Interpolation, Numerical Differentiation, and Numerical Integration (II)

Ricard Santiago Raigada García

Date: 21/04/2023

Contents

1 Derivación

We want to calculate the delta and gamma of the option when $S_0 = 110$, $K = 108$, $r = 0.04$, $\sigma = 0.45$, $T = 0.1$, using numerical differentiation, and measure the error incurred. For this purpose, we will consider that the exact values of delta and gamma are given by the formulas in equation (3).

1.1 Valores exactos

Calculate, with the given values, the delta (δ) and gamma (γ) using the expression (3).

Solución:
The value of Δ is 0.5902566.
The value of Γ is 0.02483112.

To obtain the results, the code in R derived from the Black-Scholes model was executed:

$$\delta = \Phi(d_1)$$

Φ is calculated using the pnorm function in R, as specified in the statement. d_1 is determined by:

$$d_1 = \frac{In(S_0/K) + (r + 0.5\sigma^2)T}{\sigma\sqrt{T}}$$

For the exact calculation of Gamma, as stated in the problem, it will be done using:

$$\Gamma = \frac{\phi(d_1)}{\sigma\sqrt{T}S_0}$$

In Gamma, the previously calculated value of d_1 should be substituted into Delta. We then need to pass this value through the standard normal distribution. The mean (μ) is equal to the expected return and the standard deviation is equal to the volatility. The probability density function is derived from:

$$\phi = f(x|0,1) = \frac{1}{\sqrt{2\pi}} e^{(-0.5x^2)} \qquad -\infty < x < \infty$$

This is because gamma is a prediction of the variation in delta when there are changes in the price of the underlying asset. Delta indicates the change in the price of an option or premium in response to a change in the underlying, such as a stock.

The code used in this section is as follows:

```
#parametros de la call
K = 108
r = 0.04
sigma = 0.45
T = 0.1

#delta de la call (formula exacta)
BSdeltacall=function(S)
{
  d1 = (log(S/K)+(r + 0.5 * sigma ^2)*(T))/(sigma * sqrt(T))
  deltacall = pnorm(d1)
  return(deltacall)
}

#gamma de la call (formula exacta)
BSgammacall=function(S)
{
  d1 = (log(S/K)+(r + 0.5 * sigma ^2)*(T))/(sigma * sqrt(T))
  gamma = (dnorm (d1))/(S * sigma * sqrt(T))
  return(gamma)
}

#valor exacto de delta
vd=BSdeltacall(110)

#valor exacto de gamma
vg=BSgammacall(110)
```

Listing 1: R code

1.2 Derivación numérica

Calculate the value of delta and gamma using numerical differentiation (as detailed in Section 2.1 of (2)) with h = 0.1 and h = 0.001. Write the obtained results in the table below, where the absolute error and relative error are calculated using the exact values of delta and gamma that you calculated in the previous section.

Numerical differentiation				
h	Absolute error (δ)	Relative error (δ)	Absolute error (γ)	Relative error (γ)
0.1	0.001240573	0.002101753	6.840463e-08	2.754794e-06
0.001	1.241547e-05	2.103403e-05	6.959088e-09	2.802567e-07

The approximate values for delta with $h = 0.1$ are 0.5914971.
The approximate values for delta with $h = 0.001$ are 0.590269.
The approximate values for gamma with $h = 0.1$ are 0.02483106.
The approximate values for gamma with $h = 0.001$ are 0.02483113.

The calculation of the BScall function in R comes from:

$$v(S_0, \sigma, T, r, K) = S_0\Phi(d_1) - e^{-rT}K\Phi(d_2)$$

On the other hand, finddiff and finddif2 refer to the first and second derivative, respectively, representing Delta and Gamma.

First derivative:

$$f'(a) \approx \frac{f(a+h) - f(a)}{h}$$

Second derivative:

$$f''(a) \approx \frac{f(a+h) - 2f(a) + f(a-h)}{h^2}$$

In the calculation of the absolute error, I used the absolute value of the difference between the exact value and the approximate value. For the calculation of the relative error, I used the absolute error divided by the exact value.

