VEGA CONTROL

NICK DENSON AND MARK JOSHI

Abstract. The calculation of prices and sensitivities of exotic interest rate derivatives in the LIBOR market model is often very time consuming. One approach that has been previously suggested is to use a Markov-functional model as a control variate for prices and deltas but not vegas. We present a new approach that is effective for prices, deltas and vegas. It achieves a standard error reduction by a factor of 10 for the price of five-factor, twenty-year Bermudan swaption, and of 5 for its vega.

1. Introduction

The LIBOR Market model (LMM) is almost considered the market standard interest rate model for pricing exotic interest rate derivatives. Its popularity is largely due to the fact that it can be calibrated to market-quoted caplet and swaption volatilities as well as the current discount curve. A common criticism of the model, however, is that it must be implemented via Monte Carlo and that this leads to slowness for both pricing and Greeks.

One solution often adopted by practitioners is to use the LMM for pricing and benchmarking, but to use a Markov-functional model for Greek computations. This is conceptually unappealing in that one would prefer to use the same model for prices and Greeks, and one would expect the Markov functional model's low-dimensionality to lead to errors.

A solution is to use a Markov-functional model as a control variate for the LMM for both pricing and Greeking. Piterbarg, [19], has previously suggested a methodology that works for prices and deltas but not vegas. Our objective here is to introduce a new methodology that is effective for all three. Whilst we introduce many innovations the essential improvement that allows vega computations is that our calibration of the Markov-functional control variate varies smoothly with the calibration of the LMM.

Date: May 4, 2009.

Key words and phrases. Variance reduction, control variate, LIBOR market model, LMM, BGM, Markov-functional model, vega.

We therefore compute prices and Greeks using two different implementations of the Markov-functional model: a lattice-based PDE; and a Monte Carlo which is highly correlated with the LMM. The PDE value is regarded as being precise and the difference is used to correct the LMM value.

Of course, there are other ways to construct control variates in the LMM; one could for example use simple contracts that can be analytically priced. The virtue of the approach discussed here is that it is product independent.

This paper is organised as follows. Section 2 briefly introduces the displaced-diffusion LIBOR market model and sets notation. In Section 3 the LIBOR Markov-functional model, which will be used as a control variate, is introduced. Section 4 presents details on how to use the model as a control variate within the LMM. Section 5 details the analytic calibration method for the Markov-functional model. In Sections 6 and 7, we present numerical results for the prices and model vegas of Bermudan swaptions.

2. Displaced-Diffusion LIBOR Market Model

The displaced-diffusion LIBOR market model is based on the idea of evolving discrete market observable forward/LIBOR rates. We have a set of tenor dates $0 < T_0 < T_1 < \ldots < T_n$ and n corresponding forward rates $f_0, f_1, \ldots, f_{n-1}$, where $f_i(t)$ represents the forward rate at time t for the interval $[T_i, T_{i+1})$. We set $\tau_i = T_{i+1} - T_i$.

Let $P(t,T)$ denote the price at time t of the zero coupon bond paying one at its maturity T. Throughout this paper we will work in the *terminal measure*, which corresponds to taking $P(t,T_n)$ as numeraire. This simplifies lattice-based pricing since the value of the numeraire is not path-dependent, and only depends on the values of the prevailing forward rates at any points in the lattice.

In the displaced-diffusion LIBOR market model the forward rates are assumed to have the following evolution

$$\frac{df_i(t)}{f_i(t)+\alpha_i} = \mu_i(f,t)dt + \sigma_i(t)dW(t), \tag{2.1}$$

where the σ_i's are deterministic D- $(1 \leq D \leq n-1)$ dimensional row vectors, the α_i's are constant displacement coefficients, W is a standard D-dimensional Brownian motion and the drift term under the terminal

measure is given by

$$\mu_i(f,t) = -\sum_{j=i+1}^{n-1} \frac{(f_j(t)+\alpha_j)\tau_j}{1+f_j(t)\tau_j}\sigma_i(t)\sigma_j(t)^T. \tag{2.2}$$

For more details on the LMM we refer the reader to the textbooks [4], [5] or [12].

The LMM cannot be represented as a low-dimensional Markov process. There are two reasons. The first is that the drift in equation (2.2) is state dependent. The second is that even if we set all drifts to zero, the time-dependence of the volatility loadings will cause the set of attainable forward-rate curves to be high-dimensional. For these reasons, it is usual to implement the LMM using Monte Carlo. However, if we modify the LMM appropriately, we can obtain a Markov-functional model.

3. Markov-Functional Model

There are many LIBOR Markov models in the literature, three common approaches are the LIBOR Markov-functional model introduced by [14] (see also [10] and [17]), the Cheyette model [6], and the separable LIBOR Markov-functional model [18].

We will focus on the F-factor separable LIBOR Markov-functional model, because it is similar to the LMM and will work well as a control variate. Our aim is to have a model such that all forward rates are determined by a small number (F) of Markov factors. The model can then be used to price products using a lattice method, which will be faster than Monte Carlo in low dimensions.

To obtain the separable LIBOR Markov-functional model we start with the displaced-diffusion LMM and make a restriction and modification. The restriction is that the volatility structure is *separable*. The modification is that the forward rates are evolved to each time in a single step from time zero. With this we have a Markov-functional model, which when calibrated, gives the vector of forward rates as a function of the F Markov factors.

