MAP 5611 Intro to Computational Finance HW6

Student: Sen Zhang

I. Executive Summary

Firstly, the Crank-Nicholson approximation is tested to be correct. The approximation of the test PDE has at least 6 significant digits. And we verify the error is $O\left(\Delta x^2, \Delta \tau^2\right)$.

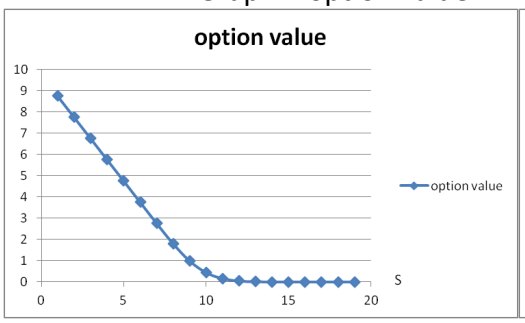
Secondly, we calculate the value of a European put with a six month expiry. The graph is as graph 1.

And the option value is 1.796453 at S^*=8.

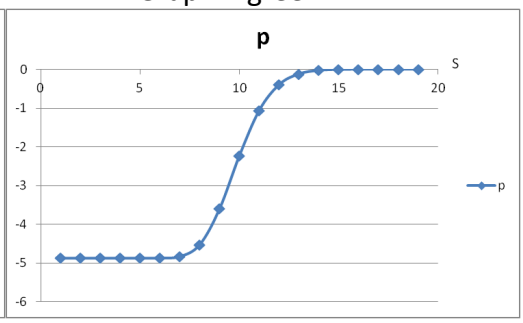
the value of the delta is $\Delta = -0.907656918$ for S^*=8.

At last, we find the option value is very sensitive to interest rate when the stock price is less than the strike. The graph of the greek is as graph 2.

Graph 1 option value



Graph 2 greek



II. Statement of Problem

1. To test the Crank-Nicolson approximation

$$\delta_\tau^+ u_j^n = \frac{u_j^{n+1} - u_j^n}{\Delta \tau} = \left\{ \frac{1}{R} \sigma_j^2 x_j^2 \delta_x^+ \delta_x^- + r_j x_j \delta_x^0 - r_j \right\} \frac{\left[u_j^{n+1} + u_j^n \right]}{2}$$

by solving a problem for which we know the answer, the easiest way is to add a forcing term $\left(1 - x - \frac{x^2}{R}\right) e^x$ to the RHS of its tri-diagonal system, and choose $\sigma = r = 1$, the exact solution is $v(x, \tau) = e^x + e^{-\tau}$.

Substitute $\frac{\partial v}{\partial \tau} = -e^{-\tau}$; $\frac{\partial v}{\partial x} = e^{x}$; $\frac{\partial^2 v}{\partial x^2} = e^{x}$ into $\frac{\partial v}{\partial \tau} - rx\frac{\partial v}{\partial x} = \frac{1}{R}\sigma^2 x^2 \frac{\partial^2 v}{\partial x^2} - rv$

$$\Rightarrow \begin{Bmatrix} -e^{-\tau} - xe^{x} = \frac{1}{R}x^2 e^{x} - e^{x} - e^{-\tau} \\ -xe^{x} = \frac{1}{R}x^2 e^{x} - e^{x} \end{Bmatrix}$$

Then $v(x,\tau) = e^{x} + e^{-\tau}$ becomes an exact solution $e^{x}\left(1 - x - \frac{1}{R}\right) = 0$

Integrate to time $\tau = 1$ and compare with the exact solution for a sequence of space and time step sizes. Verify that the error varies as $O(\Delta x^2, \Delta \tau^2)$ for the Crank-Nicolson approximation.

2. Compute the values of the European put for $E^* = \$10, r^* = 0.05/yr, \sigma^* = 0.20/yr$, and $T^* = 0.5yr$ Assume the interest rate and volatility are constant. The only purpose here is to nondimensionalize the equations.

- Report the values of the option for S^* in $[0,20]$. Compute and report the price accurate to the nearest cent.
- Report the value of the option and the delta, $\Delta = \partial V^* / \partial S^*$ for $S^* = 8$. Also plot the Δ for $S^* \in [0,20]$.
- (Extra Credit) How sensitive is the value of the option to the interest rate? Report the value of the greek $\rho = \partial V^* / \partial r^*$.

