FAST GREEKS FOR MARKOV-FUNCTIONAL MODELS USING ADJOINT PDE METHODS

NICK DENSON AND MARK JOSHI

ABSTRACT. This paper demonstrates how the adjoint PDE method can be used to compute Greeks in Markov-functional models. This is an accurate and efficient way to compute Greeks, where most of the model sensitivities can be computed in approximately the same time as a single sensitivity using finite difference. We demonstrate the speed and accuracy of the method using a Markov-functional interest rate model, also demonstrating how the model Greeks can be converted into market Greeks.

1. INTRODUCTION

Markov-functional models are popular in finance because they are accurate and computationally efficient. They offer an alternative to Monte Carlo models, particularly when the dimension of the problem is three or less. Whilst high dimensional Monte Carlo models are becoming more prolific due to the complexity of new products, because of computational requirements, low-dimensional models are often used for risk management purposes.

One very effective way of computing Greeks in Monte Carlo is to use the adjoint method introduced by Giles and Glasserman, [9]. This method applies to Lipschitz continuous payoffs, however there are techniques to handle a discontinuity, see [5]. To use the adjoint method, the asset process is simulated forwards in time until the final date, at this date, the payoff is differentiated with respect to the input parameters and the sensitivities are propagated backwards in time. The method is particularly useful for scalar payoff functions and a large vector of sensitivities, which is the case for products in the LIBOR market model (LMM) of Miltersen, Sandmann, and Sondermann [18], Brace, Gatarek and Musiela [3] and Jamshidian [13].

The literature on adjoint partial differential equations (PDEs) in finance is sparse. There is a presentation by Prideaux, [23], discussing adjoint PDE methods, however it only considers the Black-Scholes PDE. The main contribution of this paper is to introduce the adjoint PDE method for Markov-functional models, demonstrating how to accurately and efficiently compute Greeks.

To calculate the Greeks with the adjoint PDE method we first compute the price by discretising the asset process, applying the payoff function and solving the PDE backwards in time to today. We then initialise the adjoint PDE and it solve it forwards in time until expiry, using its solution along with the derivative of our payoff with respect to a model parameter to give the Greek. The key point is that the adjoint PDE is designed not to contain any information specifically related to the Greek of interest. This means the adjoint PDE is

Date: May 31, 2010.

Key words and phrases. Adjoint PDE Greeks, delta, vega, skew, adjoint method, PDE, Markov-functional model, market Greeks, cancellable inverse floater, Bermudan swaption.

solved once to obtain all of the model Greeks. Other than the computational benefit, we compute the Greeks more accurately than a finite difference or bump and revalue approach, because we are not introducing bias from perturbing an input parameter. In fact, we can compute most of the model Greeks in approximately the same time it takes to compute a single Greek using finite difference.

To demonstrate the effectiveness of the adjoint PDE method we use a version of the separable LIBOR Markov-functional model introduced by Pietersz, Pelsser and Van Regenmortel, [21]. We choose this model because of its similarities to the LMM and because it is also similar to other Markov-functional models, such as Hunt, Kennedy and Pelsser, [11], at least in the one-dimensional case (see [1]). We use a displaced-diffusion version of the model, which allows for skew in caplet or swaption volatilities, a persistent market feature (see [14]).

We demonstrate the superiority of the adjoint PDE method in computing model deltas, vegas and skew sensitivities for an interest rate cap, cancellable inverse floater swap and callable Bermudan swaption. For both the inverse floater and Bermudan swaption we demonstrate how to convert the adjoint model vegas into hedge ratios for market traded instruments i.e. market vegas.

This paper is organised as follows. Section 2 introduces a general Markov-functional model. Section 3 introduces the separable LIBOR Markov-functional model. We then discuss different methods to compute Greeks and introduce the adjoint PDE method in Section 4. This method is applied to the separable LIBOR Markov-functional model, with the results shown in Section 5. We demonstrate how to convert the model vegas into Market vegas in Section 6.

The authors would like to thank Will Wright for his insightful comments.

2. Markov-functional model

A Markov-functional model consists of Markov factors, $X \in \mathbb{R}^F$, which are used to create a vector of state variables $\xi \in \mathbb{R}^n$. The functional form of the model, which determines the relationship between ξ and X, is typically chosen to fit market dynamics whilst retaining tractability. Once the functional form is chosen, the future values of the state variables are determined by the initial model inputs and the Markov factors. These state variables are then converted into the option price by the deflated payoff function g. A sketch of the model is

$$X \mapsto \xi \mapsto g. \tag{2.1}$$

This paper focuses on computing sensitivities of the price with respect to the initial model inputs. We denote the sensitivities to the Markov terms as *Markov Greeks* and the sensitivity to the non Markov terms as *mapping Greeks*. Markov Greeks are typically the vegas for time-dependent volatility terms, while mapping Greeks are typically sensitivities to the initial state variables or non time-dependent volatility terms. This is further demonstrated in the case of an interest rate model in Section 3.

Consider an option with expiry T_n. In a Markov-functional model, we use the Feynman-Kac formula to derive a PDE for its deflated value, which has the general form

$$\begin{aligned} \frac{\partial g(t)}{\partial t} &= Pg(t), \\ g_{T_n} &= g(\xi, T_n), \end{aligned} \tag{2.2}$$

where P is the partial differential operator and the terminal condition at expiry is given by the option's payoff function. Throughout this paper we assume that g is Lipschitz continuous. This assumption is necessary for the adjoint Greeks section, because we will need to differentiate the payoff function.

An effective way to numerically solve the above PDE is to use the method of lines [24] to convert the PDE into a large system of ordinary differential equations (ODEs). We can then use the wealth of available literature to solve the system of ODEs. The well-known method of lines discretises the PDE in all dimensions except time. For example, assume we have the partial differential operator

$$P = \frac{\partial^2}{\partial x^2},$$

we construct a symmetric grid of possible x values with (for ease of exposition) equal spacing between nodes of length Δx. We then substitute in a finite difference approximation for g. If central differencing is used we have

$$\frac{\partial g(\xi, t)}{\partial x_i} = \frac{g_{i-1}(\xi, t) - 2g_i(\xi, t) + g_{i+1}(\xi, t)}{\Delta x^2},$$

where $g_i(\xi, t)$ represents the deflated payoff function evaluated at the ith node. If we let m denote the number of nodes to the power F (the number of Markov factors), then this generates an $m \times m$ matrix A (tridiagonal in this example) and allows one to write the problem as a system of ODEs

$$\begin{aligned} \frac{du(t)}{dt} &= Au(t) + b, \\ u(T_n) &= g(\xi, T_n), \end{aligned} \tag{2.3}$$

where $b \in \mathbb{R}^m$ contains the boundary conditions that cannot fit into A and $u \in \mathbb{R}^m$ is a vector of discretised g values. For ease of exposition we have assumed Dirichlet boundary conditions, so that b is not time dependent. This ODE is solved backwards in time creating an m-dimensional solution vector. The value of the option is given by the scalar function h,

$$h(u, 0) = \begin{cases} u(0), & \text{at } \theta_0, \\ 0, & \text{otherwise}, \end{cases} \tag{2.4}$$

which selects the price at the initial model parameters θ_0 or origin (usually the middle point in the case of a symmetric grid).

For Bermudan type products, which can be exercised on a discrete set of dates T_j, an exercise strategy needs to be incorporated into the pricing phase. A common method is to initialise the PDE at one exercise date and solve back to the previous exercise date. This way the exercise strategy only enters through the terminal conditions. We initialise the ODE

at exercise time T_k by comparing the solution from the previous ODE (i.e. the continuation value) with the value upon immediate exercise. We define h at node j to be

$$h_j(u, T_k) = \max\{u_j(T_{k+}), g_j(T_k)\}, \tag{2.5}$$

where $u_j(T_{k+})$ is the continuation value and $g_j(T_k)$ is the deflated exercise value. This process is repeated backwards through the exercise dates until we reach time zero.

3. Separable LIBOR Markov-functional model

We now give a concrete example of a Markov-functional interest rate model. We focus on the F-factor separable LIBOR Markov-functional model, introduced by Pietersz, Pelsser and Van Regenmortel [21], with the extension to a more general volatility structure as presented by Denson and Joshi, [8]. The model is based on a similar idea to the LIBOR market model of evolving discrete market observable forward rates.