3.1. F-factor model.

The separable volatility structure we use is

$$\sigma_i(t) = \nu_i C(t), \tag{3.1}$$

where ν_i is an F-dimensional row vector and $C(t)$ is an $F \times F$ matrix. The volatility structure gets its name because the forward rate and time dependence are separated out.

Our separable volatility structure is more general than those currently in the literature. Consider a two-factor model. The volatility structure we use is

$$\sigma_i(t) = \begin{bmatrix} \nu_{i,1} & \nu_{i,2} \end{bmatrix} \begin{bmatrix} c_{11}(t) & c_{12}(t) \\ c_{21}(t) & c_{22}(t) \end{bmatrix}. \tag{3.2}$$

This volatility form differs from Piterbarg's [19] since he sets

$$\sigma_i(t) = \begin{bmatrix} \nu_{i,1} & \nu_{i,2} \end{bmatrix} \begin{bmatrix} c_{11}(t) & c_{12}(t) \\ 0 & c_{22}(t) \end{bmatrix}.$$

Clearly, incorporating an extra parameter should allow for a better fit. It is likely that Piterbarg set $c_{21}(t) = 0$ to reduce the number of variables in his numerical calibration since he carried out a global-least-squares optimisation. However, our new analytic calibration method works equally well with the extra term, so we use a more general form.

Using our separable volatility structure, we define the F-dimensional vector of Markov factors

$$dX(t) = C(t)dW(t),$$

where $W(t)$ is the projection of the Brownian motion in (2.1) onto the first F coordinates. If we can consider a LMM driven only by the first F factors then its dynamics can be written as

$$\frac{df_i(t)}{f_i(t) + \alpha_i} = \mu_i(t)dt + \nu_i dX(t), \tag{3.3}$$

and so if the drifts were dropped the forward rates would be Markovian in $X(t)$.

Our Markov-functional model is to take this equation and define the forward rates at time t as a function of $X(t)$ to be the values that would be obtained by evolving from 0 to t using a single-step discretization.

3.2. **Drift approximation in the Markov-functional model.** Since we are using only a single step to evolve forward rates to each time, we need to use a highly-accurate drift approximation; the time-step may be as long as 20 years.

One approach is simply to freeze the drifts at the start of the time step that is to use a log-Euler drift approximation

$$\mu_i(f(t), t) = \mu_i(f(0), t).$$

and this appears to be the technique used by Piterbarg [19]. However, for the LMM it is well-known that there are much better approximations, (see [13]). We use the iterative variant of the predictor-corrector method introduced in [11]; we shall refer to this method as IPC. It was shown in [13] that this method is both computationally fast and

accurate. The method involves taking a log-Euler step for each forward rate, recalculating the drift, then taking the average of the two drifts. We correct the drift of one rate at a time in an iterative way, working backwards. If we are evolving the rates from time 0 to time T we have

$$\mu_i = -\frac{1}{2}\sum_{j=i+1}^{n-1}\left(\frac{(f_j(0)+\alpha_j)\tau_j}{1+f_j(0)\tau_j}+\frac{(\hat{f}_j+\alpha_j)\tau_j}{1+\hat{f}_j\tau_j}\right)\int_0^T \sigma_i(t)\sigma_j(t)^T dt, \tag{3.4}$$

where $f_j(0)$ represents the initial forward rates and $\hat{f}_j$ represents the evolved forward rates using IPC. This can be done because of the triangular nature of the dependence of the drifts on the rates.

3.3. **PDE pricing.** Using the Markov-functional model we can derive a PDE for the deflated price of a derivative, V. Using Ito's lemma and the fact that the deflated option price must be a martingale we obtain

$$-\frac{\partial V}{\partial t} = PV, \tag{3.5}$$

where P is the partial differential operator

$$P = \sum_{i,j=1}^{F} q_{ij}(t)\partial_i\partial_j,$$

with $\partial_i = \frac{\partial}{\partial x_i}$, where x_i represents the Markov factors. For the coefficients, let $c_{ij}(t)$ be the element in the ith row of the jth column of $C(t)$ in (3.1), and then

$$q_{ij}(t) = \frac{1}{2}\sum_{f=1}^{F} c_{if}(t)c_{jf}(t).$$

This PDE can be numerically solved to price callable LIBOR exotic derivatives. Note that path dependent derivatives can be handled using the auxiliary variable technique, see [12].

4. Markov-functional model as a control variate

The Markov-functional model can be applied as a control variate for the displaced-diffusion LMM. We price a product in the LMM and then price the same product in the Markov-functional model using both a lattice and Monte Carlo implementation. If the two models produce highly correlated prices then the control price will have a reduced variance.

