# Efficient Monte Carlo Simulation of the Delta Vector of a Bermudan Swaption in the LIBOR Market Model

**Ralf Korn**
Fachbereich Mathematik, TU Kaiserslautern (Germany),
e-mail: korn@mathematik.uni-kl.de
**Qian Liang**
Fachbereich Mathematik, TU Kaiserslautern (Germany)

**Abstract**
We consider various methods for an efficient numerical computation of the Delta vector of a Bermudan swaption in a LIBOR market model setting. All methods are based on the least-squares Monte Carlo method of Longstaff & Schwarz (2001). Among them, we present three new approaches: a new version of the adjoint method introduced by Leclerc et al. (2009), a pathwise method based on the use of the forward drift, and a likelihood ratio approach. The new version of the adjoint method shows superior performance compared with the other methods.

## 1 Introduction

The computation of Bermudan swaption prices and price sensitivities in a LIBOR market model (LMM) setting is a demanding task due to the high dimension of the LMM and the Bermudan character of the payoffs. As Bermudan swaptions are amongst the most liquidly traded callable LIBOR exotics, computing their price and sensitivities efficiently is practically very relevant.

In this paper we focus on the calculation of the Delta vector of a Bermudan swaption. Based on the least-squares Monte Carlo (LSM) of Longstaff & Schwarz (2001) for the calculation of the Bermudan swaption price, we compare standard finite difference methods with variants of the pathwise method and the likelihood ratio method. We will in particular introduce a new version of the adjoint method that exhibits superior performance and is easy to understand and to implement.

We will start by introducing different methods for calculating the Deltas of an interest rate derivative under the LMM, and concentrate on the application to Bermudan swaptions thereafter. Finally, we highlight the performance of the different methods via numerical examples.

## 2 Computing Deltas in the LIBOR market model

The Delta $\Delta$ of a derivative is the partial derivative of its price with respect to the initial value of the underlying(s). According to Glasserman (2004) and Korn et al. (2010) there are three main numerical methods to compute it: the *finite difference method* (FDM), the *pathwise method* (PM), and the *likelihood ratio method* (LRM).

The FDM is the industry standard as it is easy to implement. The PM is unbiased and among the three, it has proved to be the most efficient one given the required smoothness assumptions on the payoff functions are satisfied. If the payoff function is not sufficiently smooth to apply the PM, the LRM is an unbiased alternative for the computation of Delta.

In this section, we mainly discuss the last two methods and consider the forward LIBOR rates as the underlyings. For this, we first introduce the basic notations and concept of the LMM as introduced by Brace et al. (1997) and Miltersen et al. (1997).

### 2.1 Basics of the LIBOR market model

We look at the *tenor structure* $0 = T_0 < T_1 < \ldots < T_M = T$ with $\delta_i = T_{i+1} - T_i$ for $i = 0, \ldots, M-1$. Given $M$ zero bonds $B_1(t), \ldots, B_M(t)$ with maturities $T_1, \ldots, T_M$, we introduce the forward LIBOR rate $L_i(t)$ at time $t$ for the time interval $[T_i, T_{i+1})$ as

$$L_i(t) = \frac{1}{\delta_i}\left(\frac{B_i(t) - B_{i+1}(t)}{B_{i+1}(t)}\right) \quad 0 \le t \le T_i, \quad i = 0, \ldots, M-1.$$

Let $\eta(t)$ be the index of the next tenor after $t$, namely $T_{\eta(t)-1} \le t < T_{\eta(t)}$. By using the following process

$$B^*(t) = B_{\eta(t)}(t) \cdot \prod_{j=0}^{\eta(t)-1} \left(1 + \delta_j L_j(T_j)\right), \quad 0 \le t \le T$$

as a numeraire, Jamshidian (1997) introduced a LMM under the *spot measure*. The forward LIBORs $L_i(t)$ under the spot measure have dynamics given by

$$\frac{dL_i(t)}{L_i(t)} = \mu_i(t)dt + \sigma_i^\top(t)dW^*(t), \quad t \in [0, T_i], \quad i = 1, \ldots, M-1 \tag{1}$$

with $W^*$ a $d$-dimensional Brownian motion under the spot measure, deterministic volatility functions $\sigma_i: [0, T) \to \mathbb{R}^d$, and drift terms of the form

$$\mu_i(t) = \sum_{j=\eta(t)}^{i} \frac{\delta_j L_j(t)\sigma_i^\top(t)\sigma_j(t)}{1 + \delta_j L_j(t)}.$$

This path dependence of the drift and the high dimension of the LMM are reasons that Monte Carlo simulation is typically the method of choice for price or sensitivitiy calculations of LIBOR derivatives.

We perform our simulations on a time grid $\{t_0, \ldots, t_N\} \supseteq \{T_0, \ldots, T_{M-1}\}$ with $t_{N_i} = T_i$ for $i = 0, \ldots, M-1$ and thus $N_0 = 0$ and $N_{M-1} = N$. Under the spot measure, we use the Euler discretization of the Itô formula in $\ln(L_i(t))$ on $\{t_0, \ldots, t_N\}$ via the new notations $L_i(n) = L_i(t_n)$, $\mu_i(n) = \mu_i(t_n)$, and $\sigma_i(n) = \sigma_i(t_n)$ with

$$L_i(n+1) = L_i(n) \exp\left(\left(\mu_i(n) - \frac{1}{2}\|\sigma_i(n)\|^2\right)h_n + \sqrt{h_n}\sigma_i^\top(n)Z(n+1)\right) \tag{2}$$

for $n = 0, \ldots, N-1$ and $i = \eta(t_n), \ldots, M-1$ with drift terms

$$\mu_i(n) = \sigma_i^\top(n) \cdot \sum_{j=\eta(t_n)}^{i} \frac{\delta_j L_j(n)\sigma_j(n)}{1 + \delta_j L_j(n)},$$

where $h_n = t_{n+1} - t_n$ for $n = 0, \ldots, N-1$ and $Z(1), \ldots, Z(N) \sim \mathcal{N}(0, I_d)$ are independent. Note that we have $L_i(n) = L_i(N_i)$ for $n \geq N_i$.

## 2.2 Introduction of the pathwise method

To introduce the PM let $g(L(T_{M-1}))$ be the discounted payoff function of a general LIBOR derivative. Then, the Delta vector has the form

$$\mathbf{\Delta}(g(L(T_{M-1})) = \frac{\partial \mathbb{E}^*(g(L(T_{M-1})))}{\partial L(0)}, i = 1, \ldots, M$$

where $\mathbb{E}^*$ denotes the expectation under the spot measure.

PM is based on interchanging the derivative with respect to $L(0)$ with the expectation. We then calculate this derivative along each simulated path as $\frac{\partial g(L(T_{M-1}))}{\partial L(0)}$ and average over all paths. Hence, PM is applicable and unbiased if we have (in vector notation)

$$\Delta(g(L(T_{M-1})) = \mathbb{E}^*\left(\frac{\partial g(L(T_{M-1}))}{\partial L(0)}\right). \tag{3}$$

A sufficient condition for the validity of Equation (3) is the Lipschitz continuity of $g$ (see Glasserman (2004)). The PM comes in two different variants: the forward method (FM) and the adjoint method (AM).

