Prof. Chow Ying-Foon, Ph.D.
Room 1215, Cheng Yu Tung Building
Department of Finance
The Chinese University of Hong Kong

2013–2014 Second Term
Phone: (+852) 3943 7638
Fax: (+852) 2603 6586
Email: yfchow@baf.cuhk.edu.hk

FIN 4140 A Computational Finance
Course Project Guidelines
Due Date: April 30, 2014

Instructions

You do not have to use Microsoft Excel or VBA for the assignment since all computations can also be done using programming environments such as C, C++, EViews, GAUSS, MATLAB, Octave, Ox, R, SAS, or S-PLUS. Please do not turn in the project in reams of unformatted computer output and without comments! Make little tables of the numbers that matter, copy and paste all results and graphs into a document prepared by typesetting system such Microsoft Word or LaTeX while you work, and add any comments and answer all questions in this document. Your project will be checked for correctness, completeness, and clearness of the presentation. I will also check your program and run it myself. If necessary, I will ask you to give me a demo. In addition, I will probably ask you further questions regarding your work. In this case, your response will be also evaluated.

1 Geometric Brownian Motion Model (10%)

- The purpose of this part is to well understand the Geometric Brownian Motion (GBM) model. As we know, the commonly used constant GBM model for stock prices is given by
$$\frac{dS_t}{S_t} = \mu dt + \sigma dW_t \quad \text{or equivalently} \quad dS_t = S_t\left(\mu dt + \sigma dW_t\right) \tag{1}$$
where S_t is the stock price at time t, μ is the expected rate of return of the stock, σ is the volatility of the stock, and W_t is a standard Brownian motion (or Wiener process).

- Its discrete version, which is practically more convenient to use than the continuous model, is written as
$$\Delta S_t = S_t\left(\mu\Delta t + \sigma\sqrt{\Delta t}\varepsilon_t\right) \tag{2}$$
where Δt is the length of the small time interval, ΔS_t is the stock price change in the time interval (i.e., $\Delta S_t = S_{t+\Delta t} - S_t$), ε_t is a standard Gaussian random variable (i.e., $\varepsilon_t \sim \mathcal{N}(0,1)$). This discretized GBM model can be used to generate future stock prices by Monte Carlo simulation once the two parameters μ and σ are known.

- This part consists of two parts: (1) fit the model using market data; (2) generate future prices by Monte Carlo simulation.

1.1 Use Market Data to Estimate Model Parameters μ and σ

1. Choose a specific stock or index you want to study. For instance, MSFT, IBM, WMT, KO, etc., for individual stock; Dow, S&P500, Nasdaq, HSI for index.
2. Choose a period of one year (or more) from the recent past and download the historical prices of the stock for the chosen period. For example, 01/01/2013 – 12/31/2013, 04/01/2013 – 03/31/2014, etc.
3. Display the prices on a graph. Use the horizontal axis for time, vertical axis for price, and put the asset name in the title of the graph.
4. Estimate the annualized expected rate of return μ and the volatility rate σ from the historical data.

Requirements: You must clearly state which asset you choose, the exact time period for which the historical prices you use, where you obtain the data, what method you use for estimating the two parameters (include mathematical formula if necessary).

1.2 Generate Future Stock Prices Using Monte Carlo Simulation

Here you are asked to use the parameter values for μ and σ obtained from above and the discretized model (2) to randomly generate future prices for the stock and then answer questions based on your simulated future price trajectories.

1. Find the stock price on 03/31/2014. Assume that is the most recent end-of-day price of the stock that you can get from the market quote. Use this number as the initial price S_0 in your simulation.
2. Select a small number for the time interval Δt. One day would be a good choice and is recommended for use. However, you can try different values, for example, one hour, one minute, etc.
3. Use simulation to randomly generate 1000 trajectories for the future stock prices for one year. If you use one day for Δt, 250 random prices should be generated for each trajectory; one for each trading day, assuming there are 250 trading days in a year.
4. Display the first 10 trajectories on graph. Use the horizontal axis for time and vertical axis for price. You may plot each trajectory on a separate graph, or put all of them on one graph. For the latter, try to use different color or line type for different trajectories.
5. For each of the trajectory you produced, find/calculate and report the following statistical quantities:
   - Maximum and minimum prices.
   - Average price for the one year period.
   - Average rate of return.
   - Standard deviation of the rate of return.
6. Based on your generated price trajectories and the calculated quantities, answer the following questions:

- Are the simulated future price trajectories similar to the historical realization you obtained in part 1.1?
- Are the average rates of return and the standard derivations you calculated from the simulated future price trajectories close to or significantly different from the estimated values for μ and σ in part 1.1?
- What comments/explanations do you have?