A product V is priced using

$$V_{control} = V_{LMM} - \theta(V_{MC} - V_{PDE}),$$

where V_{MC} and V_{PDE} represent the deflated price in the Markov-functional model implemented via a Monte Carlo and PDE method respectively, and θ is a constant value. Using different numerical implementation methods should still arrive at the same price, so the expected price is equal to that from the LMM

$$\mathbb{E}[V_{control}] = \mathbb{E}[V_{LMM}].$$

To see why the control variate works, consider the price's variance

$$\mathrm{Var}[V_{control}] = \mathrm{Var}[V_{LMM}] + \theta^2\mathrm{Var}[V_{MC}] - 2\theta\mathrm{Cov}[V_{LMM}, V_{MC}]. \quad (4.1)$$

The control price will give lower variance than the LMM price if

$$\theta^2\mathrm{Var}[V_{MC}] < 2\theta\mathrm{Cov}[V_{LMM}, V_{MC}].$$

The variance reduction depends on how correlated the prices between the two models are. The only differences between the LMM and the Markov-functional model, apart from the number of driving factors, is the volatility and drift restrictions. If we closely match the volatility structures, the two models will produce highly correlated prices and therefore considerable variance reduction.

The value of θ which minimises (4.1) is

$$\theta = \frac{\mathrm{Cov}[V_{LMM}, V_{MC}]}{\mathrm{Var}[V_{MC}]}.$$

To estimate θ we could run a small number of simulations, calculate it using the above formula and then run a set of independent simulations to price. Since our control is very similar to the model by construction, we simply take $\theta = 1$, which saves the time taken to run the extra simulations. This is justified by our estimates of θ for all examples considered in this paper, see Appendix A. For more details on control variates see Section 4.1 of [9].

4.1. **Monte Carlo Implementation.** The Markov-functional model implementation needs to be as close as possible to that of the LMM. This means the same random numbers must be used for the first two factors when evolving rates.

To price an early exercisable product an exercise strategy is needed and here there are a few choices. We could:

- calculate the strategy for the Markov-functional Monte Carlo implementation
- use the exercise strategy derived from the lattice method

- use the strategy derived from the LMM, or
- use the same set of exercise times as the LMM.

There is a subtle difference between the last two choices. Applying the exercise strategy, in the form of a set of functions to compute the continuation value, may result in a slightly different exercise time between the two models. The variance reduction will be maximised when a product is exercised at the same time in both models as the cash flow amounts will be similar. Therefore we recommend applying the LMM exercise times to the Markov-functional model.

4.2. **Lattice Implementation.** The PDE in equation (3.5) needs to be numerically solved using an accurate and efficient method. There are many different ways to do this and so we do not list them here. The method we use is to solve the large system of stiff ODEs, following [16], that arise after spatial discretisation of the PDE. We chose this method because of its stability and accuracy. Note that time spent solving the PDE directly reduces the time savings of the control variate method, so an efficient scheme needs to be used.

Similarly with the Markov-functional Monte Carlo implementation we need to choose an exercise strategy. To ensure the price is not biased we need

$$\mathbb{E}(V_{MC}) = \mathbb{E}(V_{PDE}),$$

which will hold if the same exercise strategy is used for both methods. However, applying the same exercise time from the LMM to the lattice method is nonsensical. As an alternative, we apply the exercise strategy from the LMM to the lattice method. This slight difference will result in a bias, however, our results show this bias to be very small.

4.3. **LMM Exercise Strategy.** To calculate the price of an early exercisable product we need to find the exercise strategy for the LMM. One method that works consistently well is to use least-squares regression (LS) with the Andersen method [1] superimposed as suggested in [3]. The continuation value at each exercise date, G_i is estimated by performing a least-squares regression against a chosen set of basis functions. Using this continuation value the product is exercised into the value E_i if

$$G_i + H_i \leq E_i,$$

where we perform a one dimensional optimisation at each exercise date to find the best H_i. As the one dimensional optimisations are performed quickly the method is almost as fast as LS, but with the benefit of the optimisation picking up some of the value missed by the chosen

set of basis functions. We use the exercise strategy with a new set of independent pricing paths to avoid an upwardly biased price.

5. Calibration

The effectiveness of this control variate technique rests on how well the Markov-functional model is calibrated to the LMM. We need to ensure that the Markov-functional model's volatility structure is as close as possible to that of the LMM's volatility structure. We do not make any assumptions on the calibration of the original LMM; our objective is to find a separable calibration that is close by and which varies smoothly with it.

We will take $C(t)$ to be constant over each interval $[T_{i-1}, T_i)$. Let $C(i)$ represent the matrix of constant values over that interval, with the convention that $T_{-1} = 0$. For an F-dimensional Markov-functional model and a product that depends on n forward rates, there are $Fn + F^2n$ parameters to be fitted: the Fn forward rate specific scalars and F^2n piecewise-constant time-dependent values.

A calibration of the LMM is a choice of pseudo-square root for each time step in the simulation. Since different pseudo-square roots can yield the same covariance matrix, it will be useful in what follows to make a specific choice. In particular, we assume that the pseudo-square root has been computed via spectral decomposition. In particular, if the covariance matrix, for step i has eigenvalues λ_j in decreasing order, with corresponding eigenvectors e_j, then we take

$$b_j = \lambda_j^{0.5} e_j$$

to be column j of our pseudo-square root, B^i. This maximizes the impact of the first F factors. We will denote the pseudo-square roots arising from our separable fit as A^i.

One possible methodology, as in Piterbarg [19], is to use a global least-squares numerical optimisation, however, there is no guarantee in that case that the matrix A^i varies smoothly with B^i. Here we present a calibration method that is both analytic and stable. It has the features of using spectral methods to fit one matrix and inductively fitting the other matrices one by one.

