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

FINA 4140 A Computational Finance
Assignment 4
Due Date: April 17, 2014

Answer the following questions for a total of 150 points (30 points for each question) and show all your work carefully. 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. Through the course, we shall often make use of the expression "MATLAB command": in this case, MATLAB should understood as the *language* which is the common subset of both programs MATLAB and Octave. Please do not turn in the assignment 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 assignment will be checked for correctness, completeness, and clearness of the answers. 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. Implement the Newton method in order to compute implied volatilities for European call and put options in a programming language. The program should ask for the interest rate r, initial stock price S_0, time to maturity T, strike price K, the observed option price $V^\star$, the number N of steps which should be carried out by the program in the iteration for σ_k and σ_0 for the initialization. The program should then print σ_N. The following programs from Higham (2004) may serve as a reference and parameters in `ch14` will be used as the test case.

```
%CH14     Program for Chapter 14
%
% Computes implied volatility for a European call

%%%%%%%%%%%%%%%%%%%%%%% parameters %%%%%%%%%%%%%%%%%%%%%%
r = 0.03; S = 2; E = 2; T = 3; tau = T; sigma_true = 0.3;
[C_true, Cdelta, P, Pdelta] = ch08(S,E,r,sigma_true,tau);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%starting value
sigmahat = sqrt(2*abs( (log(S/E) + r*T)/T ) );

%%%%%%%%%%%%% Newton's Method %%%%%%%%%%
tol = 1e-8;
sigma = sigmahat;
sigmadiff = 1;
k = 1;
kmax = 100;
```

```
while (sigmadiff >= tol & k < kmax)
    [C, Cdelta, Cvega, P, Pdelta, Pvega] = ch10(S,E,r,sigma,tau);
    increment = (C-C_true)/Cvega;
    sigma = sigma - increment;
    k = k+1;
    sigmadiff = abs(increment);
end
sigma
```

```
function [C, Cdelta, P, Pdelta] = ch08(S,E,r,sigma,tau)
% Program for Chapter 8
% This is a MATLAB function
%
% Input arguments: S = asset price at time t
%                  E = Exercise price
%                  r = interest rate
%                  sigma = volatility
%                  tau = time to expiry (T-t)
%
% Output arguments: C = call value, Cdelta = delta value of call
%                   P = Put value, Pdelta = delta value of put
%
%   function [C, Cdelta, P, Pdelta] = ch08(S,E,r,sigma,tau)

if tau > 0
   d1 = (log(S/E) + (r + 0.5*sigma^2)*(tau))/(sigma*sqrt(tau));
   d2 = d1 - sigma*sqrt(tau);
   N1 = 0.5*(1+erf(d1/sqrt(2)));
   N2 = 0.5*(1+erf(d2/sqrt(2)));
   C = S*N1-E*exp(-r*(tau))*N2;
   Cdelta = N1;
   P = C + E*exp(-r*tau) - S;
   Pdelta = Cdelta - 1;
else
   C = max(S-E,0);
   Cdelta =  0.5*(sign(S-E) + 1);
   P = max(E-S,0);
   Pdelta = Cdelta - 1;
end
```

```
function [C, Cdelta, Cvega, P, Pdelta, Pvega] = ch10(S,E,r,sigma,tau)
% Program for Chapter 10
% This is a MATLAB function
%
% Input arguments: S = asset price at time t
%                  E = Exercise price
%                  r = interest rate
%                  sigma = volatility
%                  tau = time to expiry (T-t)
%
% Output arguments: C = call value, Cdelta = delta value of call
%                   Cvega = vega value of call
%                   P = Put value, Pdelta = delta value of put
%                   Pvega = vega value of put
```

```
%
%   function [C, Cdelta, Cvega, P, Pdelta, Pvega] = ch10(S,E,r,sigma,tau)

if tau > 0
   d1 = (log(S/E) + (r + 0.5*sigma^2)*(tau))/(sigma*sqrt(tau));
   d2 = d1 - sigma*sqrt(tau);
   N1 = 0.5*(1+erf(d1/sqrt(2)));
   N2 = 0.5*(1+erf(d2/sqrt(2)));
   C = S*N1-E*exp(-r*(tau))*N2;
   Cdelta = N1;
   Cvega = S*sqrt(tau)*exp(-0.5*d1^2)/sqrt(2*pi);
   P = C + E*exp(-r*tau) - S;
   Pdelta = Cdelta - 1;
   Pvega = Cvega;
else
   C = max(S-E,0);
   Cdelta =  0.5*(sign(S-E) + 1);
   Cvega = 0;
   P = max(E-S,0);
   Pdelta = Cdelta - 1;
   Pvega = 0;
end
```

