Mälardalen University
Department of Mathematics and Physics
Analytical Finance
Date

Using pegged strike and Richardson Extrapolation to smoothen the European option pricing curve

Instructor: Jan Röman

Tom Schöblom
Sylvester Jarlee

TABLE OF CONTENTS

1.0 INTRODUCTION

During this seminar, we use excel VBA to evaluate the prices of European put and call option. Also, pricing models such as the Binomial CRR and Black Scholes were used to determine the pegging strike and extrapolated European call and put option prices.

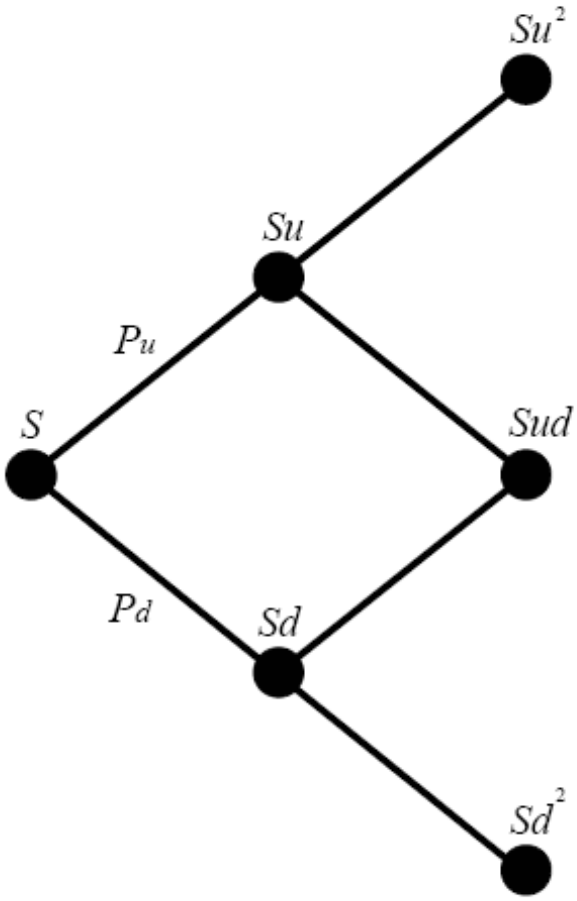
In all the graphs that is included in the seminar the following inputs have been used:

Calculating the price for an European Call option:
Stock (S): 100,
Strike (K):95,
Life of the option (T): 0.25,
Risk-free rate (rf): 0.07,
Standard deviation (Sigma): 0.2,
Number of steps (n): 514 (even numbers from 2-1028)

Calculating the price for an European Put option:
Stock (S): 100,
Strike (K):150,
Life of the option (T): 0.25,
Risk-free rate (rf): 0.07,
Standard deviation (Sigma): 0.2,
Number of steps (n): 514 (even numbers from 2-1028)

2.0 FRAMEWORK

2.1 Binomial Model of CRR



Given the graphical discription of the binomial tree above, we priced obtions in this paper using the Cox, Russ & Rubenstein model where,

$u = e^{\sigma\sqrt{\Delta t}}$,

$d = 1/u = e^{-\sigma\sqrt{\Delta t}}$ and

$$p = \frac{e^{r\Delta t} - d}{u - d}$$

2.2 **Black-Scholes**

To obtain the Black & Scholes prices for the European call and put options, we consider the following;

$$d_1 = \frac{\ln(S/X) + (r + (\sigma^2/2))T}{\sigma\sqrt{T}} = \frac{\ln(S/X)}{\sigma\sqrt{T}} + (rT + \frac{\sigma\sqrt{T}}{2}),$$

$$d_2 = \frac{\ln(S/X) + (r - (\sigma^2/2))T}{\sigma\sqrt{T}} = \frac{\ln(S/X)}{\sigma\sqrt{T}} + (rT - \frac{\sigma\sqrt{T}}{2}) = d_1 - \sigma\sqrt{T}$$

