M.Phil. in Mathematical Finance

Numerical Methods in Finance 2020

Assignment

Instructions

- You are required to hand in four Matlab function/scripts, named `gendirnums`, `Sobol`, `Basket` and `Hedge` corresponding to the four questions below. Please use these exact names. Please do not hand in any other files, include all your code in these four files — if you wish to use other functions, use subfunctions.
- Submission is by email to `tom@analytical.co.za` or `Thomas.McWalter@gmail.com`.
- The due date for the assignment is 12 noon on Monday the 13th of July — marks will be deducted for late submission (5% for every half hour, or part thereof, that the assignment is late). The arrival time stamp of the email will be used to decide if submission is late or not. Absolutely no excuses accepted — so, rather submit early.
- Absolutely no coding together or copying is allowed. If such activity is detected, all individuals involved will receive a mark of zero and disciplinary action will be taken.
- Marks (44%) are allocated for speed and efficiency — The run-times of the code in Question 4 (which produce the correct results) will be ranked. The mark (out of 44) assigned for speed will be calculated as $2(23-n)$ where n is the ranking (e.g. if the code was ranked 3rd fastest then 40 marks will be allocated out of 44).

Sobol' Sequences

In this assignment, you will implement code to generate and use Sobol' sequences.

1. Write a function called `gendirnums.m` that takes as input a non-negative integer p, a vector of initial direction numbers m_i and an integer k. When converted to binary, the input p represents the coefficients of a primitive polynomial (for example, $p = 13 = 1101_2$ represents the polynomial $x^3 + x^2 + 1$). The function should return a list of direction numbers of length k according to the algorithm found in Definition 5.9 in the notes. If the number of initial direction numbers d is too small for the primitive polynomial supplied, then issue the error message `'too few initial direction numbers for corresponding primitive polynomial'`. If you have implemented things correctly, issuing the command `gendirnums(13,[1 3 5],10)` should generate the vector $\begin{bmatrix}1 & 3 & 5 & 3 & 29 & 23 & 53 & 159 & 401 & 703\end{bmatrix}$.
2. Write a function called `Sobol.m` that takes as input a vector of direction numbers and n a positive integer. The function should return the first n Sobol' numbers by implementing the recurrence relation in Definition 5.11. If the number of direction numbers supplied is not large enough to generate n Sobol' numbers, issue the error `'n too large, either decrease n or provide more direction numbers'`. Note that the command `bitxor` does not work for fractions. Thus, in order to implement the recurrence relation, you will need to modify it by implementing a binary shift as described in class. If you have implemented the function correctly, the command `plot(Sobol(gendirnums(13,[1 3 5],10),1000),Sobol(gendirnums(19,[1 1 3 13],10),1000),'.');` will generate the output of Figure 5.5 in the notes.
3. Consider a European basket call option, with payoff function

$$f\left(S_T^{(1)}, S_T^{(2)}\right) = \max\left(S_T^{(1)} + S_T^{(2)} - K, 0\right),$$

strike $K = 100$ and maturity $T = 2$ years, on two correlated stocks with the following parameters:

$$\begin{bmatrix} S_0^{(1)} \\ S_0^{(2)} \end{bmatrix} = \begin{bmatrix} 35 \\ 65 \end{bmatrix}, \qquad \begin{bmatrix} \sigma^{(1)} \\ \sigma^{(2)} \end{bmatrix} = \begin{bmatrix} 0.25 \\ 0.20 \end{bmatrix}, \qquad \Sigma = \begin{bmatrix} 1 & 0.6 \\ 0.6 & 1 \end{bmatrix}, \qquad r = 10\%,$$

where Σ is a correlation matrix. Write a script called `Basket.m` to implement the following:

a. Using the following primitive polynomial–direction number combinations

$$\begin{aligned} 7 \qquad & \begin{bmatrix} 1 & 3 \end{bmatrix}, \\ 11 \qquad & \begin{bmatrix} 1 & 1 & 5 \end{bmatrix}, \end{aligned}$$

generate a 2 dimensional Sobol' sequence of length 50 001. Exclude the zero vector at the start of the sequence to get a sequence of length 50 000.

In a single figure, side by side, plot the first one thousand points of each of the 2 dimensional Sobol' subsequences and plot the corresponding normal numbers generated by performing the inverse transform method.

b. Compute and plot Quasi-Monte Carlo estimates for the option as a function of sample sizes in the range $n = 1\,000, 2\,000, \ldots, 50\,000$.

c. Compute and plot crude Monte Carlo estimates for the option price as a function of sample size in the same figure as used in Part b. Plot three-standard-deviation error bounds around these estimates (you can use the ***best estimate*** generated by the Quasi-Monte Carlo method as an estimate for the real value of the option).

For debugging purposes, initialize your seed at the beginning of your program with the Matlab command `rng(0);`, and, in the loop that cycles through sample size, generate the normal random numbers with the command `randn(2,n);`.

d. Compute pathwise derivative estimates for the delta of the option with respect to $S^{(1)}$ for both the crude and Sobol' sequences. In a separate figure, plot these estimates and a three-standard deviation error bound around the crude Monte Carlo estimates against sample size (use the ***best estimate*** generated by the Quasi-Monte Carlo method as an estimate for the real value of the delta).

