness and pricing standards

By using this method, we now only need the market volatilities for every strike. These need to be obtained from the market at discrete intervals, but these discretization points are unknown as we are solving for the strike on which the volatility depends in the presence of a skew. We will only obtain volatilities for a specific set of times and strikes and thus we need to fill in

the missing values. Interpolation between specific times can be done using a volatility model or making an assumption that the volatility curve follows some smooth function. Linear interpolation seems to be the most used method in the literature. Using volatility models allows us to have robust theory supporting our non-existent market prices, rather than interpolating amongst *prices* and hoping for the best. The models considered so far assume that we have a constant volatility at each time. We need to select which volatility to use e.g. at-the-money. Calibration to a smile is considered in section 7.

This method also has the advantage that it allows us to incorporate our views into the model. If we believe that the market has incorrectly priced the volatility then we can adjust for our beliefs. This is an important part of the practical application of the model as experience and different views need to be incorporated.

This approach assumes that the Black '76 formula is the market standard for pricing interest rate derivatives which are quoted as volatilities. If this is not the case then we cannot invert our strike using equation (50) directly. We have two possible options. The first is to adjust the model for the market standard which may not be easy to invert. However, this may not provide us with an internally consistent model as we have assumed a log-normal Libor rate at our terminal time. The second is to calculate the market *price*, solve for the implied Black volatility and use that. This may now be 'externally' inconsistent as our numeraire's assumptions outside the boundary are incorrect. These considerations are discussed in detail in section 7.

5.5 Scope of use of the model

Note that this model is now calibrated to caplets. We should only price caplet based products[15] on this. If we wanted to price swaptions, we need to calibrate this model to swaps. There does not seem to be any discussion in the literature of what happens when one calibrates to caplets and prices swaps. We have therefore calibrated to caplets and priced a Bermudan swaption in section 17.1.3 and analyzed the results.

[15]Products that can be expressed as a function of caplets e.g. caps

6 The Markov-Functional swap model under the terminal measure

This model is very similar to the Libor model although we work with swap rates rather than with Libor rates. The formal definition and theory behind numerically calculating the functional form of the numeraire are only minor deviations from the Libor model. For the remainder of the section we provide the formulae and explanations thereof required to numerically calculate the functional form.

This section draws from Hunt et al [10] and Pelsser [13]. Again, we use the pure discount bond, $D_{T_i T_{n+1}}$, as our numeraire. The driving Markov process, the boundary and the functional form of the numeraire on the boundary curve is defined as in the Libor Markov-Functional model. The definition of the numeraire on the boundary makes sense as our last forward par swap rate, for the period $[T_n, T_{n+1}]$, is the forward Libor rate for that period. The functional form of the numeraire is not the same as in the Libor model and we will need to find it. All the other assumptions and properties of the Markov-Functional framework remain unchanged.

Let $y^{(i)}_{T_j}$ denote the i^{th} forward par swap rate, at time $T_j \leq T_i$, which sets on date T_i and has coupons payable on dates $T_{i+1}, ..., T_{n+1}$. Define the present value per basis point (PVBP) at time t, in state x_t, for the i^{th} forward period as

$$P^i_t(x_t) = \sum_{j=i}^{n} \alpha_j D_{t,T_{j+1}}(x_t) \tag{53}$$

where $\alpha_j = T_{j+1} - T_j$. The model value at time T_0 for a digital swaption with strike K and par swap rate $y^{(i)}_{T_0}$ is, by the fundamental pricing theorem

$$V^{(i)\text{ model}}_0(K) = D_{0T_{n+1}}(x_0) E^{\mathbb{Q}}\left[\frac{P^{(i)}_{T_i}(x_{T_i})}{D_{T_i T_{n+1}}(x_{T_i})} 1_{y^{(i)}_{T_i}(x_{T_i}) > K} \right] \tag{54}$$

Assume that volatility quotes are converted to price quotes by using the Black formula[16]. The market price at T_0 of a digital swaption on the i^{th} par swap rate, settled at T_i with strike K is thus

$$V^{(i)\text{ market}}_0(K) = P^{(i)}_0(x_0)\Phi(d^{(i)}_2) \tag{55}$$

[16]In other words, we assume that the Black formula is the market standard metric to convert volatility quotes into settled prices and vice versa.

where

$$d_2^{(i)} = \frac{\log(y_0^{(i)}/k)}{\bar{\sigma}^{(i)}\sqrt{T_i}} - \frac{1}{2}\bar{\sigma}^{(i)}\sqrt{T_i} \tag{56}$$

and where $\bar{\sigma}^{(i)}$ is the respective market quoted volatility and Φ is the standard Normal cumulative distribution function. Now define

$$\begin{aligned} J_0^{(i)}(x^*) &= D_{T_0T_{n+1}} E^{\mathbb{Q}}\left[\frac{P_{T_i}^{(i)}(x_{T_i})}{D_{T_iT_{n+1}}(x_{T_i})} 1_{x_{T_i}>x^*}\right] \\ &= D_{T_0T_{n+1}} E^{\mathbb{Q}}\left[E^{\mathbb{Q}}\left(\frac{P_{T_{i+1}}^{(i)}(x_{T_{i+1}})}{D_{T_{i+1},T_{n+1}}(x_{T_{i+1}})}|\mathcal{F}_{T_i}\right) 1_{x_{T_i}>x^*}\right] \\ &= D_{T_0T_{n+1}}(x_0)\int_{x^*}^{\infty}\left[\int_{-\infty}^{\infty}\frac{P_{T_{i+1}}^{(i)}(u)}{D_{T_{i+1}T_{n+1}}(u)}\phi_{x_{T_{i+1}}|x_{T_i}=v}(u)du\right]\phi_{x_{T_i}}(v)dv \end{aligned} \tag{57}$$

$D_{T_{i+1}T_j}(x_{T_{i+1}})$, $j > i$ is known from the previous steps and we start with $D_{T_{n+1}T_{n+1}} = 1$.

Similarly to the Libor model we assume that $\xi \mapsto y_{T_i}^{(i)}(\xi)$ is monotonic and increasing in ξ. This allows us to set the identity

$$\{x_{T_i} > x^*\} = \{y_{T_i}^{(i)} > K^{(i)}(x^*)\} \tag{58}$$

for a unique $K^{(i)}(x^*)$. $J_0^{(i)}(x^*)$ can be interpreted as the model price of a digital swaption on the i^{th} forward par swap rate, settled at time T_i with strike $y_{T_i}^{(i)}(x^*)$. By finding the market strike that equates our market price and model price struck at $y_{T_i}^{(i)}(x^*)$ we can obtain the functional form of $\xi \mapsto y_{T_i}^{(i)}(\xi)$. Thus we find $K^{(i)}(x^*)$ so that

$$J_0^{(i)}(x^*) = V_0^{(i)\text{ market}}(K^{(i)}(x^*)) \tag{59}$$

and use

$$y_{T_i}^{(i)}(x^*) = K^{(i)}(x^*) \tag{60}$$

in order to calibrate to the market. This provides us with the functional form by solving for K in the Black formula, equation (55). Similarly to the Libor model, inverting Black's formula for $K = y_{T_i}^{(i)}$ results in

$$y_{T_i}^{(i)} = y_0^{(i)}\exp\left[-\frac{1}{2}(\bar{\sigma}^{(i)})^2 T_i - \bar{\sigma}^{(i)}\sqrt{T_i}\Phi^{-1}\left(\frac{J_0^{(i)}(x^*)}{P_0^{(i)}(x_0)}\right)\right] \tag{61}$$

We then calculate the discount factor using

$$D_{T_i,T_{n+1}}(x_{T_i}) = \left(1 + y_{T_i}^{(i)}(x_{T_i})\frac{P_{T_i}^{(i)}(x_{T_i})}{D_{T_iT_{n+1}}(x_{T_i})}\right)^{-1} \tag{62}$$

and then proceed iteratively backwards through time. Note that

$$\frac{P_{T_i}^{(i)}(x_{T_i})}{D_{T_iT_{n+1}}(x_{T_i})} \tag{63}$$

is the inner integral in $J(x^*)$ which we have already calculated.

When coding this one should take care with the PVBP's. In equation (57) we use $P_{T_{i+1}}^{(i)}$. Note that the index of the subscript and superscript no longer match. This is the PVBP at time T_{i+1} for time T_i. This therefore includes the current payment at time T_{i+1}. Thus we have

$$P_{T_{i+1}}^{(i)} = P_{T_{i+1}}^{(i+1)} + 1 \cdot \alpha_i = \text{PVBP}(T_{i+1}) + \alpha_i \tag{64}$$

6.1 The displaced diffusion swap Markov-Functional model

Displaced diffusion dynamics are designed to be able to fit a volatility skew. Pietersz and Pelsser [14] provide a swap Markov-Functional model in a displaced diffusion model setting which we will present here. The notation is defined in the same way as the swap model. Note that σ_i denotes the volatility of the swap rate, $y_t^{(i)}$. The displaced diffusion dynamics are those of Rubinstein [15] and are

$$y_t^{(i)} = \hat{y}_t^{(i)} - r_i, \quad \frac{d\hat{y}_t^{(i)}}{\hat{y}_t^{(i)}} = \sigma_i dW_t \tag{65}$$

where dW_t denotes Brownian motion under the terminal measure and r_i is the displacement parameter which is calculated from the market prices. Fitting to the skew will be done by fitting the displaced diffusion model formula for an option price to the observed market prices. A least squares fit can then be obtained. This has the solution

$$y_t^{(i)} = -r_i + (y_0^{(i)} + r_i)\exp\left[\sigma_i W_t - \frac{1}{2}\sigma_i^2 t\right] \tag{66}$$

The value of a digital swaption on rate $y_{T_i}^{(i)}$ with strike K is given by

$$V_0^{(i)\text{ market}}(K) = P_0^{(i)}(x_0)\Phi(\hat{d}_2^{(i)}) \tag{67}$$

where

$$\hat{d}_2^{(i)} = \frac{\log\left(\frac{y_0^{(i)} + r_i}{K + r_i}\right)}{\sigma_i\sqrt{T_i}} - \frac{1}{2}\sigma^{(i)}\sqrt{T_i} \tag{68}$$

Similarly to equations (42) and (43), our bond price on the boundary now becomes

$$\begin{aligned} D_{T_nT_{n+1}}(x_{T_n}) &= \frac{1}{1+\alpha_n y^n_{T_n}} \qquad (69) \\ &= \frac{1}{1+\alpha_i\left(-r_i + [y_0^{(n)} + r_i]\exp\left[\frac{\sigma_n}{(e^{2aT_n}-1)}x_{T_n} - \frac{1}{2}\sigma_n^2 T_n\right]\right)} \end{aligned}$$

and our inversion equation becomes

$$y^{(i)}_{T_i} = -r_i + (y_0^{(i)} + r_i)\exp\left[-\frac{1}{2}(\bar{\sigma}_i)^2 T_i - \bar{\sigma}_i\sqrt{T_i}\Phi^{-1}\left(\frac{J_0^{(i)}(x^*)}{P_0^{(i)}(x_0)}\right)\right] \tag{70}$$

These are the only changes that need to be made to the original model. Pietersz and Pelsser [14] implement the displaced diffusion model and find the following: the displaced diffusion model fits market volatility well for at-the-money (ATM) and out-the-money (OTM) options but fits in-the-money (ITM) options poorly with underfitting errors of up to 21% in their example. These errors are solely due to the displaced diffusion, and not the Markov-Functional model (or Libor Market model) that they implement.

Pietersz and Pelsser [14] implement the following models for their comparison

- The Markov-Functional model using the volatility at the strike price.
- The Markov-Functional model using the ATM volatility.
- The displaced diffusion Markov-Functional model.
- The Separable Market Model (SMM).

They find that a 10% change[17] in the mean reversion parameter of the Markov-Functional model is equal to a 1% parallel volatility shift. For ATM volatilities, the difference in Vega due to the smile corresponds to a parallel

[17]Which they mention is a rather large figure in the market

volatility shift that ranges from 5.4% to -0.9%. For per strike volatilities[18], this difference ranges from -7.9% to 1%. Also note that the model is underfitting the market, meaning that the effect is actually larger. It would thus seem that the effect of the volatility smile is much more significant than the effect of mean reversion.

7 The volatility smile

Other than the displaced diffusion model, all the models have implicitly assumed that there is no volatility smile or skew by using the Black formula, with constant volatility, to calculate market prices. With the exception of the displaced diffusion, no papers have considered the impact of the smile on pricing or provided formulae to calibrate in the presence of a smile. Hunt and Kennedy [11] note that one of the strengths of Markov-Functional models is that they can easily calibrate to a smile. This is a very important consideration in currencies where there is a large smile/skew [19]. Modelling the rates using a log-normal assumption as we have done above is inadequate in these situations.

To clarify, in equations (61) and (50) we calibrated to the market by inverting the Black formula using a constant market quoted volatility which is independent of the strike. We cannot simply substitute a series of different market volatilities in those equations due to the considerations discussed below, which must be taken into account in order to use a strike dependent volatility. The considerations are

- Consistency of the model inside the boundary curve with the model definitions outside of the boundary curve. This includes considering models such as the displaced diffusion model which ensure this consistency.
- Changing the boundary curve so that there is no need for this consistency.
- Interpreting the Black formula as a metric and why this is consistent with the other assumptions made.
- Dealing with strike dependent volatility as we are solving for the strike.

[18] These are the volatilities at different strikes.

[19] Hunt and Kennedy [11] mention that the Yen has a large skew.

We saw in the displaced diffusion model that only two things changed when we changed the model to calibrate to a skew: the formula to obtain market prices and the functional form of the PDB for times outside of the boundary curve. There are two routes that can be followed to calibrate to a smile.

1. Develop or use a model that incorporates 'smile' dynamics such as the displaced diffusion model and incorporate this model in the specification of the Markov-Functional model.

2. Change the definition of the boundary so that we will not have inconsistent prices and calibrate to prices calculated in an arbitrary manner.

The first method was applied in section 6.1. It has the advantage that we have a pricing formula consistent with the smile and that we have consistency with the functional form assumed outside the model boundary. The second method is discussed here. Recall assumption (2) in the framework of the model:

The PDB prices are of the form: $D_{tS} = D_{tS}(x_t), \quad 0 \leq t \leq \partial_B \leq S$ *for some boundary curve* $\partial_B : [0, S] \to [0, S]$

This assumption only requires us to specify the functional form of the PDB for $D_{Bt}(x_B),\ B \leq t \leq S$ for reasons discussed in section 4. If we have $S > B$ then we need to specify a functional form that is consistent with the rest of our model as we did in the Libor model. If we define B so that $B = S$ then the only specification required is $D_{BB}(x_B) \equiv 1$. This means that we do not need a market pricing formula that is consistent with our assumptions outside of the boundary as we do not make any assumptions outside of our boundary. This leaves only the method to interpolate between various market prices.

If we had a continuum, or a least a very fine grid, of market prices, this next step would not be necessary. We would numerically calculate the strike that corresponded to a specific option price. In practice, we will only have a few prices and need to interpolate between them. We assumed that the quoted market price and quoted volatility are equivalent due to the use of the Black '76 formula. This assumption can be changed with ease to suit the market at hand.

We could use the Black formula for interpolation between prices at different *strikes* by interpolating between the quoted volatilities and calculate the corresponding continuum of market prices. Although this seems inconsistent

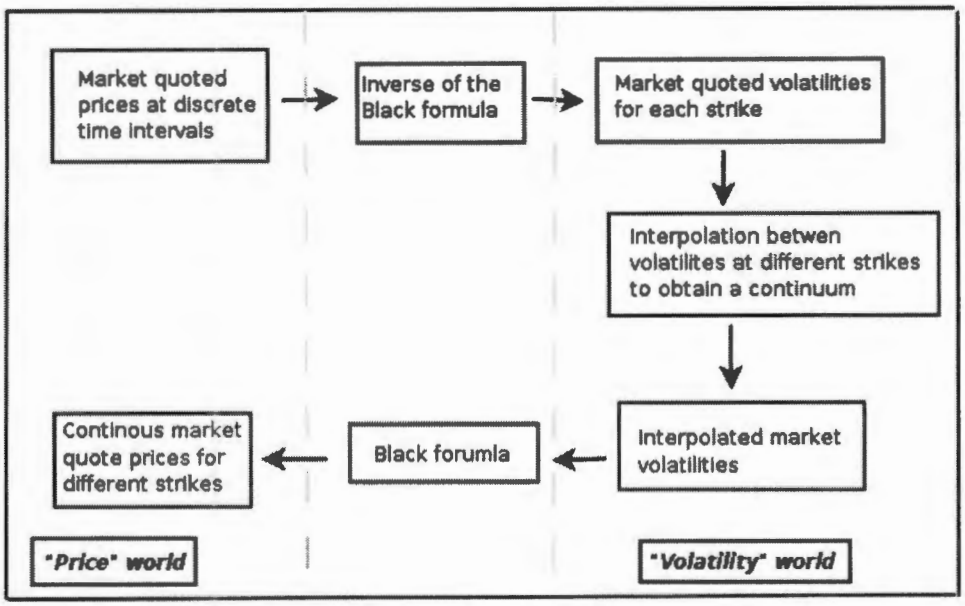

Figure 4: Interpolation between prices using volatility interpolation and the Black formula to move between different 'worlds'

with the Black formula[20] it is not. This is because Black formula is not our pricing model but rather a way of converting quoted volatilities into prices. In other words, the Black formula is now a metric. By selecting the correct boundary curve, there are no assumptions in our model that require us to use the Black formula. This means that it can be seen as an interpolation tool. Instead of interpolating between prices directly, we move to a simpler 'world', interpolate between volatilities and move back to the 'price world'[21]. The Black formula is the door between the 'worlds'. Note that this approach of 'playing with the equations' is not consistent with the underlying assumptions of the Black model. Figure 4 illustrates this.

The Black formula can be inverted to find the strike price as a function of the model price by equating the formula's price and the model price. The volatility may depend on the strike. The volatility could be represented by a parameterized function such as a polynomial. Parameterization should not pose a problem as the form of the volatility smile is typically simple. This parameterized formula should be put in place of the volatility in the inverting equation. In some cases the strike could be found numerically, but in others

[20]The Black model assumes that volatility is constant at different strikes.
[21]This concept is similar to using Fourier transforms for integrating.

it could be found analytically.

As a simple example, we assume that we can calibrate volatility to the smile/skew by using

$$\sigma^{(i)}(K) = a_0 + a_1 K + a_2 K^2 \tag{71}$$

where K is the strike and $a_i,\ i = 0, 1, 2$ are calculated parameters. The Black formula for a digital becomes

$$V_0^{(i)}(K) = D_{T_0 T_{i+1}}(x_0)\Phi(d_2^{(i)}) \tag{72}$$

where

$$d_2^{(i)} = \frac{\log(L_0^{(i)}/K)}{(a_0 + a_1 K + a_2 K^2)\sqrt{T_i}} - \frac{1}{2}(a_0 + a_1 K + a_2 K^2)\sqrt{T_i} \tag{73}$$

This equation cannot be analytically inverted and would need to be solved numerically. i.e. solve for K in

$$\frac{1}{2}(a_0 + a_1 K + a_2 K^2)^2 + T_i \log(K) = \log(L) - \Phi^{-1}\left(\frac{J(x^*)}{D_{T_0 T_{n+1}}(x_{T_0})}\right) \tag{74}$$

Also note that there could be up to four roots. We would need to select the correct one and ensure that the strikes generated, i.e. the roots selected, are consistent and smooth with each other. Most notably, we need to ensure that the functional form of the modelled rate is still monotonic and increasing as assumed.

8 Specific models under the spot measure

8.1 The Libor Markov-Functional model in the spot measure

So far all the models have been derived under the terminal measure. This has allowed us to avoid path dependency of the numeraire and exploit the Markovian nature of the model to its fullest. Spot measure models provide an alternative approach. Note that the assumptions and theory of the model framework remain unchanged even though we have a different numeraire and associated measure.

Spot measure models are important for the following reasons. Fries and Rott [7], mention that certain products are best priced under an appropriate measure change. A certain measure is chosen in order to simplify the pricing. Market conventions will dictate which measure is more appropriate. Both the spot and terminal measure models can be calibrated to relevant products, although it will be more difficult if the inappropriate measure is chosen. The spot measure turns out to be the natural measure for the three-factor cross currency model.

Spot measure models have a different interpretation of the time horizon than terminal measure models. The reason for this is that we are moving forward through time, starting at T_0, instead of backwards from some set point. To start our recursive procedure we need the values of our PDB's on the boundary. As we are starting at time zero, we need only the values of the PDB's at time zero. These PDB's expire at times ranging from T_0 till T_{n+1}, so in this sense, there is still a boundary curve similar to the terminal measure models.

This section describes the model as found in Fries and Rott [7]. The spot measure, $\mathbb{Q}_0$, is the measure associated with the numeraire

$$N_{T_i} \equiv \Pi_{k=0}^{i-1}(1 + L_{T_k}(T_{k+1} - T_k)) \tag{75}$$

This is a product of repeated investments, which in the continuous case would be an integral, but here it is the product of the shortest bonds in our time grid. This numeraire fits into the Markov-Functional framework as follows: we assume that our Libor rate is a deterministic function of a Markov process x_{T_k} i.e.

$$L_{T_k} \equiv L_{T_k}^{(k)}(x_{T_k}) = \frac{1 - D_{T_kT_{k+1}}(x_{T_k})}{D_{T_kT_{k+1}}(x_{T_k})(T_{k+1} - T_k)} = f(x_{T_k}) \tag{76}$$

This leads us to a path dependent numeraire as N_{T_i} depends on all previous rates and hence on all the previous x_{T_i}. Our numeraire, N_{T_i} is clearly $\mathcal{F}_{T_{i-1}}$ measurable. The path dependency would appear to make our computations inefficient, but this problem can be mitigated and will be discussed in section 14.4.3.

We use a forward induction procedure to calculate the functional form of the Libor rate. We have $N_{T_0} \equiv 1$ so we now need to define the forward induction step. Let $V_{T_i}(T_k)$ be the value at time T_k of a product with a time T_{i+1} value $V_{T_{i+1}}(L_{T_i})$ depending only on L_{T_i}. Using the principle of risk neutral valuation, the value is

$$\begin{aligned} V_{T_i,K}(T_0) &= E^{\mathbb{Q}_0}\left(\frac{V_{T_i,K}(T_{i+1})}{N_{T_{i+1}}}|\mathcal{F}_{T_0}\right) \\ &= E^{\mathbb{Q}_0}\left(\frac{V_{T_i,K}(T_{i+1})}{(1+L^{(i)}_{T_i}(x_{T_i})(T_{i+1}-T_i))N_{T_i}}|\mathcal{F}_{T_0}\right) \end{aligned} \tag{77}$$

Define our derivative payout function, paid at time T_{i+1}, equal to

$$V_{T_i,K}(T_{i+1}, L_{T_i}) \equiv \begin{cases} 1 + L_{T_i}(T_{i+1} - T_i) & \text{if } L_{T_i} - K > 0 \\ 0 & \text{else} \end{cases}$$

This can be seen as a portfolio of a caplet and a digital caplet or as a digital caplet in arrears. Equation (77) becomes

$$\begin{aligned} & E^{\mathbb{Q}_0}\left(\frac{1_{L^{(i)}_{T_i}(x_{T_i})>K}}{N_{T_i}}|\mathcal{F}_{T_0}\right) \\ = \quad & E^{\mathbb{Q}_0}\left(E^{\mathbb{Q}_0}[1_{L^{(i)}_{T_i}(x_{T_i})>K}|\mathcal{F}_{T_{i-1}}]\frac{1}{N_{T_i}}|\mathcal{F}_{T_0}\right) \end{aligned} \tag{78}$$

due to the $F_{T_{i-1}}$ measurability of N_{T_i}. We assume that the functional form of the Libor rate

$$\xi \mapsto L_{T_i}(\xi) \tag{79}$$

is monotonic and increasing. We then have the identity (35) allowing us to set

$$L^{(i)}_{T_i}(x^*) = K^{(i)}(x^*) \tag{80}$$

The inner expectation in equation (78) becomes

$$
\begin{aligned}
V_{T_i,K}(T_{i-1}) &= E\left(\frac{V_{T_i,K}(T_i)}{N_{T_i}}|\mathcal{F}_{T_{i-1}}\right) && (81)\\
&= E^{\mathbb{Q}_0}(1_{L_{T_i}(x_{T_i})>K}|\mathcal{F}_{T_{i-1}}) && (82)\\
&= E^{\mathbb{Q}_0}(1_{L_{T_i}(x_{T_i})>K}|(T_{i-1},\xi)) && (83)\\
&= E^{\mathbb{Q}_0}(1_{x_{T_i}>x^*}|(T_{i-1},\xi)) && (84)\\
&= \int_{x^*}^{\infty}\phi[(y-\xi),\sigma(T_{i-1},T_i)]dy && (85)\\
&= \int_{x^*}^{\infty}\phi_{x_{T_i}|x_{T_{i-1}}}(y)dy && (86)
\end{aligned}
$$

where $\phi(\mu,\sigma)$ is the standard Normal probability density function and $\phi_{x_{T_i}|x_{T_{i-1}}}$ is the density of the Markov process conditioned on the state $x_{T_{i-1}}$. Our Libor rate maps to the Markov process in the same way as the terminal measure models. Equation (80) links the market strikes to the Libor rates. The expectation is thus a conditional Gaussian integral of an indicator function due to the definition of Markov process. To get the value at time zero we take the expectation

$$
\begin{aligned}
& E^{\mathbb{Q}_0}\left[\frac{V_{T_i,K}(T_{i-1})}{N_{T_{i-1}}}|\mathcal{F}_0\right]\\
= \quad & E^{\mathbb{Q}_0}\left[\int_{x^*}^{\infty}\phi_{x_{T_i}|x_{T_{i-1}}}(y))dy\frac{1}{N_{T_i}}|\mathcal{F}_0\right] && (87)
\end{aligned}
$$

where $\phi_{x_{T_i}}$ is the unconditional density of the Markov Process. We cannot represent this expectation by a Gaussian integral over the inner integral as we did in the terminal measure model. The reason for this is that we have path dependence. Instead we must use the tower law.

$$
\begin{aligned}
& E^{\mathbb{Q}_0}\left[\frac{V_{T_i,K}(T_{i-1})}{N_{T_{i-1}}}|\mathcal{F}_0\right] && (88)\\
= \quad & E^{\mathbb{Q}_0}\left[E^{\mathbb{Q}_0}(1_{L_{T_i}(x_{T_i})-K)}|\mathcal{F}_{T_{i-1}})|\mathcal{F}_0\right] && (89)\\
= \quad & E^{\mathbb{Q}_0}\left(E^{\mathbb{Q}_0}\left[E^{\mathbb{Q}_0}(1_{L_{T_i}(x_{T_i})-K}|\mathcal{F}_{T_{i-1}})\frac{1}{N_{T_i}}|\mathcal{F}_{T_{i-2}}\right]|\mathcal{F}_0\right) && (90)\\
= \quad & E^{\mathbb{Q}_0}(E^{\mathbb{Q}_0}....(E^{\mathbb{Q}_0}(E^{\mathbb{Q}_0}(1_{L_{T_i}(x_{T_i})-K}|\mathcal{F}_{T_{i-1}})\frac{1}{N_{T_i}}|\mathcal{F}_{T_{i-2}})|...)|...)|\mathcal{F}_0) && (91)
\end{aligned}
$$

Thus we work iteratively back through time. We can do this as we know the functional form of the numeraire at all the previous steps.

Note that in a practical implementation this is quite different to the terminal measure: in the terminal measure the outer expectation could be calculated in one step due to the Markovian nature of the numeraire. Here we need to proceed iteratively backwards through time. Efficient methods for calculating this expectation are dealt with in section 14.4.3.

Once we have this expectation we find the market strike, $K_{T_i}^{(i)}(x^*)$, which corresponds to this price. We do this for all the x^*'s and proceed forwards through time.

We will need a continuum, or at least a very fine grid, of market prices to find the corresponding strike. This means that we will need to interpolate between the actual market prices for various strikes. This could be done in a similar fashion to the terminal measure model: we interpolate between prices by using volatility interpolation and calculate the prices using the Black '76 formula. This allows calibration to a volatility skew or smile.

8.2 Relating the spot and terminal measure models

This section draws on Fries and Rott [7]. In the continous case, the spot and terminal measure models are equivalent by a change of numeraire. However, in the discrete case, the spot and terminal measure models are not equivalent although one would initially suspect that they are by a change of numeraire. The reason for the non-equivalence is the fact that we are working in discrete time and the drift process that links the two measures via a Girsanov transformation cannot be discretized. Allowing this drift process to be a continuous process induces path dependency in the driving process which is a very undesirable characteristic. This means that we cannot use the same functionals for the spot and terminal measure by changing the respective driving process and measure. This is explained formally below.

Suppose that we have numeraires N and M with associated EMM's Q^N and Q^M. We have

$$E^{Q_M}\left(\frac{V_{T_{i+1}}}{N_{T_{i+1}}}\left[\frac{N_{T_{i+1}}M_{T_i}}{M_{T_{i+1}}N_{T_i}}\right]|F_{T_i}\right) = E^{Q_N}\left(\frac{V_{T_{i+1}}}{N_{T_{i+1}}}|\mathcal{F}_{T_i}\right) \tag{92}$$

where

$$C_{T_i,T_{i+1}} \equiv \left[\frac{N_{T_{i+1}}M_{T_i}}{M_{T_{i+1}}N_{T_i}}\right] \tag{93}$$

is our $\mathcal{F}_{T_{i+1}}$-measurable change of numeraire integration kernel. To prove by contradiction that V cannot remain the same under a change of numeraire in discrete time, assume that V, N and M are all functions of our Markov process and that the functional form of V does not change under the new measure. Our Euler discretization of the Markov process is

$$x_{T_{i+1}} = \begin{cases} x_{T_i} + \sigma(T_i)\Delta W_{T_i,T_{i+1}} & \text{under } Q^N \\ x_{T_i} + \mu(T_i, x_{T_{i+1}})\Delta T_i + \sigma(T_i)\Delta W_{T_i,T_{i+1}} & \text{under } Q^M \end{cases}$$

where $\Delta(T_i) = T_{i+1} - T_i$ and the additional drift term is due to the change in measure[22]. Writing out our conditionings [23] and using the Markov property we obtain

$$E^{Q_N}\left(\frac{V_{T_{i+1}}}{N_{T_{i+1}}} C_{T_i,T_{i+1}} | x_{T_i} + \mu(T_i, x_{T_i})\Delta T_i\right) = E^{Q_N}\left(\frac{V_{T_{i+1}}}{N_{T_{i+1}}} | x_{T_i}\right) \quad (94)$$

$$E^{Q_N}\left(\frac{V_{T_{i+1}}}{N_{T_{i+1}}} C_{T_i,T_{i+1}} | x_{T_i}\right) = E^{Q_N}\left(\frac{V_{T_{i+1}}}{N_{T_{i+1}}} | x_{T_i} + \mu(T_i, x_{T_i})\Delta T_i\right) \quad (95)$$

If we let V_{T_i} be a bond maturing at time T_{n+1} and consider the equations at time T_n we can see that $\mu(T_i, x_{T_i})$ is determined by $C_{T_i,T_{i+1}}$. If we hold μ fixed, equation (95) cannot hold for general functions V. The reason for this is that when we change the numeraire over a discrete time step we get a change in the conditional probability distribution which cannot necessarily be accounted for by a shift in the mean as $\int_t^{t+\Delta t} \mu(t)dt$ is not $\mathcal{F}_t$ previsible. i.e. Girsanov's theorem can't be extended to this discrete time situation as we do not have infinitesimal time steps. The solution is to have a path dependent drift

$$x_{T_{i+1}} = x_{T_i} + \int_{T_i}^{T_{i+1}} \mu(t, x_t)dt + \sigma(T_i)\Delta W_{T_i,T_{i+1}} \quad (96)$$

under Q^M which will work as we now do have infinitesimal time steps. This is just another way of phrasing Girsanov's theorem. The other solution is to recalculate the functional under the new measure. Another way of seeing this problem is that our model is arbitrage-free in continuous time but our discrete time approximation is not. This is the reason for developing an arbitrage free model in discrete time for the multi-factor case which is discussed in section 9.