Requirements: You must clearly state which specific day the stock price you use as S_0, what value you choose for Δt in your simulation, how the Gaussian random numbers are generated (if you use a built-in function from the software you use, mention the function), how the future prices are generated (include mathematical equations if necessary), what method/formula you use to calculate the required statistical quantities.

2 Black–Scholes–Merton Option Pricing Model (20%)

- The purpose of this part is to understand the Black–Scholes–Merton (BSM) model for option pricing and to be able to calculate the risk-neutral prices for standard options. To successfully complete the question, you need to go over the BSM model as well as the algorithms thoroughly.
- You need to specify the options first. A list of requirements is stated below that you should follow to specify the options you are going to study.
- The stock that you have chosen in part 1 should be used as the underlying asset of the options. Choose option parameters in the way specified as following:

  1. **Stock Price** S. Set $S = S_0$ and use the 1000 trajectories that you have generated in part 1.
  2. **Strike Price** K. Choose from (a) $K = S$, (b) $K = 1.1S$, (c) $K = 1.2S$, (d) $K = 0.9S$, (e) $K = 0.8S$, where S is the stock price.
  3. **Volatility** σ. Use the annualized volatility that you have estimated from the historical data in part 1.
  4. **Maturity** T. Choose from (a) $T = 3$ months, (b) $T = 6$ months, (c) $T = 9$ months, (d) $T = 1$ year, (e) $T = 2$ years.
  5. **Risk-Free Interest Rate** r. Use $r = 5\%$ for all options.
  6. **Dividend Yield** q. Choose from (a) $q = 0$ (no dividend), (b) $q = 1\%$.
  7. **Option Type**. Call and Put, European and American.

  This would produce totally 200 different options (100 European and 100 American options) for you to use in the project.

- This part consists of three parts: (1) valuing European options using the BS formula; (2) valuing European options using Monte-Carlo simulations; (3) valuing both European and American options using binomial trees.

2.1 Calculate Option Price Using the BSM Formula

Calculate the prices of 4 European call options and 4 European put options using the BSM formula. Among the 8 options you select:

- At least one option should have a non-zero dividend yield (i.e., $q = 1\%$), at least one option should be at-the-money (i.e., $K = S$);
- At least one option should be in-the-money (i.e., $K < S$ for call and $K > S$ for put);
- At least one option should be out-of-the-money (i.e., $K > S$ for call and $K < S$ for put).

Requirement: You must clearly state the specifications for each option you use and report your calculated option value. A neat way would be using a table.
Note: For this part you can either work out the calculations in Excel, or write a macro using VBA.

2.2 Calculate Option Price Using Monte Carlo Simulation

- Write a program (e.g., macro using VBA) to implement the Monte Carlo simulation algorithm for pricing European options. In your implementation, set the number of independent sample paths of the underlying stock prices as a variable (say n).
- Use your program to calculate the price of one European call option and one European put option. For the purpose of comparison, the two options you use here should be among those already valued in part 2.1 by BSM formula. Calculate each option price using the following values for $n : n = 100, 1000, 5000, 10000$ (**Optional**: $n = 50000, 100000, 500000, 1000000, \ldots$) Report the results and compare them with the exact price obtained by BSM formula. You should observe that as n increases, the Monte Carlo approximations become closer and closer to the exact value.

Note: The same procedure used in part 1 for generating future stock prices can be used here to generate each sample path. However, for option pricing, the GBM model should be the **risk-neutral model**, not the real world model.

2.3 Calculate Option Price Using Binomial Tree Algorithm

Use the following values for the number of steps m of the binomial tree: $m = 4, 8, 16, 32$ (**Optional:** $m = 100, 500, \ldots$)

1. For each m, determine the length of step Δt, the risk-neutral probability p, the up move factor u, and the down move factor d for the binomial tree. Report your calculation results.
2. Calculate the prices of the same two European options you have used in part 2.2 by the Monte Carlo method for different m as specified. Report and compare your results.
3. Change the option type from European to American and keep all other parameters to be exactly the same. Calculate the prices of the two American options by the binomial tree method for different m as specified. Report and compare your results. Also compare the prices between the European option and the corresponding American option.

