Dissertation

Pricing Bermudan Swaptions in the LIBOR Market Model

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Mathematical and Computational Finance

Steffen Hippler, Candidate Number 9202

June 2008

Contents

1 Introduction 2

2 Preliminaries 3
2.1 Bonds, LIBOR rates and Derivative Contracts 3
2.2 Change of Numeraire 7
2.3 Black's Formulas 8

3 The LIBOR Market Model 11
3.1 Model Set-Up 11
3.2 Pricing Approaches 13

4 Regression-Based Monte Carlo Methods 17
4.1 Dynamic Programming Formulation 17
4.2 Approximate Continuation Values 18
4.3 The Longstaff-Schwarz Algorithm 19

5 Calibration of the LIBOR Market Model 20
5.1 General Considerations 20
5.2 Functional Forms of Instantaneous Volatilities and Correlations 22
5.3 Calibration to Co-terminal European Swaptions 24

6 Numerical Experiments 26
6.1 Experimental Set-Up and Conduct 26
6.2 Discussion of Results 34

7 Conclusion 35

8 Appendix 37
8.1 Matlab Listings of the Experimental Section 37
8.1.1 Black's Formulas 37
8.1.2 Rebonato's Formula 38
8.1.3 Brigo-Mercurio Calibration Algorithm 39
8.1.4 LIBOR Rate Path Simulation 41
8.1.5 Longstaff-Schwarz Algorithm 43
8.1.6 Auxiliary Functions 44

1 Introduction

The LIBOR market model (LMM) belongs to the family of interest rate market models. The market models owe their popularity, at least partly, to the fact that they are consistent with well-established formulas for the pricing of simple interest rate derivative contracts. These contracts, namely caps and swaptions, are typically priced with Black's formulas that were developed - and have become the de facto market standard - well before the more advanced interest rate models were introduced, see, e.g., Brigo and Mercurio [1].

Historically, the LIBOR market model was preceded by a number of different interest rate modelling frameworks (for an overview of historic developments see Rebonato [9]). Simply put, the difference between the different frameworks lies in the interest rate being modelled. At first, this may seem a minor issue but it does have astonishingly far-reaching consequences. The fundamental entity modelled in the LIBOR market model is the forward interest rate curve that specifies the *simply* compounded interest rate at any two given points in time. A simply compounded interest of L between times T_1 and T_2 guarantees the investor of \$1 at time T_1 a return of $\$1 + L(T_2 - T_1)$ at time T_2. In contrast to the modelling of the entire interest rate curve are the simpler short rate models, which instead model the evolution of the spot rate. A consequence of this simpler modelling approach is that the forward rates are perfectly correlated and that these models are inconsistent with Black's formulas. The Heath-Jarrow-Morton (HJM) framework [2] is similar in notion to the LIBOR market model in that the entire forward rate curve is modelled. Here, however, the continuously compounding rate at any given point in time is modelled. As in the case of short-rate models, the HJM framework leads to interest rate dynamics that are inconsistent with Black's formulas.

Bermudan swaptions are interest rate derivatives with early exercise features that are among the most liquidly traded (exotic) interest rate derivative contracts. Consequently, their pricing and risk management is of high practical importance. The pricing of these instruments, however, poses significant conceptual and theoretical difficulties. This is due to the fact that, in general, the pricing in the LIBOR market model has to be carried out via Monte Carlo simulation techniques, which in turn do not lend themselves naturally to the pricing of options with early exercise features.

In this thesis, we consider the pricing of Bermudan swaptions in the LIBOR market model. We will review the underlying theory of the model and of recently developed methods for pricing early-exercise contracts in a Monte Carlo regime. We have implemented the pricing algorithm and have conducted numerical experiments. We will present the results of these experiments and will critically discuss our findings in the light of existing literature, particularly with respect to results obtained in Brigo and Mercurio [1] and in Lvov [7].

This work is organised as follows. In section 2, we will introduce the basic building blocks and derivative instruments in our market setting. We will derive Black's formulas for the pricing of caps and swaptions. In section 3, we will introduce the LIBOR market model and discuss the general approach of pricing derivative instruments in in this framework. Section 4 is devoted to a description of regression-based Monte Carlo methods that allow the pricing of early exercise contracts. As a special case of these methods, we will introduce and discuss in detail an implementation of the Longstaff-Schwarz algorithm in our LIBOR market setting. In section 5 we turn to the question of calibration of the model. This is a delicate task and we will give a brief overview and present a calibration approach suitable for the Bermudan swaption setting. Section 6 contains the results of our numerical experiments. We compare our results to those obtained in [1] and reflect on the various components of the pricing algorithm. We conclude in section 7.

2 Preliminaries

The definitions and theory presented in this section is a selection of material presented in [1].

2.1 Bonds, LIBOR rates and Derivative Contracts

The elementary building blocks in our setting are the zero-coupon bonds.

Definition 2.1: **Zero-coupon bond.** A T-maturity zero-coupon bond is a contract that guarantees its holder the payment of one unit of currency at time T with no intermediate payments. The contract value at time $t \leq T$ is denoted by $P(t, T)$.

The LIBOR (London Interbank Offered Rate) rate is a simply compounded interest rate that can be defined in terms of the values of zero-coupon bonds.

Definition 2.2: **LIBOR rate and spot LIBOR rate.** The LIBOR rate at time t, $L(t, T_{i-1}, T_i)$, is defined by

$$L(t, T_{i-1}, T_i) := -\frac{P(t, T_i) - P(t, T_{i-1})}{(T_i - T_{i-1})P(t, T_i)}. \tag{1}$$

We will use the shorthand $L_i(t) := L(t, T_{i-1}, T_i)$. The spot LIBOR rate at time t is given by

$$L(t, T) := L(t, t, T),$$

from which follows $L(t, T) = -\frac{P(t,T)-1}{(T-t)P(t,T)}$.

We have defined the LIBOR rates in terms of bond prices. Inverting this relationship, we can express bond prices in terms of LIBOR rates. The underlying financial idea is to invest at the current spot LIBOR and 'roll over' the investment at the subsequent LIBOR rates. This approach yields the following representation of the T_i-bond price at time $t < T_1$, a formula which we will frequently use later on,

$$P(t, T_i) = \prod_{j=1}^{i} \frac{1}{(L_j(t)(T_j - T_{j-1}) + 1)}. \tag{2}$$

As in Rebonato [9], we do not define a continuously compounding spot interest rate and corresponding bank account growing at this spot rate. For discounting purposes, we instead use the bond prices. A payout of one unit of currency at time T then has time-t value $P(t, T)$. We will discuss this further in the section 2.2.

With these definitions at hand, we are ready to define our derivative instruments. The first is an interest rate swap, which is a contract between two parties to exchange cash flows at a number of payment dates.

Definition 2.3: **Interest rate swap.** We are given a number of payment dates T_i, $i = \alpha + 1, \ldots, \beta$ (called the *tenor structure*) and a nominal value N

(the *notional*). We set $\delta_i := T_i - T_{i-1}$ in the following. The time-T_i *floating leg* of an *interest rate swap* with the given tenor structure has value

$$N\delta_i L_i(T_{i-1}),$$

which is exchanged for the *fixed leg* of value

$$N\delta_i K,$$

where K is a pre-specified *swap rate.* Note that the rate to be applied for the floating leg at time T_i, $L_i(T_{i-1})$, is fixed at the *reset time* T_{i-1}. If the fixed leg is payed and the floating leg is received, the contract is called a *payer swap*, in the opposite case it is called a *receiver swap.* The *time-t discounted payoff* of a payer swap can be expressed as

$$N \sum_{i=\alpha+1}^{\beta} P(t,T_i)\delta_i(L_i(T_{i-1}) - K).$$

The corresponding payoff for the receiver swap is obtained by changing the sign of the payer payoff, i.e. by a multiplication with -1.

The swap rate $S_{\alpha\beta}(t)$ that makes the payoff fair at time t, that is, equal to zero, can easily be obtained from the payoff definition and is given by

$$S_{\alpha\beta}(t) = \frac{P(t,T_\alpha) - P(t,T_\beta)}{\sum_{i=\alpha+1}^{\beta} \delta_i P(t,T_i)}. \tag{3}$$

The next derivative product we consider is called a swaption. A *payer swaption* gives its holder the right to enter an interest rate swap at time T_α at a pre-specified swap rate K.

Definition 2.4: **(European) payer swaption.** We are given a tenor structure T_i, $i = \alpha+1, \ldots, \beta$, a notional N, a swap rate K and a time t. The *payer swaption* is an option to enter into a swap at time T_α with swap rate K. It thus has time-t discounted payoff

$$P(t,T_\alpha)N\left(\sum_{i=\alpha+1}^{\beta} P(T_\alpha,T_i)\delta_i(L_i(T_\alpha) - K)\right)^+. \tag{4}$$

Expressed in terms of the swap rate defined in (3), the time-t discounted payoff of the swaption is

$$P(t,T_\alpha)N(S_{\alpha\beta}(T_\alpha)-K))^+\sum_{i=\alpha+1}^{\beta}\delta_i P(T_\alpha,T_i), \tag{5}$$

that is, it can be viewed as a call option on the swap rate.

Receiver swaptions can be defined in the obvious way. The swaption has limited optionality, namely the choice to enter the swap at time T_α. In contrast, a *Bermudan swaption* offers the possibility to enter the swap at any of the dates T_i, $i=\alpha,\dots,\beta-1$, for the remainder of the swap's lifetime.

Definition 2.5: **Bermudan payer swaption.** We are given a tenor structure T_i, $i=\alpha+1,\dots,\beta$, a notional N, a swap rate K and a time t. The *Bermudan payer swaption* is an option to enter at any time T_i, $i=\alpha,\dots,\beta-1$, into a payer swap with swap rate K maturing at time T_β. At any time T_k, $k\in\{\alpha,\dots,\beta-1\}$, the holder of the contract has the right to receive

$$N\left(\sum_{i=k+1}^{\beta}P(T_k,T_i)\delta_i(L_i(T_\alpha)-K)\right)^+,$$

provided that the option has not been exercised before.

In this definition, notice that the tenor of the underlying swap that can be entered shrinks with each exercise date T_i that is passed. This type of Bermudan swaption is known as *co-terminal* whereas in a *fixed maturity* Bermudan swaption, the tenor has a constant length. In terms of pricing and modelling, the fixed maturity Bermudan swaption does not pose any additional difficulties and so we restrict the exposition to the co-terminal case. Also note that we have assumed the dates at which the Bermudan swaption can be entered to be identical with the corresponding swap dates. The value of the Bermudan swaption will depend on the *exercise strategy* of the holder. We will later see that this additional feature makes the pricing of the Bermudan payer swaption significantly more difficult than that of the plain European swaption.

