CF_HW4:

Exercises

1. Use the Binomial Method to price a 6-month European Call option with the following information: the risk-free interest rate is 5% per annum and the volatility is 24%/annum, the current stock price is $32 and the strike price is $30. Divide the time interval into n parts to estimate the price of this option. Use $n = 10, 15, 20, 40, 70, 80, 100, 200$ and 500 to compute the approximate price and draw them in one graph, where the horizontal axis measures n, and the vertical one– the price of the option. Compare the convergence rates of the four methods below:

   (a) Use the binomial method in which

$$u = \frac{1}{d}, d = c - \sqrt{c^2 - 1}, \qquad c = \frac{1}{2}\left(e^{-r\Delta} + e^{(r+\sigma^2)\Delta}\right), \qquad p = \frac{e^{r\Delta} - d}{u - d}$$

   (b) Use the binomial method in which

$$u = e^{r\Delta}\left(1 + \sqrt{e^{\sigma^2\Delta} - 1}\right), \qquad d = e^{r\Delta}\left(1 - \sqrt{e^{\sigma^2\Delta} - 1}\right), \qquad p = 1/2$$

   (c) Use the binomial method in which

$$u = e^{\left(r - \frac{\sigma^2}{2}\right)\Delta + \sigma\sqrt{\Delta}}, \qquad d = e^{\left(r - \frac{\sigma^2}{2}\right)\Delta - \sigma\sqrt{\Delta}}, \qquad p = 1/2$$

   (d) Use the binomial method in which

$$u = e^{\sigma\sqrt{\Delta}}, \qquad d = e^{-\sigma\sqrt{\Delta}}, \qquad p = \frac{1}{2} + \frac{1}{2}\left(\frac{\left(r - \frac{\sigma^2}{2}\right)\sqrt{\Delta}}{\sigma}\right)$$

2. Take the current price of GOOG. Use risk-free rate of 2% per annum, and strike price that is the closest integer (divisible by 10) to 110% of the current price. Estimate the price of the call option that expires on January of next year, using the binomial approach. GOOG does not pay dividends. To estimate the historical volatility, use 60 months of historical stock price data on the company. You may use *Bloomberg* or *finance.yahoo.com* to obtain historical prices and the current price of GOOG. Compare your price with the one you can get from *Bloomberg* or *finance.yahoo.com*. If the two are different, how would you change the volatility in your code to get the market price?

3. Consider the following information on the stock of a company and options on it: $S_0 = \$49, K = \$50, r = 0.03, \sigma = 0.2, T = 0.3846$ (20 weeks), $\mu = 0.14$. Using the binomial method (any one of them) estimate the following and draw the graphs:

(i) Delta of the call option as a function of S_0, for S_0 ranging from \$10 to \$80, in increments of \$2.

(ii) Delta of the call option, as a function of T (time to expiration), from 0 to 0.3846 in increments of 0.01.

(iii) Theta of the call option, as a function of S_0, for S_0 ranging from \$10 to \$80 in increments of \$2.

(iv) Gamma of the call option, as a function of S_0, for S_0 ranging from \$10 to \$80 in increments of \$2.

CF_HW5:

4.3 Exercises

1. Consider the following situation on the stock of company XYZ: The current stock price is \$40, and the volatility of the stock price is $\sigma = 20\%$ per annum. Assume the prevailing risk-free rate is $r = 6\%$ per annum. Use the following method to price the specified option:

(a) Use the LSMC method with 100,000 paths simulations (50,000 plus 50,000 antithetic) to price an American put option with strike price of $X = \$40$, maturity of 0.5-years, 1-year, 2-years, and current stock prices of \$36, \$40, \$44. Use Laguerre polynomials for $k = 2, 3, 4$.

(b) Use the LSMC method with 100,000 paths simulations (50,000 plus 50,000 antithetic) to price an American put option with strike price of $X = \$40$, maturity of 0.5-years, 1-year, 2-years, and current stock prices of \$36, \$40, \$44. Use Hermite polynomials for $k = 2, 3, 4$.

(c) Use the LSMC method with 100,000 paths simulations (50,000 plus 50,000 antithetic) to price an American put option with strike price of $X = \$40$, maturity of 0.5-years, 1-year, 2-years, and current stock prices of \$36, \$40, \$44. Use simple monomials for $k = 2, 3, 4$.

(d) Compare all your findings above and comment.

2. Compute the prices of American Call options on the same stock with same specifications as in part (c) of the previous problem. Compare with the exact (Black-Scholes) formula and comment.

3. Forward start options are path dependent options that have strike prices to be determined at a future date. For example, a forward start put option payoff at maturity is

$$max(S_t - S_T, 0)$$

where the strike price of the put option is S_t. Here $0 \leq t \leq T$.

(a) Estimate the value of the forward-start European put option on a stock with these characteristics: $S_0 = \$65, X = \$60, \sigma = 20\%$ per annum, risk-free rate is $r = 6\%$ per annum, $t = 0.2$ and $T = 1$.

(b) Estimate the value of the forward-start American put option on a stock with these characteristics: $S_0 = \$65, X = \$60, \sigma = 20\%$ per annum, risk-free rate is $r = 6\%$ per annum. $t = 0.2$ and $T = 1$. The continuous exercise starts at time $t = 0.2$.

CF_Finite:

Option Pricing using Explicit/Implicit and Crank-Nicolson Methods

Computational Methods in Finance

Homework # 4

Submitted by

Maitreyi Mandal

Question 1:

Steps Taken:

1. I created four functions called Q1a_bino, Q1b_bino, Q1c_bino, Q1d_bino.
2. Also Created a main function to calculate Binomial Tree
3. Finally Created a main script called Project to call all the above functions

Functions written:

1. Q1a_bino

```
function [u,d,p] = Q1a_bino(sigma,r,delta)
c=0.5*(exp(-r*delta)+exp((r+sigma^2)*delta));
d=c-sqrt(c^2-1);
u=1/d;
p=((exp(r*delta)-d)/(u-d));
end
```

2. Q1b_bino

```
function [u,d,p] = Q1b_bino(sigma,r,delta)
u=exp(r*delta)*(1+sqrt(exp((sigma^2)*delta)-1));
d=exp(r*delta)*(1-sqrt(exp((sigma^2)*delta)-1));
p=0.5;
end
```

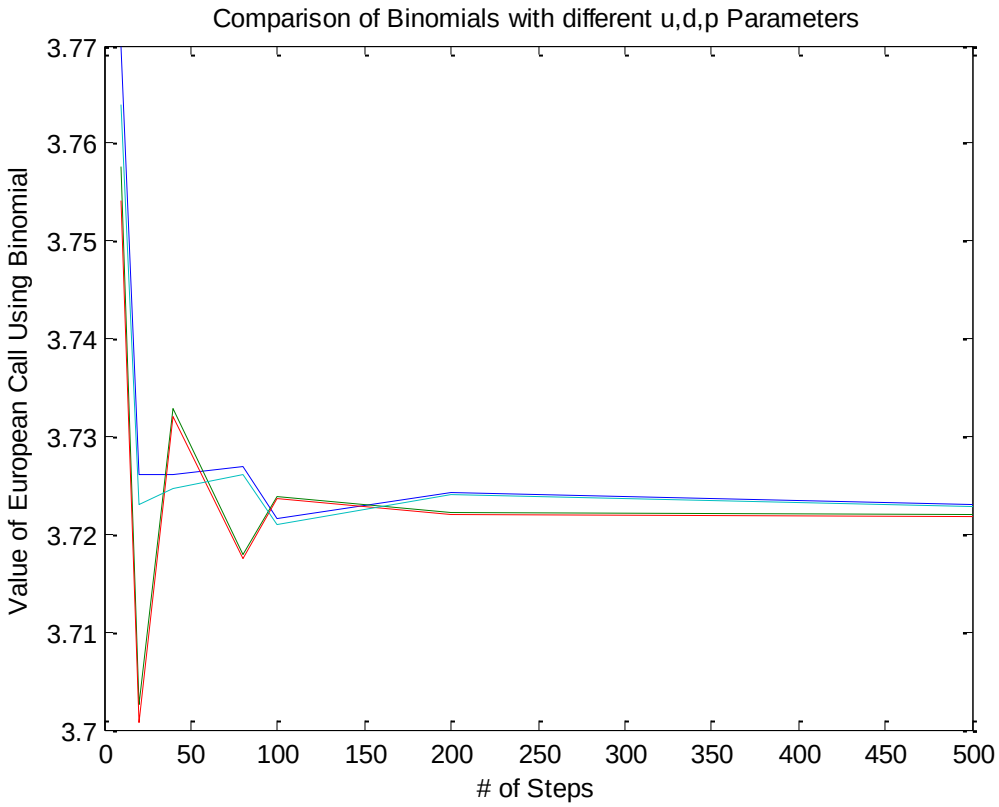
3. Q1c_bino

```
function [u,d,p] = Q1c_bino(sigma,r,delta)
u=exp((r-(sigma^2)/2)*delta+sigma*sqrt(delta));
d=exp((r-(sigma^2)/2)*delta-sigma*sqrt(delta));
p=0.5;
end
```

4. Q1d_bino

```
function [u,d,p] = Q1d_bino(sigma,r,delta)
u=exp(sigma*sqrt(delta));
d=exp(sigma*sqrt(delta)*(-1));
p= 0.5+0.5*(((r-(sigma^2)/2)*sqrt(delta))/sigma);
end
```

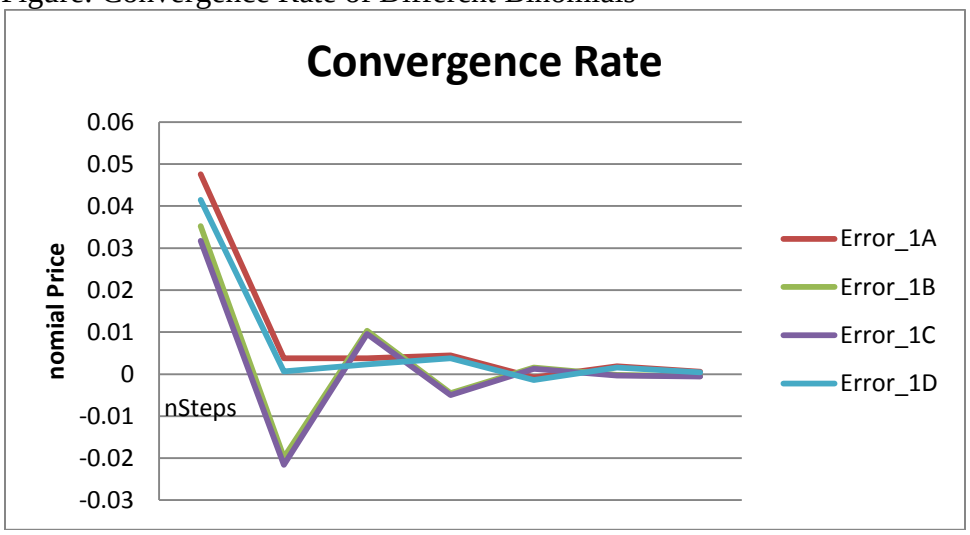
Figure: Comparison of Different Binomial Prices with different u, d, p parameters



Convergence Rate:

nSteps	1A	1B	1C	1D	bls	Error_1A	Error_1B	Error_1C	Error_1D
10	3.769944	3.757627	3.754155	3.763889	3.7224	0.047544	0.035227	0.031755	0.041489
20	3.726138	3.702577	3.700798	3.723075	3.7224	0.003738	-0.01982	-0.0216	0.000675
40	3.726147	3.732759	3.731974	3.724741	3.7224	0.003747	0.010359	0.009574	0.002341
80	3.72688	3.717871	3.71745	3.726142	3.7224	0.00448	-0.00453	-0.00495	0.003742
100	3.721626	3.723945	3.723637	3.72103	3.7224	-0.00077	0.001545	0.001237	-0.00137
200	3.724276	3.722242	3.722088	3.723983	3.7224	0.001876	-0.00016	-0.00031	0.001583
500	3.722967	3.721912	3.721847	3.722851	3.7224	0.000567	-0.00049	-0.00055	0.000451

Figure: Convergence Rate of Different Binomials



Question 2

Code Written:

```
%Question 2
clear all
S0    = 615;% Current GOOGLE Price taken from Yahoo! Finance
T     = 9/12;% Time Period
r     = .02; % Risk-Free rate
sigma = 3.2;% Volatility as obtained from Historical 60 months of Google Stock Price
K=round(1.1*S0); % Setting the Strike/Exercise Price
M = 200; % No of Steps taken
delta = T / M; % each time step

% Calculating Google Call Price Using 1b method
u=exp(r*delta)*(1+sqrt(exp((sigma^2)*delta)-1));
d=exp(r*delta)*(1-sqrt(exp((sigma^2)*delta)-1));
p=0.5;

Goog_C = max(S0 * ( (d.^(0:M)) .* (u.^(M:(-1):0)) )' - K, 0);
for m = 1:M
   Goog_C = 1/(1 + r * delta ) * ( Goog_C(1:(end-1)) * p + Goog_C(2:end) * (1-p) );
end

Goog_C
```

Value Obtained: $47.6057
From Yahoo Finance with strike of 675: $25.80.