For this model, the n state variables ξ in (2.1) are discrete forward rates $f_0, \ldots, f_{n-1}$ with a corresponding tenor structure $0 < T_0 < T_1 < \ldots < T_n$, with $\tau_i = T_{i+1} - T_i$. The forward rate f_i is for the interval $[T_i, T_{i+1})$. We assume that the forward rates have the following dynamics

$$\frac{d(f_i(t) + \alpha_i)}{f_i(t) + \alpha_i} = \mu_i(f, t)dt + \nu_i^T dX(t),$$

where the α_i's are constant displacement coefficients and the forward rate specific volatility terms ν_i are F-dimensional deterministic vectors. The F-dimensional vector of Markov factors $X(t)$ is defined by

$$dX(t) = C(t)dW(t), \tag{3.1}$$

with $X(0) = 0$ and $W(t)$ being a standard F-dimensional Brownian motion under the pricing measure. We assume that $C(t)$ is an $F \times F$ matrix that is constant across tenor dates $[T_i, T_{i+1})$. The separable volatility structure is therefore given by

$$\nu_i^T C(t), \tag{3.2}$$

which gets its name because the forward rate and time dependence are separated out. Let $C(k)$ represent the matrix of constant values over $[T_{k-1}, T_k)$, with the convention that $T_{-1} = 0$. The log-forward rates are evolved across the step 0 to T_k using

$$\log(f_i(T_k) + \alpha_i) = \log(f_i(0) + \alpha_i) + \mu_i(0) - \mathrm{cov}_{ii}(0)/2 + \nu_i X(T_k), \tag{3.3}$$

where $\mu_i(0)$ is the integrated drift over the evolution step $[0, T_k)$, $\mathrm{cov}_{ii}(0)$ is the variance of the log-forward rate across the step and $X(T_k)$ is the integral of (3.1) over the step. This equation gives the functional form of the model. To ensure that the model remains Markovian we make the modification that the forward rates are evolved to each time in a single step from time zero. If we are working in the terminal measure, which corresponds to using the zero-coupon bond $P(t, T_n)$ as numeraire, the drift of the ith forward rate is given by

$$\mu_i(0) = -\sum_{j=i+1}^{n-1} \frac{(f_j(0) + \alpha_j)\tau_j}{1 + f_j(0)\tau_j} \mathrm{cov}_{ij}(0). \tag{3.4}$$

One issue with single stepping the forward rates is that the integral of the drift needs to be approximated over long intervals, possibly ten years or more. We therefore need an accurate drift approximation or the model will suffer from a large discretisation bias. One accurate approximation is predictor-corrector introduced by Hunter, Jäckel and Joshi [12], which is an extension of the scheme introduced in equation (15.5.4) by Kloeden and Platen [16]. The algorithm, from [12], is

(1) Evolve the logarithms of the forward rates as if the drifts were constant and equal to their initial values according to the log-Euler scheme (3.3).
(2) Compute the drifts at the terminal time with the so evolved forward rates.
(3) Average the initially calculated drift coefficients with the newly computed ones.
(4) Re-evolve using the same Markov variates as initially but using the new predictor-corrector drift terms.

If we let a hat above a forward rate denote that it has been estimated through an initial log-Euler evolution using (3.3) with (3.4) as the drift, we define

$$\mu_i(0)^{\mathrm{PC}} = -\frac{1}{2}\left(\sum_{j=i+1}^{n-1}\frac{(f_j(0)+\alpha_j)\tau_j}{1+f_j(0)\tau_j}\mathrm{cov}_{ij}(0) + \sum_{j=i+1}^{n-1}\frac{(\hat{f}_j(0)+\alpha_j)\tau_j}{1+\hat{f}_j(0)\tau_j}\mathrm{cov}_{ij}(0)\right).$$

In practice, the number of Markov factors used will typically be three or less, so a PDE implementation will be faster than Monte Carlo. The Feynman-Kac formula yields a PDE with 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 ith Markov factor. For the coefficients, let $c_{ij}(t)$ be the element in the ith row of the jth column of $C(t)$ in (3.2)

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

For this model the mapping Greeks will be the sensitivity of the price to any of the n initial forward rates $f(0)$, nF volatility terms ν or n displacement coefficients α. The Markov Greeks will be the sensitivity of the price with respect to the time-dependent volatility terms $c_{ij}(t)$.

4. PDE Greeks

4.1. **Finite difference Greeks.** One simple method to compute sensitivities is via finite difference, or bump and revalue. We compute the price with the initial market parameters, θ_0, then perturb a single parameter by a small amount, ϵ and recompute the price. The sensitivity is then given by

$$h'(u(\theta_0),0) = \frac{h(u(\theta_0+\epsilon),0) - h(u(\theta_0),0)}{\epsilon}.$$

The main benefit of this method is that it is simple to implement. There are, however, many drawbacks (see, for example [22]) the main ones being

- Careful choice of the perturbation size ϵ.
- At least one extra price needs to be computed for each extra sensitivity.

Choosing an appropriate perturbation size can be difficult and depends on the size of the parameter θ. Ideally we want to chose a small value to reduce the bias of the estimate, however a value too small will lose the estimate in numerical noise.

For most financial models, particularly interest rate models, there are a large number of Greeks. For a medium dated product the number can easily be greater than fifty. If we need to compute an extra fifty prices, then computing the Greeks will be computationally expensive, even for a fast model.

4.2. **Computing Greeks directly.** We can avoid the bias of the finite difference Greeks by computing them directly. Since we are considering Lipschitz-continuous payoff functions, we can differentiate and derive a PDE for the particular Greek of interest. Consider computing the sensitivity to a model parameter θ. We let

$$\Delta_\theta := \frac{\partial g}{\partial \theta}. \tag{4.1}$$

The idea is to apply this partial derivative to the pricing equation (2.2). If we are computing mapping Greeks then the partial differential operator will not contain θ and we have

$$\begin{aligned}\frac{\partial}{\partial \theta}\left(\frac{\partial g}{\partial t} = Pg\right) &\Rightarrow \left(\frac{\partial^2 g}{\partial t \partial \theta} = \frac{\partial}{\partial \theta} Pg\right), \\ &\Rightarrow \left(\frac{\partial \Delta_\theta}{\partial t} = P\Delta_\theta\right),\end{aligned}$$

since partial differentiation commutes. We can use the method of lines to transform this PDE into an ODE and solve for the single Greek.

If we are computing Markov Greeks the PDE will explicitly contain θ. We can still derive a PDE for the Greek, however, the partial differential operator will change during differentiation.

The main benefit of this method is that the Greeks are computed without bias. One problem, however, is that we need to solve an extra PDE for each extra Greek. We can reuse some information, particularly for the mapping Greeks as the only change is the terminal condition, not the PDE itself. However, this requires adjustments to the particular lattice solver, which is not ideal.

4.3. **Mapping adjoint Greeks.** The adjoint method computes all of the first order mapping Greeks, without bias, in approximately the same time as it takes to compute the price. It works by calculating an intermediate quantity first and then computing the Greeks. We initially consider a single time horizon 0 and T_n, before moving on to multiple time horizons in the next sub-section, therefore for ease of notation let $h(u, 0) = h(u)$.

The following derivation is based on the more general case of differential algebraic equations presented by Cao, Li, Petzold and Serban [4]. We want to find the adjoint of the large system of ODEs given by (2.3), where we ignore the vector of boundary conditions for ease of exposition. In any case, the spatial discretisation is usually chosen to be large

enough that the boundary conditions do not significantly affect the price, see [25]. The solution to this system of ODEs is given by

$$u(t) = e^{At} u_{T_n}.$$

Rewriting this solution gives

$$\gamma(u, \theta) := e^{-At} u(t) - u_{T_n} = 0. \tag{4.2}$$

We want to compute the sensitivity of the initial price, given by $h(u)$ in (2.4), to a mapping parameter θ. To do so we introduce a Lagrange multiplier λ

$$I(u, \theta) := h(u) - \lambda^T \gamma(u, \theta) = h(u),$$

where the right equality comes from the definition of γ. Here λ is not a free parameter in the usual Lagrange multiplier sense, but will be specified by the adjoint equation. The sensitivity of the price is now given by

$$\begin{aligned} \frac{\partial I}{\partial \theta} &= \frac{\partial h}{\partial u}\frac{\partial u}{\partial \theta} - \lambda^T \left(\frac{\partial \gamma}{\partial \theta} + \frac{\partial \gamma}{\partial u}\frac{\partial u}{\partial \theta} \right), \\ &= -\lambda^T \frac{\partial \gamma}{\partial \theta} + \left(\frac{\partial h}{\partial u} - \lambda^T \frac{\partial \gamma}{\partial u} \right) \frac{\partial u}{\partial \theta}. \end{aligned} \tag{4.3}$$

The issue with computing the Greeks directly in the previous sub-section, is that a new equation needs to be solved for each parameter θ. This problem also appears above, expressed as the term $\partial u / \partial \theta$. We therefore choose the parameter λ to eliminate this term by setting

$$\frac{\partial h}{\partial u} - \lambda^T \frac{\partial \gamma}{\partial u} = 0.$$

Using (4.2) we can compute

$$\frac{\partial \gamma}{\partial u} = e^{-At},$$

which, after transposing, gives

$$e^{-A^T t} \lambda(t) - \left(\frac{\partial h}{\partial u} \right)^T = 0.$$

This equation is the solution of an ODE in the same way as (4.2) is. We can therefore specify our adjoint system of ODEs as

$$\begin{aligned} \frac{d\lambda(t)}{dt} &= A^T \lambda(t), \\ \lambda(0) &= \frac{\partial h(u)}{\partial u}. \end{aligned} \tag{4.4}$$

This adjoint system is solved forwards in time, since the function h is only defined at time 0. Its partial derivative, which gives the initial conditions, is

$$\lambda(0) = \begin{cases} 1, & \text{at } \theta_0, \\ 0, & \text{otherwise}. \end{cases} \tag{4.5}$$

This adjoint equation is no more difficult to solve than (2.3). We simply solve the equation forwards in time to get the vector solution $\lambda(T_n)$. To calculate a Greek we put the adjoint solution back into (4.3)

$$\begin{aligned}\frac{\partial h}{\partial \theta} &= -\lambda^T \frac{\partial \gamma}{\partial \theta}, \\ &= -\lambda^T \left(\frac{\partial e^{-At}}{\partial \theta} u - \frac{\partial u_{T_n}}{\partial \theta} \right).\end{aligned}$$

Since A is a matrix the derivative of the first term appearing inside the brackets above is given by the general formula for matrix exponentials from [19]

$$\frac{\partial e^{-At}}{\partial \theta} = -\int_0^{T_n} e^{-As} \frac{\partial A}{\partial \theta} e^{-A(T_n - s)} ds,$$

which gives

$$\frac{\partial h}{\partial \theta} = \int_0^{T_n} \left(\lambda^T (T_n - s) \frac{\partial A}{\partial \theta} u(s) \right) ds + \lambda(T_n)^T \frac{\partial u_{T_n}}{\partial \theta}, \tag{4.6}$$

where $\partial u_{T_n}/\partial \theta$ is the derivative of the payoff function with respect to a parameter of interest at each of the nodes. For the mapping Greeks the term $\partial A/\partial \theta = 0$ and the expression is significantly simplified

$$\frac{\partial h}{\partial \theta} = \lambda(T_n)^T \frac{\partial u_{T_n}}{\partial \theta}.$$

This equation along with (4.4) are used to compute all of the mapping Greeks. The key point is that the adjoint equation (4.4) does not contain θ and only needs to be solved once. Each additional sensitivity then only requires an extra dot product, we do not need to solve another equation.