[22] See Girsanov's theorem (2.8)

[23] i.e. writing out all the relevant information that the filtration contains, which is the Markov process under the corresponding measure

When we calculate different functional forms for the two different measures the transitional distributions will differ between the spot and the terminal measure resulting in different option prices for options that are path dependent.

8.3 Which model is appropriate

The terminal measure and spot measure models give different prices. Both the models are arbitrage free and use the same framework and thus they are both appropriate. The reason for the difference in the prices is that we are using a *single* factor model which can only calibrate to a single distribution. Therefore we cannot calibrate to all the marginal distributions. Both the models approximate the marginal distributions differently. Certain products such as Bermudan swaptions are dependent on the marginal distributions resulting in different prices when we have different marginal distributions.

9 Multi Factor Markov-Functional Models

9.1 A two factor Markov-Functional model

The following section presents the two factor Markov-Functional model as found in Fries and Rott [7]. Intermediate workings and explanations have been added.

Heuristically, the model works as follows: we are interested in calculating the price of a derivative based on a foreign bond. We have two factors and hence there are two Markov processes, one which has a non-zero deterministic drift. One factor drives the domestic bond and the other drives the foreign exchange rate. We are working under a single EMM[24] $\mathbb{Q}$ with associated numeraire N. We calibrate our model by selecting a functional form of the domestic bond and the foreign exchange rate. The functional form of the domestic bond is calculated in exactly the same way as the single factor model, without regard to the second factor. We then assume that the foreign exchange rate and the domestic bond are independent and that the foreign bond is deterministic. It will become necessary to determine a relation between the functional form of the foreign exchange rate and the drift of the associated driving Markov process. By using the identity

$$\text{foreign bond} = \text{domestic bond} \times \text{forex rate} \tag{97}$$

and rearranging equations we can find the relation between the drift of the process driving the foreign exchange rate and the functional form of the forex rate. We then calibrate by selecting or calculating a functional and calculating the drift.

Formally, we have two factors and thus two Markov processes that are defined as

$$\begin{aligned} dx &= \sigma_x(t)dW_1 \\ dy &= \mu(t,x,y)dt + \sigma_y(t)dW_2 \end{aligned} \tag{98}$$

with the assumption of independent increments so that $< dW_1, dW_2 >= 0$. $\sigma_x(t)$, $\sigma_y(t)$ and $\mu(t,x,y)$ are deterministic functions. We have added the drift in the equation of dy due to change in numeraire considerations. We will be calibrating to two different markets and hence require the addition of a drift term to be able to use a single measure and numeraire[25].

[24] Any EMM, this could be the terminal or spot measure

[25] See Girsanov's theorem (2.8)

Note that in this model we do require the functional forms of the foreign exchange rate and the domestic bond under the same measure in order to be able to calculate expectations.

There are two routes that can be followed in order to make the model arbitrage free: the first is to derive an arbitrage free model in continuous time and then approximate this by using a discrete version which will then only be arbitrage free to the extent that our approximation is accurate[26]. The reason for the simple[27] approximation not being perfectly accurate is that we have non-linear drift in the y term. Previously, this was not a problem as we had zero drift. Section 8.2 shows why this non-linear drift is a problem due to the discretization.

The second route is to derive an arbitrage free model in discrete time. We will therefore focus on making the discrete time model arbitrage free. The discretization of our Markov processes gives us

$$\begin{aligned} \Delta x_{T_{i-1}} &= \sigma_{x,i-1}\sqrt{\Delta T_{i-1}}\Delta W_1 \\ \Delta y_{T_{i-1}} &= \mu(T_{i-1}, x_{T_{i-1}}, y_{T_{i-1}})\Delta T_{i-1} + \sigma_{y,i-1}\sqrt{\Delta T_{i-1}}\Delta W_2 \end{aligned} \tag{99}$$

where $\Delta W_i,\ i = 1, 2$, are independently distributed $N(0, 1)$ and

$$\sigma_{j,i-1} \equiv \sqrt{\frac{1}{\Delta T_{i-1}} \int_{T_{i-1}}^{T_i} \sigma_j^2(t)dt}, \quad j = x, y \tag{100}$$

The approximations of $x_0 + \int_0^{T_i} dx$ and similarly for y are

$$\begin{aligned} x_{T_0} &= x_0 \\ y_{T_0} &= y_0 \\ x_{T_i} &= x_{T_{i-1}} + \Delta x_{T_{i-1}} \\ y_{T_i} &= y_{T_{i-1}} + \Delta y_{T_{i-1}} \end{aligned} \tag{101}$$

Define $\hat{D}_{T_i,T_{i+1}}$ as the value of a foreign pure discount bond at time T_i which pays out at time T_{i+1}. Define FX_{T_i} as the foreign exchange rate in units of local currency per foreign currency at time T_i. Our numeraire, N_{T_i}, which is a function of the Markov process is defined as in the single factor case.

[26]The approximation can be made exact, but this requires a state dependent drift which is computationally intensive

[27]Linear drift approximation

Calibration is done as if we were only considering a single factor and we must therefore only consider the calibration of the second factor. We have

$$\begin{aligned}\frac{FX_{T_i}\hat{D}_{T_iT_{i+1}}}{N_{T_i}} &= E^{\mathbf{Q}_0}\left[\frac{FX_{T_{i+1}}\hat{D}_{T_{i+1}T_{i+1}}}{N_{T_{i+1}}}|\mathcal{F}_{T_i}\right] \\ &= E^{\mathbf{Q}}\left[\frac{FX_{T_{i+1}}}{N_{T_{i+1}}}|\mathcal{F}_{T_i}\right] \end{aligned} \tag{102}$$

Using the assumption that the interest rates and the foreign exchange rates are independent over each time step we obtain

$$\begin{aligned}\frac{FX_{T_i}\hat{D}_{T_iTi+1}}{N_{T_i}} &= E^{\mathbf{Q}}\left[\frac{FX_{T_{i+1}}}{N_{T_{i+1}}}|\mathcal{F}_{T_i}\right] \\ &= E^{\mathbf{Q}}\left[FX_{T_{i+1}}|\mathcal{F}_{T_i}\right]E^{\mathbf{Q}}\left[\frac{1}{N_{T_{i+1}}}|\mathcal{F}_{T_i}\right] \end{aligned} \tag{103}$$

Now we divide this equation by

$$\frac{D_{T_iT_{i+1}}}{N_{T_i}} = E^{\mathbf{Q}}\left[\frac{D_{T_{i+1}T_{i+1}}}{N_{T_{i+1}}}|\mathcal{F}_{T_i}\right] = E^{\mathbf{Q}}\left[\frac{1}{N_{T_{i+1}}}|\mathcal{F}_{T_i}\right] \tag{104}$$

and obtain

$$\frac{FX_{T_i}\hat{D}_{T_iTi+1}}{D_{T_iTi+1}} = E^{\mathbf{Q}}\left[FX_{T_{i+1}}|\mathcal{F}_{T_i}\right] \tag{105}$$

Next, as in the one dimensional case, we make the assumption that FX_{T_k} is a function of y_{T_k} only. Writing equation (105) in terms of the functional form we obtain

$$FX_{T_i}(y_{T_i}) = \frac{D_{T_iT_{i+1}}(x_{T_i})}{\hat{D}_{T_iT_{i+1}}(y_{T_i})}E^{\Delta y_{T_i}}\left[FX_{T_{i+1}}(y_{T_{i+1}})|(T_i, x_{T_i}, y_{T_i})\right] \tag{106}$$

We denote $E^{\mathbf{Q}}$ by $E^{\Delta y_{T_i}}$ when the expectation contains only the y terms. This is done because the calculation of the expectation of the random variable $y_{T_{i+1}} = y_{T_i} + \Delta y_{T_i}$ conditioned on a particular state, $y_{T_i} = \xi$ requires only the distribution of $\Delta y_{T_i}(\xi)$. Loosely speaking, the functional form depends only on the distribution of the increment. Similarly, this is done for x and for the combination of x and y in expectation.

This equation (106) is very important. It gives us the relation between the functional form of the FX rate, $\xi \mapsto FX_{T_i}(\xi)$, and the specification (μ, σ_y) of

the transition probability, $\Delta y(T_i)$. Thus specifying the functional form and the underlying volatility, σ_y, will give us the no arbitrage drift, μ, according to equation (106). Alternatively we can specify the drift and calculate the functional form. If we choose to do this, setting the terminal functional form will specify all the functional forms for times prior to the terminal time. This is therefore not done.

All that remains is the calibration framework. There are two methods to calibrate the model: the first is to choose a specific functional form and calibrate the model by selecting the volatility $\sigma_{y,i}$ and implying the drift. This allows us to, in some instances, calculate analytical formulae for the drift. The second is to choose the functional form at each time step, given our implied market volatility and then calculate the drift according to equation (106). This allows for a full calibration of a volatility smile, but clearly will have additional computational costs due to the calculation of the no-arbitrage drift.

9.2 Calibrating the two-factor model with independent increments

This section draws from Fries and Rott [7]. The model price of a foreign exchange (FX) option struck at time T_i and strike K is

$$\begin{aligned} & V^{T_i,K}_{\text{FXOption}}(T_0) \qquad (107)\\ = \; & N_{T_0} E^{\Delta x^{T_i}_{T_0}, \Delta y^{T_i}_{T_0}} \left[\frac{(FX_{T_i}(y_{T_i}) - K)^+}{N_{T_i}(x_{T_i})} | (T_0, x_0, y_0) \right] \\ = \; & N_{T_0} E^{\Delta x^{T_i}_{T_0}, \Delta y_{T_0}} \left[E^{\Delta x^{T_i}_{T_0}, \Delta y_{T_0}} \left(\frac{(FX_{T_i}(y_{T_i}) - K)^+}{N_{T_i}(x_{T_i})} | (T_{i-1}, x_{T_{i-1}}, y_{T_{i-1}}) \right) | (T_0, x_0, y_0) \right] \end{aligned}$$

by the tower law. We calibrate by moving forward in time and thus have $FX_{T_{i-1}}$ and $\sigma(T_j, y)$ for $T_j < T_{i-1}$ from our previous calibration. Now consider

$$\begin{aligned} & \hat{V}^{T_i,K}_{\text{FX Option}} \qquad (108)\\ = \; & E^{\Delta x, \Delta y} \left(\frac{[FX_{T_i}(y_{T_i}) - K]^+}{N_{T_i}(x_{T_i})} | (T_{i-1}, \xi, \nu) \right) \\ = \; & E^{\Delta y} \left([F_{T_i}(y_{T_i}) - K]^+ | (T_{i-1}, \nu + \mu_{i-1}(\xi) \Delta T_{i-1}) \right) \cdot E^{\Delta x} \left(\frac{1}{N_{T_i}(x_{T_i})} | (T_{i-1}, \xi) \right) \end{aligned}$$

where ξ and ν denote states relating to $x_t(\xi)$ and $y_t(\nu)$ for relevant t. The expectations with respect to Δx and Δy can be denoted by expectations

with respect to $\sigma_{x,i-1}\sqrt{\Delta T_{i-1}}\Delta W_1$ and $\sigma_{y,i-1}\sqrt{\Delta T_{i-1}}\Delta W_2$ respectively, due to the discretization in equation (99). We can split the expectation as the marginal distributions are independent of each other with known laws.

The addition of the drift term to the information we condition on when we split the expectation can be explained by recalling equation (99): the process for the increment in y_{T_i} has an additional drift term. This needs to be included in the information on which we are conditioning. An alternative way of seeing this is that we need to change the drift term (which was previously zero) when we want to change the measure[28]. We changed the measure in such a way that both the expectations can be taken with respect to $\mathbb{Q}$.

Suppose that the backward transition probabilities are known for time T_{i-1} to time T_0. We calculate our model prices for fixed states, ν^* and map this back to the strike price of the option in the market to obtain $\nu \mapsto FX_{T_i}(\nu)$ and proceed iteratively forward through time, similarly to the Markov-Functional model under the spot measure. We don't have to consider $\xi \mapsto D_{T_i}(\xi)$. The reason for this is, as mentioned earlier, the functional form of $N_{T_i} = D_{T_iT_{n+1}} = D_{T_i}$ is calculated as if we were only considering a single factor model. The functional form of D_{T_i} is thus calculated as before.

In order to calculate the model price, which involves an expectation with respect to the Markov process, we need to solve for the drift of the Markov process using equation (105) which has the disadvantage that it is numerically intensive. Solving for the drift numerically has the advantage that we can calibrate to a smile. An alternative approach would be to set the functional form of the forex rate and obtain an analytical or numerical solution for the drift. Specific functional forms such as the linear and exponential form are discussed in Fries and Rott [7] although no functional form that can calibrate to a smile or a skew are presented.

9.3 A three-factor cross currency model

Fries and Rott [7] also derive a three-factor cross currency model that is the natural extension of the two factor model. The details are sketched below[29].

[28] See Girsanov's Theorem (2.8)

[29] Other than a few extra explanatory steps the equations are taken exactly as they are in Fries and Rott [7]

Under the spot measure, $\mathbb{Q}_0$, the three Markov processes are

$$\begin{aligned} dx &= \sigma_x(t)dW_1 \\ dy &= \mu(t,x,y,z)dt + \sigma_y(t)dW_2 \\ dz &= \sigma_z(t)dW_3 \end{aligned} \tag{109}$$

with the associated approximations

$$\begin{aligned} \Delta x_{T_{i-1}} &= \sigma_{x,i-1}\Delta W_1 \\ \Delta y_{T_{i-1}} &= \mu(T_{i-1}, x_{T_{i-1}}, y_{T_{i-1}}, z_{T_{i-1}})(T_i - T_{i-1}) + \sigma_{y,i-1}\Delta W_2 \\ \Delta z_{T_{i-1}} &= \sigma_{z,i-1}\Delta W_3 \end{aligned} \tag{110}$$

Assume that all three Brownian motions are independent i.e. they have independent increments[30]. Also assume that the Libor rate is a function of x only, that the foreign exchange rate is a function of y only and that the foreign Libor rate, $\hat{L}$ is a function of z only.

Let $\hat{V}_{T_i}(\hat{L}_{T_i})$ denote the value at time T_i of a foreign currency product that depends on the foreign Libor rate at time T_i, $\hat{L}_{T_i}$, which has the value $\hat{V}_{T_{i+1}}(\hat{L}_{T_i}) \cdot FX_{T_{i+1}}$ at time T_{i+1} to a domestic investor. This is a traded asset and can thus be valued using the fundamental pricing theorem.

$$\begin{aligned} \hat{V}_{T_0}(\hat{L}_{T_i}) \cdot \frac{FX_{T_0}}{N_{T_0}} &= E^{\mathbb{Q}_0}\left(\frac{\hat{V}_{T_i}(\hat{L}_{T_i}) \cdot FX_{T_{i+1}}}{N_{T_{i+1}}} | \mathcal{F}_{T_0}\right) \\ &= E^{\mathbb{Q}_0}\left(\hat{V}_{T_i}(\hat{L}_{T_i}) \cdot E\left[\frac{FX_{T_{i+1}}}{N_{T_{i+1}}} | F_{T_i}\right] | \mathcal{F}_{T_0}\right) \end{aligned} \tag{111}$$

by the tower law. Now

$$E^{\mathbb{Q}_0}\left(\frac{FX_{T_{i+1}} D_{T_{i+1}T_{i+1}}}{N_{T_i}} | \mathcal{F}_{T_i}\right) = E^{\mathbb{Q}_0}\left(\frac{FX_{T_{i+1}}}{N_{T_i}} | \mathcal{F}_{T_i}\right) = \frac{\hat{D}_{T_iT_{i+1}}}{N_{T_i}} FX_{T_i} \tag{112}$$

Substituting this back in we get

$$\hat{V}_{T_0}(\hat{L}_{T_i}) \cdot \frac{FX_{T_0}}{N_{T_0}} = E^{\mathbb{Q}_0}\left(\frac{\hat{V}_{T_i}(\hat{L}_{T_i})}{1 + \hat{L}_{T_i}(T_{i+1} - T_i)} \cdot \frac{FX_{T_i}}{N_{T_i}} | \mathcal{F}_{T_0}\right) \tag{113}$$

[30]This assumption can be relaxed at the expense of additional computational costs. See the section on correlated multi-factor models

by the definition of the (foreign) Libor rate. Using the assumption of independent increments we obtain

$$
\begin{aligned}
& E^{\mathbf{Q}_0}\left(E^{\mathbf{Q}_0}\left[\frac{\hat{V}_{T_i}(\hat{L}_{T_i})}{1+\hat{L}_{T_i}(T_{i+1}-T_i)}|\mathcal{F}_{T_{i-1}}\right]\cdot E^{\mathbf{Q}_0}\left[\frac{FX_{T_i}}{N_{T_i}}|\mathcal{F}_{T_{i-1}}\right]|\mathcal{F}_{T_0}\right) \\
&= E^{\mathbf{Q}_0}\left(E^{\mathbf{Q}_0}\left[\frac{\hat{V}_{T_i}(\hat{L}_{T_i})}{1+\hat{L}_{T_i}(T_{i+1}-T_i)}|\mathcal{F}_{T_{i-1}}\right]\cdot\left[\frac{FX_{T_{i-1}}}{(1+\hat{L}_{T_{i-1}}(T_i-T_{i-1}))N_{T_{i-1}}}\right]|\mathcal{F}_{T_0}\right) \\
&= E^{\mathbf{Q}_0}\left(E^{\mathbf{Q}_0}\left[\frac{\hat{V}_{T_i}(\hat{L}_{T_i})}{(1+\hat{L}_{T_i}(T_{i+1}-T_i))\cdot(1+\hat{L}_{T_{i-1}}(T_i-T_{i-1}))}|\mathcal{F}_{T_{i-1}}\right]\cdot\left[\frac{FX_{T_{i-1}}}{N_{T_{i-1}}}\right]|\mathcal{F}_{T_0}\right) \\
&= \ldots \\
&= E^{\mathbf{Q}_0}\left(\frac{\hat{V}_{T_i}(\hat{L}_{T_i})}{\Pi_{k=0}^{i}(1+\hat{L}_{T_k}(T_{k+1}-T_k))}|\mathcal{F}_{T_0}\right)\cdot\frac{FX_{T_0}}{N_{T_0}} \qquad (114)
\end{aligned}
$$

This means that

$$
\hat{V}_{T_0}(\hat{L}_{T_i}) = E^{\mathbf{Q}_0}\left(\frac{\hat{V}_{T_i}(\hat{L}_{T_i})}{\Pi_{k=0}^{i}(1+\hat{L}_{T_k}(T_{k+1}-T_k))}|\mathcal{F}_{T_0}\right) \qquad (115)
$$

Thus we end up using the foreign money market account as our numeraire. This is a reason for picking the spot measure: we end up with an intuitive interpretation of our numeraire. Next we would proceed with the calibration in a similar way to the two-factor model i.e. we would adjust the functional form and volatility, solve for the no-arbitrage drift and calculate a model price.

9.4 Correlated factors

If our factors are significantly correlated, we will need to use a correlated multi-factor model. Deriving the correlated factor case is simple in theory but computationally intensive as we will have n-dimensional integrals in the n-factor case. As an example, consider the two dimensional case in the context of the models described above. The model will have a double integral as we cannot split up the expectation as we did in equation (114) and (108). The actual process of calculating the model price is rather simple - it is a double integral. The relation of the functional form and the no-arbitrage drift will be similar[31] but the expectation will be a double integral and contain two functionals[32].

[31] It is a trivial exercise to write this out.

[32] In the two factor case described above we will have $FX_t(x)$ and $N_t(x)$ in the expectation.

In Hunt and Kennedy [11] a log-normal two-factor swap model with correlated factors is developed.

This model is developed in continuous time and differs to the models discussed already. It has two factors driving the Libor rate and Libor rate drives the swap rate. This is different to explicitly having one factor driving the Libor rate and one factor driving the swap rate. The functional form of $L_t^{(i)}$ as a function of two factors is specified and derived from an approximate[33] Libor market model. We have our swap rate as a function of our Libor rate which is a function of the two factors i.e. we proceed as we did in the single factor model, but instead of using our Markov process, x_t, as our factor we use

$$\hat{L}_t^{(i)} = f(x_t^{(1)}, x_t^{(2)}) \tag{116}$$

where f is specified and model

$$\hat{y}_{T_i}^{(i)} = \hat{y}_{T_i}^{(i)}(\hat{L}_{T_i}^{(i)}, L_{T_i}^{(j)},\ j > i) = g^{(i)}(x_t^{(1)}, x_t^{(2)}, T_i) \tag{117}$$

This allows us to specify the correlation structure between $L_{T_i}^{(i)}$ and $y_{T_i}^{(i)}$ but clearly results in a double integral. Note that the model is arbitrage free.

[33] Alterations are made to the drift

10 The volatility of the Markov process

The volatility of the Markov Process is the function $\sigma(t)$ in the definition of the process as

$$dx = \sigma(t)dW_t \tag{118}$$

One may be interested to see how different volatilities affect the pricing of the model and exactly what this volatility means. Recall that we are assuming that the entire market of interest is driven by this single underlying process. Intuitively this should mean that the volatility of this process is very important. On the other hand, by definition, the calibration will ensure that we obtain correct market prices for all the options to which we have calibrated. When we price an option such as a cap which uses all the calibration points[34], it seems intuitive that the volatility function, $\sigma(t)$, is arbitrary. However, there could be other properties such as mean reversion that are induced by the volatility function.

In order to understand volatility we need to understand the relation between the functionals and the volatility as well as the properties induced by specific volatility processes.

There is a vast literature in a seemingly related field that links the volatility process to the functionals. By placing volatility restrictions on the Heath-Jarrow-Morton (HJM) framework we obtain models that are Markovian. These models are referenced in section 11. However, there does not appear to be a clear link between the restricted HJM model and the Markov-Functional model.

10.1 Does the constant volatility matter

Consider the following situation: We are using the Libor Markov-Functional model where the Markov process has constant volatility, which is quite low. The market volatility is high. In the calibration we find the function that maps the Markov process to the real world Libor rates. In this function a small jump in the Markov Process results is a large jump in the real world Libor rate in order for our calibration to be correct. In a different situation where the Markov volatility is high and the real world volatility is low we have that a large jump in our Markov process results in a small jump in the

[34] We use all the times for which we obtained market prices to which we calibrated i.e. we use all the times $T_0,...,T_{n+1}$

Libor rates. Thus the choice of our constant Markov volatility does not seem to affect our pricing. When pricing options we did not get any numerical differences by changing the volatility. The options tested were

- A cap
- A cap that pays out $\max[(L-K),0]^2$
- A cap that pays out every second period only.
- All of the above but using a barrier cap instead of a cap

These results do not change depending on the number of discretization points used. One could initially expect that they would as it would imply a poorer total calibration. When pricing we only use the points to which we calibrated meaning that the calibration for all the points used is correct, regardless of the number of points. Note however that this does not mean that the model is calibrated correctly when insufficient points are used as the model is not calibrated to the missing points.

Recall that the rate of interest is a function of the Markov process, x e.g. $L_{T_i}^{(i)}(x_{T_i}) = f^{(i)}(x_{T_i})$. Taking a first order linear approximation yields

$$f^{(i)}(x+0) \approx f^{(i)}(0) + \frac{\partial}{\partial y}[f^{(i)}(y)]_{y=x} \cdot x \tag{119}$$

The variance of this rate is

$$\mathrm{Var}[f^{(i)}(x+0)] = \left(\frac{\partial}{\partial y}[f^{(i)}(y)]_{y=x}\right)^2 \cdot \mathrm{Var}[x] \tag{120}$$

From this it should be clear that whatever we pick the variance of x to be, the functional form of the rate of interest, $f(x_t)$, can be chosen so that the variance of the rate is unchanged. Although this is not robust, it does help in understanding why our results do not change if we change our Markov processes' volatility.

This result may warrant further investigation but because constant volatility is so simple and a more complex volatility structure should be used in practice, an investigation may not be necessary. Rather, more complex volatility structures that provide an intuitive and more realistic approach are examined.

10.2 Mean reversion due to the volatility process

This section draws from the ideas in Hunt et al [10]. For the remainder of this section, volatility refers to average mean square volatility. The question is: how does a specific volatility process, defining the Markov process, affect mean reversion? For a heuristic explanation, consider two separate volatility processes on a Libor rate from time zero (now) till time T_n. They both have the same (average mean square) volatilities that are, say, $\bar{\sigma}$ over the period $[0, T_n]$. However, at time T_i between now and time T_n, the first process has volatility $\sigma_1(T_i)$ and the second process has volatility $\sigma_2(T_i)$, where $\sigma_1(T_i) < \sigma_2(T_i)$. Recall that volatility is an increasing function over time else arbitrage would exist[35]. Thus both the volatility functions are monotonic. This allows us not to have to consider a reversed situation where $\sigma_1(T_i) > \sigma_2(T_i)$ at a later stage.

In order for our second process to have the same volatility, $\bar{\sigma}$, over the whole time period we require its Libor rate to be much less volatile than that of the Libor rate of the first process for the remaining time. Recall that volatility measures the deviation of the rate from the mean rate, which is the expected rate. The only way that the second process is going to be less volatile is if it has a smaller absolute deviation from the mean for the remaining time period. The reason it has a higher volatility at time T_i is because it deviated further from the mean in the period $[0, T_i]$. In order to ensure that the rate of increase in volatility of the second process is less than that of the first process, it needs to deviate less from the mean causing it to be closer to the mean. This results in mean reversion. On the other hand, the first process is not forced to revert to the mean as it can have a higher volatility for the remaining period. Thus if we have two processes with the same volatility over a fixed period of time and one of them has a lower volatility at an intermediate period, the process with the lower volatility will exhibit mean reversion properties. The extent of the mean reversion will be affected by the difference in the volatilities at an intermediate time. This heuristic explanation is not watertight, but it highlights the idea underlying the formal explanation.

We have seen how a volatility process results in mean reversion. Now consider a process that has a constant volatility function.

$$dx = \sigma dW_t \tag{121}$$

[35] If volatility did not increase over time, we would have a process that has zero volatility over some future period. This would mean that the process is predictable and thus arbitrage would exist.

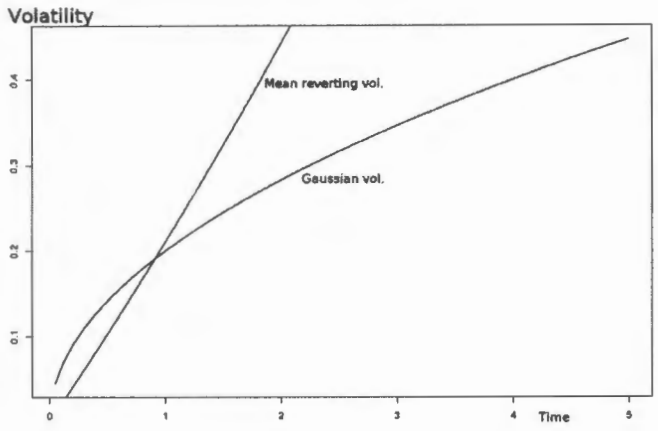

Figure 5: Exponential volatility vs. Gaussian volatility ($\sigma\sqrt{t}$). The mean reversion parameter is 10% and the constant volatility is 20% i.e. $\sigma = 0.2$.

This volatility increases with the square root of time. Let

$$dx = e^{at} dW_t \tag{122}$$

be our other Markov process. The evolution of the volatility follows $\sigma\sqrt{t}$ for the constant volatility and

$$\sqrt{\int_0^t e^{2ax} dx} = \sqrt{\frac{1}{2a}(e^{2at} - 1)} \tag{123}$$

for the exponential volatility. Graphing these functions on the same set of axes we obtain Figure 5. Suppose the terminal time was 1, where the graphs cross in Figure 5. The root mean square of the exponential process increases quicker than that of the process with constant volatility. Market volatilities are quoted assuming that the Black formula holds and hence that the volatility increases with the square root of time. Thus with the exponential volatility we need to start at a much lower model volatility at earlier durations to obtain the same volatility as the market. We need to obtain the same volatility as the market because we are equating market and model prices in order to calibrate. This is exactly the same as the heuristic example: one process has a much lower volatility at an earlier duration but they both have the same terminal volatility. The process with the lower volatility provided mean reversion. Note that this mean reversion property hinges on the assumption that market quoted volatilities increase with the

square root of time. This is the case if we use the Black formula to convert market quoted volatilities to market prices. If this was not the case a simple analysis would need to be done to see which process result in mean reversion.

Any other process that has a root mean square that grows faster than the square root of time will thus give us mean reversion. The choice of the exponential function was not arbitrary. This specific process was chosen due to its close link to the Vasicek-Hull-White (VHW) model which is a well known mean reverting short rate process. Hunt et al [10] show the similarity between the models as follows. The VHW model has the short rate process, r_t, that solves the SDE

$$dr_t = (\theta_t - a_t r_t)dt + \sigma_t dW_t \tag{124}$$

Solving this for the short rate and using the short rate to calculate the forward Libor rate results in

$$L_t^{(i)} = x_t - \alpha_b^{-1} \tag{125}$$

where

$$\begin{aligned} dx_t &= \left(\frac{e^{-aT_i} - e^{aT_{n+1}}}{a}\right) \sigma e^{at} x_t dW_t \\ &= \text{constant} \times x_t \times e^{at} \end{aligned} \tag{126}$$

W_t is standard Brownian motion under the terminal measure. Note that the terminal time here is T_{n+1}. In the Markov-Functional model we model the Libor rate as a function of our Markov process x_t which has a known law and calibrate by selecting a functional form for the Libor rate. In the VHW model we have the functional form of the Libor rate as well as the law of the driving process. Thus the VHW model is similar to the Markov-Functional model although we do not calibrate by selecting the functional form of the Libor rate. Selecting $\sigma(t) = e^{at}$ in the Libor Markov-Functional model is therefore intuitive. In the implementation, $\sigma(t) = \sigma e^{at}$ was selected although, as with the constant volatility, the constant term had no impact on pricing. This σ is referred to as the *Markov base volatility.*

In this section we had a heuristic explanation of how the volatility process causes mean reversion and how the volatility function was selected. In the next section we provide a formal explanation of how the volatility process causes mean reversion.

10.3 A formal explanation of mean reversion due to the volatility process

This section draws from Hunt et al [10]. Let the implied market cap volatilities be $\bar{\sigma}$ for all the forward Libor rates, $L_0^{(i)}$, and let the initial forward rates be the same. i.e. $L_0^{(i)} = L_0$. Our volatility is defined as $\sigma(t) = \sigma e^{at}$, where a is a known mean reversion parameter[36]. Now suppose that our Markov Process has value $x_{T_1} > x_{T_0}$ at time $T_0 < T_1 < T_n$. This results in L_1 and L_n being larger than L_0. However, L_1 has increased by more as it has an average volatility of $\bar{\sigma}$ over the time period $[0, T_1]$ whereas L_n has that same average volatility over the whole time period resulting in it having a smaller volatility at time T_1. This holds as volatility is increasing exponentially. Now L^n is a martingale under our terminal measure and thus

$$E[L^n_{T_n}|\mathcal{F}_{T_1}] = L^n_{T_1} < L^1_{T_1} \tag{127}$$

This means that when the spot Libor increases from its initial value to a higher value at time T_1, the expected value of the Libor at time T_n is less than the spot rate. We can reverse the argument to show that if the value at time T_1 is lower, then the expected spot rate at T_n is higher.