III. Description of The Mathematics

The initial and boundary conditions of the European put is:

$$\frac{\partial V^*}{\partial t^*} + \frac{1}{2}(\sigma^*)^2 (S^*)^2 \frac{\partial^2 V^*}{\partial (x^*)^2} + r^* S^* \frac{\partial V^*}{\partial S^*} = r^* V^*$$
$$V^*(0, t^*) = K^* e^{-r^*(T^* - t^*)}$$
$$V^*(S^*, T^*) = \max(E^* - S^*, 0)$$
$$V^*(S^*, t^*) = 0 \quad S^* \to \infty$$

Scale this by

$$x = S^* / E^*$$
$$U = V^* / E^*$$
$$\tau = r_0^*(T^* - t^*)$$
$$\sigma = \sigma^* / \sigma_0^*$$
$$r = r^* / r_0^*$$
$$\frac{1}{R} = \frac{1}{2}\frac{(\sigma_0^*)^2}{r_0^*}$$

So the scaled IBVP is:

$$\frac{\partial U}{\partial \tau} - rx\frac{\partial U}{\partial x} = \frac{1}{2}\sigma^2 x^2 \frac{\partial^2 U}{\partial x^2} - rU$$
$$U(x,0) = \max(1-x,0)$$
$$U(0,\tau) = e^{-r\tau}$$
$$U(x,\tau) = 0 \qquad as\ S \to \infty$$

Use Crank-Nicolson approximation to get the numerical solution to the problem.

Since we have:

$$\frac{\partial^2 U}{\partial x^2} = \delta_x^+ \delta_x^- x + O(\Delta x^2)$$
$$\frac{\partial U}{\partial x} = \delta_x^0 x + O(\Delta x^2)$$

Then we should get the following formulas:

$$\begin{cases} \delta_x^+ \delta_x^- u_j = \dfrac{u_{j+1} - 2u_j + u_{j-1}}{\Delta x^2} + O(\Delta x^2) \\ \delta_x^0 u_j = \dfrac{u_{j+1} - u_{j-1}}{2\Delta x} + O(\Delta x^2) \end{cases}$$

Use Trapezoidal Rule to integrate τ from 0 to T:

$$\delta_\tau^+ u_j^n = \frac{u_j^{n+1} - u_j^n}{\Delta \tau} = \{\frac{\sigma_j^2 x_j^2}{2}\delta_x^+ \delta_x^- + r_j x_j \delta_x^0 - r_j\}\frac{[u_j^{n+1} + u_j^n]}{2}$$

Then we can write the whole problem in tri-diagonal system:

$$M_1 U^{n+1} = M_2 U^n + \Delta \tau g$$

$$U^n = \begin{bmatrix} u_1 \\ u_2 \\ . \\ . \\ . \\ u_{N_x-1} \end{bmatrix}^n , \; g = \begin{bmatrix} a_1\left(\frac{u_0^n + u_0^{n+1}}{2}\right) \\ 0 \\ . \\ . \\ . \\ 0 \\ c_{N_x-1}\left(\frac{u_\infty^n + u_\infty^{n+1}}{2}\right) \end{bmatrix}$$

$$\begin{cases} M_1 = diag(\frac{-\Delta\tau}{2}a, 1-\frac{\Delta\tau}{2}b, \frac{-\Delta\tau}{2}c) \\ M_2 = diag(\frac{\Delta\tau}{2}a, 1+\frac{\Delta\tau}{2}b, \frac{\Delta\tau}{2}c) \\ a_j = \frac{\sigma_j^2 x_j^2}{2\Delta x^2} - \frac{r_j x_j}{2\Delta x} \\ b_j = \frac{-\sigma_j^2 x_j^2}{2\Delta x^2} - r_j \\ c_j = \frac{\sigma_j^2 x_j^2}{2\Delta x^2} + \frac{r_j x_j}{2\Delta x} \end{cases}$$

1.

Test

Implement the numerical approximation by following test PDE:

$$\frac{\partial U}{\partial \tau} - rx\frac{\partial U}{\partial x} = \frac{1}{2}\sigma^2 x^2 \frac{\partial^2 U}{\partial x^2} - rU + (1 - x - \frac{x^2}{2})e^x$$
$$u(x,0) = e^x + 1$$
$$u(0,\tau) = 1 + e^{-\tau}$$
$$u(1,\tau) = e^1 + e^{-\tau}$$

The exact value for this PDE is

$$u(x,\tau) = e^x + e^{-\tau}$$

2.

Option Value using CN approximation

$$U - \tilde{U}_{\Delta x,\Delta t} = O(\Delta x^2, \Delta t^2)$$

$$U - \tilde{U}_{\frac{\Delta x}{2},\frac{\Delta t}{2}} = O(\frac{\Delta x^2}{4}, \frac{\Delta t^2}{4})$$

$$\Rightarrow E_{\Delta x,\Delta t} = O(\Delta x^2, \Delta t^2) = \frac{-2^2}{2^2 - 1}(\tilde{U}_{\Delta x,\Delta t} - \tilde{U}_{\frac{\Delta x}{2},\frac{\Delta t}{2}})$$

We have

$$U = \tilde{U}_{\Delta x,\Delta t} + E_{\Delta x,\Delta t}$$

Delta

In order to calculate the value of the delta, we need formulas:

$$\Delta = \frac{\partial U}{\partial S} \approx \frac{U_{i+1} - U_{i-1}}{2\Delta S}$$