For the remainder of this section we shall address the problem of fitting an F-dimensional Markov-functional model to a D-dimensional LMM, where $F \leq D$. Naturally, we will only attempt to fit the first F columns the matrices B^i. However, the fact that we are using the pseudo-square root defined by spectral decomposition means that these will be the dominant factors.

5.1. **Analytic Calibration.** Consider the evolution of the forward rates over $[0, T_0)$, with the corresponding pseudo-square roots B^0 and A^0. By looking at equations (2.1) and (3.3) we see that if we have

$$dX(t) = dW(t),$$

by an appropriate choice of C values, then we can exactly calibrate the first F-columns of A^0 to B^0, simply by setting

$$\nu_j = \left[\begin{array}{ccc} b^0_{j,1} & \dots & b^0_{j,F} \end{array} \right],$$

for $j = 0, \dots, n-1$. with

$$C(0) = \left[\begin{array}{cccc} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{array} \right]. \tag{5.1}$$

Let V denote the matrix with columns ν_j (which is equal to A^0.)

Now that we have set all of the forward rate specific scalars by exactly fitting to the first F columns of the first pseudo-square root, we need to fit the remaining $n-1$ pseudo-square roots. Our remaining freedom lies in our ability to pick the matrices $C(i)$.

For each i, we want choose $C(i)$ to minimize the sum-of-squares norm on the difference

$$VC(i) - B^i.$$

Since the k-th column of $C(i)$ only contributes to the kth column of $VC(i)$, we have a separate least-squares optimization problem for each k and i. If we let b denote the target column and let c denote the relevant column of $C(i)$ we have to minimize

$$Vc - b,$$

where V is an $D \times F$ matrix, c is an F-vector and b is a D-vector. This can be solved using standard linear algebra techniques for solving an over-determined system. Since it is low-dimensional linear algebra, the calibration is performed effectively instantaneously within a computer program.

We could also perform the calibration starting with any evolution interval. Consider the evolution over $[T_{i-1}, T_i)$. We exactly fit the first F-columns of B^i by setting

$$\nu_j = \left[\begin{array}{ccc} b^i_{j,1} & \dots & b^i_{j,F} \end{array} \right],$$

with $C(i)$ equal to the identity matrix (note that some elements of ν_j will temporarily be set to zero as elements of B^i are zero due to

reset forward rates). We perform a least squares fit to the $n-i-1$ pseudo-square roots after the evolution time T_i.

To fit the pseudo-square roots before T_{i-1} we work backwards considering one evolution interval at a time. We compute the matrix $C(i-1)$ by solving the system of equations, as above, including ν_j for $j = i, \ldots, n-1$. Once $C(i-1)$ has been computed we set ν_{i-1} so that the first F columns of the $i-1$th row of B^{i-1} are exactly matched. We repeat, working backwards until all pseudo-square roots have been fitted. This approach gives the flexibility of exactly fitting the first F-columns of a particular pseudo-square root.

5.2. **Calibration Accuracy.** We measure the calibration accuracy using the statistic

$$\frac{\sqrt{\sum_{i,j,k}(b^i_{j,k})^2} - \sqrt{\sum_{i,j,k}(b^i_{j,k} - a^i_{j,k})^2}}{\sqrt{\sum_{i,j,k}(b^i_{j,k})^2}} \tag{5.2}$$

which captures the percentage of the factor loadings reproduced.

We consider calibrating to pseudo-square roots generated by taking the spectral decomposition of covariance matrices calculated using the *abcd* volatility form

$$\sigma_i(t) = [a + b(T_i - t)]e^{c(T_i - t)} + d, \tag{5.3}$$

for constants, a, b, c and d. The correlation between each forward rate is given by

$$\rho_{ij} = L + (1-L)e^{-\beta|T_i - T_j|}.$$

We consider fitting a two-factor Markov-functional model to two-, and five-factor LMMs (see Section 14.11 of [12] for details on how to factor reduce.) We also vary the volatilities. The volatility scenarios are defined as *abcd*β1 ($a = 0, b = 0, c = 0, d = 0.2, L = 0.5, \beta = 0.1$) representing flat volatility, *abcd*β2 ($a = 0.05, b = 0.09, c = 0.44, d = 0.12, L = 0.5, \beta = 0.1$) representing a long lasting hump in the caplet volatilities, *abcd*β3 ($a = -0.02, b = 0.3, c = 2, d = 0.15, L = 0.5, \beta = 0.1$) representing a volatility hump peaking at about 1-year, and *abcd*β4 ($a = -0.02, b = 0.3, c = 2, d = 0.15, L = 0.5, \beta = 0.2$) representing a low correlation scenario.

We fit the pseudo-square roots for different tenor products, all with semi-annual rates. For example, a product with a tenor of 5Y requires 10 pseudo-square roots.

