Javier Zamora
fja2117
Modeling and Market
Making in Foreign Exchange
IEOR E4722
June 13, 2017

Lecture 3 (Options Markets) Assignment

Due start of class, Wednesday June 14, 2017.

Question 1 (4 marks)

Derive the expression for the ATM strike, given the ATM volatility and other market parameters, in the case where the market convention premium currency is the asset currency. In that case the price of the portfolio is

$$\Pi(S) = v(S) - S\frac{v_0}{S_0}$$

where $\Pi(S)$ is the price of the portfolio (which varies with spot S), $v(S)$ is the price of the option (which varies with spot S), S is the spot, S_0 is the initial spot (at the time of the trade), and v_0 is the initial price of the option (equal to $v(S_0)$). Note that the option here might be a call or might be a put.

Calculate the derivative of $\Pi(S)$ with respect to S to get the delta of the portfolio (taking at spot $S=S_0$), and then solve for the strike that makes the delta of a call portfolio equal to the negative of the delta of a put portfolio (using Black-Scholes formulas for the call and put prices and deltas).

The seminal formula is

$$V(S) = \phi[Se^{-QT}N(\phi x) - Ke^{-RT}N(\phi y)]$$

where

- $\rightarrow\ x = \left[\ln\frac{S}{K} + \left(R - Q + \frac{\sigma^2}{2}\right)T\right] \cdot \frac{1}{\sigma\sqrt{T}}$
- $\rightarrow\ y = x - \sigma\sqrt{T} = \left[\ln\frac{S}{K} + \left(R - Q - \frac{\sigma^2}{2}\right)T\right] \cdot \frac{1}{\sigma\sqrt{T}}$
- $\rightarrow\ \phi = \begin{cases} 1 & \mathit{for\ a\ call\ option} \\ -1 & \mathit{for\ a\ put\ option} \end{cases}$
- $\rightarrow$ N(x), cumulative standard normal distribution function
- $\rightarrow$ K, strike price
- $\rightarrow$ σ^2, variance rate of return of the exchange rate
- $\rightarrow$ R, foreign interest rate
- $\rightarrow$ T, time to maturity
- $\rightarrow$ Q, domestic interest rate
- $\rightarrow$ S, current spot exchange rate

For a Call option,

$$V_C(S) = Se^{-QT}N(x) - Ke^{-RT}N(y)$$

And for a Put option,

$$V_P(S) = -Se^{-QT}N(-x) + Ke^{-RT}N(-y)$$

From where the premium-included deltas of a Put and Call options are,

$$\Delta_C = \frac{\partial V_C(S)}{\partial S} = \frac{K}{S}e^{-QT}N(y)$$
$$\Delta_P = \frac{\partial V_P(S)}{\partial S} = -\frac{K}{S}e^{-QT}N(-y)$$

Imposing the condition that the delta of the call to be equal to the delta of the put,

$$\Delta_C = -\Delta_P => \frac{K}{S}e^{-QT}N(y) = \frac{K}{S}e^{-QT}N(-y) => N(y) = N(-y)$$

which can only happen if $y = 0$. Then

$$y = \left[\ln\frac{S}{K} + \left(R - Q - \frac{\sigma^2}{2}\right)T\right] \cdot \frac{1}{\sigma\sqrt{T}} = 0 => \ln\frac{S}{K} + \left(R - Q - \frac{\sigma^2}{2}\right)T = 0 =>$$

$$=> \frac{S}{K} = e^{-\left(R-Q-\frac{\sigma^2}{2}\right)T} => K = Se^{\left(R-Q-\frac{\sigma^2}{2}\right)T}$$

and since

$$F = Se^{(R-Q)T}$$

then

$$K = Fe^{-\frac{\sigma^2}{2}T}$$

Question 2 (4 marks)

Assume an FX market where the spot is 1, time to expiration is 0.5y, forward points are +0.0040, and the denominated discount rate is 1.75%. What is the strike corresponding to a 25-delta call option when its implied volatility is 8.75%, using market-convention delta? Assume the market convention premium currency is the denominated currency.

- → S=1
- → T=0.5
- → $f.p. = 0.0040$
- → R=1.75%
- → $\Delta_C = 0.25$
- → $\sigma_K = 8.75\%$

$$f.p. = F - S => F = fp + S = 1 + 0.0040 = 1.0040$$

$$F = Se^{(R-Q)T} => Q = R - \frac{1}{T}\ln\left(\frac{F}{S}\right) = 0.0175 - \frac{1}{0.5}\ln\left(\frac{1.004}{1}\right) = 0.95\%$$

$$N^{-1}(\Delta_C e^{QT}) = N^{-1}(0.25e^{0.0095*0.5}) = N^{-1}(0.2512) = -0.671$$

$$K_C = Fe^{\frac{\sigma_K^2}{2}T - \sigma_K\sqrt{T}N^{-1}(\Delta_C e^{QT})} = 1.004 \cdot e^{\frac{0.0875^2}{2}\cdot 0.5 - 0.0875\cdot\sqrt{0.5}\cdot(-0.671)} = 1.048$$

Question 3 (2 marks)

Describe the risk reversal "beta".