The final derivative product that we will be interested in is the interest rate cap, which gives its holder the right to pay a fixed rate instead of LIBOR at the tenor dates.

Definition 2.6: **Interest rate cap.** We are given a tenor sructure T_i, $i = \alpha+1, \dots, \beta$, a notional N, an interest rate K and a time t. A T_i-*caplet* is a call option on the T_{i-1}-spot LIBOR rate with strike K paying at time T_i. That is, it has time-t discounted payoff

$$P(t, T_i) N \delta_i (L_i(T_{i-1}) - K)^+.$$

A *cap* is a sum of caplets with different maturities, i.e. it has time-t discounted payoff

$$P(t, T_\alpha) N \sum_{i=\alpha+1}^{\beta} P(T_\alpha, T_i) \delta_i (L_i(T_{i-1}) - K)^+.$$

The name cap stems from the fact that this contract essentially caps the interest rate to be paid to K at any of the tenor dates. Notice the difference to the payoff of the swap (4). In the cap-case, the payoff at each of the dates T_i is non-negative whereas it can be negative for the swap underlying the swaption. In algebraic terms, the $()^+$-operator is placed around the individual payoffs in the cap-case whereas it acts on the sum of payouts in the swaption case. As a consequence, as we shall see later, the prices of caps are not influenced by correlations of the underlying LIBOR rates. In contrast, the correlations do matter in the case of swaptions.

2.2 Change of Numeraire

Before continuing, we briefly review the change of numeraire technique as it is a fundamental tool used throughout the following sections. We do not give an exhaustive account and refer to Shreve [11] for a more comprehensive treatment. While the definitions given in the first section were independent of the way the interest rate processes are modelled, from this section on we assume that the LIBOR rates underlying our derivative products are expressed in terms of diffusion processes. We begin with the definition of a numeraire.

Definition 2.7: **Numeraire.** A *numeraire* is any positive non-dividend-paying asset.

In an arbitrage free market, given a numeraire N, the first fundamental theorem of asset pricing [11] establishes that the time-t price, $X(t)$, of any T-measurable tradable asset X can be expressed as

$$X(t) = N(t)\mathbb{E}^N \left[\left. \frac{X(T)}{N(T)} \right| \mathcal{F}_t \right], \tag{6}$$

where $\mathbb{E}^N$ denotes the expectation taken with respect to the measure Q^N under which N-discounted asset prices are martingales. In the standard Black-Scholes setting, the numeraire corresponds to the bank account $B(t)$ growing at a continuously compounded spot interest rate r. That is, $B(t) = B(0)e^{\int_0^t r(s)ds}$ and the associated measure Q is called the *risk-neutral* measure.

We will find it useful to work with different numeraires (namely bonds maturing at times T_i) and thus will need to know the form of the pricing formula (6) under a different numeraire. This is established in the following proposition.

Proposition 2.8: **Change of numeraire.** We assume given a T-measurable tradable asset X and a numeraire N with associated probability measure Q^N such that the time-t price $X(t)$ is given by

$$X(t) = N(t)\mathbb{E}^N \left[\left. \frac{X(T)}{N(T)} \right| \mathcal{F}_t \right].$$

Let M be an arbitrary numeraire. Then there exists a probability measure Q^M equivalent to Q^N such that

$$X(t) = M(t)\mathbb{E}^M \left[\left. \frac{X(T)}{M(T)} \right| \mathcal{F}_t \right],$$

where $\mathbb{E}^M$ is the expectation taken with respect to the measure Q^M.

The essence of the proposition is that while we can simply 'plug in' our chosen numeraire, the corresponding expectation will have to be taken under a correspondingly adjusted measure.

2.3 Black's Formulas

In this section, we will discuss the pricing of caps and swaptions via Black's formulas. These formulas represent the 'market standard' for the valuation of caps and swaptions. A reason for the development of the LIBOR market model was that previous models were not consistent with Black's formulas in terms of the pricing of these plain vanilla instruments.

We start with the pricing of caps. In order to do so, we initially consider the prices of the individual caplets. The time-T_i payoff of the caplet is given by

$$N\delta_i(L_i(T_{i-1}) - K)^+.$$

Now using the bond maturing at time T_i as a numeraire and noting $P(T_i, T_i) = 1$, by proposition 2.8, the time-t caplet price can be expressed as

$$V_{capl_i}(t, L; K, N) = P(t, T_i)N\delta_i\mathbb{E}^i\left[(L_i(T_{i-1}) - K)^+ \middle| L_i(t) = L\right], \qquad (7)$$

where $\mathbb{E}^i$ is the expectation taken with respect to the measure Q^i under which $P(\cdot, T_i)$-discounted tradable assets are martingales. Here, $L_i(\cdot)$ is a tradable asset because from definition (1) it follows that

$$L_i(t)P(t, T_i) = \frac{P(t, T_{i-1}) - P(t, T_i)}{\delta_i},$$

and thus the left-hand side is tradable since the right-hand side is a sum of two tradables. As $L_i(\cdot)$ is a martingale under the measure Q^i, its dynamics are driftless and can be written as

$$dL_i(t) = \sigma_i(t)L_i(t)dW(t)$$

with a (deterministic) volatility function $\sigma_i(\cdot)$ and Brownian motion $W(\cdot)$. The process $L_i(\cdot)$ thus has a log-normal transition density under the measure Q^i and (following the Black-Scholes analysis) the value of the caplet is given by Black's formula,

$$V_{capl_i}^{Black}(t, L; K, N, \hat{\sigma}) = P(t, T_i)N\delta_i(\Phi(d_1) - K\Phi(d_2)), \qquad (8)$$

where

$$d_1 = \frac{\ln\frac{L}{K} + \frac{1}{2}\hat{\sigma}^2}{\hat{\sigma}},$$
$$d_2 = \frac{\ln\frac{L}{K} - \frac{1}{2}\hat{\sigma}^2}{\hat{\sigma}},$$

and where $\Phi(\cdot)$ denotes the cumulative normal distribution. Here, $\hat{\sigma}$ denotes the volatility of the rate $L_i(\cdot)$ accumulated in the time interval $[t, T_{i-1}]$, that is,

$$\hat{\sigma} = \sqrt{\int_t^{T_{i-1}} \sigma_i(s)^2 ds}.$$

This quantity can be retrieved from market quotes $\tilde{\sigma}$ (given as $\tilde{\sigma} = \hat{\sigma}/\sqrt{T_{i-1}}$) and thus uniquely determines the price of the caplet as all other parameters are observable.

The price of the cap can now be expressed as the sum of the prices of the corresponding caplets,

$$V_{cap}^{Black}(t, L; K, N, \hat{\sigma}) = N \sum_{i=\alpha+1}^{\beta} P(t, T_i)\delta_i \mathbb{E}^i \left[(L_i(T_{i-1}) - K)^+ \middle| L_i(t) = L_i \right], \tag{9}$$

where here L denotes a vector of the rates $L_i(\cdot)$ at time t, that is $L = (L_{\alpha+1}, \ldots, L_\beta)$ and $\hat{\sigma}$ is now a common volatility parameter to be used in the pricing formulas of all constituting caplets of the cap. Plugging in the caplet pricing formula (8), this yields an analytic formula for the price of the cap.

For swaptions, the approach just presented will not work as smoothly. As mentioned earlier, the swaption payoff (4) expressed in terms of the LIBOR rates cannot be decomposed in similar fashion as that of the caps due to the enclosing $()^+$-operator. Thus, we cannot value the individual payoffs $P(T_\alpha, T_i)N\delta_i(L_i(T_{i-1}) - K)$ under their *individual* numeraires but have to choose a single numeraire for evaluation of the entire expression (4). An alternative is to work with the payoff expression (5) in terms of the swap rate $S_{\alpha\beta}$, treating it as the underlying asset. Using this approach and following analogous steps as presented for the caps (using the expression $\sum_{i=\alpha+1}^{\beta} P(\cdot, T_i)\delta_i$ as a numeraire), it can be established that the time-t price of the swaption is given by

$$V_{swapt}^{Black}(t, S_{\alpha\beta}; K, N, \hat{\sigma}) = N(\Phi(d_1) - K\Phi(d_2)) \sum_{i=\alpha+1}^{\beta} \delta_i P(t, T_i), \tag{10}$$

where

$$d_1 = \frac{\ln \frac{S_{\alpha\beta}}{K} + \frac{1}{2}\hat{\sigma}^2}{\hat{\sigma}},$$
$$d_2 = \frac{\ln \frac{S_{\alpha\beta}}{K} - \frac{1}{2}\hat{\sigma}^2}{\hat{\sigma}},$$

and where $\hat{\sigma}$ denotes the root of the swap rate's variance accumulated on the time interval $[t, T_\alpha]$ (similarly defined as in the caplet case).

At this point it is in order to note that the two formulas just derived are not consistent with each other. A result of the approach taken for the caps was that the LIBOR rates were log-normal with respect to their corresponding numeraires. But the same argument holds for the swaption case thus resulting in a log-normal swap rate. However, as is shown in [1], when LIBOR rates are log-normal, the swap rates cannot be, and vice versa. Consequently, the two formulas are inconsistent with each other. In the LIBOR market model to be presented in the next section, we assume the LIBOR rates to be log-normal under the respective measures (and thus we are consistent with Black's formula for caps).

3 The LIBOR Market Model

The presentation of this section follows that of [1]. We present the dynamics of the LIBOR rates under different numeraires and proceed with a description of how the results can be used in practical pricing applications.

3.1 Model Set-Up

Let t be the current time and assume we are given a number of dates $T_0, \dots, T_M$ with corresponding *year fractions* $\delta_i := T_i - T_{i-1}$, for all $i = 1, \dots, M$, see [1] for details on year fractions. Consider the LIBOR rate $L_i(\cdot)$ in the time interval between t and T_i. By definition, the rate L_i is *alive* until time T_{i-1} at which it remains fixed at the spot rate $L_i(T_{i-1}) = L(T_{i-1}, T_i)$.

We derived in the previous section that under the measure Q^i associated with the bond $P(\cdot, T_i)$ maturing at time T_i, the rate $L_i(\cdot)$ is a martingale and thus follows the dynamics

$$dL_i(t) = \sigma_i(t)L_i(t)dW^i(t), \tag{11}$$

where W^i is a one-dimensional Brownian motion and $\sigma_i(\cdot)$ is a bounded deterministic function. Thus the stochastic differential equation (11) has a unique strong solution. We further assume that the Brownian motions W^i and W^j of different rates L_i and L_j are instantaneously correlated according to $\rho = (\rho_{ij})_{i,j=1,\dots,M}$, i.e.

$$dW^i(t)dW^j(t) = \rho_{ij}dt.$$

As we have seen when considering the pricing of swaptions, the payoffs of LIBOR derivatives in general cannot be decomposed with respect to the individual rates. To carry out pricing in terms of the LIBOR rates, we thus need to specify the dynamics under different numeraires and the associated measures. For the numeraires Q^i associated with bonds maturing at times T_i, the LIBOR rate dynamics are given in the following proposition. For a proof of this proposition, we refer to [1, 9].