Question 3:

Code Written:
A function called Option_g was created as follows:

```
function [V] = option_g(S0,sigma,T,r,K )
%Function to be referenced in main script to generate the Option Greeks
M = 500;% Time Steps
delta = T / M;
u=exp(r*delta)*(1+sqrt(exp((sigma^2)*delta)-1));% Up movement
d=exp(r*delta)*(1-sqrt(exp((sigma^2)*delta)-1));%Down Movement
p=0.5;% probability
V = max(S0 * ( (d.^(0:M)) .* (u.^(M:(-1):0)) )' - K, 0);% Call Price at each node
for m = 1:M
   V = 1/(1 + r * delta ) * ( V(1:(end-1)) * p + V(2:end) * (1-p) );%Call Price at the First Node
end

V;
```

This Function was later referenced in the main Script to Compute the Greeks:

```
% Question 3: OPTION GREEKS

epsilon = 0.0001; %Is the changes in variables for different greeks
r = 0.03; % risk free interest rate as given in the question
sigma = 0.2;% Volatility as given in the question
K = 50; %Strike price as given in the question
T = 0.3846; %Time to maturity as given in the question

%Calculate the greeks for different S0 = i
S0_set = 10:2:80; % Setting Stock Price to vary from $10 to $80 with an increment of $2

delta = zeros(1, size(S0_set,2)); %Vector of deltas for different S0
gamma = zeros(1, size(S0_set,2)); %Vector of gammas for different S0
theta = zeros(1, size(S0_set,2)); %Vector of thetas for different S0
vega = zeros(1, size(S0_set,2)); %Vector of vegas for different S0
rho = zeros(1, size(S0_set,2)); %Vector of rhos for different S0

% looping for different step sizes
for i = 1:size(S0_set,2)
   S0 = S0_set(i);

   %Delta: First Option Greek which measures Sensitivity w.r.t. an
   %incremental change in stock price
   S1 = S0; %Stock price
   S2 = S0 + epsilon; %Stock price if changed little bit
   CofS1 =option_g(S1,sigma,T,r,K ); %Price of a call for certain stock price
   CofS2 = option_g(S2,sigma,T,r,K ); %Price of a call for little change in stock price
```

```
    delta1 = (CofS2 - CofS1) / epsilon; %Calculation of delta (slope)
    delta(i) = delta1;
    %The calculations for other greeks a very similar to the delta one.
    %Gamma: Second Option Greek which measures sensitivity of delta w.r.t
    %stock price
    S3 = S0 - epsilon;
    CofS3 = option_g(S3,sigma,T,r,K );
    gamma(i) = (CofS2 - 2*CofS1 + CofS3) / epsilon^2;

%   Theta: Third Option Greek which measures the sensitivity of Call price
%   w.r.t. Change of Time
    T1 = T;
    T2 = T + epsilon;
    CofT1 = option_g(S0,sigma,T1,r,K );
    CofT2 =option_g(S0,sigma,T2,r,K );
    theta(i) = (CofT2 - CofT1) / epsilon;

    %Vega: Fourth Option Greek which measures sensitivity of Call Price
    %w.r.t. change in Volatility
    sigma1 = sigma;
    sigma2 = sigma + epsilon;
    CofSigma1 = option_g(S0,sigma1,T,r,K );
    CofSigma2 = option_g(S0,sigma2,T,r,K );
    vega(i) = (CofSigma2 - CofSigma1) / epsilon;

%     Rho: Fifth Option Greek which measures sensitivity of call price
%     w.r.t change in interest rate
    r1 = r;
    r2 = r + epsilon;
    CofR1 = option_g(S0,sigma,T,r1,K );
    CofR2 = option_g(S0,sigma,T,r2,K );
    rho(i) = (CofR2 - CofR1) / epsilon;

end;
% %Draw the greeks for different stock price
subplot(2,2,1)
%figure(1)
plot(S0_set , delta, '-r', S0_set , gamma, '-b', S0_set , theta, '-g', S0_set , vega, '-k', S0_set , rho, '-
c', 'linewidth',2);
legend('Delta', 'Gamma', 'Theta', 'Vega', 'Rho');
Title('Option Greeks');
Xlabel('Stock Price Dynamics');
%Question 2ii:

T_set = 0:.01:.3846;% Time Increments
S0 = 49;% Initial Stock Price as given
```

```
delta = zeros(1, size(T_set,2)); %Vector of deltas for different T
for i = 1:size(T_set, 2)
    T = T_set(i);
    %Delta
    S1 = S0; %Stock price
    S2 = S0 + epsilon; %Stock price if changed little bit
    CofS1 =option_g(S1,sigma,T,r,K ); %Price of a call for certain stock price
    CofS2 = option_g(S2,sigma,T,r,K ); %Price of a call for little change in stock price
    delta1 = (CofS2 - CofS1) / epsilon; %Calculation of delta (slope)
    delta(i) = delta1;
end
% Draw the change of delta w.r.t Change of Time t
subplot(2,2,2)
plot(T_set, delta)
Title('Delta Dynamics w.r.t. Time');
Xlabel('Time Dynamics');
```

Figure 1: Option Greeks

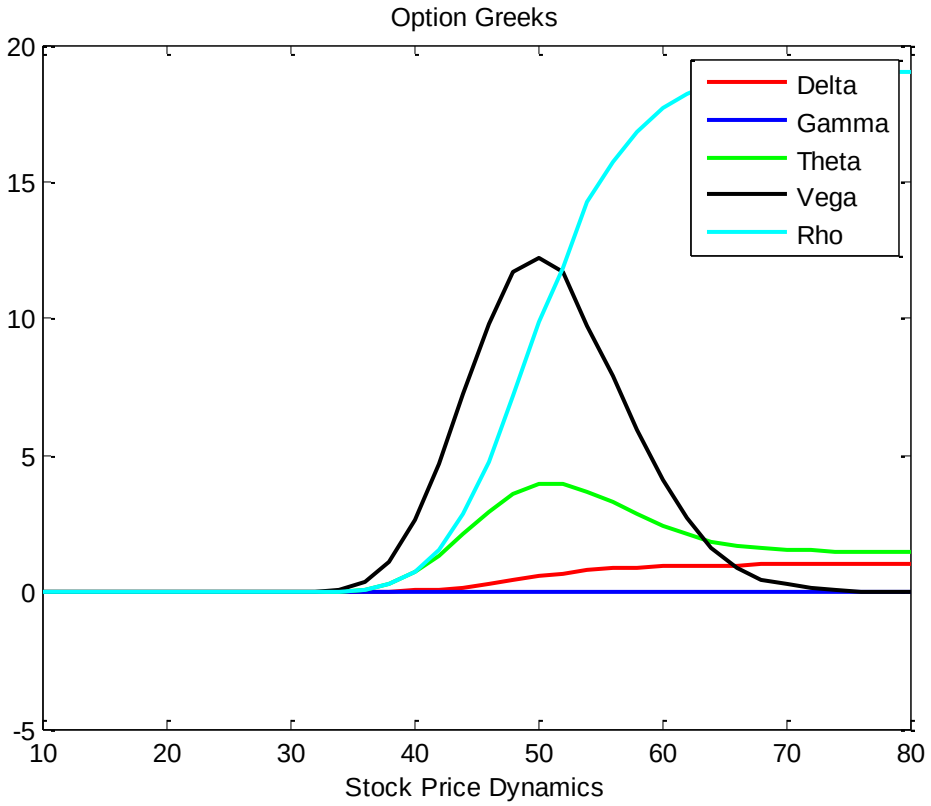
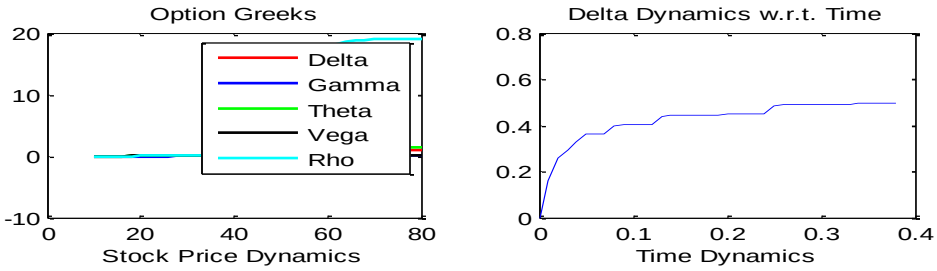


Figure 2: Using Subplot of plotting both Figures in one:



Values Obtained:

Delta	Gamma	Theta	Vega	Rho
0	-4.28E-47	7.66E-39	2.90E-38	1.39E-39
0.156585	1.12E-38	2.55E-30	9.66E-30	5.16E-31
0.260287	4.71E-32	7.32E-24	2.77E-23	1.66E-24
0.292307	-3.76E-28	5.15E-19	1.94E-18	1.30E-19
0.325687	-2.22E-23	4.20E-15	1.58E-14	1.18E-15
0.360286	2.21E-20	2.93E-12	1.10E-11	9.04E-13
0.361886	3.23E-19	9.75E-10	3.65E-09	3.37E-10
0.363359	-3.31E-16	9.41E-08	3.51E-07	3.63E-08
0.398819	-1.19E-14	3.28E-06	1.22E-05	1.40E-06
0.400144	8.47E-14	7.53E-05	0.000279	3.61E-05
0.4014	2.71E-12	0.000801	0.002953	0.000427
0.402594	9.22E-11	0.006308	0.023121	0.003806
0.403737	1.30E-10	0.028318	0.10322	0.019012
0.43982	-1.56E-09	0.103491	0.374517	0.078521
0.440895	-2.78E-09	0.308534	1.105978	0.269227
0.441932	6.94E-09	0.751997	2.661395	0.770174
0.442935	2.78E-08	1.340565	4.68414	1.573679
0.443908	6.66E-08	2.105802	7.233656	2.886216
0.444853	-4.44E-08	2.92184	9.805681	4.774835
0.445772	-8.88E-08	3.593771	11.67169	7.169073
0.446668	-1.78E-07	3.940301	12.20204	9.843518
0.447542	-1.78E-07	3.965355	11.69933	11.84001
0.448395	-2.66E-07	3.64427	9.745963	14.23426