(Optional) From https://www.hkex.com.hk/eng/stat/dmstat/dayrpt/dmrcalendar.asp acquire Hang Seng Index Option data for March call and put through the Daily Market Report (Archive) between March 3 and 28. Compare the implied volatility computed by your program to the reported value, and investigate the behavior of the implied volatility as the expiry time varies.

2. a. (Higham, Exercise 15.4) For the computational experiment that produced Figure 15.1 of Higham (2004), it was predicted that 'To reduce the error to, say, 10^{-4}, would take of the order of 10^8 samples, and to reduce it to 10^{-6} would take of the order of 10^{12} samples.' Where do these figures comes from? For the computations in Figure 15.2 of Higham (2004), roughly how many samples would be needed to reduce the error to 10^{-6}?

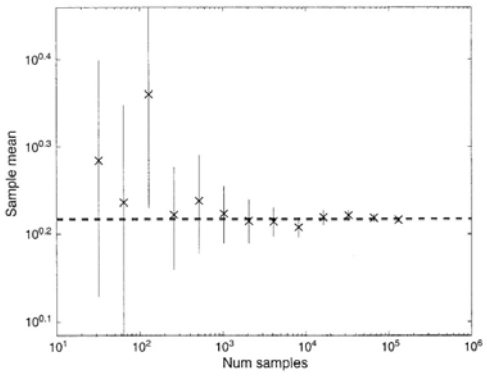

Fig. 15.1. Monte Carlo approximations to $\mathbb{E}(e^Z)$, where $Z \sim N(0,1)$. Crosses are the approximations, vertical lines give computed 95% confidence intervals. Horizontal dashed line is at height $\mathbb{E}(e^Z) = \sqrt{e}$.

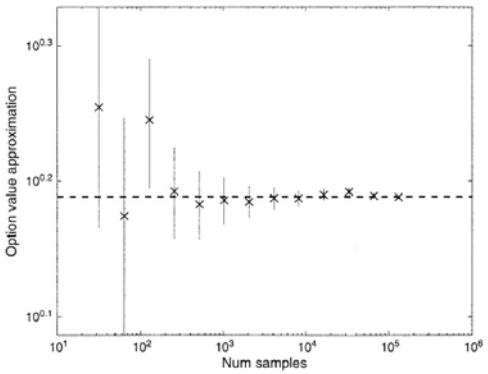

Fig. 15.2. Monte Carlo approximations to a European call option value. Crosses are the approximations, vertical lines give computed 95% confidence intervals. Horizontal dashed line is at height given by the Black–Scholes formula.

b. (Higham, Programming Exercise 15.1) Adapt the following MATLAB program to produce a picture like that in Figure 15.2 of Higham (2004):

```
%CH15      Program for Chapter 15
%
```

```
%   Monte Carlo for a European put

randn('state',100)

%%%%%%%%%%%%%%%%% Problem and method parameters %%%%%%%%%%%%%%%
S = 4; E = 5; sigma = 0.3; r = 0.04; T = 1;
Dt = 1e-3; N = T/Dt; M = 1e4;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

V = zeros(M,1);
for i = 1:M
    Sfinal = S*exp((r-0.5*sigma^2)*T+sigma*sqrt(T)*randn);
    V(i) = exp(-r*T)*max(E-Sfinal,0);
end
aM = mean(V); bM = std(V);
conf = [aM - 1.96*bM/sqrt(M), aM + 1.96*bM/sqrt(M)]
```

