

# A Note on Implementing LIBOR Market Model

September 14, 2009

The Numerical Algorithms Group Ltd  
Wilkinson House  
Jordan Hill  
Oxford  
OX2 8DR  
UK

Web Site: [www.nag.co.uk](http://www.nag.co.uk)  
General Inquires: [info@nag.co.uk](mailto:info@nag.co.uk)  
Technical Support: [support@nag.co.uk](mailto:support@nag.co.uk)

# 1 Introduction

This note describes the implementation of a standard LIBOR market model. It starts with a brief review of the model in Section 2. We assume that readers are familiar with the fundamental concepts of LIBOR rates and instruments. Section 3 outlines the discretization for Monte Carlo simulation. Section 4 introduces Monte Carlo applications in vanilla instruments. Section 5 gives numerical examples using NAG Normal random number generator (g051ac) [4]. Section 6 explains the implementation on GPU hardware.

## 2 LIBOR Market Model

LIBOR market model uses forward LIBOR rates as fundamental assets. Let  $L_t^i$  denote the forward LIBOR rate over the time interval  $[T_i, T_{i+1}]$ , where  $T_i, i = 0, 1, \dots, N$ , are LIBOR reset dates. Each forward LIBOR rate has the following dynamics.

$$dL_t^i = L_t^i(\mu_t^i dt + \sigma_t^i dW_t^i) \quad (1)$$

where  $W_t^i$  is the Brownian motion for  $L_t^i$  and forward LIBOR rates are allowed to have factor correlations,  $dW_t^i dW_t^j = \rho_t^{ij}, \forall i, j$ . The drift term  $\mu_t^i$  depends on the probability measure under which  $L_t^i$  is modelled. Here we summarize two standard probability measures, namely, terminal measure and spot-LIBOR measure in Table 1. In our implementation we use Spot-LIBOR measure.

Measure, $\mathbb{Q}$	Numeraire, $N_t$	Drift, $\mu_t^i$
<b>Terminal</b>	$P_t^N = \frac{P_t^{\gamma(t)}}{\prod_{i=\gamma(t)}^{N-1} (1 + \tau_i L_t^i)}$	$-\sum_{j=i+1}^{N-1} \frac{\tau_j L_t^j}{1 + \tau_j L_t^j} \rho_t^{ij} \sigma_t^i \sigma_t^j$
<b>Spot-LIBOR</b>	$B_t = P_t^{\gamma(t)} \prod_{i=0}^{\gamma(t)-1} (1 + \tau_i L_{T_i}^i)$	$\sum_{j=\gamma(t)}^i \frac{\tau_j L_t^j}{1 + \tau_j L_t^j} \rho_t^{ij} \sigma_t^i \sigma_t^j$

Table 1: Summary of Probability Measures

where  $\tau_i = T_{i+1} - T_i$  and  $\gamma(t) = i$  if  $T_{i-1} \leq t < T_i, i = 1, 2, \dots, N$ .  $P_t^i$  is the time  $t$  value of zero-coupon bond maturing at  $T_i$ .  $B_t$  is the rolling over bank-account. The rolling over bank-account at time  $t$  is defined as follows.

1. Investing £1 at time  $T_0$ .
2. Receiving and reinvesting interest and principal at each reset date until  $T_{\gamma(t)-1}$ .
3.  $B_t$  is the time  $t$  value of the receiving at  $T_{\gamma(t)}$ .

It is hard in general to discretize Equation (1) analytically since  $\mu_t^i$  depends on other forward LIBOR rates (except for  $\mu_t^{N-1}$  in the terminal measure). This means there is no explicit solution for  $L_t^i$ . One needs to approximate  $\mu_t^i$  in certain ways for Monte Carlo simulation. Section 3 describes an example of simulating  $L_t^i$  in Spot-LIBOR measure.