Proposition 3.1: **Dynamics in the LIBOR market model.** Under the measure Q^j (associated with the bond $P(\cdot, T_j)$), the LIBOR rate $L_i(t)$ follows the following dynamics:

$$\begin{aligned}
\text{Case 1. } i < j, t < T_{j-1}: \quad dL_i(t) &= -\sigma_i(t)L_i(t)\sum_{k=i+1}^{j}\frac{\delta_j\rho_{ik}\sigma_k(t)L_k(t)}{1+\delta_j L_k(t)}dt \\
&\quad +\sigma_i(t)L_i(t)dW^i(t), \\
\text{Case 2. } i = j, t < T_{j-1}: \quad dL_i(t) &= \sigma_i(t)L_i(t)dW^i(t), \\
\text{Case 3. } i > j, t < T_{j-1}: \quad dL_i(t) &= \sigma_i(t)L_i(t)\sum_{k=i+1}^{j}\frac{\delta_j\rho_{ik}\sigma_k(t)L_k(t)}{1+\delta_j L_k(t)}dt \\
&\quad +\sigma_i(t)L_i(t)dW^i(t).
\end{aligned}$$

The equations admit strong solutions if the coefficients $\sigma_i(\cdot)$ are bounded (as assumed in our setting).

These dynamics constitute the *log-normal forward-LIBOR model*, where log-normal refers to the fact that for case 2, the distribution of L_i is log-normal for any fixed time t. However, this is not true for cases 1 and 3, in which the drifts are state-dependent and for a rate L_i contain all other rates L_k, $k = i+1, \ldots, j$ under the measure Q^j. The transition densities associated with these processes are not known. As a consequence, when simulating the value of a rate L_i at time t, we cannot use a 'one shot'-simulation according to the transition density but will instead need to explicitly construct a path via, e.g., the Euler-Maruyama method [3].

The dynamics of the model can be fully specified by fixing the instantaneous volatility functions $\sigma_i(\cdot)$ and the instantaneous correlations ρ_{ij}. In fact, these are the parameters that are used to calibrate the model. Note, however, that when trying to calibrate these parameters to a (finite) set of market data points (e.g. quoted caplet and swaption volatilities), the model will be highly under-specified so that additional 'reasonable' constraints on the $\sigma_i(\cdot)$ and ρ_{ij} will need to be made. We will come back to this issue when discussing the calibration of the model in section 5.

3.2 Pricing Approaches

When deriving Black's cap formula (9) in section 2.3, we have decomposed the cap into the individual caplets and priced each of these under their appropriate numeraire. The relevant LIBOR rate L_i for pricing of the time-T_i caplet then was log-normal and thus evolved according to the dynamics of case 2 in proposition 3.1. The LIBOR market model introduced thus is consistent with Black's formula for caps.

To price a swaption in the LIBOR market model, we take the expectation of the payoff defined in (4) under the measure associated with the bond $P(\cdot, T_\alpha)$ maturing at the swaption's maturity. We write

$$A(T_\alpha) = A(T_\alpha, L(T_\alpha); K, N) := N \left(\sum_{i=\alpha+1}^{\beta} P(T_\alpha, T_i)\delta_i(L_i(T_\alpha) - K) \right)^+ \quad (12)$$

for the payoff and thus obtain the formula

$$V_{swapt}^{LMM}(0, L; K, N) = P(0, T_\alpha) N \mathbb{E}^\alpha \left[A(T_\alpha) \middle| L_{\alpha+1}(0) = L_{\alpha+1}, \ldots, L_\beta(0) = L_\beta \right],$$

where $L = (L_{\alpha+1}, \ldots, L_\beta)$ denotes the vector of initial rates at time $t = 0$ and where the rates $L_i(\cdot)$ now evolve according to proposition 3.1. As mentioned earlier, these dynamics are not consistent with the assumption of a log-normal swap rate leading to Black's swaption formula (10), and thus V_{swapt}^{LMM} will in general be different from V_{swapt}^{Black}. However, the differences are generally small and can be remedied by appropriate calibration of the model. We shall also see this in the next sections.

To evaluate the expectation by simulation, we need to generate the rates

$$L_{\alpha+1}(T_\alpha), \ldots, L_\beta(T_\alpha),$$

according to the LMM dynamics in proposition 3.1. This can be done by the Euler-Maruyama discretisation scheme, in which, in this particular case, it is advantageous to evolve the logarithm of the rates L_i as this leads to the more accurate Milstein scheme, [1]. Assuming time steps of length Δt and evolving the log-rates, this yields the following scheme

$$\begin{aligned} \ln \hat{L}_i(t + \Delta t) = {} & \ln \hat{L}_i(t) + \sigma_i(t) \sum_{j=\alpha+1}^{i} \frac{\rho_{ij}\delta_j\sigma_j(t)\hat{F}_j(t)}{1 + \delta_j\hat{F}_j(t)}\Delta t \\ & - \frac{\sigma_i(t)^2}{2}\Delta t + \sigma_i(t)(\hat{W}_i(t + \Delta t) - \hat{W}_i(t)), \end{aligned} \tag{13}$$

for $i = \alpha + 1, \ldots, \beta$. Here $\hat{L}_i(t)$ denotes the approximation to $L_i(t)$ at time t and $\hat{W}(t + \Delta t) - \hat{W}(t) \sim \sqrt{\Delta t}\mathcal{N}(0, \rho)$ with $\hat{W}(t + \Delta t) - \hat{W}(t) = (\hat{W}_{\alpha+1}(t + \Delta t) - \hat{W}_{\alpha+1}(t), \ldots, \hat{W}_\beta(t + \Delta t) - \hat{W}_\beta(t))$ is an approximation vector to the multidimensional Brownian increment $W(t+\Delta t) - W(t)$ underlying the LIBOR rates.

Given a realisation of the rates at time T_α, we can evaluate the bond prices $P(T_\alpha, T_i)$ via (2) and then calculate (12). We repeat this procedure for a (large) number of path realisations and take the average. This yields the Monte Carlo swaption price.

When calibrating the model to market quoted swaption prices, we will need to quickly evaluate swaptions prices produced by the model to evaluate the model fit. In this case, it will not be practical to use the just described Monte Carlo pricing approach as it is too computationally time consuming.

There exists, however, an approximation formula for swaption volatilities in the LMM resulting from a specific parameter tuple $\sigma(\cdot)$ and ρ which is known as Rebonato's formula, [1, 9].

Proposition 3.2: **Rebonato's formula.** The Black-like LMM swaption volatility can be approximated by

$$\hat{\sigma}^{Rebonato}(0)^2 = \sum_{i,j=\alpha+1}^{\beta} \frac{w_i(0)w_j(0)L_i(0)L_j(0)\rho_{ij}}{S_{\alpha\beta}(0)^2} \int_0^{T_\alpha} \sigma_i(s)\sigma_j(s)ds,$$

where

$$w_i(0) := \frac{\delta_i \prod_{j=\alpha+1}^{i} \frac{1}{(1+\delta_j L_j(0))}}{\sum_{k=\alpha+1}^{\beta} \delta_k \prod_{j=\alpha+1}^{k} \frac{1}{(1+\delta_j L_j(0))}},$$

and $S_{\alpha\beta}(0)$ corresponds to the swap rate defined in (3).

For details of the derivation of the formula, we refer to [1].

The main advantage of Rebonato's formula is that the quantities $w_i(0)$ and $S_{\alpha\beta}(0)$ are defined in terms of the initial rates $L_i(0)$ at time $t = 0$ so that no simulation of paths is necessary ($S_{\alpha\beta}(0)$ is defined in terms of the bond prices $P(0, T_i)$ which can be recovered from the initial rates via (2)). Thus, instead of having to perform a computationally expensive Monte Carlo path simulation, we can simply evaluate Rebonato's formula and plug the resulting volatility $\hat{\sigma}^{Rebonato}(0)$ into Black's swaption formula (10) instead of $\hat{\sigma}$. This leads to an approximate price $V_{swapt}^{Rebonato}(0)$ of the swaption at time $t = 0$ under the LIBOR market model regime. We will use Rebonato's formula for calibration purposes later on.

We now consider the pricing of Bermudan swaptions. Provided that the Bermudan swaption has not yet been exercised, at a time T_i, $i < \beta$, the holder of the swaption has the right to receive

$$A(T_i) = N \left(\sum_{k=i+1}^{\beta} P(T_i, T_k)\delta_k(L_k(T_i) - K) \right)^+ . \tag{14}$$

Now, as the the maturity of the Bermudan swaption is β, the appropriate numeraire for pricing is the bond maturing at terminal date T_β. The measure associated with this numeraire is called the *terminal measure*. The rate $L_\beta(\cdot)$ corresponding to time T_β will thus evolve according to case 2 in proposition 3.1, while the other rates evolve according to case 1. With this numeraire, the time-0 value of the Bermudan swaption can be expressed as

$$\begin{aligned} V_{berm}^{LMM}(0, L; K, N) &= P(0, T_\beta) \\ &\times \sup_{\tau \in \mathcal{T}} \mathbb{E}^\beta \left[\frac{A(\tau)}{P(\tau, T_\beta)} \middle| L_{\alpha+1}(0) = L_{\alpha+1}, \ldots, L_\beta(0) = L_\beta \right], \quad (15) \end{aligned}$$

with our usual notation for the initial rate vector L. In this expression, the supremum is taken over all *stopping times* τ in a 'suitable' set $\mathcal{T}$. The requirement of optimising over stopping times ensures that only information available up until any time t is used in the decision-making (in particular, future values of the rates are not taken into consideration) and by 'suitable' we mean that the stopping time takes values at the discrete exercise dates T_i of the Bermudan swaption. For a more detailed treatment of stopping times, we refer to [11] and just mention here that a rational holder of the option will seek to maximise its value and correspondingly optimise the exercise strategy. For pricing, we thus take the supremum over all feasible stopping times in (15).

In evaluating the price (15) of the Bermudan swaption in the LIBOR market framework, we now face the problem that the LIBOR rates can be evaluated only via a Monte Carlo simulation which, in turn, does not lend itself to the pricing of options with early exercise features: In the Monte Carlo simulation, the LIBOR paths are generated in a forward fashion and thus, at any given point in time, the future values of a particular rate are not known. The future values, however, are important for making the exercise decision as they determine the future value of the contract. In contrast, backward-oriented methods, such as, e.g, binomial trees [3], start at the payoff date of the contract and work backwards in time thus making the relevant information available. These methods, however, cannot easily be applied in the current setting due to the (potentially) large number of underlying LIBOR rates and the resulting *curse of dimensionality*, [3]. Fortunately, there exist methods that allow pricing of early exercise options in a Monte Carlo regime by incorporating some ideas from the backward-oriented methods. We will address one of these in the next section.