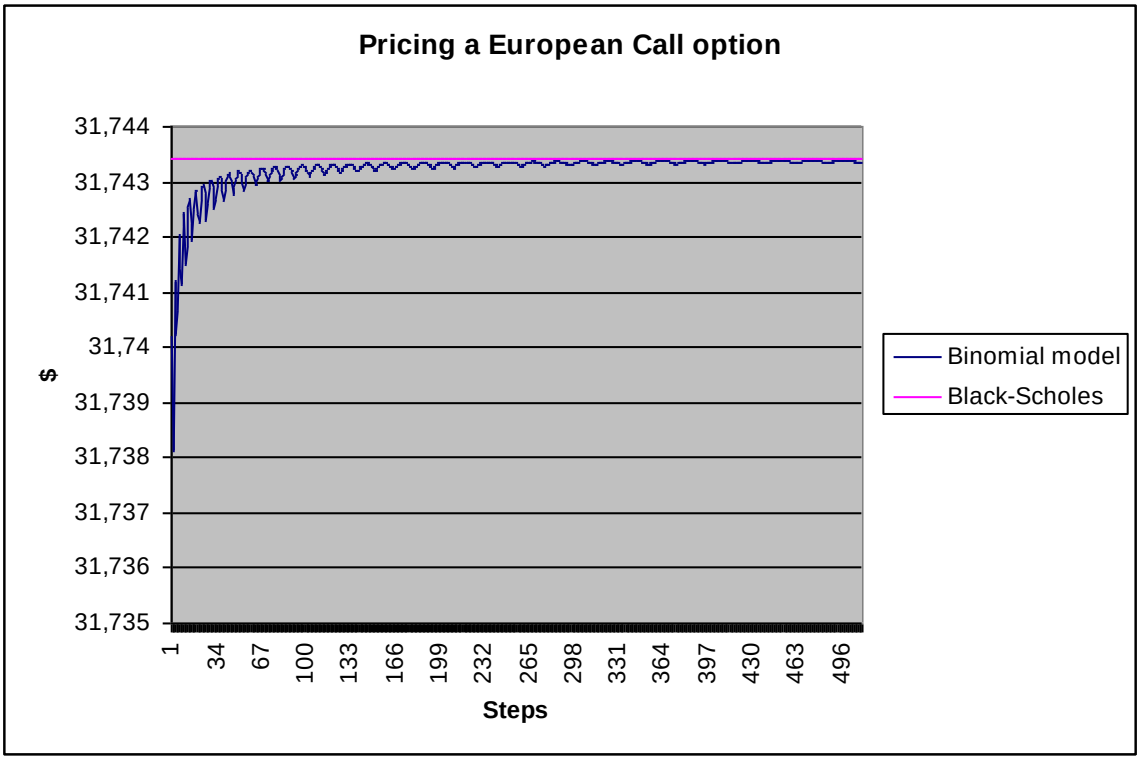
Then the call and put option prices are given as

