Interest Rate Models

8. LIBOR Market Model, II

Andrew Lesniewski

Baruch College
New York

Spring 2017

Outline

1. Simulating Brownian motion
2. Discretizing SDEs
3. Generating Monte Carlo paths for LMM

Monte Carlo methods for LMM

- LMM does not allow for a natural implementation based on recombining trees, and thus all valuations have to be performed via Monte Carlo simulations.
- We shall describe two numerical schemes for generating Monte Carlo paths for LMM: Euler's scheme and Milstein's scheme.
- They both rely on replacing continuous time stochastic differential equations by suitable finite difference schemes.

One factor Brownian motion

- There exist many more of less refined methods for simulating a Wiener process; here we describe two of them.
- The *random walk method* is easy to implement at the expense of being rather noisy. It represents a Wiener process as a random walk sampled at a finite set of event dates $t_0 < t_1 < \ldots < t_m$:

Random Walk:
W(tn)=W(t1)+{W(t2)-W(t1)}+{W(t3)-W(t2)}+…+{W(tn)-W(t_n1)}
Industry are not using this method!

$$\begin{aligned} Z(t_{-1}) &= 0, \\ Z(t_n) &= Z(t_{n-1}) + \sqrt{t_n - t_{n-1}}\,\xi_n, \quad n = 0, \ldots, m, \end{aligned} \tag{1}$$

where $t_{-1} = 0$, and where ξ_n are i.i.d. random variables with $\xi_n \sim N(0, 1)$.

- A good method of generating the ξ_n's is to first generate a sequence of uniform pseudorandom numbers u_n (using, say, the Mersenne twister algorithm), and then set

$$\xi_n = N^{-1}(u_n), \tag{2}$$

where $N^{-1}(x)$ is the inverse cumulative normal function. $N^{-1}(x)$ can be efficiently and accurately computed using e.g. the Beasley-Springer-Moro algorithm, see [1].

Also can use the packages in Python: e.g., statsmodel

One factor Brownian motion

Very genius and efficient method In HW must use this method!!!!

- The *spectral decomposition method* generally leads to much better performance than the random walk method. It assures that the simulated process has the same covariance matrix $\mathcal{C}$ as the Wiener process $Z(t)$ sampled at $t_0, t_1, \ldots, t_m$.
- The covariance matrix is explicitly given by:

$$\begin{aligned} \mathcal{C}_{ij} &= \mathrm{E}\left[Z(t_i)Z(t_j)\right] \\ &= \min(t_i, t_j). \end{aligned} \tag{3}$$

- Consider the eigenvalue problem for $\mathcal{C}$:

Assume time interval ti are equally spaced for computing eigenvalue.
Even if the time intervals are not equally spaced, we can also compute evalues in Python

$$\mathcal{C}E_j = \lambda_j E_j, \qquad j = 0, \ldots, m, \tag{4}$$

with orthonormal E_j's. Orthonormal: 标准正交化

- Since the covariance matrix $\mathcal{C}$ is positive definite, all of its eigenvalues λ_j are nonnegative, and we will assume that

$$\lambda_0 \geq \ldots \geq \lambda_m \geq 0. \tag{5}$$

One factor Brownian motion

- We will denote the n-th component of the vector E_j by $E_j(t_n)$, and consider the random variable

$$Z(t_n) = \sum_{0 \le j \le m} \sqrt{\lambda_j}\, E_j(t_n) \xi_j, \tag{6}$$

  where ξ_j are, again, i.i.d. random variables with $\xi_j \sim N(0,\, 1)$.
- These numbers are best calculated by applying the inverse cumulative normal function to a sequence of Sobol numbers. Alternatively, one could use a sequence of uniform pseudorandom numbers; this, however, leads to a higher sampling variance.
- Then, for each $n = 0, \ldots, m$, $Z(t_n) \sim N(0,\, t_n)$, and

Sobol sequence in Python:
https://pypi.python.org/pypi/sobol/0.9

$$\begin{aligned} \mathrm{E}\left[Z(t_i)Z(t_j)\right] &= \sum_{0 \le k \le m} \lambda_k\, E_k(t_i) E_k(t_j) \\ &= C_{ij}. \end{aligned} \tag{7}$$

One factor Brownian motion

- We can thus regard $Z(t_n)$ a realization of the discretized Wiener process[1].
- For computational efficiency, we may want to truncate (6) at some $p < m$. This eliminates the *high frequencies* from $Z(t_n)$, and lowers the variance. The price for this may be systematically lower accuracy.