For stability, the adjoint method is numerically stable if the pricing method is stable. Since the adjoint system is solved forwards in time we can do a change of variable from t to $T_n - t$ and the adjoint system becomes

$$\frac{d\lambda(T_n - t)}{dt} = A^T \lambda(T_n - t),$$

which is solved backwards in time. As A is a square constant matrix, its eigenvalues are the same as the eigenvalues of its transpose. Therefore the adjoint system will be stable if the pricing system is also stable. In the more general case of a non-constant matrix A the adjoint system is also stable, see Theorem 4.1 in [4].

To derive the adjoint method we started with the ODE, which arose from spatial discretisation of the PDE. We note that there can be some differences with the adjoint method derived directly from the PDE. These differences, however, are small and will not lead to incorrect results for the Greeks, see [17].

4.4. **Mapping adjoint Greeks for callable products.** The complexity of callable (or cancellable) products involves incorporating the exercise strategy into the adjoint equation. As recommended by Piterbarg, [22], we use the same exercise strategy for the price and the first order Greeks, because the sensitivity of the exercise strategy to a model parameter is only a second order effect.

We solve the adjoint PDE across exercise dates in the same way as we compute the price. The exercise strategy then only enters through the initial conditions, where we zero the nodes at points of exercise. We do this to ensure that the product is not exercised multiple times. The algorithm to compute the Greeks is

(1) Perform the pricing routine storing the exercise times and computing and storing all of the pathwise derivatives at each node on each exercise date. That is, for one particular node at time T_k we will have a vector containing elements

$$\frac{\partial g(T_k)}{\partial \theta},$$

of all model parameters of interest θ and a variable specifying whether we exercised at that date.

(2) Initialise the adjoint equation at time zero using the same conditions as the single time step case (4.5). Solve the adjoint equation (4.4) to the first exercise date, T_0.

(3) Compute and store the Greeks at time T_0 using the inner product of the adjoint solution and the stored payoff derivatives

$$\lambda(T_0)^T \frac{\partial u(T_0)}{\partial \theta},$$

for each element of θ.

(4) Using the stored exercise times from the pricing phase, initialise the adjoint equation at node j, time T_0 with

$$\lambda_j(T_0) = \begin{cases} \lambda_j(T_{0^-}), & \text{if not exercised at node } j, \\ 0, & \text{if exercised at node } j, \end{cases}$$

where $\lambda(T_{0^-})$ is the solution of the previous adjoint equation. Solve for λ to the next exercise date T_1.

(5) Compute the inner product

$$\lambda(T_1)^T \frac{\partial u(T_1)}{\partial \theta},$$

for each element of θ and add these values to the inner products computed in step (3).

(6) Repeat steps (4) and (5) until the final exercise time T_{n-1}. The sensitivity of the option price to a parameter θ_j is then

$$\frac{\partial h}{\partial \theta_j} = \sum_{k=0}^{n-1} \lambda(T_k)^T \frac{\partial u(T_k)}{\partial \theta_j}.$$

One potential issue with the adjoint method is that at step (1) for each grid point at each exercise date a vector of derivatives needs to be stored. If there are, for example, 2500 grid points (a two-factor model with a 50×50 grid), 40 exercise dates and 100 sensitivities of interest then we need to store 10 million values. This is about 100 megabytes, which is a small amount for a modern computer. In any case, if memory is an issue we can re-evolve the stochastic process for the forward adjoint step and compute the pathwise derivatives as

we need them in steps (3) and (5). We then only store the running sum in step (6), which requires a minimal amount of memory.

4.5. **Markov Greeks.** We have so far neglected the Markov Greeks, or the sensitivity to the variables which make up the Markov factors. For the case of the separable LIBOR Markov-functional model these are the volatility terms $c_{ij}(k)$ and we are interested in computing the vegas

$$\frac{\partial h(u)}{\partial c_{ij}(k)}.$$

The difference between computing mapping and Markov vegas using the adjoint method is that the integral correction term in (4.6) will no longer be zero. To compute this integral we need to store both the adjoint terms λ and the price terms u for all nodes where the integrand is to be evaluated, which makes the Markov Greeks more memory intensive than the mapping Greeks.

Consider the example of piecewise constant Markov terms $C(k+1)$ across the interval $[T_k, T_{k+1})$, which is the case for the LIBOR Markov-functional model in this paper. To compute the sensitivity to these Markov terms we have three different time intervals to consider, before T_k, across $[T_k, T_{k+1})$ and after T_{k+1}.

Over the time interval $[0, T_k)$ the derivative of $\partial A/\partial C(k+1)$ is zero and we can therefore use the same algorithm as the mapping Greeks in Section 4.4. That is we compute the summation in step (6) until time T_k.

Between reset times T_k and T_{k+1} the derivative of $\partial A/\partial C(k+1)$ will be non zero and we need to compute the integral correction term. To compute this integral we use the following algorithm:

(1) Ensure that the same time discretisation is used when numerically solving the adjoint and pricing PDEs. Store the vectors $\lambda(T_{k+1} - s)$ and $u(s)$ at each time step s.
(2) Compute $\partial A/\partial C(k+1)$.
(3) Evaluate $\lambda^T(T_{k+1} - s)\frac{\partial A}{\partial \theta}u(s)$ for each time step s.
(4) Compute the integral above using a numerical integration technique.

The resulting integral and the term $\lambda(T_{k+1})^T \frac{\partial u_{T_{k+1}}}{\partial C(k+1)}$ is then added to the summation computed up to time T_k.

After time T_{k+1} the integral correction term will be zero and we continue computing the vega using the algorithm in the mapping Greeks Section.

One way to ensure we use the same time discretisation in step (1) above is to use a fixed time stepping technique such as the Alternating Direction Implicit (ADI) method of Craig and Sneyd [6]. The integrand in step (3) will generally be well behaved because both λ and u will be well behaved for realistic payoff functions and the term $\partial A/\partial C(k+1)$ essentially pulls out a linear multiple of λ. This means that we can use the trapezoidal rule to do the numerical integration. For the LIBOR Markov-functional model we have found that this works well.

5. Numerical results

In this section we demonstrate how well the adjoint method works for the separable LIBOR Markov-functional model. The partial derivative of the payoff function involves differentiating the discretisation scheme, details of which are given in Appendix A.

Three products are considered, an interest rate cap, a cancellable inverse floater swap and a callable Bermudan swaption, all with 10 year maturities. Semi-annual forward rates are used, flat at 5% with displacements of 1% for each forward rate. We consider a strike of 7% for the cap, because an out-of-the-money product is less linear in volatility and is therefore a more difficult test.

For the cancellable inverse floater we take the point of view of the issuer who receives the floating LIBOR rate and pays the coupon

$$\max(K - 2f_i(T_i), 0)\tau_i,$$

at time T_{i+1}. The strike is 10%. Upon cancellation at any of the tenor dates $T_0, \ldots, T_{n-1}$ no rebate is paid.

For the callable Bermudan swaption we model the product as an option to enter into an interest rate swap at any of the tenor dates above. The net present value of the swap is

$$\max(\mathrm{SR}_i(T_i) - K, 0)A_i(T_i),$$

where $\mathrm{SR}_i(T_j)$ is the coterminal swap rate from T_i to T_n evaluated at time T_j and similarly $A_i(T_j)$ is the annuity of the swap. The strike is set to 7%.

We use a two-factor model with a simple volatility structure to facilitate the reproduction of results

$$\nu_i^T = \left[\begin{array}{cc} 0.10 & 0.08 \end{array} \right],$$

where the forward rate specific volatility terms are the same for all forward rates and the time-dependent matrices are all equal to

$$C(k) = \left[\begin{array}{cc} 1 & 1 \\ 1 & 0 \end{array} \right].$$

Ignoring displacements, this equates to an annual implied Black volatility of approximately 21%. To solve the pricing and adjoint PDEs we use the Krylov subspace method introduced in [20], with a symmetric and equally spaced spatial discretisation, ranging to five standard deviations in the Markov factors.

To test the accuracy of the model and adjoint method Table 5.1 displays the exact sensitivities computed using the Black formula [2] and the numerically computed sensitivities for the Markov-functional model (MFM) with 51 spatial nodes in each dimension. We display the deltas, half of the mapping vegas and the skew sensitivities. The prices of the individual caplets range from 0.43bps to 28.3bps, with the cap worth 356.6bps. We see that the adjoint method performs well, computing most of the deltas very accurately, except for a few of the later forward rates. The vega and skew sensitivity estimates are also accurate, with a maximum absolute error of 0.2%.

The small errors in the estimates come from two sources, the numerical PDE solution and the model. To eliminate the PDE errors we can use a better numerical scheme or a larger grid. We use 101 spatial nodes in each dimension for the results in Table 5.2. Here

we see that the estimates have improved, however the later deltas are still off by close to 1% in some cases. This inaccuracy is most likely due to model error, because we are single stepping the forward rates from time 0 to time T_k and the drift approximation is becoming inaccurate.

