Pricing Barrier options using Python

Rob Wenneker Esther Torres

October 18, 2014

Contents

1 Introduction

In this seminar, we will try to price simple(i.e. one barrier) options using Python. As you will see, we tried to make our programme as general as possible. In order to price this barrier options, we used the formula's given in the lecture notes of Jan Roman. Even though we will not go through them thoroughly, we will show them to you later on in this seminar. These formula's are based on the Black-Sholes model. Besides explaining our model, we will give a few examples on what the different barrier options are to give you an idea about the intuition behind barrier options. First, let's see some examples.

2 Our model and its assumptions

As we said before, we wanted to make our model as general as possible. With our programme, you can price a barrier option at any time t before maturity. In order to do that, we had to include all variables as a user-based input. Our programme will ask you for all the information which is needed to calculate the option price. Let's go through these variables:

Current Stock price
The current stock price S is a fixed number for the stock price at the moment of pricing. As we will show to you later on, changing S will have a huge impact on the price of the barrier option. The stock price can be any number $S > 0$.

Strike price
The strike price X is fixed and will be constant throughout the lifetime of the option. The strike price of a put(call) is the price agreed upon by the seller and the buyer to sell(buy) 1 unit of the underlying asset if the holder of the option decides to do so.

Barrier Height
Like the strike price, the barrier height H is a fixed and constant number. In our model, we assume that $H \neq S$.

Time to maturity
Normally, the time to maturity is given by $T - t$. However, for simplicity we assume that $t = 0$(i.e. you are pricing an option at this moment) and therefore time to maturity is given by T. Our programme will ask for T in days, but for the formula's the T will be recalculated into years.

Risk-Free interest rate
Our model uses the yearly interest rate as r. As time to maturity will be used in years as well(despite the fact that you can enter it as the number of days), we assume that interest is payed once per year(i.e. This is the effective risk-free

interest rate)

Cost of Carry rate

Some assets generate dividends in addition to a possible change of the stock price. We need to take this into account with discounting. Therefore we define the (cost of) carry rate b as $r - q$. Likewise if you are trading a currency option, b will be defined as $r - r_f$.

Volatilities

In order to estimate the risk and the expected movement of the given asset, the volatility σ will be used. What kind of volatility is used is up to the user. (e.g. Yearly or monthly volatilities). The only requirement we make is that $0 < \sigma < 1$. Therefore, a percentage will have to be recalculated to a decimal number before entering it in the programme.

Rebate

In a knock-out barrier option, there is a possibility to enter a rebate K, which will be payed out to the holder at maturity in case his option gets knocked out. This is some kind of insurance against being knocked out. A rebate will obviously increase the value of the option.

3 The Black-Scholes formula's

For Barrier options in the Black-Scholes world, there exists a number of formula's in order to price all kinds of barrier options. (e.g. Down-and-in Call or Up-and-in Put). These formula's are the following:

$$\eta = \begin{cases} 1 & if Down \\ -1 & if Up \end{cases}$$

$$\phi = \begin{cases} 1 & if Call \\ -1 & if Put \end{cases}$$

$\mu = \frac{b-\sigma^2/2}{\sigma^2}$, $\lambda = \sqrt{\mu + \frac{2r}{\sigma^2}}$

$x_1 = \frac{ln(\frac{S}{X})}{\sigma\sqrt{T}} + (1+\mu)\sigma\sqrt{T}$, $x_2 = \frac{ln(\frac{S}{H})}{\sigma\sqrt{T}} + (1+\mu)\sigma\sqrt{T}$

$y_1 = \frac{ln(\frac{H^2}{SX})}{\sigma\sqrt{T}} + (1+\mu)\sigma\sqrt{T}$, $y_2 = \frac{ln(\frac{H}{S})}{\sigma\sqrt{T}} + (1+\mu)\sigma\sqrt{T}$