PCA orthogonal:
Typically, (not always true)
set 1st component all positive
2nd orthogonal on 1st, switch sign once
3rd orthogonal on 1st and 2nd, switch sign twice
...
m-th orthogonal on 1st, 2nd,.... m-1, switch sign m-1 times -> high frequency

[1] This realization of the discretized Wiener process is related to the well known Karhounen-Loeve expansion of the (continuous time) Wiener process.

Multi factor Brownian motion

- We now consider the case of a multi-factor Brownian motion $Z_a(t)$, with

$$\mathrm{E}[dZ_a(t)\, dZ_b(t)] = \rho_{ab} dt.$$

- The Cholesky decomposition of ρ yields

$$\rho = LL^{\mathrm{T}}, \tag{8}$$

where L is a $d \times d$ dimensional, lower triangular matrix.

- For example, if

$$\rho = \begin{bmatrix} 1 & \rho_{12} \\ \rho_{12} & 1 \end{bmatrix}, \tag{9}$$

then

$$L = \begin{bmatrix} 1 & 0 \\ \rho_{12} & \sqrt{1-\rho_{12}^2} \end{bmatrix}, \tag{10}$$

Multi factor Brownian motion

- Now, if $X \in \mathbb{R}^d$ is a vector of independent standard normal variables, then LX is a multivariate normal variable with correlation matrix ρ.
- Indeed,

$$\begin{aligned}
\mathrm{E}[(LX)_a (LX)_b] &= \sum_{0 \le k,l \le d} L_{ak} L_{bl} \mathrm{E}[X_k X_l] \\
&= \sum_{0 \le k,l \le d} L_{ak} L_{bl} \delta_{kl} \\
&= \sum_{0 \le k \le d} L_{ak} L_{bk} \\
&= \sum_{0 \le k \le d} L_{ak} (L^{\mathrm{T}})_{kb} \\
&= (LL^{\mathrm{T}})_{kl} \\
&= \rho_{kl}.
\end{aligned}$$

Single equation

- Numerical solution of a stochastic differential equation amounts to generating paths of the state variables given a path of the stochastic drivers of the system, namely the underlying Brownian motion.
- This requires approximating the continuous time system by a discrete time stochastic system.
- Consider first a one factor SDE,

$$\begin{aligned} dX(t) &= A(t, X(t))dt + B(t, X(t))dZ(t), \\ X(0) &= X_0. \end{aligned} \tag{11}$$

- This is equivalent to

$$X(s) = X(t) + \int_t^s A(u, X(u))du + \int_t^s B(u, X(u))dZ(u). \tag{12}$$

Single equation

- Now, if $f(t,x)$ is twice continuously differentiable, then Ito's lemma states

$$df(t,X(t)) = \mathcal{L}^0 f(t,X(t))dt + \mathcal{L}^1 f(t,X(t))dZ(t)\,, \tag{13}$$

where the operators $\mathcal{L}^i$ are defined by

$$\mathcal{L}^0 = \frac{\partial}{\partial t} + A\frac{\partial}{\partial x} + \frac{1}{2}B^2\frac{\partial^2}{\partial x^2}\,, \tag{14}$$

and

$$\mathcal{L}^1 = B\frac{\partial}{\partial x}\,. \tag{15}$$

Single equation

- Applying Ito's lemma (13) to A yields

$$A(s, X(s)) = A(t, X(t)) + \int_t^s \mathcal{L}^0 A(u, X(u))du + \int_t^s \mathcal{L}^1 B(u, X(u))dZ(u)$$
$$\approx A(t, X(t)) + \mathcal{L}^0 A(t, X(t)) \int_t^s du + \mathcal{L}^1 B(t, X(t)) \int_t^s dZ(u).$$

