CHAPTER EIGHTEEN

Numerical Methods

All science is dominated by the idea of approximation.
Bertrand Russell

Stochastic differential equations are closely related to second-order elliptic and parabolic partial differential equations [572]. Besides the binomial tree algorithms already covered, two more prominent approaches are numerical methods for partial differential equations and Monte Carlo simulation. Both are investigated in this chapter. Techniques that reduce the variance of the Monte Carlo simulation are also explored. A deterministic version of the Monte Carlo simulation, called the **quasi-Monte Carlo method**, replaces probabilistic error bounds with deterministic bounds. It is covered in some detail.

18.1 Finite-Difference Methods

The **finite-difference method** places a grid of points on the space over which the desired function takes value and then approximates the function value at each of these points. See Fig. 18.1 for illustration. The method solves the equation numerically by introducing difference equations to approximate derivatives.

Take the Poisson equation $\partial^2\theta/\partial x^2 + \partial^2\theta/\partial y^2 = -\rho(x, y)$ as an example. By replacing the second derivatives with finite differences through central difference and introducing evenly spaced grid points with distance of Δx along the x axis and Δy along the y axis, the finite-difference form is

$$\frac{\theta(x_{i+1}, y_j) - 2\theta(x_i, y_j) + \theta(x_{i-1}, y_j)}{(\Delta x)^2} + \frac{\theta(x_i, y_{j+1}) - 2\theta(x_i, y_j) + \theta(x_i, y_{j-1})}{(\Delta y)^2} = -\rho(x_i, y_j).$$

In the preceding equation, $\Delta x \equiv x_i - x_{i-1}$ and $\Delta y \equiv y_j - y_{j-1}$ for $i, j = 1, 2, \ldots$. When the grid points are evenly spaced in both axes so that $\Delta x = \Delta y = h$, the difference equation becomes

$$\theta(x_{i+1}, y_j) + \theta(x_{i-1}, y_j) + \theta(x_i, y_{j+1}) + \theta(x_i, y_{j-1}) - 4\theta(x_i, y_j) = -h^2\rho(x_i, y_j).$$

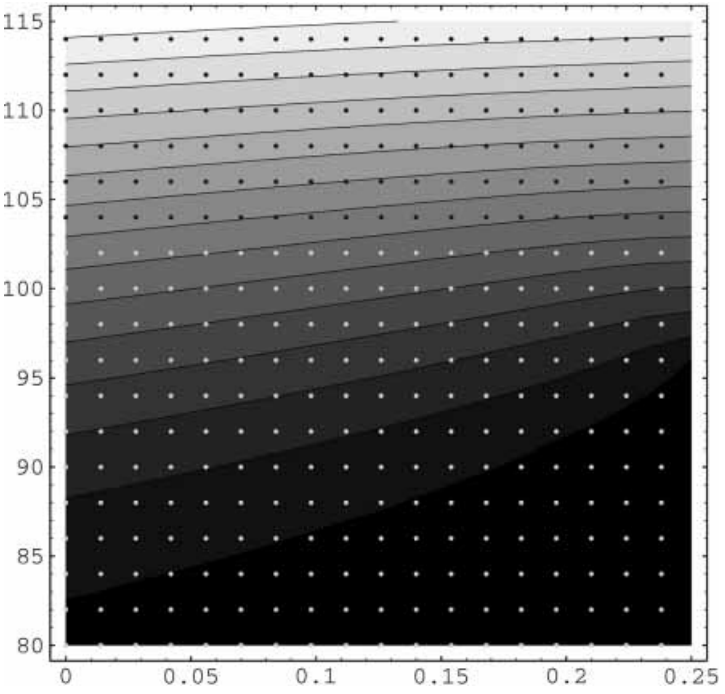

Figure 18.1: Finite-difference method. Grids are shown over the rectangle $[0, 0.25] \times [80, 115]$. The analytical solution to the Black–Scholes differential equation for a European call is illustrated here with darker gray denoting smaller value. The strike price is \$95, the expiration date is $x = 0.25$ (year), and the x axis denotes time. The finite-difference method finds values at the discrete grid points that match or approximate the analytical values.

Given the boundary values of $\theta(x, y)$ at $x = \pm L$ and $y = \pm L$, we can solve for the x_is and the y_js within the square $[\pm L, \pm L]$. From now on, $\theta_{i,j}$ denotes the finite-difference approximation to the exact $\theta(x_i, y_j)$ for clarity.

18.1.1 Explicit Methods

Consider another example, the diffusion equation $D(\partial^2\theta/\partial x^2) - (\partial\theta/\partial t) = 0$. Again, we use evenly spaced grid points (x_i, t_j) with distances Δx and Δt, where $\Delta x \equiv x_{i+1} - x_i$ and $\Delta t \equiv t_{j+1} - t_j$. We use the central difference for the second derivative and the forward difference for the time derivative to obtain

$$\left.\frac{\partial\theta(x,t)}{\partial t}\right|_{t=t_j} = \frac{\theta(x, t_{j+1}) - \theta(x, t_j)}{\Delta t} + O(\Delta t), \tag{18.1}$$

$$\left.\frac{\partial^2\theta(x,t)}{\partial x^2}\right|_{x=x_i} = \frac{\theta(x_{i+1}, t) - 2\theta(x_i, t) + \theta(x_{i-1}, t)}{(\Delta x)^2} + O[(\Delta x)^2]. \tag{18.2}$$

To assemble Eqs. (18.1) and (18.2) into a single equation at (x_i, t_j), we need to decide how to evaluate x in the first equation and t in the second. Because the central difference around x_i is used in Eq. (18.2), we might as well use x_i for x in Eq. (18.1). Two choices are possible for t in Eq. (18.2). The first choice uses $t = t_j$

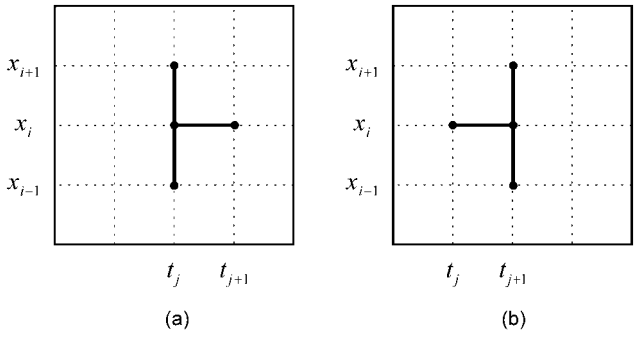

Figure 18.2: Explicit and implicit methods. Stencils of grid points for (a) explicit and (b) implicit finite-difference methods in solving a partial differential equation that is first order in t and second order in x.

to yield the following finite-difference equation:

$$\frac{\theta_{i,j+1} - \theta_{i,j}}{\Delta t} = D \frac{\theta_{i+1,j} - 2\theta_{i,j} + \theta_{i-1,j}}{(\Delta x)^2}. \tag{18.3}$$

The stencil of grid points involves four values, $\theta_{i,j+1}$, $\theta_{i,j}$, $\theta_{i+1,j}$, and $\theta_{i-1,j}$. We can therefore calculate $\theta_{i,j+1}$ from the other three, $\theta_{i,j}$, $\theta_{i+1,j}$, $\theta_{i-1,j}$, at the previous time t_j (see Fig. 18.2(a)). Starting from the initial conditions at t_0, that is, $\theta_{i,0} = \theta(x_i, t_0)$, $i = 1, 2, \ldots$, we calculate $\theta_{i,1}$, $i = 1, 2, \ldots$, and then $\theta_{i,2}$, $i = 1, 2, \ldots$, and so on. This approach is called the **explicit method** [883].