10.4 A remedy for instantaneous correlation for co-terminal Bermudan swaptions

Single factor models do not have an instantaneous correlation structure. The only instantaneous correlation that exists is between the single factor and itself. This leads to a unit instantaneous correlation between the modelled forward rates for the same tenors i.e. the forward rates are for the same forward period. However, there is a correlation between forward rates for different tenors. Not having a correlation structure for forward rates at different tenors can result in incorrect pricing.

Pietersz and Pelsser [14] provide a remedy for the single factor Markov-Functional model as follows: the theoretical price of a co-terminal[37] Bermudan swaption is fully determined by the joint distribution of the forward co-terminal swap rates at each exercise date. If there are n possible exercise dates we have $n(n+1)/2$ distributions that we need to consider. Clearly a single factor model cannot consider all these distributions. When we take

[36] We find this parameter via calibration which is discussed later.

[37] That is, all swaps expire on exactly the same contractually specified date, regardless of the exercise date

a first order approximation, the price is only determined by the joint distribution of the spot co-terminal swap rates at each exercise date and joint distributions of the rates at different times, other than the terminal time, fall away. These marginal distributions are determined by the quoted European swaption volatility. In the case where we assume log-normality, it turns out that we only need to specify the correlation between these rates. Pietersz and Pelsser [14] call this the *terminal correlation.* Thus if we can fit the Markov-Functional model to the terminal correlation we will have a model that approximately has the correct joint distribution. This is done by ensuring that the terminal correlation of the model's rates is the same as that of the market. The disadvantage of this method is that it is very product specific and requires a co-terminal option.

10.5 Calibrating the Markov-Functional model to terminal correlations by selecting a mean reversion parameter

Consider the swap model and the pricing of a co-terminal Bermudan Swaption. As before, let $y_t^{(i)}$ denote the swap rate at time t for a swap starting at T_i and ending at T_{n+1}. The terminal correlation is the correlation of the swap rates $y_t^{(i)}, y_t^{(j)} \quad i,j = 1,...,n+1$.

Correlations are not affected by linear transformations. Let x and y be two random variables with variances σ_x^2, σ_y^2 respectively and covariance σ_{xy}. Their correlation is

$$\rho_{xy} = \frac{\sigma_{xy}}{\sigma_x \sigma_y} \tag{128}$$

Now transform x and y by

$$\begin{aligned} x' &= a + bx \\ y' &= a + by \end{aligned} \tag{129}$$

These have variances and covariances

$$\begin{aligned} \sigma_{x'}^2 &= b^2 \sigma_x^2 \\ \sigma_{y'}^2 &= b^2 \sigma_y^2 \\ \sigma_{x'y'} &= b^2 \sigma_{xy} \end{aligned} \tag{130}$$

Thus their correlation is

$$\begin{aligned}\rho_{x'y'} &= \frac{\sigma_{x'y'}}{\sigma_{x'}\sigma_{y'}} \\ &= \frac{b\sigma_x . b\sigma_y}{b^2\sigma_{xy}} \\ &= \rho_{xy} \end{aligned} \tag{131}$$

The following draws from the ideas in Pietersz and Pelsser [14]. Our forward swap and Libor rates are functions of the Markov process. A first order Taylor approximation of them yields

$$y_t^{(i)}(t, x+0) \approx y_t^{(i)}(t,0) + [y_t^{(i)}]'(t,0)\cdot x \tag{132}$$

$$L_t^{(i)}(t, x+0) \approx L_t^{(i)}(t,0) + [L_t^{(i)}]'(t,0)\cdot x \tag{133}$$

where the prime denotes the derivative with respect to x i.e.

$$[y_t^{(i)}]'(t,y) = \frac{\partial}{\partial x}_{|x=y}[L_t^{(i)}(t,x)] \tag{134}$$

Any function of these rates yields a similar result

$$f(L_t^{(i)}(t, x+0)) \approx f(L_t^{(i)}(t,0)) + f'(L_t^{(i)}(t,0))[L_t^{(i)}]'(t,0)\cdot x \tag{135}$$

The swap and Libor rates are thus approximately linear transformations of the Markov process. This means that their correlations will be the same as that of our Markov process to the extent that the approximation is accurate. Pietersz and and Pelsser [14] test the accuracy of this by comparing the terminal correlations produced by the model with the correlations of the Markov process. In other words, they compare $\text{corr}(x_{T_i}, x_{T_j})$ with $\text{corr}(y_{T_i}^{(i)}, t_{T_j}^{(j)})$, where $y_{T_k}^k$, $k = i, j$ is the swap rate calculated by the model. This tests how the correlation of the rates at different times on the left hand side of equation (132) compare with the correlation on the right hand side of the equation. Pietersz and Pelsser [14] use exponential volatility for various mean reversion parameters and find that the formula is astoundingly accurate. The largest relative errors between the correlations are smaller than 0.0076%.

If we use exponential volatility, the correlation of the Markov process, and hence the swap rate is

$$\begin{aligned}\rho_{ij} = \text{corr}(y_{T_i}^{(i)}, y_{T_j}^{(j)}) &\approx \text{corr}(x_{T_i}, x_{T_j}) \\ &= \sqrt{\frac{(e^{2aT_i}-1)}{(e^{2aT_j}-1)}} \end{aligned} \tag{136}$$

It is easy to see how this is calculated when one recalls that the correlation between two standard Brownian motions, W_t and W_s, $t < s$, is $\sqrt{\frac{t}{s}}$, the ratio of the two standard deviations. The standard deviation of the Markov process with exponential volatility is given by equation (123). The ratio of these standard deviations gives us equation (136). This allows us to calibrate the mean reversion parameter, a, in the exponential volatility by fitting the terminal correlation of the Markov process with that observed in the market.

The fitting of this can be done by OLS and some optimization routine. An R[38] script was written which uses simulated annealing as well as the conjugate gradient method to find the estimates[39]. The script can be found on the attached CD.

10.6 The swap Market Model's correlations compared to the Markov-Functional Model's correlation

This section draws from Pietersz and Pelsser [14]. If we parameterize the correlation matrix in the swap Market model as

$$\sqrt{\frac{(e^{2aT_i}-1)/T_i}{(e^{2aT_j}-1)/T_j}} \tag{137}$$

we obtain the same terminal correlation as in the Markov-Functional model. Although this result seems trivial, when one looks at the following formulae it becomes clear why this is important. We start with the formula for terminal correlation in the Market model and find that this simplifies to the same formula for terminal correlation in the Markov-Functional model provided we use this specific parameterization i.e.

$$\frac{\int_0^{T_i}\sigma_{i:n}(t)\sigma_{j:n}(t)\rho_{ij}(t)dt}{\sqrt{\int_0^{T_i}\sigma_{i:n}^2(t)dt\int_0^{T_j}\sigma_{j:n}^2(t)dt}} = \frac{\sigma_{i:n}\sigma_{j:n}\rho_{ij}T_i}{\sqrt{\sigma_{i:n}^2T_i\sigma_{j:n}^2T_j}} = \rho_{ij}\sqrt{\frac{T_i}{T_j}} = \sqrt{\frac{e^{2aT_i}-1}{e^{2aT_j}-1}} \tag{138}$$

[38] An open source statistical program

[39] The methods are part of the R program and are documented there.

11 Relation of Markov-Functional models to other models

Here we explain how Markov-Functional models related to the other classes of models such as Heath-Jarrow-Morton (HJM) and Market models (MM).

Hunt and Kennedy [11] mention the following: if we define MM as models that have a driving process that depends only on the current values, then the class of Markov-Functional is larger as we can have models that have more factors than the number of rates being modelled. This is obviously never done as we are trying to keep Markov-Functional models as simple as possible. If we let the coefficients of the driving process in the MM depend on the entire sample path the MM rates will not be Markovian and thus the class of MM will be larger. This class of MM will be larger than the class of HJM models.

A further point discussed by Hunt and Kennedy [11] is the relation between 'standard' MM and Markov-Functional models. 'Standard' MM are those where the forward rates are modelled by log-normal martingales in their associated measures. Note that Markov-Functional models are driven by a process that has a log-normal *distribution* at the respective fixing dates. They can thus be represented as a function of a low (single) dimensional Markov process. 'Standard' MM are driven by a log-normal *process*, which is the rate of interest itself. It turns out that we cannot represent these models as a function of a low-dimensional Markov process because of the assumption of a log-normal *process* in the MM, rather than a log-normal *distribution* on the respective fixing dates, as assumed by the Markov-Functional model. This is proved by a theorem presented in Hunt and Kennedy [11], pp. 366.

HJM models are not necessarily Markovian and thus are a larger class of models than Markov-Functional models. Markovian HJM models can be obtained by enforcing various volatility restrictions on the HJM framework. These models are beyond the scope of this dissertation, but the most advanced and general form, along with further references to less general models can be found in Chiarella and Kwon [6].

Bennet and Kennedy [4] show how one can approximate a separable Libor market model so that the Libor rate is a function of a one-dimensional Markov process. This is discussed further is section 16.2.

12 Pricing formulae

Throughout this section, conditional expectations are Gaussian integrals which are calculated using the respective conditional Gaussian distribution[40]. Further numerical considerations are discussed in section 14.4. From the examples in this section it should be clear how any product is priced in the terminal measure Markov-Functional swap and Libor models.

12.1 Pricing a cap

The payoff of a cap at time T_i, with strike K, is given by

$$\max(L_{T_i}^{(i)} - K, 0) \tag{139}$$

The recursive procedure for calculating the cap price in the Markov-Functional models is

- Calculate the payoff at the terminal time, T_{n+1}.
- Loop over the following
  - At time T_i, calculate

$$\begin{aligned} & D_{T_iT_{i+1}}(x^*)E^{\mathbb{Q}}\left[\frac{V_{T_{i+1}}(x_{T_{i+1}})}{N_{T_{i+1}}(x_{T_{i+1}})}|\mathcal{F}_{T_i}\right] \\ = \ & \text{Inner integral}_{T_i} \cdot E^{\mathbb{Q}}\left[\frac{V_{T_{i+1}}(x_{T_{i+1}})}{N_{T_{i+1}}(x_{T_{i+1}})}|\mathcal{F}_{T_i}\right] \end{aligned} \tag{140}$$

    where $V_{T_{i+1}}(x)$ is the value of the cap at time T_{i+1} in state x and was calculated in the previous iteration. 'Inner integral' refers to the inner integral in the equation for calculating $J(x^*)$, equation (47).

$$\text{Inner integral} = \int_{-\infty}^{\infty} \frac{1}{D_{T_{i+1}T_{n+1}}(u)} \phi_{x_{T_{i+1}}|x_{T_i}=v}(u)du \tag{141}$$

    Do this for each state x^* at time T_i.
  - Add the payoff at time T_i, $\max(L_{T_i}^{(i)-K}, 0)$ to this value. This is V_{T_i}.
- Note that at time T_0 there is only one state to condition on.

[40]The conditional Gaussian distribution is a Normal distribution with the mean and variance determined by the conditioning

12.2 Pricing a barrier cap

The payoff function for a barrier cap, at each time period, can be seen to be

$$\text{payoff} = \begin{cases} \max(L_{T_i}^{(i)-K}, 0) & \text{if lower barrier} \leq L_{T_i}^{(i)} \leq \text{upper barrier} \\ 0 & \text{otherwise} \end{cases}$$

This can be done as we are moving backwards through time and thus do not need to consider whether or not the path followed will break the barrier - we have zero values where it does already.

12.3 Pricing a Bermudan swaption

A Bermudan swaption has the payoff function at each time T_i

$$\begin{aligned} &\max(\text{swap value}_{T_i}, \text{Bermudan swaption value}_{T_i}) \; i = 0, ..., n-1 \\ &\max(\text{swap value}_{T_n}, 0) \end{aligned} \tag{142}$$

The procedure is

- Calculate the terminal payoff.
- Loop over the following
  - Calculate

$$\begin{aligned} & D_{T_i T_{i+1}} E^{\mathbb{Q}} \left[\frac{V_{T_{i+1}}(x_{T_{i+1}})}{N_{T_{i+1}}(x_{T_{i+1}})} | \mathcal{F}_{T_i} \right] \\ = \; & \text{Inner integral} \cdot E^{\mathbb{Q}} \left[\frac{V_{T_{i+1}}(x_{T_{i+1}})}{N_{T_{i+1}}(x_{T_{i+1}})} | \mathcal{F}_{T_i} \right] \end{aligned} \tag{143}$$

  - Add the payoff to obtain V_{T_i}.
- Note that at time T_0 there is only one state to condition on.

The 'Inner integral' refers to the inner integral in equation (57) for calculating $J(x^*)$ in the swap model.

$$\text{Inner Integral} = \int_{-\infty}^{\infty} \frac{P_{T_{i+1}}^{(i)}(u)}{D_{T_{i+1}T_{n+1}}(u)} \phi_{x_{T_{i+1}}|x_{T_i}=v}(u) du \tag{144}$$

13 Actuarial applications

The aim of this section is to outline how one could go about applying the Markov-Functional model and more generally, the principles of stochastic option pricing, to actuarial problems. Put differently, we use Markov-Functional models as an example of how one applies the principles of stochastic option pricing to actuarial problems. The techniques, and most of the comments, remain unchanged if we use other stochastic option pricing models e.g. the Libor Market Model. We will start with an old actuarial problem: trying to match assets to liabilities. Next we will present extensions to the actuarial application of the Libor Markov-Functional model to make it more realistic and discuss the model limitations that we encounter. The partial differential equations (SDE's) of the assets and liabilities will be derived. Finally we will discuss what we can say about the real-world distribution of the surplus. For explanatory purposes we will assume that we use the Libor Markov-Functional model. The formulae and comments are similar under the swap Markov-Functional model which could be used.

13.1 Attempting to match assets and liabilities by viewing the Liabilities as a derivative

Matching means that the value of the assets are sufficient to meet the value of the liabilities under all market conditions. This requires that the value of the assets moves exactly in line with those of the liabilities. If the assets and the liabilities are not exactly the same[41] then this becomes problematic due to the stochastic nature of assets and liabilities. The financial mathematics approach to this problem is to use hedging: the Greeks of the asset and that of the liabilities should be exactly the same leading to the portfolio being insensitive towards changes in market conditions. The Greeks are calculated using a model. This means that hedging will only work to the extent that the model is accurate. In practical applications, hedging helps but results in imperfect protection as we can't trade continuously and there are transaction costs and other market imperfections.

Standard immunization theory is an example of hedging in a non-stochastic environment with a flat yield curve and small jumps in the interest rate. Here we present a stochastic version of this theory in the context of Markov-Functional models which overcomes many of the flaws of standard immunization theory.

[41] e.g. The asset is a stock and the liability is that very same stock

The setup is as follows. We want to ensure that the value of our assets remains greater than that of our liabilities when there are changes in the market conditions. We would also like to know the minimum asset value that we should hold in order to ensure that we can meet our liabilities in all market conditions. Put differently, we want a trading strategy and an associated minimum asset value to ensure that we can meet our liabilities under all market conditions. We start by assuming that there is only one interest rate in the market, the Libor rate. Effectively this means that our zero yield curve is given by the Libor rate. Next we assume that our liabilities are known in advance or that they are a known function of the Libor rate. If the liabilities are fixed and known in advance we can still view then as a function of the Libor rate. We have a Libor Markov-Functional model that is calibrated to caps. All the assumptions of the Libor Markov-Functional model apply here.

This setup leaves us with PDB's which are functions of the forward Libor rate. We can extend this further to have general bonds with known coupons. The Libor rate and liabilities are a function of our Markov process which has a known law under a known measure. We have a portfolio at time zero of

$$\Pi_0 = A_0 - LIAB_0 \tag{145}$$

where A_0 and $LIAB_0$ denote the present value of the assets and the portfolio of the liabilities at time zero respectively. We need to ensure that the value of our portfolio remains positive under changes in the forward Libor rates and volatilities. This means that we need to calculate the derivatives with respect to these factors i.e. calculate the Greeks.

The way that we calculate these Greeks is by viewing the liabilities as a derivative of the Libor rate. The liabilities are a function of the Libor rate and thus can be priced in a similar fashion to the pricing formulae for caps. From these we can calculate the Greeks using the 'bump-and-revalue' method. We can then select the appropriate assets to hedge these liabilities. Note also that we have the correct price to charge for these liabilities as we have the price of our 'liability derivative'. This dynamic hedging does not take transaction costs into account, but these can be added by assuming some hedging cost at each time period. We could show that we can include different hedging costs in different market states.

The above approach considers the liabilities separately, prices them and then attempts to find the correct assets to hedge the liabilities. An alternative

approach is to consider an existing portfolio of assets and liabilities. Again, these will form a new type of derivative which can easily be priced. This approach allows one to examine the value and the risk of an existing portfolio as well as providing information on how to hedge the remaining risk and the cost of doing so.

13.2 Extending the model to be more realistic within its limitations

So far we have only considered liabilities that are a function of the Libor rate (or known in advance). We have not considered assets that pay out based on some other stochastic factor, such as inflation linked bonds. The way in which we extend the model is by adding more factors. This requires us to calculate the functional form of these stochastic market variables.

Typically the problem of exact hedging[42] occurs when liabilities are fixed, short term and thus hedged by bonds. This means that the Markov-Functional model is appropriate. In some cases one may want to extend the model to include equity as well. This can easily be done by including another factor and calibrating to options which should be available in liquid markets. Usually there should be sufficient options on the market index to allow one to calibrate. There is no documented implementation[43] of how the Markov-Functional model prices equity derivatives although there are a few factors that do suggest that it would be an inappropriate model: it is unlikely that equity stocks will be able to be sufficiently explained by a single factor. Equity differs in nature to the *current* Libor rate which remains fixed for periods. Our time grid would need to be quite fine but we would only be able to calibrate quarterly[44] forcing us to use quite a large grid. Thus calibrating to equity may not be appropriate although more research here may be warranted.

We could calibrate to inflation linked bonds by adding another factor and calibrating to options based on these. The problem here would be that there may not be enough available options to which to calibrate.

For stochastic liabilities we could add a factor but that also would require us

[42] As when we use immunization theory.
[43] That has been found by thorough searching.
[44] Or whenever the market traded equity options expire.

to calculate functional form of the liabilities by calibrating to an option that is based on the liabilities. As options based on liabilities are probably not available, an alternative approach would be to find a common factor that the liabilities have with some other option, assume a functional form between the liabilities and that common factor and calibrate to the other option. This will allow one to calculate the functional form of the liabilities.

Note that whenever we add a factor we would need to solve for the drift term in the second driving Markov process as was done in the two-factor model. This approach would allow one to have a stochastic model of the assets and liabilities along with the equations required to ensure that their sensitivity to changes in the market environment do not have any negative consequences. This is quite heuristic and the largest practical problem is that of incomplete markets - there are no options traded on the mortality of individuals or groups[45]. However, it may still be possible to calibrate in some situations where Alternative Risk Transfer (ART), which can involve derivative like contracts, is used.

There is a further limitation which should have been clear from the assumptions: we are assuming that the yield curve that applies to us is given by the forward Libor rate. This assumption can be relaxed. We can calibrate the Libor Markov-Functional model to any yield curve as long as there are discrete fixing dates. If not, we will need to use a discrete time approximation. Note that we need options on the rates of this yield curve at each fixing date in order to calibrate this model. This limitation may be very restrictive as it is unlikely that there will be many options on other rates.

The limitations described above seem to present a bleak picture. This is not so. It must be understood that typical actuarial problems are usually found in incomplete markets leading to the use of assumptions and margins. Using Markov-Functional models allows us to model the stochastic nature of bonds within the context of known liabilities - an improvement on standard immunization theory. In some fortunate cases we can even model other stochastic factors such as the index-linked bond payments. In addition there are still the standard actuarial tools which can be used to conquer the limitations described above. As mentioned above, one could attempt to make assumptions about the functional forms of various factors. This would result in a model that is as accurate as our assumptions. The next step would be to invoke sensitivity and scenario testing. The model calibrates and prices

[45] At least in SAFEX

extremely quickly so it would not be impossible to attach a probability distribution to the liability payments, discretize these and price the derivatives under the different scenarios allowing one to see a probability distribution of the outcomes, select one that has an appropriate liability level and price, as well as hedge, the liability.

Attaching a probability distribution and discretizing the results in order to do the above simulation requires some further explanation. Firstly, we need to know what these probabilities mean. They are the *real world* probabilities and do not form part of the Markov-Functional model. These probabilities need to be obtained from some actuarial model. In essense, we have many Markov-Functional models, each with different liability payments and an associated probability. This means that we *cannot* hedge against these outcomes in the Markov-Functional model.

Discretizing a probability distribution means selecting a finite number (the number of probability discretization points) of outcomes and attaching a probability to each one. Consider the case of independent distributions for the liability payments at each time point. The algorithm to calculate the probability distribution of the price of the liability requires k^p calculations of the conditional expectation for each time step, where k is the number of different possible outcomes and p is the number of time steps from the *terminal* time. Discretizing the probability distributions too finely may result in the model taking too long to run. Assuming that we use k points to discretize our probability distribution at each time period, we have n time periods and R discretization points we will require

$$\sum_{i=0}^{n-1} R \times k^i \tag{146}$$

calculations of the conditional expectation. It may not be the case that we need to use the same number of discretization points for the probability distribution for each time, as at some times there may be no payments or payments will be known for sure. One should consider that the pricing of a standard derivative without any probability discretization points requires $n \times R+1$ calculations of the conditional expectation. Using these figures and the time taken to price a standard derivative, one can work out the maximum number of points that can be used to discretize the probability distribution so that the results can be calculated in a reasonable time frame. Note also that this calculation allows for parallel computing which can make it really quick if enough computers are used. On a Pentium 4, 3.2GHz, hyperthread-

ing processor, calibration, pricing and calculation of Delta and Vega in the derivatives in the examples in section 17 takes about 2.7 seconds when 50 Markov discretization points are used. This amounts to 18 calculations of the price amongst other things such as reading in data from files. If we were to use 5 discretization points for our probability distribution, in addition to our 50 Markov discretization points, in the examples in section 17 we would require about 10.5 hours to calculate the entire distribution. Using 10 distribution points would require over 40 days, so it is impractical. Note that using 5 points and 8 time periods gives us just under 80,000 probabilities at time zero.

The above approach is very much a 'brute force' approach, but even though the Markov-Functional model calibrates and calculates its output quickly, the approach does not allow for a large number of situations to be considered. An alternative approach would be to use Monte Carlo and determine an average, although many iterations may still be required to obtain the variance at the tails. Finally one could attempt to do scenario modelling starting with the set of worst case scenarios and work towards the first acceptable outcome.

Assuming a functional form of the liabilities under the model's measure may not be a wise idea, even if the functional form perfectly describes the distribution of the liabilities. This is because one cannot hedge against the liabilities and thus must take on the risk. Note also that this functional form assumed does not apply under the the real world measure. This makes it nearly impossible to assume the correct functional form.

We could use Ito's formula to obtain a SDE under the model's measure for the liabilities to obtain a further understanding into the behaviour of the portfolio. This is done next.

13.3 The SDE's of our portfolio

Here we consider the SDE's that can be obtained using the Markov-Functional model. First we derive the SDE of the Libor rate and next that of a derivative based on the Libor rate such as a bond or the liabilities. Note that we have a SDE for each time period but the time index has been left out for notational convenience. The SDE's are under the *model* measure and not the real-world measure.

Using Ito's formula on the i^{th} forward Libor rate, $L^i_{T_i} = L$, and the no-

tation $x_{T_i} = x$ we obtain

$$\begin{aligned} L(x) &= L(0) + \int L^{'}(x)dx + \frac{1}{2}\int L^{''}(x)d < x, x > \\ &= L(0) + \int L^{'}(x)dx + \frac{1}{2}\int L^{''}(x)\sigma_t^2 dt \\ &= L(0) + \int L^{'}(x)\sigma_t dW_t + \frac{1}{2}\int L^{''}(x)\sigma_t^2 dt \\ &\Rightarrow \frac{\partial L(x)}{\partial x} = L^{'}(x)\sigma_t dW_t + \frac{1}{2}L^{''}(x)\sigma_t^2 dt \end{aligned} \tag{147}$$

where the prime denotes the partial derivative with respect to x i.e. $L^{'} = \frac{\partial L}{\partial x}$ and the second prime denotes the second partial derivative. The quadratic variation of this is

$$d < L, L >= [L^{''}\sigma_t dW_t]^2 = (L^{''}(x)\sigma_t)^2 dt \tag{148}$$

by Ito isometry. Thus the equation for our derivative portfolio, $D_t = D$, is

$$\begin{aligned} D(L(x)) &= D(L(0)) + \int_{\mathbf{R}} D^{'}(L(x))L^{'}(x)\sigma_t dW_t \\ &\quad + \int_{\mathbf{R}} \frac{1}{2}L^{''}(x)\sigma_t^2 dt + \int_{\mathbf{R}} \frac{1}{2}D^{''}(L(x))(L^{''}(x)\sigma_t)^2 dt \\ &= D(L(0)) + \frac{1}{2}\int_{\mathbf{R}} \left[D^{'}(L(x))L^{'}\sigma_t^2 + D^{''}(L(x))(L^{''}(x)\sigma_t)^2\right] dt \\ &\quad + \int_{\mathbf{R}} D^{'}(L(x))L^{'}(x)\sigma_t dW_t \end{aligned} \tag{149}$$

where the prime on D denotes the partial derivative with respect to $L(x)$ i.e. $D^{'} = \frac{\partial D}{\partial L(x)}$ and the double prime denotes the second partial derivative. Note that the primes on D and L denote different derivatives.

In order to calculate this we require the following quantities

- $D^{'}(L_{T_i}^{(i)}(x_{T_i}))\ \forall\, i \in \mathcal{I}$
- $D^{''}(L_{T_i}^{(i)}(x_{T_i}))\ \forall\, i \in \mathcal{I}$
- $[L_{T_i}^{(i)}(x_{T_i})]^{'}\ \forall\, i \in \mathcal{I}$
- $[L_{T_i}^{(i)}(x_{T_i}]^{''}\ \forall\, i \in \mathcal{I}$

where $\mathcal{I}$ is the set of all i's corresponding to our time periods T_i. In the simple case where our bonds are our assets and hence deterministic functions of the Libor rate we can analytically calculate the partial derivatives of the first two items in the above list. In more complex cases such as when our portfolio includes exotic derivatives, we may need to calculate them numerically. The last two partial derivatives will need to be calculated numerically. Note that we require the functional form of the partial derivatives and thus need to evaluate them at all the discretization points. If we numerically calculate the derivatives for the assets, we should ensure that the jumps used in the Markov process to numerically calculate the partial derivatives for the Libor rate correspond to the jumps in the Libor rate used to calculate the partial derivatives for the assets. This allows for consistent results and emphasizes the fact that the driving process is the Markov process and not the Libor rate.

One can read the volatility of the portfolio directly off the SDE's. This can be done as the volatility does not change under a change in measure and thus we can directly see the real world volatility. This is convenient as volatility is a popular measure of risk.

13.4 The real-world probability distribution of the surplus

The surplus is defined as the difference between the value of the assets and the liabilities. The Markov-Functional model provides us with a probability distribution and thus may provide further insight into the distribution of the surplus. Typically one would like to solve for α, X or A in the following equation

$$\mathbb{P}[(A - LIAB) < X] \leq \alpha \tag{150}$$

where A and $LIAB$ denote the present value of assets and liabilities. $\mathbb{P}$ is the real world probability measure and $\mathbb{Q}$ is the Markov-Functional model measure. Now because $\mathbb{Q}$ and $\mathbb{P}$ are equivalent we should be able to say something about the real world distribution of the surplus $A - LIAB$. The bad news is that we can't say anything about equation (150) directly. We only know that

$$\begin{aligned} \mathbb{Q}(\mathcal{A}) > 0 \quad &\text{iff} \quad \mathbb{P}(\mathcal{A}) > 0 \\ \text{i.e. } \mathbb{Q}(\mathcal{A}) = 0 \quad &\text{iff} \quad \mathbb{P}(\mathcal{A}) = 0 \end{aligned} \tag{151}$$

for all $\mathcal{A}$. This means that we can only determine whether the event can happen or not. This means that we only know the value of the assets required to meet the liabilities almost surely. To solve for a parameter in equation (150) we would need to transform the model measure to the real world measure via a Girsanov transformation which would require one to calculate the state dependent drift. These limitations result in one not being able to say much about the real world probability distribution of the surplus.

On the other hand, we don't really need to know the real world distribution of the surplus. This is explained below. The only reason that we would want to know the real world distribution of the surplus is to determine how risky the portfolio is and to determine an acceptable risk and price level. To the extent that our model is representative of the real world and that our liabilities are correctly included in the model, we can perfectly hedge and price our liabilities. This means that we are only considering the case where $\mathbb{Q}(\mathcal{A}) = 0$. Usually one would attempt to hedge liabilities perfectly and charge the correct amount for them, allowing for a profit margin.

The profit margin is there, theoretically, because of the risk being taken on. The two risks are the *model risk* and the risk that the liabilities are not as assumed (if they are not fixed). Model risk is hard to quantify. If we do not calibrate to the liabilities in the Markov-Functional model and we assume a real world distribution for the liabilities, we can obtain a distribution of the real world surplus as described above (by discretizing the probability distribution). Alternatively, we could obtain the cost of perfectly hedging the liabilities in each scenario and determine an appropriate relation between the cost and the probability of a shortfall.

Furthermore, if our liabilities are fixed and we decide not to hedge them perfectly, we can determine how much risk is inherent in the portfolio by calculating the Greeks and looking at the volatility of the portfolio (which can be read off the SDE).

14 Numerical and implementation considerations

14.1 Numerical procedures

14.1.1 The Normal and cumulative Normal distributions

The Normal and cumulative Normal distribution are a critical element in the implementation of Markov-Functional models. It is thus of utmost importance that the distribution is calculated accurately. The Normal distribution calculations are straightforward. For the cumulative Normal distribution, Hart's method, implemented in West [2], which has precision to 15 significant figures was used. The code was taken from the referring paper, West [16], and modified from Visual Basic to C++ as well as some minor alterations[46].

14.1.2 Fitting a polynomial

Fitting a polynomial can be done many ways, but a common efficient method is Neville's algorithm. Pelsser [13] describes Neville's algorithm. Let $P_{i,\ldots,i+m}$ denote the polynomial defined on the points $x_i, \ldots, x_{i+m}$. The polynomial can be written as

$$P_{i,\ldots,i+m} = \sum_{k=0}^{m} c_{i,k} x^k \tag{152}$$

where $c_{i,k}$ are the coefficients we are calculating. We can obtain higher order polynomials by using the fact that[47]

$$P_{i,\ldots,i+1} = \frac{(x - x_{i+m}) P_{i,\ldots,i+m-1} + (x_i - x) P_{i+1,\ldots,i+m}}{x_i - x_{i+1}} \tag{153}$$

Note that the x_i are the observed values and x is a variable. This can be solved for $c_{i,k}$ quite easily[48] and a recursive algorithm for calculating the coefficients of a polynomial is obtained.