Tables 5.1 and 5.2 present the calibration fit for the worst and best choice of pseudo-square root to exactly fit the first 2 columns of, for two- and five-factor LMMs. We see that the calibration works very well.

tenor	$abcd\beta 1$	$abcd\beta 2$	$abcd\beta 3$	$abcd\beta 4$
5Y	100-100 %	98-99 %	92-95 %	92-95 %
7Y	100-100 %	96-98 %	91-96 %	91-96 %
10Y	100-100 %	93-96 %	90-96 %	90-96 %
20Y	100-100 %	85-92 %	83-95 %	84-95 %

Table 5.1. *Calibration accuracy range for a variety of volatility scenarios using the statistic (5.2) with a two-factor LMM.*

tenor	$abcd\beta 1$	$abcd\beta 2$	$abcd\beta 3$	$abcd\beta 4$
5Y	88-89 %	87-88 %	86-87 %	82-83 %
7Y	86-87 %	84-86 %	84-86 %	80-81 %
10Y	84-84 %	81-84 %	82-84 %	77-79 %
20Y	78-79 %	73-78 %	77-79 %	70-72 %

Table 5.2. *Calibration accuracy range for a variety of volatility scenarios using the statistic (5.2) with a five-factor LMM.*

tenor	$abcd\beta 1$	$abcd\beta 2$	$abcd\beta 3$	$abcd\beta 4$
5Y	-	8-1	7-0	7-0
7Y	-	11-1	11-0	12-0
10Y	-	15-2	17-1	18-1
20Y	-	30-2	23-4	23-4

Table 5.3. *Pseudo-square roots exactly fitted corresponding to the two-factor calibration in Table 5.1*

A perfect fit is achieved for the two-factor constant volatility (because it is separable) and even a fit of over 70% is achieved for a 20Y five-factor product. Overall, the calibration accuracy is greater than 80% for all but a few cases, indicating that the Markov-functional model will be an effective control variate.

Table 5.3 shows the corresponding pseudo-square roots exactly fitted in Table 5.1, where for example, the top right entry 7-0 means that the worst and best fit came from choosing the pseudo-square root over the time step $[T_6, T_7)$ and $[0, T_0)$ respectively. We see that the choice of pseudo-square root exactly fitted does affect the accuracy of the overall calibration. The pseudo-square roots fitted for the five-factor model are not displayed as the pattern is similar to the two-factor model. In any

case, the calibration method is very fast so it is possible to iterate over all possible choices and select the best within a fraction of a second.

6. Pricing

In this section we present prices and their standard errors using a two-factor Markov-functional model as a control variate, before moving onto the powerful application of model vegas in the following section. We consider the prices of at-the-money payer Bermudan swaptions with a variety of maturity dates. All forward rates are assumed to be semi-annual and flat, calculated using a continuously compounded rate of 5%, i.e. $P(t,T) = e^{-r(T-t)}$. We use the volatility structure described above as *abcd*β2 as this represents what we consider a fairly difficult market scenario. All displacements are equal to zero, i.e. $\alpha_i = 0 \ \forall \ i$.

All Bermudan swaptions start in 6-months time and can be exercised on any of the reset dates. To calculate the exercise strategy we use the least-squares algorithm with the Andersen method superimposed as discussed in Section 4.3. We use all quadratic polynomials in the intrinsic swaption value and the swap rate as basis functions. A Sobol quasi-random number generator with 2^{14} paths was used to determine the exercise strategy and then 10000 Mersenne Twister paths were used to estimate the price and standard error. As suggested above, we optimized the fitting procedure to get the best overall fit.

		2 factor			3 factor			5 factor		
5Y	LMM	226	(2.66)		220	(2.62)		219	(2.60)	
	Control	225	(0.01)	183x	221	(0.07)	39x	220	(0.08)	31x
7Y	LMM	330	(3.88)		329	(3.74)		327	(3.69)	
	Control	329	(0.04)	108x	332	(0.13)	29x	328	(0.15)	25x
10Y	LMM	479	(5.63)		480	(5.73)		476	(5.44)	
	Control	481	(0.10)	53x	482	(0.25)	23x	477	(0.29)	19x
20Y	LMM	852	(9.58)		850	(9.65)		839	(9.80)	
	Control	853	(0.48)	20x	853	(0.86)	11x	838	(0.97)	10x

Table 6.1. *Prices and standard errors in basis points for different tenor Bermudan swaptions. 2-, 3- and 5-factor LMMs were used with the control variate.*

Table 6.1 shows prices, standard errors and the multiple of standard error reduction using the control variate. The control variate works very well, for a 5Y Bermudan swaption a standard error reduction of 31 to 183 times is achieved. The method is also effective for longer dated products; we achieved a standard error reduction of 10 times for

a five-factor 20Y Bermudan swaption, which is an order of 100 times speed up.

The speed up will be smaller than this in practice due to time spent calculating the Markov-functional PDE and Monte Carlo prices, however, with an efficient numerical scheme to calculate the PDE price the time savings will still be very significant. We did not focus on finding an efficient numerical scheme to solve the PDE equation (3.5), so we do not present timings here.

In line with expectations the control variate works best for shorter-dated products and when coupled with a low-factor LMM. Somewhat surprising is the control variate's effectiveness for longer-dated products, which is due to the accurate calibration method.

7. Vega

Pricing, while time consuming, is not as computationally intensive as Greek computations. If we are using a finite difference approach, then for each sensitivity we will need to run an extra set of simulations with the perturbed parameters. Given that the number of sensitivities required can be large, we need an efficient way of computing prices, otherwise running many pricing paths may simply be too time consuming to be practical. We note that there are more sophisticated approaches than finite difference, for example, [8] or [7].

The method of calculating vega that we consider here is to perturb the volatility in the LMM and then map this model vega to a market vega, for a discussion of how to map the model vega to a market vega see [19].