$z = \frac{ln(\frac{H}{S})}{\sigma\sqrt{T}} + \lambda\sigma\sqrt{T}$

$A = \phi S e^{b-r} N(\phi x_1) - \phi X e^{-rT} N(\phi x_1 - \phi\sigma\sqrt{T})$
$B = \phi S e^{b-r} N(\phi x_2) - \phi X e^{-rT} N(\phi x_2 - \phi\sigma\sqrt{T})$
$C = \phi S e^{b-r} \frac{H}{S}^{2(\mu+1)} N(\eta y_1) - \phi X e^{-rT} (\frac{H}{S})^{2\mu} N(\eta y_1 - \eta\sigma\sqrt{T})$
$D = \phi S e^{b-r} \frac{H}{S}^{2(\mu+1)} N(\eta y_2) - \phi X e^{-rT} (\frac{H}{S})^{2\mu} N(\eta y_2 - \eta\sigma\sqrt{T})$
$E = K e^{-rT} \left[N(\eta x_2 - \eta\sigma\sqrt{T}) - (\frac{H}{S})^{2\mu} N(\eta y_2 - \eta\sigma\sqrt{T}) \right]$
$F = K e^{-rT} \left[(\frac{H}{S})^{\mu+\lambda} N(\eta z) - (\frac{H}{S})^{\mu-\lambda} N(\eta z - 2\eta\sigma\lambda\sqrt{T}) \right]$

Given these formula's, the prices of Call barrier options are given by:

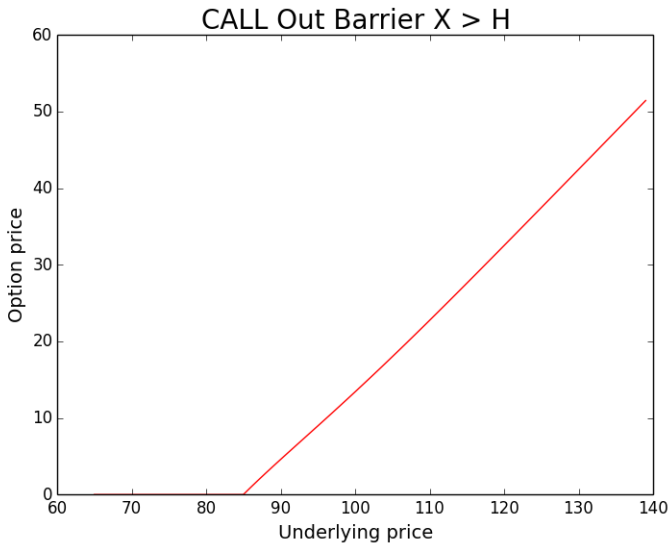
Type	X <H	X >H
Down-and-In S >H	A - B + D + E	C + E
Up-and-In S <H	B - C + D + E	A + E
Down-and-Out S >H	B - D + F	A - C + F
Up-and-Out S <H	A - B + C - D + F	F

For Put barrier options we have:

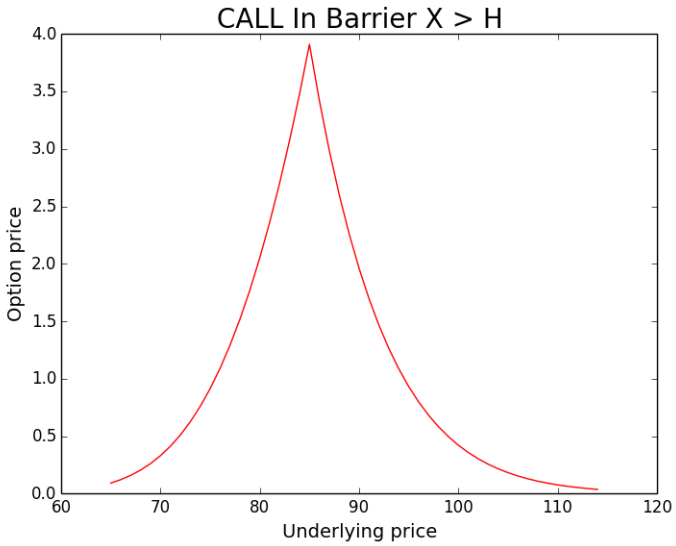
Type	X <H	X >H
Down-and-In S >H	A + E	B - C + D + E
Up-and-In S <H	C + E	A - B + D + E
Down-and-Out S >H	F	A - B + C - D + F
Up-and-Out S <H	A - C + F	B - D + F