The risk reversal beta expresses the relationship between daily risk reversal and daily spot returns. In this context risk reversal is understood as a measurement of the implied volatility skew. Specifically, beta risk reversal is the slope of a linear regression of daily risk reversal change against daily spot log returns.

Question 4 (2 marks)

Explain the two arbitrage conditions that should be avoided when interpolating in the strike direction, and the one (weak) arbitrage condition to avoid when interpolating in the time direction.

The two arbitrage conditions that should be avoided are:

→ $\frac{dC}{dK} > 0$, which would produce call prices to increase as strike increases
→ $\frac{d^2C}{dK^2} < 0$, which would produce a negative call price curvature with respect to strike

Question 5 (10 marks)

Implement a cubic spline interpolation for implied volatility vs strike which has non-standard boundary conditions to give more intuitive volatility extrapolation.

Assume you are given five implied volatilities, σ_1 through σ_5, for five known strikes, K_1 through K_5, for some known time to expiration T.

Set up the boundary condition of the cubic spline such that implied volatility reaches a constant value a certain distance beyond the edge points. The distance is defined by a parameter, the **cubic spline extrapolation parameter, F**. On the left side, for strikes less than the minimum marked strike K_1, implied volatility reaches a constant value at strike K_{min}:

$$K_{min} = K_1 e^{-F\sigma_1\sqrt{T}}$$

and on the right side, for strikes greater that the maximum marked strike K_5, implied volatility reaches a constant value at strike K_{max}:

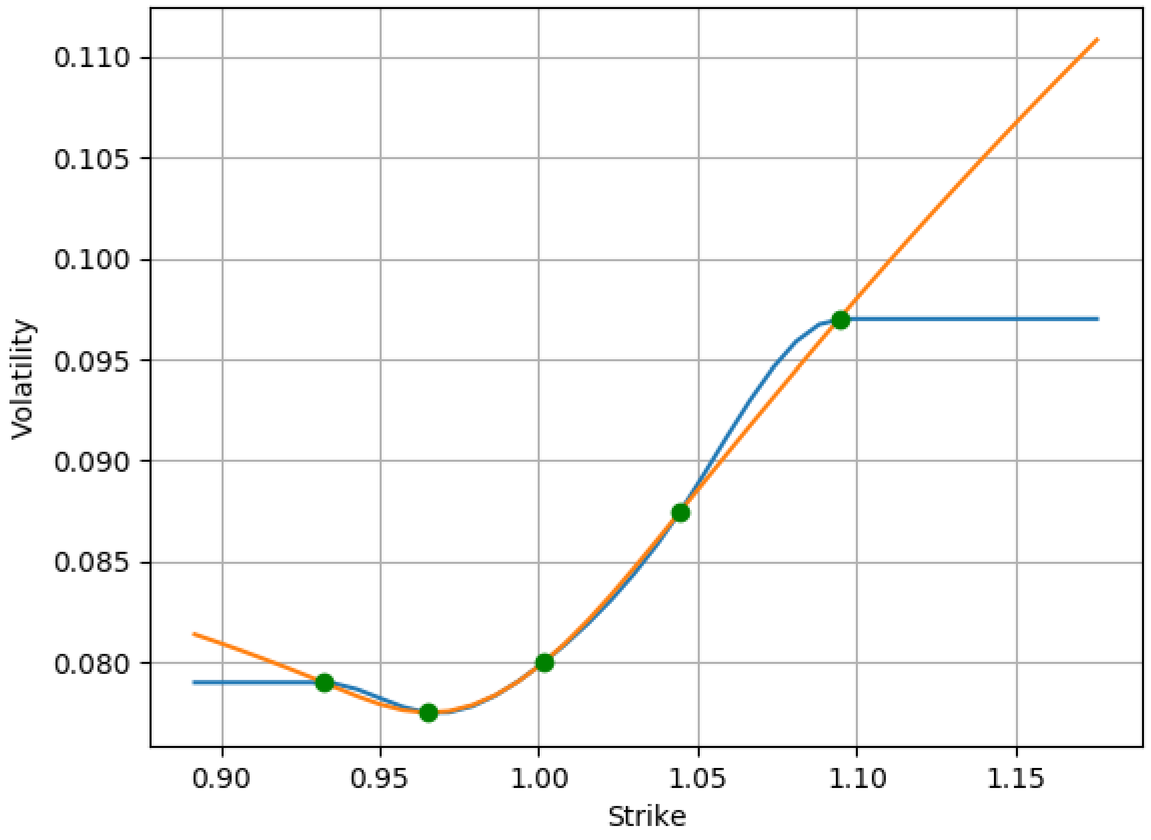
$$K_{max} = K_5 e^{+F\sigma_5\sqrt{T}}$$

