Rapid Computation of Drifts in a Reduced Factor LIBOR Market Model

Mark S. Joshi

The LIBOR market model for the pricing of exotic-interest-rate derivatives has become very popular in recent years. It is the most sophisticated and complicated model for pricing interest-rate derivatives which is in widespread use. Whilst the theory underlying the model is not particularly hard, implementing and calibrating the model in an efficient manner is tricky. A number of ad hoc techniques and approximations have been developed to do both. See Brace, Gatarek and Musiela (1997), Hunter, Jäckel and Joshi (2001), Jamishidian (1997), Jäckel (2001) and Rebonato (2002). The purpose of this article is to discuss one aspect of implementing a BGM model: how to efficiently compute the forward-rate drifts.

The basis of the model is that forward interest rates are taken to be the state variables, and are assumed to be log-normal. (or some similar process in extensions.) We fix notation. We are evolving a set of forward rates associated to a set of times $0 < t_0 < t_1 < t_2 < \cdots < t_n$. We let $f_j(s)$ denote the forward rate from time t_j to time t_{j+1}, as observed at time s. We note the zero-coupon bond maturing at t_j by $P_j(s)$. The volatility of the rate f_j is denoted by $\sigma_j(s)$.

Martingale pricing is based on picking a measure in which discounted asset price processes are martingales. Discounting is done by picking a fixed zero-coupon bond, P_N, as numeraire. We therefore have that for any asset A, that A/P_N is a martingale and therefore has zero drift. However, forward rates are not assets, nor necessarily the ratio of an asset to the numeraire. This means that their drifts can be, and generally are, non-zero.

The drifts are not just non-zero but state-dependent, and are given by quite complicated expresssions. As these computations are state-dependent, these must be done at every step of a Monte Carlo simulation, and they can easily become a computational bottleneck.

We suppose that

$$\frac{df_i}{f_i} = \mu_i dt + \sigma_i(s)dt, \tag{1}$$

where μ_i is possibly a function of time and the rates.

If we take the martingale measure associated to the numeraire P_N, then we have the following drifts expressions,

$$\mu_i = \begin{cases} -\sigma_i \sum\limits_{k=i+1}^{N-1} \frac{f_k(s)\tau_k}{1+f_k(s)\tau_k}\sigma_k\rho_{ik} & \text{for } i < N-1, \\ 0, & \text{for } i = N-1, \\ \sigma_i \sum\limits_{k=N}^{i} \dfrac{f_k(t)\tau_k}{1+f_k(t)\tau_k}\sigma_k\rho_{ik}. & \text{for } i > N. \end{cases} \tag{2}$$

In what follows for simplicity we assume that $N = n$. However, our approach is not restricted to that case. When evolving the rates from time s to time t, we perform a deterministic integral across the time step to obtain the covariance matrix, C, of the logs of the rates, and set

$$C_{ik}(s,t) = \int_s^t \rho_{ir}\sigma_i(r)\sigma_k(r)dr. \tag{3}$$

This yields an approximation to the total drift across the step

$$\mu_i(t-s) \sim -\sum_{k=i+1}^{n-1} \frac{f_k(s)\tau_k}{1+f_k(s)\tau_k} C_{ik}. \tag{4}$$

Note that if $i = n-1$, the sum is empty, and we take the value to be zero. Note that for the calibration of the model, it is only the terms C_{ik} for each time step that actually matter.

As the terms in this summation depend on i, we need to compute $n-i$ terms for each value of i, and we need to carry out approximately $n^2/2$, multiplications. However, for efficiency reduced factor models are often used in which the matrix C is of rank 2 or 3 instead of n. If we use the above algorithm we have no gain from this factor reduction, and it will be the only part of our simulation of order n^2 for each step. As a simulation typically has n steps, this means that the drift computation is growing at rate n^3, and will rapidly become the main drain on computing time. For example if n is 60 then we have of order 216,000 computations for the drifts, versus 3600 for the rest of the simulation.

We show here that the drift computation for one step can be performed using a computation of order nF where F is the number of factors. The key to our approach is to work with the pseudo-square root of the covariance matrix rather than the covariance matrix itself.

Thus first we find an $n \times k$ matrix A such that

$$C = AA^t. \tag{5}$$

(Such a matrix exists if and only if C is of rank k.) As the covariance matrices are fixed this computation need only be carried out once for each step; in any case the matrix A will be needed for evolving the Brownian part of the simulation so no additional work is required.

By definition, we have

$$C_{ik} = \sum_{r=1}^{F} A_{ir} A^t_{rk}. \tag{6}$$

Substituting into (4), we obtain

$$-\sum_{k=i+1}^{n-1} \sum_{r=1}^{F} \frac{f_k(s)\tau_k}{1+f_k(s)\tau_k} A_{ir} A^t_{rk} = -\sum_{r=1}^{F} A_{ir} \sum_{k=i+1}^{n-1} \frac{f_k(s)\tau_k}{1+f_k(s)\tau_k} A^t_{rk}. \tag{7}$$

What does this buy us? The terms in the inner sum do not involve i and we can therefore reuse them. In particular, if we define

$$e_{r,i} = \sum_{k=i+1}^{n-1} \frac{f_k(s)\tau_k}{1+f_k(s)\tau_k} A^t_{rk}, \tag{8}$$

we have that

$$e_{r,i} = e_{r,i+1} + \frac{f_{i+1}(s)\tau_{i+1}}{1+f_{i+1}(s)\tau_{i+1}} A^t_{r,i+1}. \tag{9}$$

If we have precomputed the terms

$$\frac{f_{i+1}(s)\tau_{i+1}}{1+f_{i+1}(s)\tau_{i+1}}$$

then all the terms $e_{r,i}$ can be computed using only nF multiplications.

We still need to compute

$$\sum_{r=1}^{F} A_{ir} e_{r,i},$$

for each value of i which results in another nF multiplications. Our computation is therefore of order nF.

In fact, by rethinking our order of computation slightly we can eliminate the second set of multiplications. Given a vector of independent normals $Z = (Z_j)$, we have to turn them into correlated variates by using the pseudo-square root. We can therefore absorb the two pseudo-root multiplications into one and obtain

$$\log f_i(t) = \log f_i(s) - 0.5 C_{ii}(s,t) + \sum_{r=1}^{F} A_{ir}(e_{r,i} + Z_r). \tag{10}$$

We have shown that by a judicious reordering, the amount of time spent on drift calculations can be reduced to a computational time of order nF. Whilst this idea is not deep it results in considerable time savings when n is large and F is small, and the approach does not seem to be as well known as one might expect.

REFERENCES

■ A. Brace, D. Gatarek, M. Musiela, The market model of interest-rate dynamics, *Mathematical Finance* 7, 127–155 (1997).
■ D. Brigo, F. Mercurio, Interest Rate Models: Theory and Practice, Springer Finance 2001.
■ C. Hunter, P. Jäckel, M. Joshi, Getting the drift, Risk July 2001.
■ F. Jamishidian, LIBOR and swap market models and measures, Finance and stochastics, 1, 293–330 (1997).
■ P. Jäckel, Monte Carlo Methods in Finance, Wiley 2001.
■ R. Rebonato, The modern pricing of interest rate derivatives, PUP 2002.

BIBLIOGRAPHY

Mark Johsi is Head of Model Evaluation in Royal Bank of Scotland Group Risk Management. His books "the Concepts of Mathematical Finance" and "C++ Design Patterns and Derivatives Pricing" will be published by Cambridge University Press in Summer 2003.