Comparative Empirical Study on Shanghai 50 ETF Options Pricing Model

--Based on Black-Scholes-Merton Model and Heston Model

1. Question:

Shanghai 50 ETF option is the first exchange-traded option in Chinese financial market. As we all know, Black-Scholes-Merton Model is the most widely used option pricing model in the industry and academia. However, there are many assumptions in Black-Scholes model which the real world can't satisfy, in particular, many financial instruments have fat tail and high kurtosis property. As a result, the question I want to explore is that whether 50 ETF, the underlying asset, have this kind of property? and if so, whether we should still use Black-Scholes model to price the option? or other models such as Heston model could be better? Hereinafter, I just talk about **call option**.

2. Approach

All the data was downloaded from Wind.

2.1 Explore some properties of 50 ETF

2.1.1 Non-normal distribution

First, for each day from 2016.1.4 to 2018.6.13, I take $\text{logreturn} = \log(S_i) - \log(S_{i-1})$ (S_i $denotes\ the\ net\ unit\ asset\ value\ of\ i^{th}\ day$) as the logarithmic rate of return of that day. Through the histogram of logarithmic return (see Figure

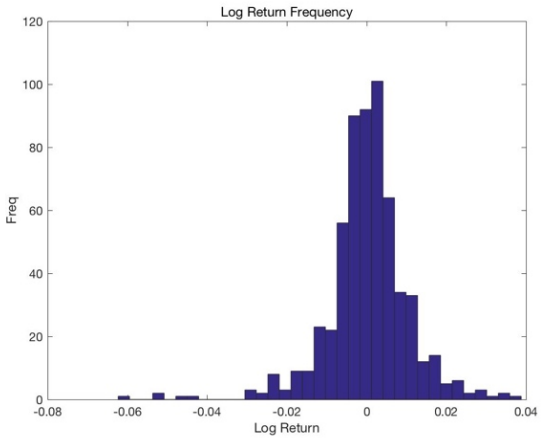

Figure 1 Histogram of Logarithmic Return

1), we notice that the distribution has fatter tail and higher kurtosis compared to normal distribution. More accurately, I did one JB test that returns 1, which shows that it's not normally distributed. The kurtosis is 8.6342, which is higher than that of normal distribution(3), and the skewness is -0.8735, the minus sign means it's left-skewed.

There are many other properties due to non-normal distribution.

2.1.2 Volatility smiles

Take option data on 2017.12.29 as an example, calculate the implied volatility of each option(using the Matlab built-in function: blsimpv), with different maturities and different strike prices, and then plot the relation between implied volatility and strike price grouped by maturity, which is respectively 1-month, 2-month, 3-month and 6-month(see Figure 2). The input parameters needed for blsimpv function include the current underlying asset price S, exercise price K, annualized continuously compounded risk-free rate r, time to expiration T-t, and the option price c. I use the 1-year treasury bond interest rate as risk free rate r (see Treasury bond interest rate.xlsx). Cause the expiration date is the fourth Wednesday if not in holidays. Here, for option expired in January, February, March and June, the expiration date is respectively 2018.1.24, 2018.2.28, 2018.3.28, and 2018.6.27. As a result, time to expiration can be directly calculated from the difference of the datenum format of current date and expiration date. When I was trying to get the graph, a very important point is to sort the data by strike price in ascending order, otherwise the graph will be a mess.

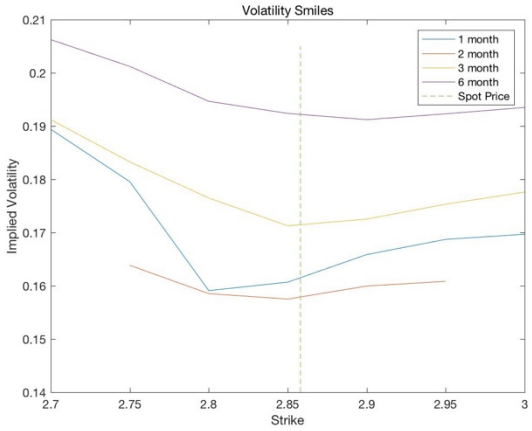

Figure 2 Volatility Smiles

From figure 2, we can see that the shape of implied volatility curve looks like a smile, which is the reason why it is called “Volatility Smile”. The fundamental reason for the existence of a smile in volatility is that the distribution of the stock is not lognormal, or that the stock's log return is not normal. Because BS model assumes the volatility is a constant, if the option price really conforms to the BS formula then the implied volatility should be a constant rather than a smile, which can be seen as a flaw of BS model. A possible explanation is that because of the fat tail, the probability of extreme situations is higher, the value of a deep out of money or deep in the money option is higher than that calculated by BS model.