4 Sensitivity Analysis

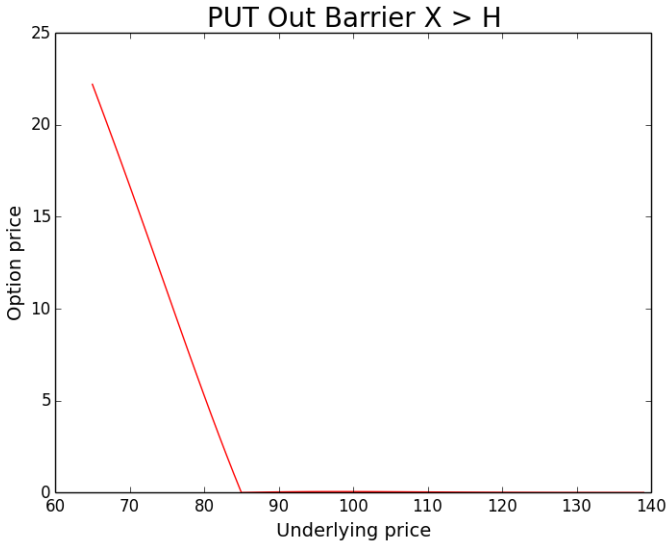
We were interested to see how much the price of a barrier option will change if the price of its underlying changes for several option types. In order to do this, we made the following model:
$T = 200$
$\sigma = 0.20$
$r = 0.05$
$b = 0.05$ (i.e. no dividends)
$K = 0$
$X = 90$
$H = 85$
We are then left with the variable S. Now, we want to do a little sensitivity analysis for both an in and an out barrier. Then we do this for the situation where $X > H$
In all of the following examples we will take $S = 90$ and then deviate (to both sides) and calculate the corresponding option price. First, let's see a call-out.



Similarly, for a call-in barrier, we have:



For a put-out the graph is like this.



5 Examples

Suppose we have a barrier option of company A with the following characteristics:
The initial stock price S is equal to 66. The barrier option is a put option, and its strike price X is equal to 64. The barrier is an in-barrier and the height H is equal to 62. The volatility of this stock σ is equal to 12%. The risk free-interest rate is equal to 3% and this option does not pay any dividends. The option expires 90 days from now. Then, our programme tells us this option is worth 0.56766315915

Suppose we have a barrier option of company B with the following characteristics:
The initial stock price S is equal to 75. The barrier option is a call option, and its strike price X is equal to 78. The barrier is an in-barrier and the height H is equal to 74. The volatility of this stock σ is equal to 17%. The risk free-interest rate is equal to 5% and the cost of carry rate b is equal to 4%. The option expires 120 days from now. If you will get knocked out, you will receive a rebate of 4. Then, our programme tells us this option is worth 1.20531787821