The explicit method is numerically unstable unless $\Delta t \leq (\Delta x)^2/(2D)$. A numerical method is said to be **unstable** if the solution is highly sensitive to changes in initial conditions [391]. The stability condition may lead to high running times and memory requirements. For instance, doubling $(\Delta x)^{-1}$ would imply quadrupling $(\Delta t)^{-1}$, resulting in a running time eight times as much. This undesirable feature can be remedied by use of the **implicit method**, to be introduced shortly.

An interesting connection exists between the explicit method and the trinomial model. Rearrange Eq. (18.3) as

$$\theta_{i,j+1} = \frac{D\Delta t}{(\Delta x)^2}\theta_{i+1,j} + \left[1 - \frac{2D\Delta t}{(\Delta x)^2}\right]\theta_{i,j} + \frac{D\Delta t}{(\Delta x)^2}\theta_{i-1,j}.$$

When the stability condition is satisfied, the three coefficients for $\theta_{i+1,j}$, $\theta_{i,j}$, and $\theta_{i-1,j}$ all lie between zero and one and sum to one. They can therefore be interpreted as probabilities. Consequently the finite-difference equation becomes identical to backward induction on trinomial trees [473, 575].

➤ **Exercise 18.1.1** Sketch the finite-difference version of the Poisson equation in matrix form.

18.1.2 Implicit Methods

If we use $t = t_{j+1}$ in Eq. (18.2) instead, the finite-difference equation becomes

$$\frac{\theta_{i,j+1} - \theta_{i,j}}{\Delta t} = D \frac{\theta_{i+1,j+1} - 2\theta_{i,j+1} + \theta_{i-1,j+1}}{(\Delta x)^2}. \tag{18.4}$$

The stencil involves $\theta_{i,j}$, $\theta_{i,j+1}$, $\theta_{i+1,j+1}$, and $\theta_{i-1,j+1}$. This method is implicit because the value of any one of the three quantities at t_{j+1} cannot be calculated unless the other two are known (see Fig. 18.2(b)). It is also called the **fully implicit backward-difference scheme**. Equation (18.4) can be rearranged as

$$\theta_{i-1,j+1} - (2+\gamma)\,\theta_{i,j+1} + \theta_{i+1,j+1} = -\gamma\theta_{i,j},$$

where $\gamma \equiv (\Delta x)^2/(D\Delta t)$. This equation is unconditionally stable [28, 464]. Suppose the boundary conditions are given at $x = x_0$ and $x = x_{N+1}$. After $\theta_{i,j}$ has been calculated for $i = 1, 2, \ldots, N$, the values of $\theta_{i,j+1}$ at time t_{j+1} can be computed as the solution to the following tridiagonal linear system:

$$\begin{bmatrix} a & 1 & 0 & \cdots & \cdots & \cdots & 0 \\ 1 & a & 1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & a & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & 1 & a & 1 \\ 0 & \cdots & \cdots & \cdots & 0 & 1 & a \end{bmatrix} \begin{bmatrix} \theta_{1,j+1} \\ \theta_{2,j+1} \\ \theta_{3,j+1} \\ \vdots \\ \vdots \\ \vdots \\ \theta_{N,j+1} \end{bmatrix} = \begin{bmatrix} -\gamma\theta_{1,j} - \theta_{0,j+1} \\ -\gamma\theta_{2,j} \\ -\gamma\theta_{3,j} \\ \vdots \\ \vdots \\ -\gamma\theta_{N-1,j} \\ -\gamma\theta_{N,j} - \theta_{N+1,j+1} \end{bmatrix},$$

where $a \equiv -2 - \gamma$. Tridiagonal systems can be solved in $O(N)$ time and $O(N)$ space [35]. The preceding matrix is nonsingular when $\gamma \geq 0$. Recall that a square matrix is **nonsingular** if its inverse exists.

Taking the average of explicit method (18.3) and implicit method (18.4) results in

$$\frac{\theta_{i,j+1} - \theta_{i,j}}{\Delta t} = \frac{1}{2}\left[D\,\frac{\theta_{i+1,j} - 2\theta_{i,j} + \theta_{i-1,j}}{(\Delta x)^2} + D\,\frac{\theta_{i+1,j+1} - 2\theta_{i,j+1} + \theta_{i-1,j+1}}{(\Delta x)^2} \right].$$

After rearrangement, the **Crank–Nicolson method** emerges:

$$\gamma\theta_{i,j+1} - \frac{\theta_{i+1,j+1} - 2\theta_{i,j+1} + \theta_{i-1,j+1}}{2} = \gamma\theta_{i,j} + \frac{\theta_{i+1,j} - 2\theta_{i,j} + \theta_{i-1,j}}{2}.$$

This is an unconditionally stable implicit method with excellent rates of convergence [810].

18.1.3 Numerically Solving the Black–Scholes Differential Equation

We focus on American puts; the technique, however, can be applied to any derivative satisfying the Black–Scholes differential equation as only the initial and the boundary conditions need to be changed.

The Black–Scholes differential equation for American puts is

$$\frac{1}{2}\sigma^2 S^2 \frac{\partial^2 P}{\partial S^2} + (r-q)\,S\,\frac{\partial P}{\partial S} - r\,P + \frac{\partial P}{\partial t} = 0$$

with $P(S, T) = \max(X - S, 0)$ and $P(S, t) = \max(\overline{P}(S, t), X - S)$ for $t < T$. $\overline{P}$ denotes the option value at time t if it is not exercised for the next instant of time.

After the change of variable $V \equiv \ln S$, the option value becomes $U(V,t) \equiv P(e^V, t)$ and

$$\frac{\partial P}{\partial t} = \frac{\partial U}{\partial t}, \quad \frac{\partial P}{\partial S} = \frac{1}{S}\frac{\partial U}{\partial V}, \quad \frac{\partial^2 P}{\partial^2 S} = \frac{1}{S^2}\frac{\partial^2 U}{\partial V^2} - \frac{1}{S^2}\frac{\partial U}{\partial V}.$$

The Black–Scholes differential equation is now transformed into

$$\frac{1}{2}\sigma^2\frac{\partial^2 U}{\partial V^2} + \left(r - q - \frac{\sigma^2}{2}\right)\frac{\partial U}{\partial V} - rU + \frac{\partial U}{\partial t} = 0$$

subject to $U(V,T) = \max(X - e^V, 0)$ and $U(V,t) = \max(\overline{U}(V,t), X - e^V)$, $t < T$. Along the V axis, the grid will span from $V_{\min}$ to $V_{\min} + N \times \Delta V$ at ΔV apart for some suitably small $V_{\min}$; hence boundary conditions at the lower ($V = V_{\min}$) and upper ($V = V_{\min} + N \times \Delta V$) boundaries will have to be specified. Finally, S_0 as usual denotes the current stock price.