- We can thus approximate

$$\int_t^{t+\delta} A(s, X(s))ds \approx A(t, X(t))\delta + \mathcal{L}^0 A(t, X(t))I_{(0,0)} + \mathcal{L}^1 B(t, X(t))I_{(1,0)}.$$

- Here

$$I_{(0,0)} = \int_t^{t+\delta} \int_t^s du\, ds, \qquad (16)$$
$$I_{(1,0)} = \int_t^{t+\delta} \int_t^s dZ(u)\ ds,$$

are iterated integrals.

Single equation

- Similarly, we make the following approximation:

$$B(s, X(s)) = B(t, X(t)) + \int_t^s \mathcal{L}^0 B(u, X(u))du + \int_t^s \mathcal{L}^1 B(u, X(u))dZ(u)$$
$$\approx B(t, X(t)) + \mathcal{L}^0 B(t, X(t)) \int_t^s du + \mathcal{L}^1 B(t, X(t)) \int_t^s dZ(u).$$

- Therefore,

$$\int_t^{t+\delta} B(s, X(s))dZ(s) \approx B(t, X(t))\Delta Z(t) + \mathcal{L}^0 B(t, X(t))I_{(0,1)} + \mathcal{L}^1 B(t, X(t))I_{(1,1)},$$

where $\Delta Z(t) = Z(t+\delta) - Z(t)$, and

$$I_{(0,1)} = \int_t^{t+\delta} \int_t^s du\, dZ(s), \qquad (17)$$
$$I_{(1,1)} = \int_t^{t+\delta} \int_t^s dZ(u)\ dZ(s).$$

Single equation

- As a result, we obtain the following approximation:

$$\begin{aligned} X(t+\delta) = X(t) + A(t,X(t))\delta + B(t,X(t))\Delta Z(t) + \mathcal{L}^0 A(t,X(t))I_{(0,0)} \\ + \mathcal{L}^1 A(t,X(t))I_{(1,0)} + \mathcal{L}^0 B(t,X(t))I_{(0,1)} + \mathcal{L}^1 B(t,X(t))I_{(1,1)}. \end{aligned} \tag{18}$$

- Note:

$$\begin{aligned} I_{(0,0)} &= \int_t^{t+\delta} (s-t)ds = \frac{1}{2}\delta^2, \\ I_{(1,1)} &= \int_t^{t+\delta} (Z(s)-Z(t))dZ(s) = \frac{1}{2}\left((\Delta Z)^2 - \delta\right), \\ I_{(0,1)} &= \int_t^{t+\delta} (s-t)dZ(s) = \delta\,\Delta Z - I_{(1,0)}, \\ I_{(1,0)} &= \int_t^{t+\delta} (Z(s)-Z(t))ds. \end{aligned} \tag{19}$$

- In particular, it is very fortuitous that $I_{(1,1)}$ can be computed in a closed, easy to simulate form.

Euler's scheme

- This approximation leads to practical discretization schemes of (11). We consider a sequence of times $0 = t_0 < t_1 < \ldots < t_m = T$.
- The first such scheme, Euler's scheme, consists in retaining the first three terms on the right hand side of (18):

$$X_{n+1} = X_n + A(t_n, X_n)\delta_n + B(t_n, X_n)\Delta Z_n, \tag{20}$$

where $\delta_n = t_{n+1} - t_n$, and $\xi_n \sim N(0, 1)$. The random variables ξ_n are assumed independent.