6 Appendix

In this appendix we will place the programming code we used in Python.

```
import math
from scipy.stats.distributions import norm
import matplotlib.pyplot as plt



error = True
message = "This input is incorrect. Please enter p for put or c for call"
while error:
    error = False
    putCall = raw_input("Do you want to price a put(p) or a call(c) option? " )
    if putCall == 'p':
        phi = -1
    elif putCall == 'c':
        phi = 1
    else:
        print message
        error = True

S = input("Please enter the current Stock price- ")
X = input("Please enter the strike price of the option- ")
T = input("Please enter the time to maturity(in days)- ")
T = float(T)/365

sigma = input("Please enter the volatility of the underlying asset- ")
b = input("Please enter the carry rate- ")
r = input("Please enter the yearly interest rate- ")
r = r

H = input("Please enter the barrier height- " )


K = input("How high is the rebate you receive if you get knocked out?- ")

message3 ="Please enter out or in."
error3 = True
while error3:
    inOut = raw_input("Please type whether it is an out or an in barrier.(out/in)- ")
    if inOut == 'out' or inOut == 'in':
        error3 = False
    else:
```

```
        print message3


X = float(X)

b = float(b)

H = float(H)

K = float(K)


S = float(S)
####Function for evaluating the option price given the underlying price
def price_option(S):

    if putCall == 'c':
        phi = 1
    else:
        phi = -1
    if S>H:
        eta = 1
    else:
        eta = -1


    sigma2 = sigma**2
    mu = (b-(sigma2 / 2))/sigma2
    landa = math.sqrt(mu**2.0 + (2*r)/(sigma2))
    x1 = (math.log(S/X)) / (sigma*math.sqrt(T)) + (1+mu)*sigma*math.sqrt(T)
    x2 = (math.log(S/H)) / (sigma*math.sqrt(T)) + (1+mu)*sigma*math.sqrt(T)
    y1 = math.log((H**2)/(S*X)) / (sigma*math.sqrt(T)) + (1+mu)*sigma*math.sqrt(T)
    y2 = math.log(H/S) / (sigma*math.sqrt(T)) + (1+mu)*sigma*math.sqrt(T)
    z = math.log(H/S) / (sigma*math.sqrt(T)) + landa*sigma*math.sqrt(T)


    n1=phi*x1-phi*sigma*math.sqrt(T)
    A= phi * S*math.exp(b-r)*norm.cdf(phi*x1,0,1) - phi*X*math.exp(-r*T)*norm.cdf(n1,0,1)

    n2=phi*x2-phi*sigma*math.sqrt(T)
    B= phi * S*math.exp(b-r)*norm.cdf(phi*x2,0,1) - phi*X*math.exp(-r*T)*norm.cdf(n2,0,1)


    n3=eta*y1
    n4=eta*y1-eta*sigma*math.sqrt(T)
```

```
C= phi * S*math.exp(b-r)*(H/S)**(2*(mu+1))*norm.cdf(n3,0,1) - phi*X*math.exp(-r*T)*(H/S)


n5=eta*y2
n6=eta*y2-eta*sigma*math.sqrt(T)
D= phi * S*math.exp(b-r)*(H/S)**(2*(mu+1))*norm.cdf(n5,0,1) - phi*X*math.exp(-r*T)*(H/S)

n9 = norm.cdf(eta*x2 - eta*sigma*math.sqrt(T),0,1)
n10 = norm.cdf(eta*y2 - eta*sigma*math.sqrt(T),0,1)
E = K*math.exp(-r*T)*(n9 - (H/S)**(2*mu)*n10)

n11 = norm.cdf(z*eta, 0,1)
n12 = norm.cdf(eta*z-2*eta*landa*sigma*math.sqrt(T),0,1)
F = K*math.exp(-r*T)*((H/S)**(mu+landa)*(n11)-(H/S)**(mu-landa)*n12)



if putCall == 'p':
    if X < H:
        if S > H:
            if inOut == 'in':
                price = A + E
            elif inOut == 'out':
                price = F
        else:
            if inOut  == 'in':
                price = C + E
            elif inOut == 'out':
                price = A - C + F
    else:
        if S > H:
            if inOut == 'in':
                price = B - C + D + E
            elif inOut == 'out':
                price = A - B + C - D + F
        else:
            if inOut  == 'in':
                price = A - B + D + E
            elif inOut == 'out':
                price = B - D + F


else:
    if X < H:
        if S > H:
            if inOut == 'in':
```

```
                    price = A - B + D + E
                elif inOut == 'out':
                    price = B - D + F
            else:
                if inOut  == 'in':
                    price = B - C + D + E
                elif inOut == 'out':
                    price = A - B + C - D + F
        else:
            if S > H:
                if inOut == 'in':
                    price = C + E
                elif inOut == 'out':
                    price = A - C + F
            else:
                if inOut  == 'in':
                    price = A + E
                elif inOut == 'out':
                    price = F
    if price < 0:
        price = 0

    return price




##Underlying price
print price_option(S)
S=int(S)
u_price = range (S-25, S+50)
##Option price
op_price = []
for i in range(len(u_price)):
    op_price.append(price_option(u_price[i]))


print u_price
print op_price

#Plotting the underlying price vs. the option price
plt.plot(u_price, op_price, color='red')
plt.xlabel('Underlying price', fontsize=14)
plt.ylabel('Option price',fontsize=14)
plt.title('PUT Out Barrier X < H', fontsize=20)
```

```
plt.show()
```

7 References

Downey A.B (2012) Think Python: How to think like a computer scientist
Roman, J.R.M. (2014) Lecture notes in Analytical Finance I