FIN 315 _ MIDTERM

Table of Contents

Files Introduction

Author: Wanli Feng h702001633

This file is main script for fin315 midterm. It utilizes some matlab built-in functions and user-defined functions. For user-defined functions, it includes 'FixMacTime.m', 'gabBNGrks.m', 'gabBNpr.m', 'gabBSGrks.m', 'gabBSpr.m', 'gabImpv.m', 'gabNRimpv.m' and 'OptionKVTD.m'. Each user-defined function has its detail instruction within their m.file respectively.

To be noted, the gabImpv integrates three approximation methods: self-defined function, newton raphson; built-in function fzero and fsolve. Specificially for the midterm requirement, I created an independent function file for newton raphson method--'gabNRimpv.m', which is also used in 'gabImpv.m' function file.

Required Comments for part1 and part2

Comments for part1 d) From the 2D graphs above, I find the followings: 1. The pricing error With respect to moneyness: 1) Call: The larger the moneyness, the closer the pricing of BS and BN model. 2) Put: Moneyness doesn't affect the pricing a lot, but we can easily found that the sources of pricing error is from numbers of steps. And the graph tells that the middle of moneyness estimate more accurately. 2. The pricing error With respect to numbers of steps: 1) Call: The pricing error behave differentfly according to different moneyness. However, the graph shows that the errors fluctures in the middle, i.e. smallest point and largest point. 2) Put: Numbers of steps doesn't affect the pricing a lot. And the graph tells that the middle of numbers of estimate more accurately.

Comments for part2 c) From the 2D graphs above, I find the followings: 1. As I change the small change, the delta and vega hardly change, but gamma change quite a bit. I think it is because both delta and vega are first deriative and gamma is second deriative. Tiny change makes gamma flucturate. 2. Based on this situation -- small change is 1e-06, delta and vega for both BS and BN models are nearly the same. For gamma, I discuss as follows: 1) Call: The shapes for BS and BN model are different. The shape of BS model holds even for put. BN model fluctures severely when is small, and move increasingly mildly as the moneyness become larger. 2) Put: The shapes are still different. The shape of BS model holds. BN model behaves reversely, i.e. it fluctures more and more severely as the moneyness become larger.

Clear environment

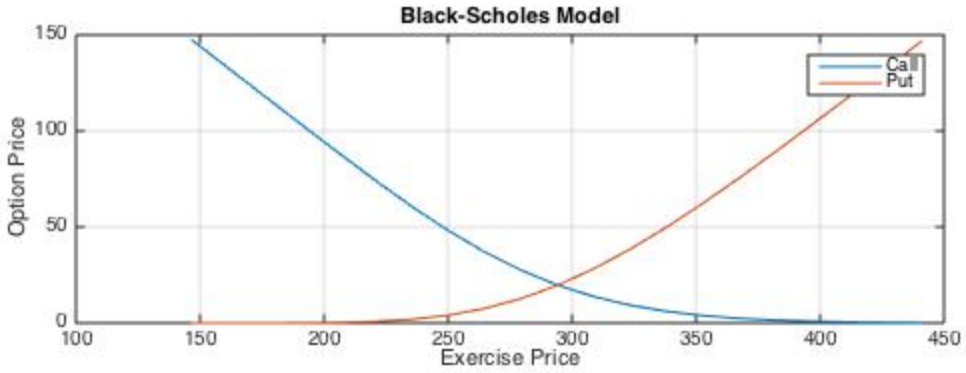
```
clc, clear, close all
```

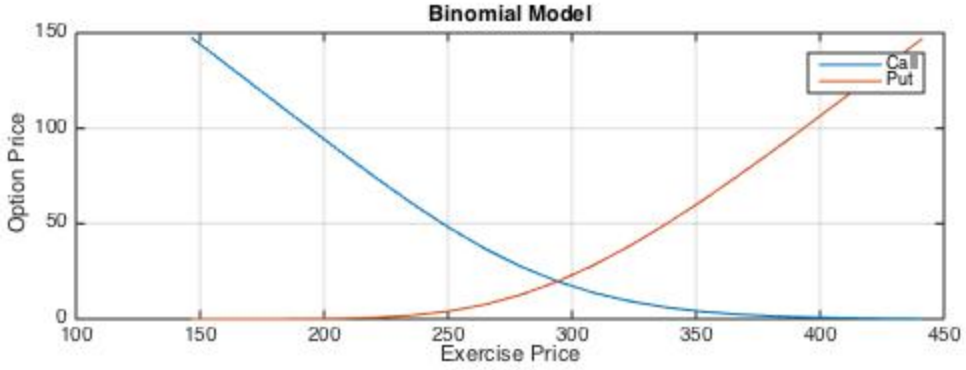
Part 1) Option Pricing Models

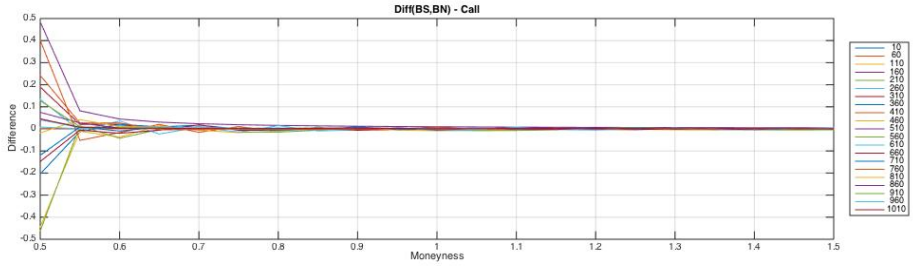
```
% a)
```

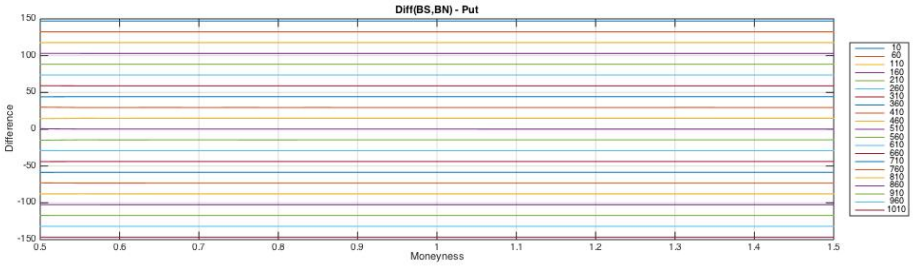
```
% Initial Values: Using the data of BIIB stock.
S0 = 293.99; TtoM = 89/360; rf = 0.00356; vola = 0.33799; div = 0;
optyp = [1,-1]; opsy = 0; w = 0.5:0.05:1.5; X = w'*S0; MNY = X./S0;
% From the graph in part C, in my example, considering the speed, 160
% steps seems the best for call and 510 steps seems the best for put.
N = [160,510];


% Calculate Call and Put Option Price Using BS and BN model, Respectively.
BSpr = zeros(length(X),length(optyp));
BNpr = zeros(length(X),length(optyp));
for j=1:2
    BSpr(:,j) = gabBSpr(S0,X,TtoM,rf,div,vola,optyp(j));
    for i =1:length(X)
        BNpr(i,j) = gabBNpr(S0,X(i),TtoM,rf,vola,N(j),optyp(j),opsy,div);
    end
end

% b) Graphically show the relationship between option prices and exercise
figure('Name','The Relationship Between Option Prices And Exercise')
subplot(2,1,1);
plot(X,BSpr); title('Black-Scholes Model'); grid on;
xlabel('Exercise Price'); ylabel('Option Price'); legend('Call','Put');
subplot(2,1,2);
plot(X,BNpr); title('Binomial Model'); grid on;
xlabel('Exercise Price'); ylabel('Option Price'); legend('Call','Put');

% c)
Nsteps = 10:50:1010;
nsteps = length(Nsteps);
BScalls = repmat(BSpr(:,1),1,nsteps); % BS calls moneyness
BSputs = repmat(BSpr(:,1),1,nsteps); % BS puts moneyness
BNcalls = zeros(length(X),nsteps); % BN calls moneyness
BNputs = zeros(length(X),nsteps); % BN puts moneyness

for n = 1:nsteps
    for i =1:length(X)
        BNcalls(i,n) = gabBNpr(S0,X(i),TtoM,rf,vola,...
            Nsteps(n),optyp(1),opsy,div);
        BNputs(i,n) = gabBNpr(S0,X(i),TtoM,rf,vola,...
            Nsteps(n),optyp(2),opsy,div);
    end
end
dcalls = BScalls-BNcalls;
dputs = BSputs-BNputs;

% d)
%Call/Put, Moneyness, Numbers of steps ----- 2D
figure('Name','Moneyness v.s. Difference','Position',[60 100 1180 680])
subplot(2,1,1);
plot(MNY,dcalls'); grid on; title('Diff(BS,BN) - Call');
xlabel('Moneyness'); ylabel('Difference');
legend(num2str(Nsteps'),'Location','eastoutside');
subplot(2,1,2);
```

