FAST DELTA COMPUTATIONS IN THE SWAP MARKET MODEL

MARK JOSHI AND CHAO YANG

ABSTRACT. We develop an efficient algorithm to implement the adjoint method that computes sensitivities of an interest rate derivative (IRD) with respect to different underlying rates in the co-terminal swap market model. The order of computation per step of the new method is shown to be proportional to the number of rates times the number of factors, which is the same as the order in the LIBOR market model.

1. INTRODUCTION

Market models of interest rates where the underlying variables are market-observable interest rates with discrete tenors have become an effective and popular choice of pricing and hedging exotic interest rate derivatives (IRDs). In particular, the LIBOR market model (LMM) developed in Brace et al (1997) has received most attention in both academia and practice. The underlying variables are discretely compounded LIBORs that are assumed to follow log-normal processes. Therefore, calibration to market prices of European caplets is automatic.

Whilst the LMM is very popular, the swap market model (SMM) introduced in Jamshidian (1997) has received far less attention due to its computationally more demanding implementations. However, SMM has certain virtues: exact trivial calibration of co-terminal swaptions; existence of closed-form formulas for European swaptions within the model; and natural adaptness to pricing callable/cancellable products. Gallucio et al (2007) showed that the co-terminal SMM (ctSMM) can be calibrated to both the caplets and the swaption markets in a fast and robust way, and argued that the ctSMM can be regarded as a fundamental pricing and risk-management tool for a large variety of exotic IRDs.

Computing sensitivities (also known as Delta and Vega) under market models is of central importance in risk-management as traders want to set up Delta-neutral or Vega-neutral portfolios that will be immune to any small change of the underlying rates or volatility parameters. One particular virtue of the LMM is the existence of fast methodologies for computing Greeks and it is a natural question as to whether such methods can be applied to the SMM. This paper introduces an efficient algorithm to calculate

Date: December 8, 2009.

1991 *Mathematics Subject Classification.* 91B24, 60G40, 60G44. JEL Classification: G13.

Key words and phrases. adjoint method, Delta, computational order, market model, Monte Carlo simulation.

Deltas under the ctSMM which obtains equal efficiency in terms of computational order to the LMM.

The simplest method to calculate the Deltas of an IRDs is the finite-difference method: one simply bumps each rate one by one and reprices. Multiple Monte Carlo simulations with different inputs are then required to estimate the sensitivities to different parameters. When the underlying number of rates n is large, this method is very time-consuming since we need to run at least $n+1$ simulations.

Various approaches have been adopted to accelerate Greek computations which involve differentiating pay-offs and underlying densities. However, these are still time consuming. A breakthrough was made in Giles and Glasserman (2006). They introduced an adjoint method in the LIBOR market model which allowed all Deltas to be obtained in one Monte Carlo simulation in efficient manner. In particular, the computational complexity of their method was of order equal to the number of rates times the number of factors per step.

Their method was motivated by the theory of algorithmic differentiation: the computational complexity of the adjoint calculation is no more than four times greater than the complexity of the original algorithm (see Griewank (2000)). Since the computational complexity of one step in the LMM is $\mathcal{O}(nF)$, it was therefore clear it could be done. Giles and Glasserman took the formulas for the drifts and worked out the adjoint explicitly. This relied on the fact that fairly simple functional formulas for drifts exist in the LIBOR market model.

An efficient drift computation for the ctSMM is also order $\mathcal{O}(nF)$ per step and so the entire step can be done in order nF time, see Joshi and Liesch (2007). Their approach involved a complicated recursion, however; we review this recursion in Section 2. To develop the adjoint computation it is therefore not as simple as rearranging a few formulas.

Nevertheless, it can be done and our purpose in this paper is to demonstrate an implementable version of the algorithm that is comprehensible, and allows the efficient computation of deltas. In particular, we show constructively that the adjoint method in the co-terminal model can be done with a total order of $\mathcal{O}(nF)$ per step. We follow the efficient implementation of the co-terminal model as in Joshi and Liesch (2007) in terms of drifts and bond ratios computations. Similarly to that paper, we do not try to find the closed-form solution for the derivatives of discretized drifts, but rather look for an efficient algorithm to compute the derivatives. The essential feature of our approach is that we decompose the one-step computation into a number of simpler vector operations. Each of these can be differentiated straightforwardly and then used for multiplication of the adjoint.

This paper is organized as follows. We briefly review the ctSMM and its implementation in section 2. We then review the adjoint method in section 3. In section 4, we first show how the computational order of the adjoint method can be reduced to $\mathcal{O}(nF)$ per step in the co-terminal model, we

then run some timing tests to confirm that the order is $\mathcal{O}(nF)$. We conclude in section 5.

2. The Co-terminal Swap Market Model

2.1. **Notations.** The tenor structure is a finite set of dates

$$0 < T_0 < T_1 < \cdots < T_{n-1} < T_n,$$

where $\{T_i\}_{i=0}^n$ are equally spaced by a real number $\tau = T_{i+1} - T_i$, for all i. We let P_i denote the price of the zero-coupon bond maturing at time T_i. We let SR_i denote the co-terminal swap-rates associated to times $T_i, \ldots, T_n$. We let A_i be the value of the annuity of SR_i, then

$$\mathrm{SR}_i = \frac{P_i - P_n}{A_i}, \tag{2.1}$$

where

$$A_i = \sum_{j \geqslant i} \tau P_{j+1}.$$

2.2. **Model set-up.** The n rates will be driven by F Brownian motions and will be evolved to each of the tenor dates step by step. We assume a piecewise constant volatility structure and therefore assign a pseudo-square root, $A = (a_{i,k})$, of the covariance matrix, C, for each step to determine the evolution. We can therefore write across each step

$$d\mathrm{SR}_i = \mu_i^{(m)} dt + \mathrm{SR}_i \sum_{k=1}^{F} a_{i,k} dZ_k \tag{2.2}$$

where $\mu_i^{(m)}$ is the drift of SR_i under the measure associated with bond P_m, and $\{Z_k\}$ is a sequence of independent Brownian motions. The drifts of the rates are determined by no-arbitrage considerations to ensure that the ratio of every bond price to the *numeraire* bond P_m is a martingale.