The Explicit Scheme

The explicit scheme for the Black–Scholes differential equation is

$$\frac{1}{2}\sigma^2\frac{U_{i+1,j} - 2U_{i,j} + U_{i-1,j}}{(\Delta V)^2} + \left(r - q - \frac{\sigma^2}{2}\right)\frac{U_{i+1,j} - U_{i-1,j}}{2\Delta V} - rU_{i,j} + \frac{U_{i,j} - U_{i,j-1}}{\Delta t} = 0$$

for $1 \le i \le N-1$. Note that the computation moves backward in time. There are $N-1$ difference equations. Regroup the terms to obtain

$$U_{i,j-1} = aU_{i-1,j} + bU_{i,j} + cU_{i+1,j},$$

where

$$a \equiv \left[\left(\frac{\sigma}{\Delta V}\right)^2 - \frac{r - q - \sigma^2/2}{\Delta V}\right]\frac{\Delta t}{2}, \quad b \equiv 1 - r\Delta t - \left(\frac{\sigma}{\Delta V}\right)^2\Delta t,$$

$$c \equiv \left[\left(\frac{\sigma}{\Delta V}\right)^2 + \frac{r - q - \sigma^2/2}{\Delta V}\right]\frac{\Delta t}{2}.$$

These $N-1$ equations express option values at time step $j-1$ in terms of those at time step j. For American puts, we assume for U's lower boundary that the first derivative at grid point $(0, j)$ for every time step j equals $-e^{V_{\min}}$. This essentially makes the put value $X - S = X - e^V$, so $U_{0,j-1} = U_{1,j-1} + (e^{V_{\min}+\Delta V} - e^{V_{\min}})$. For the upper boundary, we set $U_{N,j-1} = 0$. The put's value at any grid point at time step $j-1$ is therefore an explicit function of its values at time step j. Finally $U_{i,j}$ is set to the greater of the value derived above and $X - e^{V_{\min}+i\times\Delta V}$ for early-exercise considerations. Repeating this process as we move backward in time, we will eventually arrive at the solution at time zero, $U_{k,0}$, where k is the integer so that $V_{\min} + k \times \Delta V$ is closest to $\ln S_0$. As implied by the stability condition, given ΔV, the value of Δt must be small enough for the method to converge (see Fig. 18.3). The formal conditions to satisfy are $a > 0$, $b > 0$, and $c > 0$.

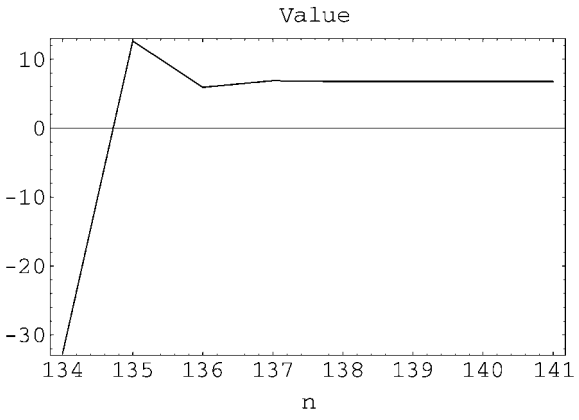

Figure 18.3: Convergence of the explicit method. Here $\Delta V = 0.031073$ and $\Delta t = 5/(6n)$. With $n \geq 137$, the numerical solution converges to the analytical European put value 6.777986.

The explicit method evaluates all the grid points in a rectangle. In practice, we are interested in only the single grid point at time zero, $(0, k)$, that corresponds to the current stock price. The grid points that may influence the desired value form a triangular subset of the rectangle. This triangle could be truncated further by the two boundary conditions (see Fig. 18.4). Only those points within the truncated triangle need be evaluated.

➤ **Exercise 18.1.2** What are the terminal conditions?

➤ **Exercise 18.1.3** Repeat the steps for American calls.

➤ **Exercise 18.1.4** Derive the stability conditions for the explicit approach to solve the Black–Scholes differential equation. Assume $q = 0$ for simplicity.

➢ **Programming Assignment 18.1.5** Implement the explicit method for American puts.

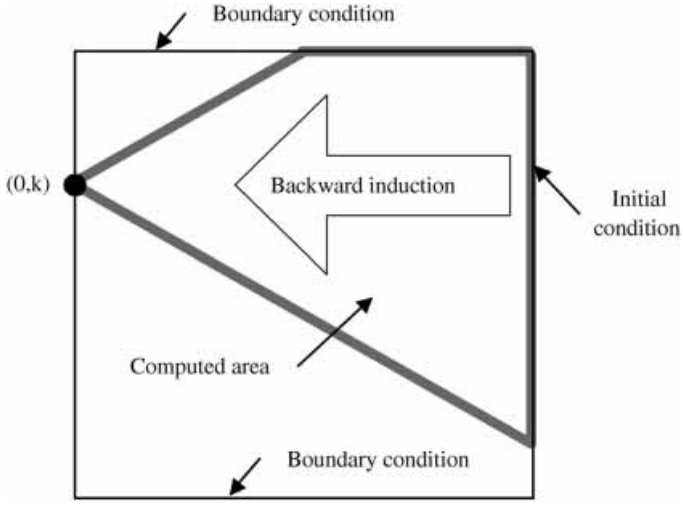

Figure 18.4: Implementation of the explicit method. Only the truncated triangle within the rectangle needs to have its grid points evaluated.

The Implicit Scheme

The partial differential equation now becomes the following $N-1$ difference equations,

$$\frac{1}{2}\sigma^2\frac{U_{i+1,j}-2U_{i,j}+U_{i-1,j}}{(\Delta V)^2}+\left(r-q-\frac{\sigma^2}{2}\right)\frac{U_{i+1,j}-U_{i-1,j}}{2\Delta V}$$
$$-rU_{i,j}+\frac{U_{i,j+1}-U_{i,j}}{\Delta t}=0$$

for $1\le i\le N-1$. Regroup the terms to obtain

$$aU_{i-1,j}+bU_{i,j}+cU_{i+1,j}=U_{i,j+1},$$

where

$$a\equiv\left[-\left(\frac{\sigma}{\Delta V}\right)^2+\frac{r-q-\sigma^2/2}{\Delta V}\right]\frac{\Delta t}{2},\qquad b\equiv 1+r\Delta t+\left(\frac{\sigma}{\Delta V}\right)^2\Delta t,$$
$$c\equiv-\left[\left(\frac{\sigma}{\Delta V}\right)^2+\frac{r-q-\sigma^2/2}{\Delta V}\right]\frac{\Delta t}{2}$$

The system of equations can be written in matrix form:

$$\begin{bmatrix} b^* & c & 0 & \cdots & \cdots & \cdots & 0\\ a & b & c & 0 & \cdots & \cdots & 0\\ 0 & a & b & c & 0 & \cdots & 0\\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots\\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots\\ 0 & \cdots & \cdots & 0 & a & b & c\\ 0 & \cdots & \cdots & \cdots & 0 & a & b \end{bmatrix}\begin{bmatrix} U_{1,j}\\ U_{2,j}\\ U_{3,j}\\ \vdots\\ \vdots\\ \vdots\\ U_{N-1,j} \end{bmatrix}=\begin{bmatrix} U_{1,j+1}-K\\ U_{2,j+1}\\ U_{3,j+1}\\ \vdots\\ \vdots\\ U_{N-2,j+1}\\ U_{N-1,j+1} \end{bmatrix},$$

where $b^*\equiv a+b$ and $K\equiv a(e^{V_{\min}+\Delta V}-e^{V_{\min}})$. We can obtain the values of $U_{1,j}, U_{2,j}, \ldots, U_{N-1,j}$ by inverting the tridiagonal matrix. As before, at every time step and before going to the next, we should set the option value obtained to the intrinsic value of the option if the latter is larger.