4 Regression-Based Monte Carlo Methods

In this section, we draw upon the theory presented in several sources. The general set-up of regression-based Monte Carlo follows that developed in Glasserman [3]. However, we adapt the presentation and notation to the LIBOR market setting, making use of the specific structure of the Bermudan swaption pricing problem. In particular, we use some results from Piterbarg [8] to generate data to be used in the regression. We discuss the Longstaff-Schwarz algorithm developed in [6] which is a special case of the more general regression-based Monte Carlo approach. The presentation of the Longstaff-Schwarz algorithm given in [1] is on a more abstract level and in terms of evolving bond prices, thus strictly speaking not in terms of the LIBOR market model.

4.1 Dynamic Programming Formulation

The pricing formula (15) for the Bermudan swaption can be reformulated in terms of dynamic programming. We will present the underlying ideas in the following and refer to [3] for a more rigorous treatment. To ease notation, we write $V(L(t))$ for the value of a Bermudan swaption at time t given strike K, notional N and time-t vector of 'alive' rates $L(t)$, i.e. $V(L(t)) := V_{berm}^{LMM}(t, L(t); K, N)$.

Assuming no prior exercise, at a given exercise opportunity T_i for the Bermudan swaption, we have to make the decision of whether to continue holding the option or to exercise immediately. The exercise value at this time is simply the payoff of the swaption maturing at T_i with payoff date T_β that is given by $A(T_i)$ as in (14). On the other hand, the time-T_i value of holding the option beyond this point up until time T_{i+1} can then be expressed as

$$C(L(T_i)) := P(T_i, T_{i+1})\mathbb{E}\left[V(L(T_{i+1}))\middle|\, L(T_i)\right], \tag{16}$$

i.e. as the the discounted expected option value at time T_{i+1} given state $L(T_i)$. The expression $C(L(T_i))$ is called *continuation value* at time T_i given state $L(T_i)$.

The value of the Bermudan swaption $V(L(T_i))$ can now be expressed in terms of the continuation and exercise values via the following dynamic programming recursion

$$V(L(T_{\beta-1})) = P(T_{\beta-1}, T_\beta) N \delta_\beta (L(T_{\beta-1}) - K)^+, \quad (17)$$
$$V(L(T_j)) = \max\left[A(T_j), C(L(T_j))\right], \quad (18)$$

for $j = i, \ldots, \beta - 2$.

The dynamic programming approach is a backward method for valuing the option based on previously observed future states: We start the evaluation at time $T_{\beta-1}$ at which the value is simply the payoff of the final swaption and work backwards using the rule (18) until we reach the time T_i. However, if we knew the continuation value $C(\cdot)$ for each possible state $L(T_i)$ without 'coming from the future', we could use the approach in a forward fashion: At time T_i, we would exercise the option if $A(T_i) > C(L(T_i))$, otherwise continue holding it. Approximating the continuation value $C(\cdot)$ in a Monte Carlo setting for use in this manner is the topic of the next section.

4.2 Approximate Continuation Values

We want to approximate the continuation value $C(\cdot)$ defined in (16) so as to use it in a forward decision rule in a Monte Carlo simulation. To this end, we will use a regression approach: Given a number of m preselected basis functions, $\phi_l(\cdot)$, we seek to identify weights λ_{il} such that

$$C(L(T_i)) \approx \hat{C}(L(T_i)) := \sum_{l=1}^{m} \lambda_{il} \phi_l(L(T_i)). \quad (19)$$

It is evident that the quality of the approximation will depend on the choice of basis function. Obvious candidates in our setting are functions of the LIBOR rates $L_i(\cdot)$, the swap rate $S_{\alpha\beta}(\cdot)$ and the exercise values of the underlying European swaption $A(\cdot)$. We will further discuss the choice of basis functions in the experimental section.

To determine the weights λ_{il}, we use a least-squares estimation for which we need to generate a set of tuples $(L^k(T_i), P^k(T_i, T_{i+1}) V(L^k(T_{i+1})),\ k = 1, \ldots, n$. As pointed out in [8], this can be achieved as follows. We simulate n paths of each of our rates underlying the Bermudan swaption, the k-th path set indexed by k, i.e. $L^k_{\alpha+1}(T_i), \ldots, L^k_\beta(T_i),\ i = \alpha, \ldots, \beta - 1,\ k = 1, \ldots, n,$

from which we can encode at each time the corresponding state $L^k(T_i)$. Now, working backwards along an *individual* path k, we exactly determine the value $V(L^k(T_{i+1}))$ by looking into the path's future, i.e. by replacing the implicit taking of expectation in (18) by our knowledge of the path. At time $T_{\beta-1}$, we have $V(L^k(T_{\beta-1})) = P^k(T_{\beta-1}, T_\beta)N\delta_\beta(L^k(T_{\beta-1}) - K)^+$, and, recursively proceeding backwards along path k, at time T_i, we have

$$V(L^k(T_i)) = \max(P^k(T_i, T_{i+1})V(L^k(T_{i+1})), A^k(T_i)),$$

where the value $V(L^k(T_{i+1}))$ is known because we know the next state $L^k(T_{i+1})$ on the path and the bond price $P^k(T_i, T_{i+1})$ can be obtained via formula (2). Repeating this procedure for each path, we can thus generate n tuples $(L^k(T_i), P^k(T_i, T_{i+1})V(L^k(T_{i+1}))$ which are then used to determine the weights λ_{il} by straight-forward regression.

4.3 The Longstaff-Schwarz Algorithm

Given the construction of the approximate continuation values (19), we can now formulate the Longstaff-Schwarz algorithm for the Bermudan swaption. The pseudo-code is given as algorithm 1.

Several comments are in order regarding this algorithm. In line 2, we initialise a data structure 'Payoff' that will, for each path k, store the time-0 discounted exercise value of the option at time T_i in case the option is exercised at this time. Otherwise, it will contain a zero-entry. Thus, at the end of the algorithm, we just take the average of all discounted exercise values by adding all entries in 'Payoff' and dividing by n. This is in contrast to the presentation given in [1] and [6], where an exercise-flag ('1-exercise, 0-hold', at time T_i on path k) is stored that is used in a second pass through the paths to determine the cash-flows which are subsequently discounted and averaged. Our presentation is motivated by an implementation of the algorithm by Leippold [5].

Longstaff and Schwarz propose in their original paper to only select 'in-the-money-paths' for the regression approach. They argue that this allows a better estimate of the continuation value in the region where exercise is relevant. Moreover, this procedure improves the run-time performance of the algorithm as fever evaluations have to be made. The approach is reflected by the construction of the index subset I_i in line 7.

In contrast to a general regression-based algorithm as described in [3], the Longstaff-Schwarz approach uses the same paths for estimating the regression coefficients to define the approximate continuation value $\hat{C}(\cdot)$ as are used for the subsequent evaluation of the payoff according to the exercise rule. Despite the fact that for an individual path k, at time T_i the time T_{i+1}-value of the option along the path thus is known (see the discussion in the last section on generating the data points for the regression), for the actual decision-making in line 10 of the algorithm, the approximate continuation value is used. Since $\hat{C}(\cdot)$ produces potentially suboptimal decisions, the algorithm is low-biased. This, however, is desirable since in the more general version of the algorithm, the direction (low vs. high) of the bias is in general not known. For a detailed discussion of low and high bias, we refer to [3].

5 Calibration of the LIBOR Market Model

5.1 General Considerations

When discussing the LIBOR rate dynamics in proposition 3.1, we have already hinted at the fact that the model can be fully specified by choosing the instantaneous volatility and correlation functions, $\sigma(\cdot)$ and ρ. However, this is a delicate task and some available standard literature ([1, 9]) is dedicated in substantial parts to questions of calibration.

To motivate the difficulties, recall from the section 2.3 on Black's formulas that the market prices of, say, caplets are quoted in terms of the Black volatilities $\tilde{\sigma}_i$, from which the market participant's view on the volatility of a rate L_i accumulated until its maturity,

$$\hat{\sigma}_i^2 = \int_0^{T_{i-1}} \sigma_i(s)^2 ds,$$

can be recovered. Evidently, given only the left-hand-side of this equation, it is not possible to uniquely pin down the instantaneous volatility function $\sigma_i(\cdot)$ - in fact, this specification allows an infinite number of choices.

We have also argued earlier that the prices of swaptions depend on the correlations of the underlying LIBOR rates. These correlations, however,

Algorithm 1: Longstaff-Schwarz Algorithm for Bermudan swaption

```
   Data: An instance of the Bermudan swaption pricing problem, i.e. a
         parameter tuple $(0, L; K, N)$
   Result: Price of the Bermudan swaption $V_{berm}(0, L; K, N)$
 1 Choose $m$ basis functions $\phi_l(\cdot)$, $l = 1, \dots, m$;
 2 Initialise data structure Payoff$(\cdot, \cdot) \longleftarrow 0$;
 3 Simulate $n$ paths of the underlying LIBOR rates $L^k_{\alpha+1}(T_i), \dots, L^k_\beta(T_i)$,
   $i = \alpha, \dots, \beta - 1$, $k = 1, \dots, n$, according to the terminal measure;
 4 Generate $n$ tuples $(L^k(T_i), P^k(T_i, T_{i+1}) V(L^k(T_{i+1}))$, $k = 1, \dots, n$ using
   the procedure described in section 4.2;
 5 Initialise time index $i \longleftarrow \beta - 2$;
 6 while $T_i \geq T_\alpha$ do
 7     Consider only paths with strictly positive exercise value, set the
       corresponding index set $I_i := \{k \in \{1, \dots, n\} : A^k(T_i) > 0\}$;
 8     For $k \in I_i$, estimate weights $\lambda_{il}$ in approximate continuation value
       $\hat{C}(\cdot)$ (19) by regression of the $P^k(T_i, T_{i+1}) V(L^k(T_{i+1}))$ on $L^k(T_i)$;
 9     foreach $k \in I_i$ do
10         if $A^k(T_i) > \hat{C}(L^k(T_i))$ then
11             Store time-0 discounted exercise value:
               Payoff$(k, T_i) \longleftarrow P(0, T_i) A^k(T_i)$;
12             Set Payoff $(k, T_j) \longleftarrow 0$ for all $j > i$;
13     Reposition time index, $i \longleftarrow i - 1$;
14 return $\sum_{k,T_i}$ Payoff$(k, T_i)/n$;
```

while evidently depending on the corresponding instantaneous parameters ρ_{ij}, are not determined uniquely in terms of them. To be more precise, the instantaneous correlation ρ_{ij} between two rates L_i and L_j quantifies the degree of dependence between *changes* of the rates. Given a time t, the *terminal* covariance between $L_i(t)$ and $L_j(t)$ at time t, that is, the quantity relevant for pricing, is given by

$$\text{COV}(L_i(t), L_j(t)) = \rho_{ij} \int_0^t \sigma_i(s) \sigma_j(s) ds,$$

and thus is not only a function of the instantaneous correlation but of the corresponding volatilities as well, see [1] for a derivation. This quantity will depend on how the volatilities $\sigma_i(\cdot)$ and $\sigma_j(\cdot)$ are distributed over the time interval with respect to each other. For example, if σ_i tends to be low when σ_j is large, and vice versa, the terminal covariance will be lower than if the volatilities peak at similar points in time (this effect is called *de-correlation*, see, e.g. [9]).