2.3. **Pricing.** A product will pay a stream of cash-flows until the product terminates, cancels or is triggered. Each cash-flow may be a function of the entire yield curve at the time it occurs and/or previous times. In practical terms, this means that each cash-flow is a function of the swap-rates underlying the model and this function may also incorporate information from previous times.

We shall concentrate on the case here where there is a single cash-flow which is a function, f, of the prevailing rates at time T_m. However, one should realise that since pricing is linear that this is not a real restriction and that generally a collection of cash-flows at different times would be aggregated along each path of the simulation.

2.4. **Drift computation.**

2.4.1. *The cross variation derivative* $\langle , \rangle$. The cross variation derivative for two Itô processes

$$dX_t = \mu_X(t)dt + \sigma_X\big(X_t, Y_t, t\big)dW_t^X,$$
$$dY_t = \mu_Y(t)dt + \sigma_Y\big(X_t, Y_t, t\big)dW_t^Y$$

is defined to be the coefficient of dt in $dX_t dY_t$. If $dW_t^X dW_t^Y = \rho dt$ then

$$\big\langle X_t, Y_t\big\rangle = \rho\sigma_X\big(X_t, Y_t, t\big)\sigma_Y\big(X_t, Y_t, t\big). \tag{2.3}$$

In particular, the cross variation derivative satisfies the linearity property:

$$\Big\langle X, \sum_i \alpha_i Y_i\Big\rangle = \sum_i \alpha_i \big\langle X, Y_i\big\rangle$$

where X and Y_i are Itô processes and $\alpha_i \in \mathbb{R}$. For detailed properties and derivations, see Joshi and Liesch (2007).

2.4.2. *Drift under the terminal bond measure.* We want to find the drift of SR_i under the measure associated with P_n. Since $\mathrm{SR}_i A_i$, A_i are tradables, then $\mathrm{SR}_i A_i/P_n$ and A_i/P_n are martingales under the terminal bond measure. It follows that the following stochastic differential equation

$$\begin{aligned} d\Big(\frac{\mathrm{SR}_i A_i}{P_n}\Big) &= \frac{A_i}{P_n} d\mathrm{SR}_i + \mathrm{SR}_i d\Big(\frac{A_i}{P_n}\Big) + \Big\langle \mathrm{SR}_i, \frac{A_i}{P_n}\Big\rangle dt \\ &= \Big(\mu_i^{(n)} \frac{A_i}{P_n} + \Big\langle \mathrm{SR}_i, \frac{A_i}{P_n}\Big\rangle\Big) dt \\ &\quad + \frac{A_i}{P_n}\mathrm{SR}_i \sum_{k=1}^{F} a_{i,k} dZ_k + \mathrm{SR}_i d\Big(\frac{A_i}{P_n}\Big). \end{aligned} \tag{2.4}$$

has no drift term, therefore for $i = 0, 1, \ldots, n-2$

$$\begin{aligned} \mu_i^{(n)} &= -\frac{P_n}{A_i}\Big\langle \mathrm{SR}_i, \frac{A_i}{P_n}\Big\rangle \\ &= -\frac{P_n}{A_i}\mathrm{SR}_i \sum_{k=1}^{F} a_{i,k}\Big\langle Z_k, \frac{A_i}{P_n}\Big\rangle, \end{aligned} \tag{2.5}$$

where $\{Z_k\}$ is a sequence of independent Brownian motions. (2.5) is a general formula for the drifts of swap-rates under a market model.

Joshi and Liesch (2007) derived the following recursive formula to compute the cross variation derivatives

$$\Big\langle Z_k, \frac{A_i}{P_n}\Big\rangle = \tau\mathrm{SR}_{i+1} a_{i+1,k} \frac{A_{i+1}}{P_n} + (1 + \tau\mathrm{SR}_{i+1})\Big\langle Z_k, \frac{A_{i+1}}{P_n}\Big\rangle \tag{2.6}$$

with $\big\langle Z_k, A_{n-1}/P_n\big\rangle = 0$, for all k. Using (2.6), we can compute the drifts with order $\mathcal{O}(nF)$ per step.

2.5. **Bond ratios.** The state of the yield curve is expressible in terms of bond prices. We therefore need a fast algorithm to deduce the ratios of bond prices to numeraires from the co-terminal swap rates. We quickly review this from Joshi and Liesch (2007). From (2.1), we have

$$\widetilde{P}_i = \frac{P_i}{P_n} = 1 + \mathrm{SR}_i \frac{A_i}{P_n}. \tag{2.7}$$

We define a sequence $\{\alpha_i\}_{i=0}^{n-1}$ recursively by setting

$$\begin{cases} \alpha_{n-1} = \tau, \\ \alpha_i = \alpha_{i+1} + \tau \widetilde{P}_{i+1}. \end{cases} \tag{2.8}$$

Then it follows that

$$\widetilde{P}_i = 1 + \alpha_i \mathrm{SR}_i, \quad i = 0, 1, \ldots, n-1. \tag{2.9}$$

Under this algorithm, the computational order of deducing the bond ratios is $O(n)$.

The bond prices can be calculated from the bond ratios

$$P_n = \frac{P_0}{\widetilde{P}_0}, \quad P_j = P_0 \frac{\widetilde{P}_j}{\widetilde{P}_0}, \quad j = 1, 2, \ldots, n-1. \tag{2.10}$$

3. ADJOINT METHOD

3.1. **Set-up.** We introduce the adjoint method in a co-terminal model where there are n rates SR_i and P_n is the numeraire bond. Each rate is driven by an F-dimensional Brownian motion

$$\mathrm{SR}_i(T_j) = \mathrm{SR}_i(T_{j-1}) \exp\Big(\mu_i^{(n)}\big(\mathbf{SR}(T_{j-1})\big) - \frac{1}{2}C_{ii} + \sum_{k=1}^{F} a_{ik} Z_k\Big), \tag{3.1}$$

where the discretized drift

$$\mu_i^{(n)} = -\frac{P_n}{A_i} \sum_{k=1}^{F} a_{i,k} \Big\langle Z_k, \frac{A_i}{P_n} \Big\rangle$$

depends on all the alive rates at time T_{j-1} under the terminal bond measure.