Greek

(Extra Credit) In order to compute the greek $\rho = \partial V^* / \partial r^*$, we need to calculate the Δ :

$$\rho = \frac{\partial U}{\partial r} \approx \frac{U(r - \Delta r) - U(r + \Delta r)}{2\Delta r}$$

IV. Description of The Algorithm

Start with payoff

$V_j = payoff\left(S_j\right), \quad j = 0,1,2,...N_x$

$for \quad n = 0 \quad to \quad N_\tau - 1$

$construct \quad M = \begin{bmatrix} b & c & 0 \\ a & \ddots & c \\ 0 & a & b \end{bmatrix}$

$construct \quad V$

Using Thomas algorithm to $solve \quad MU^{n+1} = V \quad gives \quad U^n$

$next \quad n$

$output \quad results$

V. Results

1. Test

Set

$x \in [0,1], t \in [0,1], \sigma = 1 = r, R = 2$

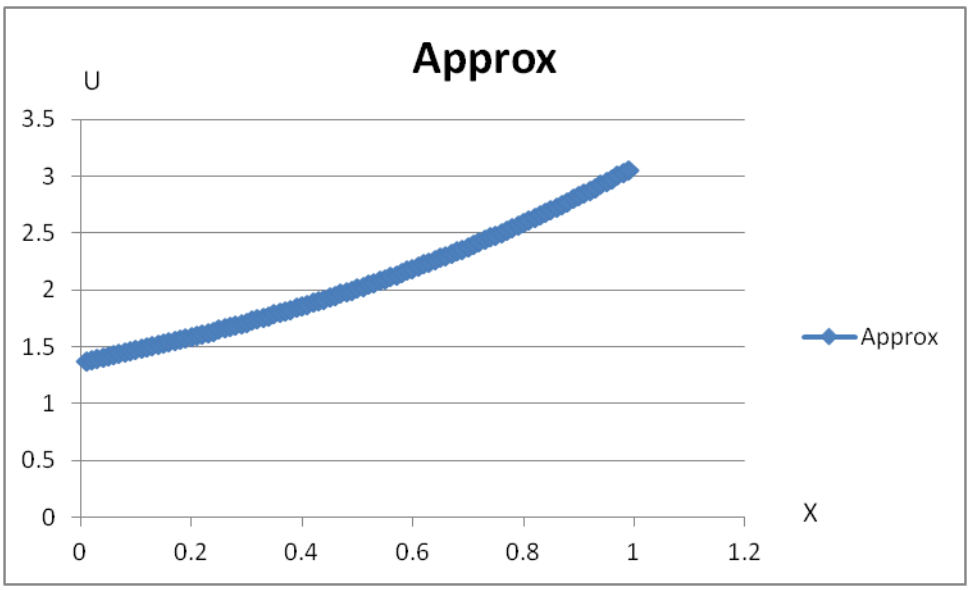
Nstep=100, Ntime=100

Part of the data is as following:

Table 1 Approximation of test PDE

x	Approximation	Abs error	True value
0.1	1.47305	8.96E-07	1.47305
0.2	1.589284	1.35E-06	1.589282
0.3	1.717741	3.13E-06	1.717738
0.4	1.859709	4.35E-06	1.859704
0.5	2.016606	5.01E-06	2.016601
0.6	2.190003	5.12E-06	2.189998
0.7	2.381637	4.67E-06	2.381632
0.8	2.593424	3.67E-06	2.59342
0.9	2.827485	2.12E-06	2.827483

The graph is as following:



Verify the error

Choose

$\Delta x = 0.1, \Delta t = 0.1$

Compute

$E_{\frac{\Delta x}{2},\frac{\Delta t}{2}}$, $E_{\frac{\Delta x}{10},\frac{\Delta t}{10}}$

X	$E_{\Delta x=0.1,\Delta t=0.1}$	$E_{\frac{\Delta x}{2},\frac{\Delta t}{2}}$	$E_{\Delta x=0.1,\Delta t=0.1} / E_{\frac{\Delta x}{2},\frac{\Delta t}{2}}$	$E_{\frac{\Delta x}{10},\frac{\Delta t}{10}}$	$E_{\Delta x=0.1,\Delta t=0.1} / E_{\frac{\Delta x}{10},\frac{\Delta t}{10}}$
0.1	9.44E-05	2.27E-05	**4.16045**	8.96E-07	**105.2522**
0.2	0.000132	3.36E-05	**3.942233**	1.35E-06	**98.1207**
0.3	0.000312	7.82E-05	**3.988623**	3.13E-06	**99.62712**
0.4	0.000435	0.000109	**3.996353**	4.35E-06	**99.88025**
0.5	0.000501	0.000125	**3.998531**	5.01E-06	**99.95142**
0.6	0.000512	0.000128	**3.999089**	5.12E-06	**99.97015**
0.7	0.000467	0.000117	**3.998713**	4.67E-06	**99.96193**
0.8	0.000368	9.19E-05	**4.00148**	3.67E-06	**100.0277**