2.1.3 Jumps

Calculate the probabilities under which the daily change is above once, twice and thrice standard deviation and compare it with that under normal distribution. (see Table 1)

Table 1 Probabilities of extreme situations in real market and Normal Distribution

P	Real Market	Normal Distribution	Relation

above std	18.67%	31.73%	<
above 2*std	5.17%	4.55%	>
above 3*std	3.79%	0.26%	>

From Table 1, we notice that the probability of "above std" in real market is lower than in normal distribution, which exactly shows the high kurtosis of the distribution, and the more it toward extreme values, the more likely it is to appear than the probability of normal distribution.

All these three properties don't meet the assumptions of BS model. Because BS model assumes constant volatility, log-normal distribution of return, and certainly no jumps. As a result, BS model can't meet the need of more accurately pricing. Maybe we can consider other models with stochastic volatility or even with jumps. Hereinafter, I calibrate one stochastic volatility model—Heston model.

2.2 Comparison of option pricing models

In this part, I use the most classic model—Black-Scholes model and one stochastic volatility model—Heston model to price the option. Fortunately, 50 ETF option is a kind of European option, which makes it more convenient to price. Compare the model fitted price with real market price, specifically using the difference between these two prices as the pricing error, to evaluate the goodness of fit.

2.2.1 Black-Scholes Model

One of the reasons that Black-Scholes model is most widely used is that it's simple and easy to calculate. The explicit solution is

$$\mathrm{C} = \mathrm{SN}(d_1) - Ke^{-rt}N(d_2)$$

where

$$d_1 = \frac{\ln\left(\frac{\mathrm{S}}{\mathrm{K}}\right) + \left(\mathrm{r} + \frac{\sigma^2}{2}\right)\mathrm{t}}{\sigma\sqrt{t}}$$

$$d_2 = d_1 - \sigma\sqrt{t}$$

The parameters in the above equations represent the following:

- S is the current price of underlying asset
- K is the exercise price
- r is the risk-free rate of return
- σ is the volatility of underlying asset
- t is the time to maturity.

S, K, r and t can be directly founded in 50etf.xlsx, 2017.12.29.xlsx, or Treasury bond interest rate.xlsx. So we only need to figure out the annualized volatility of underlying asset. Here, I calculate the standard deviation of past 20 days' log return rate and annualize it by multiplying $\sqrt{252}$ to get the annualized volatility. Then, just

use the Matlab built-in function blsprice (Price, Strike, Rate, Time) to calculate the BS price. The comparison of BS price and market price for options of different maturity is shown in Figure 3.

