Case Study 2

Haoxiang Gao   UNI:hg2412

1. Compiling and Running Instructions

The program is dependent on GSL library, so you need to install GSL libs libgsl.so and libgslcblas.so under /home/username/lib.

To compile by command line:

g++ -L/home/username/lib -lgsl -lgslcblas case2. cpp -o case2

To run by command line:

./ case2.cpp

2. The difference equation

The derivation of difference equation of UOC option is almost the same as page 180 of *Computational Methods in Finance*. However, the payoff of UOC is 0 if $x < x_0$,

$$\int_{-\infty}^{x_0-x_i} (w(x_i+y,\tau_j)-w_{i,j})k(y)\,dy\,, \quad w(x_i+y,\tau_j)=0\,.$$

$$\int_{-\infty}^{x_0-x_i} (w(x_i+y,\tau_j)-w_{i,j})k(y)\,dy = \int_{-\infty}^{x_0-x_i} (0-w_{i,j})k(y)\,dy = \frac{\lambda_n^Y}{v} g_2(i\Delta x\lambda_n)$$

The difference equation is:

$$l_{i,j+1}w_{i-1,j+1} + d_{i,j+1}w_{i,j+1} + u_{i,j+1}w_{i+1,j+1} = w_{i,j} + \frac{\Delta\tau}{\nu}R_{i,j}$$

where

$$\begin{aligned} l_{i,j+1} &= -B_l \\ d_{i,j+1} &= 1 + r\Delta\tau + B_l + B_u + \frac{\Delta\tau}{\nu}\left(\lambda_n^Y g_2(i\Delta x\lambda_n) + \lambda_p^Y g_2((N-i)\Delta x\lambda_p)\right) \\ u_{i,j+1} &= -B_u \end{aligned}$$

$$\begin{aligned} R_{i,j} &= \sum_{k=1}^{i-1} \lambda_n^Y \left(w_{i-k,j} - w_{i,j} - k(w_{i-k-1,j} - w_{i-k,j})\right)\{g_2(k\Delta x\lambda_n) - g_2((k+1)\Delta x\lambda_n)\} \\ &+ \sum_{k=1}^{i-1} \frac{w_{i-k-1,j} - w_{i-k,j}}{\lambda_n^{1-Y}\Delta x}\left(g_1(k\Delta x\lambda_n) - g_1((k+1)\Delta x\lambda_n)\right) \\ &+ \sum_{k=1}^{N-i-1} \lambda_p^Y \left(w_{i+k,j} - w_{i,j} - k(w_{i+k+1,j} - w_{i+k,j})\right)\{g_2(k\Delta x\lambda_p) - g_2((k+1)\Delta x\lambda_p)\} \\ &+ \sum_{k=1}^{N-i-1} \frac{w_{i+k+1,j} - w_{i+k,j}}{\lambda_p^{1-Y}\Delta x}\left(g_1(k\Delta x\lambda_p) - g_1((k+1)\Delta x\lambda_p)\right) \end{aligned}$$

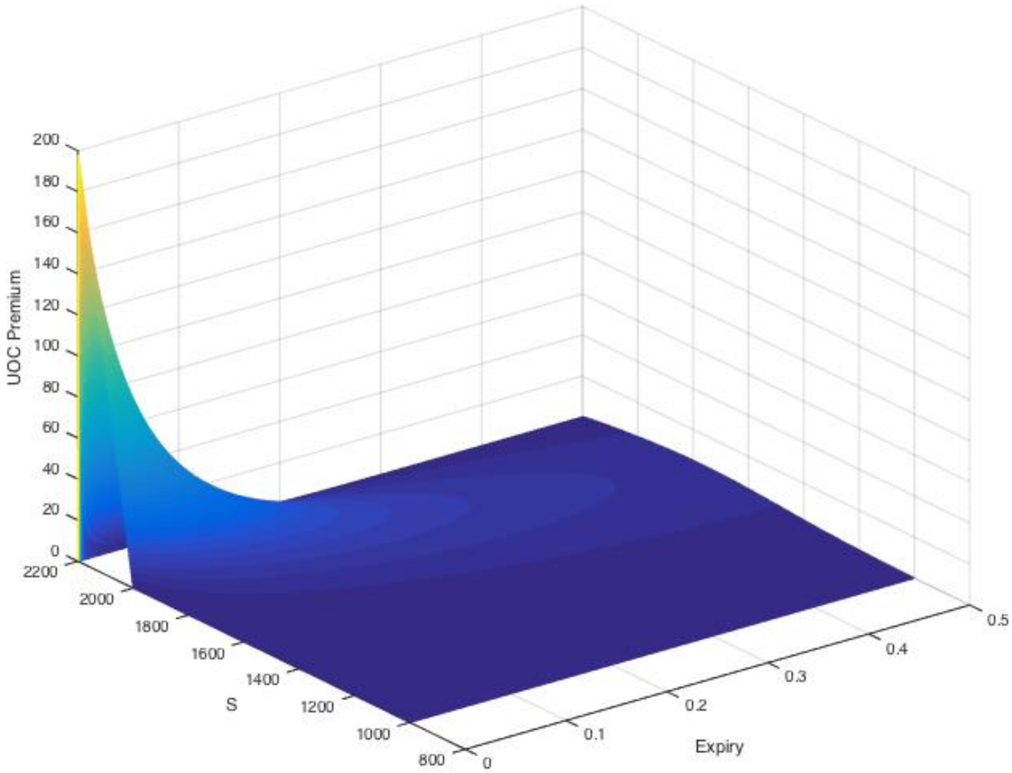
3. Experiment Results

I set $x_0 = \log(1000)$, because if x_0 is too small, the discretized x can not accurately capture the price within the moneyness. If x_0 is too large, the boundary condition that $w(x_0, \tau) = 0$ will be inaccurate.

I did six experiments, with M, N = 100, 200, 300, 400, 500, 1000. The result is as follow:

N (numer of steps for stock)	M (number of steps for time)	run time (seconds)	UOC premium
100	100	0.335398	4.23435
200	200	2.38734	4.41925
300	300	7.88509	4.43762
400	400	18.5511	4.47387
500	500	35.2927	4.4707
1000	1000	290.099	4.49638

Using Matlab, we can plot the surface of the UOC premium against stock price and expiry when setting M = 1000, N = 1000.



UOC Premium
200
180
160
140
120
100
80
60
40
20
0
2200
2000
1800
1600
1400
1200
1000
800
S
0
0.1
0.2
0.3
0.4
0.5
Expiry