Name: Fei Liu
UNI: fl2312
Tele: 9176557918

Report for HW6

PLEASE NOTE THAT:

I am using Matlab in this assignment rather than C++ as before. The README file shows how to test and the main script is named CrankNocolsonFDMain.m.

Part 1 Implementation of algorithm:

For a), the closed formula is exactly the same as in HW1. The Black-Scholes formula is as below:

$$C(S,t) = N(d_1)\ S - N(d_2)\ Ke^{-r(T-t)},$$
$$d_1 = \frac{\ln\left(\frac{S}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T-t}}$$
$$d_2 = \frac{\ln\left(\frac{S}{K}\right) + \left(r - \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T-t}} = d_1 - \sigma\sqrt{T-t}.$$

For b),

We care about the left hand matrix more than the right hand matrix because the right hand side only concerns with multiplication and this operation can be done rather fast. So, the LU Decomposition.m and LUSolver.m files are doing the algorithm to solve M*v=q system in general. Since I implement it with Matlab and Matlab is so good at Matrix manipulation, I didn't see much of a difficulty in implementing the algorithm indicated in the class slides. The most tricky park was that the note sometime marked the starting or terminating index wrong. However, because I can see the result right away after I set up a matrix, I can always figure out the right indexing very fast.

Another tricky part happened in the coding of boundary conditions. The type I boundary are not giving much influence on the call option of the spot price since they maximum spot is so far away from the current spot but if you look at the whole column of terminating option values, the impact on the close to boundary option values are quite dependent on which boundary condition you choose. I personally favor the type II boundary. It gives quite smoother result even if the barrier price is on a grid. In my code, I test if the barrier falls onto the grid and choose different boundary type accordingly. Also, I set up the type I boundary vector in a file called bounary1.m while directly implement the type II boundary in the CrankNocolsonFDMain.m Since, boundary type II ask for a boundary of the form

Name: Fei Liu
UNI: fl2312
Tele: 9176557918

a+b*V1. I actually choose a to be 0 and if I want type II boundary, I set b to be 1 and this will gives type 2 boundary, while I set b to be zero, which kill the boundary term and then add to it a constant vector for boundary type I situation. The function decomposition of the A's, B's, and C's are different only by a dividing a 0.5 to the explicit method that we did before and the formula is given below:

$$A_i^k = \frac{1}{2}\nu_1 a_i^k - \frac{1}{4}\nu_2 b_i^k,$$
$$B_i^k = -\nu_1 a_i^k + \frac{1}{2}\delta t\, c_i^k,$$

$$C_i^k = \frac{1}{2}\nu_1 a_i^k + \frac{1}{4}\nu_2 b_i^k.$$

Here is the matrix format of left hand matrix:

$$\mathrm{LHS} = \begin{bmatrix} 1-B_1^{k+1} & -C_1^{k+1} & 0 & \cdots & 0 \\ -A_2^{k+1} & 1-B_2^{k+1} & -C_2^{k+1} & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -A_{I-2}^{k+1} & 1-B_{I-2}^{k+1} & -C_{I-2}^{k+1} \\ 0 & \cdots & 0 & -A_{I-1}^{k+1} & 1-B_{I-1}^{k+1} \end{bmatrix} \begin{bmatrix} V_1^{k+1} \\ \vdots \\ V_{I-1}^{k+1} \end{bmatrix}$$
$$+ \begin{bmatrix} -A_1^{k+1}(a + bV_1^{k+1}) \\ 0 \\ \vdots \end{bmatrix}$$

The matrix notation of the above matrix is:

$$M_L^{k+1} V^{k+1} + R^k, \quad 1 \le k \le K.$$

The steps of LU decomposition are as described in the formula :

$$\begin{cases} Lw &= q, \\ Uv &= w. \end{cases}$$

Name: Fei Liu
UNI: fl2312
Tele: 9176557918

$$d_1 = 1 - B_1,$$
$$l_i d_{i-1} = -A_i,$$
$$u_{i-1} = -C_i,$$
$$d_i = 1 - B_i - l_i u_{i-1}, \quad 2 \le i \le I - 1$$

$$w_1 = q_1,$$
$$w_i = q_i - l_i w_{i-1}, \quad i = 2, 3, \ldots, I - 1.$$

For c), the closed formed barrier option formula is available from previous homework. I skip the description here.

For d), the barrier option of PDE requires clearing the entries above barrier every step because if one just brushes the first column above barrier to be 0, the following columns may still have non-zero values in the entries above barrier and this is not legitimate with the option definition. So, I check it every iteration. Except for that the remaining code is exactly the same as the vanilla option.

Part 2 Benchmark

(Note that I will put the analysis of change given different Nt, Ns, and Smax to next section)

I searched http://www.numerix.com/equity **thoroughly for option pricers but didn't find them any where. I wonder if that is not a freely available option. So, I stick to the excel benchmark which is good enough for this homework.**

	My Code for typical case	Haug's VBA Code
Closed-form European Style Vanilla call option price	9.0571	**9.0571**

Name: Fei Liu
UNI: fl2312
Tele: 9176557918

Crank Nicolson FD European Style vanilla call	8.6620	**8.389159**
The closed-form European-style Barrer call price	0.0524	**0.05077**
Explicit FD European-style Barrier call price	0.0526	**0.050769**

Part 3 Impact of change of parameters:

For Nt, I change it to 50. The corresponding values for explicit method change to 7.54657 * 10^17 and -1.43119* 10^8. Which is ridiculous.

For Ns, I change it to 150. The corresponding values for explicit method change to 1.88095* 10^225 and -1.3697* 10^151. Which is even more ridiculous.

For Smax, I change it to 400. The corresponding values for explicit method change to 7.29147 and 0.0537854. Which is not as ridiculous but is off.

The stability condition is

$$\Delta t \leq \frac{1}{\sigma^2 J^2}.$$

The right hand side is 0.0044444 if the Ns is 50. The time interval is 0.003968 in the case of 252 but not in the case of 50. So, the values go crazy. The right hand side Is 4.93827* 10^(-4) , which is obviously not satisfied and that's why we go crazy again. The safe way to go is just like what the professor told us to do.

I constructed a set of value with time steps 600 and space steps 80. This one satisfy the above condition. The values as 9.02481 and 0.415527. Not bad compared with the crazy case above. The accuracy can be Improved if the time steps go smaller.

Name: Fei Liu
UNI: fl2312
Tele: 9176557918

	My Code for Nt=2520	My Code for Ns=500	My code for Smax
Crank Nicolson FD European Style vanilla call	8.6620	8.6965	10.9913
Explicit FD European-style Barrier call price	0.0526	0.0656	0.0360

It appears that the Crank Nicolson method is good in terms of stability in space and time but not as good in the Smax. But it is much better than the explicit method.

As the method achieve unconditional second order stability in space and time, it is more favorable to use this method.