Lab15_sol

December 18, 2017

0.1 Sample solutions to exercises of Lab 15

Let us consider an average price put option (payoff function $p(s, A_T) = \max(E - A_T, 0)$, where E is the exercise price specified in the option contract). Assume $r = 0.05$, $D = 0$, $\sigma = 0.5$, $T = 0.5$, $E = 100$, $S(0) = 95$. Define $I_{max} = E\,T$, $S_{max} = 190$, $\phi_1(s,t) = 0$,

$$\phi_2(I,t) = \max\{Ee^{-r(T-t)} + \frac{e^{-D(T-t)}}{(r-D)T}(e^{-(r-D)(T-t)} - 1)S_{max} - \frac{e^{-r(T-t)}}{T}I, 0\}.$$

Write a function, that for given values n_s, n_I, m finds an approximate option price at $t = 0$ using the finite difference method described above. **Solution**

```
In [1]: import numpy as np
        from scipy import linalg

        def asian_solver(ns,nI,m,Smax,Imax,r,D,T,sigma,p,phi1,phi2):
            """sigma is assumed to be a function of s
            phi1,phi2 are functions of s,t,Imax and I,t,Smax
            solve in the region 0<=s<=Smax
            0<=I<=Imax
            0<=t<=T
            """
            delta_s=Smax/ns
            s=np.linspace(0,Smax,ns+1)
            #define the function alpha

            #define values of I_j
            I=np.linspace(0,Imax,nI+1)
            delta_I=Imax/nI
            #define t_k
            t=np.linspace(0,T,m+1)
            delta_t=T/m
            #define matrix U with dimension (n+1)x(m+1)
            V=np.zeros(shape=(ns+1,nI+1,m+1))
            #fill in the final condition
            for i in range(0,ns+1) :
                V[i,:,m]=p(s[i],I/T)
            i=np.arange(1,ns)
            #define matrix M
```

```
        M=np.zeros(shape=(ns+1,ns+1))
        M[0,0]=1
        M[ns,ns]=1
        #compute a,b,c
        rho=delta_t/delta_s**2
        a=rho/2*(-s[i]**2*sigma(s[i])**2+(r-D)*s[i]*delta_s)
        b=1+rho*s[i]**2*sigma(s[i])**2+s[i]*delta_t/delta_I+r*delta_t
        c=-rho/2*(s[i]**2*sigma(s[i])**2+(r-D)*s[i]*delta_s)
        #put into matrix M
        M[i,i-1]=a
        M[i,i]=b
        M[i,i+1]=c
        #define vector F
        F=np.zeros(ns+1)
        for k in range(m-1,-1,-1):
            #Fill in the values for I=Imax
            V[:,nI,k]=phi1(s,t[k],Imax)
            #All other values for j=nI-1,nI-2,...,0
            for j in range(nI-1,-1,-1):
                #Fill F
                F[0]=p(0,I[j]/T)*np.exp(-r*(T-t[k]))
                F[ns]=phi2(I[j],t[k],Smax)
                F[i]=s[i]*delta_t/delta_I*V[i,j+1,k]+V[i,j,k+1]
                #solve the system
                V[:,j,k]=linalg.solve(M,F)
        return V #option prices for t=0
r=0.05;D=0;T=0.5;E=100;S0=95
Imax=E*T
Smax=190
def sigma(s):
    return 0.5
def phi1(s,t,Imax):
    return 0;
def phi2(I,t,Smax):
    return np.maximum(E*np.exp(-r*(T-t))+np.exp(-D*(T-t))/((r-D)*T)*(
            np.exp(-(r-D)*(T-t))-1)*Smax-np.exp(-r*(T-t))*I/T,0)
def p(s,a):
    return np.maximum(E-a,0)

def price(m,ns,nI):
    answer=asian_solver(ns,nI,m,Smax,Imax,r,D,T,sigma,p,phi1,phi2)
    #price for t=0 and S(0)=S0
    #I is integral from 0 to t of stock price
    #at t=0 it is 0
    return answer[ns//2,0,0]
print(price(50,100,100))
```

```
12.258574036
```