We denote the discounted pay-off of an IRD maturing at time T_m, $m \leqslant n$ by

$$g(T_m) = P_n(0) \frac{f(T_m)}{P_n(T_m)}, \tag{3.2}$$

where f is the pay-off function of the IRD at maturity. The Deltas are given by

$$\begin{aligned} \Delta_i &= \frac{\partial}{\partial \mathrm{SR}_i(0)} \mathbb{E}\Big[P_n(0) \frac{f(T_m)}{P_n(T_m)}\Big] \\ &= \frac{\partial P_n(0)}{\partial \mathrm{SR}_i(0)} \mathbb{E}\big[\widetilde{f}(T_m)\big] + P_n(0) \frac{\partial}{\partial \mathrm{SR}_i(0)} \mathbb{E}\big[\widetilde{f}(T_m)\big], \end{aligned} \tag{3.3}$$

where $\widetilde{f}(T_m) = f(T_m)/P_n(T_m)$. Let Δ be a $1 \times n$ gradient vector with the ith entry equal to Δ_i, then

$$\Delta = \frac{\partial P_n(0)}{\partial \mathbf{SR}(0)} \mathbb{E}\big[\widetilde{f}(T_m)\big] + P_n(0) \frac{\partial}{\partial \mathbf{SR}(0)} \mathbb{E}\big[\widetilde{f}(T_m)\big]. \tag{3.4}$$

The differentiation operator and the expectation operator in (3.4) can be interchanged so we have

$$\Delta = \frac{\partial P_n(0)}{\partial \mathbf{SR}(0)} \mathbb{E}\big[\widetilde{f}(T_m)\big] + P_n(0) \mathbb{E}\Big[\frac{\partial \widetilde{f}(T_m)}{\partial \mathbf{SR}(0)}\Big]. \tag{3.5}$$

One of the conditions for this interchange is that f has to be Lipschitz-continuous. For detailed conditions, see Glasserman (2004). The first part in (3.5) can be easily calculated, the second part is computed using the adjoint method in a Monte Carlo simulation.

3.2. **Standard pathwise method with $\mathcal{O}(n^3)$ per step.** We consider the following mappings:

$$\mathbf{SR}(0) \xrightarrow{F_0} \mathbf{SR}(T_0) \xrightarrow{F_1} \cdots \xrightarrow{F_m} \mathbf{SR}(T_m) \xrightarrow{F} \widetilde{f}(T_m). \tag{3.6}$$

Then

$$\widetilde{f}(T_m) = F \circ F_m \circ F_{m-1} \circ \cdots F_0\big(\mathbf{SR}(0)\big). \tag{3.7}$$

The gradient vector $\partial \widetilde{f}(T_m)/\partial \mathbf{SR}(0)$ is given by

$$\frac{\partial \widetilde{f}(T_m)}{\partial \mathbf{SR}(0)} = F' F'_m F'_{m-1} \cdots F'_0. \tag{3.8}$$

where F' is a $1 \times n$ vector and $F'_k = \partial \mathbf{SR}(T_k)/\partial \mathbf{SR}(T_{k-1})$ is an $n \times n$ Jacobian matrix. Therefore, if we carry out matrix multiplications from F'_0 to F'_m in (3.8) then the order of computation per step is $\mathcal{O}(n^3)$. (Note that this assumes that we are using the naive algorithm for multiplying matrices but "fast" methodologies are not much better.)

3.3. **Standard adjoint method with $\mathcal{O}(n^2)$ per step.** We can reduce the order of computation by a factor of n per step through the adjoint method. We define an adjoint relation as follows:

$$\mathbf{V}(T_m) = F' \quad \text{and} \quad \mathbf{V}(T_{k-1}) = \mathbf{V}(T_k) F'_k. \tag{3.9}$$

Then it follows from (3.9) that

$$\frac{\partial \widetilde{f}(T_m)}{\partial \mathbf{SR}(0)} = \mathbf{V}(0). \tag{3.10}$$

The computational order per step is $\mathcal{O}(n^2)$ as we are multiplying a vector by a Jacobian matrix.

The elements of the $\mathbf{V}$ can be computed as follows

$$\begin{aligned}\mathbf{V}_i(T_{k-1}) &= \sum_{j\geqslant i} \mathbf{V}_j(T_k)\frac{\partial \mathrm{SR}_j(T_k)}{\partial \mathrm{SR}_i(T_{k-1})} \\ &= \mathbf{V}_i(T_k)\frac{\mathrm{SR}_i(T_k)}{\mathrm{SR}_i(T_{k-1})} + \sum_{j\geqslant i} \mathbf{V}_j(T_k)\mathrm{SR}_j(T_k)\frac{\partial \mu_j^{(n)}\big(\mathbf{SR}(T_{k-1})\big)}{\partial \mathrm{SR}_i(T_{k-1})}.\end{aligned} \tag{3.11}$$

To do better requires careful study of the structure of the partial derivatives $\partial \mu_j^{(n)}\big(\mathbf{SR}(T_{k-1})\big)/\partial \mathrm{SR}_i(T_{k-1})$. In the next section, we show that in fact order $\mathcal{O}(nF)$ per step can be achieved. For F small and n large, this will result in substantial time savings.

4. Efficient adjoint method with $\mathcal{O}(nF)$ per step

We divide the mapping F_k in (3.6) into four sub-mappings:

$$\begin{aligned}\mathbf{SR}(T_{k-1}) \xrightarrow{F_{k,0}} \begin{bmatrix}\mathbf{SR}(T_{k-1})\\ \mathbf{P}(T_{k-1})\end{bmatrix} \xrightarrow{F_{k,1}} \begin{bmatrix}\mathbf{SR}(T_{k-1})\\ \mathbf{P}(T_{k-1})\\ \mathbf{A}(T_{k-1})\end{bmatrix} \\ \xrightarrow{F_{k,2}} \begin{bmatrix}\mathbf{SR}(T_{k-1})\\ \underline{\mu}^{(n)}(T_{k-1})\end{bmatrix} \xrightarrow{F_{k,3}} \mathbf{SR}(T_k),\end{aligned} \tag{4.1}$$

where bold letters indicate that we are considering vectors. The sub-mappings satisfy

$$F_k = F_{k,0}\circ F_{k,1}\circ F_{k,2}\circ F_{k,3}, \quad k = 0, 1, \ldots, m,$$

where each sub-mapping $F_{k,i}$, $i = 0, 1, 2, 3$, can be divided into further sub-sub-mappings. Under this formulation, the Jacobian matrices $F'_{k,i}$ become sparse matrices that is most of their entries are zero. It is important to note that we will not carry out matrix multiplication when computing

$$\mathbf{v} = \mathbf{w}F'_{k,i}; \tag{4.2}$$

instead, we identify the non-zero elements of $F'_{k,i}$ and only multiply the corresponding entries in $\mathbf{w}$ with these non-zero entries. The order of computation of the operation in (4.2) will either be $O(n)$ or $\mathcal{O}(nF)$. Thus the computational order of

$$\mathbf{v} = \mathbf{w}F'_{k,3}F'_{k,2}F'_{k,1}F'_{k,0} = \mathbf{w}F'_k$$

is $\mathcal{O}(nF)$. Therefore, we can reduce the order of the adjoint method to $\mathcal{O}(nF)$ per step in the co-terminal model.