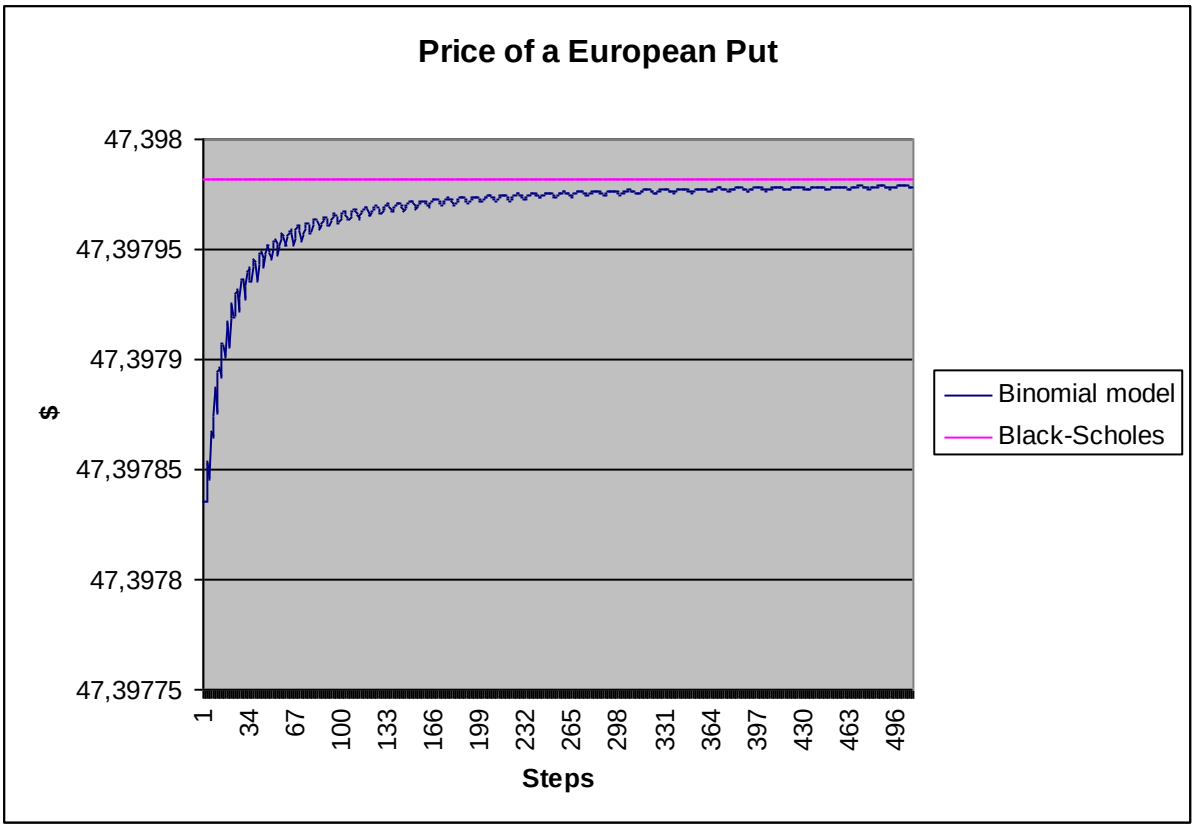
$$C = SN(d_1) - Xe^{-rT}N(d_2)$$

and

$$P = Xe^{-rT}N(-d_2) - SN(-d_1)$$

See graph below





2.3 Pegging the Strike Price

To reduce the oscillation that appears when we calculate the option prices via the binomial model, we will implement a model "called pegging the strike". This will hopefully create a smoother price curve than the one obtained using the CRR model. This is done by, for any even step (n), keeping the strike price fixed in the middle of the two corresponding nodes.

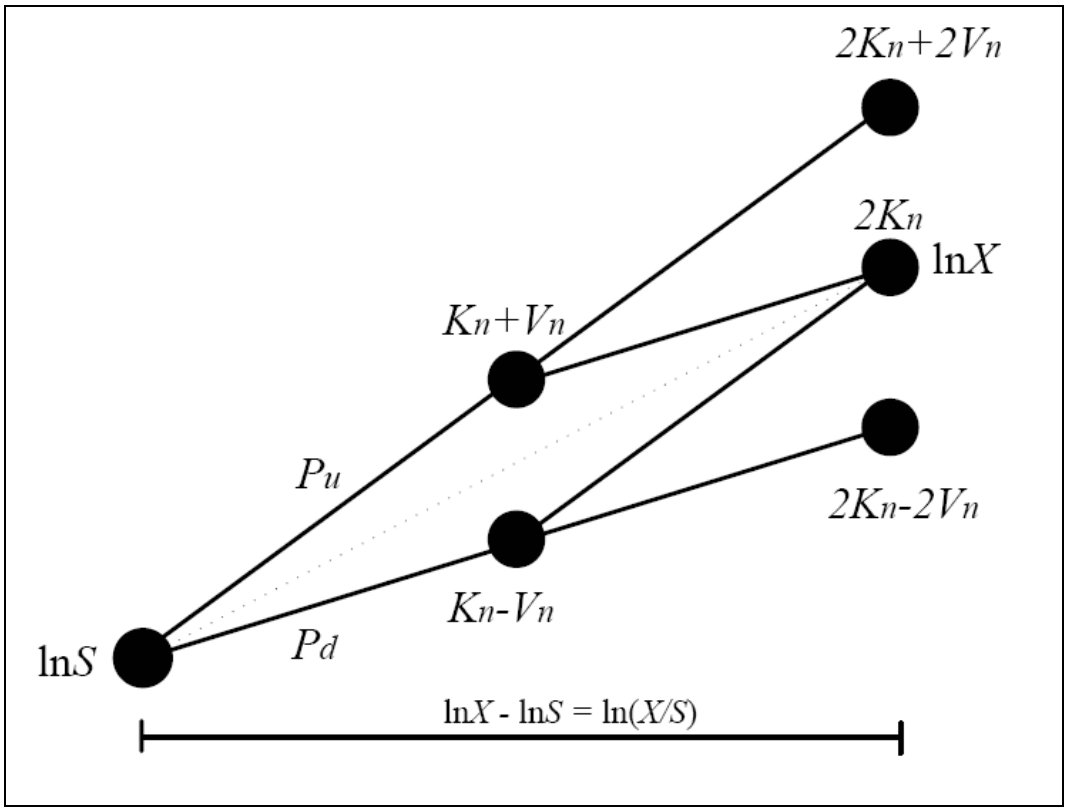
We pegged the strike using :