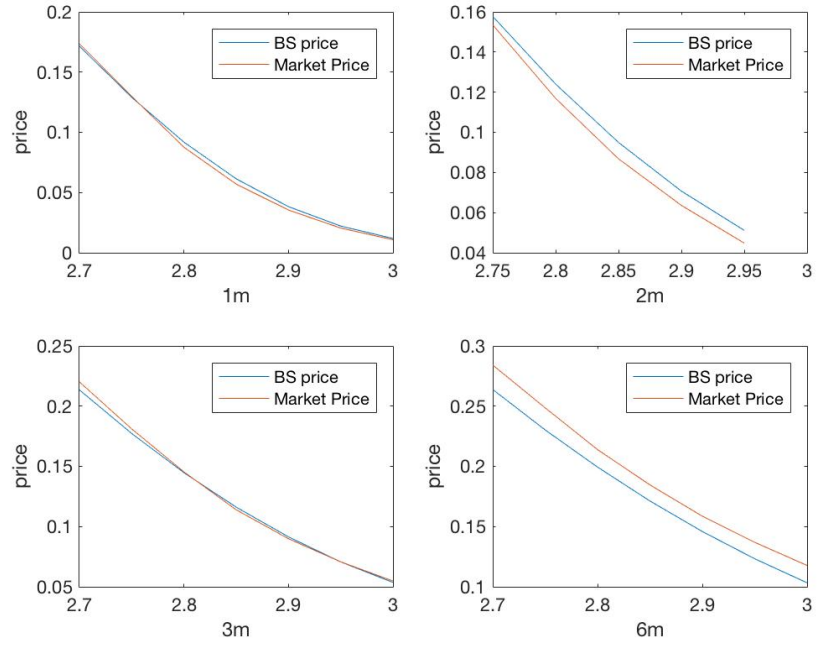

Figure 3 Black-Scholes Model Fitting results 2017.12.29

Figure 3 shows that Black-Scholes model fits really well for options of 1-month and 3-month maturity. However, for options of 2-month and 6-month maturity, there's a bigger deviation from market price.

2.2.2 Heston Model

The basic Heston model assumes that S_t, the price of the asset, is determined by a stochastic process:

$$\mathrm{d}S_t = \mu S_t + \sqrt{v_t} S_t dW_t^S$$

where v_t, the instantaneous variance, is a CIR process:

$$\mathrm{d}v_t = \kappa(\theta - v_t)dt + \xi\sqrt{v_t}dW_t^v$$

and W_t^s, W_t^v are Wiener processes (i.e., random walks) with correlation ρ, or equivalently, with covariance ρdt (because $W_t^s W_t^v = \rho dt$).

The parameters in the above equations represent the following:

- μ is the rate of return of the asset.
- θ is the long variance, or long run average price variance; as *t* tends to infinity, the expected value of v_t tends to θ.
- κ is the rate at which v_t reverts to θ.
- ξ is the volatility of the volatility, or 'vol of vol', and determines the variance of v_t.

Compared with Black-Scholes model, there are too many parameters to be calibrate in Heston model. However, it's so fortunate that we can derive an explicit solution for Heston model with Fourier-based numerical method, which makes the calibration easier. Although the solution is a little complicated, since the formula of each parameter is long and it's very possible to write the formula incorrectly when calibrating the model. Through HestonCall.m and CF_Heston.m, I successfully construct the explicit solution of Heston model, which can be used to not only calibrate the model, but price the option when you input some parameters needed. What's more, HestonDiff returns a vector of values which represent the pricing error of each option. Finally, use Matlab built-in function lsqnonlin() to search the parameters which minimize the sum of square difference. Let initial values of parameters to be [6.5482 0.0731 2.3012 -0.4176 0.1838], we get the optimal parameters as follows:

kap	th	sig	rho	vt
1.3569	0.1156	0.9884	-0.0185	0.0252

With all the parameters being figured out, we can use HestonCall.m to calculate the Heston theoretical price. The comparison of Heston price and market price for options of different maturity is shown in Figure 4.

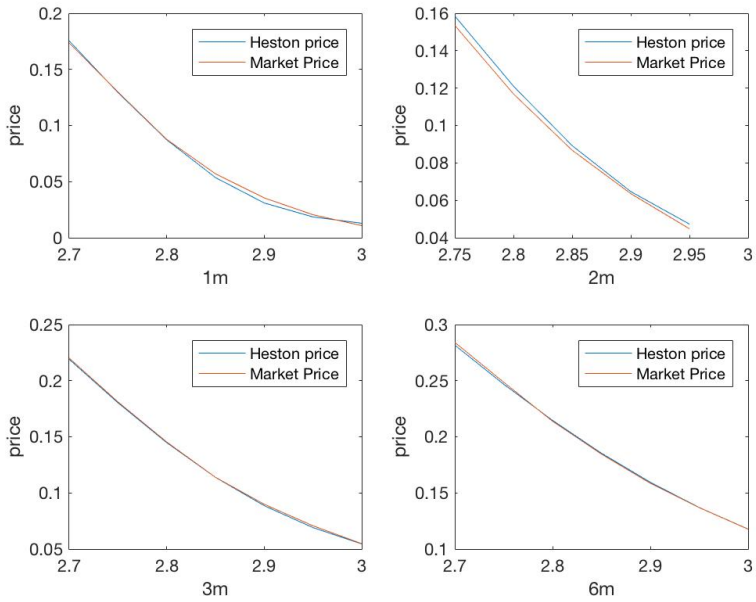

Figure 4 Heston Model Fitting results 2017.12.29

Compared with Figure 3, Figure 4 seems better, that is to say, Heston model seems to fit the model better, since the difference between Heston price and market price seems smaller than that of Black-Scoles model. More accurately, we can calculate the mean and standard deviation of pricing error of two models(see Table 2).