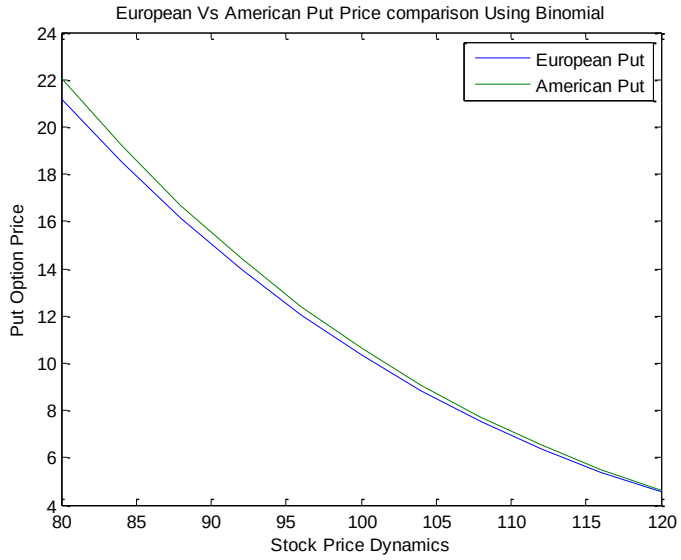
0.44923	9.77E-07	3.272828	7.876116	15.70472
0.450047	-1.60E-06	2.852127	5.914526	16.84956
0.486402	-7.11E-07	2.452133	4.127145	17.67927
0.487192	1.07E-06	2.118501	2.676024	18.23898
0.487967	0	1.869312	1.612192	18.59036
0.488728	-1.78E-06	1.700774	0.902389	18.79565
0.489475	-2.13E-06	1.596851	0.469216	18.90725
0.49021	-3.55E-07	1.554571	0.29402	18.9492
0.490933	-2.49E-06	1.515944	0.135136	18.98361
0.491644	-2.49E-06	1.501981	0.07794	18.99526
0.492344	-1.07E-06	1.490622	0.03165	19.00394
0.493033	1.78E-06	1.486967	0.016797	19.00659
0.493713	2.49E-06	1.48495	0.00862	19.00799

Question 4:

Code Written:

```
% Question 4: Comparison of European Vs. American Put Pricing
% Main Script to call functions created for European & American Put
clear all

r = .03;% Risk free interest rate as given in the question
T = 1;% Total Time
sigma = .3; % Volatility
K = 100; % Exercise or Strike Rate

S0_set = (80:4:120)'; % Set of Stock Price to vary from $80 to $120 with an increment of $4

APs = nan(size(S0_set, 1), 1); % Preallocating memory space for American Puts
EPs = nan(size(S0_set, 1), 1);%Preallocating memory space for European Puts

for i = 1:size(S0_set, 1) % Looping over stock price
   EPs(i) = CF4_EP(S0_set(i),T,r,sigma,K );% European Put Prices
   APs(i) = CF4_AP(S0_set(i),T,r,sigma,K );% American Put Prices
end
% Plotting the European Vs American
plot(S0_set, [EPs, APs])
Legend('European Put', 'American Put');
Xlabel('Stock Price Dynamics');
Ylabel('Put Option Price');
Title('European Vs American Put Price comparison Using Binomial');
```

Figure: European Vs American Put Pricing



The above figure again confirms that American Option is of more worth than European ones.

Question 5:
Part A: Steps Taken:

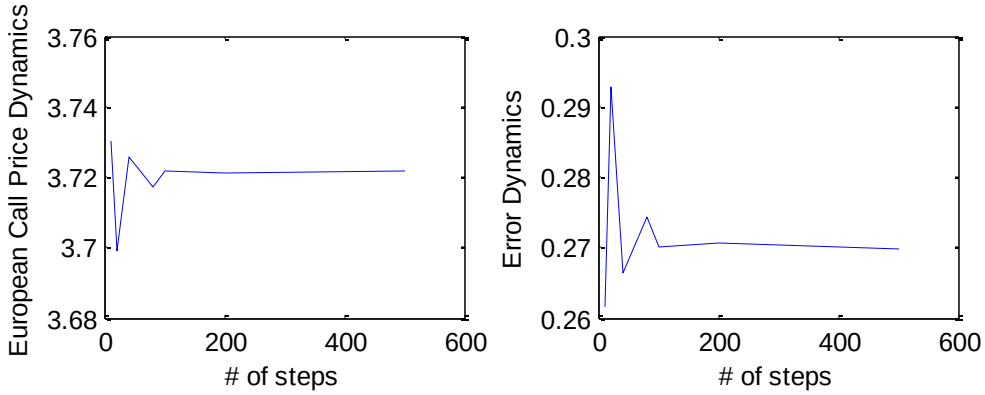
1. Creating a function called CF4_Trinomial

```
% Sets up the asset movements on the trinomial tree
function [Trinomial] = CF4_Trinomial(S, K, r, T, sigma, nSteps)
%Question5, PartA: European Call Pricing Using Trinomial Method
delta = T / nSteps;  % Allocates the time steps

%%% Specifies the probability of up, down and mid moves for trinomial tree
d=exp((-1)*sigma*sqrt(3*delta));
u=1/d;
pd=(r*delta*(1-u)+(r*delta)^2 + (sigma^2)*delta)/((u-d)*(1-d));
pu=(r*delta*(1-d)+(r*delta)^2 + (sigma^2)*delta)/((u-d)*(u-1));
pm=(1-pu-pd);
Df = exp(-r * delta);

% Sets up the asset movements on the trinomial tree
for i = 0:(2 * nSteps)
    State = i + 1;
    Value(State) = max(0, (S * u ^ max(i - nSteps, 0) * d ^ max(nSteps * 2 -
nSteps - i, 0) - K));
end

% Works backwards recursively to determine the price of the option
for l = (nSteps - 1):-1:0 % Going back one step
    for i = 0:(l * 2)
        State = i + 1;
        Value(State) = (pu * Value(State + 2) + pm* Value(State + 1) + pd *
Value(State)) * Df;
```

```
    end
end

Trinomial = Value(1);

end
```

2. Main Script to call the above function & to Loop over different time steps to obtain European Call Prices

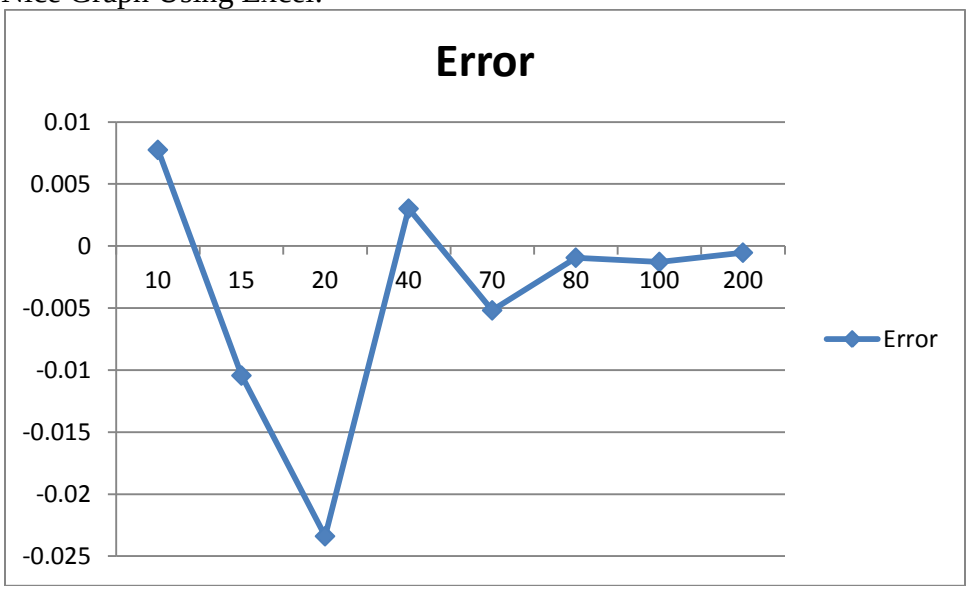
```
% Main Script to Call Function Trinomial
% Calculate Different European Call Values
clear all

nSteps = [10, 20, 40, 80, 100, 200, 500]';% Different Time Steps as specified in the problem
vals = nan(size(nSteps)); % Pre allocating memory space for Different European Option Values
obtained through Trinomial Pricing

S=32;
r=0.05;
sigma=0.24;
K=30;
T=0.5;
Acc_val=blsprice(32, 30,0.05, 0.6, 0.24);

for i = 1:size(nSteps,1)

vals(i)=CF4_Trinomial(S, K, r, T, sigma, nSteps(i));
error(i)=Acc_val-vals(i);

end
% Plotting European Call Prices w.r.t # of steps
subplot(2,2,1);

plot(nSteps,vals);
%Legend('European Call Price dynamics with # of steps');
Xlabel('# of steps');
Ylabel('European Call Price Dynamics');

subplot(2,2,2);

plot(nSteps,error);
%Legend('Error Dynamics # of steps');
Xlabel('# of steps');
Ylabel('Error Dynamics');
```



Convergence Rate:

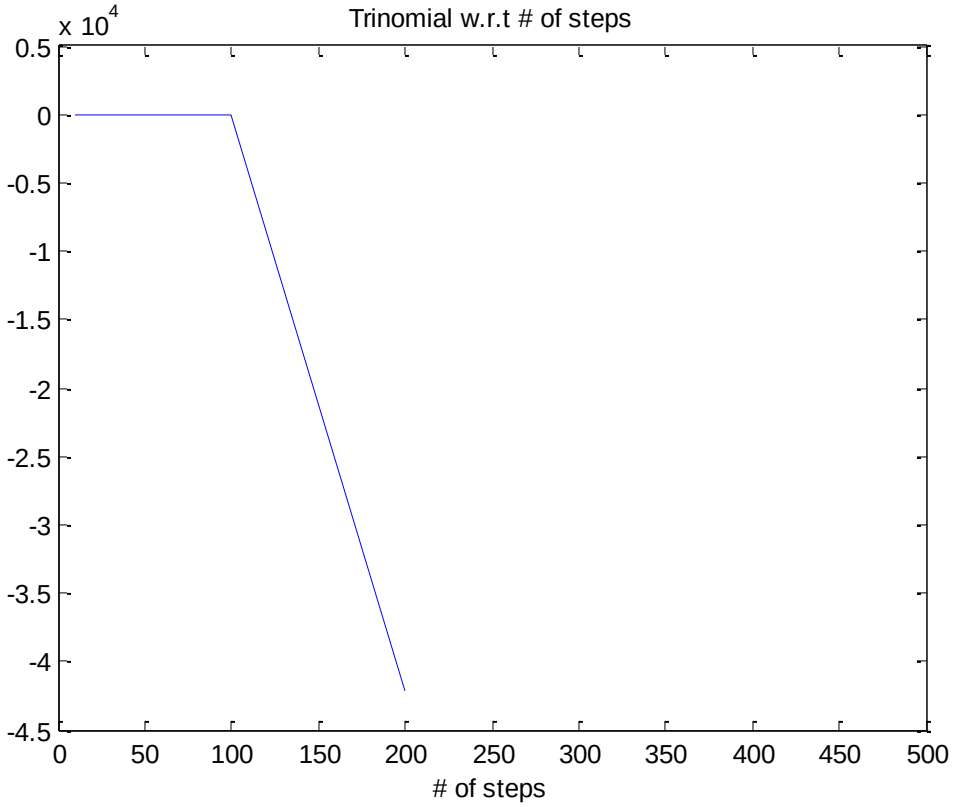
Value by Trinomial	Value by blsprice	Error
3.73015282991791	3.7224	0.007753
3.71196133849364	3.7224	-0.01044
3.69900927973654	3.7224	-0.02339
3.72541532068834	3.7224	0.003015
3.71721577479824	3.7224	-0.00518
3.72146108941862	3.7224	-0.00094
3.72112079787200	3.7224	-0.00128