DeltaZ(n)=\sqrt{\delta_n}\zeta_n

Milstein's scheme

This method does not work for SABR model

- In the second scheme, *Milstein's scheme*, in addition to the terms present in Euler's scheme, we also retain the last term on the right hand side of (18).
- Note that this term is of order of magnitude δ, while the three discarded terms are of order of magnitude $\delta^{3/2}$ and δ^2.
- Explicitly, Milstein's scheme is given by

$$X_{n+1} = X_n + A(t_n, X_n)\delta_n + B(t_n, X_n)\Delta Z_n + \frac{1}{2} B(t_n, X_n)B'(t_n, X_n)\left(\Delta Z_n^2 - \delta_n\right),$$

where $'$ denotes the derivative with respect to x.

Slower than Euler's method, but more accuate

Systems of SDEs

- We now consider an n-dimensional state variable $X \in \mathbb{R}^n$ driven by a d-dimensional Brownian motion $Z(t) \in \mathbb{R}^d$,

$$dX_i(t) = A_i(t, X(t))dt + \sum_{1 \leq a \leq d} B_{ia}(t, X(t))dZ_a(t), \tag{21}$$

  where, for simplicity, we assume that the components of Z are independent.
- This implies that

$$X_i(t+\delta) = X_i(t) + \int_t^{t+\delta} A_i(s, X(s))ds + \sum_{1 \leq a \leq d} \int_t^{t+\delta} B_{ia}(s, X(s))dZ_a(s). \tag{22}$$

- The following calculations generalize the calculations we carried out above for the case of a single factor SDE.

Systems of SDEs

- If $f(t,x)$ is twice continuously differentiable, then Ito's lemma states

$$df(t,X(t)) = \mathcal{L}^0 f(t,X(t))dt + \sum_{1\leq a\leq d} \mathcal{L}^a f(t,X(t))dZ_a(t)\,, \tag{23}$$

where the operators $\mathcal{L}^i$ are defined by

$$\mathcal{L}^0 = \frac{\partial}{\partial t} + \sum_{1\leq i\leq n} A_i \frac{\partial}{\partial x_i} + \frac{1}{2}\sum_{1\leq i,j\leq n}\sum_{1\leq a\leq d} B_{ia}B_{ja}\frac{\partial^2}{\partial x_i \partial x_j}\,, \tag{24}$$

and

$$\mathcal{L}^a = \sum_{1\leq i\leq n} B_{ia}\frac{\partial}{\partial x_i}\,,\ \text{for } a = 1,\ldots,d. \tag{25}$$

Systems of SDEs

- Applying Ito's lemma (23) to A_i yields

$$\begin{aligned} A_i(s, X(s)) &= A_i(t, X(t)) + \int_t^s \mathcal{L}^0 A_i(u, X(u))du \\ &\quad + \sum_{1 \le a \le d} \int_t^s \mathcal{L}^a B_{ia}(u, X(u))dZ_a(u) \\ &\approx A_i(t, X(t)) + \mathcal{L}^0 A_i(t, X(t)) \int_t^s du \\ &\quad + \sum_{1 \le a \le d} \mathcal{L}^a B_{ia}(t, X(t)) \int_t^s dZ_a(u). \end{aligned}$$

Systems of SDEs

- We can thus approximate

$$\int_t^{t+\delta} A_i(s, X(s))ds \approx A_i(t, X(t))\delta + \mathcal{L}^0 A_i(t, X(t))I_{(0,0)} + \sum_{1\leq a\leq d} \mathcal{L}^a B_{ia}(t, X(t))I_{(a,0)}.$$

- Here

$$I_{(0,0)} = \int_t^{t+\delta}\int_t^s du\, ds, \qquad I_{(a,0)} = \int_t^{t+\delta}\int_t^s dZ_a(u)\, ds, \tag{26}$$

are iterated integrals.