This method of fitting polynomials is quick and easy to run. It can therefore also be used to interpolate between market volatilities rather than using a piecewise linear interpolation.

[46]The code returned zero, instead of one, if $x > 37$. Although this is a very high, we did encounter it when trying different methods of discretizing the Markov Process

[47]This is Neville's Algorithm

[48]This is done in Pelsser [13].

14.2 The procedure for terminal measure models

In theory, one just needs to code the various equations provided. However, there are numerous issues that arise. The figures of the functional forms given in this section depict the general shape of the functional forms. The exact shape will depend on the data but should not be significantly different from the general shape at a first glance.

In order to be able to place checks on the intermediate values that are calculated, one needs to understand the meaning of the intermediate values. For the Libor model the values can be interpreted as follows.

- The inner integral in equation (47) is the amount of a unit carried forward from the current time till the end time, T_{n+1}. The functional form is shown is Figure 6.
- The value of the outer integral in equation (47) can be seen as the payoff for the relevant digital caplet.
- The value of $J(x^*)$, given by equation (47), is the value of the corresponding digital caplet at time zero. This is shown in Figure 7.
- The Libor rate and discount factor are shown in Figures 8 and 9.

Note that in the figures, x(max) and x(min) are our discretization points that correspond to values suitably large and small to be ∞ and $-\infty$ respectively. When we refer to 'a sufficient value is attained', we mean that the value is sufficiently close to the required value and that there are no numerical differences to the results if the value gets any closer to its required value. If the required value is (minus) infinity we have that the Normal and conditional Normal distribution are zero at the 'sufficient' values.
For the swap model the values can be interpreted as follows:

- The inner integral in equation (57) is the PVBP at time T_{i+1} plus a payment of α_i, both carried forward from the current time till the end time T_{n+1}.
- The value of the outer integral in equation (57) can be seen as the payoff for a digital swaption that was discussed there.
- The value of $J(x^*)$, given by equation (57), is the value of the corresponding digital swaption at time zero.
- The par swap rate and discount factor are interpreted as usual.

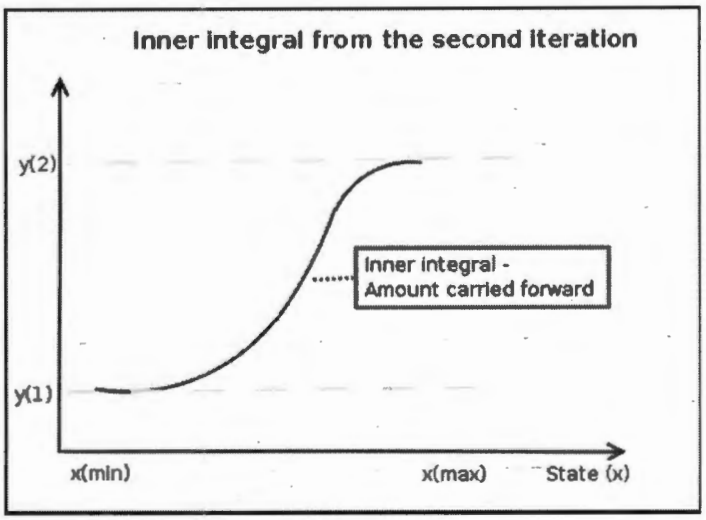

Figure 6: The inner integral as a function of x. Note that the inner integral is a function of the value taken by x_{T_i}. This value is denoted by x in the figure. The asymptotic limit y(1) is 1 and y(2) corresponds to an extremely large value. Theoretically, the graph should branch off to infinity, but due to numerical issues such as inversion problems, it flattens out. That is, y(2) is a value large enough to be infinity for all practical purposes i.e. our Normal and conditional Normal distribution is not noticeably different from zero at the corresponding discretization points.

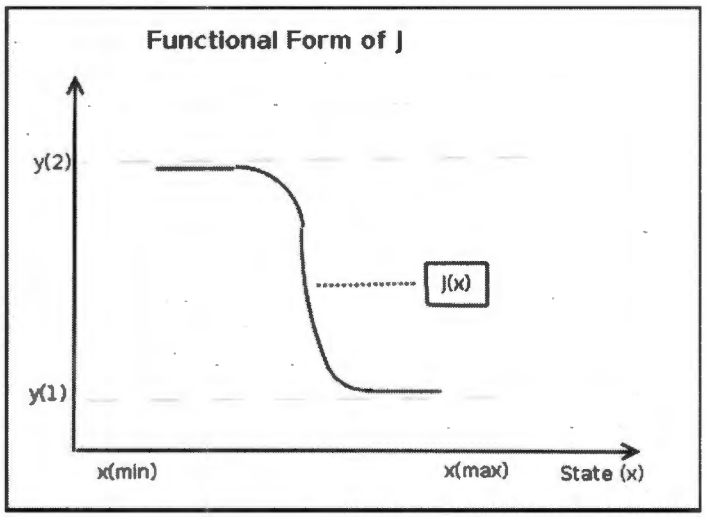

Figure 7: The model price of a digital caplet at time T_0, J, as a function of x. The asymptotic limit y(2) is the discount factor, corresponding to a payoff for sure and y(1) is zero, corresponding to a zero payoff for sure.

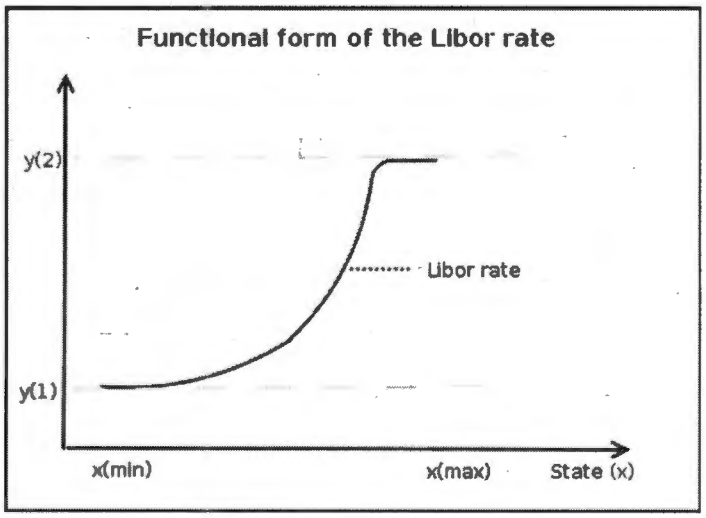

Figure 8: The Libor rate as a function of x. As with the 'inner' function, y(2) represents infinity and y(1) represents zero. The values are not attained due to numerical considerations, but a sufficient value is.

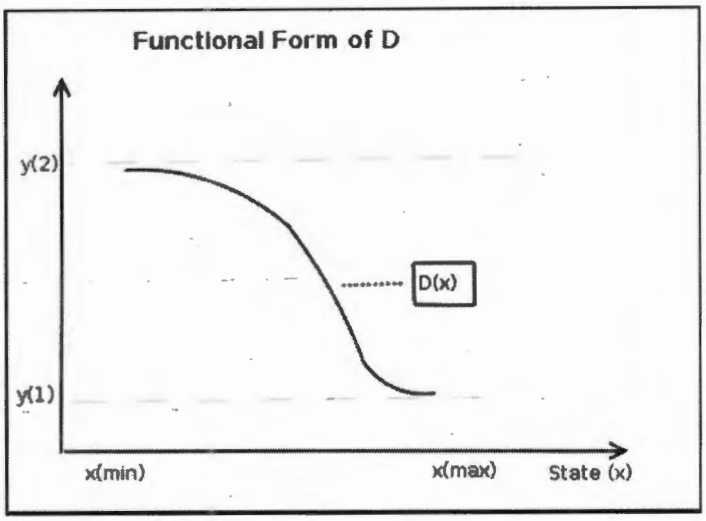

Figure 9: The discount factor as a function of x. The limit y(2) is one due to there being no discounting and the limit y(1) is zero, when there is infinite discounting. Again, these values are not attained numerically, but a sufficient value is.

Here we provide the procedure for the calibration of the Libor model. The graphs of how the values calculated by each step look as functions of x were provided in Figures 6, 7, 8 and 9. The graphs of the swap model's functional forms as well as the procedure are similar and thus not repeated. The most important part of the graphs is the general shape and the asymptotic limit which allows one to place elementary checks on the output.

1. Make all the required arrays
2. Read in the data (Libor rates, Market Volatility)
3. Call a procedure that calibrates the mean reversion parameter to market volatility.
4. Initialize the arrays. In particular, one needs to calculate the PDB prices using the given spot rates and set the initial discount factor to 1.
5. Loop over all time periods
   (a) Calculate the inner integral in the equations for $J(x^*)$ for all discretization points

(b) Calculate $J(x^*)$ for all discretization points. This is a function which decreases from the discount factor to zero.

(c) Use the $J(x^*)$'s to calculate the Libor rate which is an increasing function of the x^*'s.

(d) Use these Libor rates to calculate the discount factor for the next iteration. This is a decreasing function of the x^*'s.

This calibrates the model. Pricing is done in a similar fashion and discussed in section 12.

14.3 Discretizing the Markov process

All our values of interest are functions of the Markov process. We need to evaluate them at suitable discretization values. Note also that we have to integrate over various Normal distributions which have means at each of the discretization points (to calculate the inner integral in step (5a)). At this point is will be useful to recall equations (47) and (57) which calculate $J(x^*)$ for the Libor and swap models. These equations are important when considering the discretization grid. When we refer to the conditional integral (or distribution) we are referring to the inner integral (or the respective Gaussian distribution) in equation (47) or (57). The unconditional integral (or distribution) refers to the outer integral (or respective distribution) in those equations. For simplicity we will assume that the Markov process is defined as $dx = \sigma dW_t$ for the remainder of this section i.e. the Markov process has a constant volatility function.

14.3.1 Using a constant discretization grid

Consider discretizing the Markov process by selecting set values for this process that are constant over time. This grid is depicted in Figure 10. The upper and lower values are set at

$$\pm n \times \sigma \times \text{max number of standard deviations} \tag{154}$$

There are two problems with using this grid. The first is at the endpoints. At each time step we need to integrate the function of the Markov process in the next time period with respect to a Gaussian distribution that has our current state's value as its mean. This is the inner integral in the equation for $J(x^*)$. This is problematic near the endpoints as we do not have up to half of the required values in the next time period over which to define our function which we integrate with respect to the conditional distribution.

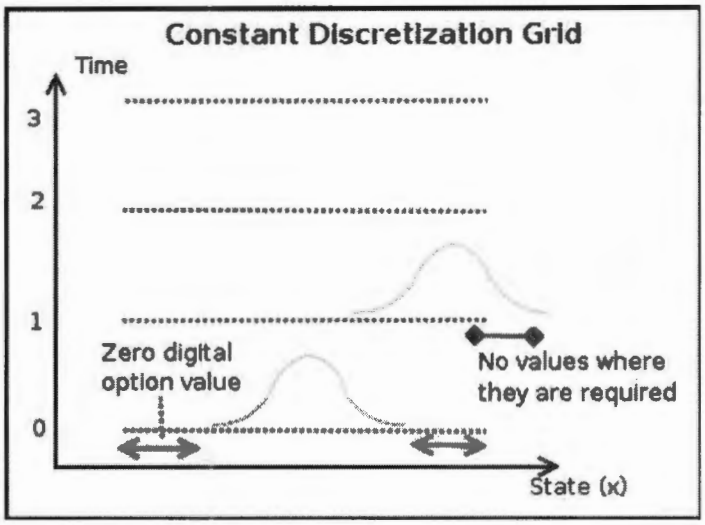

Figure 10: A constant discretization grid. As illustrated by the purple line ending with dots, we do not have values for the conditional distribution at the endpoints as illustrated. In the earlier time periods, we also have digital option payoff of zero and one for sure as illustrated by the red lines with arrows.

This is shown in Figure 10.

A possible solution for this is to fit a polynomial to the last few points and extrapolate for a few standard deviations so that we have the function value for the other half of the distribution. The function is well behaved at the endpoints as these are asymptotic limits resulting in a near constant function. To place checks to ensure correct extrapolation is easy as we know the limits which are discussed in section 14.2. Numerically, this procedure works very well but results in unnecessary extra calculations that can be avoided if we use one of the other methods described below.

The second problem relates to the inability to be able to invert the option prices at the extreme endpoints. As we get closer to the current time the probability of jumping to a far away point (e.g. 15 standard deviations) becomes zero. When we calculate the digital option value conditioned on some point extremely far in or out of the money we get a payment of one or zero almost surely. This is a problem, which we call the inversion problem, and it arises in the other discretization methods so it is discussed in section 14.3.4.

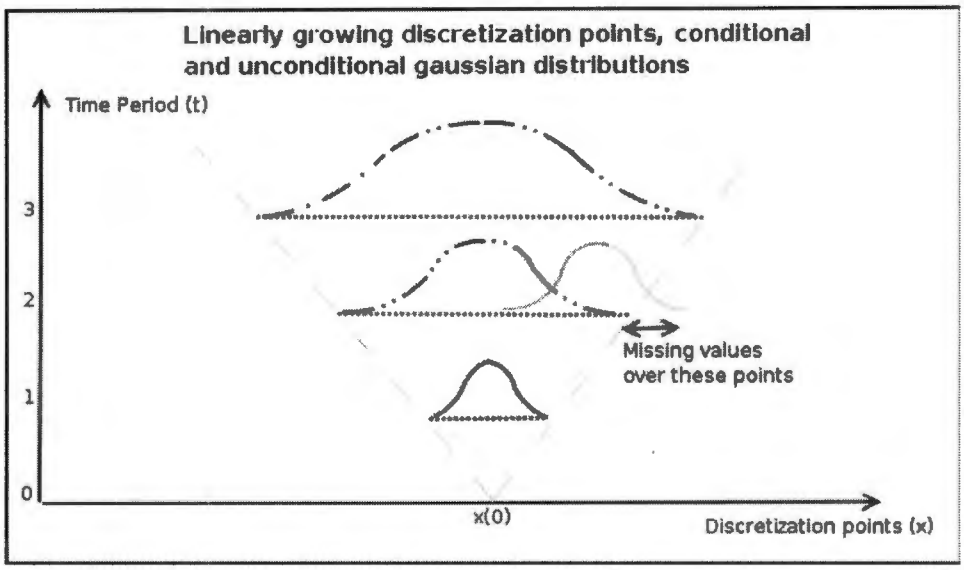

Figure 11: Linearly growing discretization points. The unconditional distribution, denoted by the blue dash-dot line, always fits exactly perfectly on the discretization grid which means that we can ensure that we do not have digital option payoff of one and zero for sure. The conditional distribution, conditioned on the endpoints, denoted by the solid green line, does not have discretization points over which to be defined for half the distribution as shown by the red line with arrows.

14.3.2 Using a linearly growing grid

We could attempt to correct the inversion problem by letting our grid grow linearly. This means that the unconditional distribution will fit exactly on our grid so that we do not have any payoffs of zero and one almost surely for our digital options. This does not fix the problem of not having sufficient values for our conditional distribution which requires more points. This is illustrated in Figure 11.

14.3.3 Using the evolution of the Markov process to find a discretization grid

A more intuitive approach is to look at the evolution of the Markov process itself. The Markov process follows a Brownian motion process with deterministic volatility. Thus it should not jump more that a few standard deviations from it's current point at each time step. Using this information we can make a time dependent discretization grid which has endpoints that grow over time.

We need to calculate, for each volatility specification, what the unconditional volatility at each time period is. Consider the simplest case of constant volatility, $\sigma(t) = \sigma$. The outer integral in the equation[49] for $J(x^*)$ contains a unconditional distribution which has variance $\sigma\sqrt{\text{time}}$. The inner integral contains a conditional distribution with variance σ and a mean of the discretization point that the process is at. We are integrating with respect to the Normal distribution over the payoff in the next time period.

Now consider the process evolving forward over time: at the second time period at the left hand endpoint (of the discretization points) we need values for the third time period that range, say[50], 6 standard deviations left and right of our current value to be able to calculate the conditional distribution accurately. Thus at the third time period, we need discretization points that range from $-2\times6\times\sigma$ to $+2\times6\times\sigma$ to have all the values we require. However, when calculating the unconditional distribution over the third time period (with mean zero) we have values that are "discount" and zero (left and right limits respectively) for all values outside the range of $\pm\sqrt{2}\times6\times\sigma$. This is because outside of this range of value of the cumulative Normal distribution is zero or one respectively[51]. Clearly $\sqrt{i} < i,\ 1 < i$, so we will always get these values for options that expire more than a year away. Note that at later time periods the difference between $\sqrt{i}$ and i is so large that we are many standard deviations, e.g. 40, away from the mean. Here again we get digital options that pay out one or zero for sure as we are extremely far in or out of the money. This is problematic as discussed in the next section.

If we attempt to avoid this problem by shrinking the second time period's discretization points so that we only need $\pm\sqrt{2}\times6\times\sigma$ as our limits we do not have all the points required for the calculation of the values at time period 1.

Figure 12 illustrates the problems discussed.

[49] Either equation (47) or (57).

[50] This number is selected so that our Normal distribution is zero for all numerical purposes after this many standard deviations from the mean.

[51] We have assumed this above by assuming that our Normal distribution is zero after 6 standard deviations from the mean.

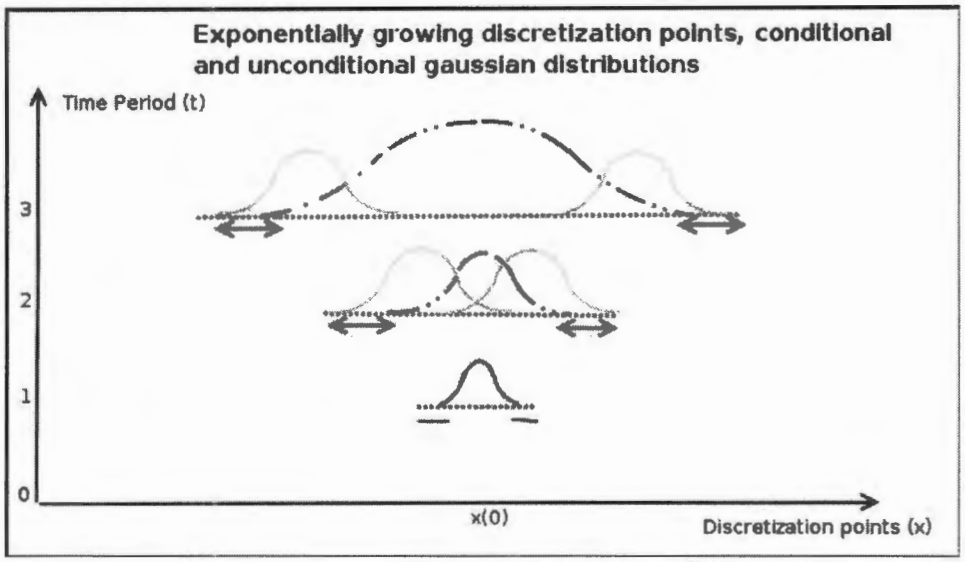

Figure 12: Exponentially growing discretization points. The unconditional distribution, denoted by the dash-dot blue line, has 'excess' points on the grid resulting in digital option payoff of one and zero for sure as shown by the red line with arrows. The conditional distribution, denoted by the solid green line, has exactly the required number of discretization points so that it is defined over its entire range when it is conditioned on the endpoints. i.e. there are exactly enough discretization points so that our conditional distribution fits onto the grid.

14.3.4 Inversion problems

In step (5d) in section 14.2 we calculate the Libor rate or swap rate using, amongst other values, $J(x^*)$ in the equations (50) and (61) which we will call the inversion equations. For a fixed time period $J(x^*)$ can be interpreted as

$$J(x^*) = \text{Discount Factor} \times \text{Digital Option Payoff for strike } K(x^*) \quad (155)$$

In the inversion equations we use the inverse of the standard Normal cumulative distribution of

$$\frac{J(x^*)}{\text{Discount Factor}} = \text{Digital Option Payoff for strike } K(x^*) \quad (156)$$

At the extreme values the digital option's payoff will be zero or one. The inverse of the cumulative Normal distribution at zero and one are undefined.

14.3.5 Solution the inversion problem

A possible way around this problem is, when calculating the value to be inverted, to substitute the next closest value of $J(x)$ that isn't "discount" or "zero" for the problematic value. This results in the first few and last few values of functions of the Markov process at each time period being the same. Note that the substituted values differ at extremely small values e.g. use 10E-15 instead of zero. In the actual calculation later values may be slightly negative (with an error of approximately 10E-15) due to these approximations. These values must be adjusted to be their asymptotic limits. The values that we are adjusting are at the extreme limits and the adjustments are very small. Therefore they should not generate a significantly incorrect price.

14.3.6 Further issues

If we attempt to change the number of discretization points over time we will need to change array sizes at each iteration. There is only one theoretical reason to change the size of the discretization grid with time: at the earlier points, the grid may become overly fine leading to the function having very little variation between consecutive points. Fitting these functions by using a polynomial results in extremely large absolute polynomial coefficients. This issue did not cause any problems in any of the examples discussed in the dissertation, but for some extreme cases it could.

One should also note that we condition on specific x_i's when calculating

$J(x^*)$ and that we need to take care that we condition on the correct x_i. The function is defined on the x's in the next time period and the conditioning is on the x's in the current time period.

14.4 Integration

There are a large number of integrations necessary to implement this model. For pricing purposes, a simple method like Simpson's rule should suffice. Pelsser [13] notes two problems with using this simple rule. Firstly, when we are calculating the Greeks we are taking the derivatives and hence any approximation errors are aggravated leading to unstable hedging parameters, most notably gamma's. Secondly, when we are calculating the integrals in equation (47), near the strike time we have very little variance in our conditional probability distribution resulting in a very spiked distribution. This is likely to be calculated incorrectly if we do not take curvature into account. We also need to ensure that this spike doesn't fall between two points so that we miss it entirely. This is done by selecting a fine enough grid and using the considerations discussed in section 14.3.

Using a method that allows for curvature to be taken into account and ensures that the discretization grid is fine enough will mitigate these problems. This section discusses methods to take the curvature into account and to reduce numerical inaccuracies.

14.4.1 Integrating by fitting a polynomial

The generic algorithm for fitting a polynomial has been discussed in section 14.1.2. First we fit a polynomial to the function in the integral, excluding the Normal distribution part. i.e. we fit a function to $f(x)$ in

$$\int_a^b f(x)\phi_{\mu,\sigma}(x)dx \tag{157}$$

where $\phi_{\mu,\sigma}(x)$ denotes the Normal distribution with mean μ and variance σ. We then use Hunt and Kennedy's algorithm, taken from Pelsser [13], to calculate integrals: we calculate the integrals of the powers of the polynomial with respect to the Normal distribution up to a defined limit, multiply them by the respective coefficients and sum them. The integrals are

$$G(k,h,\mu,\sigma) = \int_{-\infty}^{h} x^k \frac{\exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)\right]}{\sigma\sqrt{2\pi}}dx \tag{158}$$

and can be calculated recursively as

$$G(k) = \mu G(k-1) + (k-1)\sigma^2 G(k-2) - \sigma^2 h^{k-1} \frac{\exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)\right]}{\sigma\sqrt{2\pi}} \quad (159)$$

$$G(0) = \Phi\left(\frac{h-\mu}{\sigma}\right) \text{ and } G(-1) = 0 \quad (160)$$

where Φ denotes the cumulative Normal distribution.

In practice, fitting the function is not straightforward. We have numerous discretization points and thus would need a very high order polynomial if we want to fit it in one step. The function values differ after a few decimal places between each discretization point and thus a low order polynomial fits well, but as the order increases the absolute value of the coefficients become extremely large and tend to alternate in sign. This results in values exceeding the computer's maximum precision capacity and hence serious numerical inaccuracies (e.g. 6E25 being the coefficient of x^{17} results in a big number). These numerical inaccuracies are further aggravated by the fact that we need to use the cumulative Normal distribution which may not be precise enough.

The solution is to split the polynomial up, fitting various segments of the curve, and then to add up the parts. Just splitting up the function into non-overlapping intervals will not be sufficient as we will miss some of the curvature that is implied by previous values. Thus we need to fit to overlapping intervals. Pelsser [13] suggests that one fits a polynomial over a number of points and integrates only over the middle two. For efficiency reasons we rather fit over a larger part of the fitted polynomial, but not the whole polynomial. At the endpoints we need to modify our algorithm to integrate over the whole fitted polynomial as we don't have values to the left and right (for the left and right endpoints respectively) that we use for curvature information. This is not problematic as the functions are very smooth at the endpoints because they approach their asymptotic limits.

Another consideration mentioned by Pelsser [13] is that the function will be discontinous at the payoff. Thus at this point we should split the polynomial into two parts and not have any overlapping intervals.

When using the polynomial fitting to calculate a Gaussian integral we calculate the integral from minus infinity to a set upper limit. However, this is

not what is required when calculating $J(x^*)$ or when calculating the integral between two points[52]. These values are obtained by taking the differences between integrals to obtain the required result.

The infinity we use is specified by the highest value of the discretization points as we don't have any higher values. By selecting a suitably high number of standard deviations at which the Normal distribution is deemed to be zero, e.g. 7 standard deviations, and ensuring that our discretization points grow at a suitable rate, as discussed in section 14.3.3, we are assured that our values representing infinity are suitable.

14.4.2 Integrating by fitting to a parameterised function

Pelsser [13] notes that even though fitting by a polynomial works very well in most cases, there are cases where the propagation of minor numerical errors is significant e.g. a 40 year Bermudan swaption with quarterly payments resulting in 160 time periods. Using a parameterized function mitigates this flaw but introduced the problem that we will not be able to fit market prices exactly. For long term options this shouldn't be a major problem as the market prices at distant tenors are very sparse requiring us to interpolate between market prices. This means that our fit to market prices isn't so important as we are 'guessing' them anyway. Pelsser [13] presents the method to calibrate to the parameterized function

$$\frac{1}{D_{T_0T_{n+1}}(x_t)} = 1 + a_t e^{b_t x_t} + d_t e^{-\frac{1}{2}c_t(x_t - m_t)^2} \tag{161}$$

where a_t, b_t, c_t, d_t and m_t are parameters that need to be fitted for each time period. Empirical runs showed that the method did not fit prices as well as integrating by fitting a polynomial when the volatility was high. He concluded that this parameterization may not work well for volatile currencies such as the Japanese Yen.

14.4.3 Numerical considerations for the spot Markov-Functional model

A major computational drawback is the calculation of the expectation in the spot measure models, as we need to do this iteratively for each time step

[52]We calculated the integral between two points when we integrate by splitting the function into segments.

until we get to time zero. Fries and Rott [7] overcome this by the following method: consider the integral conditioned on $x_{T_i} = x_{T_i,k}$

$$\int_{T_i}^{T_{i+1}} f(\xi)\phi(\xi - x_{T_i,k}, \sigma^{(n)})d\xi \tag{162}$$

When we integrate a function numerically we only sample from the function values at our discretization points. Let F denote the column vector of the function values at our discretization points

$$F = (f_1, ..., f_{m+1})^T \tag{163}$$

The integral can thus be represented as

$$\approx A_{T_i}^{T_{i+1}} F \tag{164}$$

where $A_{T_i}^{T_{i+1}}$ is a $m_i \times m_{i+1}$ matrix and m_i represents the number of discretization points at time T_i. If we now define

$$A_{T_0}^{T_{i+1}} \equiv A_{T_0}^{T_i} . A_{T_i}^{T_{i+1}} \tag{165}$$

as our large time step operator we obtain

$$E^{\mathbb{Q}_0}[f(x_{T_{i+1}})|(x_0, T_0)] = A_{T_0}^{T_{i+1}} F \tag{166}$$

directly. This is quick as $A_{T_0}^{T_{i+1}}$ is a row vector.

The representation of the integral as a linear multiplication is simple to see if one considers how a numerical integration such as Simpson's rule or a Riemann sum would work. The $A_{T_i}^{T_{i+1}}$ matrix is a $m_i \times m_{i+1}$ matrix as each row represents a conditioning on a different $x_{T_i,k}$. Thus multiplying out the equation for the integral gives us a $m_i \times 1$ vector, each element representing the conditional expectation, conditioned on a specific state.

The definition of the large time step operator, equation (166) can be understood as follows: at time T_0 we only have a single state to condition on. It thus makes sense that the $A_{T_0}^{T_{i+1}}$ vector is a row vector. For the first time period, T_0, $A_{T_0}^{T_1}$ will be a row vector. When we are at the second time period, T_1, we want to calculate the expected values of the values at time T_2, at time T_1, conditioned at each point at time T_1. This is done using $A_{T_1}^{T_2}$, an $m_1 \times m_2$ matrix. We then have 'new' values, which are our expected values of the values at time T_2, for each state at time T_1. These values need to be brought

back, through the expectation operator, to time zero. $A_{T_0}^{T_1}$ does that. So we have the function that 'brings back' values from time T_2 as

$$A_{T_0}^{T_1} A_{T_1}^{T_2} = A_{T_0}^{T_2} \tag{167}$$

We then move forwards through time. The function $f(\xi)$ can be seen as $\frac{V(\xi)}{N_{T_{i+1}}(\xi)}$ in the practical application.

Using these precalculated projection vectors is useful, even for the terminal measure model as it prevents numerically inconsistent ways of calculating the large time expectation. The method ensures that the tower law is always valid. This is crucial due to the large use of this law. However, if these projection vectors are slightly incorrect, we will have a propagation of errors. Fries and Rott [7] mention this and say that the terminal distributions will not be perfectly Normal, but that the exact sampling of the terminal distribution is not crucial to the model and that the calibration does not suffer. This may not be entirely true. The next paragraph explains why.

If our terminal distributions[53] are incorrect our terminal correlations will be incorrect. The errors here are likely to be small. Here one should recall the discussion of the terminal correlation in section 10.4. As discussed the terminal correlation is crucial to the correct pricing of co-terminal options[54].

In the terminal-measure models we did not have this problem. Firstly, we made limited use of the tower law meaning that a slight model inconsistency due to numerical errors will not propagate and hence not be too problematic. Secondly, we started with the terminal distribution and worked backwards resulting in our terminal distribution being perfectly accurate.

In order to be able to use the method with the precalculated projection vector we would not be able to integrate using fitted polynomials as we did with the terminal measure. This is because fitted polynomial integration does not allow for an intuitive way of calculating the projection vectors.

There is thus a conflict between integrating using precalculated projection vectors and integration by fitting a polynomial. Pelsser [13] mentions that integrating using simple methods such as Simpson's rule is inaccurate. This was mentioned at the start of this section. Fries and Rott [7] say that their

[53] These are the distributions that each rate has under the terminal measure.
[54] Such as a Bermudan swaption

method is consistent and quick. Fries and Rott [7] did not document an implementation or results of this method whereas Pelsser [13] did. One should thus be wary of this large time step operator.

15 Obtaining relevant market data

Market data such as caplet volatility is not usually directly observable in the market and needs to be derived from prices of other options such as such as caps. Forward par swap rates need to be calculated from forward (Libor) rates. Here we briefly explain how this is done.

Par swap rates are calculated using discount factors which are calculated from the forward rates. We have

$$y_t^n = \frac{1 - D_{t,T_n}}{P_t^n} \tag{168}$$

In order to calculate caplet volatility we require caps expiring at *each* of the caplet expiry dates. For now, we assume that the caps all start at time T_0. Calculating the volatility is then an iterative procedure. The first caplet volatility is just the volatility of the cap expiring on the first tenor because this cap is effectively already a caplet. For the remainder of the caplet volatilities we use the observation that a cap is the sum of caplets and that caps and caplets are priced using the Black formula. We require the volatilities that are inputs into the Black formula. The procedure for calculating the caplet volatilities from the second time period onwards is thus

1. Obtain the market price of a cap expiring at time T_i.
2. Calculate the sum of the caplet prices until time T_{i-1}.
3. 'Add' the formula for the caplet at time T_i to the sum of the caplet prices, equate this to the price obtained in step (1) and solve for the caplet volatility in this formula.
4. Step forward in time.