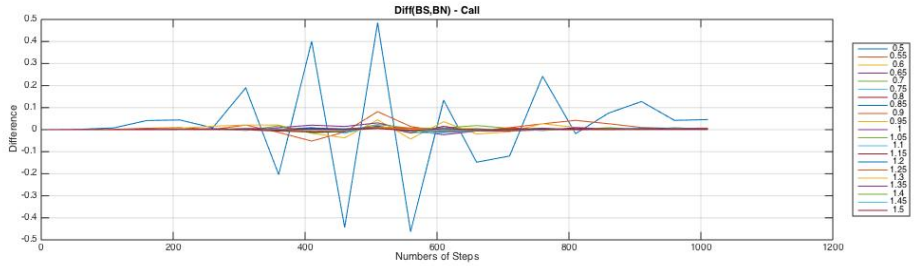
```
plot(MNY,dputs'); grid on; title('Diff(BS,BN) - Put');
xlabel('Moneyness'); ylabel('Difference');
legend(num2str(Nsteps'),'Location','eastoutside');
figure('Name','Numbers of Steps v.s. Difference','Position',[60 100 1180 680])
subplot(2,1,1);
plot(Nsteps,dcalls); grid on; title('Diff(BS,BN) - Call');
xlabel('Numbers of Steps'); ylabel('Difference');
legend(num2str(MNY),'Location','eastoutside');
subplot(2,1,2);
plot(Nsteps,dputs); grid on; title('Diff(BS,BN) - Put');
xlabel('Nsteps'); ylabel('Difference');
legend(num2str(MNY),'Location','eastoutside');

%Call/Put, Moneyness, Numbers of steps ----- 3D
figure('Name','Moneyness v.s. Numbers of steps v.s.Differences for Calls'...
    ,'Position',[60 100 1180 680])
mesh(MNY,Nsteps,dcalls); axis tight; grid on; colorbar;
title('Diff(BS,BN) - Calls','Fontsize',24,'FontWeight','Bold')
xlabel('Moneyness','Fontsize',20,'FontWeight','Bold')
ylabel('Numbers of Steps','Fontsize',20,'FontWeight','Bold')
zlabel('Differences','Fontsize',20,'FontWeight','Bold')
set(gca,'Fontsize',16,'FontWeight','Bold','LineWidth',2);
figure('Name','Moneyness v.s. Numbers of steps v.s.Differences for Puts',...
    'Position',[60 100 1180 680])
mesh(MNY,Nsteps,dputs); axis tight; grid on; colorbar;
title('Diff(BS,BN) - Calls','Fontsize',24,'FontWeight','Bold')
xlabel('Moneyness','Fontsize',20,'FontWeight','Bold')
ylabel('Numbers of Steps','Fontsize',20,'FontWeight','Bold')
zlabel('Differences','Fontsize',20,'FontWeight','Bold')
set(gca,'Fontsize',16,'FontWeight','Bold','LineWidth',2);
```

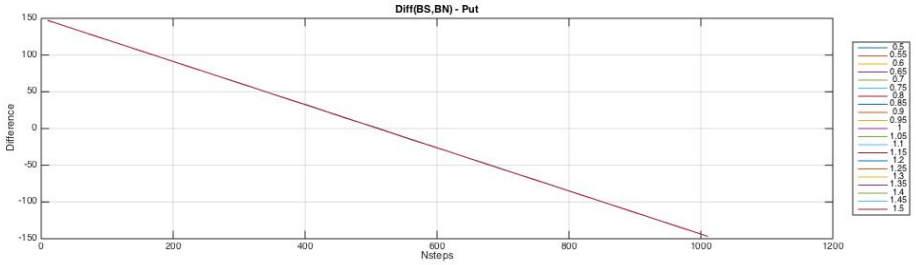

Black-Scholes Model
Option Price
Exercise Price
Call
Put
150
100
50
0
100
150
200
250
300
350
400
450

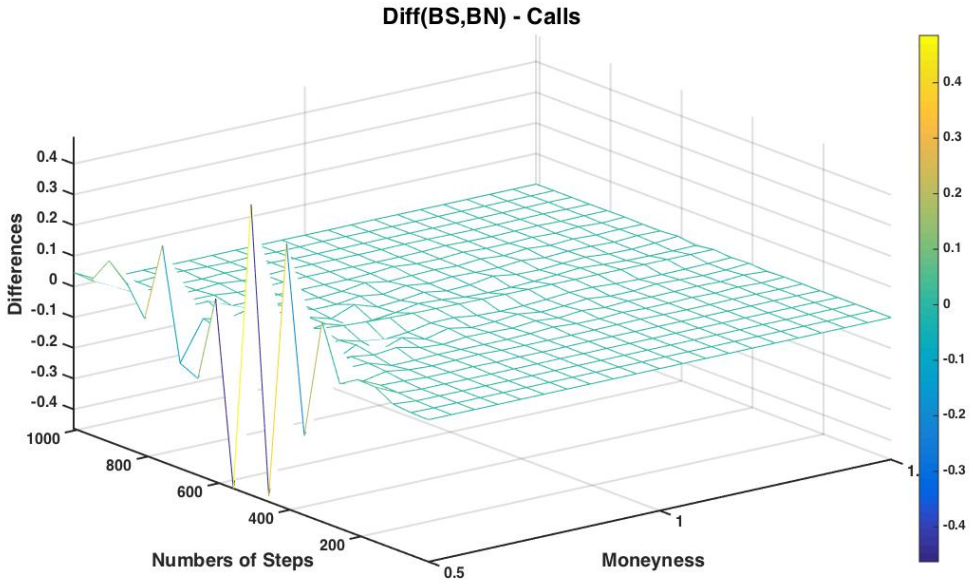

Binomial Model
Option Price
Exercise Price
Call
Put
150
100
50
0
100
150
200
250
300
350
400
450