### 2.2.1 The forward method

The FM has been introduced by Glasserman & Zhao (1999). We apply the FM in the LMM as given by the dynamics (1) and the simulation formula (2) under the spot measure. Introducing the notation

$$\Delta(n) = \frac{\partial L(n)}{\partial L(0)}, \quad \text{i.e.} \quad \Delta_{ij}(n) = \frac{\partial L_i(n)}{\partial L_j(0)}$$

for $n = 0, \ldots, N$, $i, j = 1, \ldots, M$, the FM for the Delta vector is given by

$$\mathbf{\Delta}(g(L(N)) = \frac{\partial g(L(N))}{\partial L(0)} = \frac{\partial g(L(N))}{\partial L(N)} \cdot \Delta(N). \tag{4}$$

Further, to calculate the factor $\Delta(N)$ in formula (4), we introduce the $(M \times M)$-matrices $D(n)$, $n = 0, \ldots, N-1$, by

$$D(n) = \frac{\partial L(n+1)}{\partial L(n)}, \quad \text{i.e.} \quad D_{ij}(n) = \frac{\partial L_i(n+1)}{\partial L_j(n)}$$

for $i, j = 0, \ldots, M-1$. By the chain rule we obtain $\Delta(N)$ via the recursion

$$\Delta(n+1) = D(n) \cdot \Delta(n), \quad \text{i.e.} \quad \Delta_{ij}(n+1) = \sum_{k=0}^{M-1} D_{ik}(n) \cdot \Delta_{kj}(n) \tag{5}$$

for $n = 0, \ldots, N-1$ and $i, j = 0, \ldots, M-1$ with the initial matrix $\Delta(0) = I_M$. Taking the derivative of both sides of Equation (2) yields the exact formulas of the recursions (5) in the LMM

$$\Delta_{ij}(n+1) = \frac{L_i(n+1)}{L_i(n)}\Delta_{ij}(n) + L_i(n+1)h_n\sigma_i^\top(n) \cdot \sum_{k=\eta(t_n)}^{i} \frac{\delta_k\sigma_k(n)\Delta_{kj}(n)}{(1 + \delta_k L_k(n))^2}, \ i \geq \eta(t_n) \tag{6}$$

$$\Delta_{ij}(n+1) = \Delta_{ij}(n), \quad i < \eta(t_n) \tag{7}$$

for $n = 0, \ldots, N-1$, $i = 0, \ldots, M-1$, and $j = 0, \ldots, i$. Note that the computing time for (6) is mainly determined by the summation of the right-hand side of (6), which only is $O(M)$ for each $j = 1, \ldots, i$. Hence, the total time per recursion step is bounded by $O(M^2)$.

Comparing the recursions (6), (7) with the equation (5) yields the representation of the matrix $D(n)$ in the LMM as

$$D(n) = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & & D_{\eta(t_n),\eta(t_n)}(n) & & \\ & & & \vdots & \ddots & \\ & & & D_{M-1,\eta(t_n)}(n) & \cdots & D_{M-1,M-1}(n) \end{pmatrix} \tag{8}$$

with entries

$$D_{ii}(n) = \begin{cases} 1 & i < \eta(t_n) \\ \dfrac{L_i(n+1)}{L_i(n)} + \dfrac{L_i(n+1)\|\sigma_i(n)\|^2 \delta_i h_n}{(1 + \delta_i L_i(n))^2} & i \geq \eta(t_n) \end{cases}$$

$$D_{ij}(n) = \begin{cases} \dfrac{L_i(n+1)\sigma_i^\top(n)\sigma_j(n)\delta_j h_n}{(1 + \delta_i L_i(n))^2} & i > j \geq \eta(t_n) \\ 0 & \text{else.} \end{cases}$$

### 2.2.2 The adjoint method

For presenting the AM of Giles & Glasserman (2006) we use the notation of FM. Its main idea is an inversion of the recursion direction to obtain a more efficient way to compute the Delta. For this, note that from Equation (4) we get

$$
\begin{aligned}
\Delta(g(L(N)) &= \frac{\partial g(L(N))}{\partial L(N)} \cdot \Delta(N) \\
&= \underbrace{\frac{\partial g(L(N))}{\partial L(N)} \cdot D(N-1)D(N-2)\cdots D(0)}_{=V^\top(0)} \cdot \Delta(0) \\
&= V^\top(0), \qquad (9)
\end{aligned}
$$

where $V(0)$ can be computed backwards on the grid $\{t_0,\ldots,t_N\}$ via

$$V(n) = D^\top(n)\cdot V(n+1) \qquad (10)$$

for $n = N-1, \ldots, 0$ starting with

$$(N) = \left(\frac{\partial g(L(N))}{\partial L(N)}\right)^\top. \qquad (11)$$

Relations (9) and (10) constitute AM. Replacing each $D^T(n)$ term in the detailed form of recursion (10),

$$V_i(n) = \sum_{j=1}^{M-1} D_{ij}^\top(n)\cdot V_j(n+1), \qquad (12)$$

by its explicit value, we obtain

$$
\begin{aligned}
V_i(n) &= \frac{L_i(n+1)}{L_i(n)} V_i(n+1) \\
&\quad + \frac{\sigma_i^\top(n)\delta_i h_n}{(1+\delta_i L_i(n))^2} \sum_{j=i}^{M-1} L_j(n+1)V_j(n+1)\sigma_j(n) \quad i \geq \eta(t_n) \qquad (13)\\
V_i(n) &= V_i(n+1) \qquad i < \eta(t_n) \qquad (14)
\end{aligned}
$$

for $n = N-1,\ldots, 0$, $i = 0,\ldots, M-1$, and the start vector (11). As the summation on the right-hand side of (13) needs a time effort of $O(M)$, the whole effort per recursion step is bounded by $O(M)$. This leads to a clear advantage of AM compared with the recursion (6) of FM. A disadvantage of AM is that it needs more storage space than FM.

### 2.2.3 Pathwise method under forward drift

To speed up the simulation of the LIBOR paths, Glasserman & Zhao (1999) recommend the so-called *forward drift approximation*. This consists of using the (constant and known) initial LIBOR $L(0)$ in each drift term

$$\mu_i^0(n) = \sigma_i^\top(n)\cdot \sum_{j=\eta(t_n)}^{i} \frac{\delta_j L_j(0)\sigma_j(n)}{1+\delta_j L_j(0)}. \qquad (15)$$

As a direct consequence, we now have a representation for the forward drift approximation of the forward LIBORs that does not need a recursion, namely

$$L_i(n) = L_i(0)\exp\left(\sum_{l=0}^{n-1}\left(\left(\mu_i^0(l) - \frac{1}{2}\|\sigma_i(l)\|^2\right)h_l + \sqrt{h_l}\sigma_i^\top(l)Z(l+1)\right)\right) \qquad (16)$$

for $n = 0, \ldots, N_i$ and $i = 0, \ldots, M-1$. We use this concept as the basis for an alternative pathwise method under forward drift (PFD), which is a biased simulation method without the need for recursions. Specifying the simulation formula (16) of the $L_i(N)$ for $i = 0, \ldots, M-1$, we get

$$L_i(N_i) = L_i(0)\exp\left(\sum_{l=0}^{N_i-1}\left(\left(\mu_i^0(l) - \frac{1}{2}\|\sigma_i(l)\|^2\right)h_l + \sqrt{h_l}\sigma_i^\top(l)Z(l+1)\right)\right), \qquad (17)$$

with $Z(1), \ldots, Z(N) \sim \mathcal{N}(0, I_d)$ and $\mu_i^0(l)$ of formula (15). Taking the derivatives on both sides of Equation (17) leads to

$$\Delta_{ij}(N_i) = \mathbf{1}\{i=j\}\frac{L_i(N_i)}{L_i(0)} + \mathbf{1}\{i \geq j\}\frac{L_i(N_i)\delta_j}{(1+\delta_j L_j(0))^2}\sum_{l=0}^{N_j-1} h_l \sigma_i^\top(l)\sigma_j(l) \qquad (18)$$

for $i, j = 0,\ldots, M-1$. This allows a direct simulation of the Delta vector via formula (4). As this method is a direct simulation and as the direction of simulation plays no role, there is no adjoint version. Of course, the bias (by using only the initial LIBOR) is a disadvantage. However, the method is fast and much easier to implement than the exact pathwise methods.