➢ **Programming Assignment 18.1.6** Implement the implicit method for American puts.

18.2 Monte Carlo Simulation

Monte Carlo simulation is a sampling scheme. In many important applications within finance and without, Monte Carlo simulation is one of the few feasible tools. It is also one of the most important elements of studying econometrics [550]. When the time evolution of a stochastic process is not easy to describe analytically, Monte Carlo simulation may very well be the only strategy that succeeds consistently [386].

Assume that $X_1, X_2, \ldots, X_n$ have a joint distribution and $\theta\equiv E[\,g(X_1, X_2, \ldots, X_n)\,]$ for some function g is desired. We generate

$$\left(x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}\right),\quad 1\le i\le N,$$

independently with the same joint distribution as $(X_1, X_2, \ldots, X_n)$ and set

$$Y_i \equiv g\left(x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}\right).$$

Now $Y_1, Y_2, \ldots, Y_N$ are independent and identically distributed random variables, and each Y_i has the same distribution as that of $Y \equiv g(X_1, X_2, \ldots, X_n)$. Because the average of these N random variables, $\overline{Y}$, satisfies $E[\,\overline{Y}\,] = \theta$, it can be used to estimate θ. In fact, the **strong law of large numbers** says that this procedure converges almost surely [699]. The number of **replications** (or *independent* trials), N, is called the **sample size**.

EXAMPLE 18.2.1 To evaluate the definite integral $\int_a^b g(x)\,dx$ numerically, consider the random variable $Y \equiv (b-a)\,g(X)$, where X is uniformly distributed over $[\,a, b\,]$. Note that $\text{Prob}[\,X \le x\,] = (x-a)/(b-a)$ for $a \le x \le b$. Because

$$E[\,Y\,] = (b-a)\,E[\,g(X)\,] = (b-a)\int_a^b \frac{g(x)}{b-a}\,dx = \int_a^b g(x)\,dx,$$

any unbiased estimator of $E[\,Y\,]$ can be used to evaluate the integral.

The Monte Carlo estimate and the true value may differ owing to two reasons: sampling variation and the discreteness of the sample paths. The former can be controlled by the number of replications, as we shall see in the following paragraph, and the latter can be controlled by the number of observations along the sample path [147].

The statistical error of the sample mean $\overline{Y}$ of the random variable Y grows as $1/\sqrt{N}$ because $\text{Var}[\,\overline{Y}\,] = \text{Var}[\,Y\,]/N$. In fact, this convergence rate is asymptotically optimal by the **Berry–Esseen theorem** [413]. As a result, the variance of the estimator $\overline{Y}$ can be reduced by a factor of $1/N$ by doing N times as much work [721]. This property is amazing because the same order of convergence holds independently of the dimension n. In contrast, classic numerical integration schemes have an error bound of $O(N^{-c/n})$ for some constant $c > 0$. The required number of evaluations thus grows exponentially in n to achieve a given level of accuracy. This is a case of the curse of dimensionality. The Monte Carlo method, for example, is more efficient than alternative procedures for securities depending on more than one asset, the multivariate derivatives [530].

The statistical efficiency of Monte Carlo simulation can be measured by the variance of its output. If this variance can be lowered without changing the expected value, fewer replications are needed. Methods that improve efficiency in this manner are called **variance-reduction techniques**. Such techniques, covered in Subsection 18.2.3, become practical when the added costs are outweighed by the reduction in sampling.

18.2.1 Monte Carlo Option Pricing

For the pricing of European options on a dividend-paying stock, we may proceed as follows. From Eq. (14.17), stock prices $S_1, S_2, S_3, \ldots$, at times $\Delta t, 2\Delta t, 3\Delta t, \ldots$, can be generated by

$$S_{i+1} = S_i e^{(\mu - \sigma^2/2)\,\Delta t + \sigma\sqrt{\Delta t}\,\xi}, \quad \xi \sim N(0,1)$$

Monte Carlo method for pricing average-rate calls on a non-dividend-paying stock:

```
input:   S, X, n, r, σ, τ, m;
real     P, C, M;
real     ξ( );   // ξ( ) ~ N(0, 1).
integer  i, j;
C := 0;   // Accumulated terminal option value.
for (i = 1 to m) { // Perform m replications.
        P := S; M := S;
        for (j = 1 to n) {
                P := P × e^((r − σ^2/2)(τ/n) + σ√(τ/n) ξ( ));
                M := M + P;
        }
        C := C + max(M/(n + 1) − X, 0);
}
return Ce^(−rτ)/m;
```

Figure 18.5: Monte Carlo method for average-rate calls. m is the number of replications, and n is the number of periods.

when $dS/S = \mu\, dt + \sigma\, dW$. We can generate non-dividend-paying stock prices in a risk-neutral economy by setting $\mu = r$. Figure 18.5 contains a pricing algorithm for arithmetic average-rate calls.

The sample standard deviation of the estimation scheme in Fig. 18.5 is proportional to $1/\sqrt{m}$, where m is the number of replications. To narrow down the confidence interval by a factor of f, f^2 times as many replications need to be carried out. Although we do not know how small $\Delta t \equiv \tau/n$ should be to yield acceptable approximations, it is not hard to figure out m. Because the estimate is composed of a simple average across replications, the central limit theorem says that the error of the estimate is distributed as $N(0, s^2/m)$, with s^2 denoting the variance of each replication. Hence the confidence interval can be used to derive the desired m.

The discreteness of sample paths and the variance in prices do not necessarily make Monte Carlo results inferior to closed-form solutions. The judgment ultimately depends on the security being priced. In reality, for instance, a case may be made that, as prices do not move continuously, discrete-time models are more appropriate.

Monte Carlo simulation is a general methodology. It can be used to value virtually any European-style derivative security [147]. A standard Monte Carlo simulation, however, is inappropriate for American options because of early exercise: It is difficult to determine the early-exercise point based on one single path. Intriguingly, Tilley showed that Monte Carlo simulation can be modified to price American options [842]; the estimate is biased, however [108].

➤ **Exercise 18.2.1** How do we price European barrier options by Monte Carlo simulation?

➤ **Exercise 18.2.2** Consider the Monte Carlo method that estimates the price of the American call by taking the maximum discounted intrinsic value per simulated path and then averaging them: $E[\max_{i=0,1,\ldots,n} e^{-ri\Delta t} \max(S_i - X, 0)]$. Show that it is biased high.

Monte Carlo simulation of Ito process:

input: $x_0, T, \Delta t$;
real $X[0..\lceil T/\Delta t \rceil]$;
real $\xi()$; // $\xi() \sim N(0,1)$.
integer i;
$X[0] := x_0$; // Initial state.
for $(i = 1$ to $\lceil T/\Delta t \rceil)$
$X[i] := X[i-1] + a(X[i-1])\,\Delta t + b(X[i-1])\sqrt{\Delta t}\,\xi()$;
return $X[\,]$;

Figure 18.6: Monte Carlo simulation of Ito process. The Ito process is $dX_t = a(X_t)\,dt + b(X_t)\,dW_t$. A run of the algorithm generates an approximate sample path for the process.