Systems of SDEs

- Similarly, we make the following approximation:

$$
\begin{aligned}
B_{ia}(s, X(s)) &= B_{ia}(t, X(t)) + \int_t^s \mathcal{L}^0 B_{ia}(u, X(u))du \\
&\quad + \sum_{1 \le b \le d} \int_t^s \mathcal{L}^b B_{ia}(u, X(u)) dZ_b(u) \\
&\approx B_{ia}(t, X(t)) + \mathcal{L}^0 B_{ia}(t, X(t)) \int_t^s du \\
&\quad + \sum_{1 \le b \le d} \mathcal{L}^b B_{ia}(t, X(t)) \int_t^s dZ_b(u).
\end{aligned}
$$

Systems of SDEs

- Therefore,

$$\int_t^{t+\delta} B_{ia}(s, X(s))dZ_a(s) \approx B_{ia}(t, X(t))\Delta Z_a(t) + \mathcal{L}^0 B_{ia}(t, X(t))I_{(0,a)} + \sum_{1\le b\le d} \mathcal{L}^b B_{ia}(t, X(t))I_{(a,b)},$$

where

$$\begin{aligned} I_{(0,b)} &= \int_t^{t+\delta}\int_t^s du\, dZ_b(s), \\ I_{(a,b)} &= \int_t^{t+\delta}\int_t^s dZ_a(u)\, dZ_b(s). \end{aligned} \tag{27}$$

- The integral $I_{(a,b)}$, for $a \neq b$, is known as the *Levy area*. There is no close form expression for the Ley area, and it is computationally expensive to simulate.

Systems of SDEs

- As a result, we obtain the following approximation:

$$
\begin{aligned}
X_i(t+\delta) = X_i(t) + A_i(t, X(t))\delta + \sum_{1\le a\le d} B_{ia}(t, X(t))\Delta Z_a(t) \\
+ \mathcal{L}^0 A_i(t, X(t)) I_{(0,0)} + \sum_{1\le a\le d} \mathcal{L}^a A_i(t, X(t)) I_{(a,0)} \\
+ \sum_{1\le b\le d} \left(\mathcal{L}^0 B_{ib}(t, X(t)) I_{(0,b)} + \sum_{1\le a\le d} \mathcal{L}^a B_{ib}(t, X(t)) I_{(a,b)} \right),
\end{aligned}
\tag{28}
$$

where

$$
\Delta Z_a(t) = Z_a(t+\delta) - Z_a(t).
$$

Systems of SDEs

- Note that:

$$
\begin{aligned}
I_{(0,0)} &= \int_t^{t+\delta} (s-t)ds \\
&= \frac{1}{2}\delta^2, \\
I_{(a,a)} &= \int_t^{t+\delta} (Z_a(s) - Z_a(t))dZ_a(s) \\
&= \frac{1}{2}\left((\Delta Z_a)^2 - \delta\right), \\
I_{(0,a)} &= \int_t^{t+\delta} (s-t)dZ_a(s) \\
&= \delta\Delta Z_a - I_{(a,0)}, \\
I_{(a,0)} &= \int_t^{t+\delta} (Z_a(s) - Z_a(t))ds
\end{aligned}
\tag{29}
$$

- Note, in particular, that $I_{(a,a)}$ admits a simple, closed form expression.

Integrability condition

- In order to deal with the Levy areas $I_{(a,b)}$, we impose the following *integrability condition*:

$$\mathcal{L}^a B_{ib} = \mathcal{L}^b B_{ia}, \tag{30}$$

or explicitly

$$\sum_{1 \le k \le n} B_{ka} \frac{\partial B_{ib}}{\partial x_k} = \sum_{1 \le k \le n} B_{kb} \frac{\partial B_{ia}}{\partial x_k} . \tag{31}$$

- Note that then

$$\begin{aligned} I_{(a,b)} + I_{(b,a)} &= \int_t^{t+\delta} \int_t^s \left(dZ_a(u)\, dZ_b(s) + dZ_b(u)\, dZ_a(s) \right) \\ &= \Delta Z_a \, \Delta Z_b . \end{aligned} \tag{32}$$

- In other words, the Levy areas $I_{(a,b)}$ and $I_{(b,a)}$ conspire to add up to a simple, easy to simulate expression!