Nice Graph Using Excel:



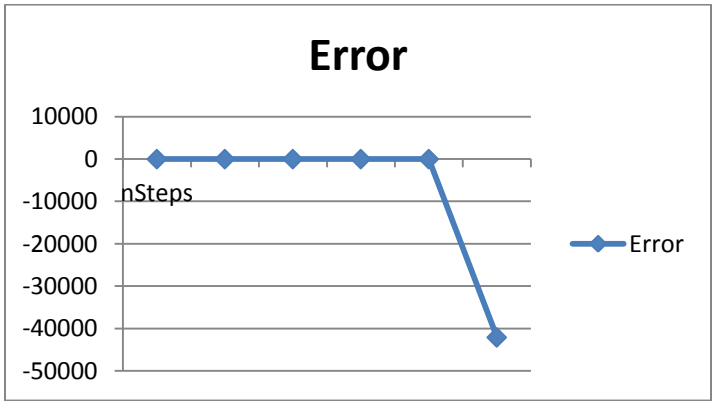
Part B:

Figure: Trinomial log (Stock Price) w.r.t # of steps



Convergence Rate:

Value obtained by Trinomial	Value obtained by BLSprice	Error
0.255164612647873	2.1590	-1.90384
0.580929019538946	2.1590	-1.57807
1.17474876242542	2.1590	-0.98425
2.33345996551427	2.1590	0.17446
2.91044304474064	2.1590	0.751443
-42086.3292510632	2.1590	-42088.5
-Inf	2.1590	



Question 6:

Steps Taken:

1. Wrote a Function called **GetHalton.m**

```
function Seq = GetHalton(HowMany, Base)
Seq = zeros(HowMany,1);
NumBits = 1+ceil(log(HowMany)/log(Base));
VetBase = Base.^(-(1:NumBits));
WorkVet = zeros(1,NumBits);
for i=1:HowMany
j=1;
ok = 0;
while ok == 0
WorkVet(j) = WorkVet(j)+1;
if WorkVet(j) < Base
ok = 1;
     else
        WorkVet(j) = 0;
        j = j+1;
  end
```

```
    end
    Seq(i) = dot(WorkVet,VetBase);
end
```

2. Main Script to call GetHalton and Finally price the European Call

```
%Question 6: Using Halton's Low-Discrepancy Sequences to price European call
options
%Halton's Low-Discrepancy Sequence
clear all;
N_vec = 100; % Length of vector
seed = 98242;% Use the seed
trials = 200;% # of iterations
haltonSeqB = zeros(N_vec/2 , 2);% Create The Vector
b1=2; % First Halton Series of Base 1
b2=7; % First Halton Series of Base 1
r = 0.05; % interest rate as given
sigma = 0.24; % volatility rate as given
S0=32;% Initial Stock Price as given
K=30; % Strike or Exercise Rate as given
T=0.5; % Time as given in the problem
 seqBase_b1 = GetHalton(N_vec/2 , b1); % Use the function GetHalton (base b1) as
defined
seqBase_b2 = GetHalton(N_vec/2 , b2);% Use the function GetHalton (base b2) as
defined

for i = 1:N_vec/2 % looping Over the length of the vector
    haltonSeqB(i, 1) = seqBase_b1(i);
    haltonSeqB(i, 2) = seqBase_b2(i);
end;
%Box-Muller to generate Normal Distribution
  z1(:,1) = sqrt(-2*log(haltonSeqB(:, 1))).*cos(2*pi*haltonSeqB(:, 2));
  z2(:,1)= sqrt(-2*log(haltonSeqB(:, 1))).*sin(2*pi*  haltonSeqB(:, 2));
z=([z1; z2]);% Stacking the Normals
% Call Pricing
call= exp(-r*T)*max((S0*exp((r-(sigma^2)/2)*T)+sigma*z)-K,0);
% Final Call Value
mean(call);
```

Final Call Price with the given Parameters: **2.2870**

Computational Methods in Finance

Homework # 5

Submitted by

Maitreyi Mandal

Q1. Steps Taken:

1. 3 Different Functions named Hermite, Laguerre and Monomial were written.
2. The main script was following:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Computational Finance Homework#5
%Question#1A: Estimation of American Put Option by Least-Square
%MonteCarlo Using Different Basis Functions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all
% Parameters Given in the question
      T = 0.5;% Time Period
      r = .06;% Interest Rate
      S0 = 36;% Initial Stock Price As Provided
      K = 40; % Strike Price As Provided
      sigma = .2;% Volatility As Provided
      N = 200;
      M = 50000;% # of paths
      z1=randn(M,N);% Gdenerating normal random number
      z2=randn(M,N)*(-1);% Antithetic Variable

      h = T / N;% Discretization of time step

      diffusionp = [zeros(M, 1) sigma * sqrt(h) * cumsum( z1, 2)];%
   Calculating the diffusion component
      diffusionm=[zeros(M, 1) sigma * sqrt(h) * cumsum( z2,
   2)];%antithetic var
      drift     = [zeros(M, 1) repmat((r-sigma^2/2)*h*(1:N), M, 1) ];%
   Calculating the drift Component
      % Capturing Stock Price Dynamics
      Sp = S0*exp(drift + diffusionp);
      Sm = S0*exp(drift + diffusionm);
      S=[Sp;Sm];
      dmat = repmat( exp(-r*h*(0:N)), M*2, 1);
      ind = zeros(M*2, N+1);% Index Generation
      ind(:, end) = ( (K - S(:, end))>0 );
      val = max(0, K - S);
      % Running the Itearation
      n = N;
      for n = N:(-1):1
        if( mod(n, 50) == 0 )
          disp(['Iteration ' num2str(n) ' complete'])% Just a check to
          %see if the code is running
        end
        V = val(:,n);
        itm = (V > 0);
        % Continuation Value

        CV = sum( val .* dmat .* ind, 2);
```

```
45.
46.         bases = Hermite(S(:,n), 3);
47.
48.         Y = CV(itm,:);
49.         X = bases(itm,:);
50.         beta = regress(Y,X);% Running the regression
51.         ECV = bases * beta;
52.
53.         ex = (itm & ( ECV < V) );
54.         ind(ex,:) =0;
55.         ind(ex, n) = 1;
56.
57.       end
58.
59.       mean(  sum( val .* dmat .* ind, 2) ) % Finally Calculating the
    American Put Pricing
```

The following tables represent the values that were obtained:

General Trend: Values obtained using Laguerre Polynomial was consistently lower than the values obtained using other two. But all the values were very close to the value obtained by American Put Prices by Binomial Pricing. Also using the higher order polynomial seemed the values obtained were closer to the binomial prices, hence the error was reduced. Again it was checked against European put options and was found that European options are of lesser value than American Puts. Also, from the results it was observed that keeping the stock price constant, value of the option increases with maturity, price decreases with increasing current stock price which are consistent with American Option Pricing theory.

Q2. European Put Pricing

LSMC is not optimized for European put pricing. Hence either a regular MC or Binomal/Trinomial can be used. I wrote a binomial European Put Pricing formula as follows:

```
function [EP] = CF4_EP(S0,T,r,sigma,K )
% Question 4
% Estimation of European and American Put Pricing
% European Put Pricing
M = 500;% No of Time Steps
delta =T/M;
p = .5 + r * sqrt(delta)/(2*sigma);%
u = 1 + sigma * sqrt(delta);
d = 1 - sigma * sqrt(delta);
EP = max(S0 * (-1)*( (d.^(0:M)) .* (u.^(M:(-1):0)) )' +K, 0);
for m = 1:M
   EP = 1/(1 + r * delta ) * ( EP(1:(end-1)) * p + EP(2:end) * (1-p) );
end
EP;
End
```

Table 2: European Put Pricing As Asked in Question 2

r=0.06	sigma=0.2	K=40			
	European Put Pricing			Binomial	BlsPrice
		Maturity=0.5	S0=36	3.809067453	3.809586554
			S0=40	1.680465938	1.680179764
			S0=44	0.601578902	0.601194708
		Maturity=1	S0=36	3.842473146	3.844307792
			S0=40	2.06601405	2.066401004
			S0=44	1.017249995	1.016915226
		Maturity=2	S0=36	3.763302604	3.763000928
			S0=40	2.35734055	2.355866282
			S0=44	1.429410412	1.429215131

3. Forward start option pricing: Forward start options are path dependent options that have strike prices to be determined at a future date.

a. Forward-start European put option is quite easy to value:

For this I wrote the following code:

```
%Question 3a: Forward Start European Put Pricing
```

```
S0=65;%Initial Stcok Price as asked in the question
sigma=.2;% Volatility as provided
r=.06; % Risk Free Interest Rate
t=.2; %small t
T=1;% Big T, entire Time Interval
N=1000000; %# of simulations
St=S0*exp((r-(sigma^2)/2)*t+sqrt(t)*sigma*randn(N,1));% Stock Price at St

% % [Call,Put] = blsprice(Price, Strike, Rate, Time, Volatility)
%
 [C,P] = blsprice(St,St,r,T-t,sigma);% Just using in-built
 %blsprice to compute Forward Start European Put Option Price
 mean(P) % Forward Start European Put Price
```

Value I got: 3.200532385080308

b. Forward Start American Put Pricing:

Part A: I wrote the following code

```
Smin=min(St);% Checking what the minimum stock price is
Smax=max(St);% Checking what the maximum stock price is
nbin=200;
Amputstrike=linspace(Smin,Smax,nbin)'; % Creating a grid with nbin for Strike
Prices
Amputprice=nan(nbin,1); % Defining the American Put Price
for i=1:nbin
    Amputprice(i,1)=CF4_AP(Amputstrike(i,1),T-t,r,sigma,Amputstrike(i,1) );
end
% CF4_AP was previously written for assignmnet 4
counts=nan(nbin-1,1);
for j = 2:nbin
    counts(j-1,1)=sum((St>=Amputstrike(j-1,1))&((St<=Amputstrike(j,1))));
end
% summing up how many times St>= and <= Amputstrike
binmidpoints = (Amputprice(1:(end-1),1) + Amputprice(2:end,1))/2;

weights = counts / sum(counts); % Creating weights
sum( weights .* binmidpoints )% Forward Start American Put Price
```

Part B: Binomial American Put Pricing

```
function [AP] = CF4_AP(S0,T,r,sigma,K )

% Question 4
Part A: Use the Binomial Put Pricing for American Option
% American Put Pricing
M = 500;
delta =T/M;
p = .5 + r * sqrt(delta)/(2*sigma);
u = 1 + sigma * sqrt(delta);
d = 1 - sigma * sqrt(delta);
AP = max(S0 * (-1)*( (d.^(0:M)) .* (u.^(M:(-1):0)) )' +K, 0);
```

```
for m = 1:M
    CV = 1/(1 + r * delta ) * ( AP(1:(end-1)) * p + AP(2:end) * (1-p) );
    EV = max( K - S0.*(d.^(0:(M-m))).*(u.^((M-m):-1:0)),0 )';
    AP = max(CV, EV);
end
AP;end
```

Part B: Modify for Forward Start American Put Pricing