To calculate the vega we compute the price using the control variate method, then perturb the LMM volatility (e.g. d in the case of the *abcd* volatility structure used here), recalibrate our Markov-functional model (which is performed very quickly) and then compute the new price. The crucial fact that we use here is that the eigenvalues and eigenvectors of a matrix are analytic functions of its entries, [2], provided the eigenvalues are distinct. The constraint of distinctness will generally be fulfilled since the first eigenvalue is generally dominant when we consider correlation matrices for forward rates, see for example [15].

When bumping we use the same set of random numbers for both simulations. We also use the same set of exercise times to avoid artificial trigger effects as discussed in [19].

Denoting the ith unperturbed and perturbed prices as $V_i(x)$ and $V_i(x+h)$ respectively, the statistic we are interested in is

$$\text{model vega} = \frac{1}{N}\sum_{i=1}^{N}(V_i(x+h) - V_i(x)),$$

and its standard error, for N simulations. We could have used a central difference above, but instead we chose a one-sided difference as traders are often interested in what happens to their portfolio when volatilities increase by 1%.

7.1. **Alternative Calibration.** We briefly discuss an alternate approach for calculating vegas. Assume we have used the calibration technique in Section 5 to calibrate the set of unperturbed pseudo-square root matrices A_i to B_i for $i = 0, \ldots, n-1$. To calibrate the set of perturbed pseudo-square root matrices A_i^* to those from the LMM, B_i^*, we could use the same technique, that is we minimise the sum of squares differences between the elements of A_i^* and B_i^*. Alternatively, we could find B_i^* such that

$$[(A_i^* - B_i^*) - (A_i - B_i)]^2 \to \min, \tag{7.1}$$

so that the differences between the unperturbed and perturbed separable matrices is as similar as possible to the differences between the unperturbed and perturbed non-separable matrices. Finding the matrices A_i^* is very similar to Section 5 so we do not present the details.

This calibration technique is appealing because the vega is driven by the differences in the pseudo-square root matrices. Making these differences similar should cause the LMM and Markov-functional model to produce correlated vegas, therefore resulting in useful standard error reductions. In our testing, however, we found that minimising equation (7.1) produced similar or worse results than using the calibration method of Section 5 (see Appendix B), so we do not use it.

7.2. **Vega results.** We present results for the computed model vega for the corresponding Bermudan swaption prices in Table 6.1. Three different vega scenarios were considered, which are presented separately in Tables 7.1, 7.2 and 7.3. The presentation of the results is the same as before with the estimate, standard error and multiple of standard error reduction shown.

Table 7.1 shows the vega estimate when d in the *abcd* volatility structure was perturbed by 1%. Once again we have used 2^{14} Sobol paths to determine the exercise strategy and 10000 Mersenne Twister paths to determine both the unperturbed and perturbed prices. We exactly fit to the pseudo-square root that produces the best global fit. Focusing on

		2 factor			3 factor			5 factor		
5Y	LMM	10.8	(1.5E-01)		10.6	(1.5E-01)		10.5	(1.4E-01)	
	Control	10.8	(1.9E-03)	80x	10.5	(9.3E-03)	16x	10.7	(5.6E-03)	26x
7Y	LMM	16.1	(2.3E-01)		16.1	(2.2E-01)		16.0	(2.2E-01)	
	Control	16.2	(8.8E-03)	26x	16.3	(1.1E-02)	21x	16.1	(1.1E-02)	20x
10Y	LMM	24.4	(3.7E-01)		24.9	(3.8E-01)		24.2	(3.6E-01)	
	Control	24.5	(1.7E-02)	21x	25.0	(2.9E-02)	13x	24.4	(3.0E-02)	12x
20Y	LMM	46.1	(7.7E-01)		46.2	(7.8E-01)		45.5	(8.2E-01)	
	Control	46.2	(9.6E-02)	8x	46.3	(1.1E-01)	7x	45.6	(1.7E-01)	5x

Table 7.1. *Model vegas and standard errors in basis points for the Bermudan swaptions priced in Table 6.1. All caplet volatilities were increased by 1%.*

the second row of the table, for example, we see that for a 7Y Bermudan swaption the standard error reduction for the vega calculation ranges from 20 to 26 times.

Instead of calculating the vega when every caplet's volatility is increased by 1%, in Table 7.2 we perturbed three caplet volatilities in the middle of the product's term structure (e.g. caplets resetting at times T_4, T_5 and T_6 for a 5Y product) by 1%. We perturbed the pseudo-square root matrices directly by multiplying elements such that the caplet volatilities increased by 1%. We notice that the standard error reduction is less than the case where all caplet volatilies were perturbed, but is still significant.