4. Finally, you will investigate the efficacy of discrete-time delta hedging. Consider the situation where you have sold a basket option on the two underlying stocks and must now hedge the risk until maturity. The process proceeds as follows:

- Firstly, compute the price, p, of the option at inception ($t = 0$). Assume that you have received this price from the counterparty you sold the option to. The initial delta-neutral hedge, in each of the stocks, is set up by purchasing a holding $h_0^{(1)} = \Delta\big(S_0^{(1)}\big)$ in stock $S^{(1)}$ and $h_0^{(2)} = \Delta\big(S_0^{(2)}\big)$ in stock $S^{(2)}$ and depositing the remainder of the purchase price in the bank account $b_0 = p - h_0^{(1)} S_0^{(1)} - h_0^{(2)} S_0^{(2)}$.
- Now, given that the (real-world, not risk-neutral) stock price paths $S_i^{(1)}$ and $S_i^{(2)}$ are observed and tradable at times $t_i = i\Delta t$ for $1 \le i < N$, with $\Delta t = T/N$, the hedge portfolio is rebalanced by computing the new delta-neutral holdings $h_i^{(1)} = \Delta\big(S_i^{(1)}\big)$ and $h_i^{(2)} = \Delta\big(S_i^{(2)}\big)$, with the new holding in the bank account given as

$$b_i = b_{i-1} e^{r\Delta t} - \big(h_i^{(1)} - h_{i-1}^{(1)}\big) S_i^{(1)} - \big(h_i^{(2)} - h_{i-1}^{(2)}\big) S_i^{(2)}.$$

  It is clear, from the above formula, that the strategy we are implementing is self-financing with the bank account growing at interest rate r. We have assumed that there are no market frictions (transaction costs, taxes, etc.)
- Finally, at the maturity of the option, the hedge portfolio profit/loss (P&L) may be computed as

$$b_N = b_{N-1} e^{r\Delta t} + h_{N-1}^{(1)} S_N^{(1)} + h_{N-1}^{(2)} S_N^{(2)} - f\big(S_N^{(1)}, S_N^{(2)}\big).$$

Write a function called `Hedge.m` that takes no inputs and returns the price and a vector of hedge portfolio P&L values for a European basket call option, in that order (i.e., `function [Price, PnL]=Hedge`). Use the payoff function and parameters given in Question 3 with the exception of the maturity, which should now be set to $T = 1/12$.

a. Compute the price of the option using a Sobol' sequence of length 50 000 generated in the same manner as Question 3a.

b. Generate the $n = 10\,000$ P&L values using the method outlined above with the following additional parameters:

$$N = 300, \qquad \begin{bmatrix} \mu^{(1)} \\ \mu^{(2)} \end{bmatrix} = \begin{bmatrix} 0.12 \\ 0.15 \end{bmatrix}.$$

Note that each pair of stock paths, used to compute a single P&L, should be simulated using standard normal random numbers generated with the Matlab command `randn(2,N);`. You should compute $\Delta\big(S_i^{(1)}\big)$ and $\Delta\big(S_i^{(2)}\big)$, for $0 \le i < N$, using the pathwise method and a quasi-Monte Carlo sample consisting of the first 5 000 elements of the 2 dimensional Sobol' sequence used in Part 4a. Do not set the random seed at any time, this will be done before calling your function.

In order to test your function, you can perform the following commands:

```
[~, PnL] = Hedge;
histogram(PnL,50);
xlim([-0.8 0.8]);
title('Hedging PnL');
```

The output should look something like Figure 1, although it won't be exactly the same because the seed is not set.

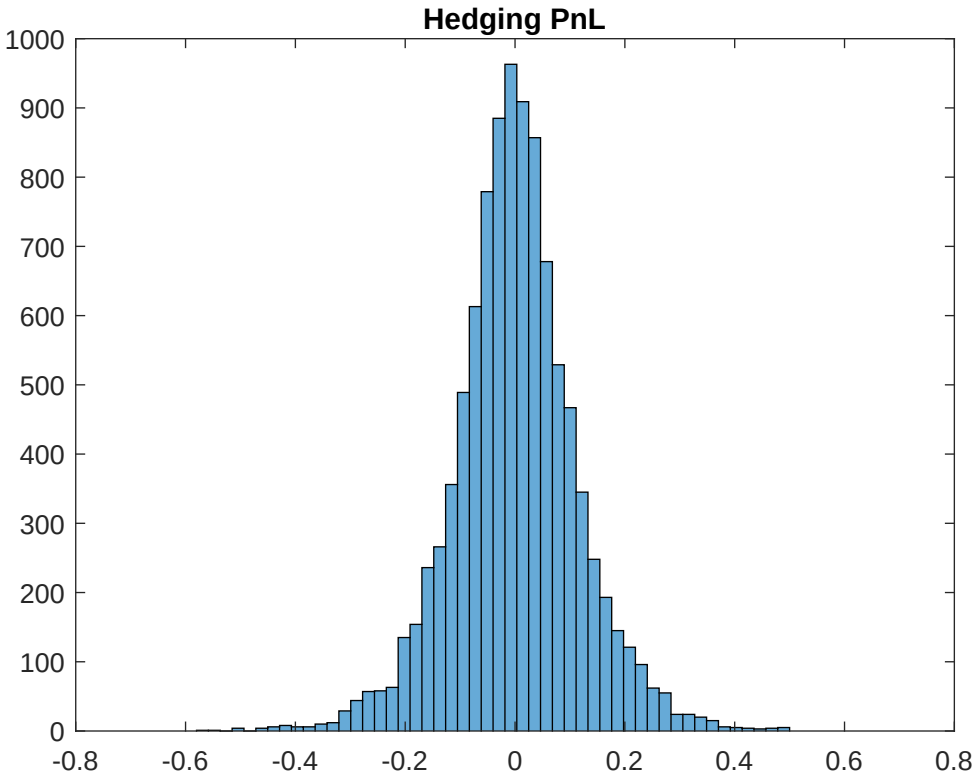

Figure 1: Hedging profit and loss histogram for a basket option.