Algorithmic Finance 1 (2011) 17–33
DOI 10.3233/AF-2011-002
IOS Press

Efficient greek estimation in generic swap-rate market models

Mark Joshi and Chao Yang
Centre for Actuarial Studies, Department of Economics, University of Melbourne, Victoria 3010, Australia
E-mail: mark@markjoshi.com;chaoyangps@gmail.com

Abstract. We first develop an efficient algorithm to compute Deltas of interest rate derivatives for a number of standard market models. The computational complexity of the algorithms is shown to be proportional to the number of rates times the number of factors per step. We then show how to extend the method to efficiently compute Vegas in those market models.

Keywords: adjoint method, Delta, Vega, computational order, market model, Monte Carlo simulation.

1. Introduction

Market models have become a standard tool for pricing exotic interest rate derivatives (IRDs). The most popular of these is the LIBOR market model (LMM) also known as BGM, Brace et al. (1997). In the LMM the underlying variables are market-observable LIBORs with discrete tenors. By construction, the model can reproduce the forward yield curve perfectly and is able to justify the use of Black's formula for European IRDs. Thus calibration of the LMM to the market prices of caplets is automatic. Jamshidian (1997) introduced another class of market models, namely, the (co-terminal) swap-rate market model (ctSMM) where the underlying variables are overlapping swap-rates with discrete tenors. Consequently, drift computations and deriving bond ratios in the SMM are computationally more demanding than the LMM.

The greater simplicity of the LMM has resulted in its receiving greater attention in both practice and academia than SMMs. For a recent overview of research in the LMM, see Brigo and Mercurio (2006) or Brace (2008). One particular strand of research is to investigate the computational complexity of the model: how many operations are required to evolve the model across one step and how many to compute Greeks? Joshi (2003a) presented an algorithm that efficently computes the drifts in the LMM with complexity proportional to the number of rates (n) times the number of factors driving the LMM (F) per step. Since the other computational operations for each step are straight-forward, this resulted in $\mathcal{O}(nF)$ operations per step. Giles and Glasserman (2006) developed an adjoint method that computes Deltas and Vegas (sensitivities with respect to the underlying rates and volatility parameters, respectively) with order $\mathcal{O}(nF)$ per step. Capriotti and Giles (2010) extended the method to the calculation of correlation risk for portfolio default options.

Less attention has been given to such questions for SMMs. In fact, there are now many swap-rate market models since one is not restricted to using co-terminal rates. Galluccio et al. (2007) used graph theory to identify three main classes of generic market models, namely the co-terminal, co-initial and co-sliding models, and they introduced a novel calibration technique to allow simultaneous calibration to caplets and swaption prices. This followed on from Galluccio and Hunter (2004) where the co-initial SMM (ciSMM) was investigated and an algorithm to compute the drifts with order $\mathcal{O}(n^2)$ per step was derived. Pietersz and van Regenmortel (2005) studied constant maturity SMM (cmSMM) and other generic market models. They developed generic algorithms to evaluate drifts for various market models. The order to compute drifts

in the ctSMM is $\mathcal{O}(nF)$, and the order to compute drifts approximately in the cmSMM is $\mathcal{O}(nF)$. In addition, their method to derive bond ratios has order $\mathcal{O}(n^3)$.

Many exotic IRDs are written on swap-rates rather than LIBORs. For example, the underlyings of a Bermudan swaption are the co-terminal swap-rates and the underlyings of a CMS spread option are two co-initial swap-rates. Hence the need for different SMMs is to price and hedge specific IRDs in a customized and efficient way. Furthermore, Black's formula for European swaptions holds exactly in SMMs so that it is more consistent to price exotic products written on swap-rates in SMMs.

The fact that the rates overlap in the SMMs means that the analysis is considerably more complicated than for the LMM. However, progress has been made. Joshi and Liesch (2007) introduced efficient algorithms for implementing generic market models with order $\mathcal{O}(nF)$ per step for a wide class of market models. Since the computational order of all the operations for each step has been shown to be $\mathcal{O}(nF)$, it should be possible to apply the adjoint method as in Giles and Glasserman (2006) to estimate Deltas and Vegas with this order of computations per step. In the case of the log-normal ctSMM, Joshi and Yang (2011a) established that this was true for Deltas of European and Bermudan products.

In this paper, we develop efficient algorithms that compute both Deltas and Vegas in the three displaced-diffusion (DD) SMMs: DDctSMM, DDcmSMM and DDciSMM. The reason for choosing the DD versions is that these type of models are able to generate implied volatility skews, which is a prevalent observation in the IRD market. The essential idea of our method is that we decompose the one-step computation into a number of simpler vector operations in each simulation path. Each of these can be differentiated straightforwardly and then used for multiplication of the adjoint. A slight modification of the method of computing Deltas in generic market models will compute market Vegas with order $\mathcal{O}(nF)$ per step. The main advantage of our methodology is that provided the product has a Lipschitz-continuous pay-off function, we can compute its Deltas and Vegas in a fast and efficient manner. We do not study Gammas in this paper; however, it is clear that the results in Joshi and Yang (in press b) could be adapted to this case.

In section 2, we review the efficient implementation of the displaced-diffusion SMMs. In section 3, we briefly discuss the efficient adjoint method in Joshi and Yang (in press a). We then show how to extend the method to the DDcmSMM and the DDciSMM in sections 4 and 5. In section 6, we show how to compute Vegas in different swap-rate market models. Results of numerical tests, together with analysis of the results, are presented in section 7. We conclude in section 8.

2. Generic market models

2.1. Notations

The tenor structure is a finite set of dates

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

where $\{T_i\}_{i=0}^n$ are spaced by a set of real numbers $\tau_{i-1} = T_i - T_{i-1}$, for all i. We let $P_i(t)$ denote the time-t price of the zero-coupon bond maturing at time T_i. We let $\mathrm{SR}_{ij}(t)$ denote the time-t swap-rates associated to times $T_i, \ldots, T_j$, then it follows from simple no-arbitrage conditions (see Joshi, 2003a,b) that

$$\mathrm{SR}_{ij}(t) = \frac{P_i(t) - P_j(t)}{A_{ij}(t)}, \tag{1}$$

where $A_{ij}(t) = \sum_{k=i+1}^{j} \tau_{k-1} P_k(t)$ is the annuity of $\mathrm{SR}_{ij}(t)$. In what follows, if no confusion arises we will omit the dependence on time for notational purposes.

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, $A = \{a_{ik}\}$, of the covariance matrix, C, for each step to determine the evolution. We can therefore write across each step

$$\mathrm{d}\mathrm{SR}_{ij} = \mu_i^{(N)} \mathrm{d}t + \left(\mathrm{SR}_{ij} + \alpha_i\right) \sum_{k=1}^{F} a_{ik} \mathrm{d}Z_k \tag{2}$$

where $\mu_i^{(N)}$ is the drift of SR_{ij} under the measure associated with the bond P_N, $\{\alpha_i\}$ is a set of displaced-diffusion coefficients, and $\{Z_k\}$ is a vector of independent Brownian motions. We note that this formulation is general: if we take $F = n$, then any set

of correlated Brownian motions can be decomposed as a linear combination of such independent Brownian motions.

2.2.1. The cross variation derivative

Definition 2.1. *The cross variation derivative for two Itô processes*

$$\mathrm{d}X_t = \mu_X\big(X_t, Y_t, t\big)\mathrm{d}t + \sigma_X\big(X_t, Y_t, t\big)\mathrm{d}W_t^X,$$
$$\mathrm{d}Y_t = \mu_Y\big(X_t, Y_t, t\big)\mathrm{d}t + \sigma_Y\big(X_t, Y_t, t\big)\mathrm{d}W_t^Y$$

is defined to be the coefficient of $\mathrm{d}t$ *in* $\mathrm{d}X_t\mathrm{d}Y_t$*. If* $\mathrm{d}W_t^X\mathrm{d}W_t^Y = \rho\mathrm{d}t$ *then*

$$\big\langle X_t, Y_t\big\rangle = \rho\sigma_X\big(X_t, Y_t, t\big)\sigma_Y\big(X_t, Y_t, t\big). \qquad (3)$$

We give a summary of the main properties of the cross variation derivative: for Itô processes X_t, Y_t and Z_t,

1. (Linearity) $\big\langle X_t, Y_t+Z_t\big\rangle = \big\langle X_t, Y_t\big\rangle+\big\langle X_t, Z_t\big\rangle$;
2. (Product rule) $\big\langle X_t, Y_tZ_t\big\rangle = Z_t\big\langle X_t, Y_t\big\rangle + Y_t\big\langle X_t, Z_t\big\rangle$;
3. (Quotient rule) $\big\langle X_t, Y_t^{-1}\big\rangle = -Y_t^2\big\langle X_t, Y_t\big\rangle$.

For detailed discussion of the cross variation derivative, we refer the reader to Joshi and Liesch (2007).

2.2.2. General drift formulae

Proposition 2.1. *The general formula for the state-dependent drifts in the stochastic differential equation* (2) *is*

$$\mu_i^{(N)} = -\frac{\mathrm{SR}_{ij}+\alpha_i}{\overline{A}_{ij}}\sum_{k=1}^{F} a_{ik}\Big\langle Z_k, \overline{A}_{ij}\Big\rangle, \qquad (4)$$

where $\overline{A}_{ij} = A_{ij}/P_N$ *is the deflated annuity price.*

Proof. 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_N is a martingale. Since $\mathrm{SR}_{ij}A_{ij}$, A_{ij} are tradables, then $\mathrm{SR}_{ij}\overline{A}_{ij}$ and $\overline{A}_{ij}$ are martingales under the measure associated with bond P_N. Therefore the $\mathrm{d}t$ term in the following stochastic differential equation

$$\begin{aligned}&\mathrm{d}\Big(\mathrm{SR}_{ij}\overline{A}_{ij}\Big)\\ &\quad= \overline{A}_{ij}\mathrm{dSR}_{ij} + \mathrm{SR}_{ij}\mathrm{d}\overline{A}_{ij} + \Big\langle \mathrm{SR}_{ij}, \overline{A}_{ij}\Big\rangle\mathrm{d}t\\ &\quad= \Big(\mu_i^{(N)}\overline{A}_{ij} + \Big\langle \mathrm{SR}_{ij}, \overline{A}_{ij}\Big\rangle\Big)\mathrm{d}t\\ &\qquad+ \overline{A}_{ij}\mathrm{SR}_{ij}\sum_{k=1}^{F} a_{i,k}\mathrm{d}Z_k + \mathrm{SR}_{ij}\mathrm{d}\overline{A}_{ij}\end{aligned}$$

is equal to zero, then it follows that

$$\begin{aligned}\mu_i^{(N)} &= -\frac{1}{\overline{A}_{ij}}\Big\langle \mathrm{SR}_{ij}, \overline{A}_{ij}\Big\rangle\\ &= -\frac{\mathrm{SR}_{ij}+\alpha_i}{\overline{A}_{ij}}\sum_{k=1}^{F} a_{ik}\Big\langle Z_k, \overline{A}_{ij}\Big\rangle. \qquad (5)\end{aligned}$$

□

2.3. Derivatives pricing

An IRD will pay a stream of cashflows 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 market model and this function may also incorporate information from previous times.

For simplicity, we shall concentrate on the case where there is a single cash-flow which is a function, f, of the prevailing rates at the maturity of the IRD. However, if a product pays a stream of cash-flows, then these cash-flows will be aggregated along each path of the simulation. We make the (very mild) assumption that the cash-flow can be computed as a function of the yield curve at the time is determined with order n computations.