		2 factor			3 factor			5 factor		
5Y	LMM	4.3	(5.9E-02)		4.3	(6.0E-02)		4.3	(6.0E-02)	
	Control	4.4	(8.3E-04)	71x	4.3	(9.3E-03)	6x	4.2	(1.1E-02)	5x
7Y	LMM	4.5	(6.5E-02)		4.6	(6.6E-02)		4.6	(7.0E-02)	
	Control	4.6	(1.4E-03)	47x	4.7	(1.3E-02)	5x	4.7	(1.6E-02)	4x
10Y	LMM	4.7	(6.7E-02)		4.7	(7.0E-02)		4.8	(7.1E-02)	
	Control	4.8	(2.8E-03)	24x	4.7	(1.8E-02)	4x	5.0	(2.1E-02)	3x
20Y	LMM	3.9	(5.8E-02)		4.0	(6.4E-02)		4.1	(6.9E-02)	
	Control	3.7	(6.3E-03)	9x	4.1	(2.5E-02)	3x	4.1	(3.2E-02)	2x

Table 7.2. *Model vegas and standard errors in basis points for the Bermudan swaptions priced in Table 6.1. Three caplet volatilities in the middle of the product's term structure were perturbed.*

Table 7.3 considers the case where just one caplet volatility was perturbed. We see that even for a small change in the pseudo-square roots useful standard error reductions are achieved, for example, 4 to 40 times for a 7Y Bermudan swaption.

		2 factor			3 factor			5 factor		
5Y	LMM	1.4	(2.0E-02)		1.4	(2.0E-02)		1.4	(2.1E-02)	
	Control	1.5	(3.2E-04)	62x	1.5	(3.7E-03)	5x	1.5	(4.8E-03)	4x
7Y	LMM	1.5	(2.2E-02)		1.5	(2.2E-02)		1.5	(2.4E-02)	
	Control	1.6	(5.5E-04)	40x	1.6	(4.9E-03)	5x	1.6	(6.3E-03)	4x
10Y	LMM	1.6	(2.3E-02)		1.6	(2.4E-02)		1.6	(2.4E-02)	
	Control	1.6	(1.1E-03)	21x	1.7	(6.3E-03)	4x	1.7	(7.8E-03)	3x
20Y	LMM	1.3	(2.0E-02)		1.4	(2.1E-02)		1.4	(2.3E-02)	
	Control	1.3	(2.3E-03)	8x	1.4	(8.6E-03)	2x	1.4	(1.1E-02)	2x

Table 7.3. *Model vegas and standard errors in basis points for the Bermudan swaptions priced in Table 6.1. The middle caplet volatilities were perturbed i.e. those corresponding to the forward rates resetting at times T_5, T_7, T_{10} and T_{20} for the 5Y, 7Y, 10Y and 20Y products respectively.*

The contribution of the third and above factors appears to be larger for the vega than the price and so we obtain smaller variance reductions, however, a factor of two translates into a factor of 4 speed-up so these are still significant.

8. Conclusion

We have presented a LIBOR Markov-functional model as a control variate for the displaced-diffusion LIBOR market model and shown that it is effective in reducing the standard error of Bermudan swaption prices and vegas. The control variate's effectiveness is due to the accurate analytic calibration method presented, which allows for small perturbations in the LMM volatility structure to be translated into small perturbations in the separable volatility structure.

The control variate produces large standard error reductions for pricing, ranging from 10 to 20 times for the 20Y Bermudan swaptions. It also produces useful reductions for vega calculations, ranging from 5 to 8 times for the 20Y product. These standard error reductions are significant as they dramatically reduce the number of required simulations for a desired level of accuracy.

Appendix A. Control Variate θ

This section considers the estimation of θ in the control variate equation

$$V_{control} = Z_i - \theta(Y_i - \bar{Y}). \tag{A.1}$$

The value θ can be thought of as the slope of the least-squares regression line between the LMM prices, Z_i and the Markov-functional model

prices, Y_i. To minimise the variance of (A.1) using the sample data, we use

$$\hat{\theta} = \frac{\sum_{i=1}^{n}(Y_i - \bar{Y})(Z_i - \bar{Z})}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2},$$

which has a standard error of

$$SE(\hat{\theta}) = \frac{\sqrt{\sum_{i=1}^{n}(Z_i - \bar{Z})^2/(n-2)}}{\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}.$$

For the pricing Table 6.1 and the three vega Tables 7.1-7.3, we compute the corresponding point estimate and standard error of θ, where all of the pricing paths were used to compute it, i.e. 10000.

	2 factor		3 factor		5 factor	
5Y	1.00	(0.01)	1.00	(0.01)	1.00	(0.01)
7Y	1.01	(0.01)	1.01	(0.01)	1.01	(0.01)
10Y	1.01	(0.01)	1.01	(0.01)	1.01	(0.01)
20Y	1.03	(0.01)	1.03	(0.01)	1.02	(0.01)

Table A.1. *Estimate and standard error of θ in equation (A.1), corresponding to the pricing Table 6.1*

	2 factor		3 factor		5 factor	
5Y	1.01	(0.01)	1.01	(0.01)	1.01	(0.01)
7Y	1.01	(0.01)	1.01	(0.01)	1.01	(0.01)
10Y	1.02	(0.01)	1.02	(0.01)	1.02	(0.01)
20Y	1.06	(0.01)	1.05	(0.01)	1.06	(0.01)

Table A.2. *Estimate and standard error of θ in equation (A.1), corresponding to the vega Table 7.1*

	2 factor		3 factor		5 factor	
5Y	1.01	(0.01)	1.00	(0.01)	1.00	(0.01)
7Y	1.01	(0.01)	1.02	(0.01)	1.02	(0.01)
10Y	1.03	(0.01)	1.01	(0.01)	1.03	(0.01)
20Y	1.08	(0.01)	1.06	(0.01)	1.08	(0.01)

Table A.3. *Estimate and standard error of θ in equation (A.1), corresponding to the pricing Table 7.2*

As we can see from Tables A.1 - A.4, the value of $\hat{\theta}$ is often within two standard deviations of 1. it gets farther from 1 as as the length of the product increases, but even the worst case it is only 1.08.