$u = e^{K_n + V_n}$, $d = e^{K_n - V_n}$,

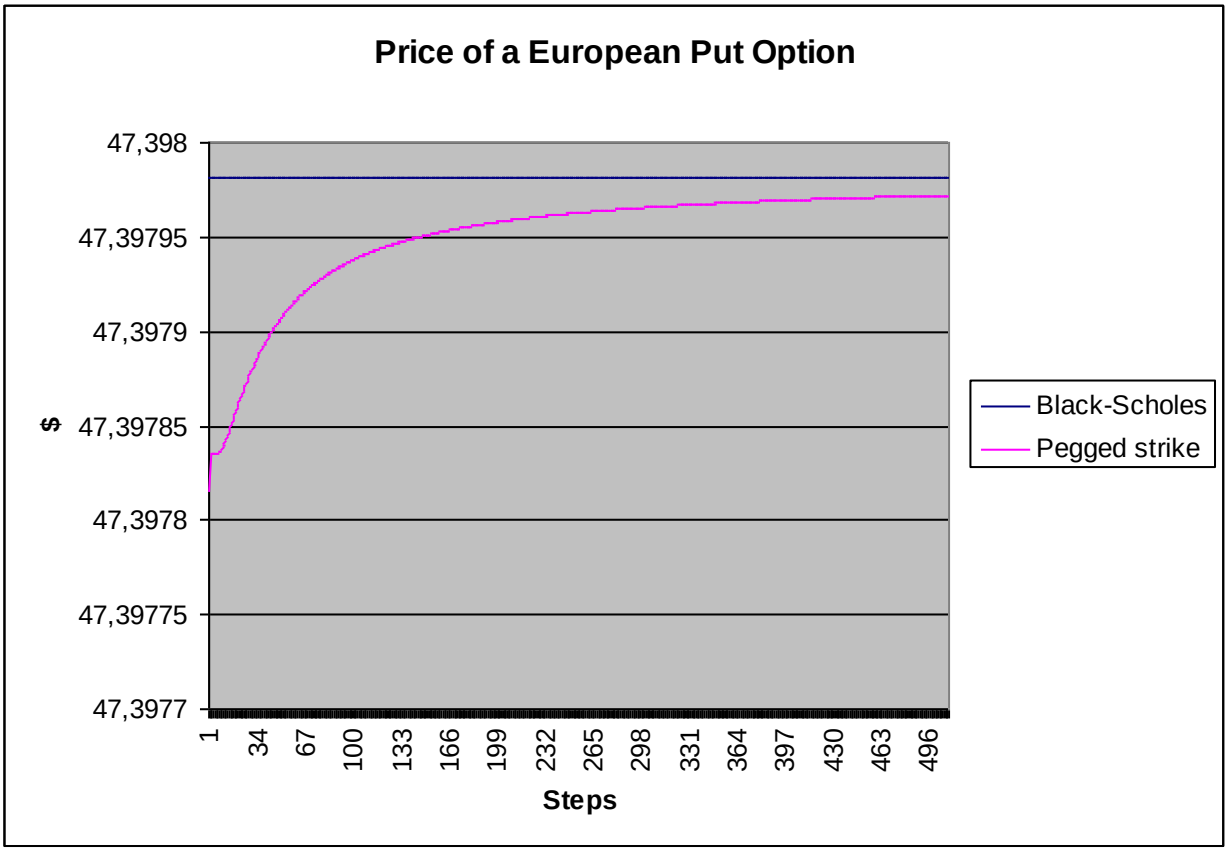
$K_n = \frac{\ln(X/S)}{n}$, and $V_n = \sigma\sqrt{\Delta t}$

