Question1

(a)

Stepping.m

This matlab file is used to determine the option value by implicit method, CN method and CN-R method

```
function [result1,result2,result3] = stepping(S,N)
alpha_parameter =0.8;    % constant related to alpha
r = .02;         % risk free rate
T = 1;           % time to expiry
K = 10;          % strike price
S0 = 10;         % initial stock price
delt = T/N;     % time interval

m = length(S);  %number of grids in row

alpha_central = zeros(m,1);
beta_central = zeros(m,1);
alpha_forward = zeros(m,1);
beta_forward  = zeros(m,1);
alpha_backward = zeros(m,1);
beta_backward = zeros(m,1);
alpha = zeros(m,1);
beta = zeros(m,1);


for i = 2: m - 1
    %%central alpha and beta formula
    alpha_central(i) = alpha_parameter^2*S(i)/((S(i) - S(i-1))*(S(i+1) - S(i-
1)))...
        -r*S(i)/(S(i+1) - S(i-1));
    beta_central(i) = alpha_parameter^2*S(i)/((S(i+1) - S(i))*(S(i+1) - S(i-1)))...
        +r*S(i)/(S(i+1) - S(i-1));

    %%forward alpha and beta formula
    alpha_forward(i) = alpha_parameter^2*S(i)/((S(i) - S(i-1))*(S(i+1) - S(i-1)));
    beta_forward(i) = alpha_parameter^2*S(i)/((S(i+1) - S(i))*(S(i+1) - S(i-1)))...
        + r*S(i)/(S(i+1) - S(i));  % same as beta_central

    %%backward alpha and beta formula

    alpha_backward(i) = alpha_parameter^2*S(i) / ((S(i) - S(i-1))*(S(i+1) - S(i-
1))) ...
        - r * S(i) / (S(i+1) - S(i));
    beta_backward(i) = alpha_parameter^2*S(i)/((S(i+1) - S(i))*(S(i+1) - S(i-1)));
end


%% choosing parameter

for i = 2:m-1
   if(alpha_central(i) >=0 && beta_central(i) >=0)
       alpha(i) = alpha_central(i);
       beta(i) = beta_central(i);
   elseif (alpha_forward(i) >=0 && beta_forward(i) >=0)
       alpha(i) = alpha_forward(i);
       beta(i) = beta_forward(i);
   else
       alpha(i) = alpha_backward(i);
       beta(i) = beta_backward(i);
   end
end
```

```
M_matrix = [delt.*-alpha,delt.*(alpha + beta + r) , delt.*-beta];
M = spdiags(M_matrix, [-1,0,1], m-1, m);
M = full([M;zeros(1,m)]);
I = eye(m);

A1 = sparse(I + M);     % create space matrix
[L1,U1,P1,Q1] = lu(A1); % lu factorization

V = zeros(m, N+1);
V_init = max(K - S.^2, S.^2 - K)';
V(:,1) = V_init;
V_old = V_init;
V_new = zeros(m,1);

for i = 1: N
            V_new = Q1 * ((L1*U1) \ P1) * V_old;
        V_old = V_new;
        V(:,i+1) = V_new;
end
X1 = sprintf('The option value for fully implict in N = %d, S= %d,
is: %s',N,length(S),V_new(S == S0));
disp(X1)
result1 = V_new(S == S0);

%% for CN time stepping
M_matrix_CN = [delt.*-alpha/2, delt.*(alpha + beta + r)/2, delt.*-beta/2];
M_CN = spdiags(M_matrix_CN, [-1,0,1], m-1, m);
M_CN = full([M_CN;zeros(1,m)]);
A2 = sparse(spdiags(ones(m,1),0,m,m)+M_CN);     % create space matrix
[L2,U2,P2,Q2] = lu(A2); % lu factorization
V_2 = zeros(m, N+1);
V_init_2 = max(K - S.^2, S.^2 - K)';
V_2(:,1) = V_init_2;
V_old_2 = V_init_2;
V_new_2 = zeros(m,1);

for i = 1: N
        Vt = (I - M_CN)*V_old_2;