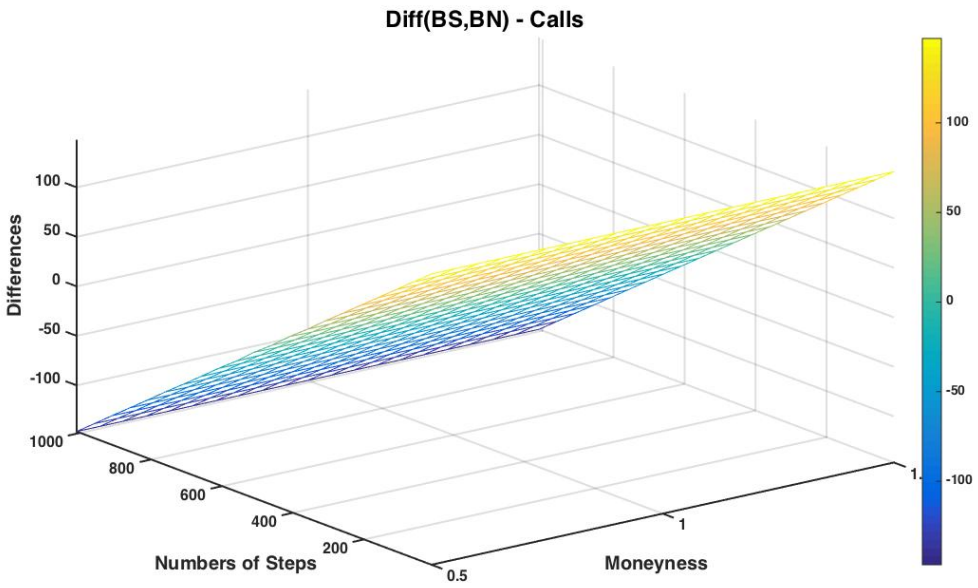

Diff(BS,BN) - Call
Difference
Moneyness
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
0.5
0.6
0.7
0.8
0.9
1
1.1
1.2
1.3
1.4
1.5


Diff(BS,BN) - Put
Difference
Moneyness
150
100
50
0
-50
-100
-150
0.5
0.6
0.7
0.8
0.9
1
1.1
1.2
1.3
1.4
1.5


Diff(BS,BN) - Call
Difference
Numbers of Steps


Diff(BS,BN) - Put
Difference
Nsteps

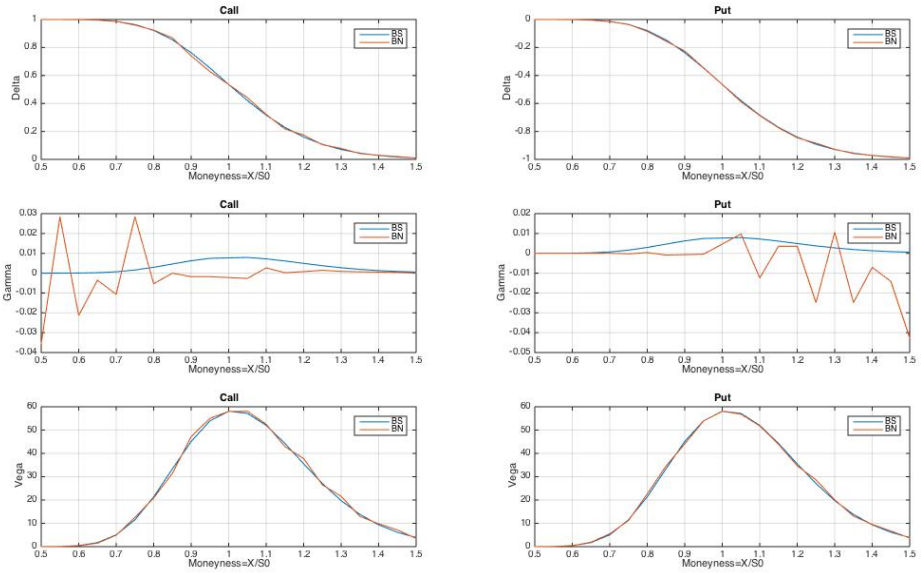

Diff(BS,BN) - Calls
Differences
Numbers of Steps
Moneyness



Part 2) Option Sensitivities

```
% a) & b)
% smch = 1e-07; BN_Gamma got extreme value.
% smch = 1e-05; BN_Gamma are all nearly zero.
smch = 1e-06; %Good
grk = {'delta','gamma','vega'};
grkL = length(grk); XL = length(X); tL = length(optyp);
BSGrks = zeros(XL,tL,grkL);
BNGrks = zeros(XL,tL,grkL);
for j=1:tL
    for k=1:length(grk)
        for i=1:length(X)
            BSGrks(i,j,k) = gabBSGrks(S0,X(i),TtoM,rf,div,...
                vola,optyp(j),grk{k});
            BNGrks(i,j,k) = gabBNGrks(S0,X(i),TtoM,rf,div,...
                vola,N(j),optyp(j),opsy,smch,grk{k});
        end
    end
end
BS_Delta = BSGrks(:,:,1); BN_Delta = BNGrks(:,:,1);
BS_Gamma = BSGrks(:,:,2); BN_Gamma = BNGrks(:,:,2);
BS_Vega = BSGrks(:,:,3); BN_Vega = BNGrks(:,:,3);

% c)
% Remove outliers, which is larger than 1.
intv = 1;
ind = sum(abs(BN_Gamma)<intv,2)==2;
BS_Gamma = BS_Gamma(ind,:);
BN_Gamma = BN_Gamma(ind,:);
opty = {'Call','Put'};
```

```
figure('Name','Compare Greeks Calculated By BS and BN Model',...
    'Position',[60 100 1180 680])
for k = 1:2
    subplot(3,2,k)
    plot(MNY,[BSGrks(:,k,1),BNGrks(:,k,1)]); grid on;
    title(opty(k)); xlabel('Moneyness=X/S0'); ylabel('Delta'); legend('BS','BN');
end
subplot(3,2,3)
plot(MNY(ind),[BS_Gamma(:,1),BN_Gamma(:,1)]); grid on; title(opty(1));
xlabel('Moneyness=X/S0'); ylabel('Gamma'); legend('BS','BN');
subplot(3,2,4)
plot(MNY(ind),[BS_Gamma(:,2),BN_Gamma(:,2)]); grid on; title(opty(2));
xlabel('Moneyness=X/S0'); ylabel('Gamma'); legend('BS','BN');
for k = 1:2
    subplot(3,2,k+4)
    plot(MNY,[BSGrks(:,k,3),BNGrks(:,k,3)]); grid on; title(opty(k));
    xlabel('Moneyness=X/S0'); ylabel('Vega'); legend('BS','BN');
end
```



Part 3) Implied Volatility

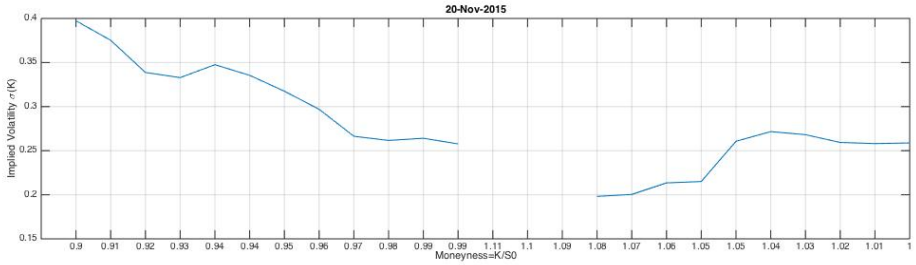
```
% a) & d)
% initial input
% S0 = [294.85,294.75,294.78,295.14,295.48,297.08];
% rf = [0.0016,0.0022,0.0029,0.0053,0.0066,0.0093];
% div = [0.0066,0.0077,0.0054,0.0023,0.0017,0.0009];
% T = [9, 37, 65, 156, 219, 436];
S0 = 293.99; rf=0.00356; div=0;
filename = 'BIIBopt.xlsx'; Kcol=2; Vcol=3;

% Construct call and put option structure for implied volatility
```