For concreteness and readability, we will only show the computational order of the operations

$$\mathbf{v} = \mathbf{w}F'_{0,3}F'_{0,2}F'_{0,1}F'_{0,0} = \mathbf{w}F'_0. \tag{4.3}$$

i.e. the adjoint operations for the first step. The adjoint operations when $k = 1, \ldots, m$, are essentially the same: the only real difference is that certain

rates will have fixed and therefore will have zero volatility during these steps.

4.1. **Computational order of $\mathbf{w}F'_{0,0}$.** We divide the mapping

$$\mathbf{SR}(0) \xrightarrow{F_{0,0}} \begin{bmatrix} \mathbf{SR}(0) \\ \mathbf{P}(0) \end{bmatrix}$$

into the following sub-mappings

$$\begin{aligned}
\mathbf{SR}(0) &\xrightarrow{G_{0,0}} \begin{bmatrix} \mathbf{SR}(0) \\ \alpha_{n-1} \end{bmatrix} \xrightarrow{G_{0,1}} \begin{bmatrix} \mathbf{SR}(0) \\ \alpha_{n-1} \\ \widetilde{P}_{n-1}(0) \end{bmatrix} \xrightarrow{G_{0,2}} \begin{bmatrix} \mathbf{SR}(0) \\ \alpha_{n-2} \\ \widetilde{P}_{n-1}(0) \end{bmatrix} \xrightarrow{G_{0,3}} \begin{bmatrix} \mathbf{SR}(0) \\ \alpha_{n-2} \\ \widetilde{P}_{n-1}(0) \\ \widetilde{P}_{n-2}(0) \end{bmatrix} \\
&\xrightarrow{G_{0,4}} \begin{bmatrix} \mathbf{SR}(0) \\ \alpha_{n-3} \\ \widetilde{P}_{n-1}(0) \\ \widetilde{P}_{n-2}(0) \end{bmatrix} \cdots \xrightarrow{G_{0,2n-2}} \begin{bmatrix} \mathbf{SR}(0) \\ \alpha_0 \\ \widetilde{P}_{n-1}(0) \\ \widetilde{P}_{n-2}(0) \\ \vdots \\ \widetilde{P}_1 \end{bmatrix} \xrightarrow{G_{0,2n-1}} \begin{bmatrix} \mathbf{SR}(0) \\ \widetilde{P}_{n-1}(0) \\ \widetilde{P}_{n-2}(0) \\ \vdots \\ \widetilde{P}_1(0) \\ \widetilde{P}_0(0) \end{bmatrix} \\
&\xrightarrow{G_0} \begin{bmatrix} \mathbf{SR}(0) \\ \mathbf{P}(0) \end{bmatrix}
\end{aligned} \tag{4.4}$$

The even-numbered mappings $G_{0,j}$ update α's and the odd-numbered mappings $G_{0,j}$ computes a new $\widetilde{P}$ term.

4.1.1. *Jacobian matrices of $G_{0,j}$ where j is even.* From (2.8), we can see that α_i only depends on α_{i+1} and $\widetilde{P}_{i+1}$, then

$$\frac{\partial \alpha_i}{\partial \alpha_{i+1}} = 1, \quad \frac{\partial \alpha_i}{\partial \widetilde{P}_{i+1}} = \tau.$$

Then the Jacobian matrix $G'_{0,j}$ has 1s on the diagonal, one entry equal to τ on the last column, and 0s elsewhere. Therefore the operation $\mathbf{v} = \mathbf{w}G'_{0,j}$ is trivial. It has one computation for each even-numbered j.

4.1.2. *Jacobian matrices of $G_{0,j}$ where j is odd.* From (2.9), we can see that $\widetilde{P}_i$ only depends on α_i and SR_i, then

$$\frac{\partial \widetilde{P}_i}{\partial \alpha_i} = \mathrm{SR}_i, \quad \frac{\partial \widetilde{P}_i}{\partial \mathrm{SR}_i} = \alpha_i.$$

Then the Jacobian matrix $G'_{0,j}$ has 1s on the diagonal, two entries equal to SR_i and α_i on the last row. Therefore the operation $\mathbf{v} = \mathbf{w}G'_{0,j}$ is trivial. It has two computations for each odd-numbered j.

4.1.3. *Jacobian matrix of G_0.* From (2.10), we can see that P_n depends on $\widetilde{P}_0$, and P_i depend on $\widetilde{P_0}$ and $\widetilde{P}_i$, $i = 1, \ldots, n-1$, then

$$\begin{aligned}\frac{\partial P_i}{\partial \widetilde{P_0}} &= -\frac{P_i}{\widetilde{P_0}}, \quad i = 1, \ldots, n,\\ \frac{\partial P_i}{\partial \widetilde{P_i}} &= -\frac{P_i}{\widetilde{P_i}}, \quad i = 1, \ldots, n-1.\end{aligned}$$

Then the Jacobian matrix G_0' has the following general form

$$\begin{bmatrix}
1 & & & & & & & & \\
 & 1 & & & & & & & \\
 & & 1 & & & & & & \\
 & & & 1 & & & & & \\
 & & & & 1 & & & & \\
 & & & & & & & X & X \\
 & & & & & & X & & X \\
 & & & & & X & & & X \\
 & & & & X & & & & X \\
 & & & & & & & & X
\end{bmatrix}$$

where the blanks are 0s, the n entries on the last column are equal to $-P_i/\widetilde{P}_0$, the rest $n-1$ non-zero entries are equal to $-P_i/\widetilde{P}_i$. Therefore the operation $\mathbf{v} = \mathbf{w}G_0'$ has order $O(n)$.