## 2.3 The likelihood ratio method under forward drift

The forward drift approximation of the last setting and the ideas given in Glasserman and Zhao (1999) allow the use of an LRM to compute the Delta vector of a LIBOR derivative. For this, we consider the logarithm of the forward drift approximation in relation (17):

$$\ln L_i(N_i) = \ln L_i(0) + \sum_{l=0}^{N_i-1}\left(\mu_i^0(l) - \frac{1}{2}\|\sigma_i(l)\|^2\right)h_l + \sum_{l=0}^{N_i-1}\sqrt{h_l}\sigma_i^\top(l)Z(l+1) \qquad (19)$$

for $i = 1,\ldots, M-1$, $Z(1), \ldots, Z(N) \sim \mathcal{N}(0, I_d)$, and $\mu_i^0(l)$ as given in (15). We then rewrite formula (19) in the common vector form

$$X(L(0)) = \bar{\mu}(L(0)) + B_{L(0)}\cdot Z_{L(0)}$$

with

$$X(L(0)) = \begin{pmatrix} \ln L_1(N_1) \\ \vdots \\ \ln L_{M-1}(N) \end{pmatrix} \in \mathbb{R}^{M-1} \qquad Z_{L(0)} = \begin{pmatrix} Z(1) \\ \vdots \\ Z(N) \end{pmatrix} \in \mathbb{R}^{N*d}$$

$$\bar{\mu}(L(0)) = \begin{pmatrix} \ln L_1(0) + \sum_{l=0}^{N_1-1}\left(\mu_{M-1}^0(l) - \frac{1}{2}\|\sigma_{M-1}(l)\|^2\right)h_l \\ \vdots \\ \ln L_{M-1}(0) + \sum_{l=0}^{N-1}\left(\mu_{M-1}^0(l) - \frac{1}{2}\|\sigma_{M-1}(l)\|^2\right)h_l \end{pmatrix} \in \mathbb{R}^{M-1}$$

$$\underbrace{\begin{pmatrix} \sqrt{h_0}\sigma_1^\top(0) & \cdots & \sqrt{h_{N_1-1}}\sigma_1^\top(N_1-1) & & \\ \vdots & & \vdots & \ddots & \\ \sqrt{h_0}\sigma_{M-1}^\top(0) & \cdots & \sqrt{h_{N_1-1}}\sigma_{M-1}^\top(N_1-1) & \cdots & \sqrt{h_{N-1}}\sigma_{M-1}^\top(N-1) \end{pmatrix}}_{=B_{L(0)}\in\mathbb{R}^{(M-1)\times(N*d)}},$$

where the matrix $B_{L(0)}$ has rank $M-1$ and thus $\Sigma_{L(0)} = B_{L(0)}\cdot B_{L(0)}^T$ also has rank $M-1$. Also, we have $X(L(0)) \sim \mathcal{N}(\bar{\mu}(L(0)), \Sigma_{L(0)})$. Using the alternative form of the payoff function $\tilde{g}$

$$\tilde{g}\begin{pmatrix} \ln L_0(0) \\ X(L(0)) \end{pmatrix} = g(L(N))$$

and due to Glasserman & Zhao (1999), we can compute the Delta vector in the forward drift approximation LIBOR framework under the spot measure as:

$$\frac{\partial \mathbb{E}^*(g(L(N)))}{\partial L(0)} = \frac{\partial \mathbb{E}^*\left(\tilde{g}\begin{pmatrix}\ln L_0(0)\\ X(L(0))\end{pmatrix}\right)}{\partial L(0)}$$

$$= \mathbb{E}^*\left(\tilde{g}\begin{pmatrix}\ln L_0(0)\\ X(L(0))\end{pmatrix}(X(L(0)) - \bar{\mu}(L(0)))^\top \Sigma_{L(0)}^{-1} \cdot \frac{\partial \bar{\mu}(L(0))}{\partial L(0)}\right)$$

$$= \mathbb{E}^*\left(g(L(N))(X(L(0)) - \bar{\mu}(L(0)))^\top \Sigma_{L(0)}^{-1} \cdot \frac{\partial \bar{\mu}(L(0))}{\partial L(0)}\right), \qquad (20)$$

where the terms of the matrix $\partial\overline{\mu}(L(0))/\partial L(0)$ are given by

$$\frac{\partial \overline{\mu}_i(L(0))}{\partial L_j(0)} = \frac{\mathbf{1}\{i=j\}}{L_i(0)} + \frac{\mathbf{1}\{i \geq j\}\delta_j}{(1+\delta_j L_j(0))^2}\sum_{l=0}^{N_j-1} h_l \sigma_i^\top(l)\sigma_j(l)$$

for $i, j = 1,\dots, M-1$. If $B_{L(0)}$ is quadratic, i.e. if we have $M-1 = N*d$, then according to Glasserman & Zhao (1999) we obtain the following simplified formula of Equation (20):

$$\frac{\partial \mathbb{E}^*(g(L(N)))}{\partial L(0)} = \mathbb{E}^*\left(g(L(N))Z_{L(0)}^\top \cdot \underbrace{B_{L(0)}^{-1}\frac{\partial \bar{\mu}(L(0))}{\partial L(0)}}_{=B^{-1}\cdot\bar{\mu}'}\right). \qquad (21)$$

Note in particular that the factor $B^{-1}\cdot\overline{\mu}'$ in (21) is independent of the random numbers and thus only has to be calculated once, independent of all paths. The advantage of the LRM is that the method needs no smoothness assumptions on the payoff functions. It is further fast to compute, but tends to be inaccurate as it is both biased and typically admits a high variance.

## 3 Computing the Delta vector of a Bermudan swaption

In this section we specialize to the calculation of the Deltas of a Bermudan swaption. We will mainly build on the idea of Piterbarg (2003), who used the LIBOR paths simulated in the LSM algorithm of Longstaff & Schwarz (2001) together with the computed optimal exercise times to apply the forward version of the pathwise method for the calculation of the Deltas.