2.4. Co-terminal swap-rate market model

The DDctSMM is characterized by the set of swap-rates

$$\mathrm{SR}_{in} = \frac{\overline{P}_i - 1}{\overline{A}_{in}}, \quad i = 0, 1, \ldots, n-1, \qquad (6)$$

where $\overline{P}_i = P_i/P_n$, and $\overline{A}_{in} = A_{in}/P_n$.

2.4.1. Bond and annuity ratios

Joshi and Liesch (2007) developed the following algorithm to compute annuity ratios:

$$\overline{A}_{in} = \overline{A}_{i+1,n} + \tau_i\overline{P}_{i+1} \qquad (7)$$

where the bond ratios are given by

$$\overline{P}_i = 1 + \mathrm{SR}_{in}\overline{A}_{in}. \qquad (8)$$

2.4.2. Recursive formula for the cross variation derivatives

Proposition 2.2. *The cross variation derivatives for the co-terminal swap-rates in* (5) *are given by*

$$\Big\langle Z_k, \overline{A}_{in}\Big\rangle = \tau_i a_{i+1,k}\big(\mathrm{SR}_{i+1,n}+\alpha_{i+1}\big)\overline{A}_{i+1,n} + (1+\tau_i \mathrm{SR}_{i+1,n})\Big\langle Z_k, \overline{A}_{i+1,n}\Big\rangle \quad (9)$$

under the terminal bond measure.

Proof. From (7) and the linearity property of cross variation derivatives:

$$\Big\langle Z_k, \overline{A}_{in}\Big\rangle = \Big\langle Z_k, \overline{A}_{i+1,n}\Big\rangle + \tau_i\Big\langle Z_k, \overline{P}_{i+1}\Big\rangle. \quad (10)$$

From (8) and the product rule of cross variation derivatives, the second term on the right hand side of (10) is given by

$$\Big\langle Z_k, \overline{P}_{i+1}\Big\rangle = a_{i+1,k}\big(\mathrm{SR}_{i+1,n}+\alpha_{i+1}\big)\overline{A}_{i+1,n} + \Big\langle Z_k, \overline{A}_{i+1,n}\Big\rangle \mathrm{SR}_{i+1,n}. \quad (11)$$

Substituting (11) into (10) gives the result. □

2.5. Constant maturity swap-rate market model

The DDcmSMM is characterized by the set of swap-rates

$$\mathrm{SR}_{i,i+r} = \frac{\overline{P}_i - \overline{P}_{i+r}}{\overline{A}_{i,i+r}}, \quad i=0,1,\ldots,n-1, \quad (12)$$

where r is a fixed integer number, $\overline{P}_i = P_i/P_n$, and $\overline{A}_{i,i+r} = A_{i,i+r}/P_n$. For notational simplicity, we write $\mathrm{SR}_{i,i+r}$ as SR_i^r and $\overline{A}_{i,i+r}$ as $\overline{A}_i^r$. We make the convention that if $i+r \geq n$ then we set $i+r=n$, thus the last r rates will be the co-terminal swap-rates. If we set $r=1$ then the cmSMM becomes the LMM, and if we set $r=n$ then the cmSMM becomes the ctSMM.

2.5.1. Bond and annuity ratios

Joshi and Liesch (2007) developed the following algorithm to compute annuity ratios:

$$\overline{A}_i^r = \begin{cases} \overline{A}_{i+1}^r + \tau_i \overline{P}_{i+1} & \text{for } i \geq n-r, \\ \overline{A}_{i+1}^r + \tau_i \overline{P}_{i+1} - \tau_{i+r}\overline{P}_{i+r+1} & \text{for } i < n-r, \end{cases} \quad (13)$$

where the bond ratios are given by

$$\overline{P}_i = \begin{cases} 1+\mathrm{SR}_i^r \overline{A}_i^r & \text{for } i \geq n-r, \\ \overline{P}_{i+r} + \mathrm{SR}_i^r \overline{A}_i^r & \text{for } i < n-r. \end{cases} \quad (14)$$

2.5.2. Recursive formula for the cross variation derivatives

Proposition 2.3. *The cross variation derivatives for the constant-maturity swap-rates in* (5) *are given by*

$$\Big\langle Z_k, \overline{A}_i^r\Big\rangle = \tau_i a_{i+1,k}\big(\mathrm{SR}_{i+1}^r+\alpha_{i+1}\big)\overline{A}_{i+1}^r + (1+\tau_i \mathrm{SR}_{i+1}^r)\Big\langle Z_k, \overline{A}_{i+1}^r\Big\rangle + (\tau_i - \tau_{i+r})\Big\langle Z_k, \overline{P}_{i+r+1}\Big\rangle \quad (15)$$

with

$$\Big\langle Z_k, \overline{P}_i\Big\rangle = \Big\langle Z_k, \overline{P}_{i+r}\Big\rangle + a_{ik}\big(\mathrm{SR}_i^r+\alpha_i\big)\overline{A}_i^r + \mathrm{SR}_i^r\Big\langle Z_k, \overline{A}_i^r\Big\rangle \quad (16)$$

under the terminal bond measure.

Proof. We note that we can use (9) to compute the cross variation derivatives for the last r swap-rates. Now, from (13) and the linearity property of cross variation derivatives:

$$\Big\langle Z_k, \overline{A}_i^r\Big\rangle = \Big\langle Z_k, \overline{A}_{i+1}^r\Big\rangle + \tau_i\Big\langle Z_k, \overline{P}_{i+1}\Big\rangle - \tau_{i+r}\Big\langle Z_k, \overline{P}_{i+r+1}\Big\rangle. \quad (17)$$

From (14), the linearity property and the product rule for cross variation derivatives:

$$\Big\langle Z_k, \overline{P}_{i+1}\Big\rangle = \Big\langle Z_k, \overline{P}_{i+r+1}\Big\rangle + a_{i+1,k}\big(\mathrm{SR}_{i+1}^r+\alpha_{i+1}\big)\overline{A}_{i+1}^r + \Big\langle Z_k, \overline{A}_{i+1}^r\Big\rangle \mathrm{SR}_{i+1}^r. \quad (18)$$

Substituting (18) into (17) gives the result.

If $\tau_i = \tau$, for $i = 0, 1, \ldots, n-1$, then we can see that the final term in (15) will become zero so that the recursive formula can be simplified to

$$\Big\langle Z_k, \overline{A}_i^r\Big\rangle = \tau a_{i+1,k}\big(\mathrm{SR}_{i+1}^r+\alpha_{i+1}\big)\overline{A}_{i+1}^r + (1+\tau \mathrm{SR}_{i+1}^r)\Big\langle Z_k, \overline{A}_{i+1}^r\Big\rangle, \quad (19)$$

which is the same as the recursive formula (7) in the DDctSMM. □

2.6. Co-initial swap-rate market model

The DDciSMM is characterized by the set of swap-rates

$$\mathrm{SR}_{0i} = \frac{1-\overline{P}_i}{\overline{A}_{0i}}, \quad i = 1, 2, \ldots, n, \tag{20}$$

where $\overline{P}_i = P_i/P_0$ and $\overline{A}_{0i} = A_{0i}/P_0$.

2.6.1. Bond and annuity ratios

Joshi and Liesch (2007) developed the following algorithm to compute annuity ratios:

$$\overline{A}_{0i} = \overline{A}_{0,i-1} + \tau_{i-1}\overline{P}_i \tag{21}$$

where the bond ratios are given by

$$\overline{P}_i = \frac{1-\mathrm{SR}_{0i}\overline{A}_{0,i-1}}{1+\tau_{i-1}\mathrm{SR}_{0i}}. \tag{22}$$

2.6.2. Recursive formula for the cross variation derivatives

Proposition 2.4. *The cross variation derivatives for the co-initial swap-rates in* (5) *are given by*

$$\left\langle Z_k, \overline{A}_{0i}\right\rangle = \frac{\left\langle Z_k, \overline{A}_{0,i-1}\right\rangle - a_{ik}\tau_{i-1}\big(\mathrm{SR}_i+\alpha_i\big)\overline{A}_{0i}}{1+\tau_{i-1}\mathrm{SR}_{0i}} \tag{23}$$

under the spot measure.

Proof. From (21) and the linearity property of cross variation derivatives:

$$\left\langle Z_k, \overline{A}_{0i}\right\rangle = \left\langle Z_k, \overline{A}_{0,i-1}\right\rangle + \tau_{i-1}\left\langle Z_k, \overline{P}_i\right\rangle. \tag{24}$$

From (22) and the product rule of cross variation derivatives, the second term on the right hand side of (24) is given by

$$\left\langle Z_k, \overline{P}_i\right\rangle = -\Big[a_{ik}\big(\mathrm{SR}_i+\alpha_i\big)\overline{A}_{0i} + \mathrm{SR}_{0i}\left\langle Z_k, \overline{A}_{0i}\right\rangle\Big]. \tag{25}$$

Substituting (25) into (24) gives the result. □

3. Delta estimation in swap-rate market models

3.1. Set-up

Assume that the market model has n underlying rates with the tenor structure $\{T_j\}_{j=0}^n$, and each rate is driven by F factors. We denote the pseudo-root square root matrix over the period $[T_{j-1}, T_j]$ by $A(T_j) = \{a_{ik}(T_{j-1})\}$.

3.1.1. Numerical scheme

To evolve the swap-rates step by step (i.e., from T_{j-1} to T_j), we use the simple log-Euler scheme: for $i = j, j+1, \ldots, n-1$, we set

$$\mathrm{SR}_i\big(T_j\big) = \Big(\mathrm{SR}_i\big(T_{j-1}\big) + \alpha_i\Big)\exp\Bigg(\mu_i^{(N)}\big(\mathrm{SR}(T_{j-1})\big) - \frac{C_{ii}(T_{j-1})}{2} + \sum_{k=1}^{F} a_{ik}(T_{j-1})Z_k\Bigg) - \alpha_i, \tag{26}$$

where $C_{ii} = \sum_{k=1}^F a_{ik}^2$, $\{Z_k\}$ is a sequence of independent normal variates and the discretized drift

$$\mu_i^{(N)}\big(\mathrm{SR}(T_{j-1})\big) = -\frac{1}{\overline{A}_{ij}\big(T_{j-1}\big)}\sum_{k=1}^{F} a_{ik}\big(T_{j-1}\big)\Big\langle Z_k, \overline{A}_{ij}\big(T_{j-1}\big)\Big\rangle., \tag{27}$$

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

3.1.2. Deflated pay-off and deltas

Let $f(T_m)$ denote the pay-off of an IRD maturing at time T_m, $m \le n$. The discounted pay-off is given by

$$g = P_N(0)\frac{f(T_m)}{P_N(T_m)}. \tag{28}$$

It is trivial to cope with multiple cash-flows (i.e., path-dependent IRDs with early exercise features) in this setting, we simply rewrite (28) as

$$g = P_N(0)\sum_{m=0}^{M}\frac{f(T_m)}{P_N(T_m)}. \tag{29}$$

However, we shall concentrate on computing Greeks of IRDs with a single cash-flow in this section.