4.1.4. *Total computational order of $\mathbf{w}F_{0,0}'$.* The number of the sub-sub-mappings $G_{0,j}$ depends on n, then the computational order of the operation

$$\mathbf{v} = \mathbf{w}G_{0,2n-1}'G_{0,2n-2}'\cdots G_{0,1}'$$

is clearly $O(n)$ since each operation has a constant computational order. We ignore the Jacobian matrix $G_{0,0}'$ as α_{n-1} is a constant so that $G_{0,0}'$ is equivalent to an identity matrix.

Hence the operation $\mathbf{w}G_0'G_{0,2n-1}'G_{0,2n-2}'\cdots G_{0,1}' = \mathbf{w}F_{0,0}'$ has order $O(n)$.

4.2. **Computational order of $\mathbf{w}F_{0,1}'$.** The mapping

$$\begin{bmatrix}\mathbf{SR}(0)\\ \mathbf{P}(0)\end{bmatrix} \xrightarrow{F_{0,1}} \begin{bmatrix}\mathbf{SR}(0)\\ \mathbf{P}(0)\\ \mathbf{A}(0)\end{bmatrix}$$

has Jacobian matrix $F'_{0,1}$ of the general form

$$\begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & 1 & & & & \\ & & & 1 & & & \\ & & & & 1 & & \\ & & & & & 1 & \\ & & & \tau & \tau & \tau & \\ & & & & \tau & \tau & \\ & & & & & \tau & \end{bmatrix}, \tag{4.5}$$

since

$$\frac{\partial A_i}{\partial P_j} = \tau, \quad j > i.$$

In particular, we have $2n$ 1s and $n-j$ τs in the $(2n-j)$th column.

Although the operation $\mathbf{v} = \mathbf{w}F'_{0,1}$ seems to have order $\mathcal{O}(n^2)$, we can reduce it to $O(n)$ due to the special structure of $F'_{0,1}$. We define a sum variable, and update it as follows:

- Set sum equal to $\tau \mathbf{w}_{2n}$ in loop 0.
- Update sum to $sum + \tau \mathbf{w}_{2n+j}$ in loop j.

If we add the updated sum to $\mathbf{v}_{n+j}$ in the jth loop, then we have executed the operation $\mathbf{v} = \mathbf{w}F'_{0,1}$. Hence, the order of the operation is $O(n)$.

4.3. **Computational order of $\mathbf{w}F'_{0,2}$.** We divide the mapping

$$\begin{bmatrix} \mathbf{SR}(0) \\ \mathbf{P}(0) \\ \mathbf{A}(0) \end{bmatrix} \xrightarrow{F_{0,2}} \begin{bmatrix} \mathbf{SR}(0) \\ \underline{\mu}^{(n)}(0) \end{bmatrix}$$

into the following sub-mappings

$$\begin{bmatrix} \mathbf{SR}(0) \\ \mathbf{P}(0) \\ \mathbf{A}(0) \end{bmatrix} \xrightarrow{H_{0,0}} \begin{bmatrix} \mathbf{SR}(0) \\ \mathbf{P}(0) \\ \mathbf{A}(0) \\ \left\langle Z_k, \frac{A_{n-2}}{P_n} \right\rangle \end{bmatrix} \xrightarrow{H_{0,1}} \begin{bmatrix} \mathbf{SR}(0) \\ \mathbf{P}(0) \\ \mathbf{A}(0) \\ \left\langle Z_k, \frac{A_{n-2}}{P_n} \right\rangle \\ \left\langle Z_k, \frac{A_{n-3}}{P_n} \right\rangle \end{bmatrix}$$

$$\cdots \xrightarrow{H_{0,n-2}} \begin{bmatrix} \mathbf{SR}(0) \\ \mathbf{P}(0) \\ \mathbf{A}(0) \\ \left\langle Z_k, \frac{A_{n-2}}{P_n} \right\rangle \\ \left\langle Z_k, \frac{A_{n-3}}{P_n} \right\rangle \\ \vdots \\ \left\langle Z_k, \frac{A_0}{P_n} \right\rangle \end{bmatrix} \xrightarrow{H_0} \begin{bmatrix} \mathbf{SR}(0) \\ \underline{\mu}^{(n)}(0) \end{bmatrix} \tag{4.6}$$

4.3.1. *Jacobian matrix of $H_{0,j}$.* The dimension of $H'_{0,j}$ is $\big(3n+(j+1)F\big)\times\big(3n+jF\big)$. According to (2.6), $\big\langle Z_k, A_j/P_n\big\rangle$ depend on SR_{j+1}, A_{j+1}, P_n and $\big\langle Z_k, A_{j+1}/P_n\big\rangle$, then

$$\begin{aligned}
\frac{\partial}{\partial \mathrm{SR}_{j+1}}\big\langle Z_k, A_j/P_n\big\rangle &= \tau\bigg(a_{j+1,k}\frac{A_{j+1}}{P_n}+\Big\langle Z_k,\frac{A_{j+1}}{P_n}\Big\rangle\bigg),\\
\frac{\partial}{\partial A_{j+1}}\big\langle Z_k, A_j/P_n\big\rangle &= \tau\mathrm{SR}_{j+1}a_{j+1,k}\frac{1}{P_n},\\
\frac{\partial}{\partial P_n}\big\langle Z_k, A_j/P_n\big\rangle &= -\tau\mathrm{SR}_{j+1}a_{j+1,k}\frac{A_{j+1}}{P_n^2},\\
\frac{\partial}{\partial\big\langle Z_k, A_{j+1}/P_n\big\rangle}\big\langle Z_k, A_j/P_n\big\rangle &= 1+\tau\mathrm{SR}_{j+1},
\end{aligned}$$

for $k=1,\dots,F$. Therefore $H'_{0,j}$ has 1s on the diagonals, and 4 entries equal to the above partial derivatives on each of the last F rows. Therefore each of the operations

$$\mathbf{v}=\mathbf{w}H'_{0,j},\quad j=0,1,\dots,n-2$$

has $4F$ computations.