Note that this procedure assumes that all caps start at time zero. The procedure can easily be adjusted for caps that start later.

Swaption volatility can be obtained directly from European swaptions market.

16 Brief review of empirical comparisons

The results of two papers that compare the Markov-Functional model to other models are presented as well as any issues that arise.

16.1 Empirical comparison of a single factor Markov-Functional and a multi-factor market model

This section summarizes the findings of Pietersz and Pelsser [14]. The authors discuss various comparisons between single-factor and multi-factor models[55] They then compare Bermudan swaption prices to those calculated by other models, and not to market quoted rates. This is done for the following reason: all the models are calibrated to market quoted European swaptions, meaning that they price these perfectly. European swaptions are used to hedge Bermudan swaptions. Thus they can compare how one would hedge in the different models, which is of practical importance, rather than comparing the prices calculated by the Markov-Functional model with those of the market. The authors claim that their results are valid for callable Libor exotics as well.

The models are compared according to their bucket hedging performance. Bucket hedging is when the number of hedge instruments equals the number of instruments in the market to which the model was calibrated[56]. The results for the Markov-Functional model, SMM and LMM are very similar. As one would expect, Delta hedging significantly reduces profit and loss, which is further significantly reduced by further Vega hedging.

16.2 Empirical comparison of single factor separable market model (SLM), Markov-Functional and the approximated SLM

Bennet and Kennedy [4] conduct an analysis using Libor models. They describe the approximate SLM in order to show how we can obtain an approximate market model that has a Libor rate that is a function of a one-dimensional Markov process.

The approximate SLM uses a drift approximation derived from a single time step discretization of the forward rates. The approximate drift is calculated

[55]This will not be repeated here. See the cited paper [14] for details.

[56]The reason is that bucket hedging outperforms factor hedging for certain products and that is the method used in practice.

by replacing the $L_s^{(j)}$ terms in the integrated drift term of the i^{th} Libor rate at time t by the mean of a generalized geometric Brownian bridge, starting from $L_{T_0}^{(j)}$, with the same volatility as the Libor rate in the current period, conditioned on $\hat{L}_t^{(j)}$ at time t. $\hat{L}_t^{(j)}$, $j > i$, has been calculated from previous iterations. This approximation allows one to view the Libor rate as a function of a one-dimensional Markov process as the definition of a separable volatility structure allows one to write

$$\int_0^t \sigma_s^{(i)} dW_s = \int_0^t \gamma^i \sigma_s dW_s \tag{169}$$

$$= \gamma^i x_t \tag{170}$$

where

$$x_t = \int_0^t \sigma_s dW_s \tag{171}$$

Further details can be found in the cited paper.

Bennet and Kennedy [4] find that under stress testing the approximation to the SLM model allows for significant arbitrage while the Markov-Functional and SLM model provide similar results. For the majority of normal market conditions the one-factor SLM and the Markov-Functional model provide results that are approximately the same[57]. A better fit is achieved with shorter tenors.

[57] for practical purposes

17 Numerical results and findings

The prices and Greeks of the standard options are well documented in the literature. Here we investigate the sensitivity of the results to assumed numerical parameters such as the perturbation interval used in calculating the Greeks and the number of discretization points used. We also investigate the impact of the mean reversion parameter on the price and Greeks. For comparison purposes we use the same assumed market data, strike rates and numerical parameters for all the derivatives.

We assume that we have a constant Libor rate of 5% and a constant volatility of 20%.

The Greeks are calculated using the so called *bump-and-revalue* technique. As we will see, many anomalies arise. These are explained in the sections where they arise.

Where graphs are presented, the data used to generate the graph is included on the CD. The Greeks of interest are Delta and Vega as Pietersz and Pelsser [14] mention that hedging with these two reduces the profit and loss to an insignificant level[58]. The choice of the perturbation intervals are not obvious. Pietersz and Pelsser [14] use 0.01 for calculating Delta and 0.05 for calculating Vega. These are the default parameters that we choose. In addition we calculate Vega and Delta for all the perturbation intervals ranging from 0.0001 to 0.1.

Most of the results are presented in graphs which can be found in appendix A. Please take careful note of the *scale of the graph* which changes according to the variation in the graphs. Some seemingly large jumps are in fact extremely small.

[58]The profit and loss resulting from hedging with these two parameters was very close to zero.

Unless otherwise mentioned, the following numerical parameters were used.

Numerical Parameters

Discretization Points	150
Max Std Deviations for Normal Distribution	7
Polynomial Order	6
Pertubation interval for calculating Delta	0.01
Pertubation interval for calculating Vega	0.05
Markov process base volatility	0

17.1 The Libor Markov-Functional model

17.1.1 A cap

A cap with the following details is priced.

Cap Details	
Strike rate	5%
Period	2 years
Payable	In advance
Payment intervals	Quarterly
Markov volatility	20%
Mean reversion parameter	0

Using the standard numerical parameters the following price and Greeks were obtained.

Cap

Price	0.0104883			
Time	0	0.25	0.5	0.75
Delta	0.247417	0.175371	0.159415	0.15401
Vega	0	0.00270393	0.00451019	0.00593601

Time	1	1.25	1.5	1.75
Delta	0.151612	0.149767	0.149891	0.149606
Vega	0.00712	0.0081421	0.00904845	0.00986851

The functional form of the Libor rates is given in Figure 21. As one can see the graphs follow the expected shape and are increasing functions of the

Markov process. The features of these graphs have been discussed in section 14.2.

The graphs for Delta and Vega for different perturbation intervals are presented in Figures 13 and 14. The graphs are smooth and increasing. From the graphs it is clear that there is a second order effect with respect to the Libor rate, which is near constant at earlier durations and slightly increasing at later durations. In this example, the largest change in delta, due to changing the perturbation interval from 0.0001 to 0.1 is 0.08326 (this is for the first rate, $L_{0.25}^{(0.25)}$). This is consistent with Pietersz and Pelsser [14] who have already concluded that Delta and Vega hedging are sufficient. Vega hedging has a near constant, near zero second order effect.

The exact number of discretization points is not clear from a theoretical perspective, although there are theoretical restrictions on how the endpoints should grow with time. We thus calculated the price and the Greeks for different discretization points, starting at 20 and ending at 400. This is shown in Figures 15, 16 and 17. From these graphs it is clear that the results stabilise after 150 discretization points. The graphs for Delta seem bumpy but when one looks at the scale the graphs are very flat.

As we move forward in time we require more discretization points in order to obtain accurate results. Theoretically this can be explained by the increase in the limits on which the Markov process is defined. These limits increase exponentially as explained in section 14.3.3, so our discretization points rapidly become sparse. Furthermore one should note the relation between the number of discretization points used and the maximum number of standard deviations used. The more standard deviations used, the more the limits grow causing the points to be even more sparse. This is not clear from the graphs but it can be seen when one runs the code and carefully inspects the verbose output[59]. One can conclude that the longer the option is, the more discretization points are necessary. However, there is a numerical problem with using too many discretization points in the earlier durations which is not clear here but is clear when one runs the code. The verbose output mentions that the absolute value of the coefficients of the polynomials used in the integration get very large. The reason for this is that we are integrating using a polynomial of degree six. The discretization points are very close and the graph is nearly flat. This results in polynomial coefficients that are extremely large and then extremely small, just as we had when we tried to

[59]This needs to be enabled in the code

fit a polynomial to all the points. These numbers could get too large, so that when we use them with (multiply them by) the cumulative Normal distribution we have implemented, which has precision to only 15 decimal places, we obtain numerical inaccuracies. Using a polynomial of a lower order does not provide us with enough curvature information leading to incorrect values.

In this example the coefficients did not grow large enough to create problems. For other options these large coefficients could pose a problem. Where it is problematic an easy fix would be to have a different number of discretization points which would grow exponentially, in line with the growth of the limits of the Markov process, at each time period. Note also that one should select the lowest number or discretization points that results in stable answers to reduce the time taken to calculate the output. When we used more than double the required number of discretization points our coefficients did not grow large enough to create any numerical errors. One could conclude that this is a theoretical problem that is unlikely to emerge in practical applications.

To examine the effect of mean reversion the price and the Greeks were calculated for different mean reversion parameters, ranging from zero to 20%. The results are presented in Figures 18, 19 and 20. The results are intuitive. A higher mean reversion parameter leads to a lower cap price due to the strike value being reached less often. Note that the Delta for the first two rates is an increasing function of the mean reversion parameter while it is a decreasing function for the remainder of the rates. The reason for this is that a higher mean reversion parameter leads to a lower volatility in the first few periods and a higher volatility in later periods. This leads to the price being negatively sensitive to an increase in the mean reversion parameter for the earlier rates as the volatility is lower, and positively sensitive for later rates due to the higher volatility. As the mean reversion parameter increases, jumps in volatility become less significant as there is a greater tendency to revert to the mean. This can be seen by the decreasing graphs for Vega.

17.1.2 A barrier cap

The details of the barrier cap are the same as the cap with a lower barrier at zero and an upper barrier at 7%. The code can calculate the value for any lower (and upper) barrier, but a lower barrier of zero makes sense as well as leaving only a single source of discontinuity.

The results using the standard numerical parameters are given in the fol-

lowing table.

Barrier Cap

Price	0.00341			
Time	0	0.25	0.5	0.75
Delta	0.24916	0.072	-0.0084	-0.0188
Vega	0	0.00019	-0.0035	-0.0024

Time	1	1.25	1.5	1.75
Delta	-0.0161	0.01308	0.00269	0.0243
Vega	-0.0038	-0.0039	0.00013	-0.0003

The functional form of the Libor rates will be the same as the Cap as we are calibrating to the same market data.

The graphs of the Delta and Vega for different perturbation intervals are presented in Figures 22 and 23 respectively. The graphs for Delta drop off steeply at the earlier tenors and less steeply for later tenors. They are reasonably smooth[60] after the steep drop due to the barrier. The graphs for Vega have very smooth behaviour with a nearly constant, near zero, second order effect.

The graphs for different numbers of discretization points are presented in Figures 24, 25 and 26. The values quickly converge at approximately 100 discretization points although there is some small noise in an extremely tight band.

The graphs for different mean reversion parameters are given in Figure 27, 28 and 29. There is a clear break due to the barrier but otherwise the graphs are smooth. The graph for the price drops off like the cap but then breaks away upwards as high mean reversion parameter causes the barrier to be breached much less often. The behaviour of the Greeks can be explained in a similar fashion. Once again, note the scale of the graphs as some of the jumps that look large are in fact extremely small.

[60]Once again, examine the scale of the graphs

17.1.3 Pricing a Bermudan swaption in the Libor Markov-Functional model

Earlier we mentioned that the swap Markov-Functional model should be used for pricing swap based products only and that the Libor Markov-Functional model should be used to price cap based products only. A swap is not a cap based product. Here we calibrate a model to caps and then price a Bermudan swaption in order to determine the problems with pricing a Bermudan swaption, and more generally, swap based products, in the Libor Markov-Functional model.

In order to calculate the value of a swap we require the discount rates

$$D_{T_iT_j}(x_{T_i}) \ , 0 \leq T_i \leq T_j \leq T_{n+1} \tag{172}$$

Note that the rate is conditioned on the state at time T_i corresponding the starting date of the bond. In the Libor Markov-Functional model we have the rates $D_{T_iT_{n+1}}(x_{T_i})$. We cannot use the relationship

$$D_{T_iT_j} = \frac{D_{T_iT_{n+1}}}{D_{T_jT_{n+1}}} \tag{173}$$

directly as we do not have $D_{T_jT_{n+1}}(x_{T_i})$, but only $D_{T_jT_{n+1}}(x_{T_j})$. This is because, during calibration, we only calculate PDB prices as a function of the Markov process at the starting time of the bond, and not as a function of the Markov process at a time before the start of the bond. The required PDB's are obtained by using the tower law iteratively. For $i < j$,

$$\begin{aligned}
& \frac{1}{D_{T_iT_j}(\xi)} \\
= \ & \frac{1}{N_{T_i}(\xi)E^{\mathbf{Q}}\left(\frac{1}{N_{T_j}(x)}|\mathcal{F}_{T_i}\right)} \\
= \ & \frac{1}{D_{T_i}(\xi)E^{\mathbf{Q}}\left(\frac{1}{D_{T_j}(x)}|\mathcal{F}_{T_i}\right)} \\
= \ & \frac{1}{D_{T_i}(\xi)E^{\mathbf{Q}}\left(E^{\mathbf{Q}}\left(\frac{1}{D_{T_j}(x)}|\mathcal{F}_{T_{i+1}}\right)|\mathcal{F}_{T_i}\right)} \\
= \ & \ldots \\
= \ & \frac{1}{D_{T_i}(\xi)E^{\mathbf{Q}}\left(E^{\mathbf{Q}}\left(\ldots E^{\mathbf{Q}}\left(\frac{1}{D_{T_j}(x)}|\mathcal{F}_{T_{j-1}}\right)|\ldots\right)|\mathcal{F}_{T_i}\right)}
\end{aligned} \tag{174}$$

These rates enable us to calculate the functional form of the required PVBP's and hence the required swap rates and derivative prices. The swap model did not require this as we modelled the swap rate directly. From a computational perspective, this adds many extra calculations and hence increases the time taken to calculate the required output, showing that it is probably not a good idea to attempt to price swaps using the Libor model. Furthermore we will have problems calibrating to terminal correlation as demonstrated later.

The results are stable showing that it is still possible to correctly price a swap in the Libor model. The functional form of the swap rates are drawn in Figure 47. The rates are still monotonic increasing functions of the Markov process and do not differ much from those in the swap Markov-Functional model.

A Bermudan swaption with the same details as in section 17.2.1 is priced. Note that the price does not correspond to that of the Bermudan swaption priced below as the volatility used (20%) is the *caplet* volatility and not the digital swaption volatility. Differences between these results and those of section 17.2.1 are expected as the functional form in the two models is different as well as the models having slightly different market data.

In this numerical example, the Normal distribution was deemed to be zero after nine standard deviations from the mean. The results calculated using the standard numerical parameters are given in the following table.

Bermudan Swaption in the Libor Model

Price	0.00993			
Time	0	0.25	0.5	0.75
Delta	-0.0024	-0.0225	-0.0534	-0.0759
Vega	0	0.00207	0.00486	0.00674

Time	1	1.25	1.5	1.75
Delta	-0.0923	-0.1045	-0.1126	-0.1205
Vega	0.00804	0.00938	0.01005	0.01076

The graphs for the price, Delta and Vega for changes in the number of discretization points are given in Figures 41, 42 and 43 respectively. The price converges quickly after about 80 discretization points and the Greeks stabilize after 150 discretization points and become near constant after 200 discretiza-

tion points. There is very little noise in the results after 200 discretization points.

The graphs for the price, Delta and Vega for changes in the mean reversion parameter are presented in Figures 44, 45 and 46 respectively. This clearly shows a flaw in using the Libor Markov-Functional model to price Bermudan swaptions. The graph for the price *decreases* with an increase in the mean reversion parameter instead of increasing as it did in the swap model. Now recall that the mean reversion parameter was a parameter on the Markov process. The terminal correlation approximation was done by taking a first order approximation of the logs of the swap rate. Here the swap rate is a function of the all the forward Libor rates from the current time till the terminal time and hence the Markov process at all the forward times. The mean reversion approximation held when the rate of interest was a function of the Markov process in *only* the current state.

The effect of the perturbation interval for the Delta and Vega are shown in Figures 42 and 43 respectively. The graphs for Delta are smooth and increasing showing a relatively small but possibly significant second order effect. The size of the second order effect is similar to that of the Bermudan swaption priced in the swap model. The slopes of the graphs for Vega change sign in places. This behaviour can be explained by the fact that here, Vega is defined as the sensitivity to the caplet volatility and not the swaption volatility which is a function of all the forward rate volatilities until the terminal time.

17.2 The swap Markov-Functional model

Here we price a Bermudan swaption using the swap Markov-Functional model. Note that the swap rates are calculated from the given market Libor rates. Delta is calculated with respect to changes in the Libor rate and not with respect to changes in the swap rate. This is done for comparison purposes with the Libor Markov-Functional model. The market data assumed is the same as that used in the Libor Markov-Functional model. As with the swap priced above, the Normal distribution was deemed to be zero after nine standard deviations from the mean. All the other numerical parameters are the standard numerical parameters.

17.2.1 A Bermudan Swaption

A Bermudan Swaption with the following details is priced.

Bermudan Swaption Details

Strike rate	5%
Termination of option	2 years from start date
Swap start and end dates	From exercise till two year from time zero
Swap is payable	In advance
Callable dates	Quarterly
Markov volatility	20%
Mean reversion parameter	10%
Position	Receive fixed

The results obtained by using the standard numerical parameters are given in the following table.

Bermudan Swaption

Price	0.00453			
Time	0	0.25	0.5	0.75
Delta	-0.1465	-0.0956	-0.047	-0.0235
Vega	0	0.00562	0.01349	0.01111

Time	1	1.25	1.5	1.75
Delta	-0.0118	-0.007	-0.0045	0
Vega	0.00516	0.00226	0.0011	0

The functional form of the swap rates is given in Figure 38. As one can see the swap rates follow the expected functional form and are increasing functions of the Markov Process. The features of these graphs have been discussed in section 14.2.

The Delta and Vega for different perturbation intervals are presented in Figures 30 and 31 respectively. The behaviour of Delta at the earlier durations is smooth and has a continuous second order effect. The second order effect is not that small and thus gamma hedging may be appropriate where Delta and Vega hedging fail to reduce profit and loss to an insignificant level. The graphs for Vega are smooth and have a very small second order effect.

The Greeks and prices for different discretization points are presented in Figures 32, 33 and 34. The price converges quickly after about 80 discretization points but the Greeks require 150-200 discretization points to be calculated accurately. The results converge to a near constant value with deviations due to numerical values being extremely small.

The Greeks and prices for different mean reversion parameters are presented in Figures 35, 36 and 37. Here we have predictable smooth behaviour. When interpreting the graphs for Delta and Vega recall that Delta is defined as the derivative with respect to the Libor rate from which the swap rate is calculated and not the swap rate directly. The graph for Vega is decreasing for the first four time periods and then increasing for the fifth and sixth time periods. This can be explained by the fact that the volatility in the earlier durations is exponentially smaller than that in later durations. This causes volatility to increase (with the mean reversion parameter) at later durations and decrease at earlier durations.

18 The code

This section describes the code used to implement the model. Parts of it may require knowledge of computer programming, most notably Object Oriented (OO) programming. The code is written in C++ using an OO approach. This was done to ensure that the code runs quickly and to allow easy extension of the code in order to price other derivatives and use other Markov-Functional models. It is designed in a library type of format so that one can easily price any new derivative with the code. We explain how to do this in section 18.2. The code is too large to print out[61] and attach as an appendix so only the header files are attached. These can be found in Appendix B. The design of the code is discussed below. The entire code can be found on the attached CD. Note that the code is designed to run on a Linux system. There are very few system specific calls and these are all put in a separate object that will need to be changed if the code is ported. The GCC compiler[62] was used along with only the standard C++ libraries. This allows easy porting of the code across platforms. The code has checks all along so if one is pricing something very exotic and there is a numerical or logical error that it cannot correct, an error message is given. Generally the code manages to automatically correct most numerical errors. The code runs quickly and can calibrate, calculate the prices and the Greeks and write all output in about 2.7 seconds when 50 discretization points are used in the example in section 17.1.1. The code was run thousands of times and there was no sign of a memory leak[63]

18.1 Design of the code

Here we will describe how the code is designed and the objects that are available. It can be split as follows:

1. Functions e.g. Cumulative Normal, Integration, Driver functions.
2. Objects that are used as parameters e.g. file object, volatility object.
3. The Markov-Functional Object that contains a generic framework and functions relating to all one-dimensional terminal measure Markov-Functional Models.
4. Objects relating to the specific models: Libor and swap models.

[61] Over 100 pages

[62] Probably the most standards compliant compiler.

[63] The system memory was monitored closely. A memory leak should lead to an accumulation of memory used, which could be very small, as more runs are done

5. Objects relating to specific products.

The list is in the order that objects *inherit* from each other. All the objects have functions that simplify the use of the object e.g. calculate the conditional expectation with the arguments current state, current time and the payoff in the next time period.

The code is well documented so that each step can be understood by anyone wanting to understand the finer details. Normally, all that a programmer requires are the header files as these provide a list of functions that the object can execute. The functions and parameters are labelled so that one can understand how to use the code by reading only the header files.

There are a large number of places in the code where one can change a *boolean* parameter to make the output more verbose. In addition to screen output, the code writes a large number of files with intermediate values such as the functional forms calculated. All output is written in comma separated values format so that the files can be read in a spreadsheet or imported to a large variety of programs. We use R[64] to quickly generate graphs of the output. The scripts that automatically generate the graphs and write them to postscript files can be found on the attached CD.

The code makes use of a large number of pointers. One must thus be wary when altering the code as a *memory leak* can easily be created.

All market data is read in from the data files. [65] One should refer to the header files for more details of the exact functions and how the code is designed.

18.2 How to price new derivatives

In this section we will outline how to price new derivatives using this code. For further details one just needs to look at the code for pricing a barrier cap or Bermudan Swaption.

One should select the appropriate Markov-Functional model to use i.e. swap or Libor model. Create an object that inherits from the selected model ob-

[64] A statistical program

[65] The location of these files is specified in the code.

ject. The constructor function must be fully parameterized[66] as there is no option to change all the parameters later. This is because the object sets up the array sizes according to the parameters. Select the parameters required for your option e.g. strike rate, expiry, exercise times and add these to your object. One only needs three functions in the object: price, calculate_delta and calculate_Vega.

The procedure to calculate the price is

1. Calibrate the model by executing the *calibrate()* function.
2. Create the required variables which are
   (a) A pointer to a payoff array with *num_discretization_points* in it.
   (b) An array to temporarily store the next payoff function.
   (c) Integers *state* and *time*
3. Loop over all states and calculate the terminal payoff
4. Loop over all time periods, starting at the penultimate period and ending at time zero.
   (a) Loop over all states
      i. Calculate the conditional expectation of the payoff (in the next time period) using the *calculate_conditional_expectation(time,state,*payoff)* function. This returns a value that must be stored.
      ii. Multiply the value by the discount factor in this state (D[time][state]) in order to get the numeraire rebased value at the current time.
      iii. Add and change any values relating to the option e.g. set the value to zero if it broke the barrier or max(value,0). If there is no optionality at this time period then this value remains unchanged
      iv. Store this value in the temporary array.
   (b) Set the payoff to be the values that are stored in the temporary array (requires another loop).
5. Return the price which is the value of any element of the payoff array (all the values should be the same now as there is only one state).

[66] i.e. all the inputs, such as the mean reversion parameter, need to be inputs into the constructor function.

To calculate the Delta one uses the following procedure

1. Price the model and store this price
2. Loop over all time periods
   (a) Increase the Libor rate *L[0][time][0]* by the perturbation interval.
   (b) Price the model (this automatically calibrates) and store this price.
   (c) Calculate Delta and store it.
   (d) Change the Libor rate back.
3. Return Delta or print Delta to a file.

Vega is calculated in a similar way but one changes the volatility instead. This done by using the function to set the market volatility.

There are a few further items not mentioned here which are implemented in the code in order to increase efficiency, such as ensuring that we only calibrate when needed and don't calibrate to the same data twice.

19 Conclusion

We introduced Markov-Functional models and their framework, went through the specific details on how the models work, considered some practical applications, described in detail how the models were to be implemented, provided a literature review of the results of the models, implemented the models and checked their behaviour with various market and numerical parameters. This provides the reader with a discussion of the current class of Markov-Functional models.

The next stage in developing Markov-Functional models would be to test models that have many factors (more than three) and to test the performance of hybrid models, most notably models that include a factor to model equity. All the tools required to do this have been developed in this dissertation, which just leaves the issue of testing the model that fits one's specific needs. For terminal measure models, this is very quick and easy to do as the code is designed so that one can easily alter it to meet specific needs.

References

[1] An Introduction to R. Web Page. http://cran.r-project.org/doc/manuals/R-intro.html.

[2] *Computer Approximations.* Wiley, 1968. Algorithm 5666 for the error function. Cited in West [16].

[3] P.J. Acklam. An algorithm for computing the inverse normal cumulative distribution function. Web Page. http://home.online.no/ pjacklam/notes/invnorm/.

[4] M.N. Bennet and J.E. Kennedy. Common interests. Technical report, University of Warwick, 2004.

[5] T. Bjork. *Arbitrage Theory in Continuous Time.* Oxford University Press, second edition, 2004.

[6] C. Chiarella and O.K. Kwon. State variables and the affine nature of markovian hjm term structure models. Technical report, 2001.

[7] P. Fries and M. Rott. Cross-currency and hybrid markov-functional models. Technical report, Dresdner Bank, 2004.

[8] H. Geman, N. El Karoui, and J. Rochet. Change of numeraire, changes of probabilty measures and pricing of options. *Journal of Applied Probability*, 32(pp. 443-458), 1995.

[9] J.M. Harrison and S.R. Pliska. Martingales and stochastic integrals in the theory of continuous trading. *Stochastic Processes and Their Applications*, 11(pp. 215-260), 1981.

[10] A. Hunt, P. Kennedy, and J. Pelsser. Markov-functional interest rate models. *Finance and Stochastics*, (pp.391-408), 2000.

[11] K. Hunt and J. Kennedy. *Financial Theory and Derivatives in Practice.* John Wiley and Sons, Ltd., 2004.

[12] J. Liberty. *Teach Yourself C++ in 24 Hours.* Sams publishing, 1999.

[13] A. Pelsser. *Efficient Methods for Valuing Interest Rate Derivatives.* Springer, 2000.

[14] R. Pietersz and A.J. Pelsser. A comparison of single factor markov-functional and multi factor market models. Technical report, Erasmus University Rotterdam, 2005.

[15] M Rubinstein. Displaced diffusion option pricing. *Journal of Finance*, 38(pp.213-217), 1983.

[16] G West. Better approximations to cumulative normal functions. Technical report, University of Witwatersrand, http://www.cam.wits.ac.za/mfinance/papers/accuratecumnorm.pdf.

A Figures

The figures that follow refer to rate 0,1,...,7 on the y-axis. These numbers refer to the index in the tenors i.e. they refer to the i in T_i. The tenors are at quarterly intervals.

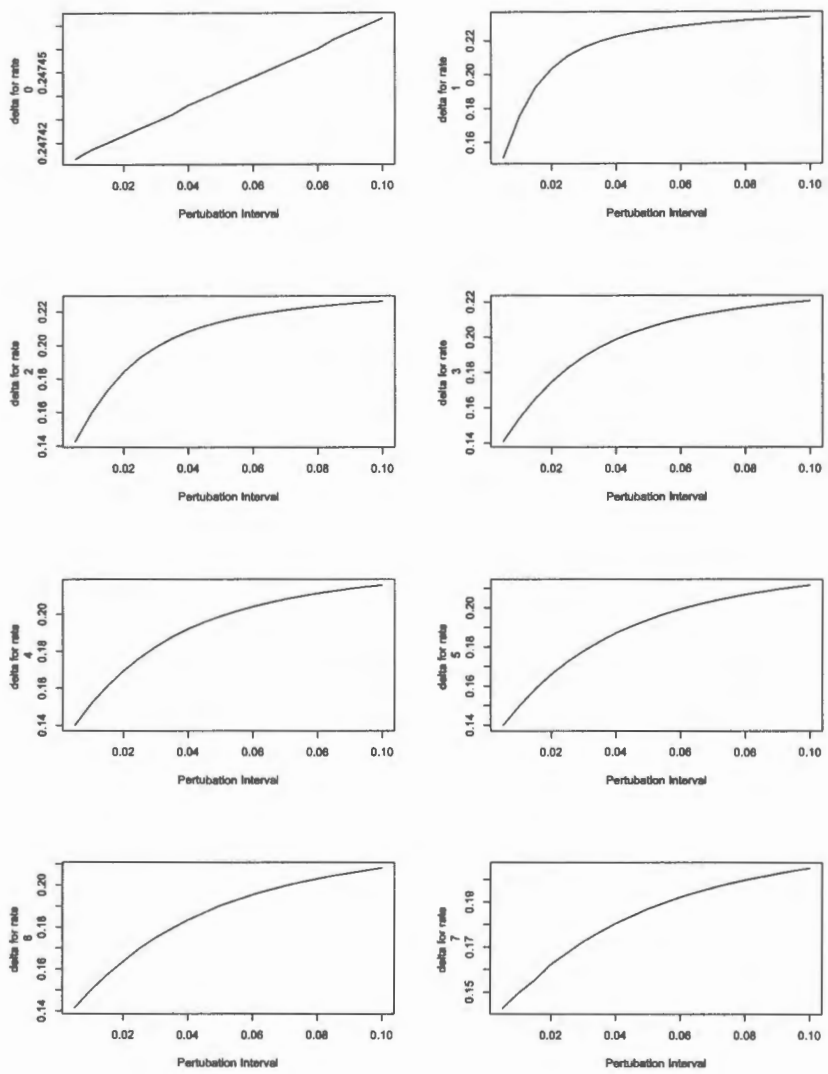

Figure 13: The Delta of the cap for various pertubation intervals.

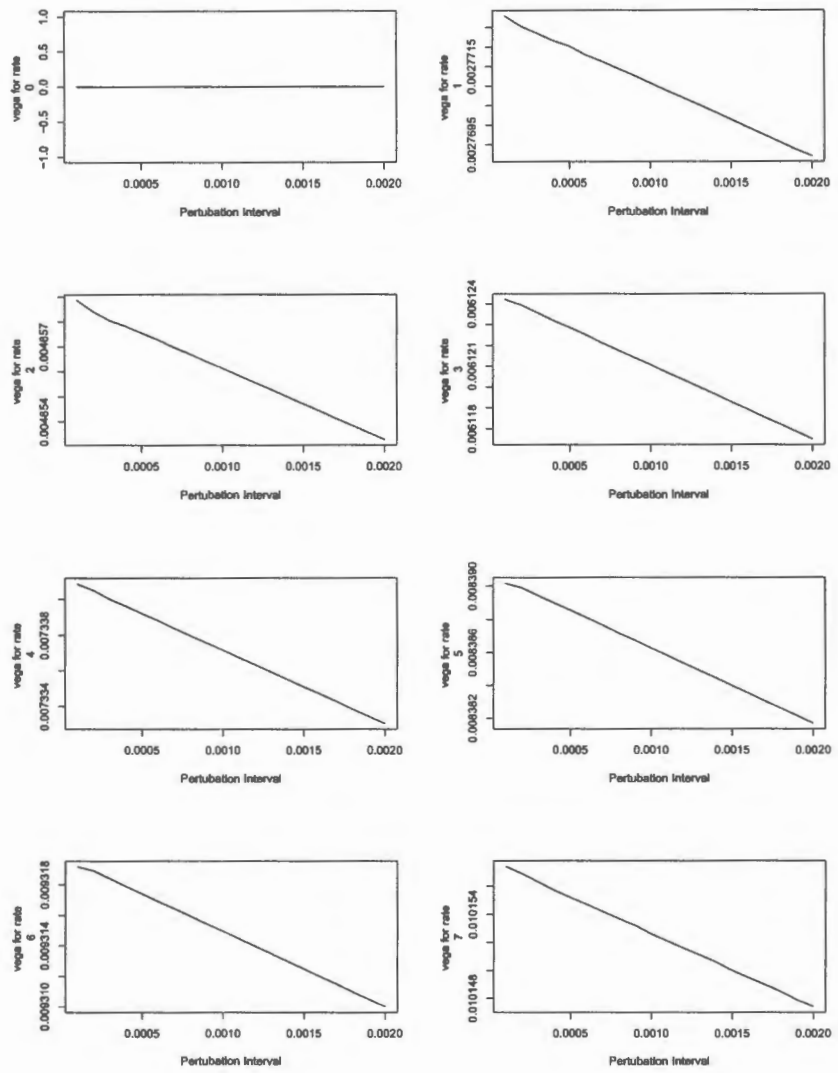

Figure 14: The Vega of the cap for different pertubation intervals. Vega is defined as the sensitivity of the price to the respective caplet volatility.