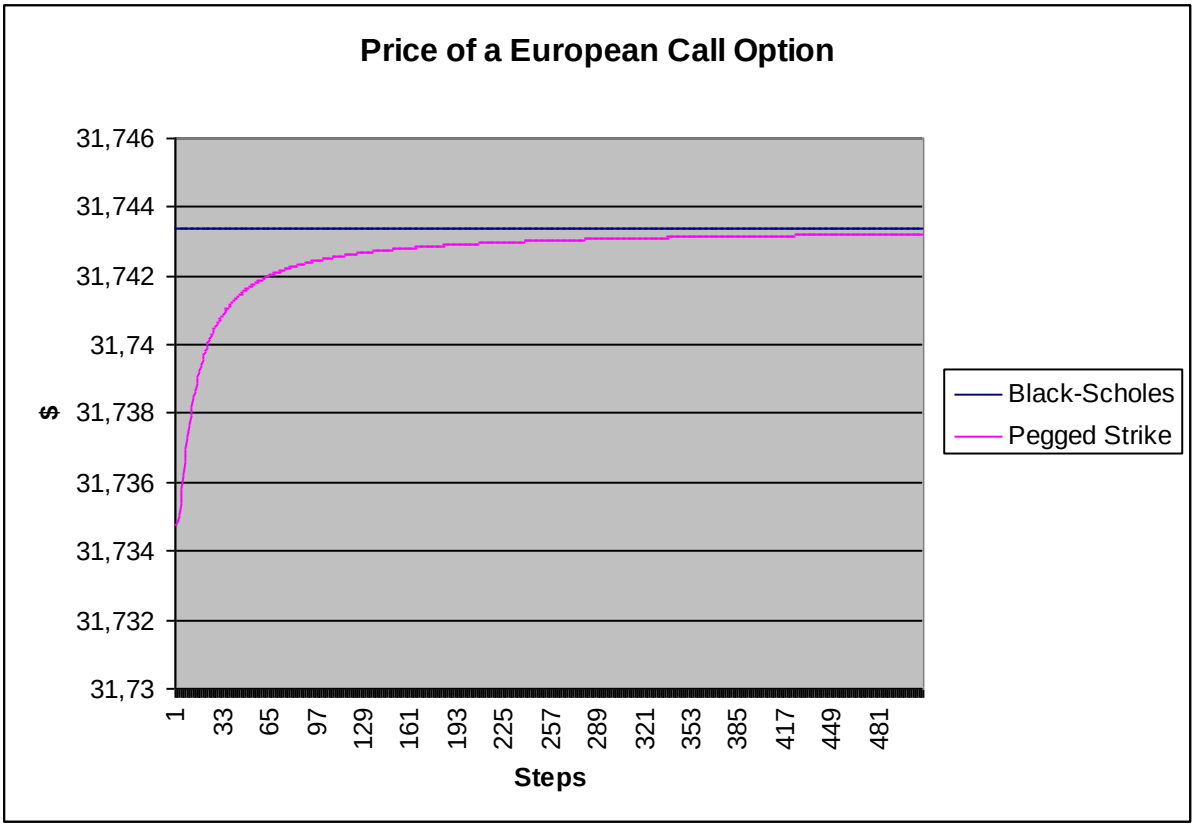
where,

u = *up factor*
d = *down factor*
K_n = *Pegged Strike*
V_n = *Change in Strike (nodes)*



Source: Chao-Jung Chen "Pricing European and American Options with Extrapolation"