Start with the standard method for defining a cubic spline (eg here: http://www.aip.de/groups/soe/local/numres/bookcpdf/c3-3.pdf), but add in two extra points at K_{min} and K_{max} where you know that the slope of vol vs strike goes to zero, and the second derivative of vol vs strike goes to zero, but where you do not know the value of volatility (that comes out of the solution for the cubic spline). Of course the first and second derivatives of vol vs strike should be continuous across K_1 and K_5.

Implement a Python class that is initialized with the five strikes, five implied volatilities, time to expiration, and cubic spline extrapolation parameter. As part of initialization it will need to solve a linear system to generate the cubic spline parameters, for which you can use the functions in the scipy or numpy packages. The class should include a “volatility” method that takes a strike and returns an interpolated volatility.

Finally, generate plots of implied volatility vs strike for F=0.01 and F=10 for the following market: T=0.5y; ATM volatility of 8%, 25-delta risk reversal of 1%, 10-delta risk reversal of 1.8%, 25-delta butterfly of 0.25%, and 10-delta butterfly of 0.80%; spot=1; forward points=0; denominated discount rate=0%. For this you can use the plotxy function in the WST library wst/util/plot.py; you should return one chart with the two curves on it, showing implied volatility vs strike for all strikes between the 1-delta put and 1-delta call.

Give some intuition behind the shape of the two plots in terms of the value of F.