Obviously, both mean and standard deviation pricing error of Heston model are much smaller than that of Black-Scholes model. Thus, from this perspective, Heston model is much better than Black-Scholes model.

Table 2 mean and std of pricing error 2017.12.29

	Mean ERR	Std ERR
Black-Scholes model	0.00666662	0.00858756
Heston model	0.00163262	0.00216893

Nevertheless, does this result mean Heston model is absolutely better than Black-Scholes model? Of course no. Because what I did for Heston model is to calibrate the parameters using the data of that day, if it couldn't be fit the option price of next day or even further, it would be useless. That is to say, we should test the robustness of the model.

2.3. Robustness Test

If the parameters calibrated yesterday can be used to forecast the price today, it's at least useful. Thus, we should at least do 1-day roubustness test. The next trading day of 2017.12.29 is 2018.1.2. Use the methods similar as before, we can get two similar pictures of the fitting result Black-Scholes model and Heston model. Here, use the parameters already calibrated instead of calibrating again in Heston model. However, for Black-Scholes model, just calculate the new past-20 standard deviation and annualize it. Then we will obtain Figure 5 and Figure 6.

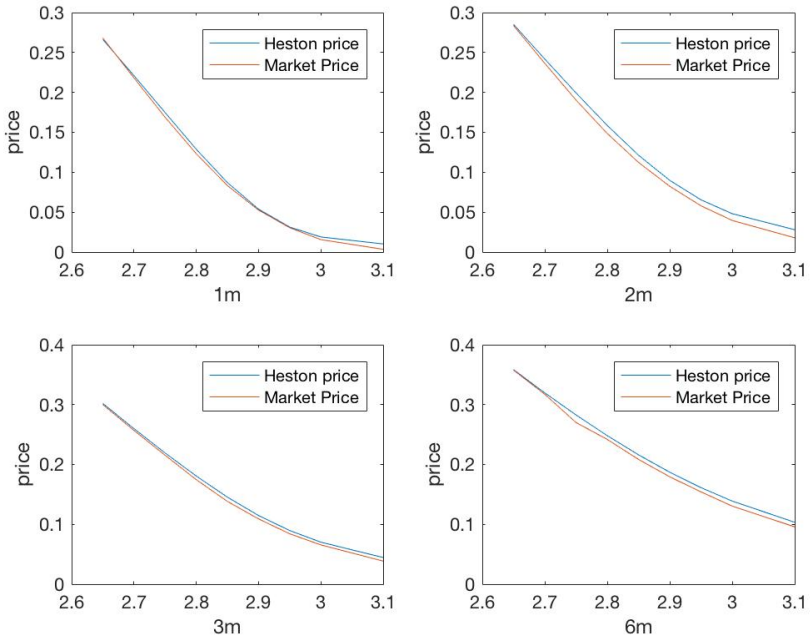

Figure 4 Heston Model Fitting results 2018.1.2(using parameters calibrated on 2017.12.29)