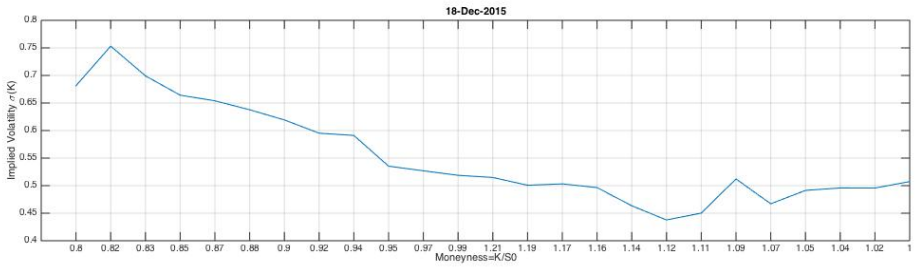
```
[optCall,importCall] = OptionKVTD(filename,'Call',Kcol,Vcol,S0,rf,div,...
    N(1),optyp(1),opsy);
[optPut,importPut] = OptionKVTD(filename,'Put',Kcol,Vcol,S0,rf,div,...
    N(2),optyp(2),opsy);

% Data pre-process
dateL = size(optCall.date,2);
mnyC = optCall.M; mnyP = optPut.M;
% Indices of ITM for call and put
indC = mnyC<=1; indP = mnyP>1; indP1 = mnyP==1;

% Construst Volatility Smile
plotnum = repmat([1,2],1,dateL/2);
% figure('Name','Volatility Smile','Position',[60 100 1180 680])
for i = 1:dateL
    if (i-1)/2-floor((i-1)/2) == 0
        figure('Name','Volatility Smile','Position',[60 100 1180 680])
    end
    impvC = optCall.impv(indC(:,i),i); impvP = optPut.impv(indP(:,i),i);
    optITMimpv = [impvC;impvP(end:-1:1)];
    impvP1 = optPut.impv(indP1(:,i),i);
    % Moneyness
    mC = mnyC(indC(:,i),i); mP = mnyP(indP(:,i),i); mP1 = mnyP(indP1(:,i),i);
    format short
    xVal = round([mC;mP(end:-1:1)],2);
    subplot(2,1,plotnum(i));
    plot(optITMimpv); hold on; grid on
    plot(mP1,impvP1,'*','MarkerSize',5); hold off
    xStr = cellfun(@num2str,num2cell(xVal),'UniformOutput',false);
    set(gca,'Xtick',1:25,'XTickLabel',cellstr(xStr))
    title([optCall.date{1,i}])
    xlabel('Moneyness=K/S0'); ylabel('Implied Volatility \sigma(K)');
end


dates = optCall.date(1,:);
disp('Call Option Implied Volatility');
disp(dates); disp(optCall.impv);
disp('Put Option Implied Volatility');
disp(dates); disp(optPut.impv);
```

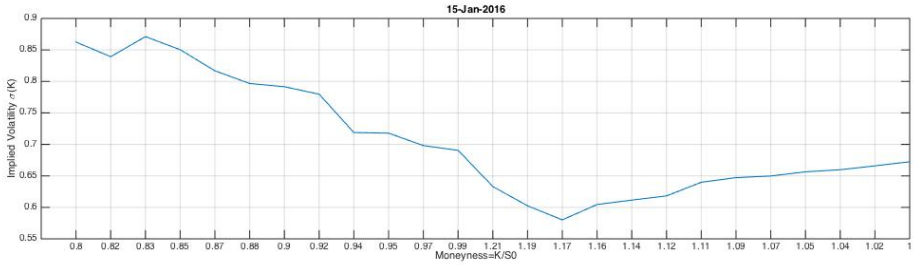
```
Call Option Implied Volatility
  Columns 1 through 4

    '20-Nov-2015'    '18-Dec-2015'    '15-Jan-2016'    '15-Apr-2016'

  Columns 5 through 6

    '17-Jun-2016'    '20-Jan-2017'

    0.3974    0.6813    0.8625    1.3314    1.7876    2.4199
    0.3752    0.7531    0.8393    1.3289    1.8107    2.4891
    0.3387    0.6993    0.8711    1.3019    1.7871    2.4629
    0.3329    0.6642    0.8504    1.2872    1.7219    2.3763
```

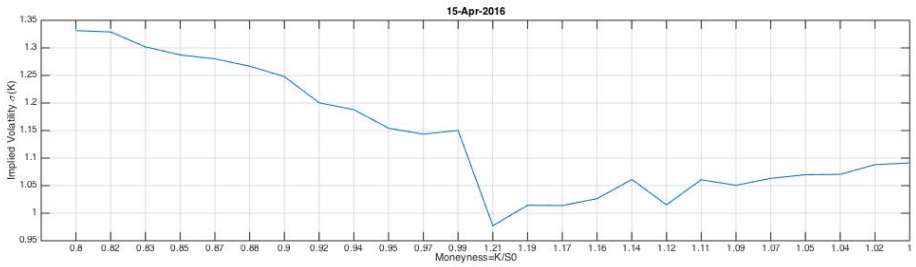
```
    0.3476    0.6540    0.8169    1.2801    1.7118    2.2959
    0.3355    0.6377    0.7967    1.2668    1.6905    2.2794
    0.3175    0.6193    0.7916    1.2479    1.6449    2.2341
    0.2970    0.5952    0.7797    1.2004    1.5974    2.1599
    0.2663    0.5911    0.7188    1.1879    1.5552    2.1309
    0.2616    0.5353    0.7179    1.1541    1.5110    2.1066
    0.2641    0.5269    0.6981    1.1435    1.4869    2.0683
    0.2577    0.5188    0.6905    1.1503    1.4463    2.0370
    0.2574    0.5059    0.6773    1.1486    1.4371    2.0186
    0.2566    0.5048    0.6750    1.1368    1.4171    1.9849
    0.2578    0.5030    0.6711    1.1359    1.3975    1.9638
    0.2582    0.5037    0.6681    1.1230    1.3787    1.9429
    0.2634    0.5038    0.6665    1.1170    1.3635    1.9119
    0.2644       NaN    0.6626    1.1183    1.3440    1.9069
       NaN       NaN    0.6616    1.1055    1.3350    1.8821
       NaN       NaN       NaN    1.0800    1.3269    1.8788
       NaN       NaN       NaN    1.1016       NaN    1.8697
       NaN       NaN       NaN    1.0662       NaN    1.8457
       NaN       NaN       NaN    1.0836       NaN    1.8300
       NaN       NaN       NaN    1.0976       NaN    1.8143
       NaN       NaN       NaN       NaN       NaN       NaN

Put Option Implied Volatility
  Columns 1 through 4

    '20-Nov-2015'    '18-Dec-2015'    '15-Jan-2016'    '15-Apr-2016'

  Columns 5 through 6

    '17-Jun-2016'    '20-Jan-2017'

       NaN       NaN       NaN       NaN       NaN       NaN
       NaN       NaN       NaN       NaN       NaN       NaN
       NaN       NaN       NaN       NaN       NaN       NaN
       NaN       NaN       NaN    1.2048       NaN       NaN
       NaN       NaN       NaN    1.1834       NaN       NaN
       NaN       NaN       NaN    1.1772       NaN       NaN
       NaN       NaN    0.7353    1.1400       NaN    2.0024
    0.2758    0.5542    0.7251    1.1269    1.4952    1.9755
    0.2704    0.5441    0.7107    1.1395    1.4645    1.9424
    0.2723    0.5321    0.6989    1.1004    1.4318    1.9301
    0.2655    0.5195    0.6908    1.0975    1.4079    1.8831
    0.2633    0.5117    0.6813    1.1029    1.3665    1.8734
    0.2588    0.5073    0.6724    1.0912    1.3513    1.8355
    0.2581    0.4957    0.6659    1.0881    1.3348    1.8133
    0.2593    0.4958    0.6598    1.0705    1.3049    1.7897
    0.2681    0.4914    0.6564    1.0698    1.2827    1.7647
    0.2717    0.4670    0.6498    1.0632    1.2692    1.7599
    0.2607    0.5122    0.6472    1.0506    1.2548    1.7380
    0.2150    0.4502    0.6399    1.0607    1.2481    1.6610
    0.2135    0.4377    0.6182    1.0153    1.2367    1.6561
    0.2004    0.4635    0.6115    1.0612    1.2242    1.6320
    0.1981    0.4965    0.6045    1.0263    1.1836    1.6120
       NaN    0.5032    0.5802    1.0140    1.1641    1.5515
```