	2 factor		3 factor		5 factor	
5Y	1.01	(0.01)	1.00	(0.01)	1.00	(0.01)
7Y	1.01	(0.01)	1.02	(0.01)	1.02	(0.01)
10Y	1.03	(0.01)	1.04	(0.01)	1.03	(0.01)
20Y	1.08	(0.01)	1.06	(0.01)	1.07	(0.01)

Table A.4. *Estimate and standard error of θ in equation (A.1), corresponding to the pricing Table 7.3*

We computed the standard error reductions using the above values of $\hat{\theta}$, instead of $\theta = 1$, and found the additional variance reduction to be very small. We therefore recommend using a blanket value of $\theta = 1$. This is also useful from a computational point of view as we are not required to spend extra time computing $\hat{\theta}$.

Appendix B. Alternative Calibration

Here we compare calibrating A_i^* to B_i^* directly to the calibration technique from Section 7.1. That is can we find A_i^* by

$$[A_i^* - B_i^*]^2 \to \min, \tag{B.1}$$

or

$$[(A_i^* - B_i^*) - (A_i - B_i)]^2 \to \min. \tag{B.2}$$

We denote the first method as method 1 and the second as method 2. In Table B.1 we present the standard error reductions using calibration method 1 and 2 when computing the vega for a single caplet, as in Table 7.3.

		2 factor	3 factor	5 factor
5Y	method 1	61x	6x	4x
	method 2	30x	5x	4x
7Y	method 1	42x	4x	4x
	method 2	21x	4x	3x
10Y	method 1	21x	4x	3x
	method 2	12x	4x	3x
20Y	method 1	8x	2x	2x
	method 2	8x	2x	2x

Table B.1. *Comparison of calibration techniques when computing the vega corresponding to Table 7.3. Method 1 uses equation (B.2), while method 2 uses equation (B.2).*

Table B.1 shows that the alternative calibration methodology, method 2, performs the same or worse that method 1. Therefore we recommend using method 1.

References

[1] Leif B. Andersen. *A Simple Approach to the Pricing of Bermudan Swaptions in the Multi-Factor Libor Market Model*, volume 3. Journal of Computational Finance, 2 edition, 1999.

[2] Alan L. Andrew, K.-W. Eric Chu, and Peter Lancaster. Derivatives of eigenvalues and eigenvectors of matrix functions. *SIAM J. Matrix Anal. Appl.*, 14(4):903–926, 1993.

[3] C. Bender, A. Kolodko, and J. Schoenmakers. Iterating cancellable snowballs and related exotics. *Risk*, pages 126–130, 2006.

[4] A. Brace. *Engineering BGM*. Chapman & Hall/CRC, 2007.

[5] Damiano Brigo and Fabio Mercurio. *Interest rate models - theory and practice: with smile, inflation and credit*. Springer, 2nd edition, 2006.

[6] Oren Cheyette. *Markov Representation of the Heath-Jarrow-Morton Model*. SSRN, http://ssrn.com/abstract=6073, 1996.

[7] Christian P. Fries and Mark Joshi. Partial proxy simulation schemes for generic and robust monte-carlo greeks. *Journal of Computational Finance*, 1(12), 2008.

[8] M. Giles and P. Glasserman. Smoking adjoints: fast monte carlo greeks. *Risk*, January:92–96, 2006.

[9] Paul Glasserman. *Monte Carlo methods in financial engineering*. Springer, 2003.

[10] P. J. Hunt and J. E. Kennedy. *Financial derivatives in theory and practice*. Wiley, 2004.

[11] C. Hunter, P. Jackel, and M. Joshi. Getting the drift. *Risk*, 14(7):81–84, 2001.

[12] M. S. Joshi. *The Concepts and practice of mathematical finance*. Cambridge University Press, 2003.

[13] Mark Joshi and Alan Stacey. New and robust drift approximations for the libor market model. *Quantitative Finance*, 8(4):427 – 434, 2008.

[14] Joanne E. Kennedy, P. J. Hunt, and Antoon Pelsser. Markov-functional interest rate models. *Finance and Stochastics*, 4(4):391, 2000.

[15] Roger Lord and Antoon Pelsser. Level-slope-curvature - fact or artefact? *Applied Mathematical Finance*, 14(2):105 – 130, 2007.

[16] J. Niesen and W.M. Wright. A krylov subspace algorithm for evaluating the φ-functions appearing in exponential integrators. *Working paper*, 2009.

[17] Antoon Pelsser. *Efficient methods for valuing interest rate derivatives*. Springer, 2000.

[18] Raoul Pietersz, Antoon A. Pelsser, and Marcel Van Regenmortel. Fast drift approximated pricing in the bgm model. *Journal of Computational Finance*, 8(1):93–124, 2003.

[19] Vladimir Piterbarg. A practitioner's guide to pricing and hedging callable libor exotics in forward libor models. *Journal of Computational Finance*, 8:65–119, 2004.