        V_new_2 = Q2 * ((L2*U2) \ (P2*Vt)) ;
        V_old_2 = V_new_2;
        V_2(:,i+1) = V_new_2;
end
X2 = sprintf('The option value for NC in N = %d, S= %d,
is: %s',N,length(S),V_new_2(S == S0));
disp(X2)
result2 = V_new_2(S == S0);
%% CN-Rannacher time stepping


V_3 = zeros(m, N+1);
V_init_3 = max(K - S.^2, S.^2 - K)';
V_3(:,1) = V_init_3;
V_old_3 = V_init_3;
V_new_3 = zeros(m,1);

%first two use fully implicit method
for i = 1: 2
        V_new_3 = Q1 * ((L1*U1) \ (P1 * V_old_3));
        V_old_3 = V_new_3;
        V_3(:,i+1) = V_new_3;
end

for i = 3: N
        Vt_R = (I - M_CN)*V_old_3;
        V_new_3 = Q2 * ((L2*U2) \ (P2 * Vt_R));
        V_old_3 = V_new_3;
        V_3(:,i+1) = V_new_3;
end
```

```
X3 = sprintf('The option value for NC-R in N = %d, S= %d,
is: %s',N,length(S),V_new_3(S == S0));
disp(X3)
result3 = V_new_3(S == S0);
end


K = 10;
N = 25;
S = [0:0.1*K:0.4*K,...  %input S value
    0.45*K:0.05*K:0.8*K,...
    0.82*K:0.02*K:0.9*K,...
    0.91*K:0.01*K:1.1*K,...
    1.12*K:0.02*K:1.2*K,...
    1.25*K:.05*K:1.6*K,...
    1.7*K:0.1*K:2*K,...
    2.2*K, 2.4*K, 2.8*K,...
    3.6*K, 5*K, 7.5*K, 10*K];
% for node 62 and timestep 25
L = length(S);
[V,V_CN,V_CNR] = stepping(S,N);

% for node 123 and timestep 50
N1 = 50;
S1 = movmean(S,2);
S1 = [S,S1];
S1 = sort(S1);
S1 = S1(2:end);
L1 = length(S1);
[V1,V_CN1,V_CNR1] = stepping(S1,N1);

% for node 245 and timestep 100
N2 = 100;
S2 = movmean(S1,2);
S2 = [S1,S2];
S2 = sort(S2);
S2 = S2(2:end);
L2 = length(S2);
[V2,V_CN2,V_CNR2] = stepping(S2,N2);

% for node 489 and timestep 200
N3 = 200;
S3 = movmean(S2,2);
S3 = [S2,S3];
S3 = sort(S3);
S3 = S3(2:end);
L3 = length(S3);
[V3,V_CN3,V_CNR3] = stepping(S3,N3);

% for node 489 and timestep 200
N4 = 400;
S4 = movmean(S3,2);
S4 = [S3,S4];
S4 = sort(S4);
S4 = S4(2:end);
L4 = length(S4);
[V4,V_CN4,V_CNR4] = stepping(S4,N4);

T_1 =table([N;N1;N2;N3;N4],...
           [L;L1;L2;L3;L4],...
           [V;V1;V2;V3;V4],...
           [NaN;V1-V;V2-V1;V3-V2;V4-V3],...
           [NaN;NaN;(V1-V)/(V2-V1);(V2-V1)/(V3-V2);(V3-V2)/(V4-V3)]);
T_1.Properties.VariableNames ={'Timesteps','Node','Value','Change','Ratio'};

T_2 =table([N;N1;N2;N3;N4],...
           [L;L1;L2;L3;L4],...
           [V_CN;V_CN1;V_CN2;V_CN3;V_CN4],...
```

```
            [NaN;V_CN1-V_CN;V_CN2-V_CN1;V_CN3-V_CN2;V_CN4-V_CN3],...
            [NaN;NaN;(V_CN1-V_CN)/(V_CN2-V_CN1);(V_CN2-V_CN1)/(V_CN3-V_CN2);(V_CN3-
V_CN2)/(V_CN4-V_CN3)]);
T_2.Properties.VariableNames ={'Timesteps','Node','Value','Change','Ratio'};


T_3 =table([N;N1;N2;N3;N4],...
            [L;L1;L2;L3;L4],...
            [V_CNR;V_CNR1;V_CNR2;V_CNR3;V_CNR4],...
            [NaN;V_CNR1-V_CNR;V_CNR2-V_CNR1;V_CNR3-V_CNR2;V_CNR4-V_CNR3],...
            [NaN;NaN;(V_CNR1-V_CNR)/(V_CNR2-V_CNR1);(V_CNR2-V_CNR1)/(V_CNR3-
V_CNR2);(V_CNR3-V_CNR2)/(V_CNR4-V_CNR3)]);
T_3.Properties.VariableNames ={'Timesteps','Node','Value','Change','Ratio'};
```