Integrability condition

- Therefore, when the integrability condition holds, (28) can be written as

$$
\begin{aligned}
&X_i(t+\delta) \\
&\quad = X_i(t) + A_i(t, X(t))\delta + \sum_{1\le a\le d} B_{ia}(t, X(t))\Delta Z_a(t) + \frac{1}{2}\mathcal{L}^0 A_i(t, X(t))\delta^2 \\
&\quad + \sum_{1\le a\le d}\Big(\left(\mathcal{L}^a A_i(t, X(t))\mathcal{L}^0 B_{ia}(t, X(t))\right) I_{(a,0)} + \mathcal{L}^0 B_{ia}(t, X(t))\Delta Z_a\delta \\
&\quad + \sum_{1\le a\le d}\Big(\frac{1}{2}\mathcal{L}^a B_{ia}(t, X(t))(\Delta Z_a^2 - \delta) + \sum_{a+1\le b\le d} \mathcal{L}^a B_{ib}(t, X(t))\Delta Z_a \Delta Z_b\Big).
\end{aligned}
\tag{33}
$$

See notes in jupyter notebook

- This approximation leads to the following two discretization schemes.

Euler's scheme

- Euler's scheme is obtained by discarding all but the first three terms on the right hand side of (28):

$$X_{i,n+1} = X_{i,n} + A_{i,n}\delta_n + \sum_{1 \le a \le d} B_{ia,n}\,\Delta Z_{a,n}. \tag{34}$$

- Euler's scheme is of order of convergence $1/2$ meaning that the approximate solution converges in a suitable norm to the actual solution at the rate of $\delta t^{1/2}$, as $\delta t \equiv \max \delta t_n \to 0$.

Milstein's scheme

- Milstein's scheme includes the last term in (33):

$$X_{i,n+1} = X_{i,n} + A_{i,n}\delta_n + \sum_{1\le a\le d} B_{ia,n}\,\Delta Z_{a,n} + \sum_{1\le a\le d}\Big(\frac{1}{2}\,\mathcal{L}^a B_{ia,n}(\Delta Z_{a,n}^2 - \delta_n) + \sum_{a+1\le b\le d} \mathcal{L}^a B_{ib,n}\,\Delta Z_{a,n}\,\Delta Z_{b,n}\Big).$$

- This can be rewritten in a more symmetric form as:

$$X_{i,n+1} = X_{i,n} + \Big(A_{i,n} - \frac{1}{2}\sum_{1\le a\le d}\mathcal{L}^a B_{ia,n}\Big)\delta_n + \sum_{1\le a\le d} B_{ia,n}\,\Delta Z_{a,n} + \frac{1}{2}\sum_{1\le a,b\le d}\mathcal{L}^a B_{ib,n}\,\Delta Z_{a,n}\,\Delta Z_{b,n}. \tag{35}$$

- Milstein's scheme is of order of convergence 1 meaning that the approximate solution converges in a suitable norm to the actual solution at the rate of δt, as $\delta t \to 0$.

Discretizing LMM: Euler's scheme

- We choose a sequence of event dates $t_0, t_1, \ldots, t_m$, and denote by $L_{jn} \simeq L_j(t_n)$ the approximate solution. We also set

$$\begin{aligned} \Delta_{j,n} &= \Delta_j(t_n, L_n), \\ B_{ja,n} &= B_{ja}(t_n, L_{j,n}), \end{aligned} \tag{36}$$

  and $\delta_n = t_{n+1} - t_n$. Forward: stop at time tj for each Lj
- Applied to LMM, Euler's scheme (34) reads:

$$L_{j,n+1} = L_{j,n} + \Delta_{j,n}\delta_n + \sum_{1 \le a \le d} B_{ja,n}\, \Delta Z_{a,n}\ , \tag{37}$$

  where, as before, $\Delta Z_{a,n} = Z_a(t_{n+1}) - Z_a(t_n)$ is the discretized Brownian motion.