Delta	Black	MFM	Vega	Black	MFM	Skew	Black	MFM
f_0	-0.4%	0.4%	$\nu_{0,0}$	0.2%	0.3%	α_0	0.4%	0.6%
f_1	2.8%	3.1%	$\nu_{1,0}$	0.7%	0.7%	α_1	1.4%	1.6%
f_2	5.3%	4.9%	$\nu_{2,0}$	1.1%	1.1%	α_2	2.4%	2.3%
f_3	7.1%	7.2%	$\nu_{3,0}$	1.4%	1.5%	α_3	3.2%	3.2%
f_4	8.5%	8.5%	$\nu_{4,0}$	1.7%	1.8%	α_4	3.8%	3.8%
f_5	9.5%	9.0%	$\nu_{5,0}$	1.9%	2.0%	α_5	4.3%	4.2%
f_6	10.3%	10.4%	$\nu_{6,0}$	2.1%	2.3%	α_6	4.8%	4.8%
f_7	11.0%	10.7%	$\nu_{7,0}$	2.3%	2.4%	α_7	5.1%	5.1%
f_8	11.5%	11.4%	$\nu_{8,0}$	2.4%	2.6%	α_8	5.5%	5.5%
f_9	11.9%	11.7%	$\nu_{9,0}$	2.6%	2.7%	α_9	5.7%	5.7%
f_{10}	12.2%	12.3%	$\nu_{10,0}$	2.7%	2.9%	α_{10}	6.0%	6.0%
f_{11}	12.5%	12.1%	$\nu_{11,0}$	2.7%	2.9%	α_{11}	6.2%	6.1%
f_{12}	12.7%	12.5%	$\nu_{12,0}$	2.8%	3.0%	α_{12}	6.3%	6.3%
f_{13}	12.9%	12.7%	$\nu_{13,0}$	2.9%	3.1%	α_{13}	6.5%	6.5%
f_{14}	13.0%	12.3%	$\nu_{14,0}$	2.9%	3.1%	α_{14}	6.6%	6.5%
f_{15}	13.1%	12.4%	$\nu_{15,0}$	3.0%	3.1%	α_{15}	6.7%	6.6%
f_{16}	13.2%	12.5%	$\nu_{16,0}$	3.0%	3.1%	α_{16}	6.8%	6.7%
f_{17}	13.3%	12.4%	$\nu_{17,0}$	3.0%	3.1%	α_{17}	6.8%	6.7%
f_{18}	13.3%	12.4%	$\nu_{18,0}$	3.0%	3.1%	α_{18}	6.9%	6.8%
f_{19}	13.4%	12.6%	$\nu_{19,0}$	3.0%	3.1%	α_{19}	6.9%	6.9%

Table 5.1. *Deltas, vegas and skew sensitivities for a cap, strike = 7%. A 51×51 spatial grid was used.*

Table 5.3 displays some of the Markov vegas for the interest rate cap. Instead of using the Krylov subspace method to solve the pricing and adjoint PDEs we used the ADI method [6], because it uses a constant step size which makes it easier to compute the integral correction term (4.6). A spatial grid of 101×101 was used with 50 time steps between each tenor date. To compute the integral correction term we evaluated the integrand at all 50 time points and used the trapezoidal rule.

We notice that the Markov vegas are computed reasonably accurately when compared to their true values from the Black formula. The early vegas have the largest error, but these absolute errors are 0.02% or less. The results are similar for the other Markov vega terms c_{11} and c_{22}, which are not shown here.

Table 5.4 shows the sensitivities of the cancellable inverse floater, whose computed value is 2856bps. Since there is no analytic price we compare the adjoint method to the finite difference method. We use a double-sided finite difference with a perturbation of 1bp to each of the model parameters. As we expect the adjoint method matches the finite difference almost exactly. The small differences between a few Greeks are mostly likely caused by

Delta	Black	MFM	Vega	Black	MFM	Skew	Black	MFM
f_0	-0.4%	-0.2%	$\nu_{0,0}$	0.2%	0.2%	α_0	0.4%	0.4%
f_1	2.8%	2.7%	$\nu_{1,0}$	0.7%	0.7%	α_1	1.4%	1.4%
f_2	5.3%	5.5%	$\nu_{2,0}$	1.1%	1.1%	α_2	2.4%	2.4%
f_3	7.1%	6.9%	$\nu_{3,0}$	1.4%	1.5%	α_3	3.2%	3.1%
f_4	8.5%	8.4%	$\nu_{4,0}$	1.7%	1.8%	α_4	3.8%	3.8%
f_5	9.5%	9.4%	$\nu_{5,0}$	1.9%	2.0%	α_5	4.3%	4.3%
f_6	10.3%	10.4%	$\nu_{6,0}$	2.1%	2.3%	α_6	4.8%	4.8%
f_7	11.0%	10.9%	$\nu_{7,0}$	2.3%	2.4%	α_7	5.1%	5.1%
f_8	11.5%	11.5%	$\nu_{8,0}$	2.4%	2.6%	α_8	5.5%	5.5%
f_9	11.9%	11.7%	$\nu_{9,0}$	2.6%	2.7%	α_9	5.7%	5.7%
f_{10}	12.2%	12.0%	$\nu_{10,0}$	2.7%	2.8%	α_{10}	6.0%	5.9%
f_{11}	12.5%	12.2%	$\nu_{11,0}$	2.7%	2.9%	α_{11}	6.2%	6.1%
f_{12}	12.7%	12.3%	$\nu_{12,0}$	2.8%	2.9%	α_{12}	6.3%	6.2%
f_{13}	12.9%	12.5%	$\nu_{13,0}$	2.9%	3.0%	α_{13}	6.5%	6.4%
f_{14}	13.0%	12.5%	$\nu_{14,0}$	2.9%	3.0%	α_{14}	6.6%	6.5%
f_{15}	13.1%	12.6%	$\nu_{15,0}$	3.0%	3.1%	α_{15}	6.7%	6.6%
f_{16}	13.2%	12.6%	$\nu_{16,0}$	3.0%	3.1%	α_{16}	6.8%	6.6%
f_{17}	13.3%	12.5%	$\nu_{17,0}$	3.0%	3.1%	α_{17}	6.8%	6.7%
f_{18}	13.3%	12.4%	$\nu_{18,0}$	3.0%	3.0%	α_{18}	6.9%	6.7%
f_{19}	13.4%	12.5%	$\nu_{19,0}$	3.0%	3.0%	α_{19}	6.9%	6.8%

Table 5.2. *Deltas, vegas and skew sensitivities for a cap, strike = 7%. A 101×101 spatial grid was used.*

numerical noise in the solution of the PDE. In any case, we have found the adjoint method to be more accurate than the finite difference approach as it does not suffer from bias introduced by the numerical bumping.

We do not present the results for the Bermudan swaptions at this stage, instead we present them in the following section.

It is worthwhile considering how long the Greek computations take in the adjoint method. To compute the price and all of the 20 deltas, 40 mapping vegas and 20 skew sensitivities (80 Greeks in total) using a 51×51 grid took 1.0 seconds using single threaded C++ code on a desktop computer[1]. Of this time, 0.6 seconds were spent computing all of the Greeks. For the case of a 101×101 grid the price and sensitivities were computed in 4.0 seconds with 2.2 seconds spent computing the Greeks. This demonstrates that we can compute all of the sensitivities in approximately the same time as it takes to compute only one Greek using finite difference. These results are for a two-factor model and it is therefore likely that a one-factor model would be significantly faster.

For the Markov vegas we used the slower ADI method to solve the pricing and adjoint PDEs, which took 12 seconds to compute the price and all of the mapping and Markov Greeks for a 101×101 grid. Of this time 7 seconds were spent computing the Greeks,

[1] 3.16Ghz Intel CPU, with 4GB of RAM.

Vega	Black	MFM	Vega	Black	MFM
$c_{12}(0)$	0.18%	0.17%	$c_{21}(0)$	0.25%	0.27%
$c_{12}(1)$	0.17%	0.16%	$c_{21}(1)$	0.24%	0.26%
$c_{12}(2)$	0.16%	0.15%	$c_{21}(2)$	0.23%	0.24%
$c_{12}(3)$	0.15%	0.13%	$c_{21}(3)$	0.21%	0.22%
$c_{12}(4)$	0.13%	0.12%	$c_{21}(4)$	0.19%	0.20%
$c_{12}(5)$	0.12%	0.11%	$c_{21}(5)$	0.17%	0.19%
$c_{12}(6)$	0.11%	0.10%	$c_{21}(6)$	0.16%	0.17%
$c_{12}(7)$	0.10%	0.09%	$c_{21}(7)$	0.14%	0.15%
$c_{12}(8)$	0.09%	0.08%	$c_{21}(8)$	0.13%	0.13%
$c_{12}(9)$	0.08%	0.07%	$c_{21}(9)$	0.11%	0.12%
$c_{12}(10)$	0.07%	0.06%	$c_{21}(10)$	0.10%	0.10%
$c_{12}(11)$	0.06%	0.05%	$c_{21}(11)$	0.09%	0.09%
$c_{12}(12)$	0.05%	0.04%	$c_{21}(12)$	0.07%	0.08%
$c_{12}(13)$	0.04%	0.04%	$c_{21}(13)$	0.06%	0.06%
$c_{12}(14)$	0.04%	0.03%	$c_{21}(14)$	0.05%	0.05%
$c_{12}(15)$	0.03%	0.02%	$c_{21}(15)$	0.04%	0.04%
$c_{12}(16)$	0.02%	0.02%	$c_{21}(16)$	0.03%	0.03%
$c_{12}(17)$	0.02%	0.01%	$c_{21}(17)$	0.02%	0.02%
$c_{12}(18)$	0.01%	0.01%	$c_{21}(18)$	0.02%	0.01%
$c_{12}(19)$	0.01%	0.00%	$c_{21}(19)$	0.01%	0.01%

Table 5.3. *Markov vegas for a cap, strike = 7%. A 101×101 spatial grid was used.*

including 4 seconds spent computing the Markov vegas. This demonstrates that while the Markov vegas involve an extra integral, this integral can computed relatively efficiently.