The curves will become horizontal as soon as will arrive to the control points K_{min} and K_{max}. With $F = 0.01$ that happens very closely to the 10-delta points, but with F=10 the control points are located beyond the (roughly approximated) 1-delta points.

```
import scipy
import scipy.stats
import matplotlib.pyplot as plt

def fit_vol(Y,K,F,T):
    K_min=K[0]*scipy.exp(-F*Y[0]*scipy.sqrt(T))
    K_max=K[4]*scipy.exp(F*Y[4]*scipy.sqrt(T))
    X=scipy.concatenate((K_min,K,K_max),axis=0)
    X2=scipy.multiply(X,X)
    X3=scipy.multiply(X2,X)

    M=scipy.matrix([[-1,float(X3[0]),float(X2[0]),float(X[0]),1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
        [0,   float(X3[1]),float(X2[1]),float(X[1]),1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
        [0,0,0,0,0,float(X3[1]),float(X2[1]),float(X[1]),1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
        [0,0,0,0,0,float(X3[2]),float(X2[2]),float(X[2]),1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
        [0,0,0,0,0,0,0,0,0,float(X3[2]),float(X2[2]),float(X[2]),1,0,0,0,0,0,0,0,0,0,0,0,0,0],
        [0,0,0,0,0,0,0,0,0,float(X3[3]),float(X2[3]),float(X[3]),1,0,0,0,0,0,0,0,0,0,0,0,0,0],
        [0,0,0,0,0,0,0,0,0,0,0,0,0,float(X3[3]),float(X2[3]),float(X[3]),1,0,0,0,0,0,0,0,0,0],
        [0,0,0,0,0,0,0,0,0,0,0,0,0,float(X3[4]),float(X2[4]),float(X[4]),1,0,0,0,0,0,0,0,0,0],
        [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,float(X3[4]),float(X2[4]),float(X[4]),1,0,0,0,0,0],
        [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,float(X3[5]),float(X2[5]),float(X[5]),1,0,0,0,0,0],
        [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,float(X3[5]),float(X2[5]),float(X[5]),1,0],
        [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,float(X3[6]),float(X2[6]),float(X[6]),1,-1],
        [0,-3*float(X2[0]),-2*float(X[0]),-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
        [0,3*float(X2[1]),2*float(X[1]),1,0,-3*float(X2[1]),-2*float(X[1]),-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
        [0,0,0,0,0,3*float(X2[2]),2*float(X[2]),1,0,-3*float(X2[2]),-2*float(X[2]),-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
        [0,0,0,0,0,0,0,0,0,3*float(X2[3]),2*float(X[3]),1,0,-3*float(X2[3]),-2*float(X[3]),-1,0,0,0,0,0,0,0,0,0,0],
        [0,0,0,0,0,0,0,0,0,0,0,0,0,3*float(X2[4]),2*float(X[4]),1,0,-3*float(X2[4]),-2*float(X[4]),-1,0,0,0,0,0,0],
        [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3*float(X2[5]),2*float(X[5]),1,0,-3*float(X2[5]),-2*float(X[5]),-1,0,0],
        [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3*float(X2[6]),2*float(X[6]),1,0,0],
        [0,-6*float(X[0]),-2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
        [0,6*float(X[1]),2,0,0,-6*float(X[1]),-2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
        [0,0,0,0,0,6*float(X[2]),2,0,0,-6*float(X[2]),-2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
        [0,0,0,0,0,0,0,0,0,6*float(X[3]),2,0,0,-6*float(X[3]),-2,0,0,0,0,0,0,0,0,0,0,0],
        [0,0,0,0,0,0,0,0,0,0,0,0,0,6*float(X[4]),2,0,0,-6*float(X[4]),-2,0,0,0,0,0,0,0],
        [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6*float(X[5]),2,0,0,-6*float(X[5]),-2,0,0,0],
        [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6*float(X[6]),2,0,0,0]])


n=scipy.matrix([0,float(Y[0]),float(Y[0]),float(Y[1]),float(Y[1]),float(Y[2]),float(Y[2]),float(Y[3]),float(Y[3]),float(Y[4]),
float(Y[4]),0,0,0,0,0,0,0,0,0,0,0,0,0,0,0])
    n=scipy.matrix.transpose(n)
    coef=M.I*n

    z3=scipy.matrix(scipy.zeros((3,1)))
    coef=scipy.concatenate((z3,coef[0:-1],z3,coef[-1]),axis=0)
    coef=scipy.reshape(coef,(8,4))
    a=coef[:,0]
    b=coef[:,1]
    c=coef[:,2]
    d=coef[:,3]

    max_strk=float(K[-1]+0.50*(K[-1]-K[0]))
    min_strk=float(K[0]-0.25*(K[-1]-K[0]))
    num_points=40
    step=(max_strk-min_strk)/(num_points-1)
    vol=scipy.matrix(scipy.zeros((num_points,1)))
    strk=scipy.arange(min_strk,max_strk+step,step)
    for cont in range(scipy.size(strk)):
        inx=(X<=strk[cont])
        if scipy.all(inx==0):
            inx=0
        else:
            inx=inx.nonzero()
            inx=max(inx[0])+1
        strkk=float(strk[cont])
        vol[cont]=float(a[inx])*(strkk**3)+float(b[inx])*(strkk**2)+float(c[inx])*strkk+float(d[inx])

    return strk,vol

def main():

    spot=1
    vol_atm=0.08
    dRR_25=0.01
    dRR_10=0.018
    dBF_25=0.0025
    dBF_10=0.0080
    T=0.5
    F=[0.01, 10]

    vol=scipy.matrix(scipy.zeros((5,1)))
    vol[0]=vol_atm+dBF_10-0.5*dRR_10
    vol[1]=vol_atm+dBF_25-0.5*dRR_25
    vol[2]=vol_atm
    vol[3]=vol_atm+dBF_25+0.5*dRR_25
    vol[4]=vol_atm+dBF_10+0.5*dRR_10

    K=scipy.matrix(scipy.zeros((5,1)))
    K[0]=spot*scipy.exp(vol[0]*vol[0]*0.5*T+vol[0]*scipy.sqrt(T)*scipy.stats.norm.ppf(0.10))
    K[1]=spot*scipy.exp(vol[1]*vol[1]*0.5*T+vol[1]*scipy.sqrt(T)* scipy.stats.norm.ppf(0.25))
    K[2]=spot*scipy.exp(vol[2]*vol[2]*0.5*T)
    K[3]=spot*scipy.exp(vol[3]*vol[3]*0.5*T-vol[3]*scipy.sqrt(T)*scipy.stats.norm.ppf(0.25))
    K[4]=spot*scipy.exp(vol[4]*vol[4]*0.5*T-vol[4]*scipy.sqrt(T)*scipy.stats.norm.ppf(0.10))

    [K1,v1] = fit_vol(vol,K,F[0],T)
    [K2,v2] = fit_vol(vol,K,F[1],T)

    plt.plot(K1, v1)
    plt.plot(K2, v2)
    plt.plot(K,vol,'go')
    plt.xlabel('Strike')
    plt.ylabel('Volatility')
    plt.grid()
    plt.show(block=False)

if __name__ == '__main__':
    main()
```