Discretizing LMM: Milstein's scheme

Milstein's does not work for Normal LLM

- Fortunately, LMM is in the category of models which satisfy the integrability condition required for Milstein's scheme to work.
- In order to lighten up the notation, let us define:

$$\Upsilon_{jab,n} \equiv B_{ja}(t_n, L_{j,n}) \frac{\partial B_{jb}(t_n, L_{j,n})}{\partial L_j} . \tag{38}$$

- Then Milstein's scheme (35) applied to the LMM model reads:

$$\begin{aligned} L_{j,n+1} = L_{j,n} + \left(\Delta_{j,n} - \frac{1}{2} \sum_{1 \le a \le d} \Upsilon_{jaa,n}\right) \delta_n \\ + \sum_{1 \le a \le d} B_{ja,n} \, \Delta Z_{a,n} + \frac{1}{2} \sum_{1 \le a,b \le d} \Upsilon_{jab,n} \, \Delta Z_{a,n} \, \Delta Z_{n,b} . \end{aligned} \tag{39}$$

Efficient drift calculation

- A bit of a challenge lies in handling the drift terms. Because of their complexity, their calculation (at each time step) takes up to 50% of the total computation time.
- On the other hand, they are relatively small as compared to the initial values of the LIBOR forwards, and it would be desirable to develop an efficient methodology for accurate approximate evaluation of the drift terms.
- The first and simplest approach consist in "freezing" the values of $F_j(t)$ at the initial value $F_{j,0} \equiv F_j(0)$. We precompute the values

$$\Delta_{j,0} \equiv \Delta_j(t, F_0), \tag{40}$$

and use them for the drift terms throughout the simulation. This approximation, *the frozen curve approximation*, is rather crude, and does not perform very well when applied to pricing longer dated instruments.

- Going one step in the low noise expansion beyond the frozen curve approximation produces satisfying results.

Efficient drift calculation

- The second approach is a refinement of the frozen curve approximation, and consists in the following. From Ito's lemma,

$$\begin{aligned}\Delta_j(t, F(t)) &= \Delta_{j,0} + \int_0^t \mathcal{L}^a \Delta_j(s, F(s)) dZ_a(s) \\ &\simeq \Delta_{j,0} + \mathcal{L}^a \Delta_j(0, F_0) Z_a(t),\end{aligned} \tag{41}$$

  where we have suppressed all terms of order higher than $1/2$.
- We thus arrive at the following approximation8, *the order $1/2$ approximation*:

$$\Delta_{j,1/2}(t) \equiv \Delta_{j,0} + \mathcal{L}^a \Delta_j(0, F_0) Z_a(t). \tag{42}$$

Efficient drift calculation

- The coefficients $\mathcal{L}^a\Delta_j$ in the formula above are explicitly given by the following expressions.
- Under the forward measure Q_k:

$$\mathcal{L}^a\Delta_j = U_{ja}C_j \times \begin{cases} -\sum\limits_{j+1\le i\le k} \frac{\rho_{ji}\delta_i C_i}{1+\delta_i F_i}\left[U_{ja}\frac{\partial C_j}{\partial F_j} + U_{ia}\left(\frac{\partial C_i}{\partial F_i} - \frac{\delta_i C_i}{1+\delta_i F_i}\right)\right], & \text{if } j<k \\ 0, & \text{if } j=k, \\ \sum\limits_{k+1\le i\le j} \frac{\rho_{ji}\delta_i C_i}{1+\delta_i F_i}\left[U_{ja}\frac{\partial C_j}{\partial F_j} + U_{ia}\left(\frac{\partial C_i}{\partial F_i} - \frac{\delta_i C_i}{1+\delta_i F_i}\right)\right], & \text{if } j>k. \end{cases} \tag{43}$$