Note: For this part you can either work out the calculations in Excel, or write a macro using VBA.

Important Note

Softwares that implement these standard option valuation methods can be easily found on Web (for example, you can download Hull's DerivaGem program from his web). By simply typing the option parameters, you could get the correct results that are expected in this project. Obtaining option prices in this way is NOT ACCEPTABLE. You MUST implement the methods yourself. However, you may use these tools to check the results obtained from your implementations.

3 Option Pricing with Finite Difference Method (30%)

- In this part, you will implement a finite difference method for option pricing. You should think of this part as a starting point for investigating additional issues.
- Under a no-arbitrage model, option price can typically be computed using a lattice method, a Monte Carlo method, or a numerical partial differential equation (PDE) method. See, e.g., Wilmott, Dewynne & Howison (1993).
- In a local volatility function model, see, e.g., Dupire (1994), the underlying price is assumed to satisfy the following stochastic differential equation:

$$\frac{dS}{S} = (\mu - q)dt + \sigma(S,t)dW$$

  where the expected rate of return μ and the continuous dividend yield q are positive constants, W is a standard Brownian motion, and $\sigma(S,t)$ is a deterministic local volatility function (LVF) satisfying some suitable regularity conditions. Assume that the interest rate $r > 0$ is a constant and S_0 denotes the initial underlying price. The European option value $V(S,t)$ satisfies the following backward parabolic PDE,

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma(S,t)^2 S^2 \frac{\partial^2 V}{\partial S^2} + (r-q)S\frac{\partial V}{\partial S} - rV = 0 \tag{3}$$

  where $S \in [0,\infty)$ and $t \in [0,T]$. For an American option, the model satisfies a partial differential complementarity equation and numerical methods for PDE can similarly be applied; we will discuss American option pricing later.
- As an alternative to the classical BSM model, examples of LVF models have been considered for option pricing, see, e.g., Cox and Ross (1976). In particular, the following constant elasticity of variance (CEV) model is considered,

$$\frac{dS}{S} = (\mu - q)dt + \alpha S^{1-\beta} dW \tag{4}$$

  where $\alpha > 0$ and $\beta \geq 1$ are constants. Setting $\beta = 1$ yields the classical BSM model.

Remark *This model is known as CEV model because* $\frac{\partial \operatorname{Var}(\widetilde{R})/\operatorname{Var}(\widetilde{R})}{\partial S/S} = 2(1-\beta)$. *When* $\beta > 1$*, the variance of the asset return is a decreasing function of the asset price. Some special cases:*

  - $\beta = 1$: *Geometric Brownian Motion.*
  - $\beta = 3/2$: *Square Root Process.*
  - $\beta = 2$: *Absolute Diffusion Process.*

  In general, $S = 0$ could be either an absorbing barrier, reflecting barrier *or* inaccessible*, depending on the values of μ, α and β. Only the absorbing barrier case is the reasonable one as a model of stock prices.*

- To compute option values via the PDE (3), discretization and approximation is necessary. Assume that T is the expiry of an option. Let $\{(S^i, t_j)\}_{j=1,\ldots,M+1}^{i=1,\ldots,N+1}$ denote a discretization of the region $[0, \infty) \times [0, T]$ where

$$S^1 < S^2 < \cdots < S^{N+1} \quad \text{and} \quad t_1 < t_2 < \cdots < t_{M+1}$$

  A uniform discretization can be set up from $\Delta t = \frac{T}{M}$, $\Delta S = \frac{S_{\max}}{N}$, $t_j = (j-1)\Delta t$, and $S^i = (i-1)\Delta S$, where $S_{\max} > 0$ is some large constant. Since the boundary conditions at $S = S^1$ and $S = S^{N+1}$ are typically approximate option values based on $S \to 0$ and $S \to \infty$, it is desirable for a discretization to cover a large region. However, for computational efficiency, one wants to avoid a large number of discretization points. Thus a non-uniform discretization, which is done in the region in which option value changes rapidly (e.g., near strike prices) but coarse in the region of slow change (e.g., near the boundary $S = S^1$ and $S = S^{N+1}$) can lead to better computational efficiency and accuracy. If the same discretization is used to price options of different strikes, one can choose a non-uniform discretization which has the highest density around $S = S_0$ and gradually become coarser further away from $S = S_0$. Most frequently traded calls and puts have strikes in the interval $[0.8S_0, 1.2S_0]$.