### 3 Simulating LIBOR Rates

We first re-express Equation (1) in exponential form.

$$L_{t_{n+1}}^i = L_{t_n}^i \exp \left[ \int_{t_n}^{t_{n+1}} \mu_s^i ds - \frac{1}{2} \int_{t_n}^{t_{n+1}} (\sigma_s^i)^2 ds + \int_{t_n}^{t_{n+1}} \sigma_s^i dW_s^i \right] \quad (2)$$

where  $t_n, n = 0, 1, \dots, M$ , are discretized points which must satisfy  $t_n \leq T_i$ . Since  $\mu_t^i$  depends on other forward LIBOR rates, the first integral in Equation (2) needs to be approximated in certain ways. The common practice is to fix  $\mu_s^i, s \in [t_n, t_{n+1}]$ , at  $\mu_{t_n}^i$ . The second integral is deterministic and the third one is Normally distributed with mean zero and variance equals to the value of the second integral. Equation (2) can therefore be simulated as

$$\tilde{L}_{t_{n+1}}^i = \tilde{L}_{t_n}^i \exp \left( \mu_{t_n}^i \Delta t_n - \frac{1}{2} I + \sqrt{I} Z_n^i \right) \quad (3)$$

where  $\Delta t_n = t_{n+1} - t_n$  and  $I = \int_{t_n}^{t_{n+1}} (\sigma_s^i)^2 ds$ .  $Z_n^i$  is a standard Normal random variable for  $L_t^i$  at the  $n^{\text{th}}$  time step.

In the example code we assume constant volatilities,  $\sigma_t^i \equiv \sigma^i, \forall i$ , and discretize at LIBOR reset dates only. This means for  $L_t^i$  we simulate at  $T_j, j = 0, 1, \dots, i$ , with starting point,  $T_0$ . We also assume all LIBOR rates share the same Brownian factor  $W_t$  and evolve under Spot-LIBOR measure. The discretized version of forward LIBOR rate then reads

$$\tilde{L}_{T_{j+1}}^i = \tilde{L}_{T_j}^i \exp \left[ \sigma^i \left( \sum_{k=j+1}^i \frac{\tau_k L_{T_j}^k \sigma^k}{1 + \tau_k L_{T_j}^k} - \frac{1}{2} \sigma^i \right) \tau_j + \sigma^i \sqrt{\tau_j} Z_j^i \right], \quad i > j \quad (4)$$

With Equation (4) we are able to simulate forward LIBOR rates at particular time points and price options whose payoffs depend on LIBOR rates.

### 4 Options on LIBOR Rates

We use two vanilla instruments, caplet and swaption, in our example. Caplet is a European call option on LIBOR rates. Swaption is an option to enter into a swap contract. We will not discuss the replication techniques but only look at their payoffs at maturity,  $T_m$ .

$$m^{\text{th}} \text{ Caplet Payoff: } {}^1 V_{T_m} = \frac{(L_{T_m}^m - X)_+}{1 + \tau_m L_{T_m}^m}$$

$$m \times l \text{ Swaption Payoff: } V_{T_m} = Y_{T_m}^{m, m+l} (S_{T_m}^{m, m+l} - X)_+$$

where  $X$  is the strike,  $Y_t^{a,b} = \sum_{i=a+1}^b \tau_i P_t^i$  and  $S_t^{a,b} = \frac{P_t^a - P_t^b}{\sum_{i=a+1}^b \tau_i P_t^i}$ . Using the formula for any contingent claim under no-arbitrage assumption, we have

$$V_{T_0} = N_{T_0} \mathbb{E} \left( \frac{V_{T_m}}{N_{T_m}} \right) \quad (5)$$

---

<sup>1</sup>This is the discounted payoff at  $T_m$ . The actual payoff  $(L_{T_m}^m - X)_+$  takes place at  $T_{m+1}$ .

where  $V_t$  is the value of caplet or swaption. Recall that  $N_t$  is the numeraire corresponding to a particular probability measure. Under the Spot-LIBOR measure, we have

$$N_{T_0} = 1, \quad N_{T_m} = \prod_{i=0}^{m-1} (1 + \tau_i L_{T_i}^i) \quad (6)$$

The expectation is approximated by averaging simulated numeraire based payoffs. The Monte Carlo approximation formula finally reads

$$V_{T_0} \approx \frac{1}{M} \sum_{i=1}^M \frac{\tilde{V}_{T_m}^{(i)}}{\tilde{N}_{T_m}^{(i)}} \quad (7)$$

where  $M$  is the number of sample paths and  $\tilde{V}_{T_m}^{(i)}$  and  $\tilde{N}_{T_m}^{(i)}$  denote the  $i^{th}$  simulated payoff and numeraire values.

The next section provides some numerical examples.

## 5 Numerical Example

In this section, we provide two examples, caplet and a portfolio of swaptions. Since caplet value under the LIBOR market model can be obtained in closed form (Black's formula [1]), we use it as a test case of the NAG Normal random number generator (g051ac) and sample path simulation. We use the swaption portfolio as an example where the explicit form is not available. Table 2 gives the LIBOR market model and Monte Carlo method parameters.