- Under the spot measure:

$$\mathcal{L}^a\Delta_j = U_{ja}C_j \sum_{\gamma(t)\le i\le j} \frac{\rho_{ji}\delta_i C_i}{1+\delta_i F_i}\left[U_{ja}\frac{\partial C_j}{\partial F_j} + U_{ia}\left(\frac{\partial C_i}{\partial F_i} - \frac{\delta_i C_i}{1+\delta_i F_i}\right)\right]. \tag{44}$$

Efficient drift calculation

- The *order* $3/4$ *approximation*, uses the next order term in the low noise expansion:

$$\Delta_{j,3/4}(t) \equiv \Delta_{j,0}(t, L_0) + \Gamma_{ja} Z_a(t) + \Omega_j t.$$

- Under the forward measure Q_k, the coefficients Γ_{ja} are given by:

$$\Gamma_{ja} = \begin{cases} -C_j \sum_{j+1 \le i \le k} \frac{\rho_{ji}\delta_i C_i}{1+\delta_i L_i} \left[U_{ja} \frac{\partial C_j}{\partial L_j} + U_{ia} \left(\frac{\partial C_i}{\partial L_i} - \frac{\delta_i C_i}{1+\delta_i L_i} \right) \right], & \text{if } j < k, \\ 0, & \text{if } j = k, \\ C_j \sum_{k+1 \le i \le j} \frac{\rho_{ji}\delta_i C_i}{1+\delta_i L_i} \left[U_{ja} \frac{\partial C_j}{\partial L_j} + U_{ia} \left(\frac{\partial C_i}{\partial L_i} - \frac{\delta_i C_i}{1+\delta_i L_i} \right) \right], & \text{if } j > k, \end{cases}$$

Efficient drift calculation

- The coefficients Ω_j are given by:

$$\Omega_j = \begin{cases} -\sum\limits_{j+1\le i\le k} \frac{\rho_{ji}\delta_i}{1+\delta_i L_i}\left[\Delta_{j,0} C_i \frac{\partial C_j}{\partial L_j} + \Delta_{i,0} C_j \left(\frac{\partial C_i}{\partial L_i} - \frac{\delta_i C_i}{1+\delta_i L_i}\right)\right], & \text{if } j < k, \\ 0, & \text{if } j = k, \\ \sum\limits_{k+1\le i\le j} \frac{\rho_{ji}\delta_i}{1+\delta_i L_i}\left[\Delta_{j,0} C_i \frac{\partial C_j}{\partial L_j} + \Delta_{i,0} C_j \left(\frac{\partial C_i}{\partial L_i} - \frac{\delta_i C_i}{1+\delta_i L_i}\right)\right], & \text{if } j > k. \end{cases}$$

Efficient drift calculation

- Under the spot measure,

$$\Gamma_{ja} = C_j \sum_{\gamma(t)\le i\le j} \frac{\rho_{ji}\delta_i C_i}{1+\delta_i L_i}\Big[U_{ja}\frac{\partial C_j}{\partial L_j} + U_{ia}\Big(\frac{\partial C_i}{\partial L_i} - \frac{\delta_i C_i}{1+\delta_i L_i}\Big)\Big],$$

and

$$\Omega_j = \sum_{\gamma(t)\le i\le j} \frac{\rho_{ji}\delta_i}{1+\delta_i L_i}\Big[\Delta_{j,0} C_i \frac{\partial C_j}{\partial L_j} + \Delta_{i,0} C_j\Big(\frac{\partial C_i}{\partial L_i} - \frac{\delta_i C_i}{1+\delta_i L_i}\Big)\Big].$$

The order $3/4$ approximation leads to excellent accuracy.

- On might easily refine this approach by computing terms of higher order in stochastic Taylor's expansion. This leads, however, to more complex and computationally expensive formulas, and the benefit of using an asymptotic expansion disappears. The order $1/2$ approximation appears to offer the best performance versus accuracy profile.

References


Glasserman, P.: *Monte Carlo Methods in Financial Engineering*, Springer Verlag (2003).


Kloeden, P. E., and Platen, E.: *Numerical Solution of Stochastic Differential Equations*, Springer Verlag (1992).