### 3.1 Bermudan swaption

A $(H\times M)$-*Bermudan swaption* on the tenor structure $0 = T_0 < \dots < T_M = T$ is an interest rate derivative giving its owner the right to enter into a fixed-to-floating interest rate swap at the tenor times between $T_H$ and $T_{M-1}$. If an $(H\times M)$-Bermudan swaption is exercised at time $T_r$ with $H \leq r \leq M-1$ then the owner receives an $(r\times M)$-*interest rate swap*, i.e. a set of *coupons* $\{X_i \mid i = r, \dots, M-1\}$. In the LMM we have

$$X_i = \phi\mathcal{N}\delta_i\left(L_i(T_i) - R\right), \quad i = r, \dots, M-1,$$

with $R$ the fixed interest rate and $\mathcal{N}$ the face value. We call this a *payer swap* in case of $\phi = 1$ or a *receiver swap* for $\phi = -1$. The corresponding Bermudan swaption is then called a *payer-Bermudan swaption* or a *receiver-Bermudan swaption*. Further, each coupon $X_i$ is fixed at time $T_i$ but will be paid out at $T_{i+1}$. Thus, $X_i$ has to be discounted back from $T_{i+1}$ to the current time $T_0$ by using the discount factor under the spot measure

$$PV_{i+1} = \frac{B^*(T_0)}{B^*(T_{i+1})} = \prod_{j=0}^{i}\frac{1}{1+\delta_j L_j(T_j)}.$$

With $T_r$ the optimal (random!) exercise time, the value $V^{BS}_{H\times M}(T_0)$ at time $T_0$ of a Bermudan swaption under the spot measure is given by

$$V^{BS}_{H\times M}(T_0) = \mathbb{E}^*\left(\sum_{i=r}^{M-1} PV_{i+1}X_i\right). \qquad (22)$$

To calculate this price we use the LSM algorithm. The main reason for this is that besides the (approximate) price of the Bermudan swaption, it also yields the (approximately) optimal exercise times $T_r$ along each simulated path.

### 3.2 Forward method

In addition to our discussion of the PM in Section 2.2, we have to take into account that the owner of a Bermudan swaption has the choice of the exercise time. Piterbarg (2003) has shown that for calculating the Deltas of a Bermudan swaption one can use the optimal exercise strategy already determined during the calculation of its price. Thus, a pathwise differentiation with respect to the initial LIBOR vector $L(0)$ is valid. For this, note in particular that in the valuation formula(22) for a Bermudan swaption the payoff components $PV_{i+1}$ and $X_i$ for $i = r, \dots, M-1$ are both continuously differentiable with respect to the components of $L(0)$. Thus, suitable differentiation of both sides of Equation (22) yields

$$\mathbf{\Delta}\left(V^{BS}_{H\times M}(T_0)\right) = \mathbb{E}^*\left(\sum_{i=r}^{M-1}\mathbf{\Delta}\left(PV_{i+1}X_i\right)\right), \qquad (23)$$

i.e. the PM for calculating the Deltas of an $(H\times M)$-Bermudan swaption is unbiased. We can thus formulate the FM in the sense of Piterbarg (2003) as:

1. Execute the LSM algorithm and determine the optimal exercise times along each simulated path.
2. Along each path calculate the Delta vectors of the payments from the optimal exercise time up to the maturity of the Bermudan swaption by the FM of Section 2.2.1 and add all these Delta vectors.
3. Take the average of all results over all paths.

Here, the exact forms of $\Delta_j(PV_{i+1}X_i)$ for $i = r, \dots, M-1$ and $j = 0, \dots, M-1$ have to be calculated by the FM, and yield

$$\mathbf{\Delta}_j\left(PV_{i+1}X_i\right) = \mathbf{1}\{j \leq i\}\cdot PV_{i+1}\cdot\left(\phi\mathcal{N}\delta_i\Delta_{ij}(N_i) - X_i\cdot\sum_{l=j}^{i}\frac{\delta_l\Delta_{lj}(N_l)}{1+\delta_l L_l(N_l)}\right). \qquad (24)$$

The Delta factors $\Delta_{ij}(N)$ are obtained from the recursions (6) and (7).

### 3.3 Adjoint method

The first suggestion of AM for the calculation of the Delta vector of a Bermudan swaption is obtained by replacing Step 2 of FM by:

2. Along each path calculate the Delta vector of the payments from the optimal exercise time up to the maturity of the Bermudan swaption by the AM of Section 2.2 and add all these Delta vectors.

That is, for $i = r, \ldots, M-1$ and $j = 0, \ldots, M-1$ we compute $\Delta_j(PV_{i+1}X_i)$ by AM as presented in Section 2.2.2. To avoid a notational conflict with the multiple use of the adjoint vector $V(\cdot)$ of Section 2.2.2, we introduce

$$V^\top(N_j|T_i) = \frac{\partial\ (PV_{i+1}X_i)}{\partial L(N_j)}$$

for $i = r, \ldots, M-1$ and $j = 0, \ldots, M-1$, leading to

$$V^\top(0|T_i) = \mathbf{\Delta}\ (PV_{i+1}X_i)\ .$$

Due to Equation (9), the Delta vector $\Delta(PV_{i+1}X_i)$ can be obtained by AM as

$$\mathbf{\Delta}\ (PV_{i+1}X_i) = V^\top(0|T_i) = V^\top(N_i|T_i)\cdot D(N_i-1)\cdots D(0) \tag{25}$$

yielding the Delta vector of the Bermudan swaption as

$$\mathbf{\Delta}^\top\left(V^{BS}_{H\times M}(T_0)\right) = \mathbb{E}^*\left(\sum_{i=r}^{M-1} V(0|T_i)\right).$$

To avoid a multiple backward recursion for each time index $i = r, \ldots, M-1$, Leclerc et al. (2009) introduced a linear algebraic superposition vector based on the linearity of the backward recursion (25). After starting at $T_{M-1}$ the superposition vector collects the relevant payments at each payment time. The exact form of this vector is given by

$$\mathbf{V}(n) = \begin{cases} V(N_{M-1}|T_{M-1}) & n = N_{M-1} = N \\ D^\top(N_i)\cdot\mathbf{V}(N_i+1) + V(N_i|T_i) & n \in \{N_i|\, i = r, \ldots, M-2\} \\ D^\top(n)\cdots D^\top(N_r-1)\cdot\mathbf{V}(N_r) & n < N_r \\ D^\top(n)\cdots D^\top(N_{\eta(t_n)}-1)\cdot\mathbf{V}(N_{\eta(t_n)}) & else \end{cases} \tag{26}$$

for $n = N, \ldots, 0$. Using it, a single backward recursion will be enough to apply AM for calculating the Deltas

$$\sum_{i=r}^{M-1} V(0|T_i) = D^\top(0)D^\top(1)\cdots D^\top(N-1)\cdot\mathbf{V}(N). \tag{27}$$

For details, see Leclerc et al. (2009). By formula (29) we obtain a more efficient version of Step 2 of the AM:

2. Combine the AM of Section 2.2 with the superposition vector of Equation (26) to compute the Delta vector via the recursion (27) along each path.

To complete the above AM we note that we obain

$$V^\top(N_i|T_i)_j = \mathbf{1}\{j \le i\}PV_{i+1}\phi\mathcal{N}\delta_i\left(\mathbf{1}\{j=i\} - \frac{\delta_j(L_i(N_i)-R)}{1+\delta_j L_j(N_j)}\right) \tag{28}$$

for $i = r, \ldots, M-1$ and $j = 0, \ldots, M-1$

## 3.4 Adjoint method – New version

In this section we derive an alternative, simplified version of the AM of Leclerc et al. (2009) for calculating the Delta vector of a Bermudan swaption which is unbiased and as efficient as the original version. Note first that due to the formulae (13) and (14), the operation $D^\top(N_i)\ldots D^\top(N-1)\cdot V(N_i|T_i)$ only changes the components $i+1, \ldots, M-1$ of $V(N_i|T_i)$. Due to (28) they equal zero. So we obtain

$$\begin{aligned} V(0|T_i) &= D^\top(0)\cdots D^\top(N_i-1)\cdot V(N_i|T_i) \\ &= D^\top(0)\cdots D^\top(N_i-1)D^\top(N_i)\cdots D^\top(N-1)\cdot V(N_i|T_i) \end{aligned} \tag{29}$$