The vector of Deltas (gradient vector) of the IRD is given by

$$\underline{\Delta} = \frac{\partial \mathbb{E}(g)}{\partial \mathbf{SR}(0)} = \frac{\partial P_N(0)}{\partial \mathbf{SR}(0)} \mathbb{E}\Big(\overline{f}(T_m)\Big) + P_N(0) \frac{\partial}{\partial \mathbf{SR}(0)} \mathbb{E}\Big(\overline{f}(T_m)\Big), \tag{30}$$

where $\overline{f}(T_m) = f(T_m)/P_N(T_m)$. If f is assumed to be *Lipschitz-continuous*, then the differentiation operator and the expectation operator in (30) can be interchanged

$$\underline{\Delta} = \frac{\partial P_N(0)}{\partial \mathbf{SR}(0)} \mathbb{E}\Big(\overline{f}(T_m)\Big) + P_N(0) \mathbb{E}\left(\frac{\partial \overline{f}(T_m)}{\partial \mathbf{SR}(0)}\right), \tag{31}$$

The first gradient vector in (31) is easy to compute, and we will compute the second gradient vector using the adjoint method on a pathwise basis in a Monte Carlo simulation.

3.2. General pathwise adjoint method

We will consider the following mappings in the adjoint method:

$$\mathbf{SR}(0) \xrightarrow{\mathbf{F}_0} \mathbf{SR}(T_0) \xrightarrow{\mathbf{F}_1} \cdots \xrightarrow{\mathbf{F}_m} \mathbf{SR}(T_m) \xrightarrow{\mathbf{F}} \overline{f}(T_m), \tag{32}$$

where $\mathbf{F}_i$ and $\mathbf{F}$ are vector functions, then it follows from the chain rule that

$$\frac{\partial \overline{f}(T_m)}{\partial \mathbf{SR}(0)} = \mathbf{F}' \mathbf{F}'_m \mathbf{F}'_{m-1} \cdots \mathbf{F}'_0, \tag{33}$$

where $\mathbf{F}'_i$ are Jacobian matrices and $\mathbf{F}$ is a gradient vector. Now, if we define an adjoint relation as follows

$$\begin{cases} \mathbf{V}(T_m) = \mathbf{F}' \mathbf{F}'_m, \\ \mathbf{V}(T_{k-1}) = \mathbf{V}(T_k) \mathbf{F}'_{k-1}, \end{cases} \tag{34}$$

then the gradient vector (33) is equal to $\mathbf{V}(0)$. The elements of the adjoints $\mathbf{V}(T_{k-1})$ can be computed from $\mathbf{V}(T_k)$ via

$$\mathbf{V}_i(T_{k-1}) = \mathbf{V}_i(T_k) \frac{\mathrm{SR}_i(T_k)}{\mathrm{SR}_i(T_{k-1})} + \sum_{j \geq i} \mathbf{V}_j(T_k) \mathrm{SR}_j(T_k) \frac{\partial \mu_j^{(N)}\big(\mathbf{SR}(T_{k-1})\big)}{\partial \mathrm{SR}_i(T_{k-1})}.$$

If we proceed naïvely, the computational complexity of the adjoint method will be $\mathcal{O}(n^2)$ per step since we are multiplying a vector, $\mathbf{V}(T_k)$, by a Jacobian matrix, $\mathbf{F}'_k$. However, it is possible to reduce the complexity to $\mathcal{O}(nF)$ per step. Consequently, if n is large and F is small, the reduction in order will result in substantial time-savings.

3.3. Efficient adjoint method

Joshi and Yang (2011a) decomposed the vector functions in (32), $\mathbf{F}_k$, into a number of sub-mappings, $\mathbf{F}_{k,i}$, to an extent such that all vector operations consist of simple functions and the corresponding Jacobian matrices become sparse matrices. We then identify the non-zero elements of the Jacobian matrices $\mathbf{F}'_{k,i}$, and only carry out the multiplications of the corresponding non-zero entries in the operation

$$\mathbf{v} = \mathbf{w} \mathbf{F}'_{k,i}.$$

Thus the computational order of the adjoint relations in (34) can be reduced to $\mathcal{O}(nF)$ per step.

In this paper, we consider the following sub-mappings for the various market models:

$$\mathbf{SR}(T_{k-1}) \xrightarrow{\mathbf{F}_{k,0}} \begin{bmatrix} \mathbf{SR}(T_{k-1}) \\ \overline{\mathbf{A}}(T_{k-1}) \end{bmatrix} \xrightarrow{\mathbf{F}_{k,1}} \begin{bmatrix} \mathbf{SR}(T_{k-1}) \\ \underline{\mu}^{(N)}(T_{k-1}) \end{bmatrix} \xrightarrow{\mathbf{F}_{k,2}} \mathbf{SR}(T_k), \tag{35}$$

where the sub-mappings satisfy

$$\mathbf{F}_k = \mathbf{F}_{k,0} \circ \mathbf{F}_{k,1} \circ \mathbf{F}_{k,2}, \quad k = 0, 1, \ldots, m.$$

Therefore the adjoint relation in (34) is equivalent to

$$\mathbf{V}(T_{k-1}) = \mathbf{V}(T_k) \mathbf{F}'_{k,2} \mathbf{F}'_{k,1} \mathbf{F}'_{k,0}, \quad k = 0, 1, \ldots, m. \tag{36}$$

We illustrate how to implement the efficient adjoint method in the DDcmSMM and the DDciSMM in the following two sections.

4. Delta estimation in the constant maturity swap-rate market model

Proposition 4.1. *In the DDcmSMM, the adjoint operation* (36) *can be implemented with a computational order of* $\mathcal{O}(nF)$ *per step.*

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

$$\mathbf{V}(0) = \mathbf{V}(T_0)\mathbf{F}'_{0,2}\mathbf{F}'_{0,1}\mathbf{F}'_{0,0} = \mathbf{V}(T_0)\mathbf{F}'_0. \quad (37)$$

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. We show the detailed proofs in the following sub-sections. □

4.1. Computational order of $\mathbf{w}\mathbf{F}'_{0,0}$

We divide the mapping

$$\mathbf{SR}(0) \xrightarrow{\mathbf{F}_{0,0}} \begin{bmatrix} \mathbf{SR}(0) \\ \overline{\mathbf{A}}(0) \end{bmatrix}$$

into the following sub-mappings

$$\mathbf{SR}(0) \xrightarrow{\mathbf{G}_{0,0}} \begin{bmatrix} \mathbf{SR}(0) \\ \overline{A}^r_{n-1}(0) \\ \{\overline{\mathbf{P}}(0)\}_{i=n}^{n} \end{bmatrix} \xrightarrow{\mathbf{G}_{0,1}} \begin{bmatrix} \mathbf{SR}(0) \\ \overline{A}^r_{n-1}(0) \\ \{\overline{\mathbf{P}}(0)\}_{i=n-1}^{n} \end{bmatrix}$$
$$\xrightarrow{\mathbf{G}_{0,2}} \begin{bmatrix} \mathbf{SR}(0) \\ \overline{A}^r_{n-2}(0) \\ \{\overline{\mathbf{P}}(0)\}_{i=n-1}^{n} \end{bmatrix} \xrightarrow{\mathbf{G}_{0,3}} \begin{bmatrix} \mathbf{SR}(0) \\ \overline{A}^r_{n-2}(0) \\ \{\overline{\mathbf{P}}(0)\}_{i=n-2}^{n} \end{bmatrix}$$
$$\xrightarrow{\mathbf{G}_{0,4}} \begin{bmatrix} \mathbf{SR}(0) \\ \overline{A}^r_{n-3}(0) \\ \{\overline{\mathbf{P}}(0)\}_{i=n-2}^{n} \end{bmatrix} \cdots \xrightarrow{\mathbf{G}_{0,2n-2}} \begin{bmatrix} \mathbf{SR}(0) \\ \overline{A}^r_{0}(0) \\ \{\overline{\mathbf{P}}(0)\}_{i=1}^{n} \end{bmatrix}$$
$$\xrightarrow{\mathbf{G}_{0,2n-1}} \begin{bmatrix} \mathbf{SR}(0) \\ \overline{\mathbf{P}}(0) \end{bmatrix} \xrightarrow{\mathbf{G}_0} \begin{bmatrix} \mathbf{SR}(0) \\ \overline{\mathbf{A}}(0) \end{bmatrix} \quad (38)$$

The even-numbered mappings $\mathbf{G}_{0,j}$ update the annuity ratios using the given set of inputs, and the odd-numbered mappings $\mathbf{G}_{0,j}$ computes a new bond ratio using the given set of inputs.

4.1.1. Jacobian matrix of $\mathbf{G}_{0,j}$ where j is even

From (13), we can see that $\overline{A}^r_i$, when $i \geq n-r$, only depends on $\overline{A}^r_{i+1}$ and $\overline{P}_{i+1}$, then

$$\left\{ \frac{\partial \overline{A}^r_i}{\partial \overline{A}^r_{i+1}} = 1, \frac{\partial \overline{A}^r_i}{\partial \overline{P}_{i+1}} = \tau_i. \right. \quad (39)$$

The Jacobian matrix $\mathbf{G}'_{0,j}$ has 1s on the diagonal, one entry equal to τ on the last column, and 0s elsewhere. Therefore the operation $\mathbf{v} = \mathbf{w}\mathbf{G}'_{0,j}$ has one computation.

From (13), we can see that $\overline{A}^r_i$, when $i < n-r$, depends on $\overline{A}^r_{i+1}$, $\overline{P}_{i+1}$ and $\overline{P}_{i+r+1}$, then

$$\begin{cases} \frac{\partial \overline{A}^r_i}{\partial \overline{A}^r_{i+1}} = 1, \\ \frac{\partial \overline{A}^r_i}{\partial \overline{P}_{i+1}} = \tau_i, \\ \frac{\partial \overline{A}^r_i}{\partial \overline{P}_{i+r+1}} = -\tau_{i+r}. \end{cases} \quad (40)$$

The Jacobian matrix $\mathbf{G}'_{0,j}$ has 1s on the diagonal, one entry equal to τ on the last column, one entry equal to $-\tau$ and 0s elsewhere. Therefore the operation $\mathbf{v} = \mathbf{w}\mathbf{G}'_{0,j}$ has two computations.

4.1.2. Jacobian matrix of $\mathbf{G}_{0,j}$ where j is odd

From (14), we can see that $\overline{P}_i$ when $i \geq n-r$ only depends on $\overline{A}^r_i$ and SR^r_i, then

$$\begin{cases} \frac{\partial \overline{P}_i}{\partial \overline{A}^r_i} = \mathrm{SR}^r_i, \\ \frac{\partial \overline{P}_i}{\partial \mathrm{SR}^r_i} = \overline{A}^r_i. \end{cases} \quad (41)$$

The Jacobian matrix $\mathbf{G}'_{0,j}$ has 1s on the diagonal, two entries equal to SR_i and α_i on the last row. Therefore the operation $\mathbf{v} = \mathbf{w}\mathbf{G}'_{0,j}$ has two computations.

From (14), we can see that $\overline{P}_i$ when $i < n-r$ only depends on $\overline{A}^r_i$, $\overline{P}_{i+r}$ and SR^r_i, then

$$\begin{cases} \frac{\partial \overline{P}_i}{\partial \overline{A}^r_i} = \mathrm{SR}^r_i, \\ \frac{\partial \overline{P}_i}{\partial \overline{P}_{i+r}} = 1, \\ \frac{\partial \overline{P}_i}{\partial \mathrm{SR}^r_i} = \overline{A}^r_i. \end{cases} \quad (42)$$

The Jacobian matrix $\mathbf{G}'_{0,j}$ has 1s on the diagonal, three entries equal to SR^r_i, 1 and $\overline{A}^r_i$ on the last row. Therefore the operation $\mathbf{v} = \mathbf{w}\mathbf{G}'_{0,j}$ has three computations.