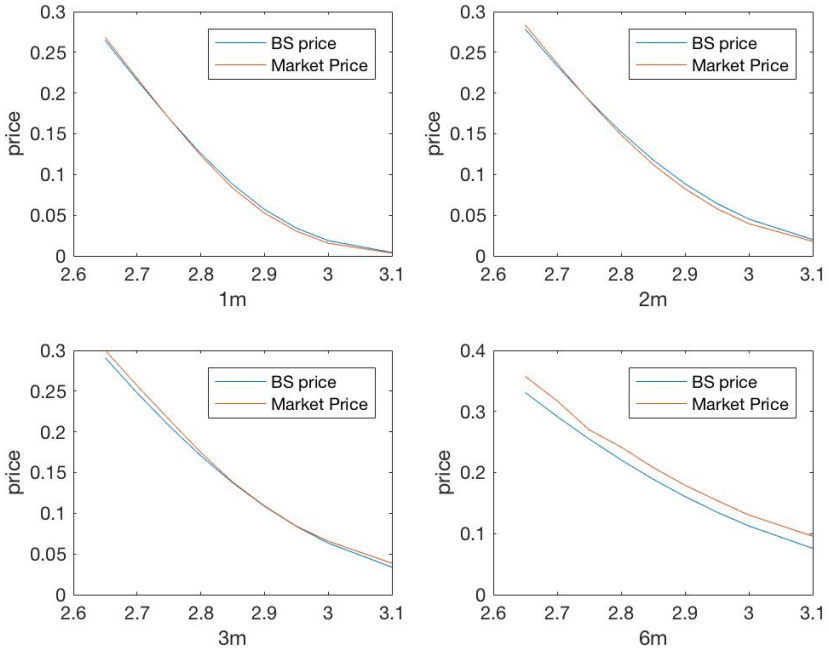

Figure 5 Black-Scholes Model Fitting results 2018.1.2

Similarly, calculate the mean and standard deviation of pricing error of two models. (see Table 3)

Table 3 mean and std of pricing error 2018.1.2

	Mean ERR	Std ERR
Black-Scholes model	0.00793604	0.00987289
Heston model	0.00568657	0.00313625

Table 3 shows that the mean and standard deviation of Heston model are still smaller than that of Black-Scholes model, i.e., it's still more accurate to price with Heston model.

3. Conclusions

Black-Scholes model and Heston model both have its own strengths and weaknesses. As for Black-Scholes model, it is really simple and can achieve a not bad fitting result, but it's not so accurate as Heston model. However, pricing with Heston model can achieve better fitting result, but it is more complicated, therefore cost more time and more energy. Maybe that's one of the reasons why industry people usually prefer to use Black-Scholes model to price options.

In conclusion, we can't say that some model is right or wrong, but can just say that which model fits the market better. Because there're a lot of factors couldn't be modeled and quantified. No model is totally right, every model has pricing errors. What we can do is just to develop and use the model which minimize pricing errors.

However, if sometimes we don’t need to know the price that accurate, we can use Black-Scholes model to get one result quickly.

4. Further Research

As I mentioned in the 2.1.3 part, jumps are possible to happen. Maybe adding the jump factor to the model can optimize the model and achieve better fitting results. In addition, for testing the robustness of Heston model, I just tested the robustness of those parameters used for next day. Actually, it’s better to test more days to figure out how long the robustness could maintain. It contributes to the pricing and forecasting.