- Let V_j^i denote the option value at t_j and price S^i. For subsequent computations, consider the parameter values given in Table 1.

S_0	α	β	r	q
100	20	2	4%	2%

Table 1: A sample parameter setting

3.1 Tasks

1. To compute European option prices numerically, appropriate boundary conditions for $S = S^1$ and $S = S^{N+1}$ need to be imposed on the option values. Describe your boundary conditions (at $S \to 0$ and $S \to \infty$) for European call and put respectively. Explain the reason of your choice.

2. Implement a Crank–Nicolson finite difference method to price European calls or puts using appropriate final and boundary conditions. It can be shown that option values $\left[V_{j-1}^2, \ldots, V_{j-1}^N\right]$ satisfy a finite difference tri-diagonal linear system. You are encouraged to choose a non-uniform grid discretization. However, if you have difficulties, you may simply use a uniform grid. Ensure that your program yields an initial option value with accuracy up to one cent either by considering the BSM model as a special case and checking with prices from the BSM formula using the MATLAB function `blsprice`, or conducting convergence analysis computationally by varying step sizes.

Remark *The following is a sample MATLAB function signature*

$$[V_0, V] = \texttt{MyCallPut}(S_0, r, q, T, K, \texttt{flag})$$

The function returns the initial option value V_0 if the function is called with one output argument. If there are two output arguments when the function is called, the second argument V is a $(N+1) \times (M+1)$ of option values for different underlying prices at different times. In MATLAB, `nargout` identifies the number of output arguments in a function call. Make your program efficient both in CPU time and memory usage. The function returns call values when `flag = 1` and put values otherwise. The input argument S_0 is the initial underlying price, r and q are constant interest rate and continuous dividend yield, K is the strike, and $T > 0$ is the expiry of the option. For computational efficiency, you can either use MATLAB sparse linear system solver or use provided MATLAB functions `TriDiLU`, `LBiDiSol`, and `UBiDiSol` below to solve tri-diagonal linear systems. Your pricing function can also take a MATLAB function name as an additional input argument to allow a user to specify a MATLAB function to evaluate a LVF.

a. You probably have noticed that for a CEV model (4) with $\beta > 1$, $\sigma(S)$ is undefined when $S = 0$. Computationally experiment setting $\sigma(0)$ to arbitrarily large values, e.g., 100, 1000, 5000, 10000, and observe the effects on your computed option values. What do you observe? What are your explanations for your observation?

b. Graphically display the matrix of the option values (e.g., use MATLAB command `mesh`) for an at-the-money put. An option is at-the-money if, at the current time $t = 0$, the underlying price S_0 equals the strike K. Comment on how the option values change with time and underlying price.

c. In addition, use your pricing function to tabulate the initial put option prices with the expiry $T = 1$ and strike equal to 90%, 92%, 94%, $\ldots$, 110% of S_0 respectively.

3. Use your MATLAB pricing function and the MATLAB function `blsimpv`, write a MATLAB program to plot the implied volatility as a function of strike and maturity for the puts with strikes $[0.8, 0.9, 1, 1.1, 1.2]S_0$ and maturities of 3-month and 6-month respectively, What trends do you observe in these implied volatilities?

4. (**Optional**) Alternative to the backward PDE (3), one can regard the *initial* European option value $V(K, T; S_0, 0)$ as a function of the strike K and expiry T. It can be shown that, with the strike K and expiry T as independent variables, the initial option value function $V(K, T; S_0, 0)$ satisfies the following *forward* parabolic PDE

$$\frac{\partial V}{\partial T} - \frac{1}{2}\sigma(K,T)^2 K^2 \frac{\partial^2 V}{\partial K^2} + (r - q)K\frac{\partial V}{\partial K} + qV = 0 \tag{5}$$

For calls, the initial condition is given by $V(K, 0; S_0, 0) = \max(S_0 - K, 0)$.

a. Based on the forward PDE (5), what would the appropriate boundary conditions for European calls and puts be? Explain.

b. Assume that the initial values of options for many different strikes and maturities are needed. Which method, backward PDE (3) or forward PDE (5), is computationally more efficient? Explain.

c. Assume that the initial as well as future option values for a single option are needed. Which method is more efficient now? Explain.