To see how the computational time changes with a varying number of forward rates we considered 40 and 60 rate examples. Using the Krylov subspace method with a 51×51 grid the price and all mapping Greeks were computed in 4.0 and 10.1 seconds for the 40 and 60 rate examples respectively. This reflects the fact that the time complexity is quadratic in the number of forward rates.

6. Market Greeks

To hedge an option, we need to know its sensitivity to changes in market traded instruments such as caplets and swaptions. The previous section, however, computed the option's sensitivity to model parameters, which are not traded. We therefore need a way of converting the model Greeks into market Greeks.

We will focus on vegas since they pose a more difficult problem. The idea is to find the linear combination of model vegas that corresponds to a market vega. For an F-factor model with n forward rates we can realistically expect to use nF market instruments. In the case of a single-factor model these would most likely be n coterminal swaptions. For the two-factor model considered in this paper they could be caplets as well as coterminal swaptions.

To calculate market vegas we adapt the method introduced by Joshi and Kwon, [15], which we now sketch. Let $Y_1, Y_2, \ldots, Y_p$ denote market implied volatilities. We want to

Delta	Finite diff	MFM	Vega	Finite diff	MFM	Skew	Finite diff	MFM
f_0	85.5%	85.5%	$\nu_{0,0}$	-2.1%	-2.1%	α_0	-5.4%	-5.4%
f_1	75.8%	75.8%	$\nu_{1,0}$	-2.9%	-2.9%	α_1	-7.3%	-7.3%
f_2	70.2%	70.2%	$\nu_{2,0}$	-3.1%	-3.1%	α_2	-7.9%	-7.9%
f_3	68.9%	68.9%	$\nu_{3,0}$	-3.2%	-3.2%	α_3	-7.4%	-7.4%
f_4	67.0%	67.0%	$\nu_{4,0}$	-2.9%	-2.9%	α_4	-6.5%	-6.5%
f_5	62.7%	62.7%	$\nu_{5,0}$	-2.6%	-2.6%	α_5	-5.5%	-5.5%
f_6	58.0%	58.0%	$\nu_{6,0}$	-2.1%	-2.1%	α_6	-4.2%	-4.2%
f_7	54.5%	54.6%	$\nu_{7,0}$	-1.7%	-1.7%	α_7	-2.7%	-2.7%
f_8	50.6%	50.5%	$\nu_{8,0}$	-1.3%	-1.3%	α_8	-1.4%	-1.4%
f_9	48.8%	48.8%	$\nu_{9,0}$	-1.0%	-1.0%	α_9	-0.4%	-0.4%
f_{10}	45.8%	45.5%	$\nu_{10,0}$	-0.5%	-0.5%	α_{10}	0.6%	0.6%
f_{11}	43.1%	43.1%	$\nu_{11,0}$	0.0%	0.0%	α_{11}	1.9%	1.9%
f_{12}	40.2%	40.2%	$\nu_{12,0}$	0.4%	0.4%	α_{12}	2.8%	2.8%
f_{13}	38.4%	38.3%	$\nu_{13,0}$	0.7%	0.7%	α_{13}	3.6%	3.6%
f_{14}	36.1%	36.0%	$\nu_{14,0}$	1.1%	1.1%	α_{14}	4.3%	4.3%
f_{15}	33.9%	33.9%	$\nu_{15,0}$	1.4%	1.4%	α_{15}	4.9%	5.0%
f_{16}	31.9%	31.8%	$\nu_{16,0}$	1.8%	1.8%	α_{16}	5.6%	5.6%
f_{17}	30.2%	30.1%	$\nu_{17,0}$	2.1%	2.1%	α_{17}	6.1%	6.1%
f_{18}	28.3%	28.3%	$\nu_{18,0}$	2.4%	2.4%	α_{18}	6.6%	6.7%
f_{19}	26.6%	26.6%	$\nu_{19,0}$	2.7%	2.7%	α_{19}	7.1%	7.1%

Table 5.4. *Deltas, vegas and skew sensitivities for a cancellable inverse floater swap, strike = 10%. A 51×51 spatial grid was used.*

compute $\partial g/\partial Y_i$ using the already computed model vegas. That is, we need to find a linear combination of model vegas so that only one market volatility increases by 1% and all other volatilities remain unchanged. Since these market volatilities are smooth functions of the model volatilities[2], which we denote by ψ, we can compute their partial derivatives

$$y_i^T = \left(\frac{\partial Y_i}{\partial \psi_1}, \frac{\partial Y_i}{\partial \psi_2}, \ldots, \frac{\partial Y_i}{\partial \psi_r} \right), \tag{6.1}$$

for a particular ordering of the r model volatility terms. To find the linear combination of model vegas, or equivalently the r-dimensional weights vector w_i, we require

$$y_i^T w_i = 1\%,$$

so that the inner product with the weights vector increases the market implied volatility by 1% and

$$y_j^T w_i = 0, \tag{6.2}$$

for $j \neq i$ so that all other implied volatilities do not change. Concentrating on a two-factor model, we have r model vegas, but we are using $2n-1$ instruments to hedge (as the last caplet and coterminal swaption coincide) and since we would most likely have $r > 2n-1$

[2]This is true for caplet volatilites and coterminal swaption volatilities when using the Hull-White swaption approximation [10].

we require an extra condition on the weights vector to uniquely determine it. We choose to impose

$$w_i^T w_i \to \min,$$

so that the minimal perturbation is found. Using these three conditions we can determine the $2n-1$ weight vectors as w_i lies in the linear span of the projection of y_i onto the subspace orthogonal to (6.2) above. The market vega is then given by

$$\frac{\partial g}{\partial Y_i} = \sum_{j=i}^{r} w_{i,j} \frac{\partial g}{\partial \psi_j}.$$

The routine will fail if it is not possible to perturb one instrument's volatility and hold the others constant. This could happen if a market volatility is a linear combination of other market volatilities, in which case this redundant market instrument would be removed from the hedging portfolio.

We initially consider hedging a product using only the $2n$ mapping vegas as these terms can be computed quicker than the Markov vegas. Figure 6.1 shows the sensitivity of the cancellable inverse floater in Section 5 with respect to caplet volatilities only. That is we do not include coterminal swaptions in the hedging portfolio. The horizontal axis represents caplets expiring at time T_{k-1} and the vertical axis displays the size of the vega. We notice that the vegas of the early caplets are negative and then become positive. This is because the caplets are trying to hedge both the put options on the LIBOR rate and the cancellability of the swap.

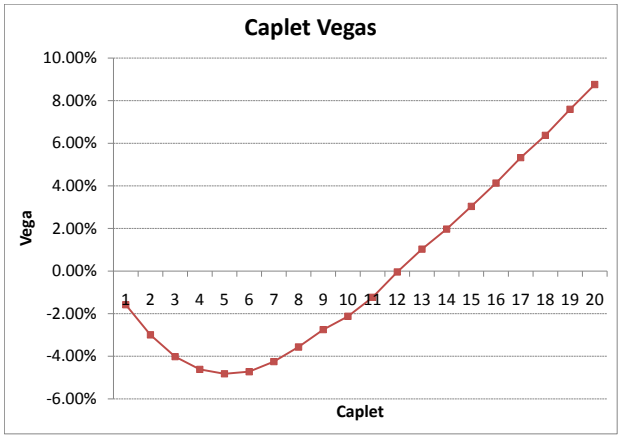

Figure 6.1. *Sensitivity of the cancellable inverse floater swap with respect to caplet volatilities.*

Figure 6.2 shows the vegas when only coterminal swaptions are considered, where the horizontal axis represents swaptions expiring at time T_{k-1} with maturity T_n. Similarly with the caplets we see that the earlier swaptions account for the put optionality feature, while the remaining swaptions account for the cancellability. For this particular market setup,

which effectively results in a one factor model, we could not include both caplets and coterminal swaptions in the hedging portfolio, because the caplets become redundant when the coterminal swaptions are included.

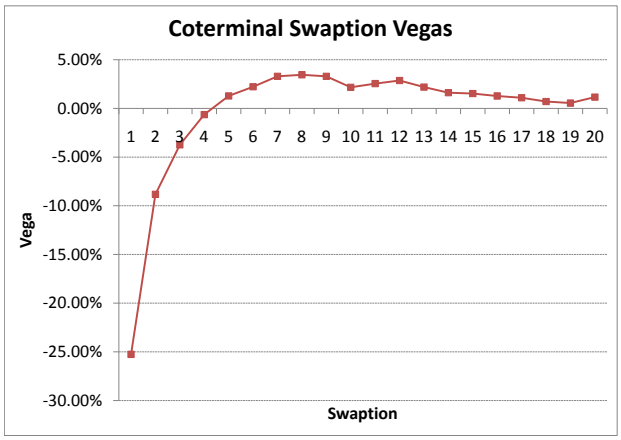

Figure 6.2. *Sensitivity of the cancellable inverse floater swap with respect to coterminal swaption volatilities.*

For a more realistic market scenario we calibrate our two-factor Markov-functional model to a five-factor LMM, using the technique introduced in [8]. The ith caplet volatility in the LMM is given by the $abcd$ volatility form

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

with $a = -0.02, b = 0.3, c = 2, d = 0.2$ and correlation between rates

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

with $L = 0.5, \beta = 0.2$. The calibrated model parameters are given in Table 6.1 and the price is computed to be 2835bps. We use the adjoint method as above to compute the mapping vegas and convert these into caplet and coterminal swaption vegas. The hedging portfolio consists of 20 caplets and 10 swaptions, using every second one, $\text{Swpn}(T_0, T_{20}), \text{Swpn}(T_2, T_{20}), \ldots, \text{Swpn}(T_{18}, T_{20})$.