Codes:

```
% Chunyan Lei
% 2018.6.10

% Final Project
% Black-Scholes model & Heston model
% data: 50ETF call options which can be traded on 2017.12.29
(maturity:1m 2m 3m 6m)
% data source: Wind

%-------------------------------------------------------------------
% 1. Import 50ETF data(underlying asset) and find some properties
%-------------------------------------------------------------------

clear all
close all

data_asset='/Users/leichunyan/Downloads/50etf.xlsx'

% I don't know why I can't xlsread asset_value directly at this step
% So I use the way a little bit more complicated as follows:
[~,~,asset_value] = xlsread(data_asset, 'E2:E601');
asset_value=str2num(cell2mat(asset_value));
[~,matlabDateVec]=xlsread(data_asset,'A2:A601');
[year,month,day]=datevec(matlabDateVec);
dateNumber = datenum(year, month, day);

S0=2.858;
r0=0.037909;
%1.1 NOT NORMAL DISTRIBUTION
%1.1.1
% This part is to show that 50ETF return is not normally distributed
%
% calculate the log rate of return
logr=zeros(600,1);
for i=1:599
    logr(i)=log(asset_value(i))-log(asset_value(i+1));
end

% From the picture, we can see that the distribution has high
kurtosis and fat tail compared to normal distribution. -> it's not
normally distributed.
hist(logr,35);
title('Log Return Frequency')
ylabel('Freq')
xlabel('Log Return')
print('50 ETF Log Return Frequency','-djpeg');
% Another way which is more convincable: use JB test
isNormal=jbtest(logr);   %isNormal=1 shows that logr is not normally
distributed.
logr_kurt=kurtosis(logr);  %>3 shows the high kurtosis property(cause
the kurtosis of Normal distribution==3).
logr_skew=skewness(logr);  %<0 shows left-skewed

% 1.2 Implied Volatility Smiles
% calculate the implied volatility of options on 2017.12.29
data_imp='/Users/leichunyan/Downloads/2017.12.29.xlsx';
maturities = xlsread(data_imp, 'G2:G27');
global strike;
global close_price;
strike=xlsread(data_imp,'D2:D27');
close_price=xlsread(data_imp,'O2:O27');
```

```
%sort by stirke in ascending order
[n,m]=size(maturities);
for s=1:n
    for k=1:(n-s)
        if strike(k)>strike(k+1)
          tmp1=strike(k);
          strike(k)=strike(k+1);
          strike(k+1)=tmp1;
          tmp2=maturities(k);
          maturities(k)=maturities(k+1);
          maturities(k+1)=tmp2;
          tmp3=close_price(k);
          close_price(k)=close_price(k+1);
          close_price(k+1)=tmp3;
        end
    end
end

% set the maturity date for each option
global maturity_date;
for i=1:26     % the 4th Wednesday if it is not legel holiday
    if maturities(i)==1  % maturity date is 2018.1.24
        maturity_date(i)=datenum(2018,1,24);
    elseif maturities(i)==2  %2018.2.28
        maturity_date(i)=datenum(2018,2,28);
    elseif maturities(i)==3  %2018.3.28
        maturity_date(i)=datenum(2018,3,28);
    else
        maturity_date(i)=datenum(2018,6,27);
    end
end

T=((maturity_date-datenum(2017,12,29))/365)';
global imp_vol;
for i=1:26
    imp_vol(i)=blsimpv(S0,strike(i),r0,(maturity_date(i)-
datenum(2017,12,29))/365,close_price(i));
end

figure(2)
for i=[1,2,3,6]
    tmpx=[];
    tmpy=[];
    cnt=1
    for j=1:26
        if maturities(j)==i  % find the maturity equals to 1,2,3,6
months seperately
           tmpx(cnt)=strike(j);    %strike
           tmpy(cnt)=imp_vol(j);   %implied volatility
           cnt=cnt+1;
        end
    end
    plot(tmpx,tmpy)
    hold on
end
plot([S0 S0],[0.14,0.205],'--');
legend('1 month','2 month','3 month','6 month','Spot Price');
xlabel('Strike');
ylabel('Implied Volatility');
title('Volatility Smiles')
print('Volitility Smiles','-djpeg');
hold off
```

```
% 1.3 JUMPS
% whether there exists jump
% calculate the probabilities under which the daily change is above
3*standard deviation
% and compare with the probabilty under normal distribution
logr_std=std(logr);
cnt1=0;cnt2=0;cnt3=0;
for i=2:600
    if logr(i)-logr(i-1)>3*logr_std
        cnt3=cnt3+1;
    end
    if logr(i)-logr(i-1)>2*logr_std
        cnt2=cnt2+1;
    end
    if logr(i)-logr(i-1)>logr_std
        cnt1=cnt1+1;
    end
end

p1=cnt1/600;   %p1<31.7%   because of high kurtosis