3.2 MATLAB Functions TriDiLU, LBiDiSol, and UBiDiSol

```
  function [l,u] = TriDiLU(d,e,f)
%
% Pre:
%   d,e,f   n-vectors that define a tridiagonal matrix
%           A = diag(e(2:n),-1) + diag(d) + diag(f(1:n-1),1).
%           A has an LU factorization.
% Post:
%    l      n-vector with the property that
%           L = eye + diag(l(2:n),-1)
%    u      n-vector with the property that
%           U = diag(u) + diag(f(1:n-1),1).

   n = length(d);
   l = zeros(n,1);
   u = zeros(n,1);
   u(1) = d(1);
   for i=2:n
      l(i) = e(i)/u(i-1);
      u(i) = d(i) - l(i)*f(i-1);
   end
```

```
  function x = LBidiSol(l,b)
%
% Pre:
%    l  n-vector that defines the lower bidiagonal matrix
%       L = I + diag(l(2:n),-1).
%    b  n-by-m matrix
%
% Post:
%    x  n-by-m matrix that solves Lx = b

   n = size(b,1);
   x = zeros(n,size(b,2));
   x(1,:) = b(1,:);
   for i=2:n
      x(i,:) = b(i,:) - l(i)*x(i-1,:);
   end
```

```
  function x = UBidiSol(u,f,b)
%
% Pre:
%  u,f  n-vectord that define the upper bidiagonal matrix
%       U = diag(u) + diag(f(1:n-1),1)
%    b  n-by-m matrix
%
% Post:
%    x  n-by-m matrix that solves Ux = b


   n = size(b,1);
   x = zeros(n,size(b,2));
   x(n,:) = b(n,:)/u(n);
   for i=n-1:-1:1
      x(i,:) = (b(i,:) - f(i)*x(i+1,:))/u(i);
   end
```

4 Option Pricing with Monte Carlo Method (40%)

4.1 Barrier Options

- Using Monte Carlo simulations, no arbitrage values for European path dependent options can be computed easily.
- Assume that a stock does not pay dividend and the stock price satisfies the GBM model (1)
$$\frac{dS_t}{S_t} = \mu\, dt + \sigma\, dW_t$$
where W_t is a standard Brownian motion and $\mu, \sigma > 0$ are constants.
- The up-and-out European call option value V_{out}, assuming that the barrier has not been reached at time t, is given by the analytic formula
$$\begin{aligned} V_{\text{out}}(S,t) &= S\left(\mathcal{N}(d_1) - \mathcal{N}(d_3) - \left(\frac{S_u}{S}\right)^{1+\frac{2r}{\sigma^2}} (\mathcal{N}(d_6) - \mathcal{N}(d_8))\right) \\ &\quad - Ke^{-r(T-t)}\left(\mathcal{N}(d_2) - \mathcal{N}(d_4) - \left(\frac{S_u}{S}\right)^{-1+\frac{2r}{\sigma^2}} (\mathcal{N}(d_5) - \mathcal{N}(d_7))\right) \end{aligned}$$
where $\mathcal{N}(d)$ is the cumulative distribution function for the standard normal and
$$\begin{aligned} d_1 &= \frac{\log\left(\frac{S}{K}\right) + \left(r + \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}} \quad , \quad d_2 = \frac{\log\left(\frac{S}{K}\right) + \left(r - \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}} \\ d_3 &= \frac{\log\left(\frac{S}{S_u}\right) + \left(r + \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}} \quad , \quad d_4 = \frac{\log\left(\frac{S}{S_u}\right) + \left(r - \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}} \\ d_5 &= \frac{\log\left(\frac{S}{S_u}\right) - \left(r - \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}} \quad , \quad d_4 = \frac{\log\left(\frac{S}{S_u}\right) - \left(r + \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}} \\ d_7 &= \frac{\log\left(\frac{SK}{S_u^2}\right) - \left(r - \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}} \quad , \quad d_8 = \frac{\log\left(\frac{SK}{S_u^2}\right) - \left(r + \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}} \end{aligned}$$
- Consider a European up-and-out barrier option with strike $K = 90$ and barrier $S_u = 120$. Assume that the expected rate of return $\mu = 15\%$, volatility $\sigma = 20\%$, the risk-free rate $r = 5\%$, and the expiry $T = 1$.