4.3.2. *Jacobian matrix of H_0.* According to (2.5), each $\mu_j^{(n)}$ depends on A_j, P_n and $\big\langle Z_k, A_j/P_n\big\rangle$, then

$$\begin{aligned}
\frac{\partial}{\partial A_j}\mu_j^{(n)} &= -\frac{\mu_j}{A_j},\\
\frac{\partial}{\partial P_n}\mu_j^{(n)} &= \frac{\mu_j}{P_n},\\
\frac{\partial}{\partial\big\langle Z_k, A_j/P_n\big\rangle}\mu_j^{(n)} &= -\frac{P_n}{A_j}a_{jk},
\end{aligned}$$

for $j=0,\dots,n-2$ and $k=1,\dots,F$.

The Jacobian matrix H'_0 has $n-1$ partial derivatives equal to $\partial\mu_j^{(n)}/\partial P_n$, $n-1$ partial derivatives equal to $\partial\mu_j^{(n)}/\partial A_j$, and $(n-1)F$ partial derivatives equal to $\partial\mu_j^{(n)}/\partial\big\langle Z_k, A_j/P_n\big\rangle$. Therefore the operation $\mathbf{v}=\mathbf{w}H'_0$ has order $\mathcal{O}(nF)$.

4.3.3. *Total computational order of $\mathbf{w}F'_{0,2}$.* The number of sub-sub-mappings $H_{0,j}$ depends on n, therefore the order of

$$\mathbf{v}=\mathbf{w}H'_{0,n-2}H'_{0,n-1}\cdots H'_{0,1}H'_{0,0}$$

is $\mathcal{O}(nF)$ since each operation has a constant order. Hence, the order of the operation $\mathbf{v}=\mathbf{w}H'_0H'_{0,n-2}\cdots H'_{0,0}=\mathbf{w}F'_{0,2}$ is $\mathcal{O}(nF)$.

4.4. **Computational order of $\mathbf{w}F'_{0,3}$.** The dimension of $F'_{0,3}$ is $n \times 2n$. From (3.1)

$$\frac{\partial \mathrm{SR}_j(T_0)}{\partial \mathrm{SR}_j(0)} = \frac{\mathrm{SR}_j(T_0)}{\mathrm{SR}_j(0)}, \quad j = 0, \ldots, n-1$$
$$\frac{\partial \mathrm{SR}_j(T_0)}{\partial \mu_j^{(n)}} = \mathrm{SR}_j(T_0), \quad j = 0, \ldots, n-2.$$

Then, $F'_{0,3}$ has the general form

$$\begin{bmatrix} \frac{SR_0(T_0)}{SR_0(0)} & & & & \mathrm{SR}_0(T_0) & & & \\ & \ddots & & & & \ddots & & \\ & & \frac{SR_{n-2}(T_0)}{SR_{n-2}(0)} & & & & \mathrm{SR}_{n-2}(T_0) & \\ & & & \frac{SR_{n-1}(T_0)}{SR_{n-1}(0)} & & & & 0 \end{bmatrix},$$

where the blanks are zero and the 0 indicates that the swap-rate SR_{n-1} does not have drift under the terminal measure. The Jacobian matrix $F'_{0,3}$ has $2n-1$ non-zero entries. Hence the operation $\mathbf{v} = \mathbf{w}F'_{0,3}$ has $2n-1$ computations so that its order is $O(n)$.

4.5. **Total computational order of $\mathbf{w}F'_k$.** We have shown that the computational order of $\mathbf{w}F'_{0,i}$ is either $O(n)$ or $\mathcal{O}(nF)$. Hence the total computational order of

$$\mathbf{v} = \mathbf{w}F'_{0,3}F'_{0,2}F'_{0,1}F'_{0,0} = \mathbf{w}F'_0$$

is $\mathcal{O}(nF)$. Since the computational order of $\mathbf{w}F'_{k,i}$ is similar to that of $\mathbf{w}F'_{0,i}$, we have shown that the operation

$$\mathbf{v} = \mathbf{w}F'_{k,3}F'_{k,2}F'_{k,1}F'_{k,0} = \mathbf{w}F'_k$$

is $\mathcal{O}(nF)$.

4.6. **Computational order of F'.** We show that the gradient matrix F' can be computed in order $O(n)$. We divide the mapping F in (3.6) into the following sub-mappings:

$$\mathbf{SR}(T_m) \xrightarrow{I_0} \begin{bmatrix} \mathbf{SR}(T_m) \\ \mathbf{P}(T_m) \end{bmatrix} \xrightarrow{I_1} \begin{bmatrix} \mathbf{SR}(T_m) \\ \mathbf{P}(T_m) \\ \mathbf{A}(T_m) \end{bmatrix} \xrightarrow{I_2} \widetilde{f}(T_m). \tag{4.7}$$

We have shown in sections 4.1 and 4.2 that the operations $\mathbf{v} = \mathbf{w}I'_j$, $j = 0, 1$, have order $O(n)$. The computation of the gradient vector I'_2 is trivial. Hence, the order of computing $I'_2 I'_1 I'_0 = F'$ is $O(n)$.

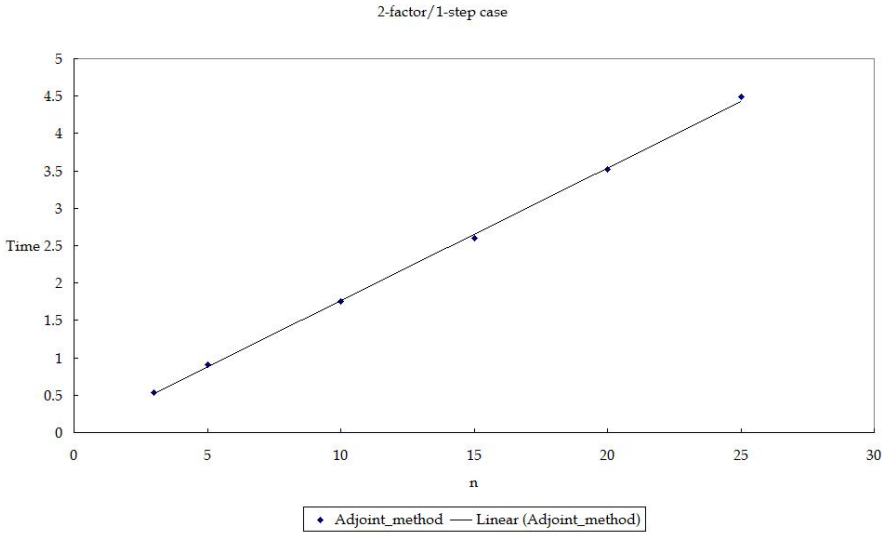

FIGURE 1. Graphs of n against time of estimating Deltas of a T_0 European swaption with $F = 2$.