p2=cnt2/600;   %p2>4.4%
p3=cnt3/600;   %p3>0.3%   i.e. there may be some extreme values

%-------------------------------------------------------------------
% 2.1 Option Pricing with Black-Scholes Model
%-------------------------------------------------------------------

% Calculate the past 20 trading days' historical volatility
(2017.12.1-2017.12.28)
% and annualize it.
[~,~,asset_value_past20] = xlsread(data_asset, 'E113:E132');
asset_value_past20=str2num(cell2mat(asset_value_past20));   %change
the format from cell to char and finally to num
[~,matlabDateVec_past20]=xlsread(data_asset,'A113:A132');
[year_,month_,day_]=datevec(matlabDateVec_past20);
dateNumber_past20 = datenum(year_, month_, day_);

logr_past20=[];
for i=1:19
    logr_past20(i)=log(asset_value_past20(i))-
log(asset_value_past20(i+1));
end

vol_past20_annual=std(logr_past20)*sqrt(252);

for i=1:26
    bs_price(i)=blsprice(S0,strike(i),r0,(maturity_date(i)-
datenum(2017,12,29))/365,vol_past20_annual);
end

% A graph of Black-Scholes Price and Real Market Price
figure(3)
for j=[1,2,3,6]
    cnt=1;
    strike_tmp=[];  %remember to clear
    bs_tmp=[];
    close_tmp=[];
    for i=1:26
        if j~=6
            subplot(2,2,j)
        else
            subplot(2,2,4)
```

```
        end
        if maturities(i)==j
            strike_tmp(cnt)=strike(i);
            bs_tmp(cnt)=bs_price(i);
            close_tmp(cnt)=close_price(i);
            cnt=cnt+1;
        end
    end
    plot(strike_tmp,bs_tmp);
    hold on
    plot(strike_tmp,close_tmp);
    xlabel([num2str(j),'m']);
    ylabel('price');
    legend('BS price','Market Price');
    hold off
end
print('BS model result 2017.12.29','-djpeg');

%calculate the mean pricing error and std error
for i=1:26
    ERR_BS(i)=bs_price(i)-close_price(i);
end

ME_BS=mean(abs(ERR_BS));
std_BS=std(ERR_BS);

%-------------------------------------------------------------------------
% 2.2 Option Pricing with Heston Model (Using Fourier pricing method)
%-------------------------------------------------------------------------
%Heston Calibration Driver
%
% starting point for unknown coefficient x0
x0=[6.5482  0.0731  2.3012  -0.4176  0.1838];
%   kap      th     sig     rho       vt
% Call LevenbergMarquardt solver build in function to solve for
% unknown x and residual norm.
disp('Heston Model Calibrating......');
[x,resnorm]=lsqnonlin(@HestonDiff,x0,[],[]);
fprintf('-----------------------------------------------\n');
fprintf('   kap     th     sig     rho     vt\n');
fprintf('%6.4f  %6.4f  %6.4f  %6.4f  %6.4f
\n',x(1),x(2),x(3),x(4),x(5));
fprintf('-----------------------------------------------\n');
% compute Call option using parameters calibrated as x
for i=1:26

heston_price(i)=HestonCall(S0,strike(i),r0,x(3),T(i),x(5),x(1),x(2),0
,x(4));
end
%HestonCall(St,K,r,sig,T,vt,kap,th,lda,rho)

figure(4)
for j=[1,2,3,6]
    cnt=1;
    strike_tmp=[];  %remember to clear the list
    heston_tmp=[];
    close_tmp=[];
    for i=1:26
        if j~=6
            subplot(2,2,j)
        else
            subplot(2,2,4)
        end
```

```
        if maturities(i)==j
            strike_tmp(cnt)=strike(i);
            heston_tmp(cnt)=heston_price(i);
            close_tmp(cnt)=close_price(i);
            cnt=cnt+1;
        end
    end
    plot(strike_tmp,heston_tmp);
    hold on
    plot(strike_tmp,close_tmp);
    xlabel([num2str(j),'m']);
    ylabel('price');
    legend('Heston price','Market Price');
    hold off
end
printf('Heston model result 2017.12.29','-djpeg');

%calculate the mean pricing error and std error
for i=1:26
    ERR_HES(i)=heston_price(i)-close_price(i);
end

ME_HES=mean(abs(ERR_HES));
std_HES=std(ERR_HES);

fprintf('-------------------------------------\n');
fprintf('         Mean ERR         std ERR         \n');
fprintf('BS     %8.8f      %8.8f    \n',ME_BS,std_BS);
fprintf('Heston   %6.8f      %8.8f    \n',ME_HES,std_HES);
fprintf('-------------------------------------\n');
%Seems that Heston model fits the data better?
%So, next, use the next trading day data to test the robustness of
the
%parameters.