```
Smin=min(St);% Checking what the minimum stock price is
Smax=max(St);% Checking what the maximum stock price is
nbin=200;
Amputstrike=linspace(Smin,Smax,nbin)'; % Creating a grid with nbin for Strike Prices
Amputprice=nan(nbin,1); % Defining the American Put Price
for i=1:nbin
   Amputprice(i,1)=CF4_AP(Amputstrike(i,1),T-t,r,sigma,Amputstrike(i,1) );
end
% CF4_AP was previously written for assignmnet 4
counts=nan(nbin-1,1);
for j = 2:nbin
   counts(j-1,1)=sum((St>=Amputstrike(j-1,1))&((St<=Amputstrike(j,1))));
end
% summing up how many times St>= and <= Amputstrike
binmidpoints = (Amputprice(1:(end-1),1) + Amputprice(2:end,1))/2;

weights = counts / sum(counts); % Creating weights
sum( weights .* binmidpoints )% Forward Start American Put Price
```

Final value for Forward Start American Put Pricing: 3.525687927831950

The code should be working right, since American Forward Start Option > European Forward Start Option.

Initial Stock Price	Time to Expiry	Type of PolyNomial	No of polynomials	Price LSMC	American Put Price Binomial	European Put Price BlackScholes
36	0.5	'Hermite'	2	4.189329441	4.207812688	3.809586554
36	0.5	'Laguerre'	2	4.012456892	4.207812688	3.809586554
36	0.5	'Monomial'	2	4.187659974	4.207812688	3.809586554
36	0.5	'Hermite'	3	4.200829195	4.207812688	3.809586554
36	0.5	'Laguerre'	3	4.050081513	4.207812688	3.809586554
36	0.5	'Monomial'	3	4.194182963	4.207812688	3.809586554
36	0.5	'Hermite'	4	4.184485375	4.207812688	3.809586554
36	0.5	'Laguerre'	4	4.164117465	4.207812688	3.809586554
36	0.5	'Monomial'	4	4.19710069	4.207812688	3.809586554
36	1	'Hermite'	2	4.445501994	4.487602325	3.844307792
36	1	'Laguerre'	2	4.049878723	4.487602325	3.844307792
36	1	'Monomial'	2	4.461780689	4.487602325	3.844307792
36	1	'Hermite'	3	4.464365433	4.487602325	3.844307792
36	1	'Laguerre'	3	4.061521323	4.487602325	3.844307792
36	1	'Monomial'	3	4.470868438	4.487602325	3.844307792
36	1	'Hermite'	4	4.44781113	4.487602325	3.844307792
36	1	'Laguerre'	4	4.234744356	4.487602325	3.844307792
36	1	'Monomial'	4	4.429031247	4.487602325	3.844307792
36	2	'Hermite'	2	4.749910786	4.848265107	3.763000928
36	2	'Laguerre'	2	4.047658921	4.848265107	3.763000928
36	2	'Monomial'	2	4.783556865	4.848265107	3.763000928
36	2	'Hermite'	3	4.790558621	4.848265107	3.763000928
36	2	'Laguerre'	3	4.084075978	4.848265107	3.763000928
36	2	'Monomial'	3	4.786477703	4.848265107	3.763000928
36	2	'Hermite'	4	4.749179196	4.848265107	3.763000928
36	2	'Laguerre'	4	4.277958891	4.848265107	3.763000928

36	2	'Monomial'	4	4.737988265	4.848265107	3.763000928
40	0.5	'Hermite'	2	1.694321358	1.800978304	1.680179764
40	0.5	'Laguerre'	2	1.571703745	1.800978304	1.680179764
40	0.5	'Monomial'	2	1.694321358	1.800978304	1.680179764
40	0.5	'Hermite'	3	1.791428384	1.800978304	1.680179764
40	0.5	'Laguerre'	3	1.688085251	1.800978304	1.680179764
40	0.5	'Monomial'	3	1.691428384	1.800978304	1.680179764
40	0.5	'Hermite'	4	1.728483326	1.800978304	1.680179764
40	0.5	'Laguerre'	4	1.763162688	1.800978304	1.680179764
40	0.5	'Monomial'	4	1.728483326	1.800978304	1.680179764
40	1	'Hermite'	2	2.239443	2.316502823	2.066401004
40	1	'Laguerre'	2	1.385145848	2.316502823	2.066401004
40	1	'Monomial'	2	2.239443	2.316502823	2.066401004
40	1	'Hermite'	3	2.123812965	2.316502823	2.066401004
40	1	'Laguerre'	3	1.915130087	2.316502823	2.066401004
40	1	'Monomial'	3	2.123812965	2.316502823	2.066401004
40	1	'Hermite'	4	2.193995235	2.316502823	2.066401004
40	1	'Laguerre'	4	2.087679925	2.316502823	2.066401004
40	1	'Monomial'	4	2.193995235	2.316502823	2.066401004
40	2	'Hermite'	2	2.504883909	2.894054898	2.355866282
40	2	'Laguerre'	2	1.505465242	2.894054898	2.355866282
40	2	'Monomial'	2	2.504883909	2.894054898	2.355866282
40	2	'Hermite'	3	2.56768357	2.894054898	2.355866282
40	2	'Laguerre'	3	1.834554838	2.894054898	2.355866282
40	2	'Monomial'	3	2.56768357	2.894054898	2.355866282
40	2	'Hermite'	4	2.634123155	2.894054898	2.355866282
40	2	'Laguerre'	4	2.143924835	2.894054898	2.355866282
40	2	'Monomial'	4	2.634123155	2.894054898	2.355866282
44	0.5	'Hermite'	2	0.456870486	0.628958701	0.601194708

44	0.5	'Laguerre'	2	0.5074949	0.628958701	0.601194708
44	0.5	'Monomial'	2	0.456870486	0.628958701	0.601194708
44	0.5	'Hermite'	3	0.525289809	0.628958701	0.601194708
44	0.5	'Laguerre'	3	0.612836215	0.628958701	0.601194708
44	0.5	'Monomial'	3	0.525289809	0.628958701	0.601194708
44	0.5	'Hermite'	4	0.57701841	0.628958701	0.601194708
44	0.5	'Laguerre'	4	0.612393759	0.628958701	0.601194708
44	0.5	'Monomial'	4	0.57701841	0.628958701	0.601194708
44	1	'Hermite'	2	0.849819747	1.116332961	1.016915226
44	1	'Laguerre'	2	0.606487679	1.116332961	1.016915226
44	1	'Monomial'	2	0.849819747	1.116332961	1.016915226
44	1	'Hermite'	3	0.954229049	1.116332961	1.016915226
44	1	'Laguerre'	3	0.870330093	1.116332961	1.016915226
44	1	'Monomial'	3	0.954229049	1.116332961	1.016915226
44	1	'Hermite'	4	1.02074242	1.116332961	1.016915226
44	1	'Laguerre'	4	1.053170564	1.116332961	1.016915226
44	1	'Monomial'	4	1.02074242	1.116332961	1.016915226
44	2	'Hermite'	2	1.324688295	1.690247158	1.429215131
44	2	'Laguerre'	2	0.74157781	1.690247158	1.429215131
44	2	'Monomial'	2	1.324688295	1.690247158	1.429215131
44	2	'Hermite'	3	1.433539943	1.690247158	1.429215131
44	2	'Laguerre'	3	1.015347971	1.690247158	1.429215131
44	2	'Monomial'	3	1.433539943	1.690247158	1.429215131
44	2	'Hermite'	4	1.504965169	1.690247158	1.429215131
44	2	'Laguerre'	4	1.273564108	1.690247158	1.429215131
44	2	'Monomial'	4	1.504965169	1.690247158	1.429215131

Computational Methods in Finance

Homework #6

Finite Difference Methods

Submitted By

Maitreyi Mandal

PROJECT 6 OF COMPUTATIONAL FINANCE

THE WORKFLOW:

The entire project was divided into writing a total of four functions and a main script to call all the functions. The description of the functions is as follows:

1. IFD.m: This function is written to evaluate option prices using Implicit Finite Difference Method.
2. EFD.m: This function is written to evaluate option prices using Explicit Finite Difference Method.
3. CrankNicolson.m: This function is written to evaluate option prices using Crank-Nicolson Finite Difference Method.
4. StockDynamics_FD.m: This function is written to generate the underlying stock price dynamics.
5. CF_HW6.m: This is the main script to call all the above functions.

Function 1: IFD.m

```
function [ Prices, St, gridS] = IFD( type, callput, S0, K, r, sigma, T,
delta_x, delta_t, Smax, Smin )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Function IFD is written to calculate option pricing using Implicit Finite
%Difefrence Method where the inputs are as follows:
%Type American (A) or European (E)
%callput is Call (C) or Put (P)
%S0 is stock price, K is strike price, r is interest rate, sigma is
%volatility, T is time to maturity
%Delta_x is the increment in stock price based on X = ln(S)
%Delta_t is the time step size in the grid
%Smax and Smin are the maximum and minimum of the underlying prices we want
%to find the option prices for.
%OUTPUT of IFD are as follows:
%OptionPrices is the prices of all the options in the grid at each time
%step for different underlying price.
%St is the prices of the underlying security
%gridS is the row in the grid where the current stock price is
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
columns = ceil(T/delta_t);%How many columns/steps we need.
%FOR IFD WE ONLY NEED LOG OF STOCK PRICES
[Xt, rows, gridS] = StockDynamics_FD(log(S0), log(Smin), log(Smax), delta_x,
0);
St = exp(Xt);%Change the prices from log
%Matrices we need to use to get the option prices for every row/node at
%each time step.
A = zeros(rows, rows);
opt = zeros(rows, 1); %Option prices at time t
b = zeros(rows, 1); %Option prices at time t + 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%IMPLICIT FINITE DIFFERENCE PARAMETERS
```

```
%Implicit X, Pu, Pm and Pd based on when prices are generated with X = ln(S)
v = r - sigma^2/2;%Nu as given with transformed Black-Scholes PDE
Pu = (-0.5)*delta_t*(sigma^2/delta_x^2+v/delta_x);
Pm = 1+delta_t*sigma^2/(delta_x^2)+r*delta_t;
Pd = (-0.5)*delta_t*(sigma^2/delta_x^2-v/delta_x);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Fill the A array according to the notes for IFD and appropriate Boundary
%Conditions
for j = 1:rows
    if(j == 1)
        A(j,j) = 1; A(j,j+1) = -1;
    elseif (j == rows)
        A(j,j-1) = 1; A(j,j) = -1;
    else
        A(j,j-1) = Pu; A(j,j) = Pm; A(j,j+1) = Pd;
    end;
end;
%Option prices at maturity
if(callput == 'C')
   opt = max(St - K, 0);%%%%% FOR CALL
else
   opt = max(K - St, 0); %%%%FOR PUT
end;
%%%%Here we start the process of backward iteration which is calculating
%%%%the price in each time step and node backward
for i = columns:-1:1
    EV = 0;%Exercise value
    b = opt;%Option value at time t+1
    if(i ~= columns)%If last column(maturity) then skip. Caluclated above
        for j = 1:rows
            %Set bondaries based on Call(C) or Put(P)
            if(callput == 'C')
                if(j == 1)
                    b(1) = St(1) - St(2);
                elseif(j == rows)
                    b(rows) = 0;
                end;
                EV = St(j) - K;
            else
                if(j == 1)
                   b(1) = 0;
                elseif(j == rows)
                    b(rows) = -(St(rows) - St(rows-1));
                end;
                EV = K - St(j);
            end;
            %The AMERICAN CONDITION
            if(type == 'A' && EV > b(j)); b(j) = EV; end;
        end;
    end;
opt = A\b; %Get option prices at time t
end;
Prices = opt; %Return option prices
end
```

Function 2: EFD.m