0.9	0.000212	5.31E-05	**3.992855**	2.12E-06	**99.8443**

From this table we have

$$\frac{E_{\Delta x,\Delta t}}{E_{\frac{\Delta x}{2},\frac{\Delta t}{2}}} \approx 4, \frac{E_{\Delta x,\Delta t}}{E_{\frac{\Delta x}{10},\frac{\Delta t}{10}}} \approx 100$$

Then we can conclude that the error should be

$$O(\Delta x^2, \Delta t^2)$$

2.

a). Option Value

For

$$E^* = 10, r^* = \frac{0.05}{year}, \sigma^* = \frac{0.2}{year}, \tau \in [0,0.5], S^* \in [0,20]$$

Set

$$x \in [0,2], t \in [0,0.025], \sigma = 1 = r, R = 2.5, Nstep = 200, Ntime = 200$$

$$U - \tilde{U}_{\Delta x,\Delta t} = O(\Delta x^2, \Delta t^2)$$

$$U - \tilde{U}_{\frac{\Delta x}{2},\frac{\Delta t}{2}} = O(\frac{\Delta x^2}{4}, \frac{\Delta t^2}{4})$$

$$\Rightarrow E_{\Delta x,\Delta t} = O(\Delta x^2, \Delta t^2) = \frac{-2^2}{2^2 - 1}(\tilde{U}_{\Delta x,\Delta t} - \tilde{U}_{\frac{\Delta x}{2},\frac{\Delta t}{2}})$$

We have

$$U = \tilde{U}_{\Delta x,\Delta t} + E_{\Delta x,\Delta t}$$

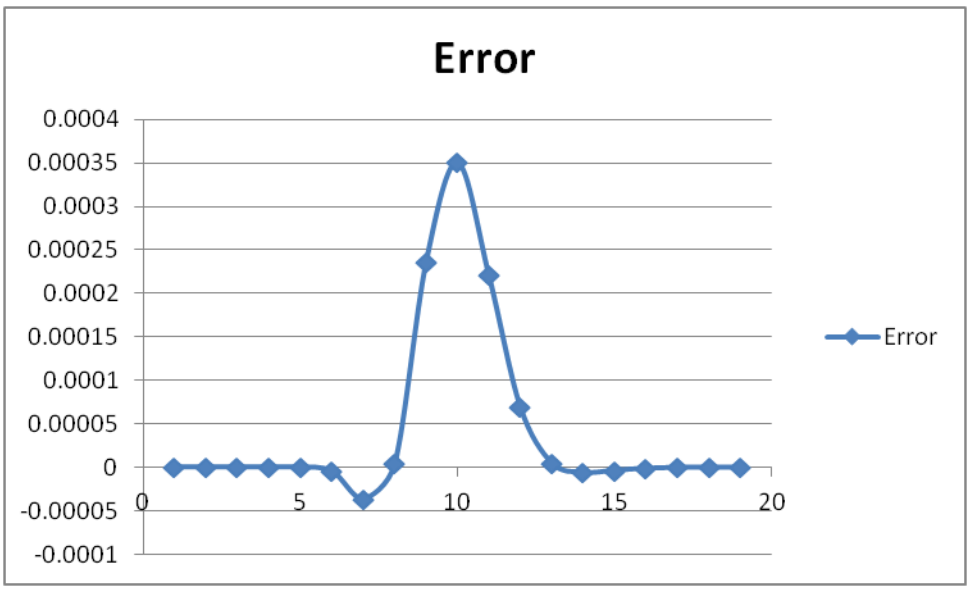
The data is as following:

Table 2 Option Value

S^*	$\tilde{U}_{\Delta x=0.1,\Delta t=0.1}$	$\tilde{U}_{\frac{\Delta x}{2},\frac{\Delta t}{2}}$	$E_{\Delta x=0.1,\Delta t=0.1}$	$U_{\Delta x,\Delta t} + E_{\Delta x,\Delta t}$

1	8.750661	8.750661	3.27E-10	**8.750661**
2	7.750661	7.750661	3.27E-10	**7.750661**
3	6.750661	6.750661	3.27E-10	**6.750661**
4	5.750661	5.750661	2.96E-10	**5.750661**
5	4.750661	4.750661	-6.1E-08	**4.750661**
6	3.750748	3.750744	-4.8E-06	**3.750738**
7	2.754453	2.754426	-3.7E-05	**2.75438**
8	1.796445	1.796448	3.82E-06	**1.796453**
9	0.986005	0.986183	0.000237	**0.986479**
10	0.440506	0.440769	0.000351	**0.441208**
11	0.159886	0.160052	0.000221	**0.160328**
12	0.048077	0.048129	6.93E-05	**0.048215**
13	0.012316	0.012319	4.89E-06	**0.012325**
14	0.002765	0.002761	-6.1E-06	**0.002753**
15	0.000558	0.000556	-3.7E-06	**0.000551**
16	0.000104	0.000103	-1.4E-06	**0.000101**
17	1.8E-05	1.77E-05	-3.9E-07	**1.72E-05**
18	2.97E-06	2.9E-06	-9.7E-08	**2.78E-06**
19	4.61E-07	4.46E-07	-2.1E-08	**4.2E-07**