%-------------------------------------------------------------------------
% 3. Robustness Test (with next trading day(2018.1.2))
% With the totally same method as before
%-------------------------------------------------------------------------
data_imp2='/Users/leichunyan/Downloads/2018.1.2.xlsx';
maturities2 = xlsread(data_imp2, 'G2:G37');
strike2=xlsread(data_imp2,'D2:D37');
close_price2=xlsread(data_imp2,'O2:O37');
S2=2.9070;
%sort by stirke in ascending order
[n2,m2]=size(maturities2);
for s=1:n2
    for k=1:(n2-s)
        if strike2(k)>strike2(k+1)
          tmp1=strike2(k);
          strike2(k)=strike2(k+1);
          strike2(k+1)=tmp1;
          tmp2=maturities2(k);
          maturities2(k)=maturities2(k+1);
          maturities2(k+1)=tmp2;
          tmp3=close_price2(k);
          close_price2(k)=close_price2(k+1);
          close_price2(k+1)=tmp3;
        end
    end
end

% set the maturity date for each option
```

```
for i=1:36     % the 4th Wednesday if it is not legel holiday
    if maturities2(i)==1  % maturity date is 2018.1.24
        maturity_date2(i)=datenum(2018,1,24);
    elseif maturities2(i)==2  %2018.2.28
        maturity_date2(i)=datenum(2018,2,28);
    elseif maturities2(i)==3  %2018.3.28
        maturity_date2(i)=datenum(2018,3,28);
    else
        maturity_date2(i)=datenum(2018,6,27);
    end
end
r2=0.036767;
T2=((maturity_date2-datenum(2017,12,29))/365)';
heston_price2=[];
for i=1:36

heston_price2(i)=HestonCall(S2,strike2(i),r2,x(3),T2(i),x(5),x(1),x(2
),0,x(4));
end

figure(5)
for j=[1,2,3,6]
    cnt=1;
    strike_tmp=[];
    heston_tmp=[];
    close_tmp=[];
    for i=1:36
        if j~=6
            subplot(2,2,j)
        else
            subplot(2,2,4)
        end
        if maturities2(i)==j
            strike_tmp(cnt)=strike2(i);
            heston_tmp(cnt)=heston_price2(i);
            close_tmp(cnt)=close_price2(i);
            cnt=cnt+1;
        end
    end
    plot(strike_tmp,heston_tmp);
    hold on
    plot(strike_tmp,close_tmp);
    xlabel([num2str(j),'m']);
    ylabel('price');
    legend('Heston price','Market Price');
    hold off
end
print('Heston model 2018.1.2(using 2017.12.29 parameters)','-djpeg');

for i=1:36
    ERR_HES2(i)=heston_price2(i)-close_price2(i);
end

ME_HES2=mean(abs(ERR_HES2));
std_HES2=std(ERR_HES2);

%However, using BSM again
%--------------------------------------------------------------------
[~,~,asset_value_past20_2] = xlsread(data_asset, 'E111:E130');
asset_value_past20_2=str2num(cell2mat(asset_value_past20_2));   %chan
ge the format from cell to char and finally to num
[~,matlabDateVec_past20_2]=xlsread(data_asset,'A111:A130');
[year_2,month_2,day_2]=datevec(matlabDateVec_past20_2);
```

```
dateNumber_past20_2 = datenum(year_2, month_2, day_2);

logr_past20_2=[];
for i=1:19
    logr_past20_2(i)=log(asset_value_past20_2(i))-
log(asset_value_past20_2(i+1));
end

vol_past20_annual_2=std(logr_past20_2)*sqrt(252);

for i=1:36

bs_price_2(i)=blsprice(S2,strike2(i),r2,T2(i),vol_past20_annual_2);
end

% A graph of Black-Scholes Price and Real Market Price
figure(6)
for j=[1,2,3,6]
    cnt=1;
    strike_tmp=[];
    bs_tmp=[];
    close_tmp=[];
    for i=1:36
        if j~=6
            subplot(2,2,j)
        else
            subplot(2,2,4)
        end
        if maturities2(i)==j
            strike_tmp(cnt)=strike2(i);
            bs_tmp(cnt)=bs_price_2(i);
            close_tmp(cnt)=close_price2(i);
            cnt=cnt+1;
        end
    end
    plot(strike_tmp,bs_tmp);
    hold on
    plot(strike_tmp,close_tmp);
    xlabel([num2str(j),'m']);
    ylabel('price');
    legend('BS price','Market Price');
    hold off
end
print('BS model result 2018.1.2','-djpeg');

for i=1:36
    ERR_BS2(i)=bs_price_2(i)-close_price2(i);
end

ME_BS2=mean(abs(ERR_BS2));
std_BS2=std(ERR_BS2);

fprintf('-------------------------------------\n');
fprintf('         Mean ERR         std ERR         \n');
fprintf('BS      %8.8f      %8.8f    \n',ME_BS2,std_BS2);
fprintf('Heston   %6.8f      %8.8f    \n',ME_HES2,std_HES2);
fprintf('-------------------------------------\n');
```