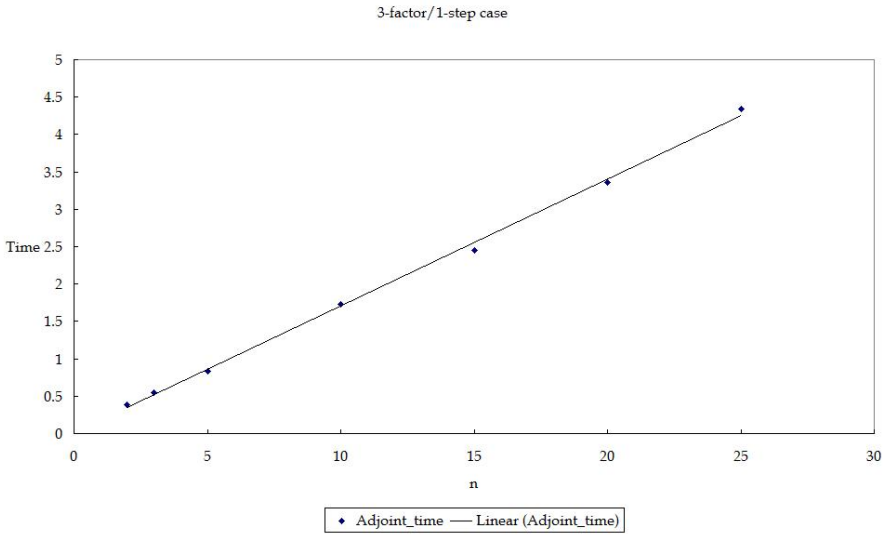

FIGURE 2. Graphs of n against time of estimating Deltas of a T_0 European swaption with $F = 3$.

4.7. **Timing tests.** We have shown that the order of the adjoint method is $\mathcal{O}(nF)$ per step. If we carry out the algorithm for 1 step, we should obtain timings that are linear in n and F. If we carry out the the algorithm for n steps, we should obtain timings that are parabolic in n.

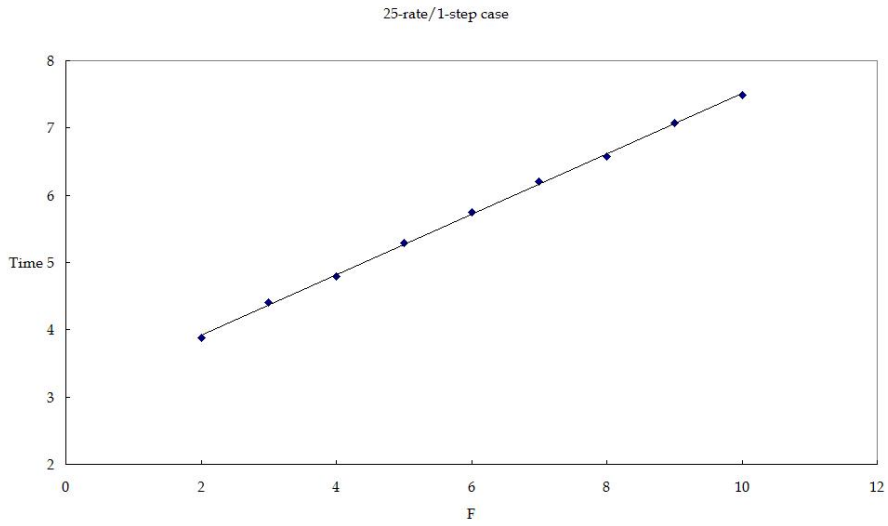

FIGURE 3. Graphs of F against time of estimating Deltas of a T_0 European swaption with $n = 25$.

n	Time	Linear Fit
2	0.391	0.351
3	0.547	0.520
5	0.828	0.860
10	1.735	1.709
15	2.453	2.557
20	3.359	3.406
25	4.344	4.255

TABLE 4.1. Timings for estimating Deltas of a T_0 European swaption with $F = 2$.

n	Time	Linear Fit
3	0.531	0.518
5	0.906	0.874
10	1.750	1.763
15	2.594	2.653
20	3.516	3.542
25	4.484	4.431

TABLE 4.2. Timings for estimating Deltas of a T_0 European swaption with $F = 3$.

In each of the following cases, we have n co-terminal swap-rates, each with the reset date $T_j = (j + 1)0.5$, $j = 0, 1, \ldots, n - 1$, and driven by F

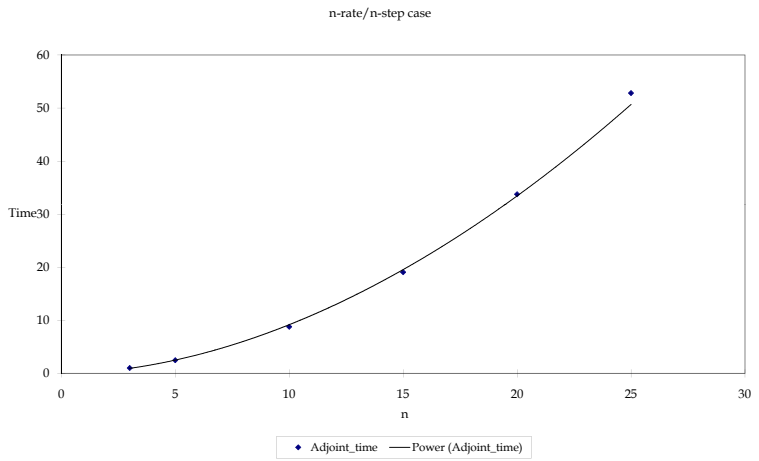

FIGURE 4. Graphs of n against time of estimating Deltas of a T_{n-1} European swaption with $F = 3$.

n	Time	Parabolic Fit
3	1.015	1.105
5	2.485	2.423
10	8.796	8.673
15	19.078	19.148
20	33.766	33.848
25	52.828	52.771

TABLE 4.3. Timings for estimating Deltas of a T_{n-1} European swaption with $F = 3$.

factors. We run 163,840 paths on a European swaption and estimate Deltas using the adjoint method.