Look at the error :

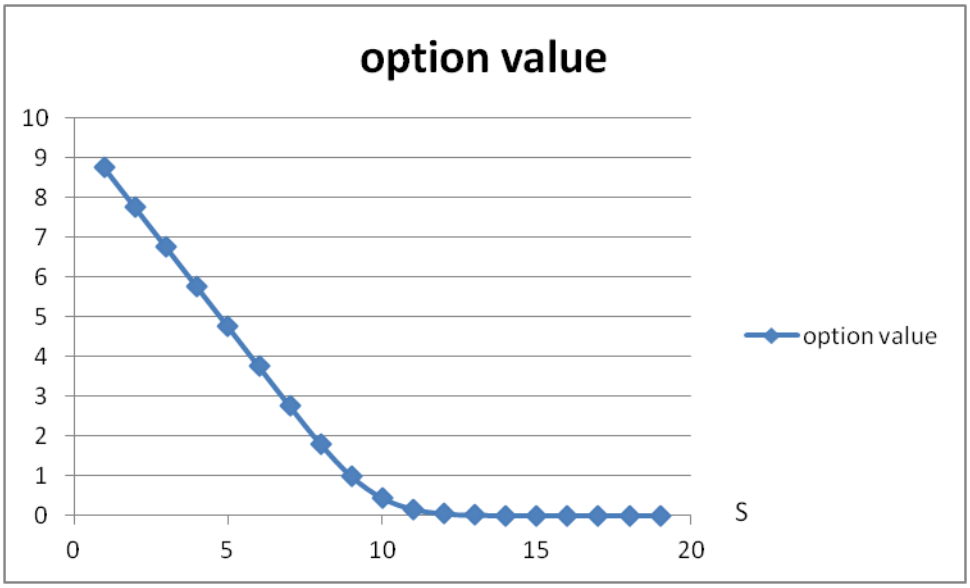


From the graph of error, We can find the biggest error is around 0.0004.

Thus we can conclude the option value accurate to at least cent.

And the graph of option value is as following:

Graph 1 Option Value



b). Delta

Approximate delta using

$$\Delta = \frac{\partial U}{\partial S} \approx \frac{U_{i+1} - U_{i-1}}{2\Delta S}$$

Set

$Nstep = 80, Ntime = 200$

Then we have

Option value U(S^*=8)=1.796453

$\Delta = -0.907656918$ for S^*=8

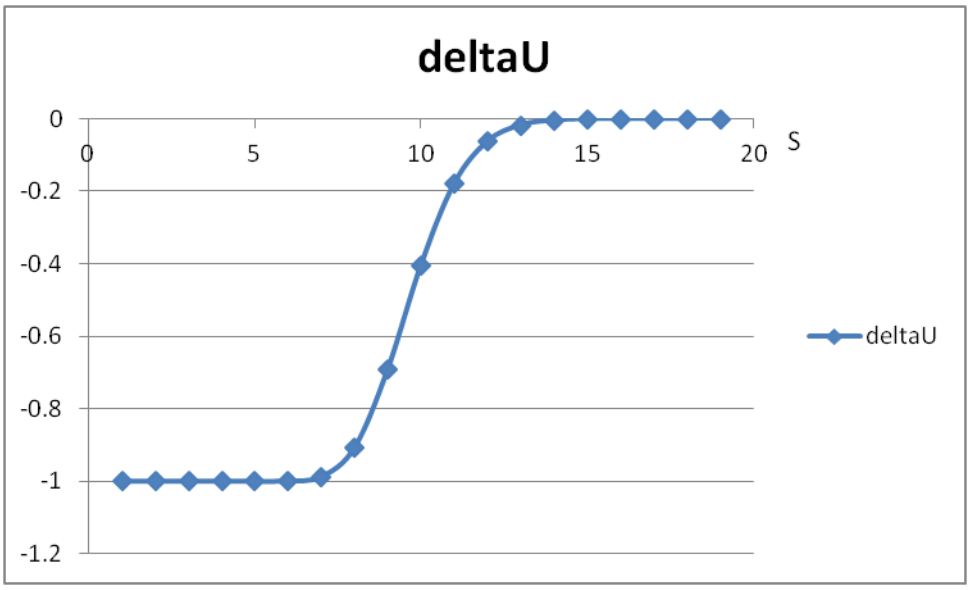
And the data is as following:

S^*	deltaU
1	-1.000000000
2	-1.000000000
3	-1.000000000
4	-0.999999991
5	-0.999994495
6	-0.999457215
7	-0.987649675
8	-0.907656918
9	-0.691277958
10	-0.402848688
11	-0.178492108
12	-0.062248552
13	-0.017870494
14	-0.004402731
15	-0.000963803
16	-0.000192773

17	-0.000036013
18	-0.000006398
19	-0.000001125

The graph of Delta is as following:

Graph 2 Delta



c). Greek

Approximate the greek

$$Nstep = 80, Ntime = 200 \quad \rho = \frac{\partial U}{\partial r} \approx \frac{U(r-\Delta r) - U(r+\Delta r)}{2\Delta r}$$

Set r=0.99, r=1.01

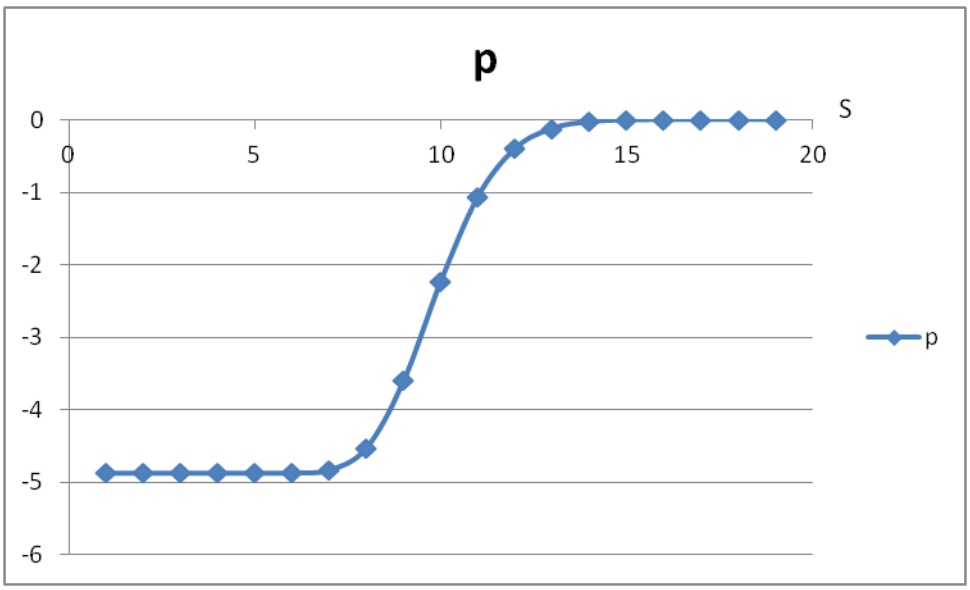
The data is as following:

S^*	p
1	-4.87654962356
2	-4.87654961524
3	-4.87654961532
4	-4.87654960139

5	-4.87653728215
6	-4.87502798571
7	-4.83569467869
8	-4.53039177478
9	-3.60329791578
10	-2.23308935376
11	-1.06119608012
12	-0.39786377368
13	-0.12270314067
14	-0.03240688205
15	-0.00758457439
16	-0.00161721643
17	-0.00032115252
18	-0.00006040808
19	-0.00001064485

The graph of greek is as following:

Graph 3 Greek



From graph 3 we can see the option value is very sensitive to interest rate when the stock price is less than the strike.

VI. Conclusions

1.

The approximation of the test PDE has at least 6 significant digits. And we verify the error is $O(\Delta x^2, \Delta \tau^2)$. Thus we can conclude the implement is correct.

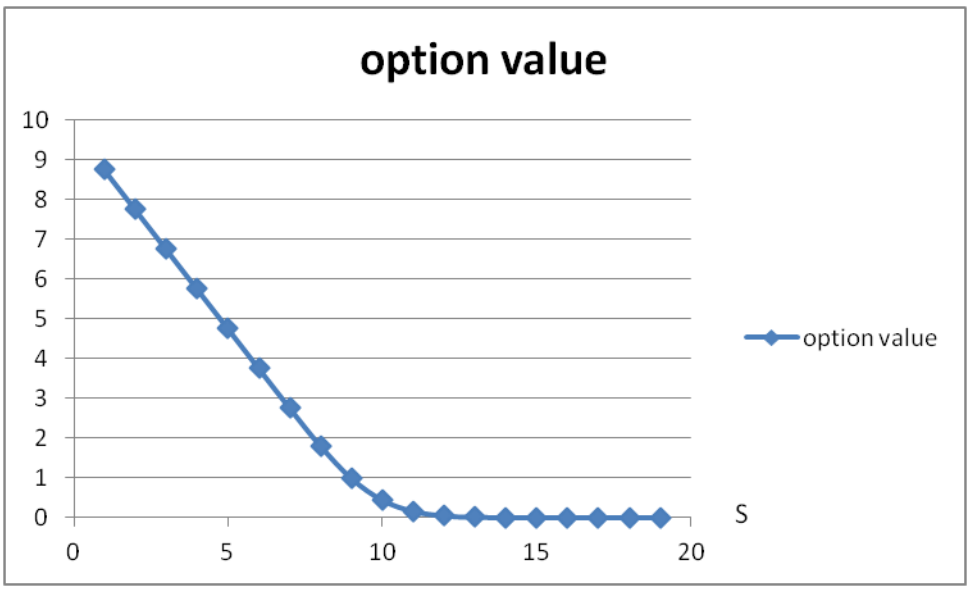
2.

a).

