Chapter 2 - Monte Carlo Simulation

Modelling Derivatives in C++

1 Monte Carlo

The following is an important (discrete) formula:

1. $S_{i+1} = S_i \exp\left\{\mu\Delta t + \sigma\sqrt{\Delta t}\epsilon_{i+1}\right\}$

2 Generating sample paths and normal deviates

Some asymptotic results (fast, but not accurate):

1. $\frac{\sum_{i=1}^{n} U_i - (n/2)}{\sqrt{n/12}} \to N(0,1)$, setting $n = 12$ gives:
2. $\sum_{i=1}^{12} U_i - 6 \xrightarrow{D} N(0,1)$

Box-Muller algorithm:

1. Generate U_1 and U_2 from two independent uniform (0, 1) distributions.
2. Set $N_1 = \sqrt{-2logU_1}cos(2\pi U_2)$ and $N_2 = \sqrt{-2logU_2}sin(2\pi U_1)$, OR:
3. Set $V_1 = 2U_1 - 1$ and $V_2 = 2U_2 - 1$.
4. Compute $W = V_1^2 + V_2^2$
5. if $W > 1$, return to step 1. Otherwise, set $N_1 = \sqrt{\frac{-2logW}{W}}V_1$ and $N_2 = \sqrt{\frac{-2logW}{W}}V_2$

3 Generating correlated normal random variables

Given a level of correlation between normal random variables X_1 and X_2, ρ, one can express:

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} + \begin{bmatrix} \sigma_1 & 0 \\ \rho\sigma_2 & \sqrt{1-\rho^2}\sigma_2 \end{bmatrix} \begin{bmatrix} N_1 \\ N_2 \end{bmatrix}$$

for $X_1, ..., X_n$, we can get the lower triangular matrix with Cholesky's decomposition.

Another approach is to utilise PCA, supposed that we have $z_1, ..., z_n$ correlated Brownian motion paths with a correlation matrix Σ. If one can decompose Σ into its constituent eigenvector/eigenvalue components represented by $\Sigma = \Gamma\Lambda\Gamma^{-1}$:

$$\Gamma = \begin{bmatrix} v_{11} & \dots & v_{1n} \\ \vdots & \ddots & \vdots \\ v_{n1} & \dots & v_{nn} \end{bmatrix}, \text{ and } \Lambda = \begin{bmatrix} \lambda_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_n \end{bmatrix}$$

The diffusion processes that underlie the correlated Brownian motion paths are hence represented as follows:

$$\begin{aligned} dz_1 &= v_{11}\sqrt{\lambda_1}dw_1 + ... + v_{1n}\sqrt{\lambda_n}dw_n \\ \dots \\ dz_n &= v_{n1}\sqrt{\lambda_1}dw_1 + ... + v_{nn}\sqrt{\lambda_n}dw_n \end{aligned}$$

4 Quasi-Random Sequences

Not too relevant - as quasi-random sequences can be easily implemented with `scipy`.

5 Variance reduction and control variate techniques

Two procedures are demonstrated here:

1. Using antithetic variates:
    (a) Produce F_k and $\hat{F}_k$ from ϵ_k and $-\epsilon_k$ respectively. Calculate $\tilde{F}_k = \frac{1}{2}(F_k + \hat{F}_k)$.
2. Using control variates:
    (a) Random variables with known means that's correlated with the variable we are trying to estimate.
    (b) Come up with the price of a complex security using: $f_c = f_s + (f_c - f_s)$
    (c) Simulate $\epsilon^* = (f_c - f_s)$ to calculate f_c^*, which approximates f_c
    (d) e.g., we can calculate the price of an arithmetic Asian option as: $f_a^* = f_c^* + \epsilon^*$

6 Monte Carlo implementation

For a sample simulate of a European call, we can use Monte Carlo to obtain:

$$\hat{C} = e^{-rT}\frac{1}{M}\sum_{j=1}^{M} max(0, S_j(T) - K)$$

But this is not too important, since Monte Carlo truly shines in a path-dependent scenario, in which S paths are computed using:

$$S_{t+\Delta t} = S_t exp\left((r - q - \frac{1}{2}\sigma^2)\Delta t + \sigma(\epsilon\sqrt{\Delta t}\right)$$

7 Hedge-control variates

Delta and gamma hedging can be used to control the volatility of the portfolio estimates. In this case - the Greeks can be used as secondary information to control the variance of the estimated price.