➢ **Programming Assignment 18.2.3** Implement the Monte Carlo method for arithmetic average-rate calls and puts.

18.2.2 Ito Processes

Consider the stochastic differential equation $dX_t = a(X_t)\,dt + b(X_t)\,dW_t$. Although it is often difficult to give an analytic solution to this equation, the simulation of the process on a computer is relatively easy [585]. Recall that Euler's method picks a small number Δt and then approximates the Ito process by

$$\widehat{X}(t_{n+1}) = \widehat{X}(t_n) + a(\widehat{X}(t_n))\Delta t + b(\widehat{X}(t_n))\sqrt{\Delta t}\,\xi,$$

where $\xi \sim N(0,1)$. See Fig. 18.6 for the algorithm. This simulation is exact for any Δt if both the drift a and the diffusion b are constants as in Brownian motion because the sum of independent normal distributions remains normal.

➤ **Exercise 18.2.4** The Monte Carlo method for Ito processes in Fig. 18.6 may not be the most ideal theoretically. Consider the geometric Brownian motion $dX/X = \mu\,dt + \sigma\,dW$. Assume that you have access to a perfect random-number generator for normal distribution. Find a theoretically better algorithm to generate sample paths for X.

➢ **Programming Assignment 18.2.5** Simulate $dX_t = (0.06 - X_t)\,dt + 0.3\,dW_t$ by using $\Delta t \equiv 0.01$. Explain its dynamics.

Discrete Approximations to Ito Processes with Brownian Bridge

Besides the Euler method and the related approximation methods in Subsection 14.2.1, a Brownian bridge is one more alternative. Let the time interval $[0, T]$ be partitioned at time points $t_0 = 0, t_1, t_2, \ldots$. Instead of using

$$W(t_n) = W(t_{n-1}) + \sqrt{t_n - t_{n-1}}\,\xi, \quad \xi \sim N(0,1)$$

to generate the discrete-time Wiener process, the new method uses

$$W(t_n) = \frac{t_{n+1} - t_n}{t_{n+1} - t_{n-1}} W(t_{n-1}) + \frac{t_n - t_{n-1}}{t_{n+1} - t_{n-1}} W(t_{n+1}) + \sqrt{\frac{(t_{n+1} - t_n)(t_n - t_{n-1})}{t_{n+1} - t_{n-1}}}\, \xi,$$

given a past value $W(t_{n-1})$ and a future value $W(t_{n+1})$. In general, the method determines a sample path $W(i(T/2^m))$, $i = 0, 1, \ldots, 2^m$, over $[0, T]$ as follows. First, set $W(0) = 0$ and $W(T) = \sqrt{T}\,\xi$. Then set the midpoint $W(T/2)$ according to the preceding equation. From here, we find the midpoints for $[W(0), W(T/2)]$ and $[W(T/2), W(T)]$, that is, $W(T/4)$ and $W(3T/4)$, respectively. Iterate for $m-2$ more times. This scheme increases the accuracy of quasi-Monte Carlo simulation to be introduced shortly by reducing its effective dimension [6, 142, 679, 876].

➢ **Programming Assignment 18.2.6** Implement the Brownian bridge approach to generate the sample path of geometric Brownian motion.

18.2.3 Variance-Reduction Techniques

The success of variance-reduction schemes depends critically on the particular problem of interest. Because it is usually impossible to know beforehand how great a reduction in variance may be realized, if at all, preliminary runs should be made to compare the results of a variance-reduction scheme with those from standard Monte Carlo simulation.

Antithetic Variates

Suppose we are interested in estimating $E[g(X_1, X_2, \ldots, X_n)]$, where $X_1, X_2, \ldots, X_n$ are independent random variables. Let Y_1 and Y_2 be random variables with the same distribution as $g(X_1, X_2, \ldots, X_n)$. Then

$$\mathrm{Var}\left[\frac{Y_1 + Y_2}{2}\right] = \frac{\mathrm{Var}[Y_1]}{2} + \frac{\mathrm{Cov}[Y_1, Y_2]}{2}.$$

Note that $\mathrm{Var}[Y_1]/2$ is the variance of the Monte Carlo method with two (independent) replications. The variance $\mathrm{Var}[(Y_1 + Y_2)/2]$ is smaller than $\mathrm{Var}[Y_1]/2$ when Y_1 and Y_2 are negatively correlated instead of being independent.

The **antithetic-variates** technique is based on the above observation. First, simulate $X_1, X_2, \ldots, X_n$ by means of the **inverse-transform technique**. That is, X_i is generated by $F_i^{-1}(U_i)$, where U_i is a random number uniformly distributed over $(0, 1)$ and F_i is the distribution function of X_i. Set

$$Y_1 \equiv g\big(F_1^{-1}(U_1), \ldots, F_n^{-1}(U_n)\big).$$

Because $1 - U$ is also uniform over $(0, 1)$ whenever U is, it follows that

$$Y_2 \equiv g\big(F_1^{-1}(1 - U_1), \ldots, F_n^{-1}(1 - U_n)\big)$$

has the same distribution as Y_1. When g is a monotone function, Y_1 and Y_2 are indeed negatively correlated and the antithetic-variates estimate,

$$\frac{g\big(F_1^{-1}(U_1), \ldots, F_n^{-1}(U_n)\big) + g\big(F_1^{-1}(1 - U_1), \ldots, F_n^{-1}(1 - U_n)\big)}{2},$$

has a lower variance than the Monte Carlo method with two replications [764]. Computation time is also saved because only n rather than $2n$ random numbers need to be generated, with each number used twice.

In general, for each simulated sample path X, we obtain a second one by reusing the random numbers on which the first path is based, yielding a second sample path Y. Two estimates are then obtained, one based on X and the other on Y. If a total of N independent sample paths are generated, the antithetic-variates estimator averages over $2N$ estimates.

EXAMPLE 18.2.2 Consider the Ito process $dX = a_t\,dt + b_t\sqrt{dt}\,\xi$. Let g be a function of n samples $X_1, X_2, \ldots, X_n$ on the sample path. We are interested in $E[\,g(X_1, X_2, \ldots, X_n)\,]$. Suppose that one simulation run has realizations $\xi_1, \xi_2, \ldots, \xi_n$ for the normally distributed fluctuation term ξ, generating samples $x_1, x_2, \ldots, x_n$. The estimate is then $g(\boldsymbol{x})$, where $\boldsymbol{x} \equiv (x_1, x_2 \ldots, x_n)$. Instead of sampling n more numbers from ξ for the second estimate, the antithetic-variates method computes $g(\boldsymbol{x}')$ from the sample path $\boldsymbol{x}' \equiv (x_1', x_2' \ldots, x_n')$ generated by $-\xi_1, -\xi_2, \ldots, -\xi_n$ and outputs $[\,g(\boldsymbol{x}) + g(\boldsymbol{x}')\,]/2$. Figure 18.7 implements the antithetic-variates method for average-rate options.

➤ **Exercise 18.2.7** Justify and extend the procedure in Example 18.2.2.

➢ **Programming Assignment 18.2.8** Implement the antithetic-variates method for arithmetic average-rate calls and puts. Compare it with the Monte Carlo method in Programming Assignment 18.2.3.