We have the biggest error is around 0.0004.

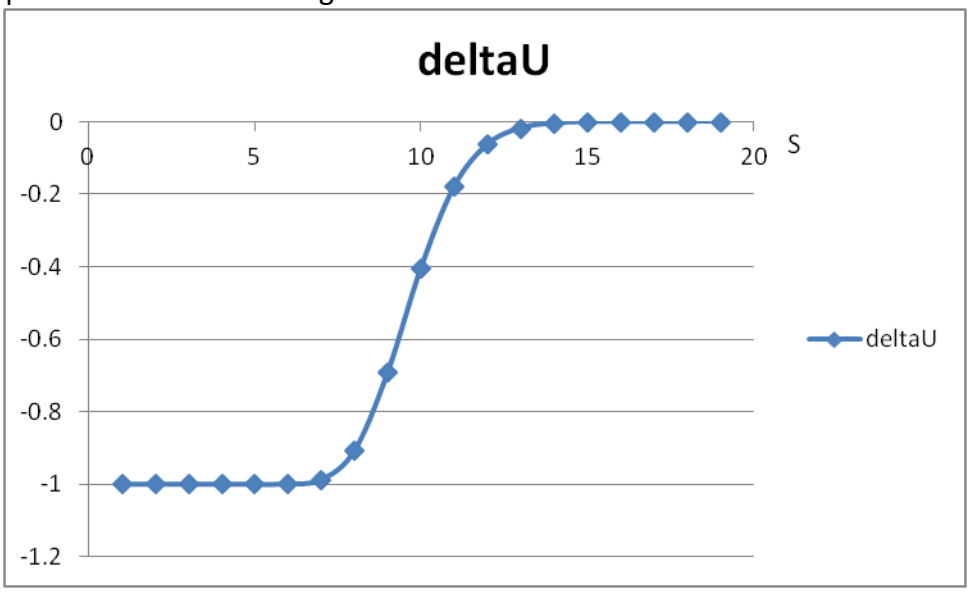
Thus we can conclude the option value accurate to at least cent.

And the graph of option value is as following:

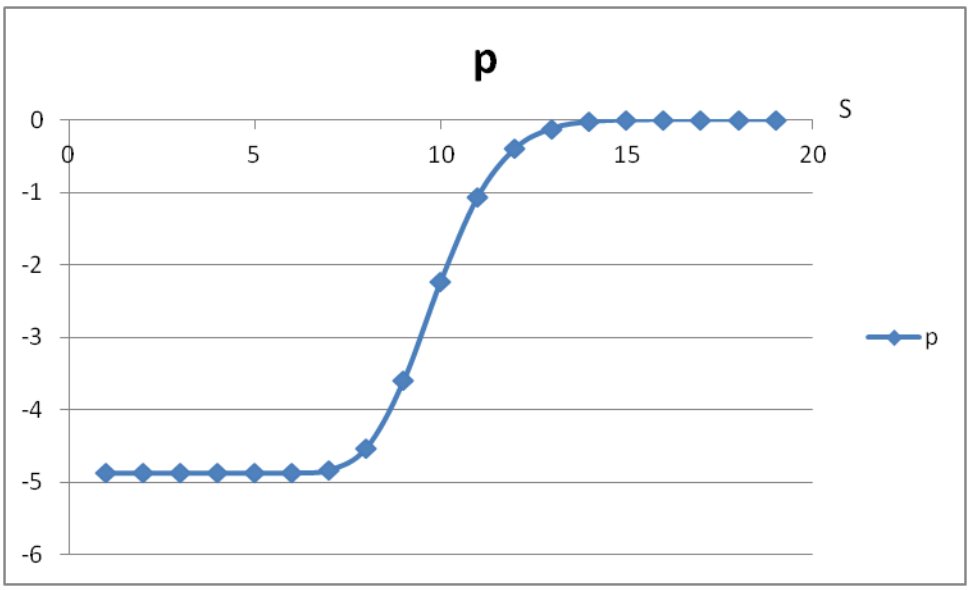


b).
we have
Option value $U(S^*=8)=1.796453$

$\Delta = -0.907656918$ for $S^*=8$
The graph of delta is as following:



c).
The graph of greek is as following:



We can conclude the option value is very sensitive to interest rate when the stock price is less than the strike.