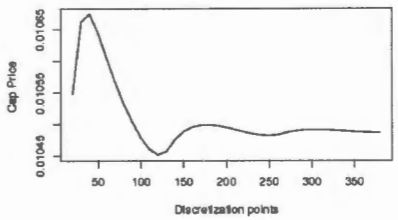

Figure 15: The price of a cap for different discretization points.

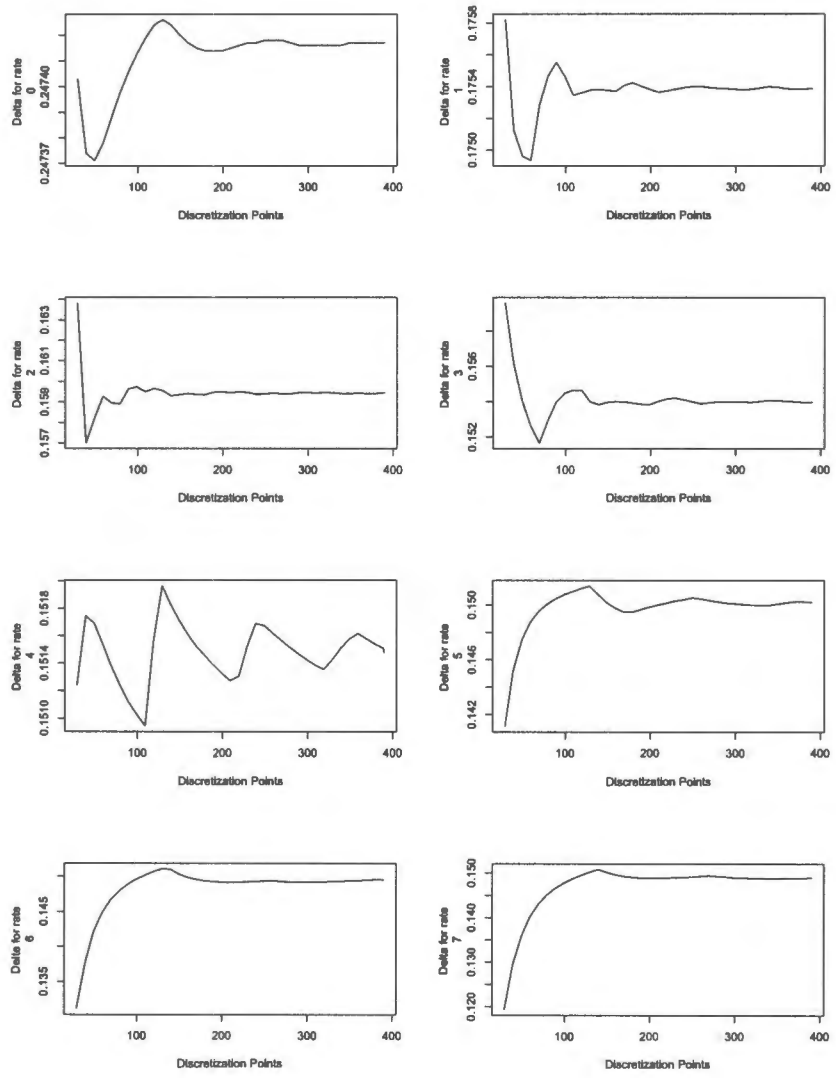

Figure 16: The Delta for the specified Libor rate of the cap for different numbers of discretization points.

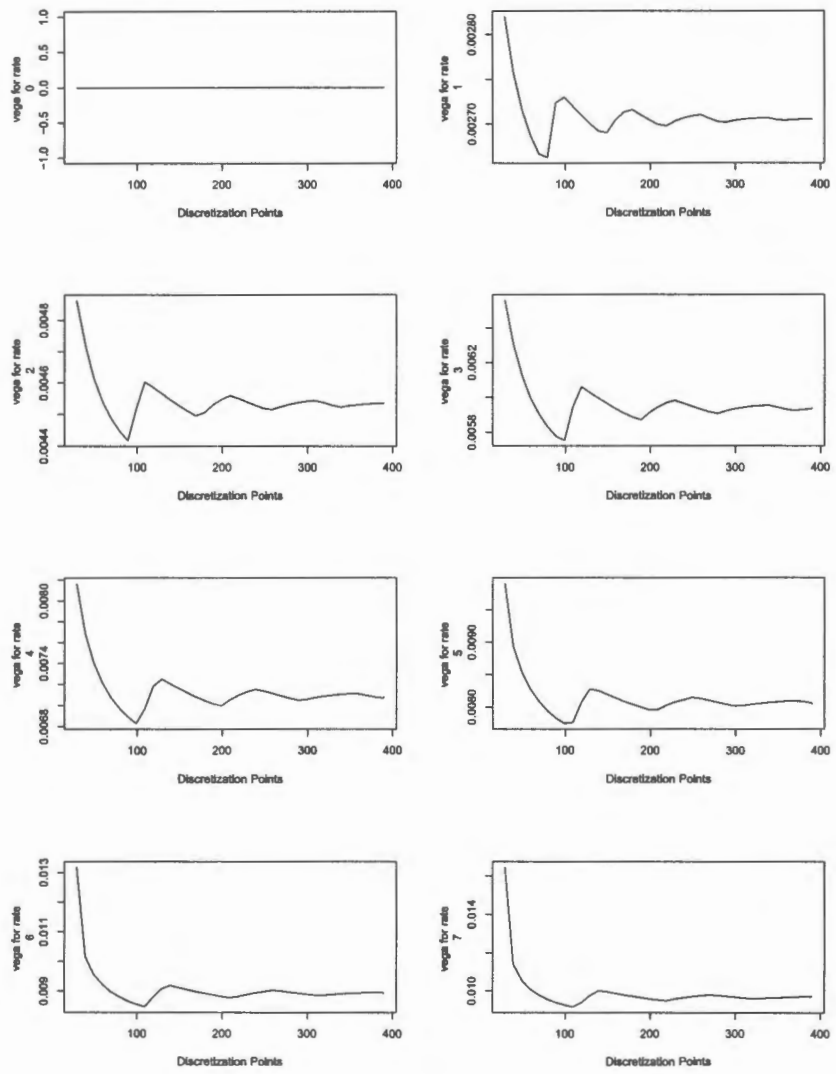

Figure 17: The vega of the cap for different numbers of discretization points.

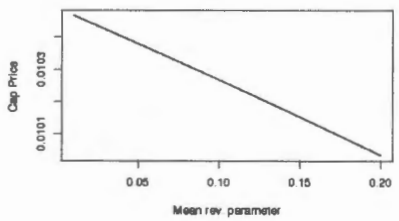

Figure 18: The price of a cap for different mean reversion parameters.

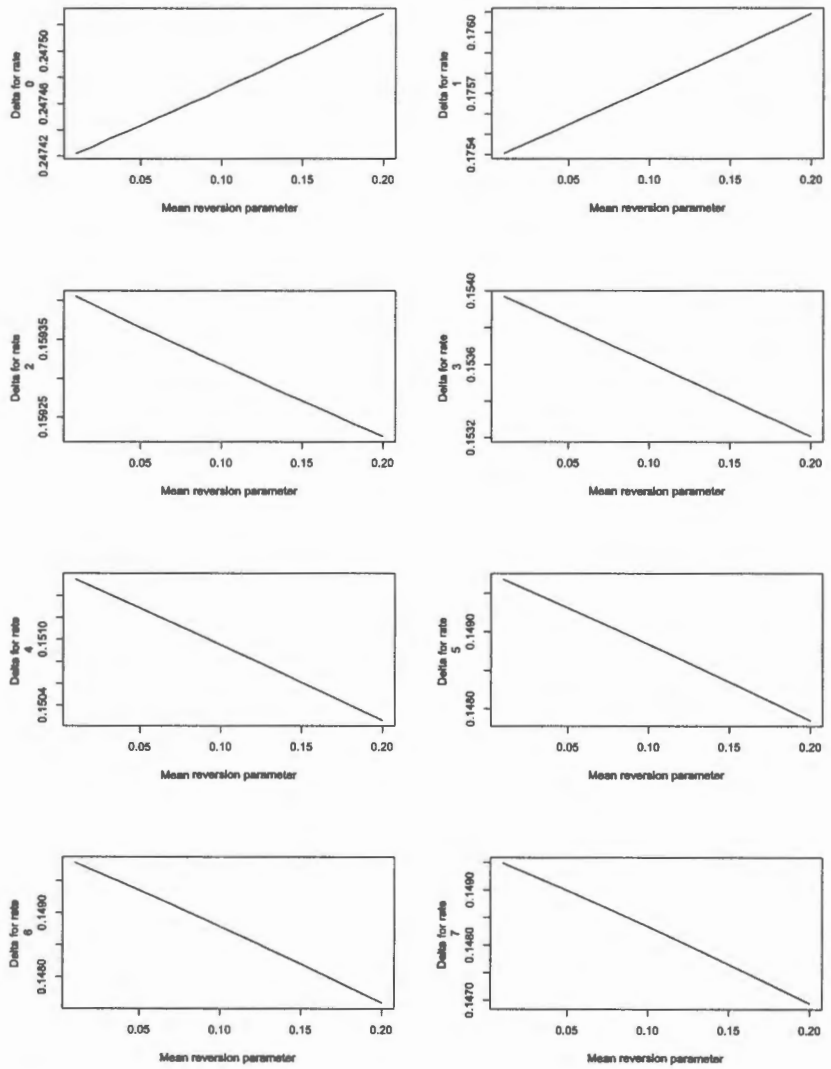

Figure 19: The Delta of the cap for different mean reversion parameters.

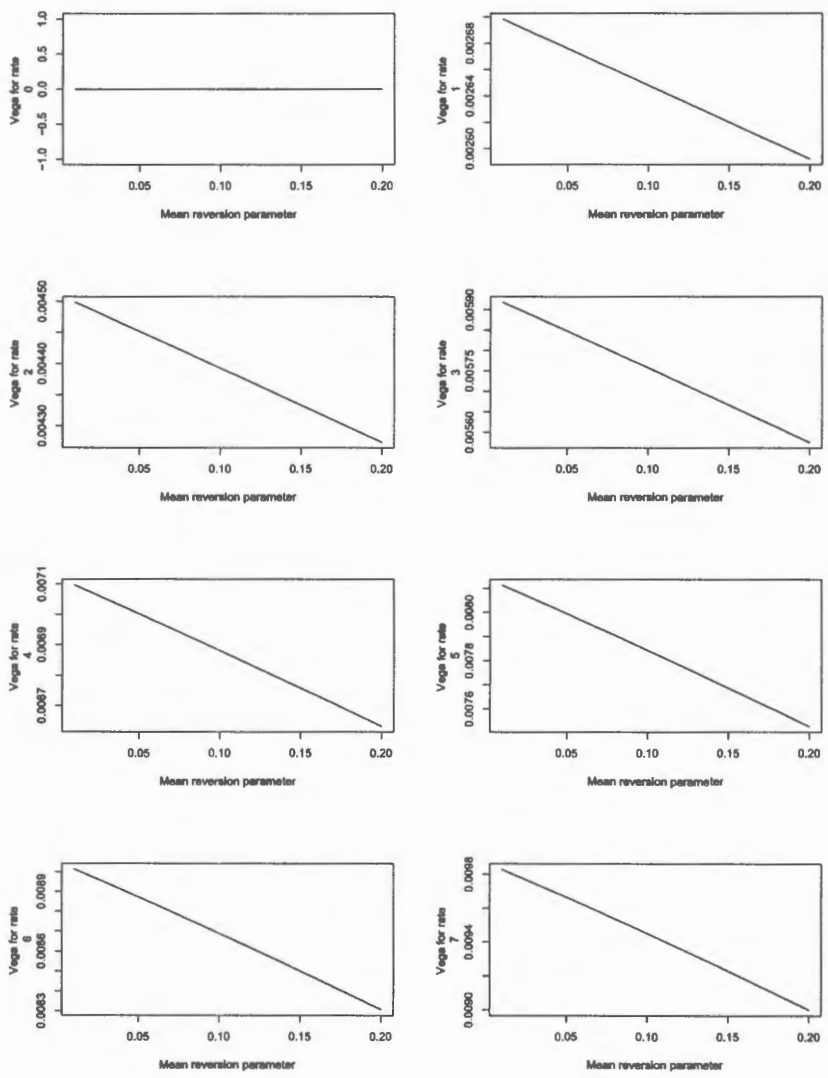

Figure 20: The Vega of the cap for different mean reversion parameters.

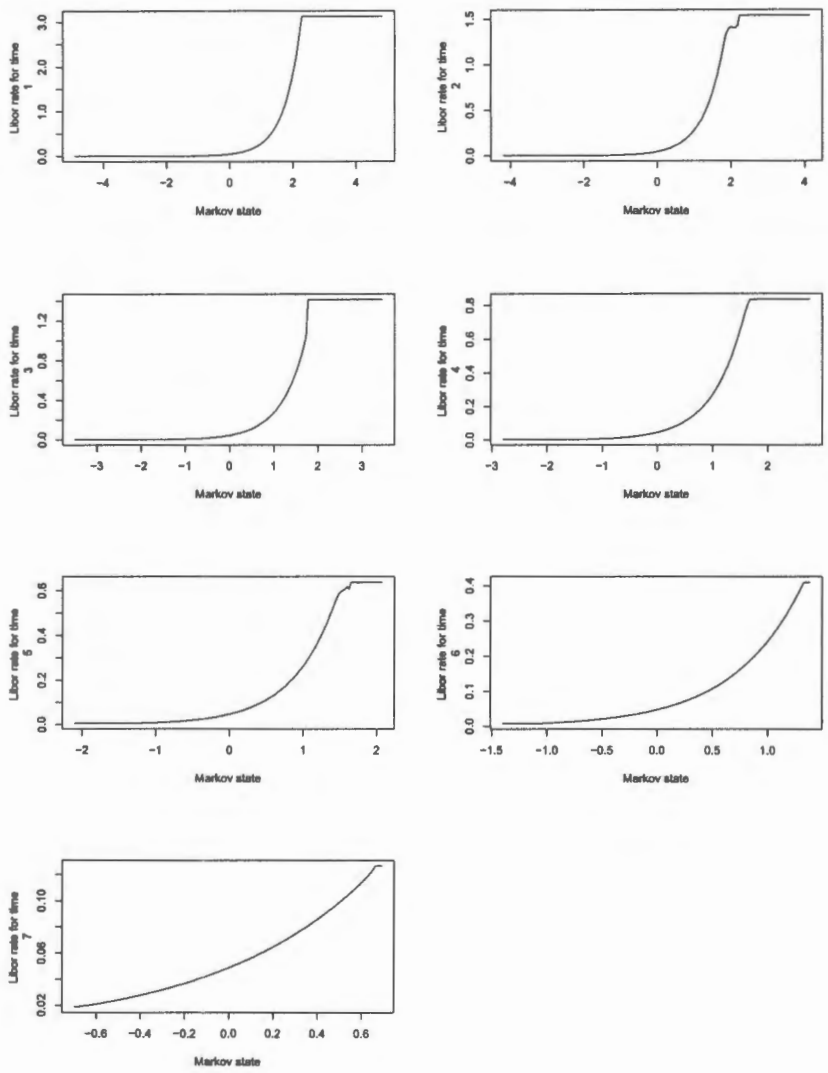

Figure 21: The functional form of the Libor rate

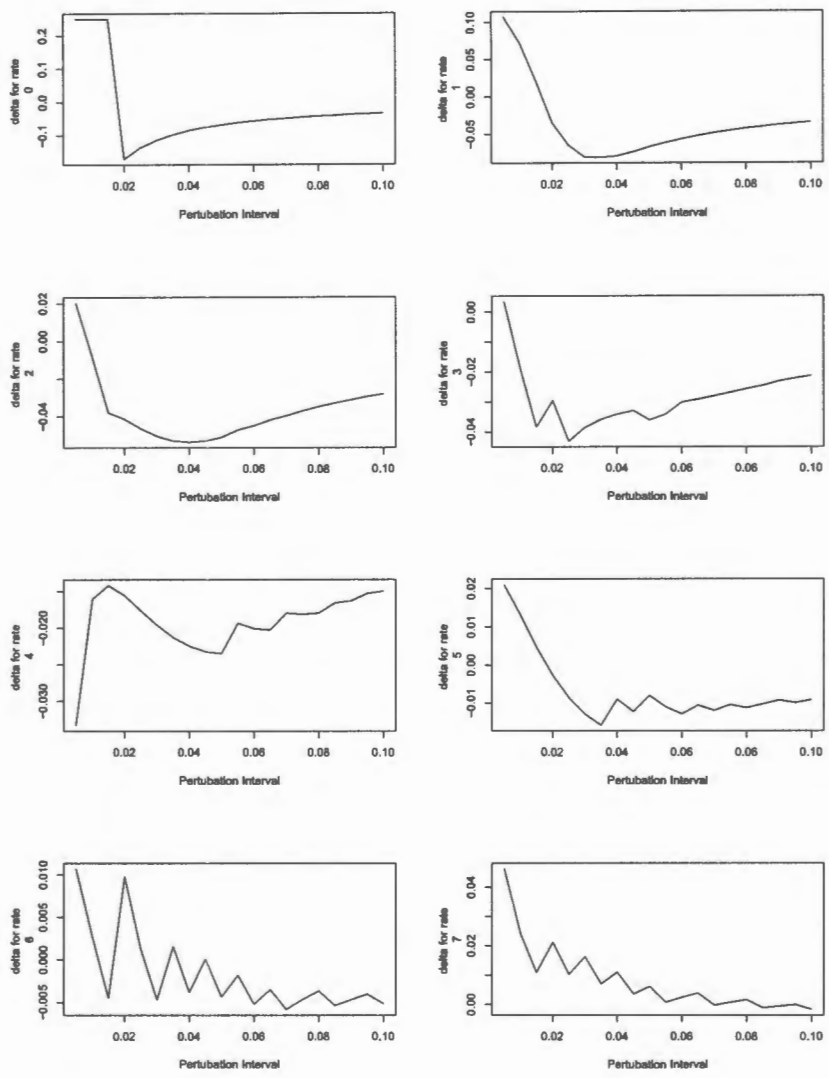

Figure 22: The Delta of the barrier cap for different pertubation intervals.

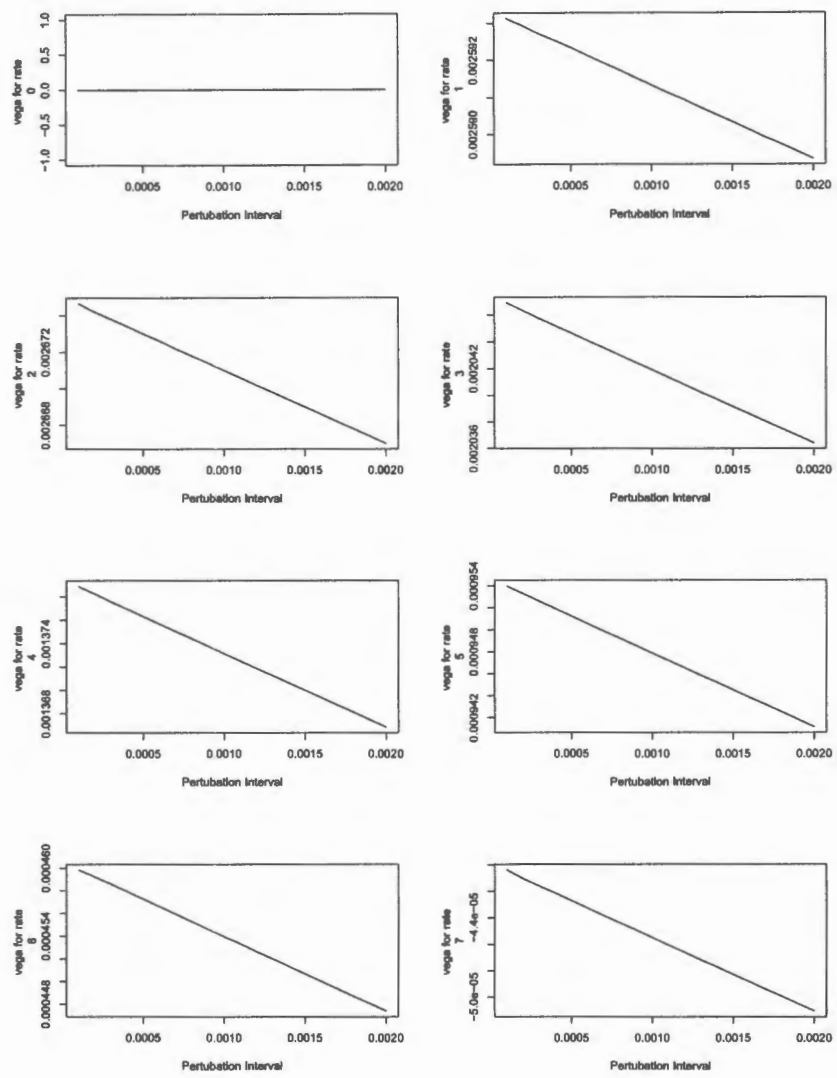

Figure 23: The Vega of the barrier cap for different pertubation intervals.

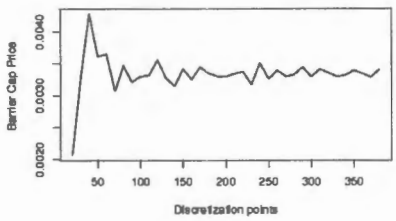

Figure 24: The price of a barrier cap for different discretization points.

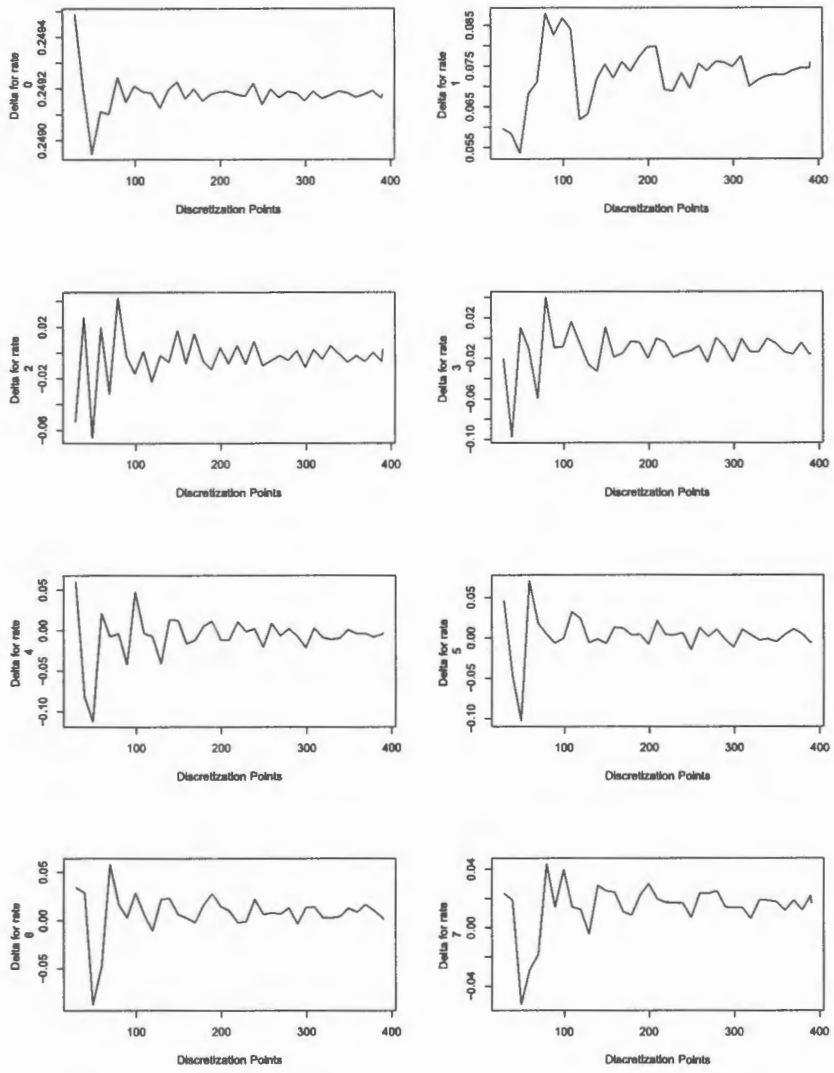

Figure 25: The Delta of the barrier cap for different numbers of discretization points.

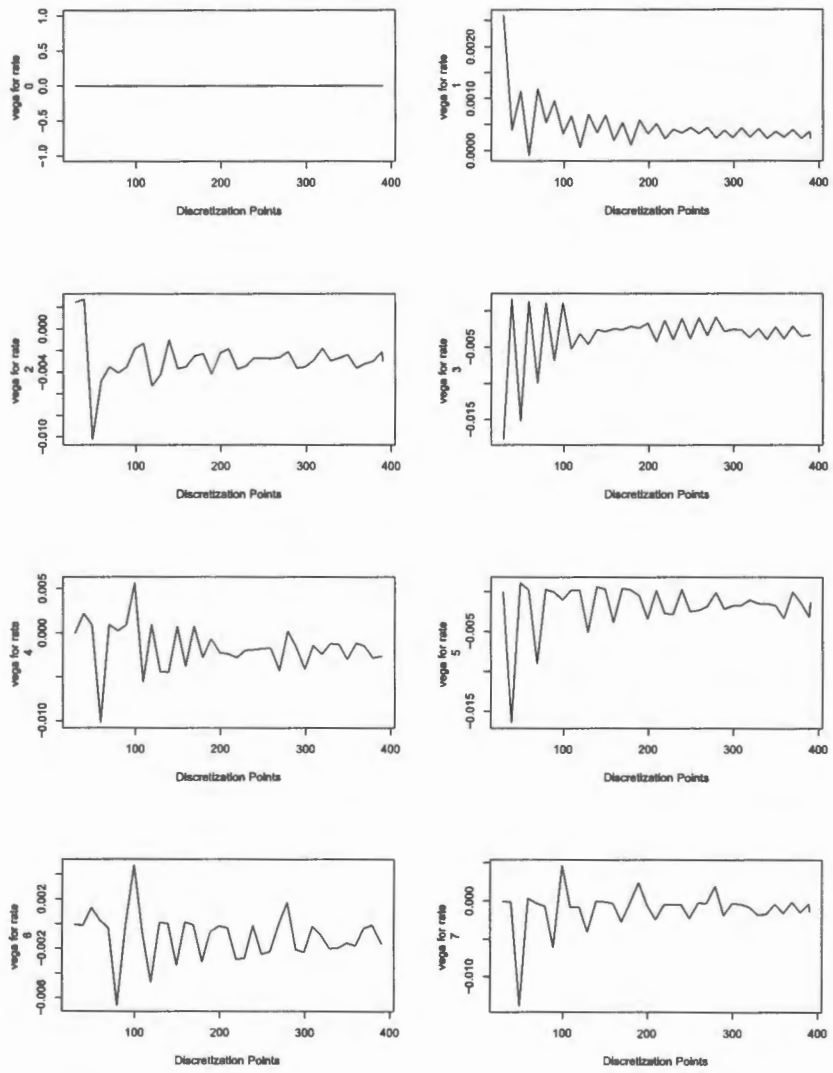

Figure 26: The vega of the barrier cap for different numbers of discretization points.

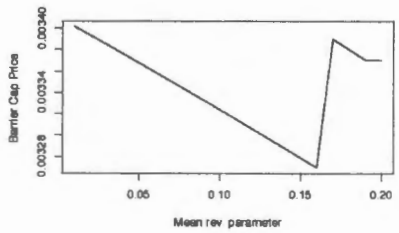

Figure 27: The price of a barrier cap for different mean reversion parameters.

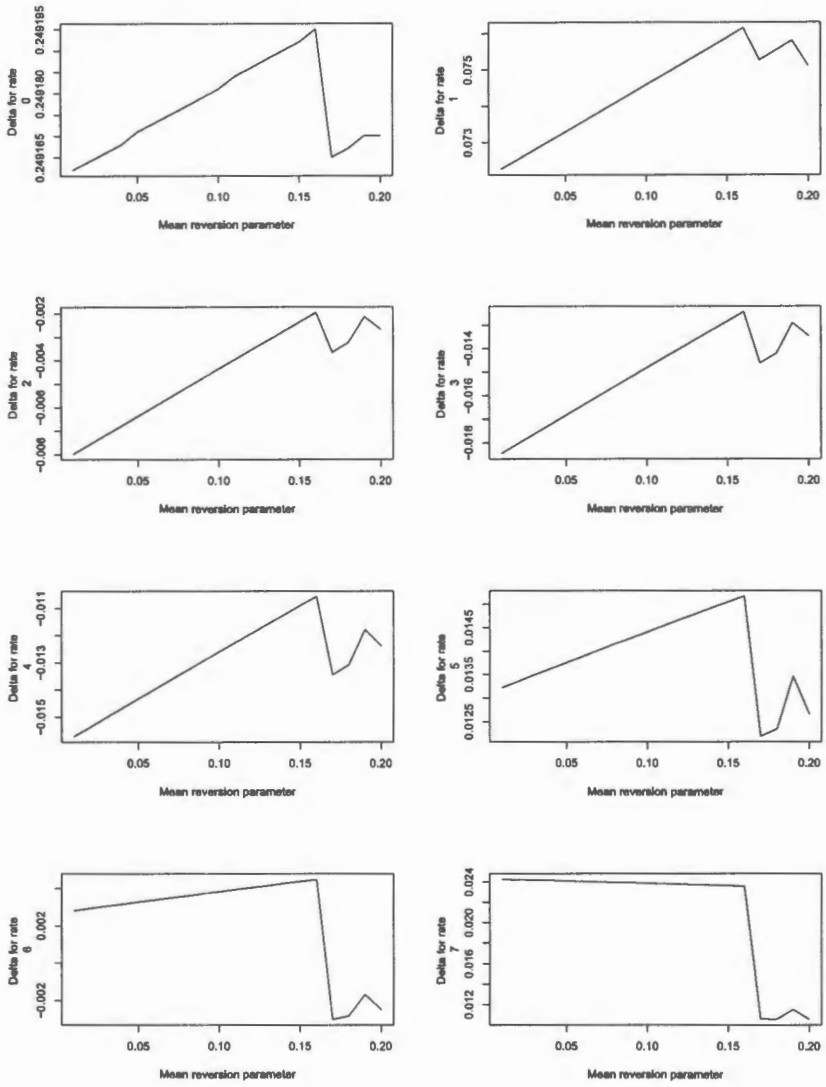

Figure 28: The Delta of the barrier cap for different mean reversion parameters.