for $i = r, \ldots, M-1$. Equation (29) yields the basic relation for the new version of the adjoint method (NAM):

$$\begin{aligned} \sum_{i=r}^{M-1} V(0|T_i) &= \sum_{i=r}^{M-1} D^\top(0)\cdots D^\top(N_i-1)\cdot V(N_i|T_i) \\ &= \sum_{i=r}^{M-1} D^\top(0)\cdots D^\top(N_i-1)\cdot D^\top(N_i)\cdots D^\top(N-1)\cdot V(N_i|T_i) \\ &= D^\top(0)\cdots D^\top(N-1)\cdot \sum_{i=r}^{M-1} V(N_i|T_i) \\ &= D^\top(0)\cdots D^\top(N-1)\cdot \sum_{i=r}^{M-1}\left(\frac{\partial(PV_{i+1}X_i)}{\partial L(N_i)}\right)^\top \\ &= D^\top(0)\cdots D^\top(N-1)\cdot \left(\frac{\partial(\sum_{i=r}^{M-1} PV_{i+1}X_i)}{\partial L(N)}\right)^\top . \end{aligned} \tag{30}$$

Clearly, this formula is easier to implement and to understand than the formula (27). It can be interpreted as formally shifting all payments from the optimal exercise onwards to the maturity of the Bermudan swaption and then performing a single backward recursion according to AM back to the current time.

Further, formula (30) shows an advantage of NAM over FM according to the statement of Giles & Glasserman (2006): "*the adjoint method is beneficial if we are interested in calculating sensitivities of a single function with respect to multiple changes in the initial condition*." Although a Bermudan swaption is a portfolio with $M-r$ instruments, the sum of the payments can be identified as a single payment that depends on $M$ parameters, the initial LIBORs $L_0(0), \ldots, L_{M-1}(0)$. This interpretation shows that we are indeed in the situation where NAM outperforms FM. Compared to AM and FM, in the NAM we replace Step 2 of the FM by:

2a. Along each path add the payments from the optimal exercise time to the maturity of the Bermudan swaption.
2b. Along each path calculate the Delta vector with respect the above sum by the AM of Section 2 applied to the above sum.

We also give the formula for the derivative of the sum of the payments after the optimal exercise time in Equation (30):

$$\frac{\partial\left(\sum_{i=r}^{M-1} PV_{i+1}X_i\right)}{\partial L_j(N)} = -\frac{\delta_j\cdot\sum_{i=\max(r,j)}^{M-1} PV_{i+1}X_i}{1+\delta_j L_j(j)} + \mathbf{1}\{j \ge r\}\cdot\phi\mathcal{N}\delta_j\cdot PV_{j+1}$$

for $j = 0, \ldots, M-1$. Let us also point out that the NAM is limited to the calculation of the Deltas because of its structural form, while the AM of Leclerc et al. (2009) can also be used for the calculation of other Greeks.

## 3.5 Pathwise method under forward drift

If we replace Step 2 of the FM by:

2. Along each path calculate the Delta vectors of all payments from the optimal exercise time to maturity by PFD of Section 2.2.3 and add all these Delta vectors.

we obtain the PFD method. It is biased as we apply an unbiased method to an approximate model. Note that we compute $\Delta_j(PV_{i+1}X_i)$ for all $i = r, \ldots, M-1, j = 0, \ldots, M-1$ via Equation (24) by the pathwise method under forward drift. There, the matrix $\Delta(N)$ is derived from the direct simulation formula (18).

### 3.6 Likelihood ratio method under forward drift

The LRM for the Bermudan swaption Deltas under forward drift as in Section 3.5 is given by the LRM formula for the components of the $\Delta(PV_{i+1}X_i)$ for $i = r, \ldots, M-1$ (see formula (20)):

$$\mathbf{\Delta}_j\,(PV_{i+1}X_i) = (PV_{i+1}X_i)\lfloor(X(L(0)) - \overline{\mu}(L(0)))\rfloor_i^\top \left[\Sigma_{L(0)}^{-1} \cdot \frac{\partial\overline{\mu}(L(0))}{\partial L_j(0)}\right]_i$$

for $j = 1, \ldots, M-1$, using the following notation for a matrix or vector $A$:

$$\lfloor A\rfloor_a = A \cdot \begin{pmatrix} I_a & 0 \\ 0 & 0 \end{pmatrix}.$$

If the matrix $B_{L(0)}$ is quadratic, we obtain the simplified version as formula (21):

$$\mathbf{\Delta}_j\,(PV_{i+1}X_i) = (PV_{i+1}X_i)\lfloor Z_{L(0)}\rfloor_{N_i*d}^\top \left[B_{L(0)}^{-1} \cdot \frac{\partial\overline{\mu}(L(0))}{\partial L_j(0)}\right]_{N_i*d}$$

for $j = 1, \ldots, M-1$. Thus, our LRM algorithm gets a new Step 2:

2. Along each path use the LRM of Section 2.3 to calculate the Delta vectors of all payments from the optimal exercise time to maturity and add up these Delta vectors.

## 4 Numerical results

We will illustrate the performance of the different algorithms by some numerical examples. For this, we compare the FDM, the FM, the AM, its new version NAM, PFD, and the LRM.

In our first example we consider a $(2 \times 20)$-receiver-Bermudan swaption and a corresponding $(2 \times 20)$-payer-Bermudan swaption with identical parameters and assumptions:

- Tenor structure $\{T_0, \ldots, T_{20}\}$.
- $\delta_i = T_{i+1} - T_i = 0.25$ (year) for all $i = 0, \ldots, 19$.
- Implementation of the forward LIBORs on $\{t_0 \ldots, t_{19}\} \equiv \{T_0, \ldots, T_{19}\}$.
- $h_n = t_{n+1} - t_n = 0.25$ (year) for all $n = 0, \ldots, 18$.
- Face value $\mathcal{N} = 10{,}000$€.
- Fixed base rate of $R = 4.5\%$.
- Flat initial LIBOR curve of $L_i(0) = 5\%$ for all $i = 0, \ldots, 19$.
- One-dimensional constant volatility functions $\sigma_1(t) = \ldots = \sigma_{19}(t) = 20\%$.

The LSM algorithm with 65,536 paths and antithetic variates yields

| | Price (€) | Standard error |
|---|---|---|
| (2×20)-receiver-Bermudan swaption | 115.94 | 0.247839 |
| (2×20)-payer-Bermudan swaption | 290.56 | 0.394865 |

For calculating the Delta vector of both Bermudan swaptions we use exactly the same paths for our six suggested methods. For FDM we use *central differences* with *mesh size* $\epsilon = 0.0000001$. Again, we use *antithetic variates* as variance reduction methods. The resulting numerical results are summarized in Figures 1, 2 and Tables 1, 2 of Appendix A.

Figure 1 and Table 1 show the simulated Delta vectors of the receiver-Bermudan swaption obtained by the six different algorithms. Figure 2 and Table 2 show the same data for the payer-Bermudan swaption. The results in both figures are quoted in changes per basis point bp. Thus, we have chosen $bp^{-1}$ as the basic unit in both figures and both tables.

As FM, AM, and NAM are all implementations of the same PM, their results agree. They only differ in the direction of the recursion, not in the results. Further, the results of FDM and PFD are both fully acceptable although the methods are biased. Only the results of LRM differ a lot from the remaining ones and exhibit a sawtooth structure. This is due to the fact that in addition to the model bias, LRM also tends to amplify the variance of the simulated input.