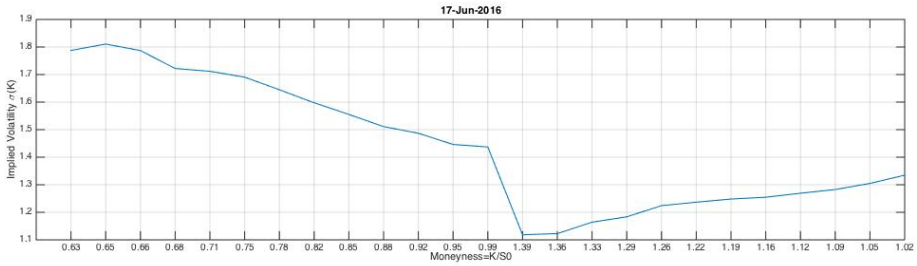
```
NaN    0.5007    0.6025    1.0144    1.1228    1.6020
NaN    0.5148    0.6330    0.9773    1.1183    1.5417
```

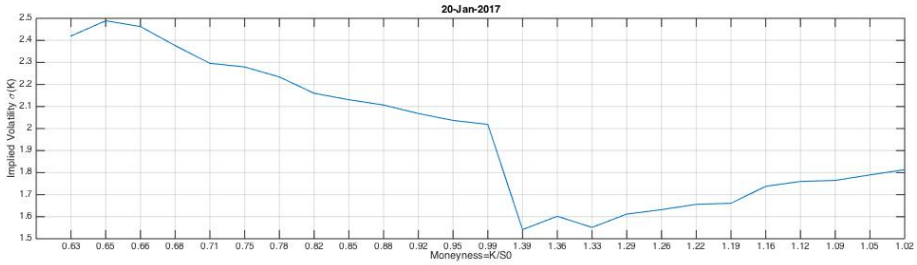













```
clear Kcol Vcol XL N n col dateL div filename grkL i impvC impvP impvP1...
    ind indC indP indP1 intv j k mC mP mP1 mnyC mnyP nsteps plotnum...
    sheetname tL xStr xVal optITMimpv
```

