University of Oxford

Monte Carlo

MSc in Mathematical Finance
January 14, 2019

Comparison of performance of Quasi Monte Carlo and standard Monte Carlo methods

Mean and Standard deviation for a European basket call option

	Mean	Standard deviation
QMC with Cholesky, Sobol scrambling with Brownian Bridge	0.694	0.000181
QMC with Cholesky, Sobol scrambling without Brownian Bridge	0.694	0.000394
Standard MC with Cholesky	0.697	0.00192
QMC with PCA, Sobol scrambling with Brownian Bridge	0.694	0.000347
QMC with PCA, Sobol scrambling without Brownian Bridge	0.694	0.000421
Standard MC with PCA	0.695	0.00231

Table 1: Mean and standard deviation values for the European basket call option

Mean and Standard deviation for a European Asian strike basket call option

An Asian strike basket call option has the same payoff as the Eurooean call basket option except the fixed strike K is replaced by the average A_T. Setting

$$\overline{S}(T) = \frac{1}{4}\sum_{j=1} S_j(T)$$

The payoff of an Asian strike basket call option is

$$max(\overline{S}(T) - A_T, 0)$$

where

$$A_T = \frac{1}{T}\int_0^T \overline{S}(t)dt$$

To compute the integral A_T, it needs to be numerically approximated:

$$A_T = \frac{1}{T}\int_0^T \overline{S}(t)dt = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{4}(S_{t_i}^1 + S_{t_i}^2 + S_{t_i}^3 + S_{t_i}^4)$$

- We have 2^{18} paths, split into groups of 2^6, each with it's own randomisation. Generate an array S of dimensions 4×2^{12}, all initialised with the current stock price $S_j(0)$ for j=1,2,3,4.

$$\begin{bmatrix} S_1(0) & \cdots & S_1(0) \\ S_2(0) & \cdots & S_2(0) \\ S_3(0) & \cdots & S_3(0) \\ S_4(0) & \cdots & S_4(0) \end{bmatrix}$$

- Initialise an array of zeros, $\overline{S}_j$, of dimensions 1×2^{12} for the average price of each stock in the basket. Over time, this vector should be updated to have the average value of the basket over each randomised path as shown:

$$\begin{bmatrix} \overline{S}_1 & \cdots & \overline{S}_M \end{bmatrix}$$

- At each timestep, calculate $\overline{S}(t)$ as shown above for each path and get a vector of dimensions 1×2^{12}.
- Divide this by the number of timesteps and increment the original vector for the average stock price by the one for the current timestep.
- As shown above, loop over all the timesteps to approximate the integral.

	Mean	Standard deviation
QMC with Cholesky, Sobol scrambling with Brownian Bridge	0.393	0.00017
QMC with Cholesky, Sobol scrambling without Brownian Bridge	0.393	0.000289
Standard MC with Cholesky	0.390	0.00102
QMC with PCA, Sobol scrambling with Brownian Bridge	0.393	0.000258
QMC with PCA, Sobol scrambling without Brownian Bridge	0.393	0.000233
Standard MC with PCA	0.392	0.00102

Table 2: Mean and standard deviation values for Asian strike basket call option

Mean and Standard deviation for a European basket lookback put option

The payoff of an lookback strike basket put option is

$$max(K - \min_{0<=t<=T} \overline{S}(t), 0)$$

To compute the minimum in the lookback put payoff:

- We have 2^{18} paths, split into groups of 2^6, each with it's own randomisation. Generate an array S of dimensions 4×2^{12}, all initialised with the current stock price $S_j(0)$ for j=1,2,3,4.

$$\begin{bmatrix} S_1(0) & \cdots & S_1(0) \\ S_2(0) & \cdots & S_2(0) \\ S_3(0) & \cdots & S_3(0) \\ S_4(0) & \cdots & S_4(0) \end{bmatrix}$$

- Initialise a vector of minimum values set to the initial stock price for each. The dimensions are 1×2^{12} as each element is the minimum value of the basket at each randomised path.

$$\begin{bmatrix} \overline{S}_1 & \cdots & \overline{S}_M \end{bmatrix}$$

- At each timestep, calculate $\overline{S}(t)$ using S and the formula shown above. Update the minimum vector if needed.
- Loop over N timesteps and the value we get finally is the approximation for $\min_{0<=t<=T} \overline{S}(t)$.