To judge the efficiency of the algorithms we look at the second example, the $(2 \times M)$-receiver-Bermudan swaptions for $M = 4, 8, 12, 16, 20, 24$. We again apply all six methods with the same 65,536 simulated paths. Figure 3 compares the *relative computing times*[1] of the algorithms to determine the Delta vectors of the $(2 \times M)$-receiver-Bermudan swaptions with $M = 4, 8, 12, 16, 20, 24$. Clearly, FDM is the slowest method with a near linearly increasing time line. Further, FM is more efficient than FDM, but needs more time than the remaining four methods. To explore the differences in efficiency of these methods, we consider Figure 4, which is a part of Figure 3 but on a different scale. Figure 4 admits that the time lines of all four methods also have a nearly linear form. However, we can see that the two adjoint methods perform best, with NAM slightly outperforming AM. Although slightly slower, the relative time consumptions of LRM and PFD are of the same order as those of NAM and AM. We further note that FDM sometimes causes overflow and thus lacks stability, which is not the case for the other five methods.

We summarize our (subjective) ranking of the six methods with regard to different characteristics in Table 3.

Thus, the NAM is the method of choice for computing the Delta vector of a Bermudan swaption. Due to its simplicity, efficiency, and acceptable accuracy, PFD is also a good choice although biased. LRM performs worst overall. However, in the case of a non-smooth payoff it might be the only applicable method. Then, one should take great care in enhancing it with a good variance reduction method.

Also, our methods can be applied for the calculation of the Delta vectors of other callable LIBOR exotics such as *callable capped floaters* and *callable inverse floaters*. They only differ from the Bermudan swaptions by the coupons $X_i$. In the case of non-Lipschitz continuous coupons $X_i$ there remains LRM as an admissible method.

There are further modern approaches of research in this area. One is the use of the predictor–corrector method of Denson & Joshi (2011) who derive the AM for the predictor–corrector drift approximation in the displaced-diffusion LMM to improve its accuracy. Another aspect is the development of the algorithmic differentiation of Capriotti & Giles (2011). It can be used as a design paradigm to implement the AM for Greeks in full generality and with minimal analytical effort.

>

## Acknowledgments
Both authors thank the State Research Center (CM)² of TU Kaiserslautern for financial support.

**Ralf Korn** is Full Professor for Financial Mathematics at the University of Kaiserslautern, Germany. He received his chair in 1999 after his PhD in 1993 and his Habilitation in 1997 at the University of Mainz, Germany. He has been an academic visitor to Imperial College London in 1995/96 and has been a visiting professor at the University of Strasbourg in 2004 and 2006. His main research areas are financial mathematics (in particular portfolio optimization, derivative pricing, computational) and stochastic control (in particular impulse control and worst-case control). He has published five books (the most recent one, *Monte Carlo Methods and Models in Financeand Insurance*, with his wife Elke Korn and Gerald Kroisandt) and over 60 refereed journal articles. He is the head of the Financial Mathematics Group of Fraunhofer ITWM, deputy head of the German Society for Actuarial and Financial Mathematics, the managing director of the European Institute for Quality Management of Products and Methods in Finance, and editor/associate editor of various bookseries and journals.

**Qian Liang**, who originates from Dalian, China, obtained his Diploma degree in mathematics from the University of Augsburg, Germany. He received a PhD in mathematics at the University of Kaiserslautern (Germany) under the supervision of Professor Ralf Korn. He was a member of the State Research Center (CM)², the Center for Mathematical and Computational Modeling at the University of Kaiserslautern. Currently, he works as a consultant for the financial industry. His main research areas are computational finance, derivatives pricing, Monte Carlo simulation, price sensitivities of LIBOR derivatives, and functional programming in financial engineering. His research articles have been published and accepted for publication in *Risk Magazine*, *AsiaRisk*, *Wilmott*, and the *Journal of Computational Finance*.

## ENDNOTE
1. Relative computing time $= \dfrac{\text{Time to compute the Delta vector of the Bermudan swaption}}{\text{Time to compute the price of the Bermudan swaption}}$

## REFERENCES
Brace A., Gatarek D., & Musiela M. (1997). The market model of interest rate dynamics. *Mathematical Finance* **7**, 127–155.
Capriotti L. & Giles M. (2011). Algorithmic Differentiation: Adjoint Greeks Made Easy. SSRN.
Denson N. & Joshi M. (2011). Fast and accurate Greeks for the LIBOR market model. *Journal of Computational Finance* **14**, 115–140.
Giles M. & Glasserman P. (2006). Smoking adjoints: Fast Monte Carlo Greeks. *Risk* January, 88–92.
Glasserman P. (2004). Monte Carlo Methods in Financial Engineering. Springer Science.
Glasserman P. & Zhao X. (1999). Fast Greeks by simulation in forward Libor models. *Journal of Computational Finance* **3**, 5–39.
Korn R., Korn E., & Kroisandt G. (2010). Monte Carlo Methods and Models in Finance and Insurance. Chapman&Hall/CRC Financial Mathematics Series.
Jamshidian F. (1997). Libor and swap market models and measures. *Finance and Stochastics* **1**, 293–330.
Leclerc M., Liang Q., & Schneider I. (2009). Fast Monte Carlo Bermudan Greeks. *Risk* July, 84–88.
Longstaff F. A. & Schwartz E. S. (2001). Valuing American options by simulation: A simple least-squares approach. *Review of Financial Studies* **14**, 113–147.
Miltersen K., Sandmann K., & Sondermann D. (1997). Closed form solutions for term structure derivatives with lognormal interest rate. *Journal of Finance* **52**, 409–430.
Piterbarg V. V. (2003). Computing Deltas of callable Libor exotics in forward Libor models. *Journal of Computational Finance* **7**(3), 107–144.