VII. Program Listing

CrankN.h

```
#define FM double

#pragma once

class BlackS

{public:

    FM *BS(FM U[], FM X, FM a[], FM b[], FM c[], FM deltaT, FM R, int j);

};
```

CrankN.cpp

```
#include <iostream>

#include <fstream>

#include <iomanip>

#include "math.h"

#include "CrankN.h"

using namespace std;
```

```
const int Nstep = 400;

const int Ntime = 400;


FM*  BlackS::BS(FM U[], FM X, FM a[], FM b[], FM c[], FM deltaT, FM R, int j)

{

        FM delta = X/Nstep;

        FM V[Nstep+1], aL[Nstep+1], bL[Nstep+1], cL[Nstep+1], aR[Nstep+1], bR[Nstep+1],
cR[Nstep+1];


        for (int i=1; i<Nstep; i++)

        {

                aL[i] = -deltaT*a[i]/2;

                bL[i] = 1-deltaT*b[i]/2;

                cL[i] = -deltaT*c[i]/2;

                aR[i] = deltaT*a[i]/2;

                bR[i] = 1+deltaT*b[i]/2;

                cR[i] = deltaT*c[i]/2;

        }


        FM U0[Ntime+1],UX[Ntime+1];


        for(int i=0;i<=Ntime;i++)

        {

                U0[i] = exp(-deltaT*i);

    UX[i] = 0;
```

```
            //U0[i] = 1.0 + exp(-deltaT*i);

            //UX[i] = exp(1.0) + exp(-deltaT*i);

        }


        //RHS

        for (int i=2;i<=Nstep-2; i++)

        {

           V[i] = aR[i]*U[i-1] + bR[i]*U[i] + cR[i]*U[i+1];// + deltaT*(1.0 - delta*i -
pow(delta*i,2)/R)*exp(delta*i) ;

        }

  V[1] = bR[1]*U[1] + cR[1]*U[2]

          + deltaT*a[1]*(U0[j]+U0[j+1])/2 ;

          // + deltaT*(1-delta-delta*delta/R)*exp(delta) ;

  V[Nstep-1] = aR[Nstep-1]*U[Nstep-2] + bR[Nstep-1]*U[Nstep-1]

                    + deltaT*c[Nstep-1]*(UX[j]+UX[j+1])/2;

                        //  + deltaT*(1-delta*(Nstep-1)-pow(delta*(Nstep-
1),2)/R)*exp(delta*(Nstep-1)) ;


        //Thomas Algorithm

        for (int i=2; i<Nstep; i++)

        {

            bL[i] = bL[i] - aL[i]/bL[i-1]*cL[i-1];

            V[i] = V[i] - aL[i]/bL[i-1]*V[i-1];

        }

        U[Nstep-1] = V[Nstep-1]/bL[Nstep-1];

        for (int i=Nstep-2; i>0; i--)

        {
```

```
                U[i] = (V[i]-cL[i]*U[i+1])/bL[i];
        }

    FM *point_U;
        point_U = U;

        return point_U;
}

int main()
{
        BlackS bs;

        FM U[Nstep+1];
        FM a[Nstep+1], b[Nstep+1], c[Nstep+1];
        FM X=2.0, T=0.025, sigma=1.0, r=1.01, R=2.5;
        FM delta = X/Nstep;
        FM deltaT = T/Ntime;

        for (int i=0; i<=Nstep; i++)
        {
                //U[i] = 1.0+exp(i*delta);
                U[i] = 1.0 - i*delta;
                if (U[i]>0)
                {
                        U[i]=U[i];
```

```
                }
                else
                {
                        U[i] = 0;
                }
                a[i] = sigma*sigma*i*i/R-r*i/2;
                b[i] = -2*sigma*sigma*i*i/R-r;
                c[i] = sigma*sigma*i*i/R+r*i/2;
        }

        FM *point_U;
  for (int j=0; j<=Ntime-1;j++)
        {
                point_U = bs.BS(U,X, a, b, c, deltaT, R, j);
        }

        FM TU[Nstep+1], deltaU[Nstep+1];
        for (int i=1; i<=Nstep-1;i++)
        {
                U[i] = point_U[i];
                deltaU[i] = (U[i+1] - U[i-1])/(2*delta);

                //TU[i] = exp(i*delta)+exp(-1.0);
                cout<<setprecision(8);
                //cout<<"Approxiamtion "<<U[i]<<"  Error "<<abs(TU[i] - U[i])<<"  True Value
"<<TU[i]<<endl;
                cout<<"Approxiamtion "<<10.0*U[i]<<endl;
```

```
        }


        ofstream myFile;
   myFile.open( "app.txt");
   for (int i=0; i<=Nstep;i=i+20)
   {
                myFile<<setprecision(15)<<i*1<< "\t"  <<10.0*U[i]<<endl;//
deltaU[i]<<endl;//10.0*U[i]<<endl;// << "\t" << abs(TU[i] - U[i])<< "\t" <<TU[i]<< endl;
   }
   myFile.close();
        return 0;
}
```