```
function [Prices, St, gridS] = EFD( type, callput, S0, K, r, sigma, T, delta,
delta_t, Smax, Smin, logPrice )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%This function called EFD is written to evaluate option pricing using
%Explicit Finite Difference Method.The Inputs are as follows:
%Type American (A) or European (E)
%callput is Call (C) or Put (P)
%S0 is stock price, K is strike price, r is interest rate, sigma is
%volatility, T is time to maturity
%Delta is the increment in stock price based on S or X = ln(S)
%Delta_t is the time step size in the grid
%Smax and Smin are the maximum and minimum of the underlying prices we want
%to find the option prices for.
%LogPrices equal to 1 if we are going to use X = ln(S) to generate the stock
%prices else equal to 0 if using S.
%OUTPUT of the Function EFD:
%OptionPrices is the prices of all the options in the grid at each time
%step for different underlying price.
%St is the prices of the underlying security
%gridS is the row in the grid where the current stock price is
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
columns = ceil(T/delta_t);%Time steps
%CASE I:If we are going to use X = ln(S) or S to generate stock prices
if(logPrice == 1)
    [Xt, rows, gridS] = StockDynamics_FD(log(S0), log(Smin), log(Smax),
delta, 0);
  %CASE II: We are going to use ordinary stock price
    St = exp(Xt);%Change the prices from log
else
    [St, rows, gridS] = StockDynamics_FD(S0, Smin, Smax, delta, 0);
end;
opt = zeros(rows, columns); %Create array to capture option prices
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%GENERTION OF THE GRID/MATRIX
%Option prices at maturity
if(callput == 'C')
   opt(:,columns) = max(St - K, 0);%%%%% FOR CALL
else
   opt(:,columns) = max(K - St, 0); %%%%FOR PUT
end;
%%%%%CASE I:
%%%EXPLICIT FINITE DIFFERENCE PARAMETERS WITH X=Ln(S)
%Explicit X, Pu, Pm and Pd based on when prices are generated with X = ln(S)
Pu = delta_t*(sigma^2/(2*delta^2)+(r-sigma^2/2)/(2*delta));
Pm = 1-delta_t*sigma^2/(delta^2); %-r*delta_t;
Pd = delta_t*(sigma^2/(2*delta^2)-(r-sigma^2/2)/(2*delta));
%Here we calculate the price in each time step and node backward.
%We start from the second last column where we have calculated the price in
%last column above.
for i = columns-1:-1:1
    EV = 0; %Exercise value
    %Go through every row in each column
    for j = 1:rows
%%%%%%% CASE II:
```

```
%If we didn't use log prices then we need to have other Pu, Pm and
        %Pd
        if(logPrice == 0)
%%%EXPLICIT FINITE DIFFERENCE PARAMETERS WITH ORIGINAL B-S PDE
            Pu = delta_t*((r*j)/2+(sigma^2*j^2)/2);
            Pm = 1-delta_t*sigma^2*j^2;
            Pd = delta_t*((-r*j)/2+(sigma^2*j^2)/2);
        end

        if(j == 1) %If first row in each column, we need to set bondaries
            if(callput == 'C')
                opt(j,i) = St(j)-K*exp(-r*(columns-i)*delta_t);
                EV = St(j) - K;
            else
               opt(j,i) = 0;
                EV = K - St(j);
            end;
        elseif(j == rows) %Other bondaries for the last row in each column
            if(callput == 'C')
               opt(j,i) = 0;
                EV = St(j) - K;
            else
                opt(j,i) = K*exp(-r*(columns-i)*delta_t);
                EV = K - St(j);
            end;
        else %If not bondaries we calculate according to the explicit formula
            if(callput == 'C'); EV = St(j) - K;
            else EV = K - St(j); end;
           opt(j,i) = (1/(1+r*delta_t))*(Pu*opt(j-
1,i+1)+Pm*opt(j,i+1)+Pd*opt(j+1,i+1));
        end;
        %THE AMERICAN CONDITION
        if(type == 'A' && EV > opt(j,i))
            opt(j,i) = EV;
        end;
    end;
end;
Prices = opt; %return option prices in each node.
end
```

Function 3: CrankNicolson.m

```
function [ Prices, St, gridS] = CrankNicolson( type, callput, S0, K, r,
sigma, T, delta_x, delta_t, Smax, Smin )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%The Function CrankNicolson is written to evaluate the option price using
%Crank-Nicolson Method of Finite Difference, the various INPUTS and OUTPUTS
%are given below:
%INPUT of this finction are:
%Type American (A) or European (E)
%callput is Call (C) or Put (P)
%S0 is stock price, K is strike price, r is interest rate, sigma is
%volatility, T is time to maturity
%Delta_x is the increment in stock price based on X = ln(S)
%Delta_t is the time step size in the grid
```

```
%Smax and Smin are the maximum and minimum of the underlying prices we want
%to find the option prices for.
%OUTPUT of this function is:
%OptionPrices is the prices of all the options in the grid at each time
%step for different underlying price.
%St is the prices of the underlying security
%gridS is the row in the grid where the current stock price is
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
columns = ceil(T/delta_t);%How many columns/steps we need.
%Get log stock prices
[Xt, rows, gridS] = StockDynamics_FD(log(S0), log(Smin), log(Smax), delta_x,
0);
St = exp(Xt);%Change the prices from log
%Matrices we need to use to get the option prices for every row/node at
%each time step.
A = zeros(rows, rows);
opt = zeros(rows, 1);%Option prices at time t
b = zeros(rows, 1);%Option prices at time t + 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Crank-Nicolson PARAMETERS
%Crank-Nicolson X, Pu, Pm and Pd based on when prices are generated with X =
ln(S)
v = r - sigma^2/2;% Nu as found in the transformed Black-Scholes PDE
Pu = (-0.25)*delta_t*(sigma^2/delta_x^2+v/delta_x);
Pm = 1+delta_t*sigma^2/(2*delta_x^2)+r*delta_t/2;
Pd = (-0.25)*delta_t*(sigma^2/delta_x^2-v/delta_x);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%GENERATION OF THE GRID and appropriate Boundary
%Conditions
%Fill the A array according to the notes for IFD
for j = 1:rows
    if(j == 1)
        A(j,j) = 1; A(j,j+1) = -1;
    elseif (j == rows)
        A(j,j-1) = 1; A(j,j) = -1;
    else
        A(j,j-1) = Pu; A(j,j) = Pm; A(j,j+1) = Pd;
    end;
end;
%Option prices at maturity
if(callput == 'C')
    opt= max(St - K, 0);
else
    opt= max(K - St, 0);
end;
%Here we calculate the price in each time step and node backward.
for i = columns:-1:1
    EV = 0;%Exercise value
    b = opt;%Option value at time t+1
    if(i ~= columns)%If last column(maturity) then skip. Caluclated above
        for j = 1:rows
            %Set bondaries based on Call(C) or Put(P)
            if(callput == 'C')
                if(j == 1)
                    b(1) = St(1) - St(2);
                elseif(j == rows)
                    b(rows) = 0;
```

```
                end;
                EV = St(j) - K;
            else
                if(j == 1)
                   b(1) = 0;
                elseif(j == rows)
                    b(rows) = -(St(rows) - St(rows-1));
                end;
                EV = K - St(j);
            end;
            %If not bondaries we calculate according to the Crank-Nicolson
formula
            if(j ~= 1 && j ~= rows)
                b(j) = -Pu*opt(j-1)-(Pm-2)*opt(j)-Pd*opt(j+1);
            end;
            %AMERICAN CONDITION
            if(type == 'A' && EV > b(j)); b(j) = EV; end;
        end;
    end;

    opt= A\b;%Get option prices at time t
end;
Prices = opt;%Return option prices

End
```

Function 4: StockDynamics_CF

```
function [ Xt, rows, gridS] = StockDynamics_FD(X0, Xmin, Xmax, delta_x, u)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Function StockDynamics_FD is written to calculate stock prices for use in
%Finite Difference Methods, the various INPUTS and OUTPUTS are as follows:
%INPUT OF THE FUNCTION StockDynamics_FD:
%X0 is the central price
%Xmax and Xmin are the maximum and minimum of the underlying prices we want
%Delta_x is the increment in stock price
%Usin u is another way to get increment in stock prices
%OUTPUT OF THE FUNCTION StockDynamics_FD:
%Xt includes all the stock prices in the grid at each time
%rows is how many rows are in the grid
%gridS is the row in the grid where the current stock price is
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Get the range we need so we cover all prices(Xmax and Xmin) that we want.
Universe = max(X0-Xmin, Xmax-X0)/delta_x;
N = ceil(Universe); %to make sure it is an integer
rows = 2*N + 1;%to have N on both sides of the grid
gridS = N + 1;
Xt = zeros(rows, 1);
Xt(gridS, 1) = X0;%Set the current price
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Generation of Stock Price Dynamics
for i = 1:N
    if(u == 0)
        Xt(gridS+i,1) = X0 - i*delta_x;
        Xt(gridS-i,1) = X0 + i*delta_x;
    else
```

```
        Xt(gridS+i,1) = X0*u^(-i);
        Xt(gridS-i,1) = X0*u^(i);
    end;
end;

end
```

Main Script:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%Computational Finance Project # 6
%%%Finite Difference Method
%%%%%Submitted by Maitreyi Mandal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Parameter Initialization as per the Given Problem
r = 0.04; %Interest rate
sigma = 0.2; %Volatility
S0 = 10; %Current Stock Price, the middle price
K = 10; %Strike Price
delta_t = 0.002; %Delta t
T = 0.5; %Time to maturity
%Delta X is used for both S and and X. We can only us S for Explicit FDM
delta_x = 1.5;
%If equal to 1 then we are going use ln(X) to generate the stock prices
%if equal to 0 then we use S. Only us S for Explicit FDM
IndicatorX = 0;
%StockPrice Range and the corersponding increment
Smax = 20;
Smin = 1;
increments = 1;
type = 'A'; %A for American, E for European
callput = 'P'; %C for Call and P for Put
%The three Finite Difference Methods that we are using here, need to
%comment out the ones which we are not using:
%[option, St, gridS] = IFD(type, callput, S0, K, r, sigma, T, delta_x,
delta_t, Smax, Smin );
[option, St, gridS] = EFD(type, callput, S0, K, r, sigma, T, delta_x,
delta_t, Smax, Smin, IndicatorX);
%[option, St, gridS] = CrankNicolsonFiniteDifferenceMethod(type, callput, S0,
K, r, sigma, T, delta_x, delta_t, Smax, Smin );
%To get how many prices we have
stockpriceArray = (Smax - Smin + 1) / increments;
%Table to compare prices from different methods
values = zeros(stockpriceArray, 2);
for i = stockpriceArray:-increments:1
    %Aim of this step is to minimize the error to get the price closest to
the prices we are
    %looking for as an underlying stock price which should be between $1 to
    %$20 as given in the problem
    delta = 1000;
    for j = 1:size(St,1)%Prices that we want
        StPrice = St(j);%Prices that we have now
        if(abs(StPrice - i) < delta)
            delta = abs(StPrice - i);
            values (i, 1) = StPrice;%First column is the price of underlying
```

```
            values (i, 2) = option(j);%Second column is the option price from
FDM
                    end;
    end;
end;
vals = values%Get the array of stock price and corresponding option prices as
wanted in the problem.
```

Comparison with BLSPrice and Binomial Tree for American Put:

BLSPRICE

```
S0_set = 1:1:20;,
K=10,
r=0.04;
T=0.5;
sigma=0.2;
for i = 1:size(S0_set,2)

[C,P] = blsprice(S0_set(i),K,r,T,sigma);
end
```

American Put Binomial Price:

```
function [AP] = CF4_AP(S0,T,r,sigma,K )

% Question 4
% American Put Pricing
M = 150;
delta =T/M;
p = .5 + r * sqrt(delta)/(2*sigma);
u = 1 + sigma * sqrt(delta);
d = 1 - sigma * sqrt(delta);
AP = max(S0 * (-1)*( (d.^(0:M)) .* (u.^(M:(-1):0)) )' +K, 0);
for m = 1:M
    CV = 1/(1 + r * delta ) * ( AP(1:(end-1)) * p + AP(2:end) * (1-p) );
    EV = max( K - S0.*(d.^(0:(M-m))).*(u.^((M-m):-1:0)),0 )';
    AP = max(CV, EV);
end
AP;
end
```

looping:

```
S0_set = 1:1:20;,
K=10,
r=0.04;
T=0.5;
sigma=0.2;
for i = 1:size(S0_set,2)

vals(i) = CF4_AP(S0_set(i),T,r,sigma,K );
end
vals
```