  1. Compute and plot the initial up-and-out European call option value $V(S,t)$ for $S = 80 : 2 : 120$ and $t = 0$. In MATLAB, the cumulative distribution function for a standard normal can be computed using `normcdf`.
  2. Using a Monte Carlo method with $N = 10000$ simulations and $\Delta t = 0.001$, compute the initial barrier option value when the initial underlying price is $S_0 = 100$. Repeat this computation 200 times and report the mean and standard deviation of your 200 estimates of the initial barrier value. *This computation can take some time; make sure that your computation is as efficient as possible.* Compare Monte Carlo estimations with the value using the analytic formula.

Application: Accumulator

See http://en.wikipedia.org/wiki/Accumulator_(structured_product) for a brief introduction of the product. Replicate the analysis in Yen and Xiang (2008) with the stock that you have chosen in part 1 as the underlying asset. Choose parameters in the way specified as follows:

1. **Stock Price** S. Set $S = S_0$ and use the 1000 trajectories that you have generated in part 1.
2. **Reference Price** K. Choose from (a) $K = 0.9S$, (b) $K = 0.8S$, where S is the stock price.
3. **Knock-Out Price** S_u. Choose from (a) $S_u = 1.02S$, (b) $S_u = 1.05S$, where S is the stock price.
4. **Volatility** σ. Use the annualized volatility that you have estimated from the historical data in part 1.
5. **Maturity** T. Use $T = 1$ year.
6. **Risk-Free Interest Rate** r. Use $r = 5\%$ for all contracts.
7. **Dividend Yield** q. Choose from (a) $q = 0$ (no dividend), (b) $q = 1\%$.

Are your results consistent with Yen and Xiang (2008)? Explain.

4.2 Hedging Analysis

- Monte Carlo simulation can also be used to analyze performance of a dynamic hedging strategy. Under the BSM option pricing model, it is assumed that hedging is performed continuously, thus option risk can be completely eliminated. In practice, continuous hedging is not possible; transaction cost consideration often limits trading frequency.
- For simplicity, we assume that none of the assets concerned here pay any dividends (i.e., $q = 0$) and the continuously compounding interest rate r is constant. Suppose that you have sold a European option with expiry T. You want to reduce risk in this (target) option by trading a set of risky assets and a bond with maturity T at a set of fixed times

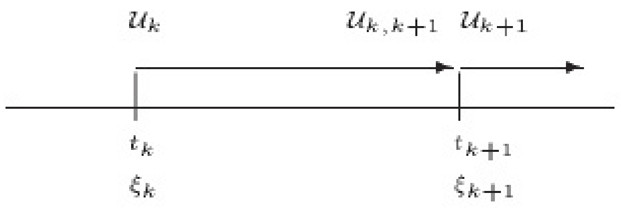
  $$0 = t_0 < t_1 < \cdots < t_M = T$$

  Here, M and N represent total number of hedging times and total number of simulations.
- Let us normalize all time t asset values (including the target) by the time t bond values, this corresponds to discounting all values to time zero. Consequently the bond value is always one, i.e., $B_k \equiv 1$.
- Let the vector $\mathcal{U}_k$ denote the (normalized) risky asset values at t_k and $\mathcal{U}_{k,k+1}$ denote time t_{k+1} value of risky assets traded at t_k. This notation allows trading different assets in different hedging periods.

- The trading positions at time t_k are denoted ξ_k for the risky assets and η_k for the bond, i.e.,

$$\underbrace{n \text{ risky assets}}_{\xi_k} + \underbrace{\text{bond}}_{\eta_k}$$

A dynamic hedging strategy can be represented by a non-anticipatory process $\{(\xi_k, \eta_k), k = 0, 1, \ldots, M\}$, e.g.,



The value of the hedging portfolio at time t_k is then given by $P_k = \xi_k \cdot \mathcal{U}_k + \eta_k$.