3. a. Prove that if $X_1, \ldots, X_n$ are independent random variables, and $f(\mathbf{X})$ and $g(\mathbf{X})$ are both increasing functions where $\mathbf{X} = (X_1, \ldots, X_n)$, then $\mathrm{E}(f(\mathbf{X})g(\mathbf{X})) \geq \mathrm{E}(f(\mathbf{X}))\,\mathrm{E}(g(\mathbf{X}))$.

   b. Let $X_1, \ldots, X_n$ are simulated random variables generated via the inverse transform method, i.e., $X_i = F_X^{-1}(U_i)$ where $U_i \sim \mathcal{U}(0,1)$. Prove that if $g(X_1, \ldots, X_n)$ is monotone, then $Y = g(F_X^{-1}(U_1), \ldots, F_X^{-1}(U_n))$ and $Y' = g(F_X^{-1}(1-U_1), \ldots, F_X^{-1}(1-U_n))$ are negatively correlated.

4. a. Implement the algorithm to compute the price of a European call option via Monte Carlo method using antithetic variates. The program should ask for the input parameters initial stock price S_0, interest rate r, volatility σ, strike price K, time to maturity T and the number N of simulations of the payoff. As an output the program should produce the computed price as well as a 95 percent confidence interval.

   b. Use the procedure outlined above to calculate the value of a European call and a European put in a Black–Scholes–Merton world with the following parameters:

$$t = 0, T = \{0.5, 1\}, S(0) = 100, K = \{80, 100, 120\}, r = 0.03, \sigma = 0.25.$$

   Compare the calculated value to the respective analytical value, and examine the effect of increasing the number of MC steps, say $N = 2^5, 2^6, \ldots, 2^{16}$.

   c. The Asymmetric Power Call has the following payoff profile:

$$c(T) = \max(0, S(T)^n - K^n)$$

   Use MC to calculate the value for $n = 2$ and 5 with the parameters from above. Again examine the convergence behavior. For this, calculate the "exact" value of the option by valuing it with a sufficient number of MC-steps (> 50000).

   d. A straddle is an option strategy which has the following payoff profile:

$$c(T) = |S(T) - K|$$

   Once more examine the convergence behavior with the parameters from above. Note: since the straddle is a combination of a call and a put with identical strike and maturity, the analytical value can be calculated.

5. (Higham, Programming Exercise 24.1) Alter program `ch24` of Higham (2004) so that it values a down-and-out call option.

```
%CH24      Program for Chapter 24
%
%    Crank–Nicolson for a European put

clf

%%%%%%% Problem and method parameters %%%%%%%
E = 4; sigma = 0.3; r = 0.03; T = 1;
L = 10; Nx = 50; Nt = 50; k = T/Nt; h = L/Nx;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

T1 =  diag(ones(Nx–2,1),1) – diag(ones(Nx–2,1),–1);
T2 = –2*eye(Nx–1,Nx–1)  + diag(ones(Nx–2,1),1) + diag(ones(Nx–2,1),–1);
mvec = [1:Nx–1];
D1 = diag(mvec);
D2 = diag(mvec.^2);
F = (1–r*k)*eye(Nx–1,Nx–1) + 0.5*k*sigma^2*D2*T2 + 0.5*k*r*D1*T1;
B = (1+r*k)*eye(Nx–1,Nx–1) – 0.5*k*sigma^2*D2*T2 – 0.5*k*r*D1*T1;
A1 = 0.5*(eye(Nx–1,Nx–1) + F);
A2 = 0.5*(eye(Nx–1,Nx–1) + B);

U = zeros(Nx–1,Nt+1);
U(:,1) = max(E–[h:h:L–h]',0);

for i = 1:Nt
    tau = (i–1)*k;
    p1 = k*(0.5*sigma^2 – 0.5*r)*E*exp(–r*(tau));
    q1 = k*(0.5*sigma^2 – 0.5*r)*E*exp(–r*(tau+k));
    rhs = A1*U(:,i) + [0.5*(p1+q1); zeros(Nx–2,1)];
    X = A2\rhs;
    U(:,i+1) = X;
end

bca = E*exp(–r*[0:k:T]);
bcb = zeros(1,Nt+1);
U = [bca;U;bcb];
mesh([0:k:T],[0:h:L],U)
xlabel('T–t'), ylabel('S'), zlabel('Put Value')
```