Antithetic variates for pricing average-rate calls on a non-dividend-paying stock:

```
input:   S, X, n, r, σ, τ, m;
real     P1, P2, C, M1, M2, a;
real     ξ( );    // ξ( ) ~ N(0, 1).
integer  i, j;
C := 0;
for (i = 1 to m) {
        P1 := S; P2 := S; M1 := S; M2 := S;
        for (j = 1 to n) {
                a := ξ( );
                P1 := P1 e^((r − σ^2/2)(τ/n) + σ√(τ/n) a);
                P2 := P2 e^((r − σ^2/2)(τ/n) − σ√(τ/n) a);
                M1 := M1 + P1;
                M2 := M2 + P2;
        }
        C := C + max(M1/(n + 1) − X, 0);
        C := C + max(M2/(n + 1) − X, 0);
}
return Ce^(−rτ)/(2m);
```

Figure 18.7: Antithetic-variates method for average-rate calls. P_1 keeps track of the first sample path, P_2 keeps track of the second sample path, m is the number of replications, and n is the number of periods.

Conditioning

Let X be a random variable whose expectation is to be estimated. There is another random variable Z such that the conditional expectation $E[X|Z=z]$ can be efficiently and precisely computed. We have $E[X] = E[E[X|Z]]$ by the law of iterated conditional expectations. Hence the random variable $E[X|Z]$ is also an unbiased estimator of μ. As $\text{Var}[E[X|Z]] \leq \text{Var}[X]$, $E[X|Z]$ indeed has a smaller variance than we obtain by observing X directly [764]. The computing procedure is to first obtain a random observation z on Z, then calculate $E[X|Z=z]$ as our estimate. There is no need to resort to simulation in computing $E[X|Z=z]$. The procedure can be repeated a few times to reduce the variance.

➢ **Programming Assignment 18.2.9** Apply conditioning to price European options when the stock price volatility is stochastic. The stock price and its volatility may be correlated.

Control Variates

The idea of **control variates** is to use the analytic solution of a similar yet simpler problem to improve the solution. Suppose we want to estimate $E[X]$ and there exists a random variable Y with a known mean $\mu \equiv E[Y]$. Then $W \equiv X + \beta(Y-\mu)$ can serve as a "controlled" estimator of $E[X]$ for any constant β that scales the deviation $Y-\mu$ to arrive at an adjustment for X. However β is chosen, W remains an unbiased estimator of $E[X]$. As

$$\text{Var}[W] = \text{Var}[X] + \beta^2 \text{Var}[Y] + 2\beta \text{Cov}[X, Y], \tag{18.5}$$

W is less variable than X if and only if

$$\beta^2 \text{Var}[Y] + 2\beta \text{Cov}[X, Y] < 0. \tag{18.6}$$

The success of the scheme clearly depends on both β and the choice of Y. For example, arithmetic average-rate options can be priced if Y is chosen to be the otherwise identical geometric average-rate option's price and $\beta = -1$ [548]. This approach is much more effective than the antithetic-variates method (see Fig. 18.8) [108].

Equation (18.5) is minimized when β equals $\beta^* \equiv -\text{Cov}[X, Y]/\text{Var}[Y]$, which was called beta earlier in Exercise 6.4.1. For this specific β,

$$\text{Var}[W] = \text{Var}[X] - \frac{\text{Cov}[X, Y]^2}{\text{Var}[Y]} = (1 - \rho_{X,Y}^2)\,\text{Var}[X],$$

where $\rho_{X,Y}$ is the correlation between X and Y. The stronger X and Y are correlated, the greater the reduction in variance. For example, if this correlation is nearly

Figure 18.8: Variance-reduction techniques for average-rate puts. An arithmetic average-rate put is priced with antithetic variates and control variates ($\beta = -1$). The parameters used for each set of data are $S = 50$, $\sigma = 0.2$, $r = 0.05$, $\tau = 1/3$, $X = 50$, $n = 50$, and $m = 10000$. Sample standard deviations of the computed values are in parentheses.

Antithetic variates		*Control variates*	
1.11039	1.11452	1.11815	1.11864
1.10952	1.10892	1.11788	1.11853
1.10476	1.10574	1.11856	1.11789
1.13225	1.10509	1.11852	1.11868
(0.009032505)		(0.000331789)	

perfect (± 1), we could control X almost exactly, eliminating practically all of its variance. Typically, neither Var[Y] nor Cov[X, Y] is known, unfortunately. Therefore we usually cannot obtain the maximum reduction in variance. One approach in practice is to guess at these values and hope that the resulting W does indeed have a smaller variance than X. A second possibility is to use the simulated data to estimate these quantities.

Observe that $-\beta^*$ has the same sign as the correlation between X and Y. Hence, if X and Y are positively correlated, $\beta^* < 0$; then X is adjusted downward whenever $Y > \mu$ and upward otherwise. The opposite is true when X and Y are negatively correlated, in which case $\beta^* > 0$.

➤ **Exercise 18.2.10** Pick $\beta = \pm 1$. The success of the scheme now depends solely on the choice of Y. Derive the conditions under which the variance is reduced.

➤ **Exercise 18.2.11** Why is it a mistake to use *independent* random numbers in generating X and Y?

➢ **Programming Assignment 18.2.12** Implement the control-variates method for arithmetic average-rate calls and puts.

Other Schemes

Two more schemes are briefly mentioned before this section closes. In **stratified sampling**, the support of the random variable being simulated is partitioned into a finite number of disjoint regions and a standard Monte Carlo simulation is performed in each region. When there is less variance within regions than across the regions, the sampling variance of the estimate will be reduced. **Importance sampling** samples more frequently in regions of the support where there is more variation.

➤ **Exercise 18.2.13** Suppose you are searching in set A for any element from set $B \subseteq A$. The Monte Carlo approach selects N elements randomly from A and checks if any one belongs to B. An alternative partitions the set A into m disjoint subsets $A_1, A_2, \ldots, A_m$ of equal size, picks N/m elements from each subset randomly, and checks if there is a hit. Prove that the second approach's probability of failure can never exceed that of the Monte Carlo approach.

18.3 Quasi-Monte Carlo Methods

The basic Monte Carlo method evaluates integration at randomly chosen points. There are several deficiencies with this paradigm. To start with, the error bound is probabilistic, not a concrete guarantee about the accuracy. The probabilistic error bound of $\sqrt{N}$ furthermore does not benefit from any additional regularity of the integrand function. Another fundamental difficulty stems from the requirement that the points be independent random samples. Random samples are wasteful because of clustering (see Fig. 18.9); indeed, Monte Carlo simulations with very small sample sizes cannot be trusted. Worse, truly random numbers do not exist on digital computers; in reality, **pseudorandom numbers** generated by completely *deterministic* means are used instead. Monte Carlo simulation exhibits a great sensitivity on the seed of the pseudorandom-number generator. The **low-discrepancy sequences**, also known as **quasi-random sequences**,[1] address the above-mentioned problems.