## Appendix A. Numerical results: Graphs and tables

**Figure 1: Delta vectors of a (2 × 20)-receiver-Bermudan swaption.**

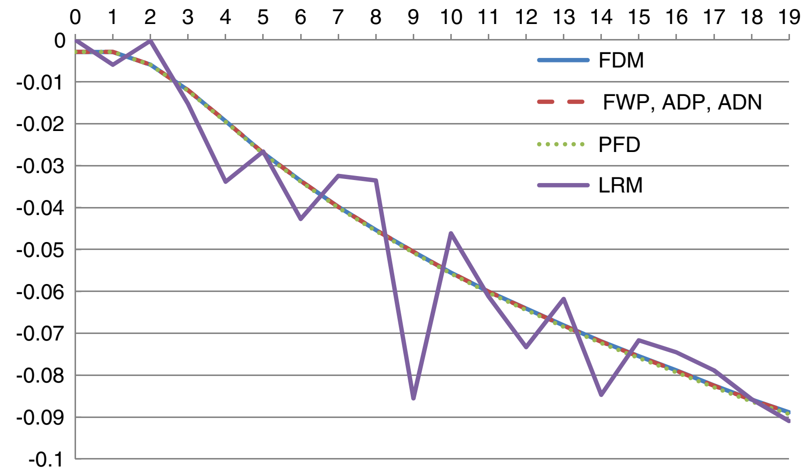


**Figure 2: Delta vectors of a (2 × 20)-payer-Bermudan swaption.**

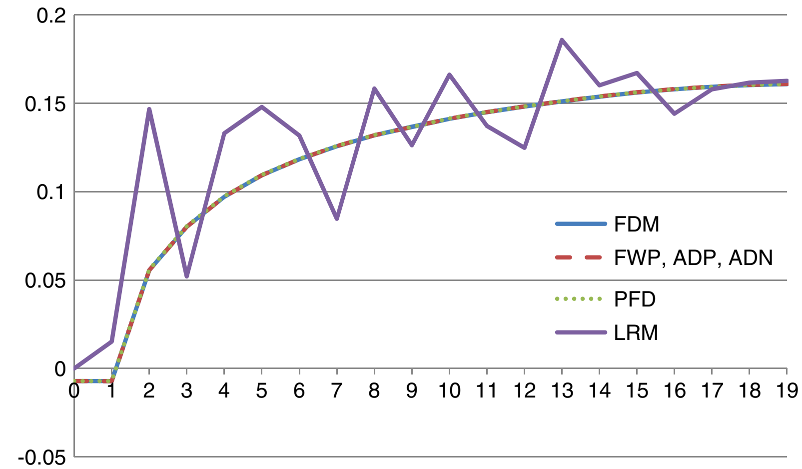

**Table 1: Delta vectors and their standard errors of a (2×20)-receiver-Bermudan swaption.**

| | FDM | | FM | | AM | | NAM | | PFD | | LRM | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Delta | s.e. | Delta | s.e. | Delta | s.e. | Delta | s.e. | Delta | s.e. | Delta | s.e. |
| 0 | −0.002867339 | 6.13724E−06 | −0.002858751 | 1.03144E−05 | −0.002858751 | 1.03144E−05 | −0.002858751 | 1.03144E−05 | −0.002845100 | 1.88939E−05 | 0.000000000 | 0.000000000 |
| 1 | −0.002864684 | 5.86818E−06 | −0.002856588 | 9.60628E−06 | −0.002856588 | 9.60628E−06 | −0.002856588 | 9.60628E−06 | −0.002843342 | 9.58990E−06 | −0.005935054 | 0.018021070 |
| 2 | −0.005893481 | 6.21278E−05 | −0.005885062 | 6.25840E−05 | −0.005885062 | 6.25840E−05 | −0.005885062 | 6.25840E−05 | −0.005937581 | 6.32215E−05 | −0.000198584 | 0.018300536 |
| 3 | −0.012002579 | 0.000101890 | −0.011993966 | 0.000102379 | −0.011993966 | 0.000102379 | −0.011993966 | 0.000102379 | −0.012127034 | 0.000102961 | −0.015206396 | 0.018270017 |
| 4 | −0.019382898 | 0.000129753 | −0.019382990 | 0.000130141 | −0.019382990 | 0.000130141 | −0.019382990 | 0.000130141 | −0.019532683 | 0.000130825 | −0.033845474 | 0.018131252 |
| 5 | −0.026974042 | 0.000147151 | −0.026973923 | 0.000148157 | −0.026973923 | 0.000148157 | −0.026973923 | 0.000148157 | −0.027158483 | 0.000148691 | −0.026635692 | 0.017702780 |
| 6 | −0.033614661 | 0.000155993 | −0.033623010 | 0.000157716 | −0.033623010 | 0.000157716 | −0.033623010 | 0.000157716 | −0.033811737 | 0.000158219 | −0.042735089 | 0.016968676 |
| 7 | −0.039804836 | 0.000159804 | −0.039832241 | 0.000162505 | −0.039832241 | 0.000162505 | −0.039832241 | 0.000162505 | −0.040034543 | 0.000162926 | −0.032434587 | 0.016367357 |
| 8 | −0.045357039 | 0.000159329 | −0.045370325 | 0.000163315 | −0.045370325 | 0.000163315 | −0.045370325 | 0.000163315 | −0.045584560 | 0.000163728 | −0.033514623 | 0.015765374 |
| 9 | −0.050549540 | 0.000156082 | −0.050552558 | 0.000162117 | −0.050552558 | 0.000162117 | −0.050552558 | 0.000162117 | −0.050788777 | 0.000162498 | −0.085522139 | 0.014775106 |
| 10 | −0.055533812 | 0.000150664 | −0.055538122 | 0.000159612 | −0.055538122 | 0.000159612 | −0.055538122 | 0.000159612 | −0.055767348 | 0.000160079 | −0.046139803 | 0.013800554 |
| 11 | −0.060036824 | 0.000144054 | −0.060071558 | 0.000156487 | −0.060071558 | 0.000156487 | −0.060071558 | 0.000156487 | −0.060328865 | 0.000156911 | −0.061196232 | 0.012811454 |
| 12 | −0.064127859 | 0.000136349 | −0.064169750 | 0.000153464 | −0.064169750 | 0.000153464 | −0.064169750 | 0.000153464 | −0.064453998 | 0.000153964 | −0.073315048 | 0.011548184 |
| 13 | −0.068123998 | 0.000128504 | −0.068176473 | 0.000151455 | −0.068176473 | 0.000151455 | −0.068176473 | 0.000151455 | −0.068462800 | 0.000151991 | −0.061808516 | 0.010288813 |
| 14 | −0.071948465 | 0.000120598 | −0.072019161 | 0.000151013 | −0.072019161 | 0.000151013 | −0.072019161 | 0.000151013 | −0.072338954 | 0.000151485 | −0.084687689 | 0.009076843 |
| 15 | −0.075455476 | 0.000114308 | −0.075545964 | 0.000152966 | −0.075545964 | 0.000152966 | −0.075545964 | 0.000152966 | −0.075884967 | 0.000153497 | −0.071653397 | 0.007691378 |
| 16 | −0.078842916 | 0.000109585 | −0.078949569 | 0.000158586 | −0.078949569 | 0.000158586 | −0.078949569 | 0.000158586 | −0.079323517 | 0.000159198 | −0.074533701 | 0.006163343 |
| 17 | −0.082454557 | 0.000108978 | −0.082525555 | 0.000165998 | −0.082525555 | 0.000165998 | −0.082525555 | 0.000165998 | −0.082896734 | 0.000166807 | −0.078850143 | 0.004629604 |
| 18 | −0.085816964 | 0.000113176 | −0.085878137 | 0.000177025 | −0.085878137 | 0.000177025 | −0.085878137 | 0.000177025 | −0.086270063 | 0.000177882 | −0.085714103 | 0.002960613 |
| 19 | −0.088852340 | 0.000120653 | −0.088890053 | 0.000190088 | −0.088890053 | 0.000190088 | −0.088890053 | 0.000190088 | −0.089277677 | 0.000190824 | −0.090946097 | 0.001367952 |

