MATH60082
Lab Class 8
American Options

Contents

8.1 Example - American Option with the Explicit Method

Tasks

8.1 Open the example code on my website CLICK HERE TO DOWNLOAD FULL CODE

```
1 /* Template code for the Explicit Finite Difference, as an American
      option
2  */
3 double explicitPutOption(double S0,double X,double T,double r,double
      sigma,int iMax,int jMax,double S_max)
4 {
```

Run the code and check it gives the required result.

8.2 We now want to make this an American option. Go to line 39 in the code where it assigns a value to the option:
CLICK HERE TO DOWNLOAD FULL CODE

```
1         vNew[j] = 1./(1.+r*dt)*(A*vOld[j+1]+B*vOld[j]+C*vOld[j-1]);
```

and change it to
`vNew[j] = max( X-S[j], 1./(1.+r*dt)*(A*vOld[j+1]+B*vOld[j]+C*vOld[j-1]) );`
run the code and check the value.

8.3 You will also need to edit the boundary conditions CLICK HERE TO DOWNLOAD FULL CODE

```
1    vNew[0] = X*exp(-r*(T-i*dt));
```

and change it to
`vNew[0] = X;`
run the code and check the values.

8.4 Run this code for different values of $iMax$, $jMax$ and S_max. Try to check the errors and convergence of the method. You could adapt $iMax$ to automatically be set according to the stability condition. I have an online calculator for American option where you can check the result – (click here).

8.2 Example - American Option with the PSOR Method

8.5 Open the example code on my website CLICK HERE TO DOWNLOAD FULL CODE

```
1  /*
2   * ON INPUT:
3   * S0            -- initial stock price
4   * X             -- exercise (strike) price
5   * T             -- Time to expiry (years)
6   * r             -- interest rate (per annum)
7   * sigma         -- volatility (per annum^12)
8   * iMax          -- number of time steps
9   * jMax          -- number of space steps
10  * SMax          -- Maximum value of S
11  * omega         -- is the relaxation parameter
12  * tol           -- is the tolerance level
13  * iterMax       -- is maximum iterations
14  * ON OUTPUT:
15  * return        -- the value of a European put option at S=S0, t=0
16  */
17 double europeanPut_CN(double S0,double X,double T,double r,double sigma,
       int iMax,int jMax,double SMax,double omega,double tol,int iterMax)
```

Run the code and check it gives the required result.

8.6 We now want to make this an American option. Go to lines 66, 73 and 79, where it generates a new guess y for to the value of the option:
CLICK HERE TO DOWNLOAD FULL CODE

```
1                   y = vNew[0] + omega*(y-vNew[0]);
2                   y = vNew[j] + omega*(y-vNew[j]);
3                   y = vNew[jMax] + omega*(y-vNew[jMax]);
```

and change the assignments of `y` to

```
y = max( X-S[0], vNew[0] + omega*(y-vNew[0]);
y = max( X-S[j], vNew[j] + omega*(y-vNew[j]);
y = max( X-S[jMax], vNew[jMax] + omega*(y-vNew[jMax]);
```

run the code and check the value. Make sure you use the correct values of 0, j , $jMax$ on the correct line.

8.7 Run this code for different values of $iMax$, $jMax$, S_max, $omega$, tol and $iterMax$. Try to check the errors and convergence of the method. You could try including a higher order interpolation using the code found here – (click here) – just copy/paste the function into your code.

8.3 Example - American Option with the Penalty Method

Here we aim to solve the classic American put option problem using the so called penalty method. We must solve the problem

$$\frac{\partial P}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 P}{\partial S^2} + rS\frac{\partial P}{\partial S} - rP = 0$$

with

$$P(S,T) = \max(X - S, 0),$$
$$P(S,t) \geq X - S,$$

and

$$S_0 = 9.735 \ , \ X = 10 \ , \ T = 1 \ , \ r = 0.05 \ , \ \sigma = 0.4 \ .$$

The penalty method works by solving a slightly different problem, which can be shown to converge to the true solution. The alternative problem we solve is given by

$$\frac{\partial P}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 P}{\partial S^2} + rS\frac{\partial P}{\partial S} - rP + \rho \max((X - S) - P, 0) = 0$$

where in the limit as $\rho \to \infty$, the value function P satisfies

$$P(S,t) \geq X - S.$$

The algorithm works as follows:

- Write down a standard finite difference scheme for the original PDE (including the boundaries)
- Take a guess at the solution v_j^q at the current time step, where v_j^q is the qth guess.
- If $v_j^q < X - S_j$ adjust the original finite difference approximation to

$$\hat{b}_j = b_j - \rho$$

$$\hat{d}_j = d_j - \rho(X - S_j)$$

- Solve new system with a Thomas algorithm to find v_j^{q+1}.

- Check for convergence in $||v^q - v^{q+1}||$.

In order to guarantee that our solution is accurate to the level 'tol', we can choose

$$\rho = \frac{1}{tol}.$$

8.8 Open the example code on my website CLICK HERE TO DOWNLOAD FULL CODE

```
/*
 * ON INPUT:
 * S0           -- initial stock price
 * X            -- exercise (strike) price
 * T            -- Time to expiry (years)
 * r            -- interest rate (per annum)
 * sigma        -- volatility (per annum^12)
 * iMax         -- number of time steps
 * jMax         -- number of space steps
 * SMax         -- Maximum value of S
 * rho          -- is the penalty parameter
 * tol          -- is the tolerance level
 * iterMax      -- is maximum iterations
 * ON OUTPUT:
 * return       -- the value of a American put option at S=S0, t=0
 */
double americanPut_Penalty(double S0,double X,double T,double r,double
    sigma,int iMax,int jMax,double SMax,double rho,double tol,int
    iterMax)
```

Run the code and check it gives the required result.

8.9 There are some more details and an interactive notebook if you click here.

8.10 Run this code for different values of $iMax$, $jMax$, S_max, rho, tol and $iterMax$. Try to check the errors and convergence of the method.

8.11 Update this code to include dividends, and make it solve for a call option. You will need to adjust the boundary conditions:- CLICK HERE TO DOWNLOAD FULL CODE

```
        // set up boundary condition for an American option
        a[0] = 0.;b[0] = 1.;c[0] = 0.;
        d[0] = X;
        // BC for American Option
        a[jMax] = 0.;b[jMax] = 1.;c[jMax] = 0.;
        d[jMax] = 0.;
```

the finite difference scheme to include dividends CLICK HERE TO DOWNLOAD FULL CODE

```
            a[j] = 0.25*(sigma*sigma*j*j-r*j);
            b[j] =-0.5*sigma*sigma*j*j - 0.5*r - 1./dt;
            c[j] = 0.25*(sigma*sigma*j*j+r*j);
```

and the penalty function

CLICK HERE TO DOWNLOAD FULL CODE

```
1            // apply penalty here to finite difference scheme
2            for(int j=1;j<jMax;j++)
3            {
4                // if current value suggests apply penalty, adjust
                     matrix equations
5                if( vNew[j] < X - S[j] )
6                {
7                    bHat[j]=b[j]-rho;dHat[j]=d[j] - rho*(X - S[j]);
8                }
9            }
```

Check your results against my online calculator for American option – (click here).