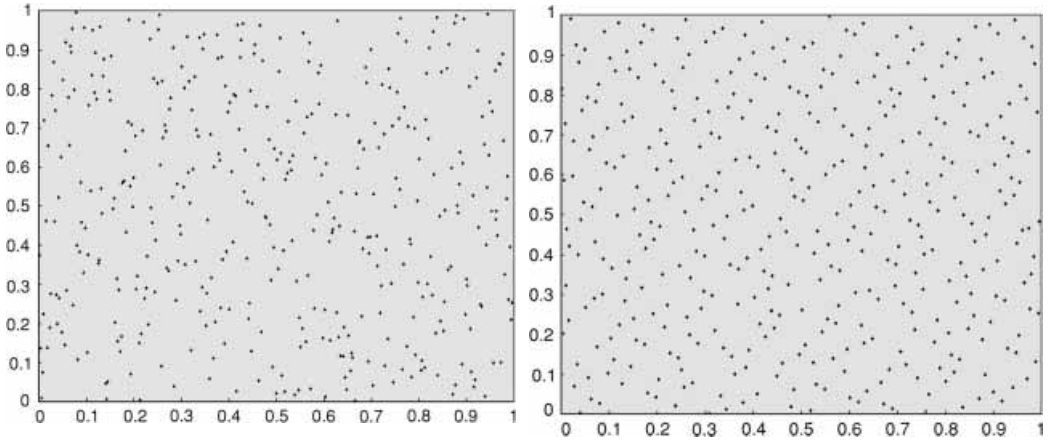

Figure 18.9: Random points (left) and Halton points (right) compared. The points are over the $[0,1] \times [0,1]$ rectangle [868]. Note the clustering of points on the left.

Proposed in the 1950s, the quasi-Monte Carlo method leaves nothing to chance [699]. In fact, it can be viewed as the deterministic version of the Monte Carlo method in that random samples are replaced with deterministic quasi-random points. If a smaller number of samples suffices as a result, efficiency has been gained. The main aim hence is to select deterministic points for which the deterministic error bound is smaller than Monte Carlo's probabilistic error bound.

The quasi-Monte Carlo method is not without limitations. Their theories are valid for integration problems, but may not be directly applicable to simulations because of the correlations between points in a quasi-random sequence. This problem can be overcome in many cases if the desired result is written as an integral. However, the resulting integral often has a very high dimension (e.g., 360 for a 30-year mortgage); in fact, the improved accuracy is generally lost for problems of high dimension or problems in which the integrand is not smooth. There is furthermore no theoretical basis for empirical estimates of their accuracy, a role played by the central limit theorem in the Monte Carlo method [142].

Although the results are somewhat mixed, the application of such methods in finance seems promising [108, 147]. A speed-up as high as 1,000 over the Monte Carlo method, for example, is reported in [711]. The success of the quasi-Monte Carlo method when compared with traditional variance-reduction techniques depends on the problem in question [715, 716]. For example, the antithetic-variates method outperforms the quasi-Monte Carlo method in bond pricing [679, 868].

18.3.1 The Halton Sequence

Every integer $k \geq 1$ can be represented uniquely as

$$k = a_0 + a_1 m + a_2 m^2 + \cdots + a_r m^r,$$

where $a_i \in [0, m-1]$ are integers and r is chosen such that $m^r \leq k < m^{r+1}$. The **radical inverse function in base** m is defined by

$$\phi_m(k) \equiv a_0 m^{-1} + a_1 m^{-2} + a_2 m^{-3} + \cdots + a_r m^{-(r+1)}.$$

In other words, a rational number in the interval $[0, 1)$ is generated by reflecting k in base m about the decimal point. For example, because $6 = 0 \times 2^0 + 1 \times 2 + 1 \times 2^2$,

$$\phi_2(6) = \frac{0}{2} + \frac{1}{4} + \frac{1}{8} = \frac{3}{8}.$$

The d-dimensional **Halton points** are defined as

$$z_k \equiv (\phi_{p_1}(k), \phi_{p_2}(k), \ldots, \phi_{p_d}(k)), \quad k \geq 1,$$

where $p_1, p_2, \ldots, p_d$ are the first d prime numbers.

Take $m = 2$ and $d = 1$. The Halton sequence in base two is

$$(0.1, 0.01, 0.11, 0.001, 0.101, 0.011, 0.111, 0.0001,$$
$$0.1001, 0.0101, 0.1101, 0.0011, 0.1011, 0.0111, 0.1111, \ldots),$$

which corresponds to

$$\left(\frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{8}, \frac{5}{8}, \frac{3}{8}, \frac{7}{8}, \frac{1}{16}, \frac{9}{16}, \frac{5}{16}, \frac{13}{16}, \frac{3}{16}, \frac{11}{16}, \frac{7}{16}, \frac{15}{16}, \ldots\right).$$

Although of no concern to numerical integration, the sequence above does not look useful for certain simulations. Write the numbers in decimal as 0.5, 0.25, 0.75, 0.125, 0.625, 0.375, 0.875, ... Were we to use them in simulating a symmetric random walk, it would alternate between right and left moves because the sequence consists eventually of pairs of monotonically increasing numbers: $0.5,\ 0.25 < 0.75,\ 0.125 < 0.625,\ 0.375 < 0.875, \ldots$. This phenomenon holds in general as a Halton sequence consists eventually of size-m subsequences of monotonically increasing numbers. For these reasons, the **scrambled Halton sequence** that permutes the a_is in $\phi_m(k)$ may be needed. Many quasi-Monte Carlo simulations cannot work without such manipulation [868]. Typically, the first 10 to 200 Halton points are discarded [368].

➤ **Exercise 18.3.1** Compute the first 10 one-dimensional Halton points in base 3.

➢ **Programming Assignment 18.3.2** Implement the two-dimensional Halton sequence, apply it to numerically evaluating $\int_0^1 x^2\,dx$, and compare it with Monte Carlo integration.

18.3.2 The Sobol' Sequence

Assume that $d = 1$ initially. A one-dimensional Sobol' sequence of length N is generated from $\omega \equiv \lceil \log_2 N \rceil$ **direction numbers** $0 < v_1, v_2, \ldots, v_\omega < 1$ through the bitwise **exclusive-**OR operation $\oplus$.[2] Each point has the form

$$b_1 v_1 \oplus b_2 v_2 \oplus \cdots \oplus b_\omega v_\omega,$$

where b_i are zero or one. Each point is therefore the result of exclusive-Oring a subset of the direction numbers, those v_i whose corresponding b_i is one. The sequence of $(b_1, b_2, \ldots, b_\omega)$ can either be $1, 2, \ldots, N$ in binary form or the **Gray code** [727].

The sequence of direction numbers is generated by a primitive polynomial with coefficients in the field Z_2 (whose elements are 0 and 1, and all operations are modulo 2). **Primitive polynomials** are, roughly speaking, polynomials that cannot be factored. Consider a primitive polynomial $P(x) \equiv x^n + a_1 x^{n-1} + \cdots + a_{n-1} x + 1$ of

degree n. The direction numbers are obtained from this recurrence formula:

$$v_i = a_1 v_{i-1} \oplus a_2 v_{i-2} \oplus \cdots \oplus a_{n-1} v_{i-n+1} \oplus v_{i-n} \oplus (v_{i-n}/2^n), \quad i > n.$$

To jump-start the recurrence, we need to specify $v_1, v_2, \ldots, v_n$. We do this by setting $v_i = m_i/2^i$ for n arbitrary odd integers $0 < m_i < 2^i$ $(1 \le i \le n)$. Finally, the Sobol' sequence $z_0, z_2, \ldots, z_{N-1}$ is obtained recursively by

$$\begin{cases} z_0 = 0 \\ z_{k+1} = z_k \oplus v_c \end{cases},$$

where c is the position of the rightmost zero bit in the binary representation of k.