A consequence of these observations is that the model is highly under-specified. Given a set of market volatilities, treating the volatilities and correlations as the primary model inputs without imposing further restrictions, we can perfectly fit the model to the given market setting. However, as [9] points out, following this approach leads to a model that loses all its predictive power, making it unsuitable for pricing and risk management purposes. Given the large number of degrees of freedom, the user of the model thus has to express views on the parameters that go beyond the information embedded in the market data. We cannot give an exhaustive overview of all factors influencing a reasonable expression of trading views and refer to [9] for a more detailed discussion. Suffice to say at this point that, given the absence of further information, it seems reasonable to assume that the volatility can be described as a time-homogenous function in that the only time-dependence arises through the time left to maturity.

5.2 Functional Forms of Instantaneous Volatilities and Correlations

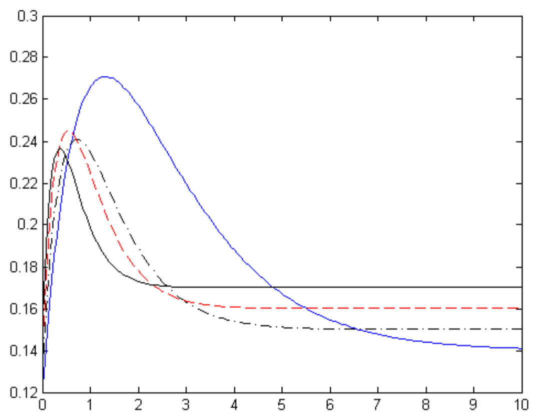
Proceeding, a time-homogenous functional form for the volatility $\sigma_i(t)$ that is widely suggested in the literature [1, 4, 9] is

$$\sigma_i(t) = v(T_{i-1} - t; \gamma) := ((T_{i-1} - t)\gamma_1 + \gamma_2) \exp^{-(T_{i-1}-t)\gamma_3} + \gamma_4,$$

with parameter $\gamma = (\gamma_1, ..., \gamma_4)$. This function has a 'humped' graph typical of that of, e.g., market caplet volatilities and can also be interpreted economically [9]. For a number of different parameter values, figure 1 shows the graphs of this function. To fit more closely to a given set of market data, the functional form can be 'enhanced' with a correction parameter so as to improve the fit, leading to a representation of the instantaneous volatility as

$$\sigma_i(t) = \psi_i v(T_{i-1} - t; \gamma). \tag{20}$$

Fig. 1: Instantaneous volatility functions $v(\cdot)$, for various parameters $\gamma_1, \dots, \gamma_4$, x-axis: time left to maturity $T_i - t$, y-axis: instantaneous volatility.



This is the functional form we will use in our experimental section.

In terms of the instantaneous correlations ρ, a parsimonious functional form proposed in [4, 9] is given by

$$\rho_{ij} = \exp(-\beta|i-j|), \tag{21}$$

with only one parameter β. A second, richer parameterisation due to Schoenmakers and Coffey [10], also examined in [1], is of the form

$$\begin{aligned}\rho_{ij} = \exp\Bigg[-\frac{|i-j|}{M-1} \\ \times \Bigg(-\ln\beta_3 + \beta_1 \frac{i^2+j^2+ij-3Mi-3Mj+3i+3j+2M^2-M-4}{(M-2)(M-3)} \\ -\beta_2 \frac{i^2+j^2+ij-Mi-Mj-3i-3j+3M^2-2}{(M-2)(M-3)}\Bigg)\Bigg],\end{aligned} \tag{22}$$

with fitting parameter $\beta = (\beta_1, \beta_2, \beta_3)$ and M denoting the total number of rates under consideration. Both of these correlation functions lead to positive-definite full-rank correlation matrices.

Reduced-rank correlations are discussed in detail in [1]. These have the advantage of being computationally more efficient. Furthermore, we note that an alternative approach to the one taken here is to exogenously specify the correlation matrix and only use the volatility parameters for calibration purposes.

5.3 Calibration to Co-terminal European Swaptions

Bermudan swaptions are typically hedged with the underlying co-terminal European swaptions. It therefore is desirable to calibrate the model to these market instruments so as to obtain consistency and allow for appropriate risk management, [9]. Here, we review a calibration routine presented in [1] for this purpose. We consider instantaneous volatility functions of the form $\psi_i v(T_{i-1} - t; \gamma)$ with parameters ψ and γ as introduced in the previous section. We assume the specific form of the instantaneous correlation is parameterised with parameter β as in the formulations (21) and (22).

The general idea of the calibration approach can be described as follows. Given market-quoted volatilities of European swaptions and initial parameters β that define the correlation matrix and γ that fix the volatilities, we identify the parameters ψ_i so as to match the market volatilities of the co-terminal swaptions maturing at the same time as our Bermudan swaption. Given the thus established parameters ψ_i, we in turn choose the parameters β and γ so as to minimise the squared differences of the volatilities $\hat{\sigma}$ derived from market data $\tilde{\sigma}$ and the volatilities implied by our model under the given parameters $\sigma(\beta, \gamma; \psi)$,

$$\min_{\beta,\gamma} |\hat{\sigma} - \sigma(\beta, \gamma; \psi)|^2.$$

In the optimisation, we use Rebonato's formula, proposition 3.2, for an efficient examination of the model volatilities and take into account the market-quotes for all swaptions maturing between the start and end date of our Bermudan swaption (that is, not only the co-terminal swaptions). With the new parameter estimates for β and γ, we restart the iteration and determine the parameters ψ_i. The entire procedure is repeated until a reasonable fit of the implied volatilities is achieved or a maximum number of iterations is exceeded. We will comment further on this in the experimental section.

This leaves us with the task of calculating the parameters ψ_i so as to match the co-terminal swaptions. This is done recursively starting with the co-terminal swaption maturing at the same time, T_β, as our Bermudan swaption. Given its market volatility $\tilde{\sigma} = \hat{\sigma}_\beta / \sqrt{T_{\beta-1}}$ and our volatility function (20), we have that

$$\tilde{\sigma}_\beta^2 = \frac{1}{T_{\beta-1}} \psi_\beta^2 \int_0^{T_{\beta-1}} v(T_{\beta-1} - s; \gamma)^2 ds,$$

from which follows

$$\psi_\beta = \sqrt{\frac{\tilde{\sigma}_\beta^2 T_{\beta-1}}{\int_0^{T_{\beta-1}} v(T_{\beta-1} - s; \gamma)^2 ds}}.$$

We next consider the swaption maturing at time $T_{\beta-2}$. Using Rebonato's formula, proposition 3.2, as an approximation for the volatility from our model and setting this equal to the market volatility, we obtain the relation

$$\begin{aligned} \tilde{\sigma}_{\beta-1}^2 = \frac{1}{T_{\beta-2}} \sum_{i,j=\beta-1}^{\beta} \frac{w_i(0) w_j(0) L_i(0) L_j(0) \rho_{ij}}{S_{\beta-2\beta}(0)^2} \psi_i \psi_j \\ \times \int_0^{T_{\beta-2}} v(T_j - s; \gamma) v(T_i - s; \gamma) ds, \end{aligned}$$

which yields

$$\frac{\tilde{\sigma}_{\beta-1}^2 S_{\beta-2\beta}(0)^2 T_{\beta-2}}{\int_0^{T_{\beta-2}} v(T_j - s; \gamma) v(T_i - s; \gamma) ds} = \sum_{i,j=\beta-1}^{\beta} w_i(0) w_j(0) L_i(0) L_j(0) \rho_{ij} \psi_i \psi_j.$$

Given that the parameter ψ_β is known at this point, the proceeding equation yields a second order polynomial in the unknown parameter $\psi_{\beta-1}$, which thus can, evidently, efficiently be identified. Then, given the parameter $\psi_{\beta-2}$, the procedure is repeated for the subsequent parameters. This yields a recursive procedure for the efficient identification of all parameters ψ_i. The details of the procedure are not complex but algebraically cumbersome and so we omit them here and refer to [1]. This is the calibration approach we use in the experimental section.

6 Numerical Experiments

6.1 Experimental Set-Up and Conduct

For our experiments, we use European swaption data provided in [1] to allow a comparison. The data comprises the implied market volatilities of European 'at the money'-swaptions shown in table 1 and initial LIBOR rates given in table 2. The swaption data table displays the volatilities for maturities from 1 to 10 years and corresponding tenor lengths of up to 10 years, i.e. in row '5 y' and column '3 y', the market volatility of a swaption with maturity in five years and underlying swaption with tenor length three years is given. The data given in columns 6, 8 and 9 has been generated from the the other columns via a log-linear interpolation (this procedure is suggested in [1] to fill the incomplete table provided there).

Tab. 1: At-the-money European swaption volatilities (in %).

	Maturities									
Tenors	1 y	2 y	3 y	4 y	5 y	6 y	7 y	8 y	9 y	10 y
1 y	25.2	21.8	19.1	17.3	15.9	14.7	13.9	13.1	12.8	**12.4**
2 y	23.5	20.1	17.9	16.3	15.0	14.0	13.3	12.5	**12.2**	11.8
3 y	21.4	18.7	16.8	15.3	14.2	13.2	12.6	**11.9**	11.7	11.3
4 y	19.4	17.4	15.7	14.4	13.4	12.6	**12.0**	11.4	11.1	10.8
5 y	18.0	16.3	14.7	13.5	12.7	**11.9**	11.4	10.8	10.6	10.3
6 y	16.8	15.3	13.8	13	**12.2**	11.6	11.2	10.6	10.4	10.1
7 y	15.9	14.6	13.4	**12.6**	12.0	11.5	11.1	10.6	10.4	10.1
8 y	15.1	14.0	**13.0**	12.4	11.8	11.3	10.9	10.4	10.3	10.0
9 y	14.5	**13.5**	12.8	12.1	11.6	11.1	10.8	10.3	10.2	9.9
10 y	**13.9**	13.2	12.5	11.9	11.5	11.1	10.8	10.3	10.2	9.9

We consider the instantaneous volatility and correlation parameterisations introduced in section 5.2. In addition to the fully parameterised volatility structure (20), we also consider the case of 'flat' volatilities, i.e. we assume $v = 1$ and fit solely via the ψ parameters. We consider all combinations of these functional forms, leading to four different 'formulations' which we will

Tab. 2: Initial forward rates (in %).