Timesteps	Node	Value	Change	Ratio
25	62	98.6908117979112	NaN	NaN
50	123	98.6865780483348	-0.00423374957637179	NaN
100	245	98.6849416490879	-0.00163639924693371	2.5872351043335
200	489	98.6842829719266	-0.000658677161240462	2.48437222850104
400	977	98.6839871382327	-0.000295833693996883	2.22651163341591

Table 1: The option value determined by the implicit method in different timesteps and nodes

Timesteps	Node	Value	Change	Ratio
25	62	98.6862878070961	NaN	NaN
50	123	98.6844088432618	-0.00187896383425823	NaN
100	245	98.6838777605093	-0.000531082752516454	3.53798692455187
200	489	98.6837540192893	-0.000123741219994145	4.29188230519777
400	977	98.6837231360294	-3.08832598676645e-05	4.0067408856571

Table 2: The option value determined by the CN method in different timesteps and nodes

Timesteps	Node	Value	Change	Ratio
25	62	98.6866506583894	NaN	NaN
50	123	98.6844956432238	-0.00215501516555605	NaN
100	245	98.6838990358936	-0.000596607330223264	3.61211647324144
200	489	98.683759308259	-0.000139727634646647	4.26978766034369
400	977	98.6837244559589	-3.48523000326395e-05	4.00913668583682

Table 2: The option value determined by the CN-R method in different timesteps and nodes

From the convergence table, we can see that the ratio of fully implicit method is around 2, but for CN method and CN-R method, there ratio is all about 4. It implies that the fully implicit is linear convergent, while CN and CN-R are quadratic convergent, but CN-R has less fluctuate than CN method.

(b)