The following sets of equations show, with delta and gamma control variates cv_1 and cv_2, how an enhanced payoff estimate is calculated with linear regression:

1. $cv_1 = \sum_{i=0}^{N-1} \frac{\partial C_{t_i}}{\partial S}(S_{t_i+1} - E(S_{t_i}))e^{r(T-t_{i+1})}$
2. $cv_2 = \sum_{i=0}^{N-1} \frac{\partial^2 C_{t_i}}{\partial S^2}(\Delta S_{t_i}^2 - E(\Delta S_{t_i}^2))e^{r(T-t_{i+1})}$
3. $C_T = C_0 e^{rT} + \sum_{k=1}^{n} \beta_k cv_k + \eta$

Monte Carlo can be used to value spread options and basket options easily. Other methods include n-variable binomial method, fast Fourier transforms, Gaussian quadratures.

8 Path dependent valuation

Algorithm to calculate the price of a path-dependent option:

1. Divide the path into N time steps, and simulate M sample paths of the underlying's diffusion process.
2. Calculate the terminal payoff for each path - the payoff should depend on the path of the option.
3. Discount by the risk-free rate.
4. Obtain the average of each path - as the price of the path-dependent option
5. A geometric average option is a good control variate of the arithmetic average option, so that can be supplemented.

9 Brownian bridge technique

Useful for stress testing, the Brownian bridge technique interpolates paths conditional on a specific point being reached at a specific time. The formulae for the original Black-Scholes solution to the underlying can be replaced with a Brownian bridge provided as follows:

$$\begin{aligned}
S_t &= S_0 exp((\mu - \frac{\sigma^2}{2})t + \sigma B_t) \\
&= S_0 exp(log\frac{S_T}{S_0}\frac{t}{T} + \sigma(Z^*(t) - \frac{t}{T}Z^*(T))) \\
Z*(t_{i+1}) &= Z^*(t_i) + \epsilon\sqrt{t_{i+1} - t_i}
\end{aligned}$$

In this scenario, only the volatility needs to be estimated, the advantage comes at the cost of requiring to estimate the final probability distribution of the price at T.

10 Jump-diffusion and constant variance elasticity diffusion model

We can incorporate jumps drawn from a Poisson process (which incorporates information such as defaults or stock price jumps) in a diffusion process, which takes this form:

$$\frac{dS}{S} = (\mu - \lambda\kappa)dt + \sigma dz + dq$$

Jump processes can be positive or negative, and yields fatter tails compared to log-normal distributions. It is easy to obtain Poisson random variables using uniform distributions with space [0, 1], the arrival times can be calculated using:

$$T_n = -\frac{1}{\lambda}\sum_{i=1}^{n} logU_i$$

With jumps, the closed-form solution of a European call option actually exists. The log of the size of the jump is consistently normal with a standard deviation of δ:

$$C = \sum_{n=0}^{\infty} \frac{e^{-\bar{\lambda}}(\bar{\lambda}t)^n}{n!} C_n(S, K, t, \bar{r}, \sqrt{\sigma^2 + \delta^2(n/t)})$$

Where $\bar{\lambda} = \lambda(1+\kappa)$, and $\bar{r} = r - \lambda\kappa + log(1+\kappa)(n/t)$

Another popular model is the constant elasticity of variance (CEV) model. The CEV model captures financial leverage. The underlying has a formula:

$$dS = \mu S dt + \sigma S^{1-\alpha} dz$$

And the reason why it is called the CEV model, is because its variance's elasticity, $\frac{\partial\sigma}{\partial S}\frac{S}{\sigma} = -\alpha$, a constant.

11 Object-oriented Monte Carlo approach

The strategic design of the Monte Carlo approach (similar to Joshi's approach), can be thought to comprise the following classes:

1. Monte Carlo method class - to call the Monte Carlo method
2. Path generator class - to generate all paths
3. Path class - methods to handle computations of drift and diffusion terms
4. Monte Carlo pricer class - price derivatives with Monte Carlo along each path
5. Sample struct - sample size is taken from here
6. European path pricer class - to price European options along each path
7. European Monte Carlo class - to do the actual pricing using the MC method
8. Statistics class - aggregates results from Monte Carlo