- The cumulative *gain* of the strategy $\{\xi_k, \eta_k\}$ at time t_k is

$$G_k = \sum_{j=0}^{k-1} \xi_j \cdot (\mathcal{U}_{j,j+1} - \mathcal{U}_j)$$

with $G_0 = 0$. the cumulative *cost* at time t_k, associated with the strategy $\{\xi_k, \eta_k\}$, is

$$C_k = P_k - G_k$$

A dynamic trading strategy is self-financing if

$$C_0 = C_1 = \cdots = C_M$$

i.e., there is no additional in or out cashflow at each rebalancing times.

- It can be shown that, for self-financing hedging strategy,

$$P_k = P_0 + G_k = P_0 + \sum_{j=0}^{k-1} \xi_j \cdot (\mathcal{U}_{j,j+1} - \mathcal{U}_j)$$

and

$$\eta_k = \eta_{k-1} + \xi_{k-1} \cdot \mathcal{U}_{k,k+1} - \xi_k \cdot \mathcal{U}_k$$

Let $\mathcal{V}_t$ denote the normalized target option value at time t, $t \leq T$ where T is the expiry. For a dynamic hedging strategy $\{\xi_k, \eta_k\}$, the hedging error at T is the difference between the normalized target value $\mathcal{V}_T$ and the self-financing portfolio value P_M. This hedging error is also called total risk.

- Hedging error at T is a random variable. We can analyze hedging error, associated with a dynamic trading strategy $\{\xi_k, \eta_k\}$ using Monte Carlo simulations.

- In the remaining part, you will perform hedging analysis for discrete delta hedging. In this case, the time t_k value of the risky hedging asset us the underlying S_k.

- A dynamic delta hedging strategy rebalances the hedging portfolio at the specified trading times t_k with the trading position in the underlying given by

$$\xi_k = \left(\frac{\partial V}{\partial S}\right)_{t=t_k}$$

where $V(S,t)$ denote the target option value at time t when the underlying price is S.

- A dynamic delta hedging strategy $\{\xi_k, \eta_k\}$ is self-financing if, at time t_k, the bond position is given by

$$\eta_k = \eta_{k-1} + \xi_{k-1} \cdot e^{-rt_k} S_k - \xi_k \cdot e^{-rt_k} S_k$$

Thus time T hedging error associated wit a dynamic delta hedging strategy is

$$\text{HedgeError} = e^{-rT} V_T - \left(\xi_0 S_0 + \eta_0 + \sum_{k=0}^{M-1} \xi_k \left(e^{-rt_{k+1}} S_{k+1} - e^{-rt_k} S_k \right) \right)$$

where the target option at time T has value V_T, $\eta_0 = V_0 - \xi_0 S_0$, and $\xi_k = \left(\frac{\partial V}{\partial S}\right)_{S=S_k, t=t_k}$.

- In a PDE pricing method, partial derivatives can be approximated using finite difference. For example, at the discretization point (k,i), delta value can be estimated using the central finite difference, i.e.,

$$\left(\frac{\partial V}{\partial S}\right)_{S=S_k, t=t_k} \approx \frac{V_k^{i+1} - V_k^{i-1}}{S^{i+1} - S^{i-1}}$$

We can use Monte Carlo simulations to analyze the hedging error and investigate characteristics of hedging error distributions.

- For delta hedging analysis below, assume that the underlying price follows a CEV model as in part 3 modified as follows:

$$\frac{dS}{S} = (\mu - q)dt + \alpha S^{1-\beta} dW \quad , \quad \sigma(S,t) = \min(100, \alpha S^{1-\beta}) \tag{6}$$

where $\alpha > 0$, $\beta \geq 1$ and other parameters are given in Table 2.

S_0	μ	α	β	r	q
100	15%	20	2	4%	0

Table 2: A sample parameter setting

- Assume that you have just sold an at-the-money (ATM) put option with a one year time to expiry.