Figure 6.3 shows both the caplet and coterminal swaption vegas. As we expect the caplets are hedging the put option feature while the swaptions hedge the cancellability. This example demonstrates that the mapping vegas computed using the adjoint method are sufficient to hedge a product with both caplets and coterminal swaptions.

We now consider the impact of the Markov vegas by including them into (6.1) along with the mapping vegas. We choose to include a parallel shift to both diagonal elements, c_{11} and c_{22} as well as a parallel shift to the off diagonal elements c_{12} and c_{21}, resulting in $2n$ extra vegas. These sensitivities were computed using the adjoint approach in Section 4.5.

Figure 6.4 is the same as Figure 6.3, however we see the changes when the Markov vegas are included. The main difference between the two hedging portfolios is the absolute

	ν_0	ν_1		c_{11}	c_{21}	c_{12}	c_{22}
f_0	0.14	-0.33	$[0, T_0)$	0.79	-0.13	0.44	0.52
f_1	0.13	-0.27	$[T_0, T_1)$	0.80	-0.13	0.42	0.56
f_2	0.13	-0.25	$[T_1, T_2)$	0.81	-0.13	0.41	0.59
f_3	0.14	-0.25	$[T_2, T_3)$	0.82	-0.13	0.40	0.62
f_4	0.14	-0.24	$[T_3, T_4)$	0.84	-0.12	0.37	0.65
f_5	0.14	-0.22	$[T_4, T_5)$	0.86	-0.11	0.34	0.69
f_6	0.15	-0.20	$[T_5, T_6)$	0.88	-0.10	0.31	0.73
f_7	0.15	-0.19	$[T_6, T_7)$	0.90	-0.10	0.27	0.77
f_8	0.16	-0.18	$[T_7, T_8)$	0.92	-0.08	0.22	0.83
f_9	0.16	-0.15	$[T_8, T_9)$	0.94	-0.06	0.16	0.90
f_{10}	0.16	-0.12	$[T_9, T_{10})$	0.97	-0.04	0.08	0.96
f_{11}	0.17	-0.11	$[T_{10}, T_{11})$	1.00	0.00	0.00	1.00
f_{12}	0.21	-0.13	$[T_{11}, T_{12})$	1.02	0.09	-0.10	1.01
f_{13}	0.20	-0.10	$[T_{12}, T_{13})$	1.05	0.18	-0.27	1.07
f_{14}	0.20	-0.05	$[T_{13}, T_{14})$	1.05	0.31	-0.50	1.14
f_{15}	0.20	0.01	$[T_{14}, T_{15})$	1.00	0.50	-0.75	1.21
f_{16}	0.18	0.08	$[T_{15}, T_{16})$	0.85	0.77	-1.04	1.30
f_{17}	0.16	0.12	$[T_{16}, T_{17})$	0.43	1.26	-1.53	1.58
f_{18}	0.14	0.14	$[T_{17}, T_{18})$	-3.02	4.49	5.56	-5.20
f_{19}	0.13	0.14	$[T_{18}, T_{19})$	1.76	0.00	0.00	1.00

Table 6.1. *Market volatilities used in Figure 6.3*

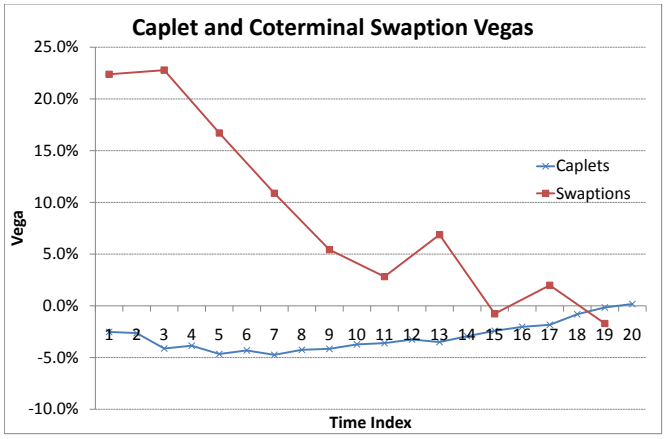

Figure 6.3. *Sensitivity of the cancellable inverse floater swap with respect to caplet and coterminal swaption volatilities.*

value of the vegas. When the Markov vegas are included the absolute value of both the caplet and swaption vegas decrease, which is particularly evident for the earlier swaptions.

Small absolute vegas are preferred in practice, because a smaller bid/ask spread is then paid on the hedging portfolio. This means that including Markov vegas is beneficial, however they take time to compute, so a balance between speed and cost is required for a market implementation.

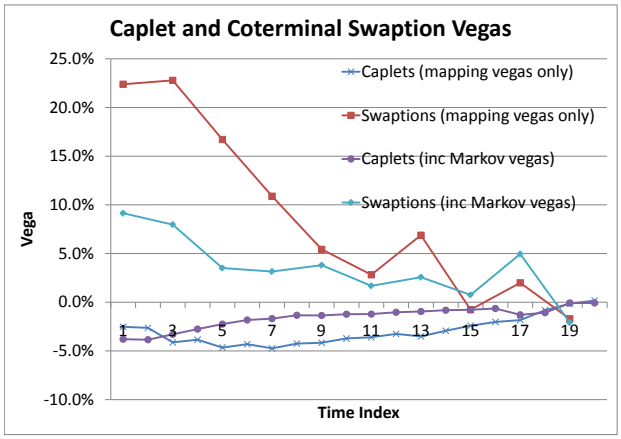

Figure 6.4. *Sensitivity of the cancellable inverse floater swap (with strike = 10%) with respect to caplet and coterminal swaption volatilities, showing the difference caused by including the Markov vegas.*

We also consider another cancellable inverse floater swap, which is identical to the previous one, but with a strike of 18%. Its value is computed to be 334.6 bps which indicates that it is likely to be cancelled earlier than the other inverse floater swap. Figure 6.5 displays the caplet and coterminal swaption vegas in the same way as Figure 6.4. Here we see that the earlier vegas are the largest, because of the likelihood of the product being cancelled early. Also of note is the difference when the Markov vegas are included, which increases the earlier caplet vegas for a reduction in the earlier swaption vegas, however the later caplet and swaptions vegas show only minor changes.

Figure 6.6 shows the vegas for the Bermudan swaption (whose value is computed to be 206.6 bps) with the mapping vegas used to create the graph shown in Table 6.2. Here we see that similarly with the inverse floater, the absolute values of the swaption vegas decrease when the Markov vegas are included. The absolute value of the caplet vegas are similar with and without the Markov vegas. In any case, it is likely that in a practical setting this product would be hedged using only coterminal swaptions.

7. Conclusion

This paper demonstrated that it is possible to use the adjoint PDE method to compute Greeks in Markov-functional models. We showed that it is both accurate and efficient. In the particular case of a separable LIBOR Markov-functional model we were able to accurately compute the price and 80 Greeks of a ten-year product in 1 second. This is the same amount of time that it takes to compute a single Greek using finite difference. We also demonstrated

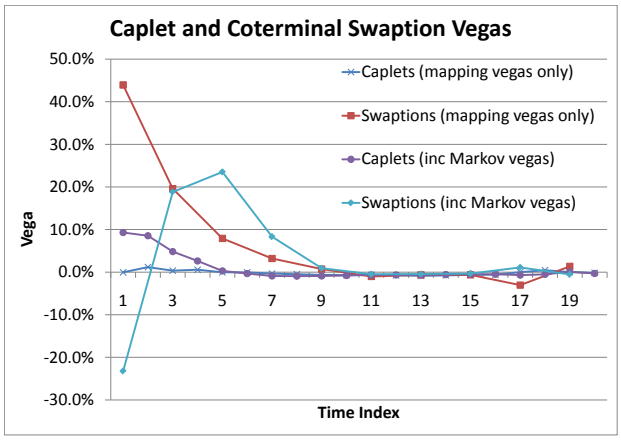

Figure 6.5. *Sensitivity of the cancellable inverse floater swap (with strike = 18%) with respect to caplet and coterminal swaption volatilities.*