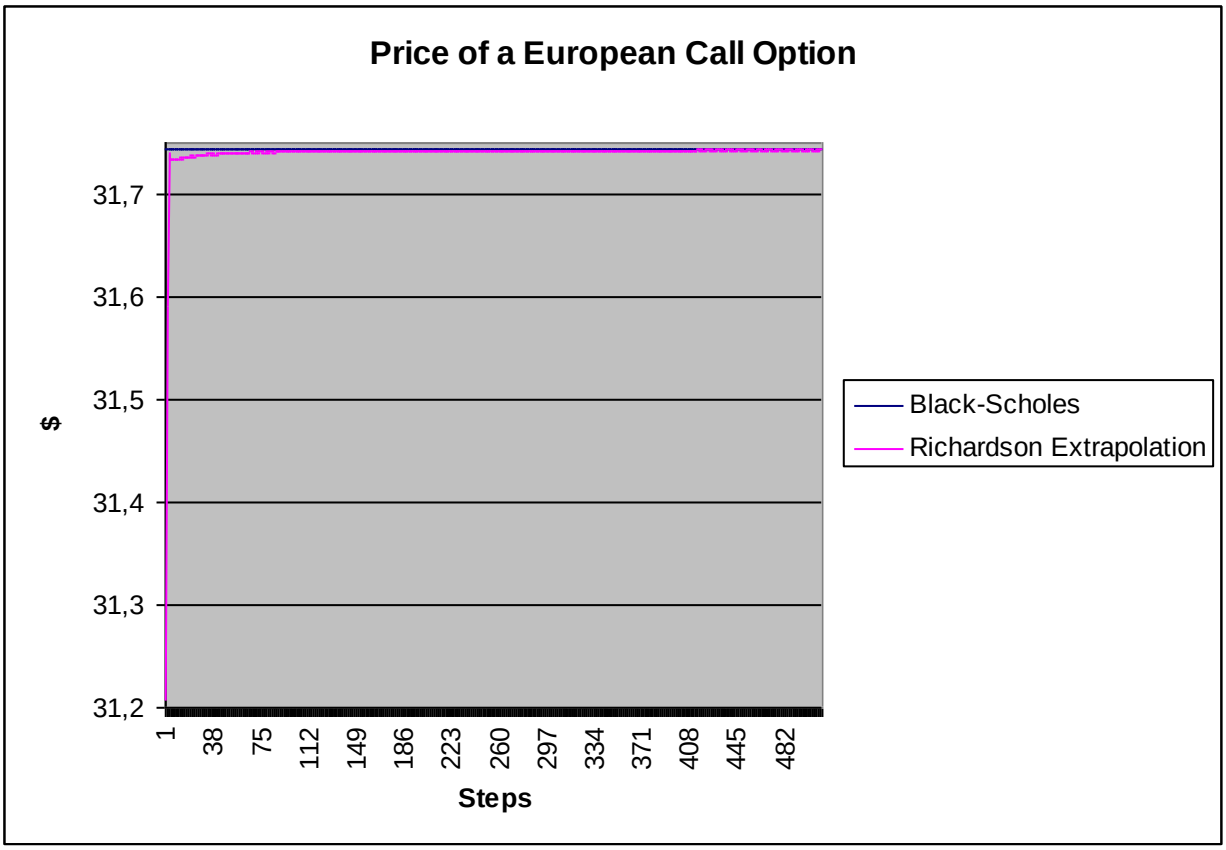


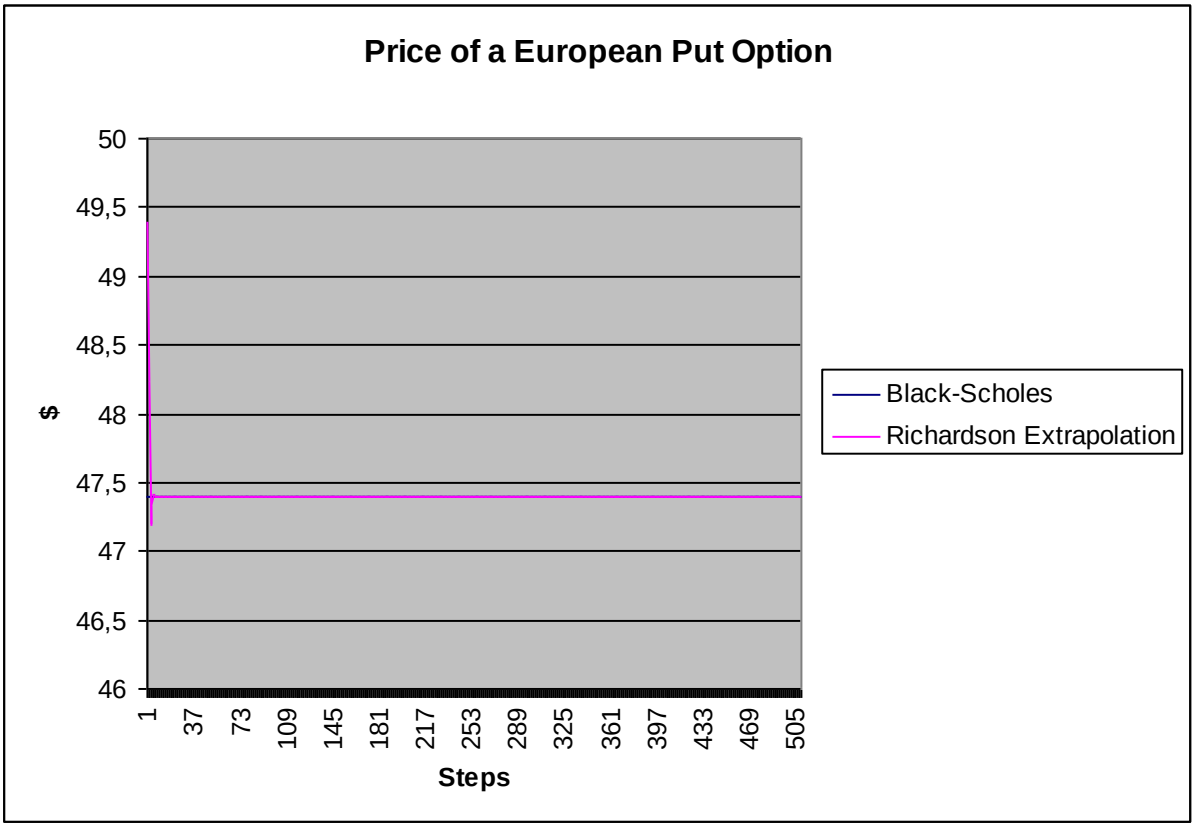


2.4 Richardson Extrapolation:

Now we have obtained a smooth price curve that converges as *n* gets larger. We will now use Richardson Extrapolation on the pegged price, and thereby increase the accuracy of the pricing. Using Richardson Extrapolation the results should be very accurate even for small *n*'s.

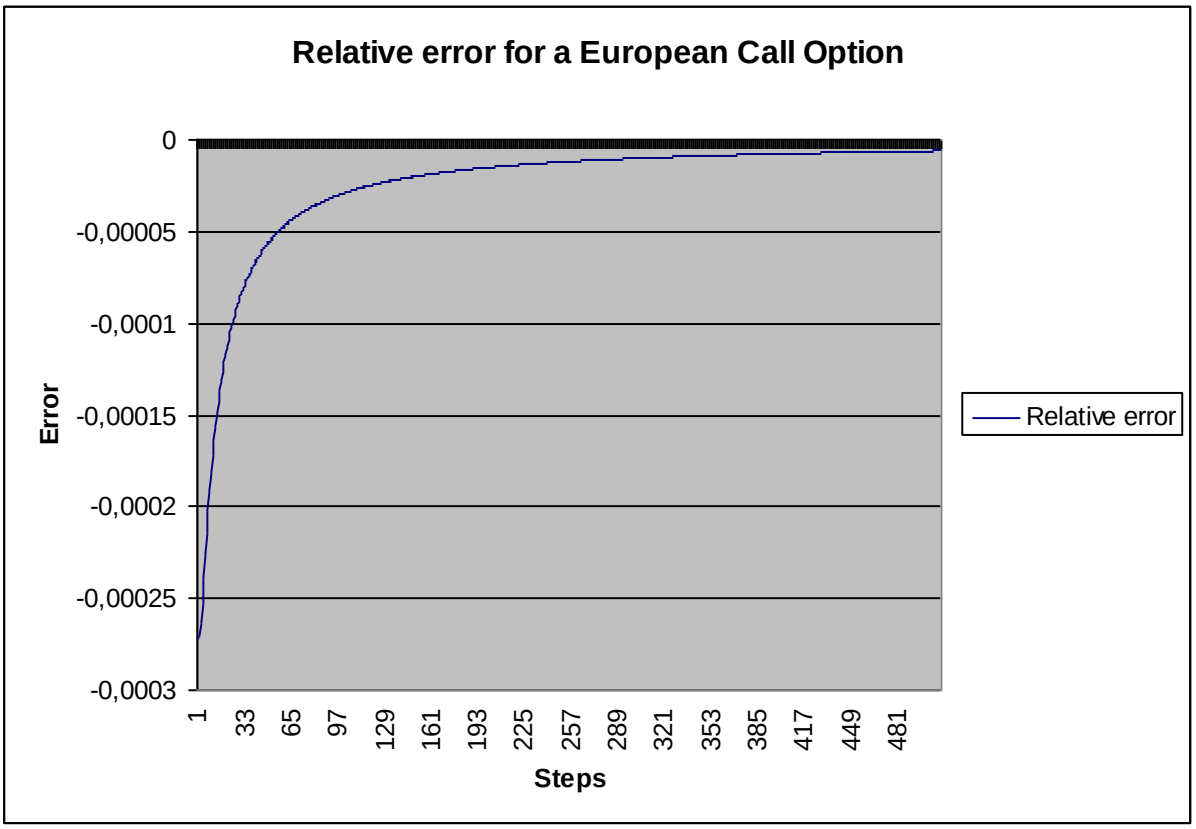
$$F_R = \frac{(h/2)^p F(h) - h^p F(h/2)}{(h/2)^p - h^p} = \frac{2^p F(h/2) - F(h)}{2^p - 1}$$