Measure	$N$	$\sigma^i$	$L_{T_0}^i$	$\tau_i$	$M$
Spot-LIBOR	80	0.2, $\forall i$	0.051, $\forall i$	0.25, $\forall i$	1000000

Table 2: Model Specifications

**Caplet.** We price a caplet of 10-year maturity with strike  $X = 0.05$ . Table 3 displays results, choosing different seeds arbitrarily, for the NAG Normal random number generator. SE is the standard error of value from Monte Carlo simulation and Bias is the difference between Monte Carlo value and explicit value. With different seeds all Monte Carlo values are close to the explicit one.

Seed	Value (bps)	SE (bps)	Bias (bps)
1, 2, 3, 4	19.35	(0.03)	-0.04
156, 695, 78, 9	19.31	(0.03)	-0.08
78, 234, 986, 785	19.36	(0.03)	-0.03
5, 34, 65, 758	19.40	(0.03)	0.01
254, 12, 69, 75	19.38	(0.03)	-0.01

Table 3: Caplet Results, Explicit Value 19.39 (bps)

**Portfolio of Swaptions.** We construct an equally weighted portfolio of 15 swaptions, each with the same maturity,  $m = 80$ , but different length,  $l$ . Each swaption has a different strike price. Table 4 summarizes the portfolio components. The simulation results are given in Table

<b>Index</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>
$l =$	4	4	4	8	8	8	20
$X =$	0.045	0.05	0.055	0.045	0.05	0.055	0.045
<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>	<b>13</b>	<b>14</b>	<b>15</b>
20	20	28	28	28	40	40	40
0.05	0.055	0.045	0.05	0.055	0.045	0.05	0.055

Table 4: Portfolio of Swaptions

5. Although we do not have an explicit value for a portfolio of swaptions, we can see these

<b>Seed</b>	<b>Value (bps)</b>	<b>SE (bps)</b>
1, 2, 3, 4	4936.6	(6.9)
156, 695, 78, 9	4929.0	(6.9)
78, 234, 986, 785	4940.3	(6.9)
5, 34, 65, 758	4940.5	(6.9)
254, 12, 69, 75	4944.3	(6.9)

Table 5: Swaption Portfolio Results

Monte Carlo values are reasonably close to each other.

## 6 GPU implementation

The implementation of LIBOR market model on a Graphics Processing Unit (GPU) uses the functions `nag_gpu_mrg32k3a_init` and `nag_gpu_mrg32k3a_normal` [5] to generate the variates from a Normal distribution. These are computed from the pseudorandom number generator MRG32k3a of L’Ecuyer [3] by applying the Box-Muller transformation to the uniform sequence. The results shown in the Table 6 are for the same swaption portfolio specified above. They were generated using an NVIDIA Quadro FX5800 on a system running Fedora 10 Linux with an Intel Dual core 1.87GHz Xeon processor. The time to compute the Portfolio value by the LIBOR model executing on the GPU is compared with the time to run the corresponding case on the CPU.

<b>Seed</b>	<b>Value (bps)</b>	<b>GPU time(msec)</b>	<b>CPU time(msec)</b>
78, 234, 786	4938.8	2879	216360
254, 758, 695			

Table 6: GPU – Swaption Portfolio Results

# Acknowledgements

This note was prepared by Kai Zhang during an internship at NAG Ltd, Oxford and is based on an implementation of the LIBOR model by Prof Mike Giles, Mathematical Institute, University of Oxford [6].

# References

- [1] Black, F., “The Pricing of Commodity Contracts”, *Journal of Financial Economics*, 3 (1976), pp. 167-179.
- [2] Brigo, D., and Mercurio, F., “Interest Rate Models - Theory and Practice with Smile, Inflation and Credit”, 2nd Edition, Springer (2006).
- [3] L’Ecuyer P “Good parameter sets for combined multiple recursive random number generators”, *Operations Research*, 47:1 (1999) pp.159-164.
- [4] “NAG C Library Mark 8”, <http://www.nag.co.uk/numeric/CL/CLdescription.asp>.
- [5] “NAG Numerical routines for GPUs”, <http://www.nag.co.uk/numeric/GPUs/>.
- [6] <http://people.maths.ox.ac.uk/~gilesm/hpc/>