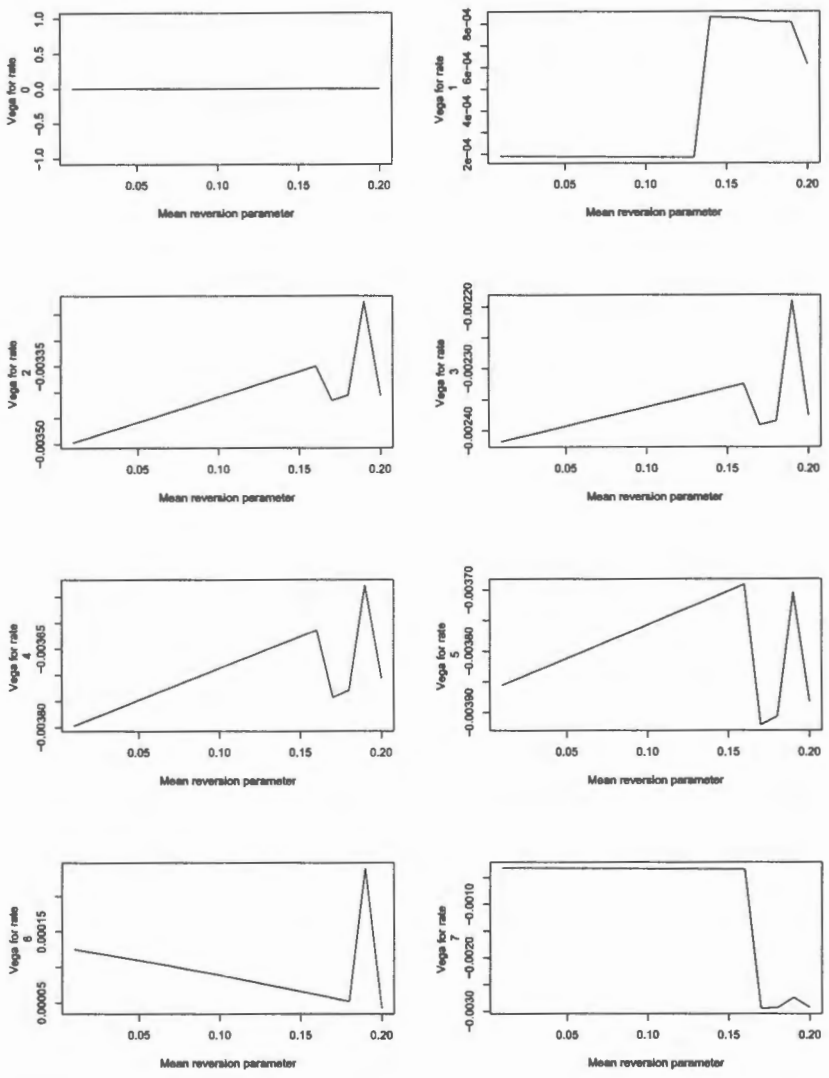

Figure 29: The Vega of the barrier cap for different mean reversion parameters.

er

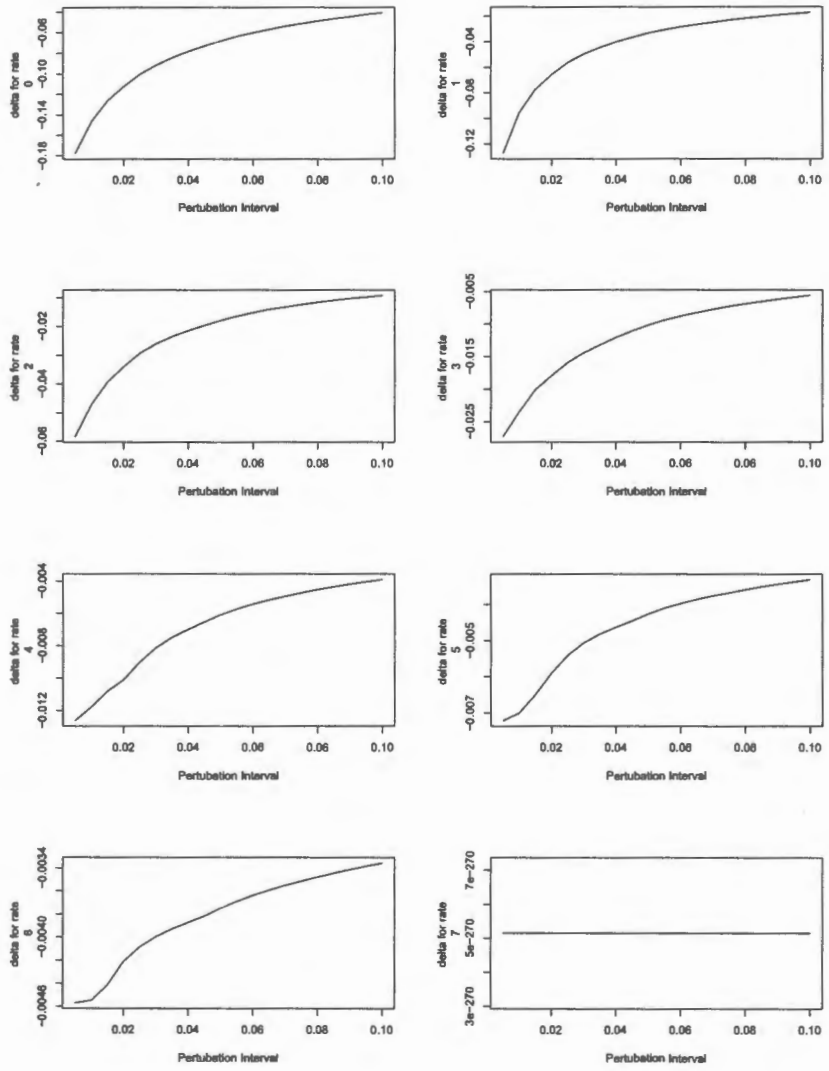

Figure 30: The Delta of the Bermudan swaption for different pertubation intervals. Delta is defined as the sensitivity of the price to the respective Libor rate

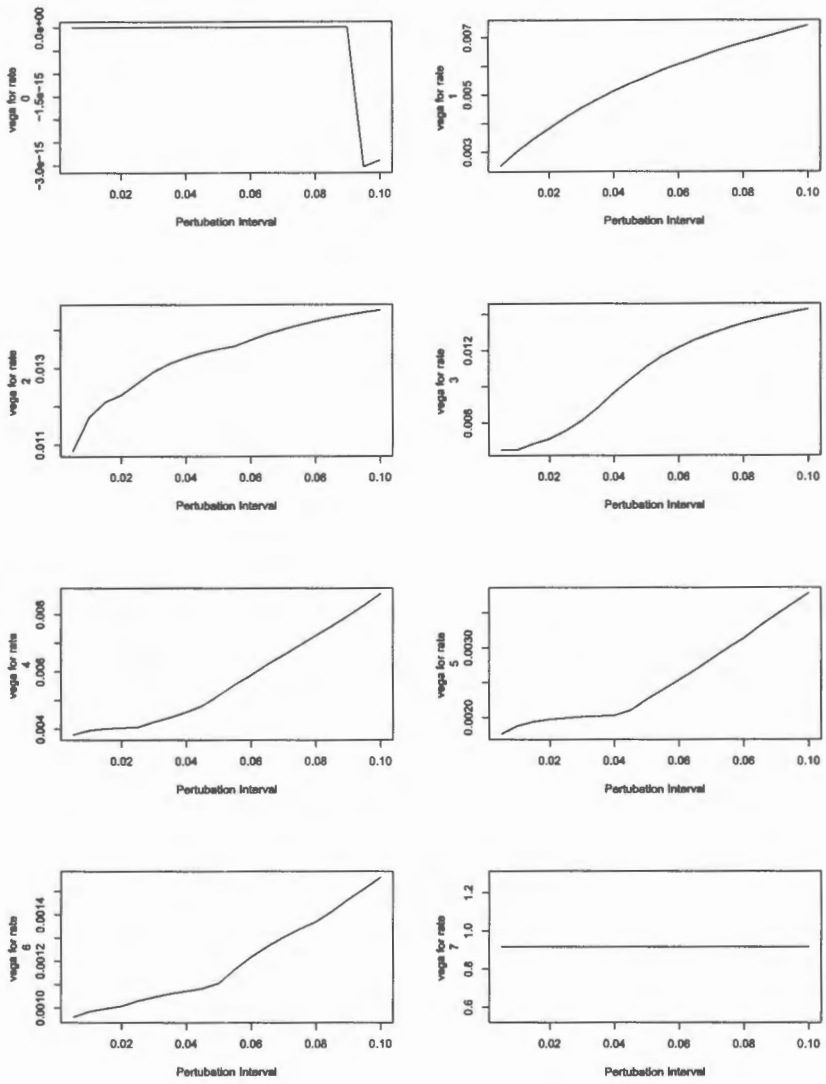

Figure 31: The vega of the Bermudan swaption for different pertubation intervals.

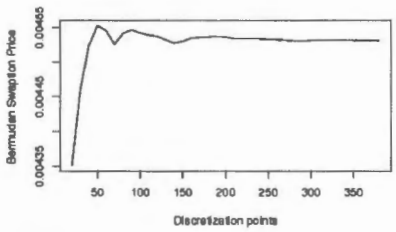

Figure 32: The price of a bermudan swaption for different discretization points.

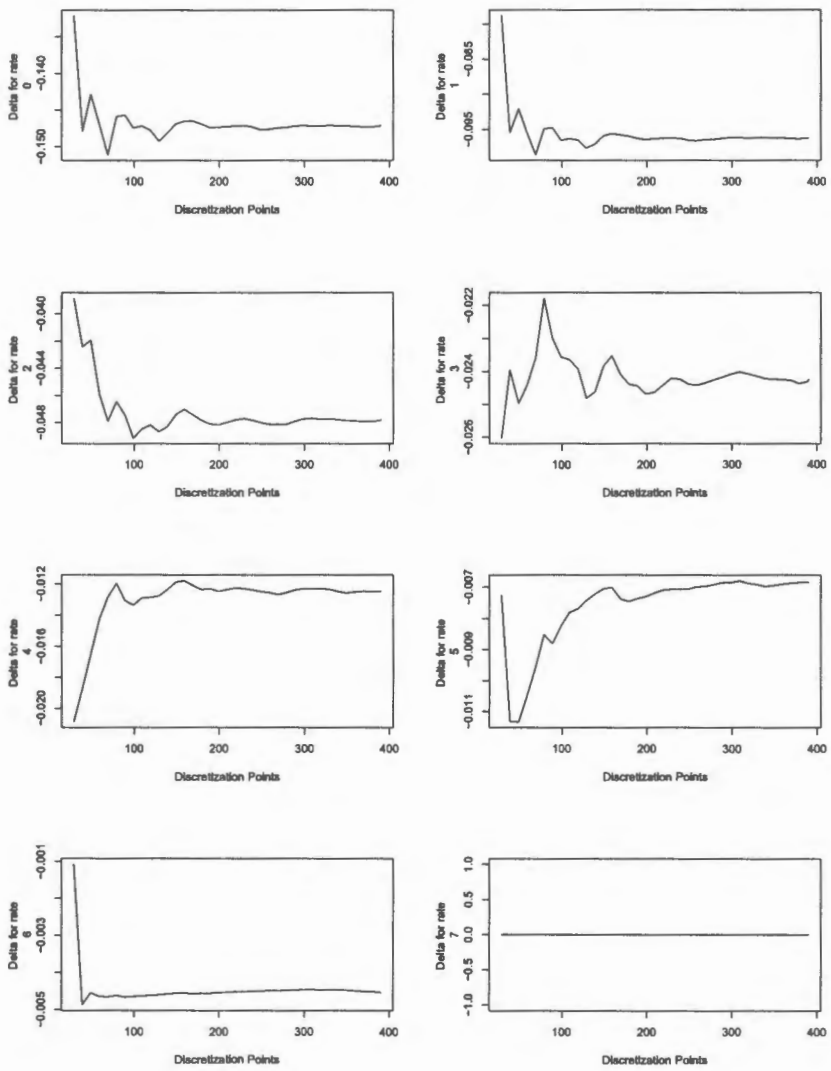

Figure 33: The Delta of the Bermudan swaption for different numbers of discretization points.

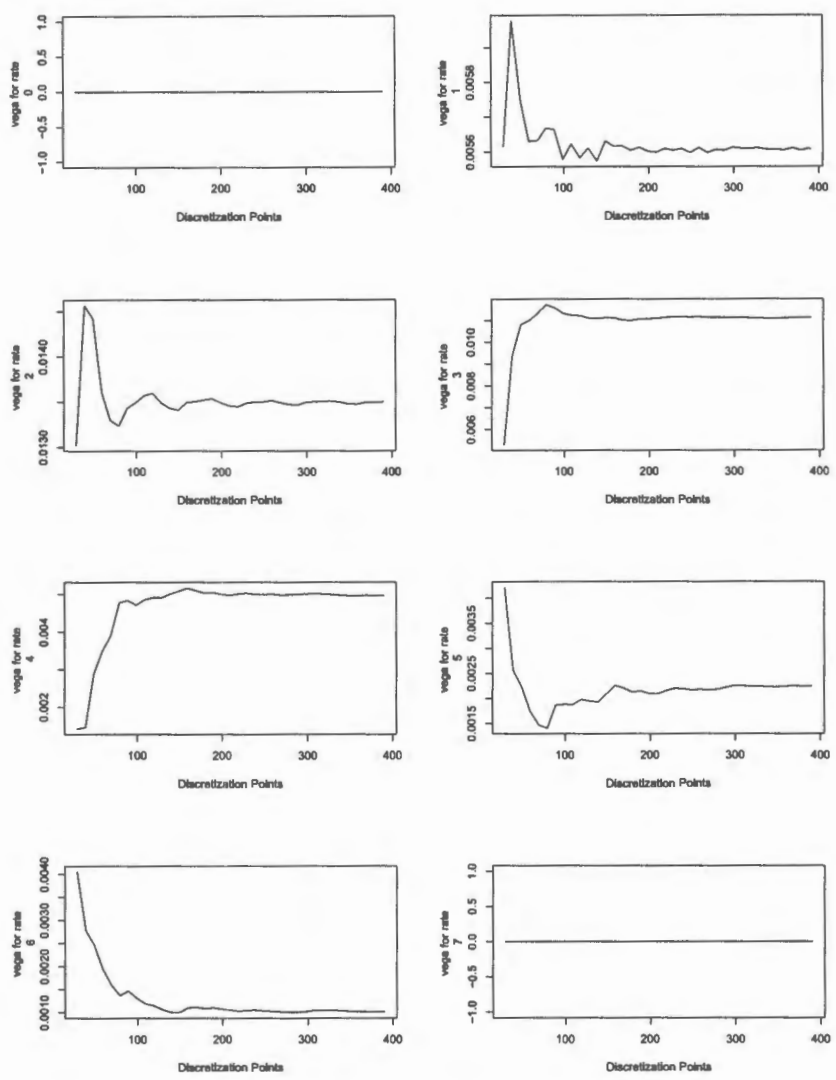

Figure 34: The Vega of the Bermudan swaption for different numbers of discretization points. Vega is defined as the sensitivity of the price to the respective swaption volatility.

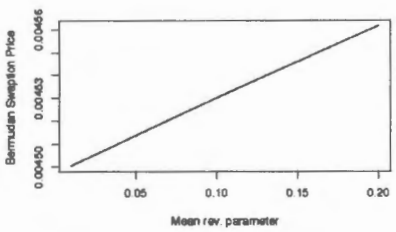

Figure 35: The price of a Bermudan swaption for different mean reversion parameters.

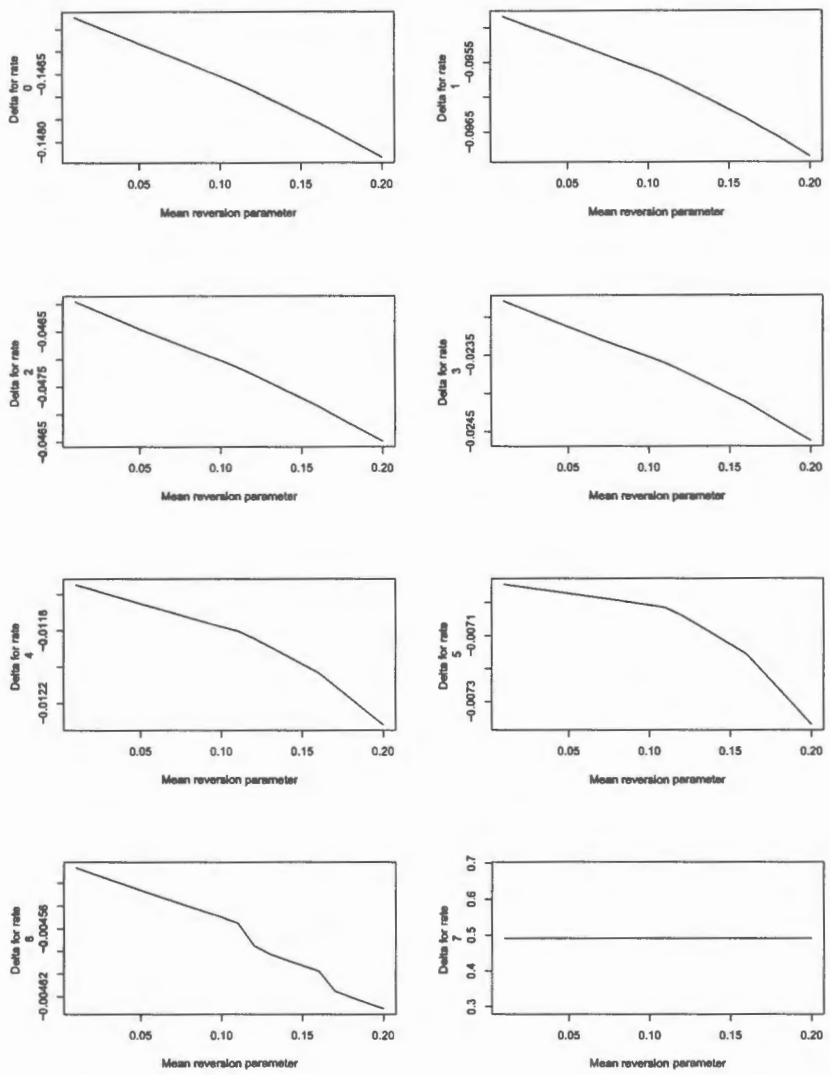

Figure 36: The Delta of the Bermudan swaption for different mean reversion parameters.

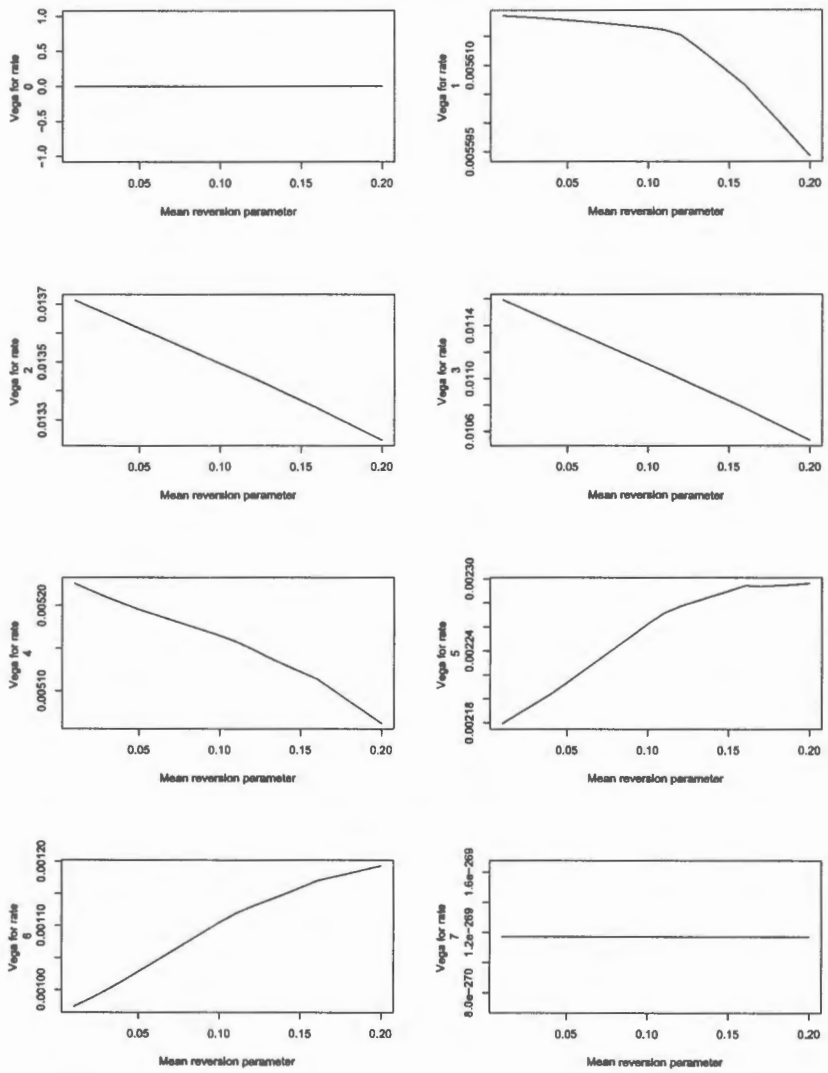

Figure 37: The Vega of the Bermudan swaption for different mean reversion parameters.

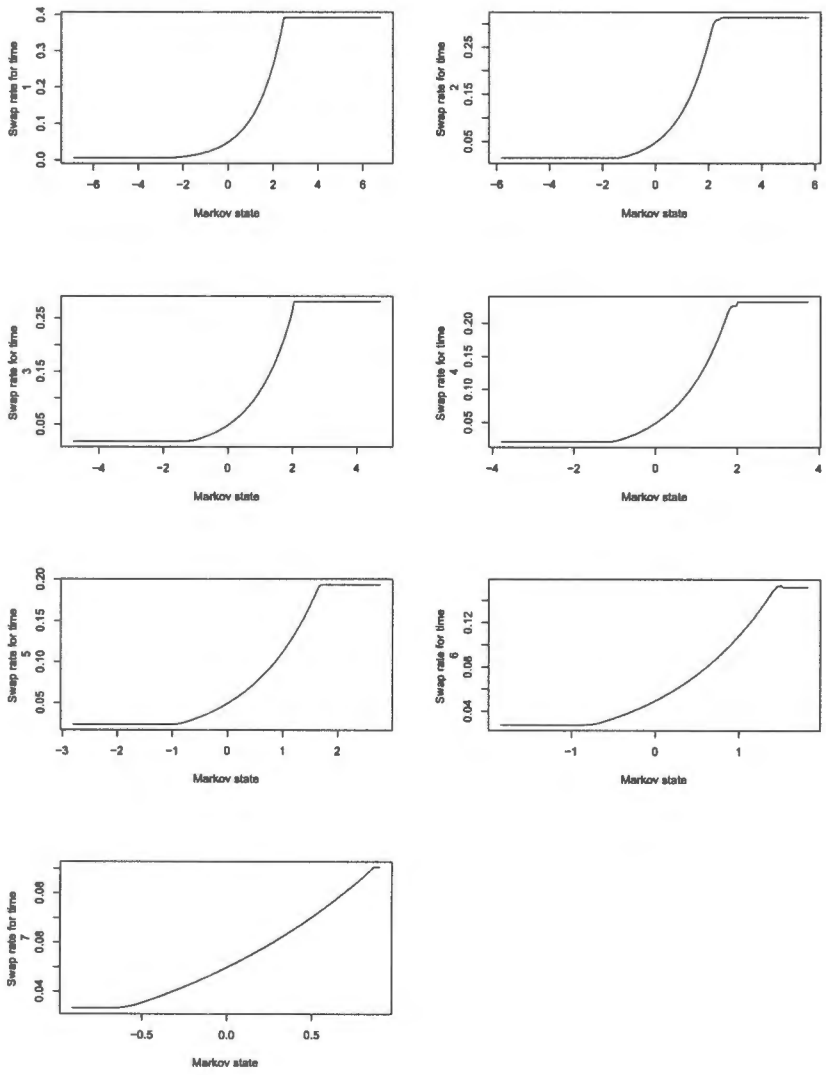

Figure 38: The functional form of the swap rates

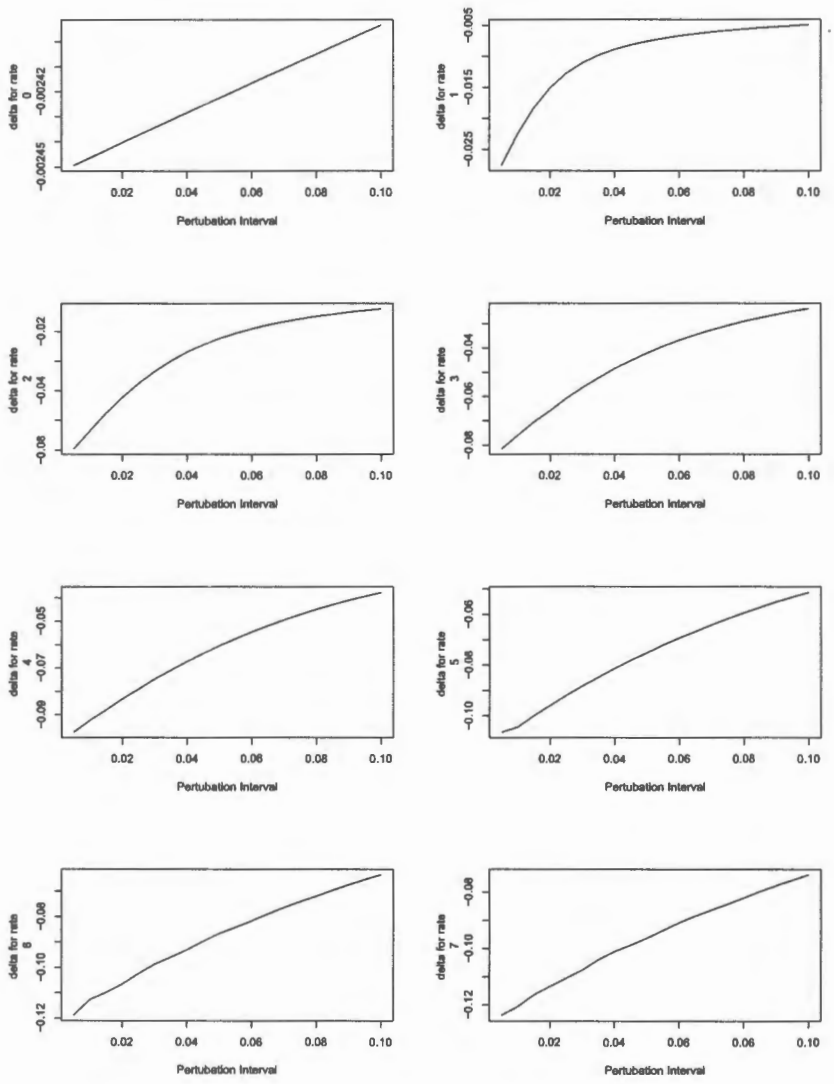

Figure 39: The Delta of the Bermudan swaption, priced using the Libor model, for different pertubation intervals. Delta is defined as the sensitivity of the price to the respective Libor rate.

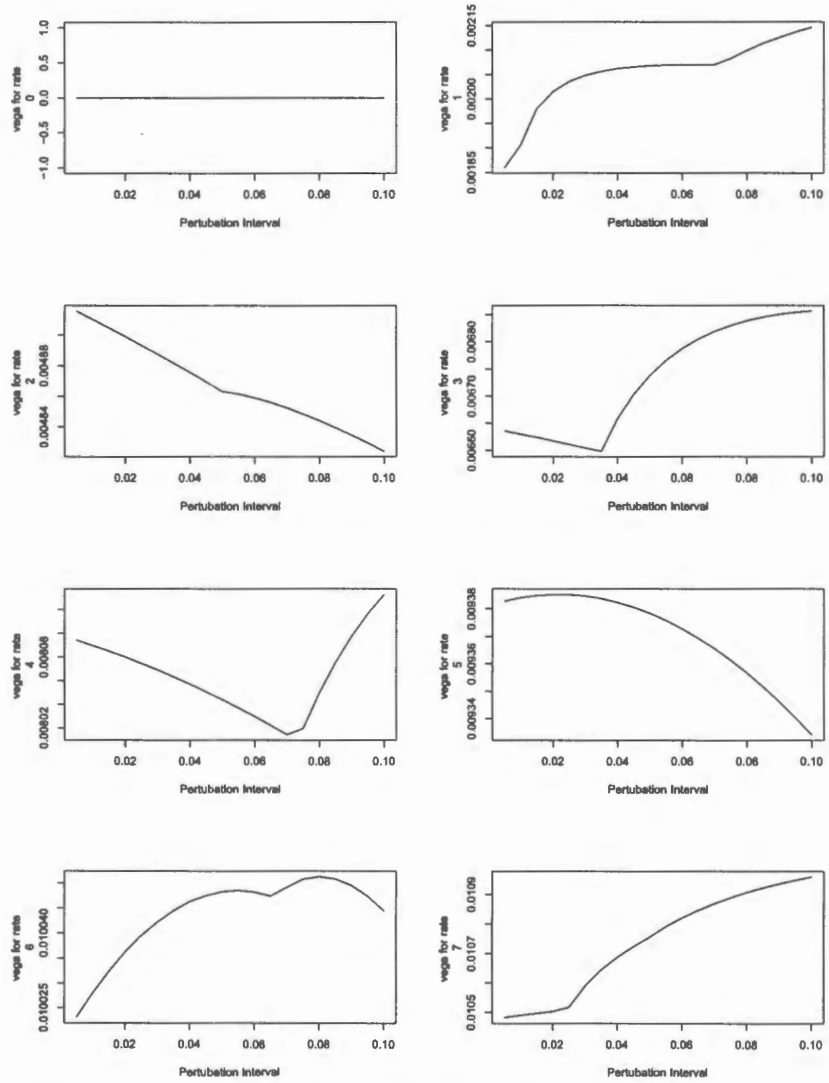

Figure 40: The Vega of the Bermudan swaption, priced using the Libor model, for different pertubation intervals. Vega is defined as the sensitivity of the price to the respective swaption volatility.

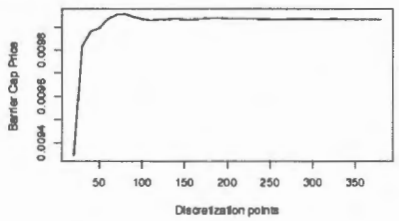

Figure 41: The price of a Bermudan swaption priced using the Libor model for different discretization points.