According to the results, the lookback option has the highest average value, with the vanilla option coming in second and the Asian being the lowest.
The European and the Asian options are call options while the lookback is a put. If we compare

	Mean	Standard deviation
QMC with Cholesky, Sobol scrambling with Brownian Bridge	1.10	0.000385
QMC with Cholesky, Sobol scrambling without Brownian Bridge	1.099	0.000377
Standard MC with Cholesky	1.103	0.00167
QMC with PCA, Sobol scrambling with Brownian Bridge	1.099	0.000377
QMC with PCA, Sobol scrambling without Brownian Bridge	1.099	0.00038
Standard MC with PCA	1.10	0.00151

Table 3: Mean and standard deviation values for the lookback strike basket put option

the values of the two calls first, we can see that the European is more expensive than the Asian. A path-dependent option is one whose payoff depends on the stock price at expiry as well as the overall path of the price process over time. The vanilla call option is not path-dependent and the payoff is determined by the stock price of the basket at expiry. The Asian option is path-dependent and it's payoff depends on the average stock price over time till expiry. By considering the average, the Asian option is safer as it protects the holder from volatile stock price moves in the market. Thus the Asian is less expensive than it's vanilla counterpart as the volatility of the average A_T is less than that of the stock prices S_T.
The lookback option is also path-dependent in that its payoff depends on the minimum stock price over a period of time till expiry. Thus the lookback option allows the holder to exercise the option at the best possible price which makes it more valuable than the Asian and European options.
Thus the results are as expected.

1 Appendix

```
import numpy as np
from numpy.random import randn
from numpy import ones, zeros, eye, sqrt, exp, log2, diag, reshape
from numpy.linalg import cholesky, eig
from scipy.stats import norm
from bb import bb
from sobol import sobol, scramble

K = 20.
S0 = 20.
T = 0.5
r = 0.01
sigma = 0.2
rho = 0.1

for option in range(1,4):
   print('---------------------------')
   if  option == 1:
      print('Basket call option\n')
   elif option == 2:
      print('Basket Asian strike call option\n')
   elif option == 3:
      print('Basket lookback put option\n')
   print('---------------------------')

   for opass in [1,2]:
     sigma_matrix = eye(4) + rho*(ones((4,4))-eye(4))

      if opass == 1:
         print('Cholesky factorisation of correlation:\n')
         L = cholesky(sigma_matrix)
      else:
         print('PCA factorisation of correlation:\n')
         D,V = eig(sigma_matrix)
         L = V.dot(diag(sqrt(D)))

      # three inner passes for Cholesky and Brownian Bridge factorisations
      # of covariance matrix in time, and use of plain MC

      for ipass in range(1,4):
         N  = 64          # number of timesteps
         M2 = 64          # number of randomisations
         M  = int(2**18/M2) # number of paths in each "family"

         unscrambled = sobol(m=int(log2(M)), s=4*N, scramble=False)

         h  = T/N
```

```
sum1 = 0.
sum2 = 0.

if ipass == 1:
    print('Sobol with BB')
elif ipass == 2:
    print('Sobol without BB')
else:
    print('plain MC')

for m in range(1,M2+1):
    if ipass == 1:
        # Sobol points with Brownian Bridge construction of Brownian increments
        U  = scramble(unscrambled).T # generate set of M Sobol points
        Z  = norm.ppf(U)       # inverts Normal cum. fn.
        dW = bb(Z,T,L)

    elif ipass == 2:
        # Sobol points without Brownian Bridge construction
        U  = scramble(unscrambled).T # generate set of M Sobol points
        Z  = norm.ppf(U)
        Z  = reshape(Z, (4,N*M), order='F')
        dW = sqrt(h)*L.dot(Z)
        dW = reshape(dW, (4*N,M), order='F')

    else:
        # standard random number generation
        dW = sqrt(h)*L.dot(randn(4,N*M))
        dW = reshape(dW, (4*N,M), order='F')

    S    = S0*ones((4,M))
    Save = zeros((M,))
    Smin = S0*ones((M,))

    for n in range(1,N+1):
        S    = S*(1+r*h+sigma*dW[4*n-4:4*n,:])
        Save = Save + 0.25*np.sum(S,0)/N
        Smin = np.minimum(Smin,0.25*np.sum(S,0))

    S = 0.25*np.sum(S,0)
    if   option == 1:
        P = exp(-r*T)*np.maximum(S-K,0)
    elif option == 2:
        P = exp(-r*T)*np.maximum(S-Save,0)
    else:
      P = exp(-r*T)*np.maximum(K-Smin,0)

    P = np.sum(P)/M
    sum1 = sum1 + np.sum(P)
    sum2 = sum2 + np.sum(P**2)

V  = sum1/M2
sd = sqrt((sum2/M2 - V**2)/(M2-1))
if ipass == 3:
    print(' MC_val     = %f \n' % V)
    print(' MC_std_dev = %f \n\n' % sd)
else:
    print(' QMC_val     = %f \n' % V)
    print(' QMC_std_dev = %f \n\n' % sd)
```