Function Files Attachment

```
% %   function price = gabBSpr(S0, K, T, r, div, sigma, optyp)
% %   %*********************************************************************
% % % Black-Scholes Option Pricing
% % %   Function calculates the price of a call or put option using
% % %   Black-Scholes analytical formula
% % %
% % %   gabBSpr(S0, K, T, r, div, sigma, optyp)
% % %
% % %=======================================================================
% % % INPUTS:
% % %   S0      - Initial price of the underlying asset
% % %
% % %   K       - Strike price of the option
% % %
% % %   T       - Option's maturity (fraction of year, i.e. in decimal)
% % %
% % %   r       - risk free rate-in annual terms (in decimal)
% % %
% % %   div     - annualized dividend yield (in decimal)
% % %
% % %   sigma   - annualized volatility (in decimal)
% % %
% % %   optyp   - option type +1=call, -1=put
% % %
% % %
% % %=======================================================================
```

```
% % % OUTPUTS:
% % %
% % %   price    - The option price from blach-scholes model
% % %
% % %
% % %
% % %=========================================================================
% % % EXAMPLE:
% % %
% % %       price = gabBSpr(60,50,4/12,0.03,0.018,0.0037,1)
% % %
% % %*************************************************************************
% %
% %     %Calculate d values
% %     d1=(log(S0./K)+((r-div+((sigma.^2)/2)).*T))./(sigma.*sqrt(T));
% %     d2=d1-(sigma.*sqrt(T));
% %
% %     %Calculate Black-Scholes Option Price
% %     price=optyp.*((S0.*exp(-div.*T).*normcdf(optyp.*d1))-...
% %         (K.*exp(-r.*T).*normcdf(optyp.*d2)));
% %
% % end


% % function [price, stp, lattice] = gabBNpr(S0,K,T,r,sigma,N,opty,opsy,div)
% % %*************************************************************************
% % % Binomial Option Pricing
% % %   Function calculates the price of a call/put,european/american option
% % %   using binomial tree model.
% % %
% % %   gabBNpr(S0,K,T,r,sigma,N,opty,opsy,div)
% % %
% % %=========================================================================
% % % INPUTS:
% % %   S0      - Initial price of the underlying asset
% % %
% % %   K       - Strike price of the option
% % %
% % %   T       - Option's maturity (fraction of year, i.e. in decimal)
% % %
% % %   r       - Risk free rate-in annual terms (in decimal)
% % %
% % %   sigma   - Annualized volatility (in decimal)
% % %
% % %   N       - The number of steps. (dt=T/N)
% % %
% % %   opty   - Option type +1=call, -1=put
% % %
% % %   opsy    - Option style Euro=0, Amer=1
% % %
% % %   div     - Annualized dividend yield (in decimal)
% % %
% % %
% % %=========================================================================
```

```
% % % OUTPUTS:
% % %
% % %   price   - The option price
% % %
% % %   stp     - The binomial stock price tree
% % %
% % %   lattice - The binomial option price tree
% % %
% % %
% % %=========================================================================
% % % EXAMPLE:
% % %
% % %       [price, ~, ~] = gabBNpr(60,50,4/12,0.03,0.3021,100,1,0,0.018)
% % %       [price, stp, lattice] = ...
% % %                           gabBNpr(60,50,4/12,0.03,0.3021,100,1,0,0.018)
% % %
% % %*************************************************************************
% %
% %
% %     deltaT = T./N;
% %     u=exp(sigma .* sqrt(deltaT));
% %     d=1./u;
% %     %Probability of a up movement
% %     p=(exp((r-div).*deltaT) - d)./(u-d);
% %
% %     stp = zeros(N,N+1);
% %     stp(1,1) = S0;
% %     % nchoosek could be useful.
% %     for j=1:N
% %         for i=1:j+1
% %             stp(i,j+1)=S0.*u.^(j+1-i).*d.^(i-1);
% %         end
% %     end
% %
% %     % Generate option price at the last steps
% %     lattice = zeros(N+1,N+1);
% %     for i=0:N
% %         lattice(i+1,N+1)=max(0 , opty.*((S0.*(u.^(N-i)).*(d.^i)) - K));
% %     end
% %
% %     % Iterate option price at the former steps from last steps
% %     for j=N-1:-1:0
% %         for i=0:j
% %             % Present Value of Expected Payoffs
% %             lattice(i+1,j+1) = exp(-r.*deltaT) .* ...
% %                 (p .* lattice(i+1,j+2) + (1-p) .* lattice(i+2,j+2));
% %             % Potential Early Exercise Payoffs
% %             exVal = opsy .* max(opty.*(stp(i+1,j+1)-K),0);
% %             lattice(i+1,j+1) = max(exVal,lattice(i+1,j+1));
% %         end
% %     end
% %
% %     % The present value of option
% %     price = lattice(1,1);
```

```
% %
% % end


% % function bsgreeks_gab=gabBSGrks(S0,K,T,r,div,sigma,optyp,greek)
% %   %*************************************************************************
% % % Greeks Calculation
% % %   Function calculates the greeks for options, only for european style
% % %   using Black-Scholes model.
% % %
% % %   gabBSGrks(S0,K,T,r,div,sigma,optyp,greek)
% % %
% % %=========================================================================
% % % INPUTS:
% % %   S0      - Initial price of the underlying asset
% % %
% % %   K       - Strike price of the option
% % %
% % %   T       - Option's maturity (fraction of year, i.e. in decimal)
% % %
% % %   r       - risk free rate-in annual terms (in decimal)
% % %
% % %   div     - annualized dividend yield (in decimal)
% % %
% % %   sigma   - annualized volatility (in decimal)
% % %
% % %   optyp   - option type +1=call, -1=put
% % %
% % %   greek   - which option sensitivity, i.e. Greek is to be calculated
% % %
% % %=========================================================================
% % % OUTPUTS:
% % %
% % %   bsGreeks_gab   - The greek value.
% % %
% % %
% % %
% % %=========================================================================
% % % EXAMPLE:
% % %
% % %       delta = gabBNGrks(60,50,4/12,0.03,0.018,0.3021,...
% % %                         1,0,'delta')
% % %
% % %*************************************************************************
% %
% %     %Calculate d values
% %     d1=(log(S0/K)+((r-div+((sigma^2)/2)).*T))/(sigma.*sqrt(T));
% %     d2=d1-(sigma.*sqrt(T));
% %
% %     %Calculate Option Sensitivities
% %     bsDelta=exp(-div.*T).*(optyp.*normcdf(optyp.*d1));
% %     bsVega=S0.*exp(-div.*T).*normpdf(d1).*sqrt(T);
% %     bsTheta = -(S0.*exp(-div.*T).*normpdf(d1).*sigma/(2.*sqrt(T))) ...
% %         - optyp.*r.*K.*exp(-r.*T).*normcdf(optyp.*d2) ...
```

```
% %         + optyp.*div.*S0.*exp(-div.*T).*normcdf(optyp.*d1);
% %     bsRho=(K.*T.*exp(-r.*T)).*(optyp.*normcdf(optyp.*d2));
% %     bsGamma=exp(-div.*T).*(normpdf(d1)/(S0.*sigma.*sqrt(T)));
% %     bsVanna = bsVega/S0 .* (1-d1/(sigma.*sqrt(T)));
% %     bsCharm = optyp.*div.*exp(-div.*T).*normcdf(optyp.*d1) ...
% %         - exp(-div.*T).*normpdf(d1).*(2.*(r-div).*sqrt(T)-d2.*...
% %         sigma)/(2.*T.*sigma);
% %     bsSpeed = -bsGamma/S0 .* (d1/(sigma.*sqrt(T))+1);
% %     bsZomma = bsGamma.*((d1.*d2-1)/sigma);
% %     bsColor = -exp(-div.*T) .* normpdf(d1)/(2.*S0.*T.*sigma.*...
% %         sqrt(T)).* (2.*div.*T + 1 + d1 .* (2.*(r-div).*sqrt(T)-...
% %         d2.*sigma)/sigma);
% %     bsVeta = S0 .* exp(-div.*T) .* normpdf(d1) .* sqrt(T).*(div +...
% %         (r-div).*d1/(sigma.*sqrt(T)) - (1+d1.*d2)/2.*T);
% %     bsVomma = bsVega .* d1 .* d2 / sigma;
% %     bsUltima = -bsVega/sigma^2 .* (d1.*d2.*(1-d1.*d2)+d1^2+d2^2);
% %
% %     %Output the Greek letter choices based on input grk
% %     grk ={'delta','gamma','vega','theta','rho','vanna','charm',...
% %         'speed','zomma','color','veta','vomma','ultima'};
% %     grkval = [bsDelta,bsGamma,bsVega,bsTheta,bsRho,bsVanna,...
% %         bsCharm,bsSpeed,bsZomma,bsColor,bsVeta,bsVomma,bsUltima];
% %     ext = strcmp(greek,grk);
% %     bsgreeks_gab = grkval(ext);
% %
% % end


% % function bsGreeks_gab=gabBNGrks(S0,K,T,r,div,sigma,N,optyp,opsy,smch,greek)
% % %*************************************************************************
% % % Greeks Calculation
% % %   Function calculates the greeks for options using binomial model.
% % %
% % %   gabBNGrks(S0,K,T,r,div,sigma,N,optyp,opsy,smch,greek)
% % %
% % %=========================================================================
% % % INPUTS:
% % %   S0      - Initial price of the underlying asset
% % %
% % %   K       - Strike price of the option
% % %
% % %   T       - Option's maturity (fraction of year, i.e. in decimal)
% % %
% % %   r       - risk free rate-in annual terms (in decimal)
% % %
% % %   sigma   - annualized volatility (in decimal)
% % %
% % %   N       - the number of steps. (dt=T/N)
% % %
% % %   optyp   - option type +1=call, -1=put
% % %
% % %   opsy    - Euro=0, Amer=1
% % %
% % %   smch    - The small change you apply to calculate.
```

```
% % %
% % %   greek   - which option sensitivity, i.e. Greek is to be calculated
% % %
% % %
% % %=========================================================================
% % % OUTPUTS:
% % %