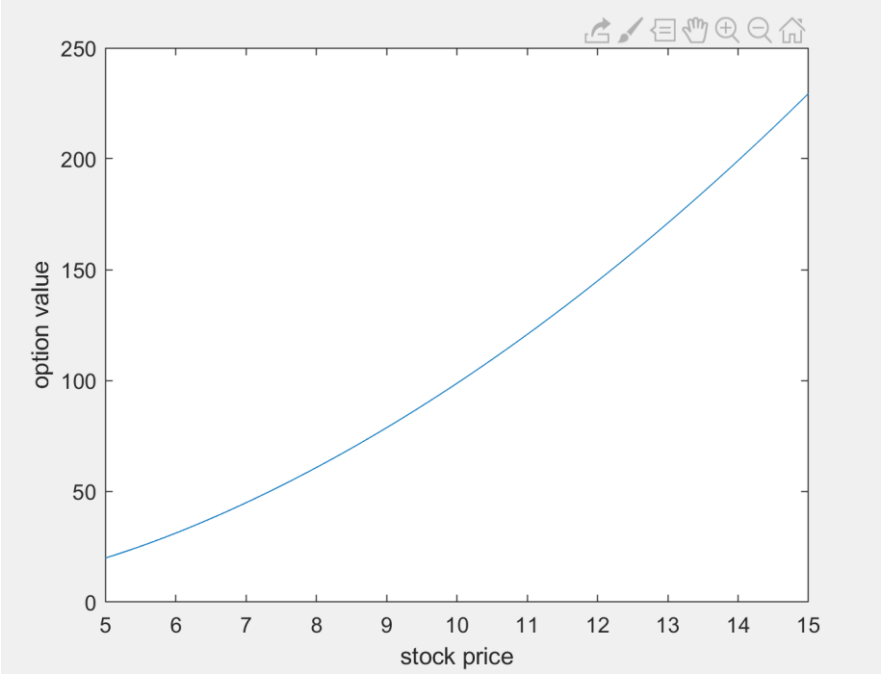

Figure1 . the plot option value varied with stock price

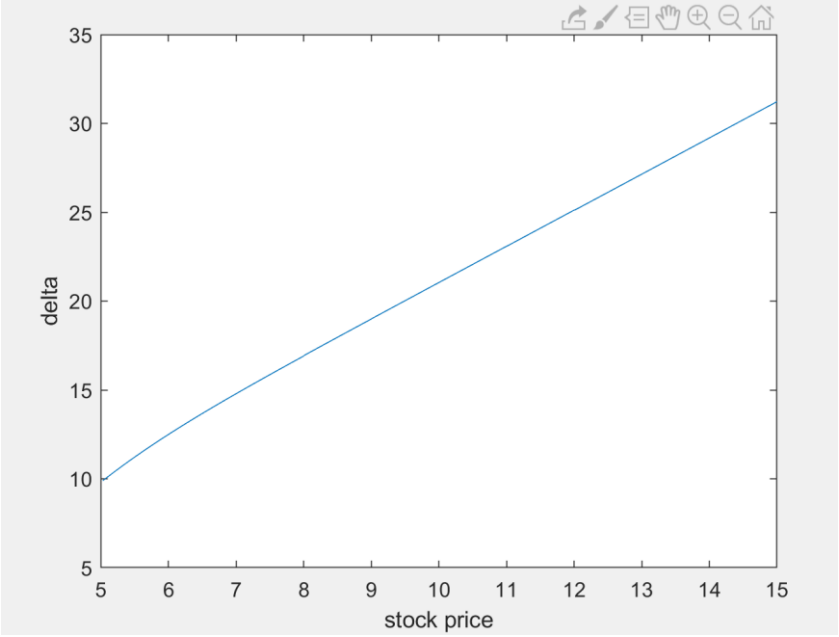

Figure2 . the plot delta varied with stock price

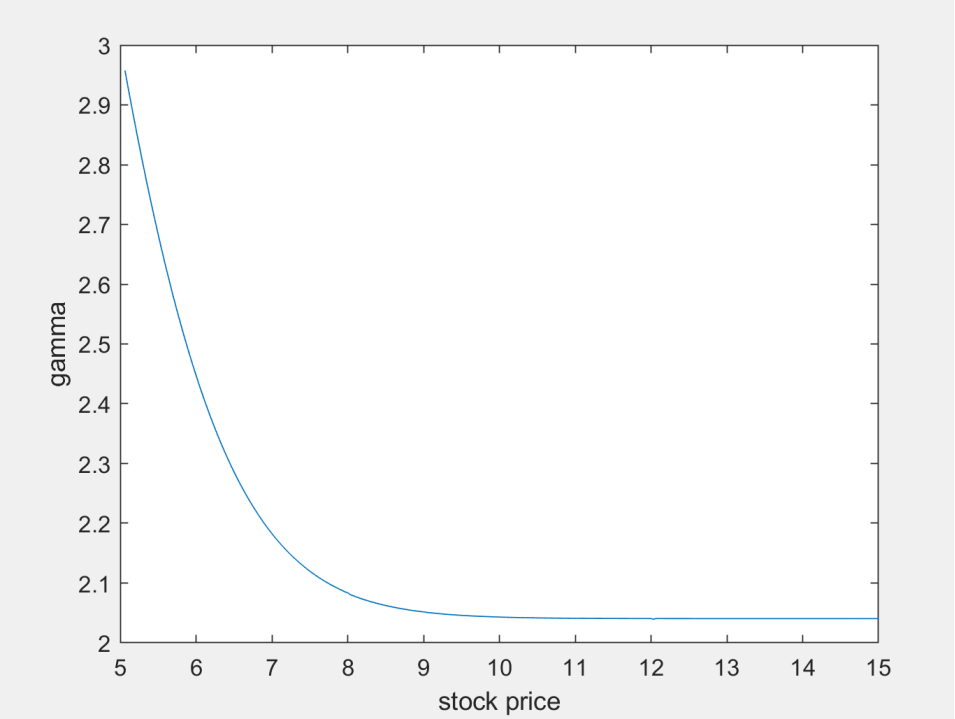

Figure2 . the plot gamma varied with stock price

From those plots, we can see that by using CN-Rannacher timestepping, I don't see any no oscillation in delta and gamma. And the option price plot quadratically increases, although the delt plot is almost linear, but not exactly linear. As for the gamma, the plot decreases suddenly and converge to about 2.