Delta		Vega		Vega		Skew	
f_0	-1.0%	$\nu_{0,0}$	0.0%	$\nu_{0,1}$	0.0%	α_0	0.0%
f_1	-1.0%	$\nu_{1,0}$	0.0%	$\nu_{1,1}$	0.0%	α_1	0.0%
f_2	-0.8%	$\nu_{2,0}$	0.0%	$\nu_{2,1}$	0.0%	α_2	0.1%
f_3	-0.3%	$\nu_{3,0}$	0.1%	$\nu_{3,1}$	0.0%	α_3	0.3%
f_4	0.5%	$\nu_{4,0}$	0.3%	$\nu_{4,1}$	-0.1%	α_4	0.7%
f_5	1.4%	$\nu_{5,0}$	0.4%	$\nu_{5,1}$	-0.1%	α_5	1.0%
f_6	2.2%	$\nu_{6,0}$	0.6%	$\nu_{6,1}$	-0.1%	α_6	1.4%
f_7	3.3%	$\nu_{7,0}$	0.8%	$\nu_{7,1}$	-0.1%	α_7	1.8%
f_8	4.3%	$\nu_{8,0}$	1.0%	$\nu_{8,1}$	-0.1%	α_8	2.3%
f_9	5.1%	$\nu_{9,0}$	1.2%	$\nu_{9,1}$	-0.1%	α_9	2.6%
f_{10}	6.0%	$\nu_{10,0}$	1.3%	$\nu_{10,1}$	-0.1%	α_{10}	2.9%
f_{11}	7.2%	$\nu_{11,0}$	1.5%	$\nu_{11,1}$	-0.1%	α_{11}	3.5%
f_{12}	9.2%	$\nu_{12,0}$	1.9%	$\nu_{12,1}$	0.0%	α_{12}	4.9%
f_{13}	10.2%	$\nu_{13,0}$	2.0%	$\nu_{13,1}$	0.1%	α_{13}	5.2%
f_{14}	11.0%	$\nu_{14,0}$	2.1%	$\nu_{14,1}$	0.2%	α_{14}	5.3%
f_{15}	12.2%	$\nu_{15,0}$	2.2%	$\nu_{15,1}$	0.5%	α_{15}	5.7%
f_{16}	13.6%	$\nu_{16,0}$	2.2%	$\nu_{16,1}$	0.7%	α_{16}	6.2%
f_{17}	14.5%	$\nu_{17,0}$	2.1%	$\nu_{17,1}$	1.0%	α_{17}	6.2%
f_{18}	15.5%	$\nu_{18,0}$	1.9%	$\nu_{18,1}$	1.4%	α_{18}	6.2%
f_{19}	15.9%	$\nu_{19,0}$	1.9%	$\nu_{19,1}$	1.4%	α_{19}	6.2%

Table 6.2. *Deltas, vegas and skew sensitivities for a callable Bermudan swaption, strike = 7%. A 101×101 spatial grid was used.*

how the computed model vegas can be converted into market vegas, showing that the adjoint method is a useful practical solution.

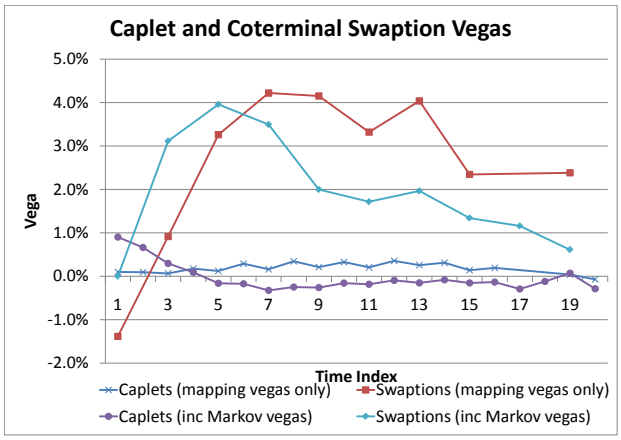

Figure 6.6. *Sensitivity of the callable Bermudan swaption with respect to caplet and coterminal swaption volatilities, showing the difference of including Markov vegas.*

Appendix A. Model Derivatives

The following model derivatives are calculated using the method introduced by Denson and Joshi, [7], which focused on the LIBOR market model. The main difference between that paper and the derivatives below is that we focus on the separable LIBOR Markov-functional model.

A.1. **Delta.** Using the predictor-corrector scheme in log coordinates the forward rates are evolved from time 0 to time T_k using

$$\log(f_i(T_k)+\alpha_i) = \log(f_i(0)+\alpha_i) + \mu_i(0)^{\mathrm{PC}} - \frac{\mathrm{cov}_{ii}}{2} + \nu_i X(T_k), \tag{A.1}$$

where $\mu_i(0)^{\mathrm{PC}}$ is the integrated drift, cov_{ii} is the covariance of the forward rate across the time step and

$$X(T_k) = \int_0^{T_k} C(t)dW(t).$$

To compute the delta we need to differentiate this discretisation scheme with respect to the initial forward rates since

$$\frac{\partial g(f(T_k))}{\partial f(0)} = \frac{\partial g(f(T_k))}{\partial f(T_k)}\frac{\partial f(T_k)}{\partial f(0)}.$$

We do so using the same technique as [7]. This involves splitting up the evolution into a series of maps, with each map updating a vector. The four maps H_i compute the initial

drifts, predicted rates, predicted drifts and then the evolved rates. Graphically we have

$$\left[\ \log(f(0)+\alpha)\ \right] \xrightarrow{H_0} \begin{bmatrix} \log(f(0)+\alpha) \\ \mu(0) \end{bmatrix} \xrightarrow{H_1} \begin{bmatrix} \log(f(0)+\alpha) \\ \mu(0) \\ \hat{f}(0) \end{bmatrix}$$
$$\xrightarrow{H_2} \begin{bmatrix} (\log f(0)+\alpha) \\ \mu(0) \\ \hat{\mu}(0) \end{bmatrix} \xrightarrow{H_3} \left[\ \log(f(T_k)+\alpha)\ \right]. \tag{A.2}$$

We differentiate these maps to compute the Jacobian matrices and then multiply backwards to time 0. The backwards multiplication can be performed with a computational order of nF. This is particularly useful for a Markov model as F is usually small, typically 1 or 2. For the structure of the Jacobian matrices and implementation details we refer the reader to Section 5 of [7].

A.2. **Mapping vegas.** To compute the mapping vegas using the adjoint method we calculate

$$\frac{\partial g(f(T_k))}{\partial \nu_{jf}}.$$

Since we are using a single-step discretisation we can express this sensitivity as

$$\frac{\partial g(f(T_k))}{\partial \nu_{jf}} = \frac{\partial g(f(T_k))}{\partial f(T_k)} \frac{\partial f(T_k)}{\nu_{jf}},$$

where the derivative of the payoff function has already been computed during the calculation of the delta. All that remains is to differentiate the discretisation scheme. Before doing so we need to express the covariances (which appear in the drift (3.4) and drift correction (A.1)) in terms of the forward-rate-specific vectors. The covariance between rates i and j over the interval $[T_a, T_b)$, for $b > a$, is

$$\mathrm{cov}_{ij}(T_a, T_b) = \sum_{l=a}^{b} [\nu_i C(l)][\nu_j C(l)]^T.$$

We now break up the discretisation scheme into four maps in the same way as computing the delta, however we now include the volatility terms

$$\begin{bmatrix} \nu \\ \log(f(0)+\alpha) \end{bmatrix} \xrightarrow{H_0} \begin{bmatrix} \nu \\ \log(f(0)+\alpha) \\ \mu(0) \end{bmatrix} \xrightarrow{H_1} \begin{bmatrix} \nu \\ \log(f(0)+\alpha) \\ \mu(0) \\ \hat{f}(0) \end{bmatrix}$$
$$\xrightarrow{H_2} \begin{bmatrix} \nu \\ \log(f(0)+\alpha) \\ \mu(0) \\ \hat{\mu}(0) \end{bmatrix} \xrightarrow{H_3} \left[\ \log(f(T_k)+\alpha)\ \right].$$

Starting with the last map (since we are working in a backwards direction) we differentiate to give

$$\frac{\partial \log(f_i(T_k)+\alpha_i)}{\partial \nu_{jf}} = \begin{cases} -e_{if}(0,T_k) + X_f(T_k), & \eta(T_k) \le i = j, \\ 0, & \text{otherwise}, \end{cases}$$

where

$$e_{if}(0,T_k) = \sum_{l=0}^{T_k} [\nu_i C(l)] \text{row}(C(l),f)^T,$$

with $\text{row}(C(l),f)$ being the fth row of the matrix $C(l)$ and $\eta(T_k)$ being the index of next forward rate to reset at time T_k. We will need to use these e_{if} terms regularly and since they only involve initial inputs we can pre-compute them in the constructor of our computer program. Focusing on the map H_2 we obtain

$$\frac{\partial \hat{\mu}_i(0)}{\partial \nu_{jf}} = \begin{cases} -\sum_{l=i+1}^{n} \frac{(\hat{f}_l(0)+\alpha_l)\tau_l}{1+\tau_l \hat{f}_l(0)} e_{lf}(0,T_k), & \eta(T_k) \le i = j, \\ -\frac{(\hat{f}_j(0)+\alpha_j)\tau_j}{1+\tau_j \hat{f}_j(0)} e_{if}(0,T_k), & \eta(T_k) \le i < j, \\ 0, & \text{otherwise}. \end{cases} \tag{A.3}$$

For the map H_1 we differentiate to give

$$\frac{\partial \hat{f}_i(0)}{\partial \nu_{jf}} = \begin{cases} (\hat{f}_i(0)+\alpha_i)\left(-e_{if}(0,T_k) + X_f(T_k)\right), & \eta(T_k) \le i = j, \\ 0, & \text{otherwise}. \end{cases}$$

For the final map H_0 the partial derivatives are the same as (A.3), but with $\hat{f}$ replaced by f. By taking advantage of the structure of the Jacobian matrices we can multiply the vega back to time zero in order nF calculations, in the same manner as [7].