Part B: Result

Question 1

Comparison of European Put Pricing Using Implicit/Explicit and Crank-Nicolson Methods and comparing them with Black-Scholes Pricing

Case I: Risk-free rate = 0.04, Volatility=0.20, DeltaX=sigma*sqrt(deltat)

Stock Price	Implicit Finite Diff	Explicit Finite Diff	Crank-NicolSon	Black-Scholes Pricing
0.994303306699825	8.38911529716799	9.73822679326761	8.38946889706434	8.8020
1.99655664839782	7.80305466395838	7.80384054621909	7.80343919653652	7.8020
2.98683497204237	6.81248729239112	6.81327145439582	6.81287154746098	6.8020
4.00907693195971	5.78606213977278	5.78684630257095	5.78644639522391	5.8020
4.98009063503270	4.77975335037171	4.78053736008970	4.78013752456359	4.8020
5.99755143215935	3.79605984939789	3.79682677611946	3.79643584867802	3.8021
7.00252378746874	2.81305205727722	2.81363964225657	2.81332530622547	2.8054
8.05020877282363	1.84783104080057	1.84768767167022	1.84784406894340	1.8443
8.97229556625484	1.03672221410548	1.03696180239227	1.03662618564379	1.0244
10.0000000000000	0.464141520648174	0.464504098803185	0.464053679373750	0.4647
10.9740861974458	0.164935852221677	0.164158834641739	0.164760128828180	0.1715
12.0430567868971	0.0552284475423092	0.0550136666958535	0.0550117359520877	0.0525
13.0129883239808	0.0144994119014947	0.0140845537733704	0.0143386458402321	0.0137
14.0610368377819	0.00294839802975165	0.00282145924856911	0.00287271343900642	0.0031
14.9599310114202	0.000712670273406816	0.000641511773236704	0.000680976197145013	0.0006
15.9162897052588	0.000115600864147759	9.65779957394182e-05	0.000106770948503108	0.0001
16.9337865120060	2.57580252921504e-05	1.98857930070621e-05	2.29532796697922e-05	0.0000
18.0163298698597	3.85493348368954e-06	2.75074913556759e-06	3.24965981259108e-06	0.0000
18.8734158242564	6.67592615573760e-07	4.18261338982559e-07	5.29275636891611e-07	0.0000
20.0799558338446	1.41910295600638e-07	7.32959357794956e-08	1.05720413889799e-07	0.0000

Case II: Risk-free rate = 0.04, Volatility=0.20, DeltaX=sigma*sqrt(3*deltat)

Stock Price	Implicit Finite Diff	Explicit Finite Diff	Crank-NicolSon	Black-Scholes Pricing
0.994303306699825	8.45971484884148	9.80277092337339	8.46011876056791	8.8020
1.99655664839782	7.80543572947618	7.80622141636298	7.80582018499217	7.8020
2.98683497204237	6.81515781007832	6.81594197695583	6.81554206753690	6.8020
4.00907693195971	5.79291523688344	5.79369940620390	5.79329949553056	5.8020
4.98009063503270	4.82190122681747	4.82268527091604	4.82228541696682	4.8020
5.99755143215935	3.80452259326402	3.80529076158990	3.80489871576799	3.8021
7.00252378746874	2.80300260035824	2.80355994748609	2.80327396274277	2.8054
8.05020877282363	1.79871029257521	1.79871776704484	1.79871084657831	1.8443
8.97229556625484	1.04349357770993	1.04329525451403	1.04339707590710	1.0244
10.0000000000000	0.463592117124893	0.463402461234844	0.463504561426770	0.4647
10.9740861974458	0.175720624066016	0.175366735120155	0.175550202321469	0.1715
12.0430567868971	0.0495270425827810	0.0490884787533799	0.0493114740589060	0.0525
13.0129883239808	0.0134827668952108	0.0131668341801959	0.0133265666693637	0.0137
14.0610368377819	0.00290757844040139	0.00275737579361903	0.00283300637092117	0.0031
14.9599310114202	0.000718693918416404	0.000655590754293566	0.000687209761249423	0.0006
15.9162897052588	0.000152656438990784	0.000131504730521501	0.000142017596054042	0.0001
16.9337865120060	2.78973827776591e-05	2.21750302140369e-05	2.49826389868258e-05	0.0000
18.0163298698597	4.39570539374774e-06	3.13422933569314e-06	3.74124609587012e-06	0.0000
18.8734158242564	9.99131238817435e-07	6.42520632321301e-07	8.10727881826674e-07	0.0000
20.0799558338446	1.22344115554552e-07	6.63108317021855e-08	9.18191516483952e-08	0.0000

Case III: Risk-free rate = 0.04, Volatility=0.20, DeltaX=sigma*sqrt(4*deltat)

Stock Price	Implicit Finite Diff	Explicit Finite Diff	Crank-NicolSon	Black-Scholes Pricing
0.994975604926678	8.47189623441948	9.80277092337339	8.47230949725566	8.8020
1.99893821921291	7.80305380269438	7.80383948789651	7.80343824430237	7.8020
3.01636558219098	6.78562657892287	6.78641074833381	6.78601083762347	6.8020
4.01593163133333	5.78605973021183	5.78684390281061	5.78644399046439	5.8020
4.97752054247307	4.82447034137279	4.82525438518911	4.82485453130991	4.8020
5.95253585103457	3.84952325046647	3.85029369546862	3.84990052029573	3.8021
6.99233267497397	2.81310289368166	2.81366464884359	2.81337643843463	2.8054
8.06813673005561	1.78246155809321	1.78246079425970	1.78245809110225	1.8443
8.98228074046654	1.03624506467684	1.03604628845885	1.03614840027403	1.0244
10.0000000000000	0.463316777367869	0.463127400651241	0.463229367507976	0.4647
10.9356469114854	0.183021997534295	0.182675106593080	0.182855166614471	0.1715
11.9588373372681	0.0550354131677085	0.0545932404183231	0.0548182449637140	0.0525
13.0777622592252	0.0123079190150113	0.0120033683343382	0.0121573063169822	0.0137
14.0478228373114	0.00297539638635711	0.00282374636434270	0.00290010976608196	0.0031
15.0898389614997	0.000589229489640517	0.000534365432776275	0.000561831558429426	0.0006
15.9217680443829	0.000153525939271013	0.000132470811993029	0.000142935925154001	0.0001
17.1027865708815	2.15181160106444e-05	1.69228840076468e-05	1.91723937353874e-05	0.0000
18.0456929586145	4.34352784603305e-06	3.11212855125256e-06	3.70481083231233e-06	0.0000
19.0405834164488	7.88599505332745e-07	5.02775344214075e-07	6.37229865737786e-07	0.0000
20.0903239166373	1.29099644288836e-07	7.13170482135650e-08	9.76931509726100e-08	0.0000

Question II:

Comparison of American Call Pricing Using Implicit/Explicit and Crank-Nicolson Methods and comparing them with Black-Scholes Pricing

Case I:Risk-free rate = 0.04, Volatility=0.20, DeltaX=sigma*sqrt(deltat)

Stock Price	Implicit Finite Diff	Explicit Finite Diff	Crank-NicolSon	Black-Scholes Pricing
1.00391485522872	8.65934089817449e-49	0	1.39197732644919e-55	0.0000
1.99893821921291	1.97762634313577e-27	3.37140178617803e-34	9.14261161064910e-30	0.0000
2.98950668394753	3.68230376782991e-17	1.89516034829052e-19	4.52650011631409e-18	0.0000
4.01593163133333	5.31873807472398e-11	9.16129381556451e-12	2.38958070257037e-11	0.0000
5.02224053539794	3.15979291263907e-07	1.62033901431055e-07	2.34288194836628e-07	0.0000
6.00601576314580	8.22395091427542e-05	6.50017645536243e-05	7.39825127492168e-05	0.0001
6.99233267497397	0.00339155648081322	0.00319497618914888	0.00328054875796847	0.0034
7.99629488677036	0.0421329525937720	0.0412054173731547	0.0417617236638916	0.0423
8.98228074046654	0.217010176792807	0.216465598184747	0.216529890863979	0.2224
10.0000000000000	0.662148946413616	0.661727356843291	0.661676847264929	0.6627
11.0338970439233	1.39684052869067	1.39527934254172	1.39628054700932	1.3696
11.9588373372681	2.21207360234481	2.21107465217457	2.21147263209672	2.2505
12.9613127519617	3.17382018189355	3.17262115362360	3.17327515677339	3.2117
14.0478228373114	4.24877947067431	4.24786836086468	4.24831952659002	4.2011
14.9554731367569	5.15419422389426	5.15333889362498	5.15377826996136	5.1986
16.0648154408167	6.26293968041410	6.26213648487133	6.26254659019804	6.1981
16.9504966737620	7.14853124765746	7.14774120202810	7.14814418225713	7.1980
18.0456929586145	8.24370584845833	8.24292056998320	8.24332098209183	8.1980
19.0405834164488	9.23859331793064	9.23780889349673	9.23820891812284	9.1980
19.9114318191605	10.1094413691303	10.1086571247027	10.1090570711013	10.1980

Case II: Risk-free rate = 0.04, Volatility=0.20, DeltaX=sigma*sqrt(3*deltat)

Stock Price	Implicit Finite Diff	Explicit Finite Diff	Crank-NicolSon	Black-Scholes Pricing
1.00391485522872	4.93019804305062e-46	0	5.82426689242423e-51	0.0000
1.99893821921291	1.96824605481238e-26	3.21419237310107e-31	2.79180002713292e-28	0.0000
2.98950668394753	9.64149780159965e-17	1.46062078293705e-18	1.63409109689768e-17	0.0000
4.01593163133333	7.41844125920045e-11	1.49951557465435e-11	3.60969318048811e-11	0.0000
5.02224053539794	2.75695757071951e-07	1.48037826802175e-07	2.05994669769147e-07	0.0000
6.00601576314580	8.30497520618294e-05	6.70438308968895e-05	7.49111846235754e-05	0.0001
6.99233267497397	0.00353601514563663	0.00330918556645664	0.00342311525278642	0.0034
7.99629488677036	0.0469293213353163	0.0461526165333855	0.0465456118037962	0.0423
8.98228074046654	0.213799953159332	0.212817448434778	0.213319186715374	0.2224
10.0000000000000	0.661603542948808	0.660629703014492	0.661131721376301	0.6627
11.0338970439233	1.34781883179384	1.34668075641934	1.34726414300600	1.3696
11.9588373372681	2.29059648115122	2.28937372827654	2.28999664430127	2.2505
12.9613127519617	3.22448432451252	3.22338420037803	3.22394385479667	3.2117
14.0478228373114	4.26195827869476	4.26102388206308	4.26149943587761	4.2011
14.9554731367569	5.15866410715349	5.15781680780415	5.15824835116996	5.1986
16.0648154408167	6.11445733733409	6.11365198709943	6.11406242551716	6.1981
16.9504966737620	7.13182999552975	7.13104007216036	7.13144280659096	7.1980
18.0456929586145	8.21435050123920	8.21356503609645	8.21396557128585	8.1980
19.0405834164488	9.07143357331887	9.07064901094360	9.07104910839294	9.1980
19.9114318191605	10.2779734300519	10.2771891653005	10.2775891215565	10.1980

Case III: Risk-free rate = 0.04, Volatility=0.20, DeltaX=sigma*sqrt(4*deltat)