**Table 2: Delta vectors and their standard errors of a (2 × 20)-payer-Bermudan swaption.**

| | FDM | | FM | | AM | | NAM | | PFD | | LRM | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Delta | s.e. | Delta | s.e. | Delta | s.e. | Delta | s.e. | Delta | s.e. | Delta | s.e. |
| 0 | −0.007152301 | 9.74833E−06 | −0.007177337 | 1.90515E−05 | −0.007177337 | 1.90515E−05 | −0.007177337 | 1.90515E−05 | −0.007147695 | 1.88939E−05 | 0.000000000 | 0.000000000 |
| 1 | −0.007176196 | 1.06960E−05 | −0.007205000 | 2.11945E−05 | −0.007205000 | 2.11945E−05 | −0.007205000 | 2.11945E−05 | −0.007174941 | 2.10235E−05 | 0.015182315 | 0.040028854 |
| 2 | 0.055603182 | 0.000273631 | 0.055569602 | 0.000274362 | 0.055569602 | 0.000274362 | 0.055569602 | 0.000274362 | 0.055952172 | 0.000274385 | 0.146747596 | 0.041012492 |
| 3 | 0.080166003 | 0.000263395 | 0.080155527 | 0.000264430 | 0.080155527 | 0.000264430 | 0.080155527 | 0.000264430 | 0.080451181 | 0.000264264 | 0.052049006 | 0.039912803 |
| 4 | 0.097124968 | 0.000241016 | 0.097140770 | 0.000242949 | 0.097140770 | 0.000242949 | 0.097140770 | 0.000242949 | 0.097377910 | 0.000242538 | 0.133057634 | 0.038741314 |
| 5 | 0.109337672 | 0.000224190 | 0.109351948 | 0.000222908 | 0.109351948 | 0.000222908 | 0.109351948 | 0.000222908 | 0.109582314 | 0.000222300 | 0.147989635 | 0.037102145 |
| 6 | 0.118357568 | 0.000209116 | 0.118372877 | 0.000205513 | 0.118372877 | 0.000205513 | 0.118372877 | 0.000205513 | 0.118575584 | 0.000204818 | 0.131820124 | 0.035698762 |
| 7 | 0.125752974 | 0.000200860 | 0.125773033 | 0.000191659 | 0.125773033 | 0.000191659 | 0.125773033 | 0.000191659 | 0.125950346 | 0.000190982 | 0.084690785 | 0.033788223 |
| 8 | 0.131895393 | 0.000198719 | 0.131970697 | 0.000183277 | 0.131970697 | 0.000183277 | 0.131970697 | 0.000183277 | 0.132091939 | 0.000182858 | 0.158415414 | 0.032100094 |
| 9 | 0.136623545 | 0.000198391 | 0.136643424 | 0.000176084 | 0.136643424 | 0.000176084 | 0.136643424 | 0.000176084 | 0.136736047 | 0.000175812 | 0.126273045 | 0.029802513 |
| 10 | 0.141302444 | 0.000200274 | 0.141384417 | 0.000170837 | 0.141384417 | 0.000170837 | 0.141384417 | 0.000170837 | 0.141439956 | 0.000170724 | 0.166238117 | 0.027641228 |
| 11 | 0.144967034 | 0.000200254 | 0.145045557 | 0.000165818 | 0.145045557 | 0.000165818 | 0.145045557 | 0.000165818 | 0.145091437 | 0.000165889 | 0.137213495 | 0.025468640 |
| 12 | 0.148215142 | 0.000203696 | 0.148302616 | 0.000161986 | 0.148302616 | 0.000161986 | 0.148302616 | 0.000161986 | 0.148280668 | 0.000162080 | 0.124847053 | 0.022984943 |
| 13 | 0.151092045 | 0.000207017 | 0.151219765 | 0.000159131 | 0.151219765 | 0.000159131 | 0.151219765 | 0.000159131 | 0.151163069 | 00.000159375 | 0.185857877 | 0.020432957 |
| 14 | 0.153765896 | 0.000209253 | 0.153918505 | 0.000156877 | 0.153918505 | 0.000156877 | 0.153918505 | 0.000156877 | 0.153845146 | 0.000157054 | 0.160228664 | 0.017538324 |
| 15 | 0.156114417 | 0.000211096 | 0.156286432 | 0.000154484 | 0.156286432 | 0.000154484 | 0.156286432 | 0.000154484 | 0.156180228 | 0.000154624 | 0.167166947 | 0.014691748 |
| 16 | 0.157961107 | 0.000211390 | 0.158104714 | 0.000151979 | 0.158104714 | 0.000151979 | 0.158104714 | 0.000151979 | 0.157990045 | 0.000152027 | 0.144156519 | 0.011627737 |
| 17 | 0.159301834 | 0.000210119 | 0.159448280 | 0.000149410 | 0.159448280 | 0.000149410 | 0.159448280 | 0.000149410 | 0.159335269 | 0.000149396 | 0.157983464 | 0.008699451 |
| 18 | 0.160334295 | 0.000208473 | 0.160473051 | 0.000146588 | 0.160473051 | 0.000146588 | 0.160473051 | 0.000146588 | 0.160358464 | 0.000146506 | 0.161742816 | 0.005540337 |
| 19 | 0.160890021 | 0.000206158 | 0.161040428 | 0.000144388 | 0.161040428 | 0.000144388 | 0.161040428 | 0.000144388 | 0.160930931 | 0.000144201 | 0.162747654 | 0.002529270 |

**Figure 3: Relative computing time of the six different algorithms.**

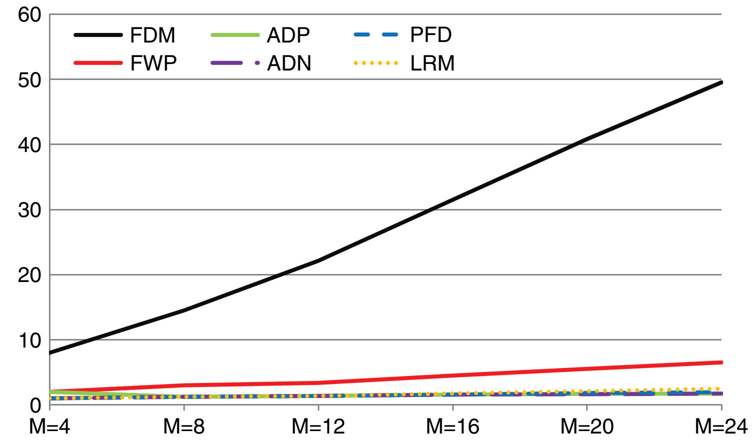


**Figure 4: Relative computing time of the four efficient algorithms.**

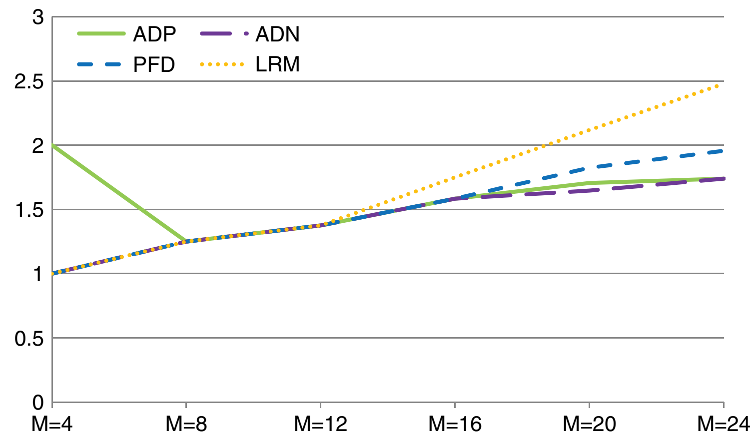


**Table 3: Summary of different characteristics of the six methods.**

| | | | | |
|---|---|---|---|---|
| | | Accuracy | | |
| high ← | | | | →low |
| FM = AM = NAM | | | FDM ≈ PFD | LRM |
| | | Speed | | |
| fast ← | | | | → slow |
| AM ≈ NAM ≈ PFD ≈ LRM | | | FM | FDM |
| | | Stability | | |
| stable ← | | | | → unstable |
| FM ≈ AM ≈ NAM ≈ PFD ≈ LRM | | | | FDM |
| | | Implementation | | |
| easy ← | | | | → difficult |
| FDM | PFD | FM | NAM | AM ≈ LRM |