4.1.3. Jacobian matrix of $\mathbf{G}_0$

For $i \geq n-r$,

$$\frac{\partial \overline{A}^r_i}{\partial \overline{P}_j} = \tau_{j-1}, \quad j > i.$$

However, for $i < n-r$,

$$\frac{\partial \overline{A}^r_i}{\partial \overline{P}_j} = \tau_{j-1}, \quad i+r \geq j > i.$$

Fortunately, the order of the operation $\mathbf{v} = \mathbf{w}\mathbf{G}'_0$ can be reduced to $\mathcal{O}(n)$. We define a sum variable, and carry out the following algorithm:

- Set $\mathbf{v}_j = \mathbf{w}_j$ for $j = 0, \ldots, n-1$.
- Set sum equal to $\tau_0 \mathbf{w}_n$ in loop 0.

- Set $sum = sum + \tau_j \mathbf{w}_{n+j}$ in loop j for $j = 1, \ldots, r-1$.
- Set $sum = sum + \tau_j(\mathbf{w}_{n+j} - \mathbf{w}_{n+j-r})$ in loop j for $j = r, \ldots, n-1$.

If we add the updated sum to $\mathbf{v}_{2n-j}$ in the jth loop, then we have executed the operation $\mathbf{v} = \mathbf{w}\mathbf{G}'_0$. Hence, the order of the operation is $\mathcal{O}(n)$.

4.1.4. Total computational order

Since the number of sub-mappings $\mathbf{G}_{0,j}$ depends on n, the computational order of the operation

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

is $\mathcal{O}(n)$ since each operation has a constant computational order. We ignore the Jacobian matrix $\mathbf{G}'_{0,0}$ as $\overline{A}^r_{n-1}$ is a constant so that $\mathbf{G}'_{0,0}$ is equivalent to an identity matrix. Hence the operation $\mathbf{v} = \mathbf{w}\mathbf{F}'_{0,0}$ has order $\mathcal{O}(n)$.

4.2. Computational order of $\mathbf{w}\mathbf{F}'_{0,1}$

We divide the mapping

$$\begin{bmatrix}\mathbf{SR}(0)\\ \overline{\mathbf{A}}(0)\end{bmatrix} \xrightarrow{\mathbf{F}_{0,1}} \begin{bmatrix}\mathbf{SR}(0)\\ \underline{\mu}^{(n)}(0)\end{bmatrix}$$

into the following sub-mappings

$$\begin{bmatrix}\mathbf{SR}(0)\\ \overline{\mathbf{A}}(0)\end{bmatrix} \xrightarrow{\mathbf{H}_{0,0}} \begin{bmatrix}\mathbf{SR}(0)\\ \overline{\mathbf{A}}(0)\\ \left\{\langle Z_k, \overline{P}^r_i\rangle\right\}_{i=n-1}^{n-1}\end{bmatrix}$$

$$\xrightarrow{\mathbf{H}_{0,1}} \begin{bmatrix}\mathbf{SR}(0)\\ \overline{\mathbf{A}}(0)\\ \left\{\langle Z_k, \overline{P}^r_i\rangle\right\}_{i=n-1}^{n-1}\\ \left\{\langle Z_k, \overline{A}^r_i\rangle\right\}_{i=n-2}^{n-2}\end{bmatrix}$$

$$\xrightarrow{\mathbf{H}_{0,2}} \begin{bmatrix}\mathbf{SR}(0)\\ \overline{\mathbf{A}}(0)\\ \left\{\langle Z_k, \overline{P}^r_i\rangle\right\}_{i=n-2}^{n-1}\\ \left\{\langle Z_k, \overline{A}^r_i\rangle\right\}_{i=n-2}^{n-2}\end{bmatrix}$$

$$\xrightarrow{\mathbf{H}_{0,3}} \begin{bmatrix}\mathbf{SR}(0)\\ \overline{\mathbf{A}}(0)\\ \left\{\langle Z_k, \overline{P}^r_i\rangle\right\}_{i=n-2}^{n-1}\\ \left\{\langle Z_k, \overline{A}^r_i\rangle\right\}_{i=n-2}^{n-3}\end{bmatrix}$$

$$\cdots \xrightarrow{\mathbf{H}_{0,2(n-2)+1}} \begin{bmatrix}\mathbf{SR}(0)\\ \overline{\mathbf{A}}(0)\\ \left\{\langle Z_k, \overline{P}^r_i\rangle\right\}_{i=1}^{n-1}\\ \left\{\langle Z_k, \overline{A}^r_i\rangle\right\}_{i=0}^{n-2}\end{bmatrix} \xrightarrow{\mathbf{H}_0} \begin{bmatrix}\mathbf{SR}(0)\\ \underline{\mu}^{(n)}(0)\end{bmatrix} \tag{43}$$

The even-numbered mappings $\mathbf{H}_{0,j}$ update the cross variation derivatives $\langle Z_k, \overline{P}^r_i\rangle$, and the odd-numbered mappings $\mathbf{H}_{0,j}$ update the cross variation derivatives $\langle Z_k, \overline{A}^r_i\rangle$.

4.2.1. Jacobian matrix of $\mathbf{H}_{0,j}$ where j is even

From (18), we have the following non-zero partial derivatives: for $j \geq n-r$

$$\begin{cases}\frac{\partial}{\partial \mathrm{SR}^r_j}\langle Z_k, \overline{P}_j\rangle = a_{jk}\overline{A}^r_j + \langle Z_k, \overline{A}^r_j\rangle,\\ \frac{\partial}{\partial \overline{A}^r_j}\langle Z_k, \overline{P}_j\rangle = a_{jk}(\mathrm{SR}^r_j + \alpha_j),\\ \frac{\partial}{\partial \langle Z_k, \overline{A}^r_j\rangle}\langle Z_k, \overline{P}_j\rangle = \mathrm{SR}^r_j.\end{cases} \tag{44}$$

For $j \leq n-r$, we have an extra partial derivative in addition to the three in (44):

$$\frac{\partial\langle Z_k, \overline{P}_j\rangle}{\partial\langle Z_k, \overline{P}_{j+r}\rangle} = 1. \tag{45}$$

4.2.2. Jacobian matrix of $\mathbf{H}_{0,j}$ where j is odd

From (15), we have the following non-zero partial derivatives: for $j \geq n-r$

$$\begin{cases}\frac{\partial}{\partial \mathrm{SR}^r_{j+1}}\langle Z_k, \overline{A}^r_j\rangle = \tau_j\left[a_{j+1,k}\overline{A}^r_{j+1} + \langle Z_k, \overline{A}^r_{j+1}\rangle\right],\\ \frac{\partial}{\partial \overline{A}^r_{j+1}}\langle Z_k, \overline{A}^r_j\rangle = \tau_j a_{j+1,k}\mathrm{SR}^r_{j+1},\\ \frac{\partial}{\partial \langle Z_k, \overline{A}^r_{j+1}\rangle}\langle Z_k, \overline{A}^r_j\rangle = 1 + \tau_j \mathrm{SR}^r_{j+1}.\end{cases} \tag{46}$$

For $j \leq n-r$, we have an extra partial derivative in addition to the three in (46):

$$\frac{\partial\langle Z_k, \overline{A}^r_j\rangle}{\partial\langle Z_k, \overline{P}_{j+r+1}\rangle} = \tau_j - \tau_{j+r}. \tag{47}$$

Therefore $\mathbf{H}'_{0,j}$ has 1s on the diagonals, and either 3 or 4 entries on each of the last F rows. Therefore each of the operations

$$\mathbf{v} = \mathbf{w}\mathbf{H}'_{0,j}, \quad j = 0, 1, \ldots, 2(n-2), 2(n-2)+1,$$

has computational order of $\mathcal{O}(F)$.

4.2.3. Jacobian matrix of $\mathbf{H}_0$

The drifts of the constant maturity swap-rates under the Euler scheme are given by

$$\mu_j^{(n)} = -\frac{1}{\overline{A}_j^r} \sum_{k=1}^{F} a_{jk} \left\langle Z_k, \overline{A}_j^r \right\rangle. \tag{48}$$

We can see that each $\mu_j^{(n)}$ depends on $\overline{A}_j^r$ and $\left\langle Z_k, \overline{A}_j^r \right\rangle$, then

$$\begin{cases} \frac{\partial}{\partial A_j^r} \mu_j^{(n)} = -\frac{1}{\overline{A}_j^r} \mu_j^{(n)}, \\ \frac{\partial}{\partial \langle Z_k, \overline{A}_j^r \rangle} \mu_j^{(n)} = -\frac{1}{\overline{A}_j^r} a_{jk}, \end{cases} \tag{49}$$

for $j = 0, \ldots, n-2$ and $k = 1, \ldots, F$. The Jacobian matrix $\mathbf{H}_0'$ has $n-1$ partial derivatives equal to $\partial \mu_j^{(n)} / \partial A_j^r$, and $(n-1)F$ partial derivatives equal to $\partial \mu_j^{(n)} / \partial \langle Z_k, \overline{A}_j^r \rangle$. Therefore the operation $\mathbf{v} = \mathbf{w} H_0'$ has order $\mathcal{O}(nF)$.

4.2.4. Total computational order

The number of mappings $\mathbf{H}_{0,j}$ whose order is constant, $j > 0$, depends on n. Therefore the order of the operation

$$\mathbf{v} = \mathbf{w}\mathbf{H}_0' \mathbf{H}_{0,2(n-2)+1}' \cdots \mathbf{H}_{0,0}' = \mathbf{w}\mathbf{F}_{0,1}'$$

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

4.3. Computational order of $\mathbf{w}\mathbf{F}_{0,2}'$

We use the following Euler scheme to simulate constant-maturity swap-rates

$$\mathrm{SR}_j^r(T_0) = \left(\mathrm{SR}_j^r(0) + \alpha_j\right) \exp\left[\mu_j^{(n)} - \frac{1}{2}\sum_{k=1}^{F} a_{jk}^2 + \sum_{k=1}^{F} a_{jk} Z_k\right] - \alpha_j, \tag{50}$$

where $\mu_j^{(n)}$ is given in (48). It follows that the non-zero partial derivatives of $\mathbf{F}_{0,2}'$ are

$$\begin{cases} \frac{\partial \mathrm{SR}_j^r(T_0)}{\partial \mathrm{SR}_j^r(0)} = \frac{\mathrm{SR}_j^r(T_0) + \alpha_j}{\mathrm{SR}_j^r(0) + \alpha_j}, \quad j = 0, \ldots, n-1. \\ \frac{\partial \mathrm{SR}_j^r(T_0)}{\partial \mu_j^{(n)}} = \mathrm{SR}_j^r(T_0) + \alpha_j, \quad j = 0, \ldots, n-2. \end{cases} \tag{51}$$

Thus, $\mathbf{F}_{0,3}'$ has the general form

$$\begin{bmatrix} \times\times & & & \times & & \\ & \ddots & & & \ddots & \\ & & \times\times & & & \times \\ & & & \times\times & & 0 \end{bmatrix},$$

where $\times\times = (\mathrm{SR}_j^r(T_0) + \alpha_j)/(\mathrm{SR}_j^r(0) + \alpha_j)$, $\times = \mathrm{SR}_j^r(T_0) + \alpha_j$ and the blanks are zero. Note that the 0 indicates that the swap-rate SR_{n-1}^r is driftless under the terminal measure. The Jacobian matrix $\mathbf{F}_{0,3}'$ has $2n-1$ non-zero entries. Hence the operation $\mathbf{v} = \mathbf{w}\mathbf{F}_{0,3}'$ has $2n-1$ computations so that its order is $\mathcal{O}(n)$.