Expiry	L(0)
1 y	2.99
2 y	3.66
3 y	4.10
4 y	4.44
5 y	4.75
6 y	4.97
7 y	5.14
8 y	5.22
9 y	5.30
10 y	5.40

consider in our experiments. We name these in order of decreasing complexity starting with 'Formulation 1' with parameters $\gamma_1, ..., \gamma_4$ and $\beta_1, \beta_2, \beta_3$ up to the parsimonious 'Formulation 4'. Table 3 gives the details of our formulations.

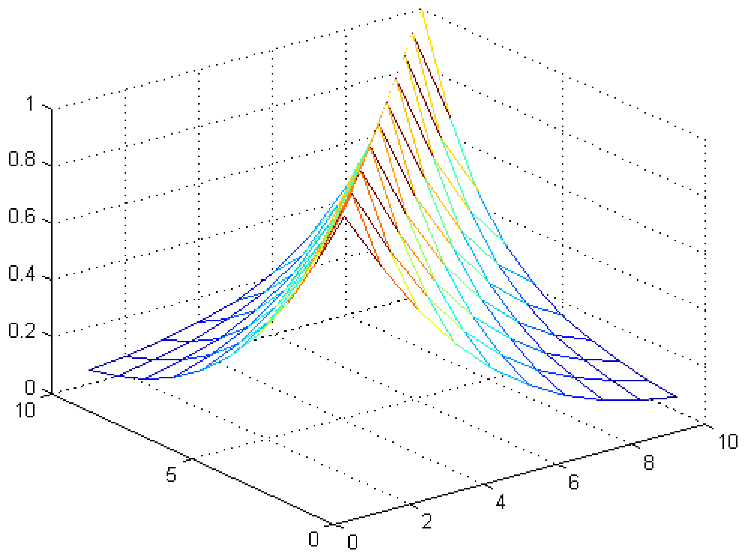
Tab. 3: Volatility and correlation structures used in the experiments.

	Correlation parameter	
Volatility parameter	$\beta_1, \beta_2, \beta_3$	β
ψ, $v = v(\cdot;\gamma)$	Formulation 1	Formulation 2
ψ, $v = 1$	Formulation 3	Formulation 4

Our aim is to price a Bermudan swaption with initial maturity at year $\alpha = 1$ and payout date $\beta = 11$ under our four volatility-correlation-parameterisations and at three different strike levels ('in the money - ITM', 'at-the-money - ATM' and 'out-of-the-money - OTM'). We proceed as follows.

First, we calibrate each of our four parameterisations to the data given in table 1 using the calibration procedure described in the previous section. We

Fig. 2: Calibration results: Instantaneous correlations.



calibrate to the co-terminal swaptions maturing at time 10, whose volatilities are given in bold in the table. These volatilities are fitted exactly by the calibration procedure and the volatility and correlation parameters are chosen such that the upper triangular part of the swaption matrix (above the bold diagonal) is fitted as closely as possible (the optimisation criterion being the sum of the squared errors). We limit the optimisation algorithm used to find the optimal parameters to a maximum of 500 iterations and choose starting values that correspond to a flat volatility curve. We give the relative errors of the calibration for our four formulations in tables 4 and 5, the underlying formula being the difference of the LMM and market volatilities divided by market volatilities. The code listing for the calibration can be found in 8.1.3.

As can be expected, the formulation with the highest number of parameters fits the data best. Errors of the calibration to this particular matrix are not given in [1], but the order of magnitude of the errors is consistent with that presented for other cases given there. Figure 2 shows the instantaneous correlation matrix obtained under formulation 1.

Tab. 4: Calibration results: Relative errors of LMM volatilities and swaption volatilities from table 1, (in %).

Formulation 1. Total sum of squared errors: 5.95

	Maturities								
Tenors	1 y	2 y	3 y	4 y	5 y	6 y	7 y	8 y	9 y
1 y	-0.83	3.84	1.19	1.59	-0.44	-1.39	-3.17	-3.84	-6.82
2 y	8.64	2.55	4.82	3.15	2.41	2.27	0.87	1.47	
3 y	-9.57	-1.20	-1.22	-0.46	0.30	1.52	1.38		
4 y	4.68	-1.00	-0.74	0.84	2.02	3.55			
5 y	-8.36	-7.54	-3.28	-0.19	1.37				
6 y	-7.13	-4.62	-0.19	1.09					
7 y	-2.86	-2.26	1.05						
8 y	-1.33	-0.13							
9 y	2.38								

Formulation 2. Total sum of squared errors: 7.02

	Maturities								
Tenors	1 y	2 y	3 y	4 y	5 y	6 y	7 y	8 y	9 y
1 y	-5.18	5.17	5.09	7.37	5.84	4.94	2.75	1.50	-2.42
2 y	5.06	1.24	4.90	3.49	2.56	1.89	-0.28	-0.70	
3 y	-11.5	-2.12	-2.22	-1.93	-1.94	-1.74	-3.09		
4 y	4.99	-1.27	-1.74	-1.15	-1.20	-1.19			
5 y	-6.97	-7.06	-3.92	-2.26	-2.44				
6 y	-4.01	-3.06	-0.32	-1.10					
7 y	1.80	-0.01	0.83						
8 y	4.02	1.81							
9 y	7.05								

Tab. 5: Calibration results: Relative errors of LMM volatilities and swaption volatilities from table 1, (in %).

Formulation 3. Total sum of squared errors: 9.56

	Maturities								
Tenors	1 y	2 y	3 y	4 y	5 y	6 y	7 y	8 y	9 y
1 y	-5.47	5.28	6.12	9.22	8.41	8.19	6.61	5.92	2.38
2 y	3.24	0.89	5.48	4.91	4.75	4.81	3.24	3.43	
3 y	-12.56	-2.48	-1.71	-0.58	0.19	1.11	0.39		
4 y	2.92	-1.94	-1.39	0.10	0.87	1.65			
5 y	-8.60	-7.54	-3.45	-0.89	-0.23				
6 y	-5.71	-3.55	0.22	0.41					
7 y	0.01	-0.42	1.56						
8 y	2.42	1.70							
9 y	5.80								

Formulation 4. Total sum of squared errors: 12.56

	Maturities								
Tenors	1 y	2 y	3 y	4 y	5 y	6 y	7 y	8 y	9 y
1 y	1.51	8.96	7.69	9.16	7.47	6.82	5.20	4.74	1.71
2 y	6.69	1.97	4.95	3.56	3.04	3.10	1.87	2.63	
3 y	-12.16	-3.65	-3.61	-2.77	-1.90	-0.55	-0.56		
4 y	0.97	-4.49	-4.13	-2.43	-1.12	0.52			
5 y	-11.94	-10.87	-6.52	-3.28	-1.58				
6 y	-10.06	-7.27	-2.67	-1.21					
7 y	-4.86	-3.87	-0.39						
8 y	-2.13	-0.70							
9 y	2.43								

Second, we use the calibrated model to reproduce the co-terminal European swaption prices. We do this for a notional of $N = 1$, strikes $K = 0.035$, 0.045 and 0.055 corresponding to ITM, ATM and OTM settings as in [1] and, as an example, the simplest calibration formulation 4. For the simulations, we generate 10,000 paths of the underlying forward rates and use a step-size of 0.5 years. The results of these computations are given in table 6, where the swaption prices are given in terms of basis points (i.e., one hundredth of 1%). We also priced the swaptions using Rebonato's formula with the given parameterisation, however, the results were indistinguishable from the exact prices and so we omit these figures here. The code for the path simulations and Rebonato's formula can be found in listings 8.1.4 and 8.1.2.

When comparing the prices obtained from our simulations (LMM) with those derived from Black's swaption formula (i.e. implying a log-normal swap rate), we see that the differences are actually minimal (in the range of a few basis points or a maximal deviation of around 1%). This is consistent with results obtained in [1, 9] and elsewhere in the literature and suggests that under the LIBOR market model, swap rates are nearly log-normal (refer also to our previous discussion of this point in section 2.3).

Finally, we have a calibrated model that we can use to price our Bermudan swaption. Under each calibration formulation, we now simulate 5,000 paths of the underlying LIBOR rates again using a step-size of 0.5 years. We use these simulated rates to calculate swap rates, exercise values and bond prices at each of the exercise time-steps of the Bermudan swaption. This data is subsequently used in the Longstaff-Schwarz-algorithm 1 to compute the time-α Bermudan swaption price. As basis functions for the algorithm, we use a constant, the swap rate S of the nearest-to-maturity swaption, the square and the cubic swap rate S. We repeat this procedure 100 times to compute the approximate Monte Carlo error in this setting. The results of our computations for the different strike levels and a notional of $N = 1,000$ are given in figure 7. The figure 3 shows a percentage ranking of the formulations in terms of prices generated (taking the lowest price in each setting as 100%). Figure 4 shows exemplary results of the approximate continuation value in the Longstaff-Schwarz algorithm. Code listings are given in 8.1.4 and 8.1.5.

Tab. 6: European co-terminal swaption prices under Black's formula and the LMM simulation (in basis points).

	3.5 (ITM)		4.5 (ATM)		5.5 (OTM)	
Tenors	Black	LMM	Black	LMM	Black	LMM
1 y	806.88	804.75	178.74	176.56	11.01	11.73
2 y	862.34	849.58	305.11	305.29	62.33	62.69
3 y	850.42	835.96	365.58	360.25	111.77	113.50
4 y	800.69	790.89	390.61	391.37	151.90	154.58
5 y	721.20	718.37	380.79	383.77	169.85	171.63
6 y	621.14	620.73	350.70	351.87	175.43	176.98
7 y	507.29	506.16	302.42	299.79	165.31	163.73
8 y	384.60	386.00	239.47	237.11	140.03	137.54
9 y	259.31	260.73	168.11	168.37	104.28	102.93
10 y	131.32	131.08	88.08	87.93	57.24	55.43

Tab. 7: Bermudan swaption prices and Monte Carlo errors (in brackets).

	Formulation			
Strike	1	2	3	4
3.5 (ITM)	91.17 (0.76)	91.50 (0.74)	90.81 (0.74)	90.69 (0.82)
4.5 (ATM)	48.19 (0.72)	48.79 (0.70)	48.09 (0.68)	47.81 (0.68)
5.5 (OTM)	25.26 (0.53)	25.34 (0.50)	24.94 (0.53)	24.57 (0.52)

Fig. 3: Bermudan swaption prices as percentages of the lowest price (100%) in each of the settings ITM, ATM and OTM. Bars correspond from left (red) to right (white) to the formulations 1 to 4.