EXAMPLE 18.3.1 Consider the primitive polynomial $x^6 + x^4 + x^3 + x + 1$. For $i > 7$, the equation for the direction numbers is

$$\begin{aligned} v_i &= 0 \cdot v_{i-1} \oplus 1 \cdot v_{i-2} \oplus 1 \cdot v_{i-3} \oplus 0 \cdot v_{i-4} \oplus 1 \cdot v_{i-5} \oplus v_{i-6} \oplus (v_{i-6}/2^6) \\ &= v_{i-2} \oplus v_{i-3} \oplus v_{i-5} \oplus v_{i-6} \oplus (v_{i-6}/2^6). \end{aligned}$$

Given the direction numbers, the Sobol' sequence can be easily generated. For instance, $z_{26} = z_{25} \oplus v_2$ because $25 = 11001$ (base two), whose rightmost 0 is at position two from the right.

The extension to a higher dimension is straightforward. Let $P_1, P_2, \ldots, P_d$ be primitive polynomials. Denote by z_k^i the sequence of one-dimensional Sobol' points generated by P_i. The sequence of d-dimensional Sobol' points is defined by

$$z_k \equiv \left(z_k^1, z_k^2, \ldots, z_k^d\right), \quad k = 0, 1, \ldots.$$

➢ **Programming Assignment 18.3.3** Implement the Sobol' sequence.

18.3.3 The Faure Sequence

The one-dimensional Faure sequence of quasi-random numbers coincides with the one-dimensional Halton sequence. To generate the d-dimensional Faure sequence, we proceed as follows. Let $p \ge 2$ be the smallest prime greater than or equal to d. The Faure sequence uses the same base p for each dimension. Denote the kth point by $z_k \equiv (c_1, c_2, \ldots, c_d)$. The first component c_1 is simply the one-dimensional Halton sequence $\phi_p(1), \phi_p(2), \ldots$. Inductively, if

$$c_{m-1} = a_0 p^{-1} + a_1 p^{-2} + \cdots + a_r p^{-(r+1)},$$

then

$$c_m = b_0 p^{-1} + b_1 p^{-2} + \cdots + b_r p^{-(r+1)},$$

where

$$b_j \equiv \sum_{i \ge j}^{r} \binom{i}{j} a_i \bmod p.$$

Note that $x \bmod p$ denotes the remainder of x divided by p. For instance, $24 \bmod 7 = 3$. The convention is $\binom{i}{0} = 1$. In practice, the sequence may start at $k = p^4$ instead of $k = 1$.

EXAMPLE 18.3.2 Take $d = 3$ and $p = 3$. Because the first component c_1 is merely the one-dimensional Halton sequence, it runs like

$$(0.1, 0.2, 0.01, 0.11, 0.21, 0.02, 0.12, 0.22, 0.001, \ldots) = \left(\frac{1}{3}, \frac{2}{3}, \frac{1}{9}, \frac{4}{9}, \frac{7}{9}, \frac{2}{9}, \frac{5}{9}, \frac{8}{9}, \frac{1}{27}, \cdots\right).$$

The second component c_2 is

$$\left(\frac{1}{3}, \frac{2}{3}, \frac{4}{9}, \frac{7}{9}, \frac{1}{9}, \frac{8}{9}, \frac{2}{9}, \frac{5}{9}, \frac{16}{27}, \cdots\right).$$

Take the fourth entry, $7/9$, which is the c_2 in z_4, as an example. Because the corresponding number in the first component is 0.11, we have $a_0 = 1$, $a_1 = 1$, and $r = 2$. The number is thus calculated by

$$b_0 = \binom{0}{0} a_0 + \binom{1}{0} a_1 = 1 + 1 = 2 \bmod 3,$$

$$b_1 = \binom{1}{1} a_1 = 1 \bmod 3,$$

$$c_2 = b_0 3^{-1} + b_1 3^{-2} = \frac{2}{3} + \frac{1}{9} = \frac{7}{9}.$$

The sequence for the third component is $(\frac{1}{3}, \frac{2}{3}, \frac{7}{9}, \frac{1}{9}, \frac{4}{9}, \frac{5}{9}, \frac{8}{9}, \frac{2}{9}, \frac{13}{27}, \ldots)$. The combined three-dimensional sequence is therefore $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}), (\frac{2}{3}, \frac{2}{3}, \frac{2}{3}), (\frac{1}{9}, \frac{4}{9}, \frac{7}{9}), \ldots$.

➤ **Exercise 18.3.4** Verify the sequence for the third component in Example 18.3.2.

➢ **Programming Assignment 18.3.5** Implement the Faure sequence. Pay attention to evaluating $\binom{i}{j} \bmod p$ efficiently.

➢ **Programming Assignment 18.3.6** Price European options with quasi-random sequences. The computational framework is identical to the Monte Carlo method except that random numbers are replaced with quasi-random numbers with n-dimensional sequences for problems with n time periods.

Additional Reading

Subsection 18.1.3 followed [229]. Finite-difference methods for the Black–Scholes differential equation for multivariate derivatives can be found in [215]. Consult [15, 381, 391, 810, 883] for more information on solving partial differential equations, [706, 707] for algorithms on parallel computers, and [861] for *Mathematica* programs. The implicit method, the explicit method, and the standard binomial tree algorithm are compared in [229, 383, 472, 897].

The term "Monte Carlo simulation" was invented by von Neumann (1903–1957) and Ulam (1909–1984) when they worked on the Manhattan Project [570]. The first paper on the subject was by Metropolis and Ulam in 1949 [699]. Complete treatments of Monte Carlo simulations can be found in [353, 773]. Consult [560, 584, 621, 721, 727] for pseudorandom-number generators. On the topic of Monte Carlo option pricing, see [313] for fast Monte Carlo path-dependent derivatives pricing, [53, 108, 129, 130, 160, 730] for the Monte Carlo pricing of American options, [106, 472] for the

control-variates approach, [214, 520, 579, 876] for handling stochastic volatility, [241] for pricing average-rate options by use of conditioning, [366] for comparing numerical integrations, analytical approximations, and control variates in average-rate option pricing, and [20, 108, 128] for general treatments of Monte Carlo simulations in computing sensitivities. Consult [6, 108] for surveys on option pricing by use of Monte Carlo simulation, quasi-Monte Carlo methods, and variance reductions. Subsection 18.2.3 drew on [147, 584, 727, 764]. One suggestion for speeding up Monte Carlo simulation is by limiting the sample space to a finite size and then conducting an exhaustive sampling [509]. In estimating sensitivities such as delta by $E[\,P(S+\epsilon)-P(S-\epsilon)\,]/(2\epsilon)$, $P(S+\epsilon)$ and $P(S-\epsilon)$ should use *common* random numbers to lower the variance.

See [142, 197, 530, 532, 711, 715, 716] for case studies on quasi-Monte Carlo methods, [368, 530, 711, 715, 716, 876] for evaluations of various quasi-random sequences, and [108, 337, 699] for their mathematical foundations. Biology-inspired approaches such as **artificial neural networks** [486, 612, 887] and **genetic algorithms** are other interesting approaches.

NOTES

1. This term is, strictly speaking, misleading as there is nothing random about such sequences.
2. The exclusive-OR operation takes two input bits. It returns 1 if the bits are different and 0 if they are identical. The operation when applied to two bit streams of the same length computes the exclusive-OR of bits at the same position. For example, $10110 \oplus 01100 = 11010$