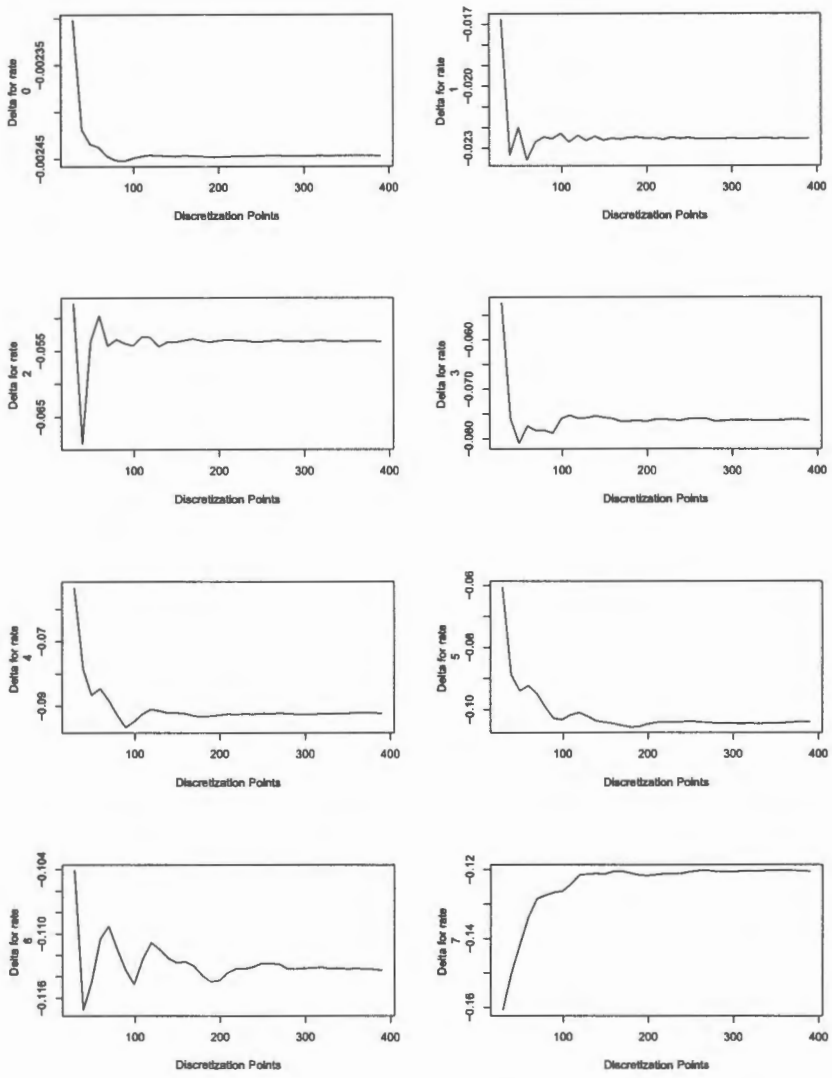

Figure 42: The Delta of the Bermudan swaption, priced using the Libor model, for different numbers of discretization points.

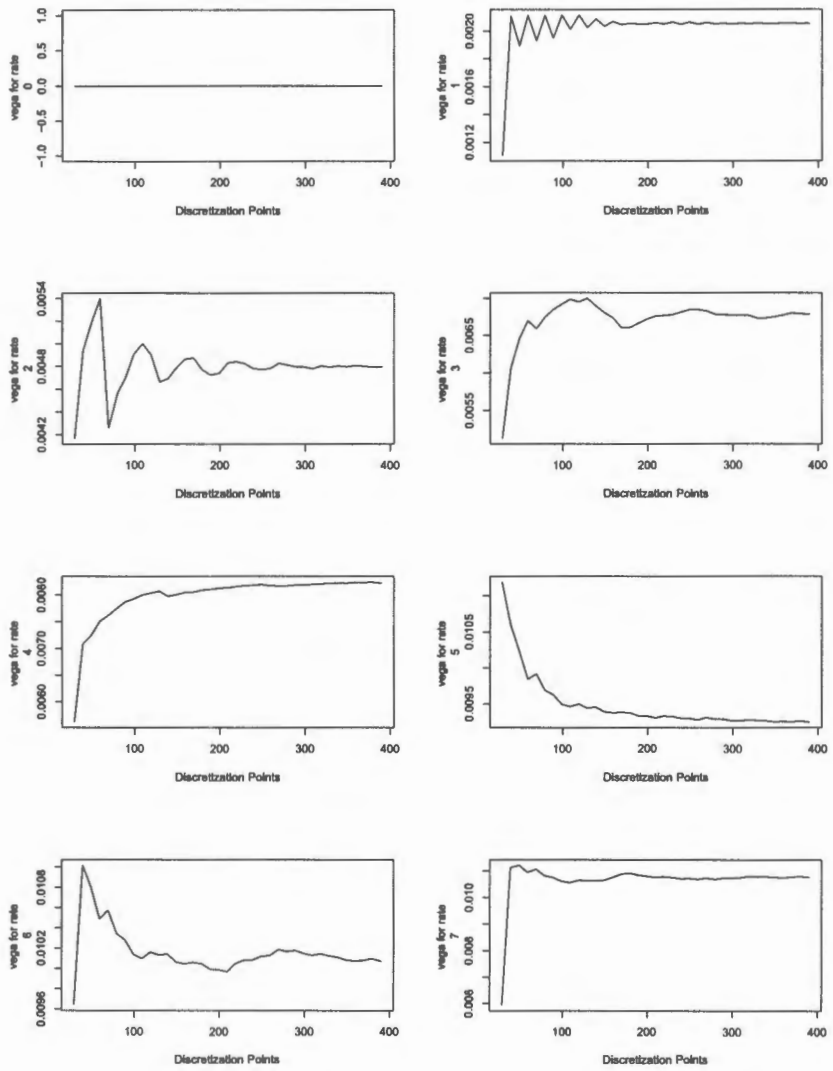

Figure 43: The Vega of the Bermudan swaption, priced using the Libor model, for different numbers of discretization points.

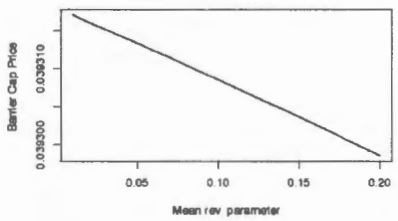

Figure 44: The price of a Bermudan swaption, priced using the Libor model, for different mean reversion parameters.

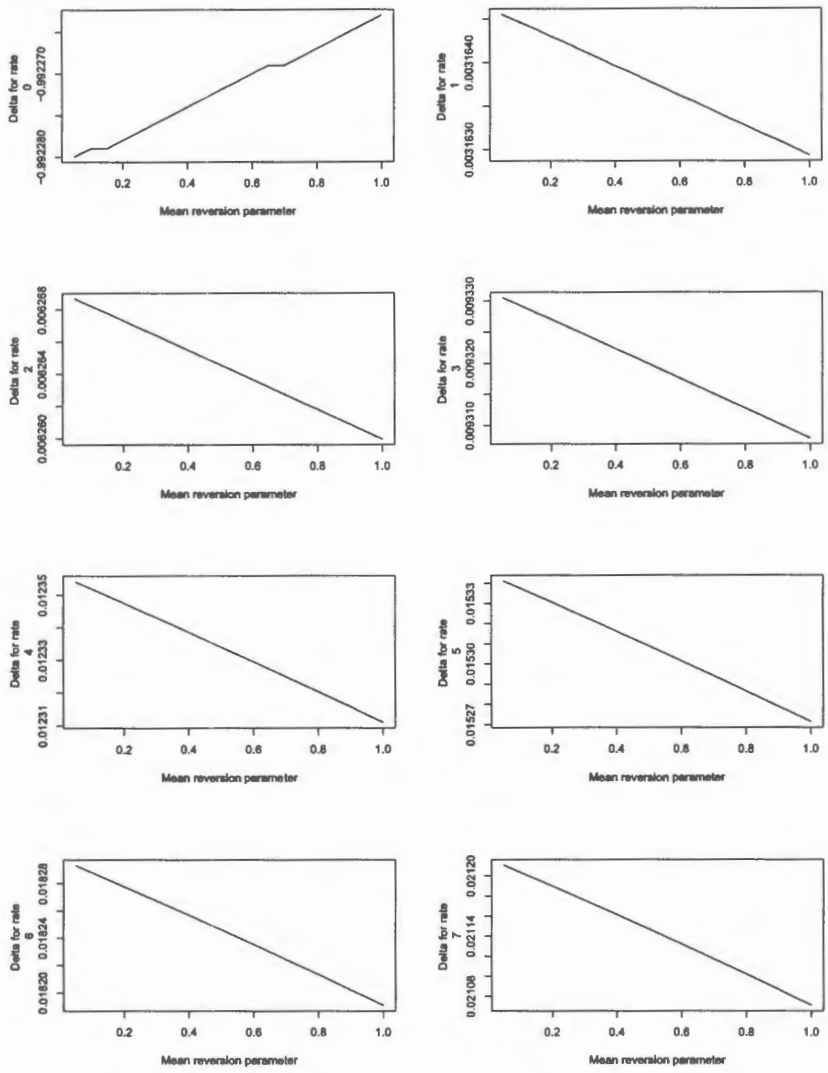

Figure 45: The Delta of the Bermudan swaption, priced using the Libor model, for different mean reversion parameters.

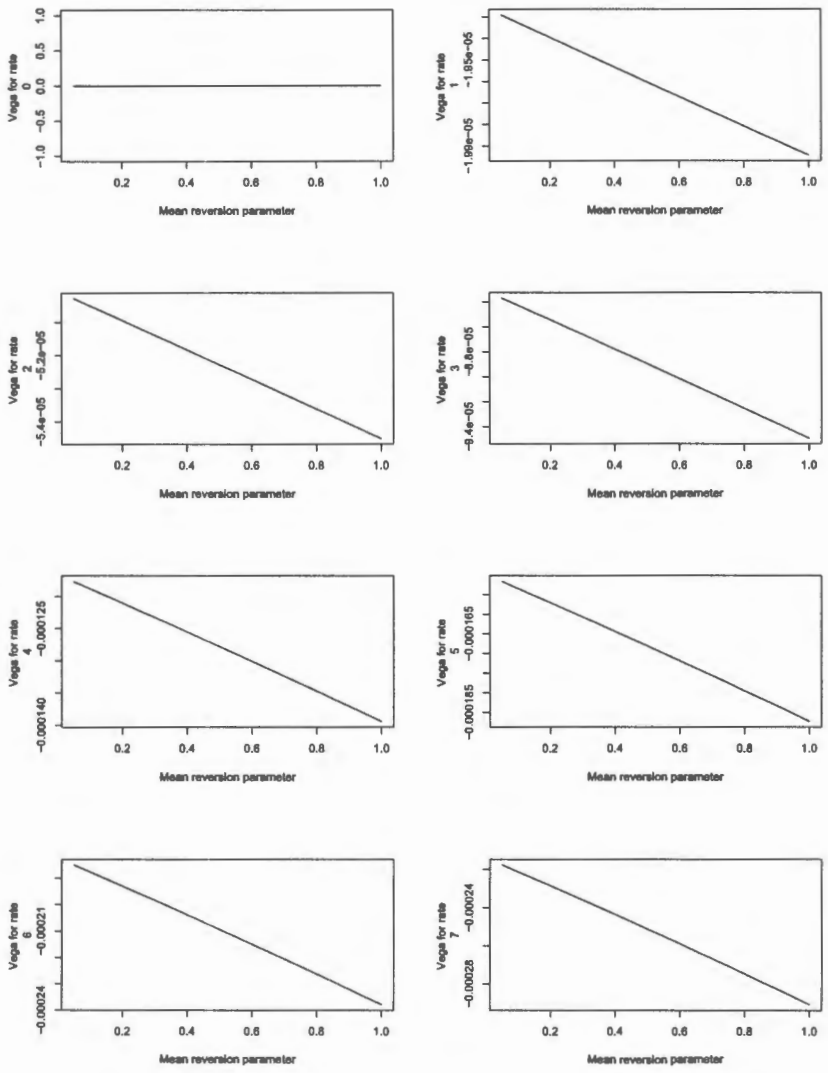

Figure 46: The Vega of the Bermudan swaption, priced using the Libor model, for different mean reversion parameters.

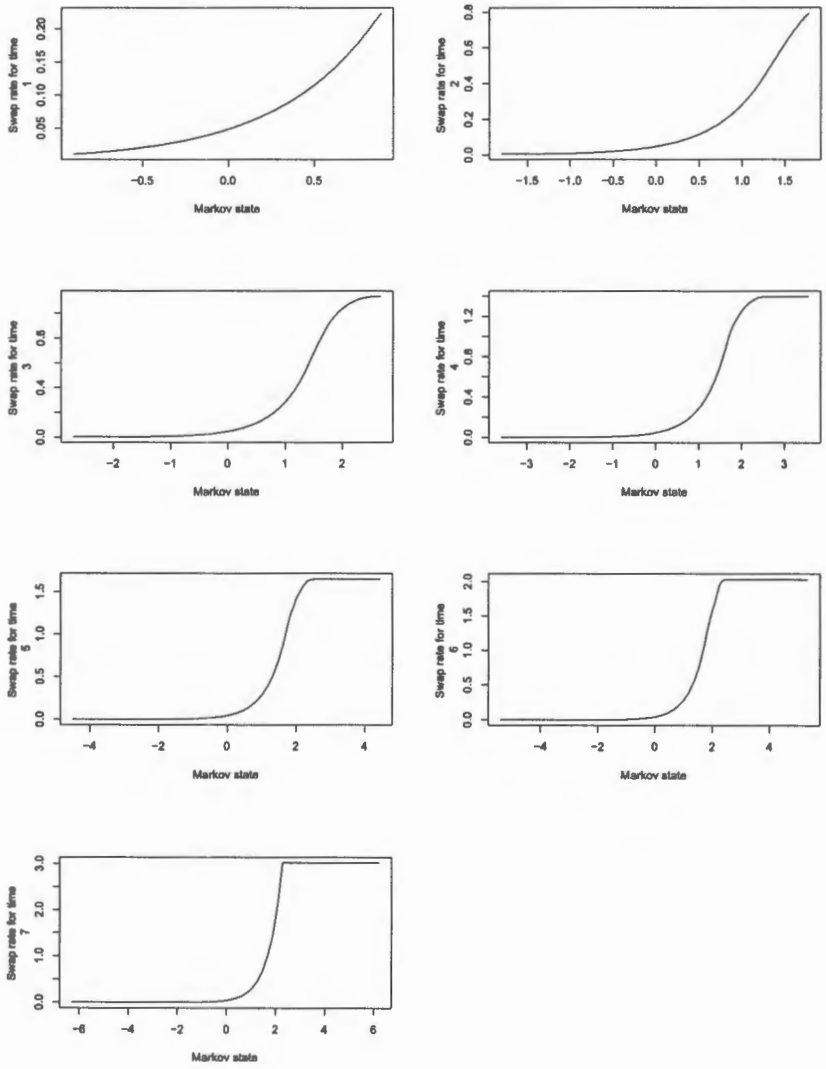

Figure 47: The functional form of the swap rates using the Libor model

B Code header files

LiborMarkovFunctional.h

```
/*************************************************************************
*This file provides a Libor Markov-Functional model. It inherits from
*MarkovFunctional class. It  calibrates the model to the  market and
*stores the values. The intermediate values can be written to specified
*files by changing the parameters. Default is not to write.
*************************************************************************/


#ifndef __included_MarkovFunctional__
#include <MarkovFunctional.h>
#define __included_MarkovFunctional__
#endif

#ifndef __included_LiborMarkovFunctional__
#define __included_LiborMarkovFunctional__

class LiborMarkovFunctional : public MarkovFunctional
{
public:
//virtual ~LiborMarkovFunctional();
~LiborMarkovFunctional();
LiborMarkovFunctional();
LiborMarkovFunctional(int to_set_num_discretization_points, int to_set_n,
int to_set_poly_order, double to_set_alpha, double to_set_markov_base_vol,
double to_set_markov_mean_rev_parameter,double to_set_norm_limit);
void calibrate();
void dbg();
double calculate_conditional_expectation(int time, int state,
 double *payoff);
protected:
void calculate_L(int current_time, double *J_local,
 double initial_D_local);
void calculate_D(double *J_local, double *D_local, double *L_local);
void calculate_J(int current_time, double *J_local, double *x_local,
double *D_local, double discount, double *future_x);
void calc_L_smile(int current_time, double *J_local,
 double initial_D_local);
```

```
void calculate_L_displaced_diffusion(int current_time, double *J_local,
 double initial_D_local);
double *displaced_diffusion_r;
void get_displaced_diffusion_parameters();


};

#endif
```

MarkovFunctional.h

```
/*************************************************************************
* This file provides the basic framework for a Markov-Functional model
* All my specific models inherit from this. This makes it easy to create
* another model.
*************************************************************************/


#include <MeanRevVolatility.h>

class MarkovFunctional
{
public:
MarkovFunctional();
MarkovFunctional(int to_set_num_discretization_points,
 int to_set_n, int to_set_poly_order,
 double to_set_alpha, double to_set_markov_base_vol,
   double to_set_market_mean_rev_parameter,
   double to_set_norm_limit);
virtual ~MarkovFunctional();
bool write_to_file;

protected:
//parameters
int num_discretization_points;
int poly_order;
int n;
double alpha;
double norm_limit;
double *inner;
double ***L;
double **x;
double **J;
double **D;
double *initial_D;
double markov_mean_rev_parameter;
MeanRevVolatility markov_vol;
MeanRevVolatility markov_unconditional_vol;
Volatility market_vol;
```

```
//functions
void set_poly_order(int to_set);
void set_alpha(double to_set);
void set_markov_points(double upper_limit, double lower_limit,
 int num_discretization_points, double *x_local);
void set_norm_limit(double to_set);
void set_markov_volatility(double to_set_base_vol,
 double to_set_mean_rev_parameter);
void set_markov_volatility_unconditional(double to_set_base_vol,
 double to_set_mean_rev_parameter);

void calculate_markov_points();
void set_initial_libor_rates();
void calculate_initial_D();
void set_initial_market_vol();

/*The following function was used for simulation
purposes and has been commented out.
It is fully functional so the comment can be removed
here and in the source file. */
//void set_market_volatility(double to_set_base_vol,
 //double to_set_mean_rev_parameter);

};
```

MeanRevVolatility.h

```
/*************************************************************************
*Provides a volatility object with the volatility calculated according
*to the mean reversion parameter. This object is only 1-D,so the state
*index is 0 always. I will not do this, but rather inefficiently fill the
*entire array with the same value. The reason for this is that now I can
*include a state term in the calculations and it will be "used" when it
*is relevant e.g. for other volatility objects that have states so I
*don't need to change the calculating code. Constant volatility is a
*special case of this
*************************************************************************/

#include<Volatility.h>

class MeanRevVolatility : public Volatility
{
public:
MeanRevVolatility();
virtual ~MeanRevVolatility();
MeanRevVolatility(double to_set_base_vol,
 int to_set_num_discretization_points,
 int to_set_time_periods, double to_set_mean_rev_parameter,
 double to_set_alpha);
void set_mean_rev_parameter(double to_set_mean_rev_parameter);
void set_all_parameters(double to_set_base_vol,
 int to_set_num_discretization_points,
 int to_set_time_periods, double to_set_mean_rev_parameter,
 double to_set_alpha);
void set_all_parameters_unconditional(double to_set_base_vol,
 int to_set_num_discretization_points,
 int to_set_time_periods,  double to_set_mean_rev_parameter,
 double to_set_alpha);
void set_mean_rev_parameter_unconditional(
 double to_set_mean_rev_parameter);
double get_base_vol();

protected:
double mean_rev_parameter;
double base_vol;
double alpha;
```

```
};
```

SwapMarkovFunctional.h

```
/*************************************************************************
*This file provides the framework and specific functions for a Swap
* Markov-Functional model. All specific swap based products inherit
* from this
*************************************************************************/



#ifndef __included_MarkovFunctional__
#include <MarkovFunctional.h>
#define __included_MarkovFunctional__
#endif

#ifndef __included_SwapMarkovFunctional__
#define __included_SwapMarkovFunctional__

class  SwapMarkovFunctional : public MarkovFunctional
{
public:
virtual ~SwapMarkovFunctional();
SwapMarkovFunctional();
SwapMarkovFunctional(int to_set_num_discretization_points,
int to_set_n, int to_set_poly_order, double to_set_alpha,
double to_set_markov_base_vol,
double to_set_markov_mean_rev_parameter,
double to_set_norm_limit);
void calibrate();
void calculate_libor_rates_from_swap_rates();
void calculate_swap_tree(double fixed_rate, bool receive_fixed);


protected:
double **PVBP;
double *initial_PVBP;
double **y; //par swap rate
double *initial_swap_rate;
double **swap_tree;
double *PVBP_zero_i;
```

```
void calculate_J(int current_time, double *J_local, double *x_local,
 double *D_local, double discount, double *future_x);
void calculate_PVBP(int current_time);
void calculate_par_swap_rate(int current_time);
void calculate_D_from_par_swap_rate(int current_time);
void calculate_initial_swap_rates();
void calculate_PVBP_zero_i();

void calculate_initial_PVBP();
void initialize_PVBP();
double calculate_conditional_expectation(int time, int state,
  double *payoff);


};
#endif


/*
Internal Workings

START
|
|
|----> Make object and set params (inherited params only)
|      [own parameters all made and dealt with - nothing needs to be set]
|
|---->calibrate
| |------>calculate initial_D
| |------>calculate initial PVBP
| |------>Other initilizing stuff
| |------>calculate_initial_swap_rates
| |------>calculate_J
| |----->calculate_gaussian_polynomial_integral_by_segments
| |------>calculate_par_swap_rate
| |------>calculate_D_from_par_swap_rate
| |------>calculate_PVBP
|
|----->price_bermudan_swaption
| |------>calculate_swap_tree
```

```
|
END

*/
```

Volatility.h

```
/*************************************************************************
* Volatility.h
* -Volatility class
* -basically a poor 2D array class
* -if the default constructor is called it makes a useless object. I could
* make it so that it fixes the object once one sets the parameters, but
* this is pointlesa as I will never use it like that
* default constructor still useful for derived classes.
*************************************************************************/
class Volatility
{
public:
Volatility(int to_set_num_discretization_points,
 int to_set_time_periods);
Volatility();
virtual ~Volatility();
double get_sigma(int time, int state);
void set_sigma(int time, int state, double value_to_set);
void set_dimensions(int to_set_num_discretization_points,
 int to_set_time_periods);


protected:
int num_discretization_points;
int time_periods;
double **sigma;
};
```

barrierCap.h

```
/***********************************************
* barrierCap.h
* The file calculate the price of a Barrier cap as
* well as the greeks using the Libor MF Model
***********************************************/


#ifndef __included_MarkovFunctional__
#include <MarkovFunctional.h>
#define __included_MarkovFunctional__
#endif

#ifndef __included_LiborMarkovFunctional__
#include <LiborMarkovFunctional.h>
#define __included_LiborMarkovFunctional__
#endif

class barrierCap: public LiborMarkovFunctional
{
public:
//virtual ~barrierCap();
barrierCap();
barrierCap(double to_set_upper_barrier,
 double to_set_lower_barrier, double to_set_strike_rate,
 double to_set_pertubation_interval_vega,
 double to_set_pertubation_interval_delta,
 int to_set_cap_period,int to_set_num_discretization_points,
 int to_set_n, int to_set_poly_order,
 double to_set_alpha, double to_set_markov_base_vol,
 double to_set_market_mean_rev_parameter,
 double to_set_norm_limit);

    /*
    void set_upper_barrier(double to_set);
void set_lower_barrier(double to_set);
void set_strike_rate(double to_set);
void set_cap_period(int to_set);
*/
void set_all_bc_parameters(double to_set_upper_barrier,
```

```
 double to_set_lower_barrier, double to_set_strike_rate,
        int to_set_cap_period);
double price();
void calculate_delta(double *delta);
void calculate_vega(double *vega);
//void calculate_initial_price();
double price_test();
void set_pertubation_interval_delta(double to_set);
void set_pertubation_interval_vega(double to_set);
protected:
double upper_barrier;
double lower_barrier;
double strike_rate;
int cap_period;
double initial_price;
bool calibrated;
bool initial_price_calculated;
double pertubation_interval_vega;
double pertubation_interval_delta;
};
```

bermudanSwaption.h

```
/*************************************************************
*bermudanSwaption.h
* The file calculate the price of a Bermudan swaption
* as well as the greeks using the Swap MF Model
*************************************************************/


#ifndef __included_MarkovFunctional__
#include <MarkovFunctional.h>
#define __included_MarkovFunctional__
#endif

#ifndef __included_SwapMarkovFunctional__
#include <SwapMarkovFunctional.h>
#define __included_SwapMarkovFunctional__
#endif


class bermudanSwaption: public SwapMarkovFunctional
{
public:
~bermudanSwaption();
bermudanSwaption();
bermudanSwaption(double to_set_fixed_rate,
 bool to_set_received_fixed,
 double to_set_pertubation_interval_delta,
 double to_set_pertubation_interval_vega,
 int to_set_num_discretization_points, int to_set_n,
 int to_set_poly_order, double to_set_alpha,
 double to_set_markov_base_vol,
 double to_set_markov_mean_rev_parameter,
 double to_set_norm_limit);
double price();
void calculate_delta(double *delta);
void calculate_vega(double *vega);
//void calculate_initial_price();
void set_delta_interval(double to_set);
void set_vega_interval(double to_set);
```

```
protected:
double fixed_rate;
double  initial_price;
bool receive_fixed;
double pertubation_interval_vega;
double pertubation_interval_delta;
double **bermudan_swaption;
};
```

capUsingSwapModel.h

```
/*************************************************************
* capUsingSwapModel.h
* The file calculate the price of a cap
* as well as the greeks using the Swap MF Model
*************************************************************/


#ifndef __included_MarkovFunctional__
#include <MarkovFunctional.h>
#define __included_MarkovFunctional__
#endif

#ifndef __included_SwapMarkovFunctional__
#include <SwapMarkovFunctional.h>
#define __included_LiborMarkovFunctional__
#endif


class capUsingSwapModel: public SwapMarkovFunctional
{
public:
~capUsingSwapModel();
capUsingSwapModel();
capUsingSwapModel(double to_set_fixed_rate,
 bool to_set_received_fixed,
 double to_set_pertubation_interval_delta,
 double to_set_pertubation_interval_vega,
 int to_set_num_discretization_points, int to_set_n,
 int to_set_poly_order, double to_set_alpha,
 double to_set_markov_base_vol,
 double to_set_markov_mean_rev_parameter,
 double to_set_market_base_vol,
 double to_set_market_mean_rev_parameter,
 double to_set_norm_limit);
double price();
void calculate_delta();
void calculate_vega();
void calculate_initial_price();
```

```
protected:
double fixed_rate;
double initial_price;
bool receive_fixed;
bool calibrated;
bool initial_price_calculated;
double pertubation_interval_vega;
double pertubation_interval_delta;
double **bermudan_swaption;
};
```

file.h

```
/*************************************************************
* A class to manage the files. The specific system calls and
* directory locations may need to be changed when porting
* the code to other systems.
*************************************************************/

//File class to manage all these files
class file
{
public:
file();
file(char *to_set_file_name, int to_set_column_size);
~file();
void write_line_to_file(double *x);
protected:
int column_size;
char *directory;
char *file_name;
char *full_file_name;
void clean();
void create();
};
```

gaussian.h

```
/***********************************************
* gaussian.h
* The file provides the cumulative normal and
* the inverse of the cumulative normal function
***********************************************/

double cumnorm(double z);
double cumnorm_inv(double p);
```

gaussianIntegrals.h

```
/*********************************************
* gaussianIntegrals.h
* Provides a class that integrates a polynomial
* with respect to a gaussian distribution from
* minus infinity to a specified upper limit
* need a high enough norm- limit. 7 does ok
*********************************************/

double calculate_gaussian_polynomial_integral(double *coefs,
 double upper_limit,  double mean, double sigma,
 int polynomial_degree);
double calculate_gaussian_polynomial_integral_by_segments(
 const int poly_order, const int num_discretization_points,
 double *x, double *y, double upper_bound,  double mean,
 double sigma);
```

liborDriverFunctions.h

```
/************************************************
* This file calculates the specific values and
* graphs presented in the paper for a cap and
* a barrier cap .
************************************************/

void fill_in_default_values(int &discretization_points,double
 &delta_interval, double &vega_interval, double &norm_limit,
 double &upper_barrier, double &lower_barrier, double &strike_rate,
 int &cap_period, int &n, int &poly_order,  double &alpha,
 double &markov_base_vol, double &markov_mean_rev);

 void calculate_bc_delta_different_pertubation_intervals(double jump_size);
 void calculate_bc_vega_different_pertubation_intervals(double jump_size);
 void calculate_bc_plain();
 void calculate_bc_different_mean_rev_parameters();
 void calculate_bc_different_discretization_points(double jump_size);
 void calculate_bc_standard_output();
 void test();
```

liborSwapDriverFunctions.h

```
/***********************************************
* This file calculates the specific values and
* graphs presented in the paper for a bermudan
* swaption calculated using the libor model.
***********************************************/

void fill_in_sul_default_values(double &fixed_rate, bool &receive_fixed,
 double &delta_interval, double &vega_interval,
 int &discretization_points, int &n, int &poly_order, double &alpha,
 double &markov_base_vol,
 double & markov_mean_rev_parameter, double &norm_limit);
void calculate_sul_mean_rev_params();
void calculate_sul_pertubation_interval(double jump_size);
void calculate_sul_disc_pts(int jump_size);
void calculate_sul_standard();
void calculate_sul_output();
```

outputFunctions.h

```
/*********************************************
* writes a vector to a file, separating elements
* with comma's. It adds a line to the file and
*  leaves the rest intact
*********************************************/

void write_line_to_file(double *x, int n,const char *myfile,
 int line_number);
```

polyFunctions.h

```
/*********************************************
* calculates the coefficients of a polynomial
* of the order of the number of elements in
* the given vector.
*********************************************/

void calculate_poly_coefs(double *x, double *y, double *coefs,
 int num_discretization_points);
```

swapDriverFunctions.h

```
/************************************************
* This file calculates the specific values and
* graphs presented in the paper for a bermudan
* swaption calculated using the swap model.
************************************************/

void fill_in_bs_default_values(double &fixed_rate, bool &receive_fixed,
 double &delta_interval, double &vega_interval,
 int &discretization_points, int &n, int &poly_order, double &alpha,
 double &markov_base_vol,
 double & markov_mean_rev_parameter, double &norm_limit);
void calculate_bs_mean_rev_params();
void calculate_bs_pertubation_interval(double jump_size);
void calculate_bs_disc_pts(double jump_size);
void calculate_bs_standard();
void calculate_all_bs_output();
```

swapUsingLibor.h

```
/*************************************************************************
*This file provides the framework and specific functions for calculating
* swap based derivatives and a bermudan swaption in the Libor MF model
*************************************************************************/


#ifndef __included_MarkovFunctional__
#include <MarkovFunctional.h>
#define __included_MarkovFunctional__
#endif

#ifndef __included_LiborMarkovFunctional__
#include <LiborMarkovFunctional.h>
#define __included_LiborMarkovFunctional__
#endif

class swapUsingLibor: public LiborMarkovFunctional
{
public:
~swapUsingLibor();
swapUsingLibor();
swapUsingLibor(double to_set_fixed_rate,
 bool to_set_receive_fixed, int to_set_num_discretization_points,
 int to_set_n, int to_set_poly_order, double to_set_alpha,
 double to_set_markov_base_vol,
 double to_set_markov_mean_rev_parameter,
 double to_set_delta_interval, double to_set_vega_interval,
 double to_set_norm_limit);
void print_swap_rate();
void calculate_delta(double *delta);
void calculate_vega(double *vega);
void set_delta_interval(double to_set);
void set_vega_interval(double to_set);
double price();
void test();

protected:
double **y;
double **PVBP;
```

```
double **swap_tree;
double ***discount;
void calculate_specific_discount(int time, int expiry);
void calculate_discount();
void calculate_PVBP();
void calculate_swap_rate();
void calculate_swap_tree(double fixed_rate,bool receive_fixed);
double fixed_rate;
bool receive_fixed;
bool calibrated;
bool swap_rate_calculated;
double pertubation_interval_delta;
double pertubation_interval_vega;
};
```