4.4. Computational order of $\mathbf{w}\mathbf{F}'$

Similar to the case of ctSMM in Joshi and Yang (in press a): the gradient $\mathbf{F}'$ can be computed in order $\mathcal{O}(n)$. We divide the mapping $\mathbf{F}$ in (32) into the following sub-mappings:

$$\mathbf{SR}(T_m) \xrightarrow{\mathbf{I}_0} \begin{bmatrix} \mathbf{SR}(T_m) \\ \overline{\mathbf{A}}(T_m) \end{bmatrix} \xrightarrow{\mathbf{I}_1} \overline{f}(T_m). \tag{52}$$

We have shown in section 4.1 that the operations $\mathbf{v} = \mathbf{w}\mathbf{I}_0'$ has order $\mathcal{O}(n)$. The computation of the gradient $\mathbf{I}_1'$ is $\mathcal{O}(n)$ (by assumption), so the order of computing $\mathbf{F}'$ is $\mathcal{O}(n)$.

4.5. Summary

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

$$\mathbf{V}(0) = \mathbf{V}(T_0)\mathbf{F}_{0,2}'\mathbf{F}_{0,1}'\mathbf{F}_{0,0}' = \mathbf{V}(T_0)\mathbf{F}_0'$$

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

$$\mathbf{V}(T_{k-1}) = \mathbf{V}(T_k)\mathbf{F}_{k,2}'\mathbf{F}_{k,1}'\mathbf{F}_{k,0}' = \mathbf{V}(T_k)\mathbf{F}_k'$$

is $\mathcal{O}(nF)$ for each step.

4.6. Delta computation in co-terminal swap-rate and LIBOR market models

Since the ctSMM ($r = n$) and the LMM ($r = 1$) are special cases of the cmSMM. Delta computations in these two market models have computational order equal to $\mathcal{O}(nF)$ per step.

5. Delta estimation in the co-initial swap-rate market model

Proposition 5.1. *In the DDciSMM, the adjoint operation* (36) *can be implemented with a computational order of* $\mathcal{O}(nF)$ *per step.*

Proof. In the one-period DDciSMM, we will consider the following mappings

$$\begin{aligned}\mathbf{SR}(0) &\xrightarrow{\mathbf{F}_{0,0}} \begin{bmatrix}\mathbf{SR}(0)\\ \overline{\mathbf{A}}(0)\end{bmatrix} \xrightarrow{\mathbf{F}_{0,1}} \begin{bmatrix}\mathbf{SR}(0)\\ \underline{\mu}^{(0)}(0)\end{bmatrix} \xrightarrow{\mathbf{F}_{0,2}} \mathbf{SR}(T_0)\\ &\xrightarrow{\mathbf{F}_{1,0}} \begin{bmatrix}\mathbf{SR}(T_0)\\ \overline{\mathbf{A}}(T_0)\end{bmatrix} \xrightarrow{\mathbf{F}} \overline{f}(T_0).\end{aligned} \tag{53}$$

We will prove in the following sub-sections that the computational order of the following operation

$$\mathbf{V}(0) = \mathbf{F}'\mathbf{F}'_{1,0}\mathbf{F}'_{0,2}\mathbf{F}'_{0,1}\mathbf{F}'_{0,0} \tag{54}$$

is $\mathcal{O}(nF)$. □

5.1. Computational order of $\mathbf{wF}'_{0,0}$

We divide the mapping

$$\mathbf{SR}(0) \xrightarrow{\mathbf{F}_{0,0}} \begin{bmatrix}\mathbf{SR}(0)\\ \overline{\mathbf{A}}(0)\end{bmatrix}$$

into the following sub-mappings

$$\begin{aligned}\mathbf{SR}(0) &\xrightarrow{\mathbf{G}_{0,0}} \begin{bmatrix}\mathbf{SR}(0)\\ \overline{P}_1(0)\end{bmatrix} \xrightarrow{\mathbf{G}_{0,1}} \begin{bmatrix}\mathbf{SR}(0)\\ \overline{P}_1(0)\\ \{\overline{\mathbf{A}}_{0i}(0)\}_{i=1}^{1}\end{bmatrix}\\ &\xrightarrow{\mathbf{G}_{0,2}} \begin{bmatrix}\mathbf{SR}(0)\\ \overline{P}_2(0)\\ \{\overline{\mathbf{A}}_{0i}(0)\}_{i=1}^{1}\end{bmatrix} \xrightarrow{\mathbf{G}_{0,3}} \begin{bmatrix}\mathbf{SR}(0)\\ \overline{P}_2(0)\\ \{\overline{\mathbf{A}}_{0i}(0)\}_{i=1}^{2}\end{bmatrix}\\ &\cdots \xrightarrow{\mathbf{G}_{0,2n-2}} \begin{bmatrix}\mathbf{SR}(0)\\ \overline{P}_n(0)\\ \{\overline{\mathbf{A}}_{0i}(0)\}_{i=1}^{n-1}\end{bmatrix} \xrightarrow{\mathbf{G}_{0,2n-1}} \begin{bmatrix}\mathbf{SR}(0)\\ \overline{\mathbf{A}}(0)\end{bmatrix}\end{aligned} \tag{55}$$

The even-numbered mappings $\mathbf{G}_{0,j}$ update a bond ratio using the given set of inputs, and the odd-numbered mappings $\mathbf{G}_{0,j}$ computes a new annuity ratio using the given set of inputs.

5.1.1. Jacobian matrix of $\mathbf{G}_{0,j}$ where j is even

From (22), we can see that $\overline{P}_i$ depends on $\overline{A}_{i-1}$ and $\mathrm{SR}_{0,i}$, then

$$\begin{cases}\frac{\partial \overline{P}_i}{\partial \overline{A}_{0,i-1}} = -\frac{\overline{A}_{0,i-1}+\tau_{i-1}}{(1+\tau_{i-1}\mathrm{SR}_{0i})^2},\\ \frac{\partial \overline{P}_i}{\partial \mathrm{SR}_{0i}} = \frac{\mathrm{SR}_{0i}}{1+\tau_{i-1}\mathrm{SR}_{0i}}.\end{cases} \tag{56}$$

The existence of two non-zero partial derivatives implies that the operation $\mathbf{v} = \mathbf{wG}'_{0,j}$ has two computations.

5.1.2. Jacobian matrix of $\mathbf{G}_{0,j}$ where j is odd

From (21), we can see that $\overline{A}_{0i}$ only depends on $\overline{A}_{0,i-1}$ and $\overline{P}_i$, then

$$\begin{cases}\frac{\partial \overline{A}_{0i}}{\partial \overline{A}_{0,i-1}} = 1,\\ \frac{\partial \overline{A}_{0i}}{\partial \overline{P}_i} = \tau_{i-1}.\end{cases} \tag{57}$$

Then the Jacobian matrix $\mathbf{G}'_{0,j}$ has 1s on the diagonal, one entry equal to τ on the last row. Therefore the operation $\mathbf{v} = \mathbf{wG}'_{0,j}$ has one computation.

5.1.3. Total computational order

Similar to the case of DDcmSMM: The number of sub-mappings $\mathbf{G}_{0,j}$ depends on n, thus the computational order of the operation

$$\mathbf{v} = \mathbf{wG}'_{0,2n-1}\mathbf{G}'_{0,2n-2}\cdots\mathbf{G}'_{0,0} = \mathbf{wF}'_{0,0}$$

is $\mathcal{O}(n)$ since each operation has a constant computational order.

5.2. Computational order of $\mathbf{wF}'_{0,1}$

We divide the mapping

$$\begin{bmatrix}\mathbf{SR}(0)\\ \overline{\mathbf{A}}(0)\end{bmatrix} \xrightarrow{\mathbf{F}_{0,1}} \begin{bmatrix}\mathbf{SR}(0)\\ \underline{\mu}^{(0)}(0)\end{bmatrix}$$

into the following sub-mappings

$$\begin{aligned}\begin{bmatrix}\mathbf{SR}(0)\\ \overline{\mathbf{A}}(0)\end{bmatrix} &\xrightarrow{\mathbf{H}_{0,0}} \begin{bmatrix}\mathbf{SR}(0)\\ \overline{\mathbf{A}}(0)\\ \{\langle Z_k, \overline{A}_i\rangle\}_{i=1}^{1}\end{bmatrix}\\ &\xrightarrow{\mathbf{H}_{0,1}} \begin{bmatrix}\mathbf{SR}(0)\\ \overline{\mathbf{A}}(0)\\ \{\langle Z_k, \overline{A}_i\rangle\}_{i=1}^{2}\end{bmatrix}\\ &\cdots \xrightarrow{\mathbf{H}_{0,n-1}} \begin{bmatrix}\mathbf{SR}(0)\\ \overline{\mathbf{A}}(0)\\ \{\langle Z_k, \overline{A}_i\rangle\}_{i=1}^{n}\end{bmatrix} \xrightarrow{\mathbf{H}_0} \begin{bmatrix}\mathbf{SR}(0)\\ \underline{\mu}^{(0)}(0)\end{bmatrix}\end{aligned} \tag{58}$$

In contrast to the DDcmSMM where we only need to compute $(n-1)F$ cross variation derivatives, we need to compute nF cross variation derivatives in the DDciSMM since all rates have a drift term under the spot measure.

5.2.1. Jacobian matrix of $\mathbf{H}_{0,j}$

From (23), we can see that $\langle Z_k, \overline{A}_{0j}\rangle$ depend on SR_{0j}, $\overline{A}_{0j}$ and $\langle Z_k, \overline{A}_{0,j-1}\rangle$, then the non-zero partial derivatives are:

$$\begin{cases} \frac{\partial}{\partial \mathrm{SR}_{0j}}\langle Z_k, \overline{A}_{0j}\rangle = -\frac{\tau_{j-1}[\langle Z_k, \overline{A}_{0j}\rangle + a_{jk}\overline{A}_{0j}]}{1+\tau_{j-1}\mathrm{SR}_{0j}}, \\ \frac{\partial}{\partial \overline{A}_{0j}}\langle Z_k, \overline{A}_{0j}\rangle = -\frac{\tau_{j-1}(\mathrm{SR}_{0j}+\alpha_j)a_{jk}}{1+\tau_{j-1}\mathrm{SR}_{0j}}, \\ \frac{\partial}{\partial \langle Z_k, \overline{A}_{0,j-1}\rangle}\langle Z_k, \overline{A}_{0j}\rangle = \frac{1}{1+\tau_{j-1}\mathrm{SR}_{0j}}, \end{cases} \tag{59}$$

for $k = 1, \ldots, F$. Therefore $\mathbf{H}'_{0,j}$ has 1s on the diagonals, and 3 entries equal to the above partial derivatives on each of the last F rows. Therefore each of the operations

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

has $3F$ computations.

5.2.2. Jacobian matrix of $\mathbf{H}_0$

The drifts of SR_{0j} under the Euler are given by

$$\mu_j^{(0)} = -\frac{1}{\overline{A}_{0j}}\sum_{k=1}^{F} a_{jk}\langle Z_k, \overline{A}_{0j}\rangle. \tag{60}$$

We can see that each $\mu_j^{(0)}$ depends on $\overline{A}_{0j}$ and $\langle Z_k, \overline{A}_{0j}\rangle$, then

$$\begin{cases} \frac{\partial}{\partial \overline{A}_{0j}}\mu_j^{(0)} = -\frac{\mu_j^{(0)}}{\overline{A}_{0j}}, \\ \frac{\partial}{\partial \langle Z_k, \overline{A}_{0j}\rangle}\mu_j^{(0)} = -\frac{a_{jk}}{\overline{A}_{0j}}, \end{cases} \tag{61}$$

for $j = 1, \ldots, n$ and $k = 1, \ldots, F$. $\mathbf{H}'_0$ has n partial derivatives equal to $\partial\mu_j^{(0)}/\partial A_{0j}$, and nF partial derivatives equal to $\partial\mu_j^{(0)}/\partial\langle Z_k, \overline{A}_{0j}\rangle$. Therefore the operation $\mathbf{v} = \mathbf{w}H'_0$ has order $\mathcal{O}(nF)$.