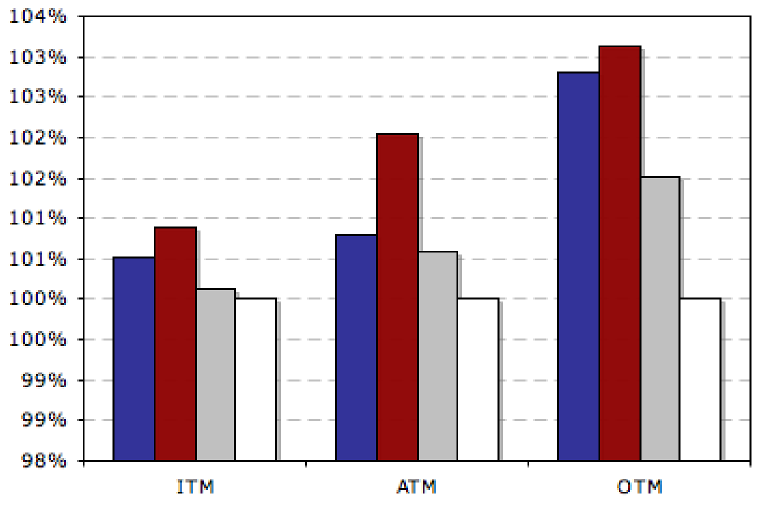
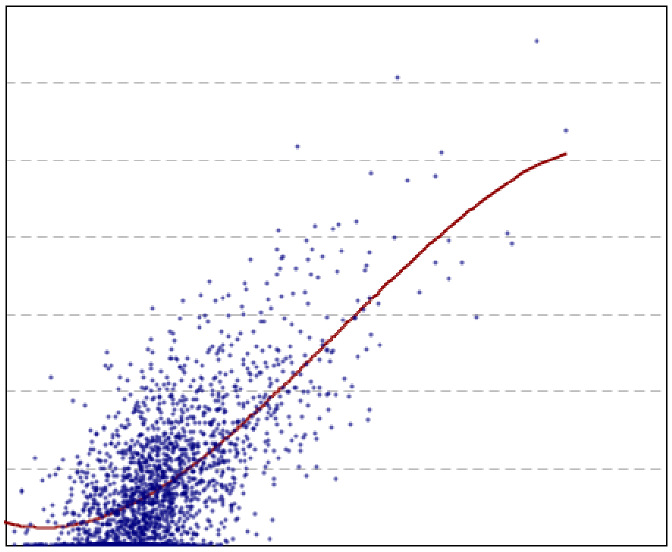


Fig. 4: Exemplary exact (blue dots) and approximate (red line) continuation value at one iteration of the Longstaff-Schwarz algorithm in our experiment. The x-axis corresponds to the swap-rate, the y-axis to the continuation value.



6.2 Discussion of Results

In a stylized setting, Rebonato [9] shows that Bermudan swaption prices can vary by up to 30% depending on the calibration method and parameterisation used. The differences in our prices, see figure 3, are much smaller than this, falling in the range of 0-3%. Our results allow an ordering of the parameterisations with formulation 2 generating the highest and formulation 4 the lowest prices in all three settings. Evidently, the more complex functional form (20) of the volatility is beneficial as formulations 1 and 2 outperform formulations 3 and 4 (that use 'flat' volatilities) in all cases. The situation is less clear cut in terms of the correlation parameterisation. In the case of formulations 3 and 4, the additional complexity in the correlation parameterisation is beneficial, in the case of the other two formulations it is not. It is also interesting to note that the best-performing formulation 2 has the lowest Monte Carlo error in two of the three cases considered. These results are somewhat in contrast to those presented in [1] - however, our formulations 2 and 4 are not included in their experiments. What can be said in addition to [1], who do not give information on the runtimes of their experiments, is that additional complexity in the formulations leads to significant increases in run-times of the calibration of the pricing algorithm. This is important as the run-time of the calibration routine is accountable for a significant share of the overall run-time and can be in the order of several minutes. The run-time of the calibration for formulation 2 was around 40% lower than that for formulation 1.

It is in order to note that our prices are consistently lower than the ones presented in [1], the differences being in the range of around 3-5%. While our results are not directly comparable to the ones in [1] as the authors do not explicitly comment on their specific implementation of the LIBOR rates simulation and do use pseudo-random (Sobol) numbers in their simulation, we attribute the differences to our varying implementation of the Longstaff-Schwarz algorithm. In fact, a conclusion of Lvov [7] is that the approximation errors made by evolving the LIBOR rates according to the discretisation (13) are small in comparison to errors resulting from using suboptimal basis functions in the Longstaff-Schwarz algorithm. Moreover, Lvov comes to the conclusion that the Longstaff-Schwarz algorithm using only the nearest-to-maturity swap rates (as in our implementation) instead of all 'alive' ones at any given time (as in [1]) yields an under-performance in the order of magnitude of 10 basis points, or 5%, in his setting. This is consistent with our results. To elaborate a bit further on this issue, consider the exemplary approximate continuation value based entirely on the swap rate of the nearest-to-maturity-swaption (see figure 4): Evidently, the approximation to the true continuation value is anything but perfect thus leading to suboptimal exercise decisions and eventually to prices that are too low.

For the pricing of Bermudan swaptions, is seems that (initial) implementation effort is spent more wisely on improving the Longstaff-Schwarz-part of an implementation than, for instance, on implementing more refined expressions for the discrete drift terms (e.g. the predictor-corrector-method, [4]) or the diffusion term (e.g., by replacing the discrete stochastic term in (13) with a more refined one based on integrated volatilities, see [1]).

7 Conclusion

We have considered the pricing of Bermudan swaptions in the LIBOR market model. We have presented the underlying theory and have conducted numerical experiments using a number of different variations of the pricing algorithm, comparing our results to those obtained elsewhere [1, 7] in the literature.

Our results indicate that the choice of the parameterisation of instantaneous volatility and correlation functions is an important one. In our - admittedly limited - experiments, the choice of a more realistic volatility function played a more transparent and important role than that of the corresponding correlation function. Too parsimonious a parameterisation turned out to produce clearly sub-optimal results, however, the quality of the results could not be improved by a simple adding of additional parameters. A successful implementation will thus need to strike a reasonable balance.

The issue of choosing ‘correct’ basis functions for the Longstaff-Schwarz algorithm appears to be even more important than that of the corresponding volatility and correlation parameterisations. Performance losses associated with sub-optimal basis functions were in the range of up to 5% compared with the maximum performance differential of 3% for the varying volatility and correlation parameterisations.

The two aforementioned choices appear to outweigh pricing errors made due to the discrete path approximations in the simulations. When pricing Bermudan swaptions in the LIBOR market model, particular care should thus be taken to improve the Longstaff-Schwarz and calibration parts of the algorithm prior to embarking on improving the simulation routine.

8 Appendix

8.1 Matlab Listings of the Experimental Section

To make this section more readable, we only list code fragments that contain the main 'implementation logic' and leave out parts for, e.g., data processing and output. Consequently, the code presented will not run stand-alone.

8.1.1 Black's Formulas

```
function V = BlackFormula(sigma,S,K,omega)

        d1 = (log(S) - log(K) + (0.5*sigma)) / (sqrt(sigma));
        d2 = (log(S) - log(K) - (0.5*sigma)) / (sqrt(sigma));

        V = (omega*S*N(omega*d1) - omega*K*N(omega*d2));
end

% Normal cumulative distribution function (Code by M. Giles)
function ncf = N(x)

        xr = real(x);
        xi = imag(x);

        ncf = 0.5*(1+erf(xr/sqrt(2))) ...
                      + i*xi.*exp(-0.5*xr.^2)/sqrt(2*pi);
end

function V = BlackSwaptionFormula(Strikes, Vols, Maturity, ...
                        BondPrice, numberOfSwaptionsPerTime)
        V=[1:1:numberOfSwaptionsPerTime];
        S=[1:1:numberOfSwaptionsPerTime];

        for k=1:numberOfSwaptionsPerTime
        denom=0;
        for j=1:k
           denom=denom+BondPrice(Maturity+j);
        end
        S(k)=(BondPrice(Maturity)-BondPrice(k+Maturity))/denom;
        V(k)=denom*BlackFormula(Maturity*...
        (Vols(Maturity,k)^2),S(k),Strikes(Maturity),1);
        end
end
```

8.1.2 Rebonato's Formula

```
function V =  RebonatoFormula(a, b, c, d, ...
                              psi, beta, maturities, rates, ...
                              maturity, tenorDate, horizon)

   CORR=CorrelationMatrix(beta, maturities, tenorDate-1, maturity, tenorDate);

   V=0;
   for indexI=maturity:tenorDate-1
       for indexJ=maturity:tenorDate-1
           test=quad(@vola,0,maturity);

            V=V+weights(indexI)*weights(indexJ)...
                *rates(indexI)*rates(indexJ)...
                *CORR(indexI,indexJ)...
                *quad(@vola,0,maturity);
       end
   end
   V=sqrt((1/maturity)*V*(1/swapRate^2));

   function v = vola(time)
        v1=VolFunc(a, b, c, d, psi(indexI), time, maturity);
        v2=VolFunc(a, b, c, d, psi(indexJ), time, maturity);
        v=v1.*v2;
   end

   function w = weights(index)
        helper= cumprod(1./(1+rates(maturity:tenorDate-1)));
        nom   = helper(index-maturity+1);
        denom = cumsum(helper);
        w     = nom./denom(tenorDate-maturity);
   end

   function S = swapRate
        S = rates(maturity:tenorDate-1)*weights(maturity:tenorDate-1)';
   end
end
```