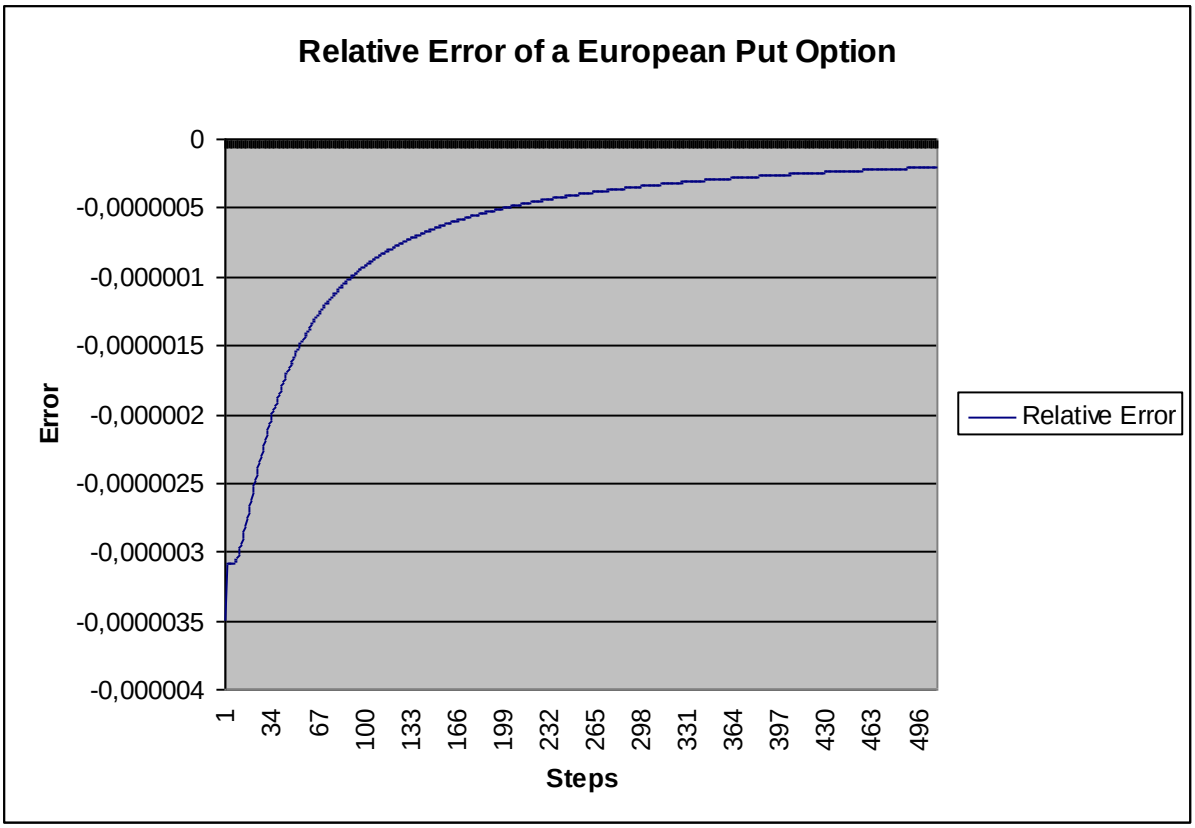

Price of a European Call Option
31,7
31,6
31,5
31,4
31,3
31,2
$
1
38
75
112
149
186
223
260
297
334
371
408
445
482
Steps
Black-Scholes
Richardson Extrapolation