5.2.3. Total computational order

The number of mappings $\mathbf{H}_{0,j}$ whose order is constant depends on n, therefore the order of

$$\mathbf{v} = \mathbf{w}\mathbf{H}'_0\mathbf{H}'_{0,n-1}\cdots\mathbf{H}'_{0,0} = \mathbf{w}\mathbf{F}'_{0,2}$$

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

5.3. Computational order of $\mathbf{w}\mathbf{F}'_{0,2}$

We have the following Euler scheme to simulate co-initial swap-rates

$$\mathrm{SR}_{0j}(T_0) = \big(\mathrm{SR}_{0j}(0) + \alpha_j\big)\exp\Bigg[\mu_j^{(0)} - \frac{1}{2}\sum_{k=1}^{F} a_{jk}^2 + \sum_{k=1}^{F} a_{j,k}Z_k\Bigg] - \alpha_j, \tag{62}$$

where $\mu_j^{(0)}$ is given in (60). It follows that the non-zero partial derivatives of $\mathbf{F}'_{0,2}$ are

$$\begin{cases} \frac{\partial \mathrm{SR}_{0j}(T_0)}{\partial \mathrm{SR}_{0j}(0)} = \frac{\mathrm{SR}_{0j}(T_0)+\alpha_j}{\mathrm{SR}_{0j}(0)+\alpha_j}, \\ \frac{\partial \mathrm{SR}_{0j}(T_0)}{\partial \mu_j^{(0)}} = \mathrm{SR}_{0j}(T_0) + \alpha_j, \end{cases} \tag{63}$$

for $j = 1, \ldots, n$. Thus, the Jacobian matrix $\mathbf{F}'_{0,3}$ has the general form

$$\begin{bmatrix} \times\times & & \times & & \\ & \ddots & & \ddots & \\ & & \times\times & & \times \end{bmatrix},$$

where $\times\times = \partial\mathrm{SR}_{0j}(T_0)/\partial\mathrm{SR}_{0j}(0)$, $\times = \partial\mathrm{SR}_{0j}(T_0)/\partial\mu_j^{(0)}$ and the blanks are zero. The Jacobian matrix $\mathbf{F}'_{0,3}$ has $2n$ non-zero entries. Hence the operation $\mathbf{v} = \mathbf{w}\mathbf{F}'_{0,3}$ has $2n$ computations so that its order is $\mathcal{O}(n)$.

5.4. Computational order of $\mathbf{w}\mathbf{F}'$

We divide the mapping F in (53) into the following sub-mappings:

$$\mathbf{SR}(T_0) \xrightarrow{\mathbf{I}_0} \begin{bmatrix} \mathbf{SR}(T_0) \\ \overline{\mathbf{A}}(T_0) \end{bmatrix} \xrightarrow{\mathbf{I}_1} \overline{f}(T_0). \tag{64}$$

We have shown in sections 5.1 that the operation $\mathbf{v} = \mathbf{w}\mathbf{I}'_0$ has order $\mathcal{O}(n)$. The computation of the gradient $\mathbf{I}'_1$ is trivial (by assumption). Therefore, the order of computing $\mathbf{F}'$ is $\mathcal{O}(n)$.

6. Vega estimation

6.1. Model elementary vegas

Assume that the set of swap-rates at time T_r is given by a vector function with the following inputs: the rates at time T_{r-1}, the set of pseudo-root elements $\{a_{ik}(T_{r-1})\}$ over the step (T_{r-1}, T_r), and a set of random variates $\mathbf{Z}$

$$\mathbf{SR}(T_r) = \mathbf{F}_r\Big(\mathbf{SR}(T_{r-1}), \big\{a_{ik}\big(T_{r-1}\big)\big\}, \mathbf{Z}\Big), \tag{65}$$

as seen from equation (26).

Definition 6.1. *We define the model elementary Vegas of an IRD with discounted price g to be the following partial derivatives*

$$\frac{\partial g}{\partial a_{ik}(T_{r-1})} = P_N(0)\frac{\partial \overline{f}(T_m)}{\partial a_{ik}(T_{r-1})},$$

for $i = 0, \ldots, n-1$, $k = 1, \ldots, F$ and $r = 0, \ldots, m$.

6.1.1. Adjoint method

From (65) and using the chain rule,

$$\begin{aligned}\frac{\partial \overline{f}(T_m)}{\partial a_{ik}(T_{r-1})} &= \frac{\partial \overline{f}(T_m)}{\partial \mathbf{SR}(T_m)}\frac{\partial \mathbf{SR}(T_m)}{\partial \mathbf{SR}(T_{m-1})} \\ &\quad \cdots \frac{\partial \mathbf{SR}(T_{r+1})}{\partial \mathbf{SR}(T_r)}\frac{\partial \mathbf{SR}(T_r)}{\partial a_{ik}(T_{r-1})} \\ &= \mathbf{V}(T_r)\frac{\partial \mathbf{F}_r}{\partial a_{ik}(T_{r-1})}, \end{aligned} \tag{66}$$

where $\mathbf{V}(T_r)$ is the adjoint vector defined in (34). In order to compute model elementary Vegas, we only need to compute the gradient $\partial \mathbf{F}_r/\partial a_{ik}(T_{r-1})$ since the adjoint vectors have already been computed in the Delta calculations. From (26), the ith entry of the gradient $\partial \mathbf{F}_r/\partial a_{ik}(T_{r-1})$ is given by

$$\begin{aligned}\frac{\partial \mathrm{SR}_i(T_r)}{\partial a_{ik}(T_{r-1})} &= \big(\mathrm{SR}_i(T_r) + \alpha_i\big)\Bigg[\frac{\partial \mu_i^{(N)}}{\partial a_{ik}(T_{r-1})} \\ &\quad + \big(Z_k - a_{ik}(T_{r-1})\big)\delta_{ij}\Bigg], \end{aligned} \tag{67}$$

where δ_{ij} is Kronecker's delta. Similar to the computation of Deltas, there is no closed-form solutions for (67) due to the complicated form of the drifts.

6.1.2. Sub-mappings

However, if we modify the mappings in (35) to

$$\begin{bmatrix}\{a_{ik}(T_{r-1})\}\\ \mathbf{SR}(T_{r-1})\\ \overline{\mathbf{A}}(T_{r-1})\end{bmatrix} \xrightarrow{\mathbf{J}_{r,1}} \begin{bmatrix}\{a_{ik}(T_{r-1})\}\\ \mathbf{SR}(T_{r-1})\\ \underline{\mu}^{(N)}(T_{r-1})\end{bmatrix} \xrightarrow{\mathbf{J}_{r,2}} \mathbf{SR}(T_r), \tag{68}$$

then the first nF entries of the vector, $\mathbf{V}(T_r)\mathbf{J}'_{r,2}\mathbf{J}'_{r,1}$, will be equal to

$$\mathbf{V}(T_r)\frac{\partial \mathbf{F}_r}{\partial a_{ik}(T_{r-1})},$$

for $j = 0, \ldots, n-1$ and $k = 1, \ldots, F$. If each of the vector multiplications has order $\mathcal{O}(nF)$, then the order to compute model elementary Vegas will be $\mathcal{O}(nF)$ per step. We show how to carry out efficient Vega computations in the DDcmSMM and the DDciSMM in the following sub-sections.

6.2. Adjoint method in the DDcmSMM

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

$$\mathbf{V}(T_0)\mathbf{J}'_{0,2}\mathbf{J}'_{0,1}. \tag{69}$$

i.e. the adjoint operations for the first step. We omit the superscript 0 in a^0_{jk} as it causes no confusion.

6.2.1. Computational order of $\mathbf{wJ}'_{0,1}$

We divide the mapping

$$\begin{bmatrix}\{a_{jk}\}\\ \mathbf{SR}(0)\\ \overline{\mathbf{A}}(0)\end{bmatrix} \xrightarrow{\mathbf{J}_{0,1}} \begin{bmatrix}\{a_{jk}\}\\ \mathbf{SR}(0)\\ \underline{\mu}^{(n)}(0)\end{bmatrix}$$

into the following sub-mappings

$$\begin{bmatrix}\{a_{jk}\}\\ \mathbf{SR}(0)\\ \overline{\mathbf{A}}(0)\end{bmatrix} \xrightarrow{\mathbf{K}_{0,0}} \begin{bmatrix}\{a_{jk}\}\\ \mathbf{SR}(0)\\ \overline{\mathbf{A}}(0)\\ \big\{\langle Z_k, \overline{P}^r_i\rangle\big\}_{i=n-1}^{n-1}\end{bmatrix}$$

$$\xrightarrow{\mathbf{K}_{0,1}} \begin{bmatrix}\{a_{jk}\}\\ \mathbf{SR}(0)\\ \overline{\mathbf{A}}(0)\\ \big\{\langle Z_k, \overline{P}^r_i\rangle\big\}_{i=n-1}^{n-1}\\ \big\{\langle Z_k, \overline{A}^r_i\rangle\big\}_{i=n-2}^{n-2}\end{bmatrix}$$

$$\xrightarrow{\mathbf{K}_{0,2}} \begin{bmatrix}\{a_{jk}\}\\ \mathbf{SR}(0)\\ \overline{\mathbf{A}}(0)\\ \big\{\langle Z_k, \overline{P}^r_i\rangle\big\}_{i=n-2}^{n-1}\\ \big\{\langle Z_k, \overline{A}^r_i\rangle\big\}_{i=n-2}^{n-2}\end{bmatrix}$$

$$\xrightarrow{\mathbf{K}_{0,3}} \begin{bmatrix}\{a_{jk}\}\\ \mathbf{SR}(0)\\ \overline{\mathbf{A}}(0)\\ \big\{\langle Z_k, \overline{P}^r_i\rangle\big\}_{i=n-2}^{n-1}\\ \big\{\langle Z_k, \overline{A}^r_i\rangle\big\}_{i=n-3}^{n-2}\end{bmatrix}$$

$$\cdots \xrightarrow{\mathbf{K}_{0,2(n-2)+1}} \begin{bmatrix}\{a_{jk}\}\\ \mathbf{SR}(0)\\ \overline{\mathbf{A}}(0)\\ \big\{\langle Z_k, \overline{P}^r_i\rangle\big\}_{i=1}^{n-1}\\ \big\{\langle Z_k, \overline{A}^r_i\rangle\big\}_{i=0}^{n-2}\end{bmatrix}$$

$$\xrightarrow{\mathbf{K}_0} \begin{bmatrix} \{a_{jk}\} \\ \mathbf{SR}(0) \\ \underline{\mu}^{(n)}(0) \end{bmatrix} \tag{70}$$

The Jacobian matrices $\mathbf{K}_{0,i}$ and $\mathbf{K}_0$ have similar structures to the Jacobian matrices $\mathbf{H}_{0,i}$ and $\mathbf{H}_0$ as in (43):