8.1.3 Brigo-Mercurio Calibration Algorithm

```
function [beta, a, b, c, d, psi, LMMVol] =
            BrigoMercurioCoTerminalCallibration(maturity, maturities, ...
            rates, tenorDate, marketData, maxIterations)

    model = @volas;

    a=0;
    b=0;
    c=0;
    d=1;
    beta= [0.2 0.1 0.05];
    alpha=[a b c d];

    LMMVol=zeros(tenorDate-1);
    CompData=zeros(tenorDate-1);
    errors=zeros(tenorDate-1);
    sse=0;
    options = optimset('MaxIter',maxIterations);
    [result] = fminsearch(model, [beta, a, b, c, d], options);
    beta= result;

    psi =  CoTerminalVolaCallibration(alpha, beta, maturities, rates, ...
                          maturity, tenorDate, tenorDate+tenorDate, ...
                          marketData);

    function [sse] = volas(point)
        LMMVol=zeros(tenorDate-1);

        alpha=[point(4) point(5) point(6) point(7) ];
        psi =  CoTerminalVolaCallibration(alpha, point(1:3), maturities,
                          rates, maturity, tenorDate, tenorDate+tenorDate, ...
                          marketData);
        for i=1:tenorDate-1
            for j=1:tenorDate-i
                 LMMVol(i,j) = ...
                           RebonatoFormula(point(4), point(5), point(6),
                           point(7), psi, point(1:3), maturities, rates, ...
                           i, i+j, tenorDate+tenorDate);

                 CompData(i,j) = marketData(i,j);
            end
        end
        errors = LMMVol - CompData;
        sse = sum(sum(errors .^ 2));
    end
end


function psi =  CoTerminalVolaCallibration(alpha, beta, maturities, rates, ...
                                           maturity, tenorDate, horizon, ...
                                           marketData)

   CORR             = CorrelationMatrix(beta, maturities,...
                                        horizon, maturity, ...
                                        tenorDate+tenorDate-2);

   psi              = [1:1:tenorDate-1];
   f_star           = F_Star(maturity, tenorDate);

   for i=tenorDate-1:-1:1
       indexI = i;
       indexJ = i;
       c_one  =  f_star(i)^2*quad(@vola,0,i);
       c_two=0;
       for j=tenorDate-1:-1:i+1
            indexJ = j;
            c_two=c_two+...
                2*f_star(i)*f_star(j)*psi(j)*CORR(i,j)*quad(@vola,0,i);
       end
       c_three=0;
       for j=tenorDate-1:-1:i+1
           for k=tenorDate-1:-1:i+1
            indexI = j;
            indexJ = k;
            c_three=c_three+...
                f_star(j)*f_star(k)*psi(j)*psi(k)*...
                CORR(j,k)*quad(@vola,0,i);
           end
       end
       marketD=marketData(i,tenorDate-i);
       helper=cumsum(f_star(i:tenorDate-1));
       swapRate=helper(tenorDate-i);
       c_three = c_three - (i)*(marketD*swapRate)^2;

       p= c_two/c_one;
       q = c_three/c_one;
       root=sqrt((p/2)^2-q);
       psi(i) = -0.5*p + root;
   end
   function v = vola(time)
        v1  = volaFuncV(alpha, maturities(indexI), time);
        v2  = volaFuncV(alpha, maturities(indexJ), time);
        v   = v1.*v2;
   end

   function w = weights(index, maturity, tenorDate)
        helper= cumprod(1./(1+rates(maturity:tenorDate-1)));
        nom   = helper(index-maturity+1);
        denom = cumsum(helper);
        w     = nom./denom(tenorDate-maturity);
   end

   function F = F_Star(maturity, tenorDate)
        F = rates(maturity:tenorDate-1)....
        *weights(maturity:tenorDate-1, maturity, tenorDate);
   end
end
```

8.1.4 LIBOR Rate Path Simulation

```
function [exerciseVals, numeraireVals, swapRate, euroSwpations] =

                                    LMMMonteCarlo(a, b, c, d, K, beta, ...
                                    timeLine, maturities, stepSize, ...
                                    spotRate, initialRates, numberOfIterations, ...
                                    strikes, tenorLength, ...
                                    initialBondPrices)

   CORR=CorrelationMatrix(beta, maturities, tenorLength-1, 1, tenorLength-1);

   L = chol(CORR,'lower');
   X = randn(tenorLength-1, numberOfIterations*length(timeLine));
   Y = sqrt(stepSize)*L*X;

   start             = log(initialRates);
   trajectories      = zeros(tenorLength-1, length(timeLine));
   exerciseVals      = zeros(numberOfIterations, tenorLength-1);
   numeraireVals     = zeros(numberOfIterations, tenorLength-1);
   euroSwpations     = zeros(tenorLength-1, tenorLength-1);
   swapRate          = zeros(numberOfIterations, tenorLength-1);
```

```
for it=1:numberOfIterations
    currTime=0;
    trajectories(:,1)=start(1:tenorLength-1);
    for step=2:length(timeLine)
        for rate=tenorLength-1:-1:1

              %only consider rates that are still 'alive' at this point
              if currTime < maturities(rate)
                  currVol=...
                  VolFunc(a, b, c, d, K(rate), currTime, maturities(rate));

                  drift=0;

                  for j=tenorLength-2:-1:rate
                      volj=VolFunc(a, b, c, d, K(j), currTime, maturities(j));
                      %drift under the terminal measure
                      drift=drift-...
                          (CORR(rate,j)*volj*...
                           exp(trajectories(j,step-1)))/...
                           (1+exp(trajectories(j,step-1)));
                  end
                  trajectories(rate,step) = trajectories(rate,step-1)+...
                            currVol*(drift-0.5*currVol)*stepSize+...
                            currVol*Y(rate,step-1+(it-1)*length(timeLine));
              else
                  trajectories(rate,step) = trajectories(rate,step-1);
              end
        end
        currTime=currTime+stepSize;
    end

    timePoints           = (1:1:tenorLength-1);
    rates                = exp(trajectories(1:tenorLength-1, ...
                           1+(1/stepSize)*timePoints));
    spotRates            = [spotRate; diag(rates)];
    bonds                = 1./(1+spotRates(1:tenorLength-1));
    x                    = tril(BondPrices(-1, tril(rates)));
    denomY               = tril(BondPrices(-1, tril(rates)),-1);
    denom                = cumsum(denomY)';
    swapRate(it,:)       = (bonds - x(tenorLength-1, 1:tenorLength-1)')...
                          ./denom(1:tenorLength-1,tenorLength-1);
    ks                   = (sqrt(strikes)'*sqrt(strikes));
    exVal                = tril(rates-ks(1:tenorLength-1,1:tenorLength-1));
    exVal1               = x'.*exVal';
```

```
       exVal2              = cumsum(exVal1,2);

       %record exercise values of swaptions under terminal measure at all
       %points in time and record numeraire values
       exerciseVals(it,:)   = exVal2(1:tenorLength-1,tenorLength-1);
       numeraireVals(it,:)  = x(tenorLength-1,1:tenorLength-1);

       %calculate values of euro swaptions for testing purposes
       euroSwpations(:,:)   = euroSwpations(:,:)...
           +diag((1./x(tenorLength-1, 1:tenorLength-1)))...
           *max(zeros(tenorLength-1),exVal2(1:tenorLength-1,1:tenorLength-1));

   end
   euroSwpations            = initialBondPrices(tenorLength)...
                             *(1/(numberOfIterations))*euroSwpations(:,:);
end
```

8.1.5 Longstaff-Schwarz Algorithm

```
function [result] = LongstaffSchwarz(exerciseVal, swapRates, ...
numeraireVals, tenorLength, maturity, discountFactor)

Payoff=zeros(length(exerciseVal), tenorLength);
Payoff(:,tenorLength)=max(exerciseVal(:, tenorLength),0);

for nn = tenorLength-1:-1:maturity
    y = max(0, exerciseVal(:, nn));
    for i = 1:length(exerciseVal)
        if y(i) > 0
            ExVal = [ExVal; y(i)];
            X   = [X; swapRates(i, nn)];
            Y   = [Y; (numeraireVals(i,nn)/numeraireVals(i,nn+1))...
            *Payoff(i, nn+1)];
        end
        Payoff(i, nn)=(numeraireVals(i,nn)/...
        numeraireVals(i,nn+1))*Payoff(i, nn+1);
        Payoff(i, nn+1)=0;
    end

    %Regression
    A = [ones(size(yex)) X  X.*X  X.*X.*X] ;
    b(:,nn)=lscov(A,Y);
    Chat = A*b(:,nn);
    j = 1;
    for i = 1:length(exerciseVal)
        if y(i)>0
            if (ExVal(j) > Chat(j))
                Payoff(i,:) = 0;
                Payoff(i,nn) = yex(j);
            end
            j = j+1;
        end
    end
end
result = discountFactor*(sum(sum(Payoff))/length(exerciseVal));
end
```

8.1.6 Auxiliary Functions

```
function [vol] = VolFunc(a, b, c, d, psi, time, maturity)
    alpha=[a b c d];
    vol=psi*volaFuncV(alpha, maturity, time);
end

function v =  volaFuncV(alpha, time_k, time)
    tau = time_k-time;
    v   = (tau.*alpha(1)+alpha(2)).*exp(-tau.*alpha(3))+alpha(4);
end

function [corr] = CorrFunc(beta, horizon, i, j)
    M=horizon;
    corr=exp(-abs(i-j)/(M-1)...
             *(-log(beta(3)) ...
             +beta(1)*(i^2+j^2+i*j-3*M*i-3*M*j+3*i+3*j+2*M^2-M-4)...
             *1/((M-2)*(M-3))...
             -beta(2)*(i^2+j^2+i*j-M*i-M*j+3*i+3*j+3*M^2-2)...
             *1/((M-2)*(M-3))));
end

function CORR = CorrelationMatrix(beta, maturities, horizon, startRate, endRate)
   CORR=zeros(endRate-startRate);
   for i = startRate:endRate
        for j = startRate:endRate
            CORR(i,j) = CorrFunc(beta, horizon, maturities(j), maturities(i));
        end
   end
end
```

References

[1] D. Brigo and F. Mercurio, *Interest rate models, theory and practice*, Springer Science + Business Media, 2007.

[2] R. Jarrow D. Heath and A. Morton, *Bond pricing and the term structure of interest rates: A new methodology*, Econometrica (1992).

[3] P. Glasserman, *Monte carlo methods in financial engineering*, Springer Science + Business Media, 2004.

[4] M. Joshi, *The concepts and practice of mathematical finance*, Cambridge University Press, 2003.

[5] M. Leippold, *Matlab implementation of the longstaff-schwarz algorithm*, www3.imperial.ac.uk/people/m.leippold (2008).

[6] F. Longstaff and E. Schwarz, *Valuing american options by simulation: A simple least-squares approach*, The Review of Financial Studies (2001).

[7] D. Lvov, *Monte carlo methods for pricing and hedging: Applications to bermudan swaptions and convertible bonds*, PhD dissertation, ISMA Centre, University of Reading (2005).

[8] V. Piterbarg, *A practitioner's guide to pricing and hedging callable libor exotics in forward libor models*, Working paper (2005).

[9] R. Rebonato, *Modern pricing of interest rate derivatives: The libor market model and beyond*, Princeton University Press, 2002.

[10] J. Schoenmakers and C. Coffey, *Systematic generation of parametric correlation structures for the libor market model*, International Journal of Theoretical and Applied Finance (2002).

[11] S. Shreve, *Stochastic calculus for finance: Continuous time models*, Springer Science + Business Media, 2004.