% % %   bsGreek_gab   - The greek value.
% % %
% % %
% % %
% % %=========================================================================
% % % EXAMPLE:
% % %
% % %       delta = gabBNGrks(60,50,4/12,0.03,0.018,0.3021,...
% % %                         100,1,0,0.00001,'delta')
% % %
% % %
% % %*************************************************************************
% %
% %     % Parameters pre-process
% %     S0ps = S0; S0ms = S0;    sigps = sigma; sigms = sigma;
% %     Tps = T; Tms = T;    rps = r; rms = r;
% %     switch greek
% %         case 'delta'
% %             S0ps = S0 + smch;    S0ms = S0 - smch;
% %         case 'vega'
% %             sigps = sigma + smch;    sigms = sigma -smch;
% %         case 'theta'
% %             Tps = T + smch;    Tms = T - smch;
% %         case 'rho'
% %             rps = r + smch;    rms = r - smch;
% %         case 'gamma'
% %             temp = gabBNGamma(S0,K,T,r,div,sigma,N,optyp,opsy,smch);
% %     end
% %
% %     %Calculate d values & option price at S0
% %     bspr_ini = gabBNpr(S0,K,T,r,sigma,N,optyp,opsy,div);
% %
% %     %Calculate d values & option price at S0+small change, add _ps to
% %     %variables that change
% %     bspr_ps = gabBNpr(S0ps,K,Tps,rps,sigps,N,optyp,opsy,div);
% %
% %     %Calculate d values & option price at S0-small change, add _ms to
% %     %variables that change
% %     bspr_ms = gabBNpr(S0ms,K,Tms,rms,sigms,N,optyp,opsy,div);
% %
% %     %Calculate discreet approximation of delta with S0 plus small change
% %     delt_ps=(bspr_ps-bspr_ini)/smch;
% %
% %     %Calculate discreet approximation of delta with S0 minus small change
% %     delt_ms=(bspr_ms-bspr_ini)/(-1*smch);
% %
% %     %Calculate the discreet approximation of the delta as the average of
```

```
% %     %delt_ps & delt_ms
% %     bsGreeks_gab = (delt_ps+delt_ms)/2;
% %     if exist('temp','var')
% %         bsGreeks_gab = temp;
% %     end
% %
% % end
% %
% % function bsGamma_drk=gabBNGamma(S0,K,T,r,div,sigma,N,optyp,opsy,smch)
% %     %This part is refered to Dr.K's function.
% %     %Calculate d values & option price at S0
% %     %Calculate Delta at S0
% %     delt_S0=gabBNGrks(S0,K,T,r,div,sigma,N,optyp,opsy,smch,'delta');
% %
% %     %Calculate d values & option price at S0+small change=(S0+smch),
% %     %add _ps to variables that change.
% %     %Calculate Delta at S0+small change
% %     delt_S0ps=gabBNGrks(S0+smch,K,T,r,div,sigma,N,optyp,opsy,smch,'delta');
% %
% %     %Calculate d values & option price at S0-small change, add _ms to
% %     %variables that change.
% %     %Calculate Delta at S0-small change
% %     delt_S0ms=gabBNGrks(S0-smch,K,T,r,div,sigma,N,optyp,opsy,smch,'delta');
% %
% %     %Calculate discreet approximation of Gamma with delt_S0ps
% %     gamm_ps=(delt_S0ps-delt_S0)/smch;
% %
% %     %Calculate discreet approximation of Gamma with delt_S0ms
% %     gamm_ms=(delt_S0ms-delt_S0)/(-1*smch);
% %
% %     %Calculate the discreet approximation of the Gamma as the average of
% %     %gamm_ps & gamm_ms
% %     bsGamma_drk=(gamm_ps+gamm_ms)/2;
% %
% %  end


% % function Sigma = gabNRimpv(OptionValue, S0, K, T, r, div, optyp,...
% %     tolerance,itermax)
% % %*************************************************************************
% % % Implied Volatility Calculation
% % %   Function calculates the implied volatility using Newton-Raphson Method
% % %   based on black-scholes model.
% % %
% % %   gabNRimpv(OptionValue, S0, K, T, r, div, optyp,tolerance,itermax)
% % %
% % %=========================================================================
% % % INPUTS:
% % %
% % %   OptionValue - The observed value of this option.
% % %
% % %   S0          - Initial price of the underlying asset
% % %
% % %   K           - Strike price of the option
```

```
% % %
% % %   T           - Option's maturity (fraction of year, i.e. in decimal)
% % %
% % %   r           - Risk free rate-in annual terms (in decimal)
% % %
% % %   div         - Rnnualized dividend yield (in decimal)
% % %
% % %   optyp       - Option type +1=call, -1=put
% % %
% % %   tolerance   - The numeric error you can tolerate.
% % %
% % %   itermax     - The max iterations.
% % %
% % %=========================================================================
% % % OUTPUTS:
% % %
% % %   Sigma   -   The estimated implied volatility
% % %
% % %
% % %
% % %=========================================================================
% % % EXAMPLE:
% % %
% % %       Value = 24.99; S0 = 309.43; K = 310; T = 90/360; rf = 0.00337;
% % %       div = 0; optyp = 1; tolerance = 1e-03; itermax = 1000;
% % %
% % %       ImpliedVol = gabNRimpv(Value, S0, K, T, r, div, optyp,...
% % %                       tolerance,itermax)
% % %
% % %*************************************************************************
% %
% %     % Parameter inputs checking
% %     input = {OptionValue, S0, K, T, r, div, optyp};
% %     if sum(cellfun(@isvector,input)) < 7
% %         error('input parameters must be vector or scalar.')
% %     end
% %     paraL = arrayfun(@(x) length(cell2mat(x)),{OptionValue,K,T},...
% %         'UniformOutput',false);
% %     if length(unique(cell2mat(paraL))) > 1
% %         error('CallPrice K T parameters must be in same length.')
% %     end
% %
% %     % Prepare for vector parameters input
% %     L = length(OptionValue);
% %     values = OptionValue; strike = K; TtoM = T;
% %     Sigma = zeros(L,1);
% %     for i = 1:L
% %         OptionValue = values(i); K = strike(i); T = TtoM(i);
% %         sigma = sqrt(2.*pi./T).*OptionValue./S0; %initial guess
% %         priceDiff = inf; %initial tolerance
% %         iternum = 0;
% %         % Object function
% %         dprice = @(x) gabBSpr(S0, K, T, r, div, x, optyp)-OptionValue;
% %         % Deriatives of dprice w.r.t sigma, which is Vega
```

```
% %         f_vega = @(x) gabBSGrks(S0,K,T,r,div,x,optyp,'vega');
% %         % Main Newton Raphson Method Iteration
% %         while iternum <= itermax && abs(priceDiff) > tolerance && ...
% %                 ~isnan(sigma)
% %             iternum = iternum+1;
% %             priceDiff = dprice(sigma);
% %             vegaEst = f_vega(sigma);
% %             %Update sigma:
% %             sigma = sigma - (priceDiff)./vegaEst;
% %         end
% %         try
% %             Sigma(i) = sigma;
% %         catch
% %             Sigma(i) = NaN;
% %         end
% %
% %     end
% %
% % end


% % function [myimpV,output] = gabImpv(OptionValue, S0, K, T, r, div, N, ...
% %     optyp, opsy,model,method,varargin)
% % %*************************************************************************
% % % Implied Volatility Calculation
% % %   Function calculates the implied volatility using NR, fzero, fsolve and
% % %   rough incremental methods, based on black-scholes model or binomial
% % %   model.
% % %
% % %   gabImpv(OptionValue, S0, K, T, r, div, optyp,model,method,varargin)
% % %
% % %=========================================================================
% % % INPUTS:
% % %
% % %   OptionValue - The observed value of this option.
% % %
% % %   S0          - Initial price of the underlying asset
% % %
% % %   K           - Strike price of the option
% % %
% % %   T           - Option's maturity (fraction of year, i.e. in decimal)
% % %
% % %   r           - Risk free rate-in annual terms (in decimal)
% % %
% % %   div         - Rnnualized dividend yield (in decimal)
% % %
% % %   N           - The number of steps. (dt=T/N)
% % %
% % %   optyp       - Option type +1=call, -1=put
% % %
% % %   opsy        - Option style Euro=0, Amer=1
% % %
% % %   model       - Pricing model. 'BS' = black-scholes, 'BN' = binomial
% % %
```

```
% % %   method      - Approximation method. 'newton' = newton-raphson,
% % %                 'fzero' = fzero, 'fsolve' = fsolve, 'incremental' =
% % %                 incremental.
% % %
% % %   varargin{:}
% % %           1   - Tolerance: The numeric error you can tolerate.
% % %
% % %           2   - Initial Guess or Itermax: The initial guess for x-point
% % %                   or the max iterations.
% % %           3   - Incremental value for 'incremental' method
% % %
% % %=========================================================================
% % %   Note:
% % %       Incremental need 4 control variables --- tolerance, iniGuess,
% % %       incremental value.
% % %       newton need 2 control variables --- tolerance, intermax.
% % %       fzero need 2 control variables --- tolerance, iniGuess.
% % %       fsolve need 2 control variables --- tolerance, iniGuess.
% % %=========================================================================
% % % OUTPUTS:
% % %
% % %   myimpV   -   The estimated implied volatility.
% % %
% % %   output   -   The approximation details.
% % %
% % %=========================================================================