1. In addition to the non-zero partial derivatives in section 4.2.1, $\mathbf{K}_{0,j}$ when j is even has an extra partial derivative with respect to a_{jk} equal to
$$\frac{\partial}{\partial a_{jk}}\left\langle Z_k, \overline{P}_j \right\rangle = \left(\mathrm{SR}_j^r + \alpha_j\right)\overline{A}_j^r. \tag{71}$$
2. In addition to the non-zero partial derivatives in section 4.2.2, $\mathbf{K}_{0,j}$ when j is odd has an extra partial derivative with respect to a_{jk} equal to
$$\frac{\partial}{\partial a_{jk}}\left\langle Z_k, \overline{A}_j^r \right\rangle = \tau_j\left(\mathrm{SR}_{j+1}^r + \alpha_{j+1}\right)\overline{A}_{j+1}^r. \tag{72}$$
3. In addition to the non-zero partial derivatives in section 4.2.3, $\mathbf{K}_0$ has an extra partial derivative with respect to a_{jk} equal to
$$\frac{\partial}{\partial a_{jk}}\mu_j^{(n)} = -\frac{1}{\overline{A}_j^r}\left\langle Z_k, \overline{A}_j^r \right\rangle. \tag{73}$$

The number of mappings $\mathbf{K}_{0,j}$ depends on n, and each operation $\mathbf{v} = \mathbf{w}K'_{0,j}$, $j = 0, \ldots, 2(n-2)+1$, has a constant order. Similar to the Delta computations, the operation $\mathbf{v} = \mathbf{w}K'_0$ has order $O(nF)$. Hence, the order of the operation

$$\mathbf{v} = \mathbf{w}\mathbf{K}'_0\mathbf{K}'_{0,2(n-2)+1}\cdots\mathbf{K}'_{0,0} = \mathbf{w}\mathbf{J}'_{0,1}$$

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

6.2.2. Computational order of $\mathbf{w}\mathbf{J}'_{0,2}$

We use (50) to evolve the constant maturity swap-rates. The non-zero partial derivatives of $\mathbf{J}'_{0,2}$ are

$$\begin{cases} \frac{\partial \mathrm{SR}_j^r(T_0)}{\partial a_{jk}} = \left[\mathrm{SR}_j^r(T_0) + \alpha_j\right]\left(Z_k - a_{jk}\right), \\ \frac{\partial \mathrm{SR}_j^r(T_0)}{\partial \mathrm{SR}_j^r(0)} = \frac{\mathrm{SR}_j^r(T_0)+\alpha_j}{\mathrm{SR}_j^r(0)+\alpha_j}, \\ \frac{\partial \mathrm{SR}_j^r(T_0)}{\partial \mu_j^{(n)}} = \mathrm{SR}_j^r(T_0) + \alpha_j, \end{cases} \tag{74}$$

for $j = 0, \ldots, n-1$ and $k = 1, \ldots, F$. Hence the operation $\mathbf{v} = \mathbf{w}\mathbf{J}'_{0,2}$ has $2n - 1 + nF$ computations so that its order is $\mathcal{O}(nF)$.

6.2.3. Total computational order

Given that the computational orders of

$$\mathbf{v} = \mathbf{w}\mathbf{J}'_{0,1} \quad \text{and} \quad \mathbf{v} = \mathbf{w}\mathbf{J}'_{0,2}$$

are $\mathcal{O}(nF)$. Hence the adjoint operation

$$\mathbf{V}(T_0)\mathbf{J}'_{0,2}\mathbf{J}'_{0,1}$$

has order $\mathcal{O}(nF)$.

6.3. Adjoint method in the DDctSMM

If we set $r = n$ in the DDcmSMM, we then have the DDctSMM. Thus we discuss Vega computations in the DDctSMM no further as it is a special case of the more general DDcmSMM.

6.4. Adjoint method in the DDciSMM

The DDciSMM is a one-period model, we consider the following mappings

$$\begin{bmatrix} \{a_{jk}\} \\ \mathbf{SR}(0) \\ \overline{\mathbf{A}}(0) \end{bmatrix} \xrightarrow{\mathbf{J}_{0,1}} \begin{bmatrix} \{a_{jk}\} \\ \mathbf{SR}(0) \\ \underline{\mu}^{(0)}(0) \end{bmatrix} \xrightarrow{\mathbf{J}_{0,2}} \mathbf{SR}(T_0). \tag{75}$$

We will show that the following operation

$$\mathbf{V}(T_0)\mathbf{J}'_{0,2}\mathbf{J}'_{0,1}$$

has order $\mathcal{O}(nF)$ to prove that the adjoint operations needed to compute model Vegas has order $\mathcal{O}(nF)$.

6.4.1. Computational order of $\mathbf{w}\mathbf{J}'_{0,1}$

We divide the mapping

$$\begin{bmatrix} \{a_{jk}\} \\ \mathbf{SR}(0) \\ \overline{\mathbf{A}}(0) \end{bmatrix} \xrightarrow{\mathbf{J}_{0,1}} \begin{bmatrix} \{a_{jk}\} \\ \mathbf{SR}(0) \\ \underline{\mu}^{(0)}(0) \end{bmatrix}$$

into the following sub-mappings

$$\begin{bmatrix} \{a_{jk}\} \\ \mathbf{SR}(0) \\ \overline{\mathbf{A}}(0) \end{bmatrix} \xrightarrow{\mathbf{K}_{0,0}} \begin{bmatrix} \{a_{jk}\} \\ \mathbf{SR}(0) \\ \overline{\mathbf{A}}(0) \\ \left\{\langle Z_k, \overline{A}_{0i}\rangle\right\}_{i=1}^{1} \end{bmatrix}$$

$$\xrightarrow{\mathbf{K}_{0,1}} \begin{bmatrix} \{a_{jk}\} \\ \mathbf{SR}(0) \\ \overline{\mathbf{A}}(0) \\ \left\{\langle Z_k, \overline{A}_{0i}\rangle\right\}_{i=1}^{2} \end{bmatrix}$$

$$\cdots \xrightarrow{\mathbf{K}_{0,n-1}} \begin{bmatrix} \{a_{jk}\} \\ \mathbf{SR}(0) \\ \overline{\mathbf{A}}(0) \\ \{\langle Z_k, \overline{A}_{0i}\rangle\}_{i=1}^{n} \end{bmatrix} \xrightarrow{\mathbf{K}_0} \begin{bmatrix} \{a_{jk}\} \\ \mathbf{SR}(0) \\ \underline{\mu}^{(0)}(0) \end{bmatrix} \tag{76}$$

The Jacobian matrices $\mathbf{K}_{0,i}$ and $\mathbf{K}_0$ have similar structures to the Jacobian matrices $\mathbf{H}_{0,i}$ and $\mathbf{H}_0$ in (58):

1. In addition to the three non-zero partial derivatives in section 5.2.1, $\mathbf{K}_{0,i}$ has an extra partial derivative with respect to a_{jk} equal to

$$\frac{\partial}{\partial a_{jk}}\left\langle Z_k, \overline{A}_{0j}\right\rangle = -\frac{\tau_{j-1}\left(\mathrm{SR}_{0j}+\alpha_j\right)\overline{A}_{0j}}{1+\tau_{j-1}\mathrm{SR}_{0j}}. \tag{77}$$

   Thus the operation $\mathbf{v} = \mathbf{w}\mathbf{K}_{0,j}$ has $4F$ computations.
2. In addition to the three non-zero partial derivatives in section 5.2.2, $\mathbf{K}_0$ has an extra partial derivative with respect to a_{jk} equal to

$$\frac{\partial \mu_j^{(0)}}{\partial a_{jk}} = -\frac{1}{\overline{A}_{0j}}\langle Z_k, \overline{A}_{0j}\rangle. \tag{78}$$

   Thus $\mathbf{K}_0'$ has an additional nF partial derivatives equal to $\partial\mu_j^{(0)}/\partial a_{jk}$ compared with $\mathbf{H}_0'$. Therefore the operation $\mathbf{v} = \mathbf{w}K_0'$ has order $\mathcal{O}(nF)$.

The number of mappings $\mathbf{K}_{0,j}$, whose operation has a constant order, is equal to n. Hence, the order of the operation $\mathbf{v} = \mathbf{w}\mathbf{K}_0'\mathbf{K}_{0,n-1}'\cdots\mathbf{K}_{0,0}' = \mathbf{w}\mathbf{J}_{0,1}'$ is $\mathcal{O}(nF)$.

6.4.2. Computational order of $\mathbf{w}\mathbf{J}_{0,2}'$

We use (62) to evolve the co-initial swap-rates. The non-zero partial derivatives of $\mathbf{J}_{0,2}'$ are

$$\begin{cases} \frac{\partial \mathrm{SR}_{0j}(T_0)}{\partial a_{jk}} = \left[\mathrm{SR}_{0j}(T_0)+\alpha_j\right]\left(Z_k - a_{jk}\right), \\ \frac{\partial \mathrm{SR}_{0j}(T_0)}{\partial \mathrm{SR}_{0j}(0)} = \frac{\mathrm{SR}_{0j}(T_0)+\alpha_j}{\mathrm{SR}_{0j}(0)+\alpha_j}, \\ \frac{\partial \mathrm{SR}_{0j}(T_0)}{\partial \mu_j^{(0)}} = \mathrm{SR}_{0j}(T_0)+\alpha_j, \end{cases} \tag{79}$$

for $j = 1, \ldots, n$ and $k = 1, \ldots, F$. Hence the operation $\mathbf{v} = \mathbf{w}\mathbf{J}_{0,2}'$ has $2n + nF$ computations so that its order is $\mathcal{O}(nF)$.

6.4.3. Total computational order

Similar to the case of DDcmSMM, given the adjoint vector $\mathbf{V}(T_0)$, the operation $\mathbf{V}(T_0)\mathbf{J}_{0,2}'\mathbf{J}_{0,1}'$ has order $\mathcal{O}(nF)$.

6.5. Market vegas

Traders and quants are not interested in the sensitivities to model parameters such as model Vegas, they are interested in the sensitivities to market observable interest rates and interest rate derivatives. The problem of converting model elementary Vegas to market Vegas in the LMM has been addressed in Joshi and Kwon (2009).

Suppose we evolve the model m steps in the simulation. The methods discussed in this section will produce an $n \times F$ matrix of model elementary Vegas at each step so that we have m matrices overall. Joshi and Kwon (2009) show that the market Vegas can be computed as linear combinations of the $m \times n \times F$ numbers, $\partial\mathbf{F}/\partial a_{jk}^r$. Similar approaches will apply to generic market models, we leave this for future research.

7. Numerical testing

7.1. Market data

The tenor structure is: $T_j = (j+1)\times 0.5$, $j = 0, \ldots, n$. The zero-coupon bond prices are given by $P_j(0) = e^{-0.05\cdot T_j}$. This corresponds to swap-rates all being equal to 5.063%. A flat volatility structure with log-volatility $\sigma_i = 10\%$ is used. The instantaneous correlations are given by $\rho_{ij} = e^{-0.1|T_i - T_j|}$. We fix the number of factors (F) to be 3.

7.2. Timing tests for deltas

We have shown that the order of the adjoint method is $\mathcal{O}(nF)$ per step in the DDcmSMM and the DDciSMM. 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 algorithm for n steps, we should obtain timings that are parabolic in n.

In each of the following cases, we ran 163,840 paths on a European swaption and estimate Deltas and model Vegas using the efficient algorithms discussed in this paper. We ran the Monte Carlo simulations on a computer with an Intel Core 2 1.6GHz CPU and 1GB RAM, using single-threaded C++ code.

7.2.1. Computational order

- In the DDcmSMM: We set $r = 3$. We plot time required to compute the Deltas of a European swaption maturing at T_n against n and we fit a

parabola to the graph in figure 1. Note that similar timing results are obtained for different values of r in $\mathrm{SR}_{j,j+r}$.