  1. Modify your PDE pricing code so that it now produces a matrix of approximate delta values for the ATM put at discretization points. Display your computed delta surface.
  2. Assume that the delta hedging portfolio is rebalanced weekly, i.e., $M = 52$. Generate $N = 10000$ independent price paths for the analysis below.

a. For each price path, determine the corresponding delta at each rebalancing time. you can use linear interpolation or extrapolation to determine an approximation to the delta value when the underlying price at the rebalancing time does not correspond to any discretization point in your PDE implementation.

b. Determine the hedge error T for each price path. Graphically display time T hedge error distribution. Report mean, standard deviation, 95% quantile, and 95% shortfall for the hedging error.

3. Repeat the above for $M = 26, 12, 1$. Tabulate mean, standard deviation, 95% quantile, and 95% shortfall for $M = 52, 26, 12, 1$. Discuss your results.

4.3 Multi-Asset Options

- A basket option is an option that provides a payoff dependent on the value of a portfolio of assets. Specifically, the value of a basket call option at its maturity T is

$$\max\left(\sum_{i=1}^{L} w_i S_i(T) - K, 0\right)$$

where L is the number of assets in the basket (portfolio) and w_i is the weight of the ith asset in the basket.

- To simplify this part, let us assume the asset price dynamics in the risk neutral world follow

$$\begin{aligned} dS_t^{(1)} &= rS_t^{(1)}dt + \sigma^{(1)}S_t^{(1)}dW_t^{(1)} \\ &\vdots \\ dS_t^{(L)} &= rS_t^{(L)}dt + \sigma^{(L)}S_t^{(L)}dW_t^{(L)} \end{aligned}$$

when we take the correlation components of the vector dW as

$$\widehat{\mathrm{E}}(dW^{(j)}dW^{(k)}) = \begin{cases} \rho_{j,k}dt & \text{for } j \neq k \\ dt & \text{for } j = k \end{cases}$$

- Suppose $w_i = 1/L$, then we can define a corresponding geometric basket option with the payoff

$$\max\left(\left(\prod_{i=1}^{L} S_i(T)\right)^{1/L} - K, 0\right)$$

and it can be shown that if we let $G(t) = \prod_{i=1}^{L} S_i(t)^{1/L}$, then we have

$$e^{-rT}\widehat{\mathrm{E}}\left(G(T) - K\right)^{+} = e^{-rT}\widehat{\mathrm{E}}\left(G(T)\right)\Phi(d_1) - Ke^{-rT}\Phi(d_2)$$

where

$$
\begin{aligned}
\mathrm{E}\left(\left(\prod_{i=1}^{L} S_i(T)\right)^{1/L}\right) &= G(t)\exp\left(\left(r-\frac{1}{2L}\sum_{i=1}^{L}\sigma_i^2\right)(T-t)+\frac{1}{2}\sigma^2(T-t)\right), \\
\sigma^2 &= \frac{1}{L^2}\sum_{i,j=1}^{L}\rho_{i,j}\sigma_i\sigma_j, \\
d_1 &= \frac{\log\frac{G(t)}{K}+\left(r+\sigma^2-\frac{1}{2L}\sum_{i=1}^{L}\sigma_i^2\right)(T-t)}{\sigma\sqrt{T-t}}, \\
d_2 &= d_1-\sigma\sqrt{T-t}
\end{aligned}
$$

- Consider the case $L = 2$, $w_1 = w_2 = 1/2$, $K = S_0$, $\sigma_1 = \sigma_2 = 0.3$, $\rho = 0.5$, $T = 1$ and $r = 5\%$. Compare the estimates of the value of a basket call on two asset given by the binomial and Monte Carlo (with and without antithetic variate) for various lattice steps (M) and number of simulations (N).

References

[1] Cox, John C., and Stephen A. Ross, 1976, The valuation of options for alternative stochastic processes, *Journal of Financial Economics* 3, 145–166.

[2] Dumas, Bernard, Jeff Fleming, and Robert E. Whaley, 1998, Implied volatility function: Empirical tests, *Journal of Finance* 53, 2059–2106.

[3] Dupire, Bruno, 1994, Pricing with a smile, *Risk* 7(2), 229–263.

[4] Wilmott, Paul, Jeff Dewynne, and Sam Howison, 1993, *Option Pricing: Mathematical Models and Computations* (Oxford Financial Press).

[5] Yen, Jerome, and Yi Xiang, 2008, Empirical analysis of accumulator, *Hong Kong Economic Journal Monthly* 378(September), 20–26. (in Chinese)