% % % EXAMPLE:
% % %
% % %       Value = 24.99; S0 = 309.43; K = 310; T = 90/360; rf = 0.00337;
% % %       div = 0; optyp = 1; tolerance = 1e-03; iniGuess = 1000;
% % %       model = 'BS'; method = 'fzero';
% % %
% % %       ImpliedVol = gabImpv(OptionValue, S0, K, T, r, div, optyp,...
% % %                       model,method,tolerance, iniGuess)
% % %
% % %*************************************************************************
% %
% %
% %
% %
% %     % Object function
% %     f = @(x) objfcn(x,S0,K,T,r,OptionValue,div,N,optyp,opsy,model);
% %
% %     % Newton Raphson Method
% %     if strcmp(method,'newton')
% %         tolerance = varargin{1}; itermax = varargin{2};
% %         myimpV = gabNRimpv(OptionValue, S0, K, T, r, div, optyp,...
% %             tolerance,itermax);
% %         % if this method fails, use next method instead.
% %         if ~exist('myimpV','var')
% %             if isnan(myimpV)
% %                 method = 'fzero';
% %                 varargin{2} = 1/itermax;
% %                 display(['Warning: the group with strike price of ',...
```

```
% %                     num2str(K),' cannot solve the function using ',...
% %                     'Newton Raphson Method. Use fzero insted.\n'])
% %             end
% %         end
% %     end
% %
% %     % fzero method
% %     if strcmp(method,'fzero')
% %         tolerance = varargin{1}; iniGuess = varargin{2};
% %         options = optimset('TolX', tolerance, 'Display', 'off');
% %         [myimpV,fval,ef,output] = fzero(f,iniGuess,options);
% %         output.method = 'fzero';output.fval = fval; output.ef = ef;
% %         output.output=output;
% %         % if this method fails, use next method instead.
% %         if ~exist('myimpV','var')
% %             if isnan(myimpV)
% %                 method = 'fsolve';
% %                 display(['Warning: the group with strike price of ',...
% %                     num2str(K),' cannot solve the function using',...
% %                     ' fzero. Use fsolve insted.\n'])
% %             end
% %         end
% %     end
% %
% %     % fsolve method
% %     if strcmp(method,'fsolve')
% %         tolerance = varargin{1}; iniGuess = varargin{2};
% %         options = optimset('TolX', tolerance,'Display', 'off');
% %         [myimpV,fval,ef,output] = fsolve(f,iniGuess,options);
% %         output.method = 'fsolve'; output.fval = fval; output.ef = ef;
% %         output.output=output;
% %         % if this method fails, use next method instead.
% %         if ~exist('myimpV','var')
% %             if isnan(myimpV)
% %                 myimpV = 'incremental';
% %                 varargin{3} = 0.0001; varargin{4} = 0.0001;
% %                 display(['Warning: the group with strike price of ',...
% %                     num2str(K),' cannot solve the function using ',...
% %                     'fsolve. Use "incremental" method, which is ',...
% %                     'rough, insted.\n'])
% %             end
% %         end
% %     end
% %
% %     % Incremental Method
% %     if strcmp(method,'incremental')
% %         tolerance = varargin{1}; iniVola= varargin{2}; %initial guess
% %         IncreVal = varargin{3};  itermax = 1/iniVola;
% %         for vola = iniVola:IncreVal:itermax
% %             delta = f(vola);
% %             comp = delta;
% %             if abs(delta) <= tolerance || comp*delta < 0
% %                 myimpV = vola;
% %                 break
```

```
% %             end
% %         end
% %         if ~exist('myimpV','var')
% %             if vola == itermax
% %                 myimpV = NaN;
% %                 disp(['Iteration ended. The estimate is not ',...
% %                     'closer than ',num2str(IncreVal)])
% %             else
% %                 myimpV = vola;
% %             end
% %         end
% %     end
% %
% %     if ~exist('myimpV','var')
% %         if isnan(myimpV)
% %             myimpV = NaN;
% %             display(['Warning: the group with strike price of ',...
% %                 num2str(K), ' have no solution based on gabImpv',...
% %                 ' or the solution is exactly NaN .\n'])
% %         end
% %     end
% %
% % end
% %
% %
% % function delta = objfcn(volatility, S0, K, T, r, OptionPrice,div,...
% %     N, optyp,opsy,flag)
% %     switch flag
% %         case 'BS'
% %             priceEst = gabBSpr(S0, K, T, r, div,volatility,optyp);
% %         case 'BN'
% %             priceEst = gabBNpr(S0,K,T,r,volatility,N,optyp,opsy,div);
% %     end
% %     delta = OptionPrice - priceEst;
% % end


% % function [opt,rawopt] = OptionKVTD(filename,sheetname,Kcol,Vcol,S0,rf,div,N,..
% %     optyp,opsy)
% % %*************************************************************************
% % % Generate 3D plot structure
% % % This function is going to generate a struct variables containing
% % %     struct(Strike price, Option Value, Time to Maturity, Expiration Date)
% % %   from excel files in specific format.
% % %
% % %   OptionKVTD(filename,sheetname,Kcol,Vcol,S0,rf,div,optyp)
% % %
% % %=========================================================================
% % % INPUTS:
% % %   filename      - Excel name
% % %
% % %   sheetname     - Excel sheetname
% % %
% % %   Kcol          - Column number of strie price location
```

```
% % %
% % %   Vcol          - Column number of option value location
% % %
% % %   S0            - Initial price of the underlying asset
% % %
% % %   rf            - Risk free rate-in annual terms (in decimal)
% % %
% % %   div           - Annualized dividend yield (in decimal)
% % %
% % %   N             - The number of steps. (dt=T/N)
% % %
% % %   opty          - Option type +1=call, -1=put
% % %
% % %   opsy          - Option style Euro=0, Amer=1
% % %
% %
% % %
% % %
% % %=========================================================================
% % % OUTPUTS:
% % %
% % %   opt   - A structure for 3-D plot.
% % %
% % %=========================================================================
% % % EXAMPLE:
% % %       S0 = 309.58; rf=0.0017; div=1.39/100; optyp=1; Kcol=2; Vcol=3;
% % %       filename = 'callputprice';sheetname = 'Call' ;
% % %
% % %       opt = OptionKVTD(filename,sheetname,Kcol,Vcol,S0,rf,div,optyp)
% % %
% % %
% % %*************************************************************************
% %
% %
% %     % Import data and pre-process
% %     rawopt = xlsread(filename,sheetname);
% %     optdate = rawopt(:,1);
% %     datetype = unique(optdate);
% %     % Fix import time problem on mac
% %     datecol = cellstr(datestr(FixMacTime(optdate)));
% %     opt.date = reshape(datecol,length(datecol)/length(datetype),...
% %         length(datetype));
% %
% %     % Create option structure
% %     for i = 1:length(datetype)
% %         ind = optdate==datetype(i);
% %         opt.K(:,i) = rawopt(ind,Kcol);
% %         opt.V(:,i) = rawopt(ind,Vcol);
% %         opt.T(:,i) = repmat(15/360,sum(ind),1);
% %         opt.impv(:,i) = gabImpv(opt.V(:,i), S0, opt.K(:,i), opt.T(:,i),...
% %             rf, div, N, optyp, opsy, 'BS','newton',1e-05,10000);
% % %         opt.impv(:,i) = gabNRimpv(opt.V(:,i), S0, opt.K(:,i), opt.T(:,i),...
% % %             rf, div, optyp,1e-05,10000);
% %     end
```

```
% %
% %     % Calculate Moneyness
% %     opt.M = opt.K./S0;
% % end


% % function FixedNum = FixMacTime(ExcelNum)
% %   %*************************************************************************
% % % Fix Excel Import Problem For Matlab On Mac System
% % %     This function is going to fix the import problem about date for
% % %     Matlab User on mac.
% % %
% % %   FixMacTime(ExcelNum)
% % %
% % %=========================================================================
% % % INPUTS:
% % %   ExcelNum    - The date number you import from excel without any
% % %   modification
% % %
% % %=========================================================================
% % % Note:
% % %     If import datestr to Matlab on OSX from excel, they will be the
% % %     following, which is not correct.
% % %     42328 == '21-Nov-0115'
% % %     42356 == '19-Dec-0115
% % %     In excel, they are supposedd to be match respectively as the
% % %     following:
% % %     42328 == '20-Nov-2015'
% % %     42356 == '18-Dec-2015
% % %=========================================================================
% % % OUTPUTS:
% % %
% % %   FixedNum   - The date number correspond to the excel date.
% % %
% % %=========================================================================
% % % EXAMPLE:
% % %
% % %       realNum = Fixed(42328)
% % %
% % %*************************************************************************
% % %
% %     dnum = 42328;
% %     dnum_f = datenum('20-Nov-2015');
% %     difference = dnum_f - dnum;
% %
% %     FixedNum = ExcelNum + difference;
% % end
```

Published with MATLAB® R2014b