- In the DDciSMM: We plot time required to compute the Deltas of a European swaption maturing at T_0 against n, and we fit a line to the graph in figure 2.

These results show that the computational order of Delta computations is $\mathcal{O}(nF)$ per step.

7.2.2. Ratio of delta time to pricing time

We list the computational times required to compute prices, Deltas and their ratios in the DDcmSMM in table 1. We see that it does not take an excessive amount of extra time to compute all the (model) Deltas.

7.3. Timing tests for vegas

Sensitivities of IRDs to constant volatility parameters are given by linear combination of model elementary Vegas via

$$\frac{\partial g}{\partial \sigma_i^\alpha} = \sum_{r,k} \frac{\partial g}{\partial a_{ik}^r} \frac{a_{ik}^r}{\sigma_i^\alpha}. \tag{80}$$

7.3.1. Model elementary vegas

Similar to the timing tests for Deltas, we compute model elementary Vegas in the DDctSMM, DDcmSMM and DDciSMM.

1. We estimate $n \times n \times F$ model elementary Vegas in DDcmSMM. We plot time against n and fit a parabola to the graph in figure 3.
2. Since the DDctSMM is a special case of the DDcmSMM, similar results are obtained for the DDctSMM.
3. We estimate $n \times F$ model elementary Vegas in DDciSMM. We plot time against n and fit a line to the graph in figure 4.

Furthermore, we can see that the time required to compute model Vegas only exceeds the time required to compute Deltas by a small proportion.

7.3.2. Market vegas

As mentioned in section 6.4: we compute all the model elementary Vegas first, then we use linear

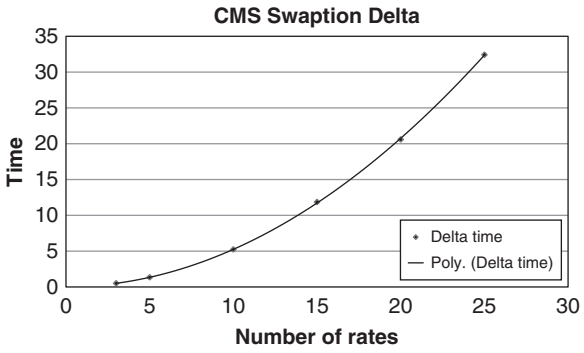

Fig. 1. Graphs of number of rates against the time required to compute Deltas of a T_n European swaption with $F = 3$ in the cmSMM ($r = 3$).

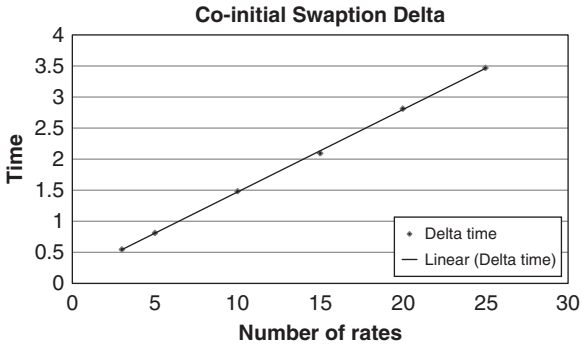

Fig. 2. Graphs of number of rates against the time required to compute Deltas of a T_0 European swaption with $F = 3$ in the ciSMM.

Table 1

Ratios of the time required to compute Deltas to the time required to compute prices in the DDcmSMM ($r = 3$).

n	Pricing	Delta	Ratio
5	0.91	1.34	1.48
10	4.20	5.24	1.25
15	8.83	11.86	1.34
20	15.20	20.59	1.35
25	23.39	32.42	1.39

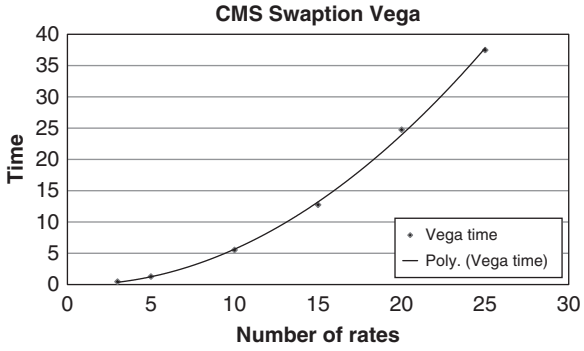

Fig. 3. Graphs of number of rates against the time required to compute Vegas of a T_n European swaption with $F = 3$ in the cmSMM ($r = 3$).

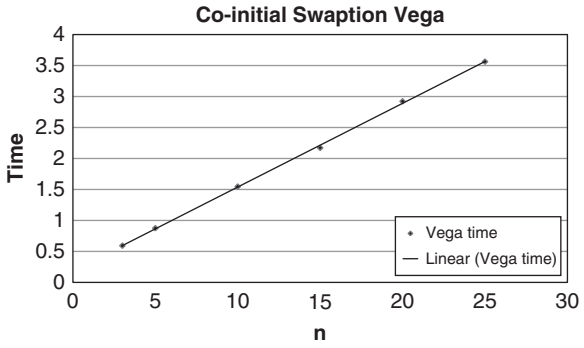

Fig. 4. Graphs of number of rates against the time required to compute Vegas of a T_0 European swaption with $F = 3$ in the ciSMM.

Table 2
Ratios of the time required to compute market Vegas only to the time required to compute Deltas only in the DDcmSMM ($r = 3$)

Number of market Vegas	5	10	20	40	60
Ratio	1.4	1.4	1.3	1.4	1.4

combinations of those to calculate market Vegas. We compute market Vegas using the model elementary Vegas computed in a DDcmSMM with 60 rates across 60 steps. We compute the ratios of the time required to compute the market Vegas only to the time required to compute Deltas only in table 2.

Since we carry out the linear combinations outside the main simulation loop so that computing market Vegas costs little extra time compared with computing model elementary Vegas. In particular, if we compute 60 market Vegas, the ratio is no greater than 1.4.

7.4. *Extension to path-dependent and early-exercisable IRDs*

For the case of computing Greeks of path-dependent products, it is necessary to introduce additional auxiliary variables which encapsulate the path-dependence – e.g. the current coupon in a ratchet, or the running maximum in a look-back. This extension is straight-forward: the main effect is to increase the dimensionality of the state-space and therefore the Jacobians by the number of state-variables.

For Bermudan-type products, Joshi and Yang (in press a) shows how to compute Deltas of Bermudan swaptions in a DDctSMM. It is trivial to extend the technique to compute Deltas and Vegas of Bermudan-type IRDs in other swap-rate market models. We refer interested readers to that paper and Leclerc et al. (2009). More efficient approaches for

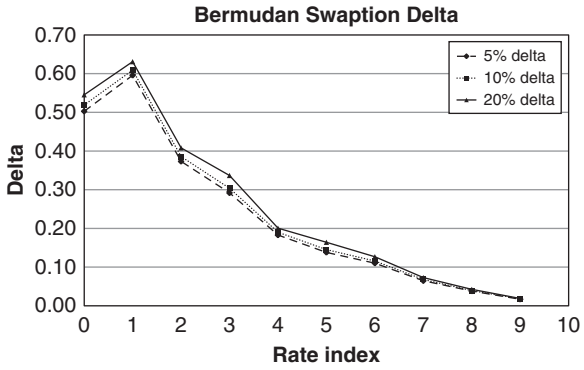

Fig. 5. Graphs of Deltas of a 10-period Bermudan swaption in the ctSMM corresponding to 3 sets of flat volatility parameters.

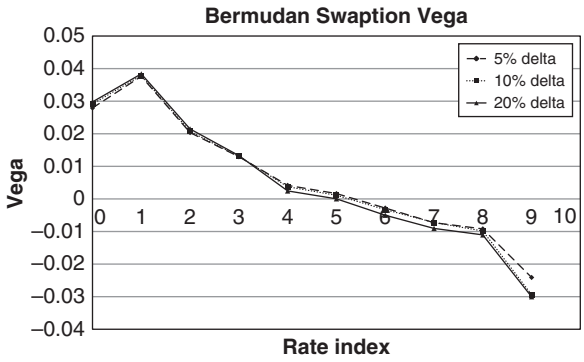

Fig. 6. Graphs of Vegas of a 10-period Bermudan swaption in the ctSMM corresponding to 3 sets of flat volatility parameters.

Greeking Bermudan options using adjoint methods were later introduced in Denson and Joshi (in press).

7.5. *Effects of different model parameters on deltas and vegas*

Deltas and Vegas (computed using equation (80)) of an at-the-money Bermudan swaption corresponding to three sets of flat volatility parameters in the DDctSMM are given in figures 5 and 6. The main observation is that the magnitude of Deltas and Vegas do not vary substantially even though we increase flat volatility from 5% to 20%.

8. Conclusion

We have presented efficient algorithms to implement the adjoint method to estimate Deltas of IRDs in the DDcmSMM and the DDciSMM. We have also shown how to extend the method to compute Vegas of

IRDs in several generic market models including the DDctSMM.

The timing tests confirm that the computational complexity is $\mathcal{O}(nF)$ per step in generic market models, and that it does not take substantial additional time to compute Vegas once all the adjoint vectors have been computed in the Delta computations.

References

Brace, A., Gatarek, D., Musiela, M., 1997. The market model of interest rate dynamics. Mathematical Finance 7, 127–155.

Brace, A., 2009. Engineering BGM, Chapman & Hall/CRC Press.

Brigo, D., Mercurio, F., 2006. Interest Rate Models: Theory and Practice, second ed. Springer-Verlag, Berlin–Heidelberg–New York.

Capriotti, L., Giles, M., 2010. Fast correlation Greeks by adjoint algorithmic differentiation. RISK 77–83.

Denson, N., Joshi, M.S., in press. Fast and accurate Greeks for the LIBOR market model. Journal of Computational Finance 14 (4), 115–140.

Galluccio, S., Hunter, C., 2004. The co-initial swap market model. Economic Notes 33 (2), 209–232.

Galluccio, S., Ly, J.M., Huang, Z., Scaillet, O., 2007. Theory and calibration of swap market models. Mathematical Finance 17, 111–141.

Giles, M., Glasserman, P., 2006. Smoking adjoints: fast Monte Carlo Greeks. Risk 92–96.

Glasserman, P., 2004. Monte Carlo Methods in Financing Engineering Springer-Verlag, Berlin–Heidelberg–New York.

Jamshidian, F., 1997. LIBOR and swap market models and measures. Finance and Stochastics 1, 293–330.

Joshi, M., 2003a. The Concepts and Practice of Mathematical Finance, Cambridge University Press.

Joshi, M., 2003b. Rapid drift computations in the LIBOR market model. Wilmott 84–85.

Joshi, M. Liesch, L., 2007. Effective implementation of generic market models. ASTIN Bulletin 37 (2), 453–473.

Joshi, M., Kwon, O.K., in press. Monte Carlo market Greeks in the displaced diffusion LIBOR market model. Journal of Risk.

Joshi, M., Yang, C., 2010. Fast and accurate pricing and hedging of long-dated CMS spread options. International Journal of Theoretical and Applied Finance 13 (6), 839–865.

Joshi, M., Yang, C., in press a. Fast Delta computations in the swap-rate market model. Journal of Economic Dynamics and Control 35, 764–775.

Joshi, M., Yang, C., in press b. Algorithmic Hessians and the fast computation of cross-Gamma risk. Transactions of the IIE.

Leclerc, M., Liang, Q., Scheider, I., 2009. Fast Monte Carlo Bermudan Greeks. Risk 84–88.

Pietersz, R., van Regenmortel, M., 2005. Generic market models. Finance and Stochastics 10(4), 507–528.