- We estimate Deltas of a European swaption with maturity T_0. We fix F and plot time against n. We show the graph in figures 1 and 2, and we display the values of a fitted line through timings in tables 4.1 and 4.2.
- We estimate Deltas of a European swaption with maturity T_0. We fix n and plot time against F. We show the graph in figure 3.

- We estimate Deltas of a European swaption with maturity T_{n-1}. We then fix F and plot time against n. We show the graph in figure 4, and we display the values of a fitted parabola through timings in table 4.3.

4.8. **General numeraire.** So far we have been working under the terminal bond measure. However, it is not too difficult to work under any arbitrary bond measure.

First, (3.5) needs to be rewritten as

$$\Delta = \frac{\partial P_N(0)}{\partial \mathbf{SR}(0)} \mathbb{E}\big[\widetilde{f}(T_m)\big] + P_N(0)\mathbb{E}\Big[\frac{\partial \widetilde{f}(T_m)}{\partial \mathbf{SR}(0)}\Big], \tag{4.8}$$

where $\widetilde{f}(T_m) = f(T_m)/P_N(T_m)$ with P_N being the numeraire bond.

Next, we adopt the change of measure formula (3.11) in Joshi and Liesch (2007):

$$\mu_i^{(N)} = \mu_i^{(n)} + \frac{P_n}{\tau P_N} \sum_{k=1}^{F} a_{ik} \left(\left\langle Z_k, \frac{A_{N-1}}{P_n} \right\rangle - \left\langle Z_k, \frac{A_N}{P_n} \right\rangle \right). \tag{4.9}$$

We then divide the mapping (4.6) as follows:

$$\begin{bmatrix} \mathbf{SR}(0) \\ \mathbf{P}(0) \\ \mathbf{A}(0) \end{bmatrix} \xrightarrow{H_{0,0}} \begin{bmatrix} \mathbf{SR}(0) \\ \mathbf{P}(0) \\ \mathbf{A}(0) \\ \left\langle Z_k, \frac{A_{n-2}}{P_n} \right\rangle \end{bmatrix} \cdots \xrightarrow{H_{0,n-2}} \begin{bmatrix} \mathbf{SR}(0) \\ \mathbf{P}(0) \\ \mathbf{A}(0) \\ \left\langle Z_k, \frac{A_{n-2}}{P_n} \right\rangle \\ \left\langle Z_k, \frac{A_{n-3}}{P_n} \right\rangle \\ \vdots \\ \left\langle Z_k, \frac{A_0}{P_n} \right\rangle \end{bmatrix}$$

$$\xrightarrow{H_0} \begin{bmatrix} \mathbf{SR}(0) \\ \mathbf{P}(0) \\ \left\langle Z_k, \frac{A_{n-2}}{P_n} \right\rangle \\ \left\langle Z_k, \frac{A_{n-3}}{P_n} \right\rangle \\ \vdots \\ \left\langle Z_k, \frac{A_0}{P_n} \right\rangle \\ \underline{\mu}^{(n)}(0) \end{bmatrix} \xrightarrow{H_1} \begin{bmatrix} \mathbf{SR}(0) \\ \underline{\mu}^{(N)}(0) \end{bmatrix} \tag{4.10}$$

From (4.9), the non-zero entries of the Jacobian matrix H_1' can be easily computed. In particular, H_1' has

- $n-1$ partial derivatives equal to $\partial \mu_i^{(N)}/\partial P_n$,
- $n-1$ partial derivatives equal to $\partial \mu_i^{(N)}/\partial P_N$,
- $(n-1)F$ partial derivatives equal to $\partial \mu_i^{(N)}/\partial \langle Z_k, \frac{A_{N-1}}{P_n} \rangle$,
- $(n-1)F$ partial derivatives equal to $\partial \mu_i^{(N)}/\partial \langle Z_k, \frac{A_N}{P_n} \rangle$.

Thus, the order of the operation $\mathbf{v} = \mathbf{w}H_1'$ is $\mathcal{O}(nF)$. Therefore, working under the general measure will not change the order of the adjoint method in the co-terminal model.

5. CONCLUSION

It is possible to compute the Deltas of a product in the swap market model by using the adjoint method and obtain a computational complexity of order nF per step. The key to our approach is that instead of attempting to develop and differentiate formulas for drifts, we divide all the computations into a sequence of vector operations each of which is easily differentiated and of low computational order.

BIBLIOGRAPHY

[1] A. Brace, D. Gatarek and M. Musiela. (1997). The market model of interest rate dynamics. Mathematical Finance, 7, 127–155.
[2] S. Galluccio, J. M. Ly, Z. Huang and O. Scaillet. (2007). Theory and calibration of swap market models. Mathematical Finance, 17, 111–141.
[3] M. Giles and P. Glasserman. (2006). Smoking adjoints: fast Monte Carlo Greeks. Risk, January 2006, 92–96.
[4] P. Glasserman and X. Zhao. (1999). Fast Greeks by simulation in forward LIBOR models. Journal of Computational Finance, 3, 5–39.
[5] P. Glasserman. (2004). *Monte Carlo Methods in Financing Engineering*. Springer-Verlag, Berlin-Heidelberg-New York.
[6] A. Griewank. (2000) *Evaluating derivatives: principles and techniques of algorithmic differentiation.* Society for Industrial and Applied Mathematics.
[7] F. Jamshidian. (1997). LIBOR and swap market models and measures. Finance and Stochastics, 1, 293–330.
[8] M. Joshi. (2003). *The Concepts and Practice of Mathematical Finance.* Cambridge University Press.
[9] M. Joshi and L. Liesch (2007). Effective implementation of generic market models. Astin Bulletin, 37(2), 453–473.

CENTRE FOR ACTUARIAL STUDIES, DEPARTMENT OF ECONOMICS, UNIVERSITY OF MELBOURNE, VICTORIA 3010, AUSTRALIA
E-mail address: mark@markjoshi.com
E-mail address: c.yang7@pgrad.unimelb.edu.au