Stock Price	Implicit Finite Diff	Explicit Finite Diff	Crank-NicolSon	Black-Scholes Pricing
1.00391485522872	6.56071847153715e-45	0	2.85401714963673e-49	0.0000
1.99893821921291	5.72408566088992e-26	3.48799346992057e-30	1.20038061979925e-27	0.0000
2.98950668394753	2.43314427095292e-16	5.75924078759762e-18	4.85332088652673e-17	0.0000
4.01593163133333	9.57301079225615e-11	2.13058319095740e-11	4.84291814068942e-11	0.0000
5.02224053539794	2.91678778403218e-07	1.59683495088115e-07	2.19777471273871e-07	0.0000
6.00601576314580	6.92893575889902e-05	5.55553659061999e-05	6.22957883768435e-05	0.0001
6.99233267497397	0.00344658836176776	0.00322416113620262	0.00333586805257816	0.0034
7.99629488677036	0.0486101685107576	0.0478252187785753	0.0482224347363249	0.0423
8.98228074046654	0.216538416831201	0.215555451730973	0.216057484182262	0.2224
10.0000000000000	0.661330203242974	0.660356634322506	0.660858523508587	0.6627
11.0338970439233	1.31668308342327	1.31555199722447	1.31613198113190	1.3696
11.9588373372681	2.21188774340360	2.21066137205717	2.21128630219093	2.2505
12.9613127519617	3.28808606636104	3.28699731343140	3.28755117886403	3.2117
14.0478228373114	4.24881489787835	4.24787904244132	4.24835533490698	4.2011
14.9554731367569	5.28844568879505	5.28760661592226	5.28803401284550	5.1986
16.0648154408167	6.11993973368087	6.11913446702245	6.11954486431732	6.1981
16.9504966737620	7.30082719718474	7.30003838656702	7.30044057022338	7.1980
18.0456929586145	8.24371716466573	8.24293171480328	8.24333224320095	8.1980
19.0405834164488	9.23860486349568	9.23782035596125	9.23822042778642	9.1980
19.9114318191605	10.2883455439889	10.2875612610699	10.2879612265632	10.1980

Comparison of American Put Pricing Using Implicit/Explicit and Crank-Nicolson Methods and comparing them with Black-Scholes Pricing

Case I: Risk-free rate = 0.04, Volatility=0.20, DeltaX=sigma*sqrt(deltat)

Stock Price	Implicit Finite Diff	Explicit Finite Diff	Crank-NicolSon	Binomial Method
1.00391485522872	5.71628188522091	9.74091951342468	7.13896544805695	9.0000
1.99893821921291	8.00026184318282	8.00106178078709	8.00066179598688	8.0000
2.98950668394753	7.00969337765574	7.01049331605247	7.01009333085603	7.0000
4.01593163133333	5.98326842944879	5.98406836866667	5.98366838305965	6.0000
5.02224053539794	4.97695952457914	4.97775946460206	4.97735947859252	5.0000
6.00601576314580	3.99318429604426	3.99398423685420	3.99358425045115	4.0000
6.99233267497397	3.00686738342703	3.00766732502603	3.00726733822845	3.0000
7.99629488677036	2.00290753817507	2.00370511322965	2.00330535104978	2.0000
8.98228074046654	1.09605489896657	1.09659386954593	1.09619732408241	1.0828
10.0000000000000	0.481885742438517	0.482493296770798	0.481958030266431	0.4837
11.0338970439233	0.169400704351821	0.168881865876328	0.169292833622359	0.1769
11.9588373372681	0.0563805751126232	0.0562102965267273	0.0561864468652823	0.0538
12.9613127519617	0.0147372669819439	0.0143517265127286	0.0145813869489885	0.0137
14.0478228373114	0.00298716808463517	0.00286198132021705	0.00291203266842448	0.0031
14.9554731367569	0.000720657272586492	0.000650839143044265	0.000688972942007649	0.0006
16.0648154408167	0.000116692980818588	9.78479767350782e-05	0.000107835520643410	0.0001
16.9504966737620	2.59737343117992e-05	2.01312958640972e-05	2.31571524189249e-05	0.0000
18.0456929586145	3.88319151901306e-06	2.77521800923174e-06	3.27505361976417e-06	0.0000
19.0405834164488	6.71984271760368e-07	4.21686753255288e-07	5.33000846650326e-07	0.0000
19.9114318191605	1.42766599640672e-07	7.40844610814506e-08	1.06404650332270e-07	0.0000

Case II: Risk-free rate = 0.04, Volatility=0.20, DeltaX=sigma*sqrt(3*deltat)

Stock Price	Implicit Finite Diff	Explicit Finite Diff	Crank-NicolSon	Binomial Method
1.00391485522872	9.31715913620512	9.80277092337339	9.707871694262082	9.0000
1.99893821921291	8.00264341080528	8.00344335160218	8.00304336520565	8.0000
2.98950668394753	7.01236508478402	7.01316502795763	7.01276504037275	7.0000
4.01593163133333	5.99012312241329	5.99092306804029	5.99052307922871	6.0000
5.02224053539794	5.01910941700983	5.01990936496730	5.01950937499048	5.0000
6.00601576314580	4.00164861744126	4.00244856784065	4.00204857664287	4.0000
6.99233267497397	2.99667625971999	2.99747621253126	2.99707622012750	3.0000
7.99629488677036	1.94900033710247	1.94979122717637	1.94939278235229	2.0000
8.98228074046654	1.10339470456323	1.10369252131510	1.10353562313651	1.0828
10.000000000000	0.481277641750413	0.481424468006362	0.481340529076387	0.4837
11.03389704392333	0.180551910249011	0.180357307441858	0.180447019395021	0.1769
11.95883733726821	0.0505351334330862	0.0501448147739402	0.0503370036735207	0.0538
12.96131275196167	0.0137009705879655	0.0133972053790714	0.0135483646309059	0.0137
14.04782283731144	0.0029456927502591 8	0.0027974973311359 5	0.0028714510985029 4	0.0031
14.95547313675699	0.0007267526876096 18	0.0006639471268216 84	0.0006952282471708 02	0.0006
16.06481544081677	0.0001541356031317 18	0.0001329912287872 66	0.0001434565472404 43	0.0001
16.95049667376220	2.81331558677986e-05	2.23998265952738e-05	2.52044182166750e-05	0.0000
18.04569295861455	4.42838691796939e-06	3.16301394384227e-06	3.77058028714492e-06	0.0000
19.04058341644888	1.00591388791936e-06	6.48033381231076e-07	8.16547233784034e-07	0.0000
19.91143181916055	1.23084499433387e-07	6.68342287975977e-08	9.24084253766570e-08	0.0000

Case III: Risk-free rate = 0.04, Volatility=0.20, DeltaX=sigma*sqrt(4*deltat)

Stock Price	Implicit Finite Diff	Explicit Finite Diff	Crank-NicolSon	Binomial Method
1.0039148552287 2	9.545806696634428	9.80277092337339	9.545806696634428	9.0000
1.9989382192129 1	8.00066179358811	8.00106178078709	8.00066179358811	8.0000
2.9895066839475 3	6.98323442898213	6.98363441780902	6.98323442898213	7.0000
4.0159316313333 3	5.98366837824045	5.98406836866667	5.98366837824045	6.0000
5.0222405353979 4	5.02207946556214	5.02247945752693	5.02207946556214	5.0000
6.0060157631458 0	4.04706415544060	4.04746414896543	4.04706415544060	4.0000
6.9923326749739 7	3.00726732983750	3.00766732502603	3.00726732983750	3.0000
7.9962948867703 6	1.93146608113222	1.93186326994439	1.93146608113222	2.0000
8.9822807404665 4	1.09559224742141	1.09572897602217	1.09559224742141	1.0828
10.000000000000 0	0.481030354278228	0.481106817500562	0.481030354278228	0.4837
11.033897043923 3	0.188005307606035	0.187916388678531	0.188005307606035	0.1769
11.958837337268 1	0.0559796191547082	0.0557859101533983	0.0559796191547082	0.0538
12.961312751961 7	0.0123562749620377	0.0122093156833668	0.0123562749620377	0.0137
14.047822837311 4	0.00293942247855743	0.00286454208133567	0.00293942247855743	0.0031
14.955473136756 9	0.000568232505304082	0.000540981202606276	0.000568232505304082	0.0006
16.064815440816 7	0.000144377874423927	0.000133947080823029	0.000144377874423927	0.0001
16.950496673762 0	1.93385755891503e-05	1.70888981285285e-05	1.93385755891503e-05	0.0000
18.045692958614 5	3.73367307258309e-06	3.14011757908888e-06	3.73367307258309e-06	0.0000
19.040583416448 8	6.41716358952637e-07	5.06948897489639e-07	6.41716358952637e-07	0.0000
19.911431819160 5	9.83179016402425e-08	7.18668538207382e-08	9.83179016402425e-08	0.0000

Special Case of Explicit Finite Difference Method:

American Call

Stock Price	DeltaS=0.5	DeltaS=1.0	DeltaS=1.5	BLS
1	8.05908906555446e-06	4.23120042849579e-05	0.000215052872448902	0.0000
2	3.73653643090094e-05	0.000149720264117137	0.000950534470810917	0.0000
3	0.000162413546695060	0.000490334858856887	0.000950534470810917	0.0000
4	0.000678934761595616	0.00157366197499995	0.00398080850618527	0.0000
5	0.00270100303164812	0.00493549392224058	0.0161721617646395	0.0000
6	0.0100773816727419	0.0149954440222290	0.0161721617646395	0.0001
7	0.0346169697378537	0.0436079916599197	0.0625643601667429	0.0034
8	0.107038492591763	0.119407852077842	0.223423928790846	0.0423
9	0.290253539993515	0.301069364317236	0.223423928790846	0.2224
10	0.672495172791614	0.678572235603822	0.699475243886247	0.6627
11	1.30877706247973	1.32013053064875	1.75809342181916	1.3696
12	2.15135269271975	2.16547172548132	1.75809342181916	2.2505
13	3.08832535844827	3.09982057269835	3.12317334648294	3.2117
14	4.04729448173187	4.05736172001512	4.55716818890778	4.2011
15	5.01184756193823	5.02095556720985	4.55716818890778	5.1986
16	6	6	6.00308598676422	6.1981
17	7	7	7.50000000000000	7.1980
18	8	8	7.50000000000000	8.1980
19	9	9	9	9.1980
20	10.1972290766266	10.1972290766266	10.6972290766266	10.1980

American Put

Stock Price	DeltaS=0.5	DeltaS=1.0	DeltaS=1.5	Binomial
1	9	9	9	9
2	8	8	7.50000000000000	8
3	7	7	7.50000000000000	7
4	6	6	6	6
5	5	5	4.50000000000000	5
6	4	4	4.50000000000000	4
7	3	3	3	3
8	2	2	1.52655698494920	2
9	1.11792851917638	1.11779372600337	1.52655698494920	1.0828
10	0.491018608129552	0.487171188552587	0.489712972707701	0.4837
11	0.149453460417341	0.151484634378574	0.0878631342376238	0.1769
12	0.0270556083115435	0.0317303547850699	0.0878631342376238	0.0538
13	0.00248906571625752	0.00432504456931289	0.00854290424785304	0.0137

14	0.000100698965808740	0.000372624091766868	0.000435480856282066	0.0031
15	1.56478225159988e-06	1.95559106734762e-05	0.000435480856282066	6.1178e-004
16	8.04556017377189e-09	5.90847251422366e-07	1.08218458198841e-05	1.1063e-004
17	1.11597386634173e-11	9.35768976770506e-09	1.12161724560212e-07	1.9105e-005
18	2.96514805770954e-15	6.54178420557635e-11	1.12161724560212e-07	2.7897e-006
19	7.19586466945315e-20	1.35810857194524e-13	3.30036253887164e-10	4.3579e-007
20	0	0	0	6.2295e-008