Price of a European Put Option
50
49,5
49
48,5
48
47,5
47
46,5
46
$
1
37
73
109
145
181
217
253
289
325
361
397
433
469
505
Steps
Black-Scholes
Richardson Extrapolation

2.5 Relative error:

$$\mathrm{Re}\,lativeError = \frac{Peggedprice - Blackscholesprice}{Blackscholesprice}$$




Relative Error of a European Put Option
0
-0,0000005
-0,000001
-0,0000015
-0,000002
-0,0000025
-0,000003
-0,0000035
-0,000004
Error
1
34
67
100
133
166
199
232
265
298
331
364
397
430
463
496
Steps
Relative Error

3.0 Conclusion

4.0 References

Appendix: A – The VBA code

'Black-Scholes model code for a European Call Option

```
Function BlackScholes(Underlying, Strike, RiskFree, expTime, Volatility)

    d1 = (Log(Underlying / Strike) + RiskFree * expTime) / (Volatility * Sqr(expTime)) + 0.5
* Volatility * Sqr(expTime)

    BlackScholes = Underlying * Application.NormSDist(d1) - Strike * Exp(-expTime *
RiskFree) * Application.NormSDist(d1 - Volatility * Sqr(expTime))

End Function
```

'Black-Scholes model code for a European Put

```
Function ePut3(S, K, rf, T, sigma)

d1 = (Log(S / K) + rf * T) / (sigma * Sqr(T)) + 0.5 * sigma * Sqr(T)

P = S * Application.NormSDist(d1) - K * Exp(-T * rf) * Application.NormSDist(d1 - sigma *
Sqr(T))

ePut3 = K * Exp(-T * rf) - S + P
End Function
```

'Binomial model code for a European Call

```
Function eCall(S, K, T, rf, sigma, n)

    u = Exp(sigma * ((T / n) ^ 0.5))
    D = Exp(-sigma * ((T / n) ^ 0.5))
    r = Exp(rf * (T / n))

    rn_u = (r - D) / (r * (u - D))
    rn_d = 1 / r - rn_u

    eCall = 0

    For i = 0 To n

        eCall = eCall + Application.Combin(n, i) * rn_u ^ i * rn_d ^ (n - i) * Application.Max(S
* u ^ i * D ^ (n - i) - K, 0)

    Next i

End Function
```

'Binomial model code for a European Put

```
Function ePut(S, K, T, rf, sigma, n)
   u = Exp(sigma * ((T / n) ^ 0.5))
   D = Exp(-sigma * ((T / n) ^ 0.5))
   r = Exp(rf * (T / n))

   rn_u = (r - D) / (r * (u - D))
   rn_d = 1 / r - rn_u

   ePut = 0
   For i = 0 To n
      ePut = ePut + Application.Combin(n, i) * rn_u ^ i * rn_d ^ (n - i) * Application.Max(K -
(S * u ^ i * D ^ (n - i)), 0)
   Next i
End Function
```

'Pegged strike model for a European Call

```
Function eCall2(S, K, T, rf, sigma, n)

X = (Log(K / S)) / n
v = sigma * ((T / n) ^ 0.5)

u = Exp(X + v)
D = Exp(X - v)
r = Exp(rf * (T / n))

rn_u = (r - D) / (r * (u - D))
rn_d = 1 / r - rn_u

eCall2 = 0

   For i = 0 To n

      eCall2 = eCall2 + Application.Combin(n, i) * rn_u ^ i * rn_d ^ (n - i) *
Application.Max(S * u ^ i * D ^ (n - i) - K, 0)

   Next i

End Function
```

'Pegged strike model for a European Put

```
Function ePut2(S, K, T, rf, sigma, n)

X = (Log(K / S)) / n
v = sigma * ((T / n) ^ 0.5)

u = Exp(X + v)
D = Exp(X - v)

r = Exp(rf * (T / n))
```

```
rn_u = (r - D) / (r * (u - D))
rn_d = 1 / r - rn_u

ePut2 = 0
For i = 0 To n

    ePut2 = ePut2 + Application.Combin(n, i) * rn_u ^ i * rn_d ^ (n - i) * Application.Max(K -
(S * u ^ i * D ^ (n - i)), 0)

Next i

End Function
```

'Richardson extrapolation code for an European Put

```
Function ePut5(S, K, T, rf, sigma, n)
  If S < K Then
    ePut5 = ((4 * ePut2(S, K, T, rf, sigma, n / 2) - ePut2(S, K, T, rf, sigma, n)) / 3)
  End If
End Function
```

'Richardson extrapolation code for a European Call

```
Function eCall5(S, K, T, rf, sigma, n)
  If S > K Then
    eCall5 = ((4 * eCall2(S, K, T, rf, sigma, n / 2) - eCall2(S, K, T, rf, sigma, n)) / 3)
  End If
End Function
```