A.3. **Markov vegas.** To compute the sensitivity of the price to a Markov vega or $c_{jq}(p)$ term we once again need to differentiate the discretisation scheme. For this subsection we will focus on a two-factor model for ease of exposition. Using the chain rule we have

$$\frac{\partial g(f(T_k))}{\partial c_{jq}(p)} = \frac{\partial g(f(T_k))}{\partial f(T_k)} \frac{\partial f(T_k)}{\partial c_{jq}(p)}.$$

We will split the discretisation scheme into the same four maps as in the above mapping vegas section. Starting with the last map, H_3 we have for $p \le k$

$$\frac{\partial \log(f_i(T_k)+\alpha_i)}{\partial c_{jq}(p)} = \begin{cases} -(\nu_{i0}^2 c_{11}(p) + \nu_{i0}\nu_{i1} c_{21}(p)) + \nu_{i0}(W_1(T_{p+1}) - W_1(T_p)) & j=1, q=1 \\ -(\nu_{i0}^2 c_{12}(p) + \nu_{i0}\nu_{i1} c_{22}(p)) + \nu_{i0}(W_2(T_{p+1}) - W_2(T_p)) & j=1, q=2 \\ -(\nu_{i1}^2 c_{21}(p) + \nu_{i0}\nu_{i1} c_{11}(p)) + \nu_{i1}(W_1(T_{p+1}) - W_1(T_p)) & j=2, q=1 \\ -(\nu_{i1}^2 c_{22}(p) + \nu_{i0}\nu_{i1} c_{12}(p)) + \nu_{i1}(W_2(T_{p+1}) - W_2(T_p)) & j=2, q=2, \end{cases} \tag{A.4}$$

where $W_i(T_{p+1}) - W_i(T_p)$ is the increment of the ith Brownian motion over the interval $[T_p, T_{p+1})$, which arises from (3.1). Focusing our attention on the map H_2 we obtain

$$\frac{\partial \hat{\mu}_i(0)}{\partial c_{jq}(p)} = \begin{cases} -\sum_{l=i+1}^{n-1} \frac{(\hat{f}_l(0)+\alpha_l)\tau_l}{1+\tau_l \hat{f}_l(0)} (2\nu_{i0}\nu_{l0}c_{11}(p) + (\nu_{i0}\nu_{l1} + \nu_{i1}\nu_{l0})c_{21}(p)) & j=1, q=1 \\ -\sum_{l=i+1}^{n-1} \frac{(\hat{f}_l(0)+\alpha_l)\tau_l}{1+\tau_l \hat{f}_l(0)} (2\nu_{i1}\nu_{l1}c_{21}(p) + (\nu_{i0}\nu_{l1} + \nu_{i1}\nu_{l0})c_{11}(p)) & j=1, q=2 \\ -\sum_{l=i+1}^{n-1} \frac{(\hat{f}_l(0)+\alpha_l)\tau_l}{1+\tau_l \hat{f}_l(0)} (2\nu_{i0}\nu_{l0}c_{12}(p) + (\nu_{i0}\nu_{l1} + \nu_{i1}\nu_{l0})c_{22}(p)) & j=2, q=1 \\ -\sum_{l=i+1}^{n-1} \frac{(\hat{f}_l(0)+\alpha_l)\tau_l}{1+\tau_l \hat{f}_l(0)} (2\nu_{i1}\nu_{l1}c_{22}(p) + (\nu_{i0}\nu_{l1} + \nu_{i1}\nu_{l0})c_{12}(p)) & j=2, q=2. \end{cases}$$

For the second map H_1 we differentiate to give

$$\frac{\partial \hat{f}_i(0)}{\partial c_{jq}(p)} = \begin{cases} (\hat{f}_i(0) + \alpha_i)(\cdot), & \eta(T_k) \leq i, \\ 0, & \text{otherwise}, \end{cases}$$

where the $(\cdot)$ term is given by (A.4). For the final map H_0 the derivative is the same as for the map H_2, but with $\hat{f}$ replaced by f so we do not repeat it here. Using these derivatives it is possible to multiply the Markov vega terms back with a computational order of nF, in a similar manner as the mapping vegas.

A.4. **Skew sensitivity.** To compute the sensitivity to a displacement coefficient α_j we use the chain rule

$$\frac{\partial g(f(T_k))}{\partial \alpha} = \frac{\partial g(f(T_k))}{\partial f(T_k)} \frac{\partial f(T_k)}{\partial \alpha}.$$

To differentiate the discretisation scheme it is easier to work with the rates themselves instead of the log-rates. Using the following maps

$$\begin{bmatrix} \alpha \\ f(0) \end{bmatrix} \xrightarrow{H_0} \begin{bmatrix} \alpha \\ f(0) \\ \mu(0) \end{bmatrix} \xrightarrow{H_1} \begin{bmatrix} \alpha \\ f(0) \\ \mu(0) \\ \hat{f}(0) \end{bmatrix} \xrightarrow{H_2} \begin{bmatrix} \alpha \\ f(0) \\ \mu(0) \\ \hat{\mu}(0) \end{bmatrix} \xrightarrow{H_3} \begin{bmatrix} f(T_k) \end{bmatrix},$$

we compute

$$\frac{\partial f_i(T_k)}{\partial \alpha_j} = \begin{cases} \frac{f_i(T_k)+\alpha_i}{f_i(0)+\alpha_i} - 1, & \eta(T_k) \leq i = j, \\ 0, & \text{otherwise}, \end{cases}$$

for H_3. For the previous map H_2 we differentiate to give

$$\frac{\partial \hat{\mu}_i(0)}{\partial \alpha_j} = \begin{cases} \frac{\tau_j}{1+\tau_j \hat{f}_j(0)} a_i a_j^T, & \eta(T_k) \leq i < j, \\ 0, & \text{otherwise}, \end{cases}$$

where a_i is i-th row of the pseudo-square root of the covariance matrix across the time step 0 to T_k. The derivative of the map H_1 is similar to that of H_3, but with f replaced by $\hat{f}$

$$\frac{\partial \hat{f}_i(T_k)}{\partial \alpha_j} = \begin{cases} \frac{\hat{f}_i(T_k)+\alpha_i}{f_i(0)+\alpha_i} - 1, & \eta(T_k) \leq i = j, \\ 0, & \text{otherwise}. \end{cases}$$

Similarly the derivative of H_0 is the same as H_2 with $\hat{f}$ replaced by f

$$\frac{\partial \mu_i(0)}{\partial \alpha_j} = \begin{cases} \frac{\tau_j}{1+\tau_j f_j(0)} a_i a_j^T, & \eta(T_k) \leq i < j, \\ 0, & \text{otherwise.} \end{cases}$$

With these derivatives we are now in a position to calculate the sensitivity of the price with respect to the displacement terms. The calculation of the skew sensitivity can be computed in order nF operations, the same as the delta and vega.

References

[1] M. N. Bennett and J. E. Kennedy. A comparison of Markov-functional and market models: The one-dimensional case. *Journal of Derivatives*, 13(2):22–43, 2005.

[2] F. Black. The pricing of commodity contracts. *Journal of Financial Economics*, 3:167–179, 1976.

[3] A. Brace, D. Gatarek, and M. Musiela. The market model of interest rate dynamics. *Mathematical Finance*, 7(2):127–155, 1997.

[4] Y. Cao, S. Li, L. Petzold, and R. Serban. Adjoint sensitivity analysis for the differential-algerbraic equations: the adjoint DAE system and its numerical solution. *SIAM J. Sci. Comput.*, 24(3):1076–1089, 2003.

[5] J. H. Chan and M. Joshi. Fast Monte-Carlo Greeks for financial products with discountinuous pay-offs. *SSRN*, http://ssrn.com/abstract=1500343, 2010.

[6] I.D.J. Craig and A.D. Sneyd. An alternating-direction implicit scheme for parabolic equations with mixed derivatives. *Comput. Math. Appl.*, 16:341–350, 1988.

[7] N. Denson and M. Joshi. Fast and accurate greeks for the libor market model. *SSRN*, http://ssrn.com/abstract=1448333, 2009.

[8] N. Denson and M. Joshi. Vega control. *SSRN*, http://ssrn.com/abstract=1398523, 2009.

[9] M. Giles and P. Glasserman. Smoking adjoints: fast Monte Carlo Greeks. *Risk*, January:92–96, 2006.

[10] J. Hull and A. White. Forward rate volatilities, swap rate volatilities and the implementation of the LIBOR market model. *Journal of Fixed Income*, 10:46–62, 2000.

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

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

[13] F. Jamshidian. LIBOR and swap market models and measures. *Finance & Stochastics*, 1(4):293, 1997.

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

[15] M. Joshi and O. K. Kwon. Monte Carlo market Greeks in the displaced diffusion LIBOR market model. *SSRN*, http://ssrn.com/abstract=1535058, 2010.

[16] P. Kloeden and Platen E. *Numerical Solution of Stochastic Differential Equations*. Springer-Verlag, 1999.

[17] S. Li and L. Petzold. Adjoint sensitivity analysis for time-dependent partial differential equations with adaptive mesh refinement. *Journal of Computational Physics*, 198:310–325, 2004.

[18] K. R. Miltersen, K. Sandmann, and D. Sondermann. Closed form solutions for term structure derivatives with log-normal interest rates. *The Journal of Finance*, 52(1):409–430, 1997.

[19] I. Najfeld and T. F. Havel. Derivatives of the matrix exponential and their computation. *Advances in Applied Mathematics*, 16(3):321–375, 1995.

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

[21] R. Pietersz, A. Pelsser, and M. Van Regenmortel. Fast drift approximated pricing in the BGM model. *Journal of Computational Finance*, 8(1):93–124, 2003.

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

[23] A. Prideaux. Use of adjoint methods with computational finance PDEs. *Seventh European Workshop on Automatic Differentiation*, 2008.

[24] W. E. Schiesser. *The Numerical Method of Lines*. Academic Press, 1991.

[25] D. Tavella and C. Randall. *Pricing Financial Instruments: The finite difference method*. Wiley, 2000.

Centre for Actuarial Studies, Department of Economics, University of Melbourne VIC 3010, Australia

E-mail address: ndenson@gmail.com

E-mail address: mark@markjoshi.com