For $h = 0.1$, $h = 0.01$, $h = 0.001$, and $h = 0.0001$, plot the Absolute Error for δ or γ. What can be observed?

It can be observed in the absolute error of delta, i.e., the first derivative, that as h becomes smaller, the error decreases, and therefore, the approximation becomes more accurate. However, this behavior is not observed in the second derivative, i.e., gamma. For gamma, it can be seen that the error decreases until a certain point, but beyond that point, the error starts to increase, and the function diverges.

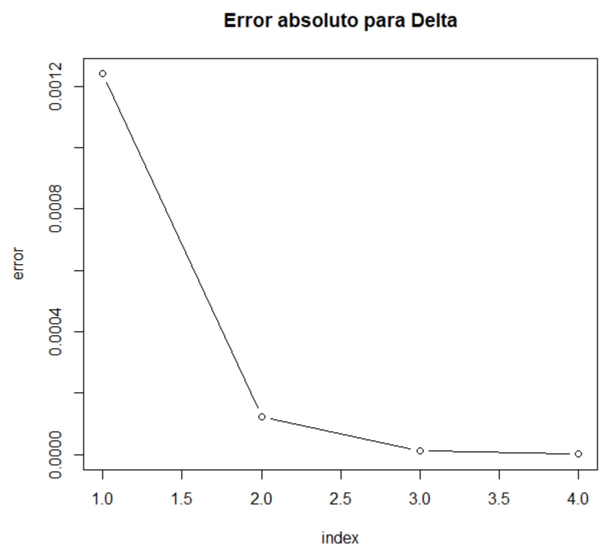

Figure 1: Absolut error for Delta

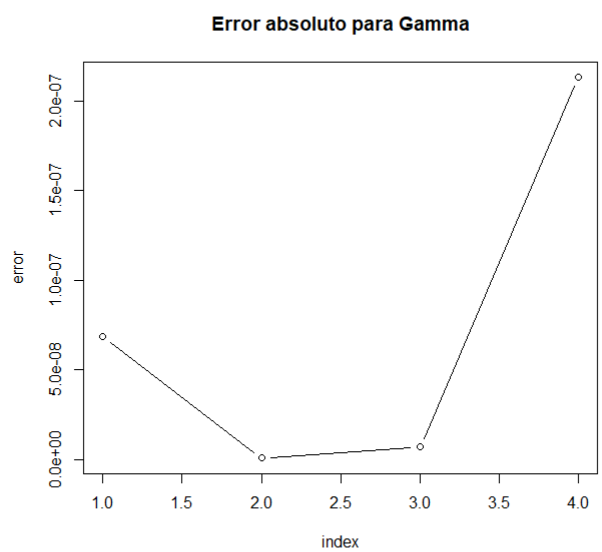

Figure 2: Absolut error for Gamma

The code used to solve this section is as follows:

```
#precio de la call
BScall=function(S)
{
  d1 = (log(S/K)+(r +0.5 * sigma ^2)*(T))/(sigma * sqrt(T))
  d2 = d1 - sigma * sqrt(T)
  call = S * pnorm (d1)-exp(-r*T) * K * pnorm (d2)
  return(call)
}

#primera derivada numerica (delta)
findiff=function(f,x,h)
{
  return((f(x+h)-f(x))/h)
}

#segunda derivada numerica (gamma)
findiff2=function(f,x,h)
{
  return((f(x+h)-2*f(x)+f(x-h))/(h^2))
}

#valor aproximado de la delta para h=0.1 y errores
vdapprox=findiff(BScall, 110, 0.1)
abserrdh1 = abs (vd - vdapprox)
relerrdh1 = abserrdh1/vd

#valor aproximado de la delta para h=0.001 y errores
vdapprox=findiff(BScall, 110, 0.001)
abserrdh2 = abs(vd - vdapprox)
relerrdh2 = abserrdh2 /vd

#valor aproximado de la gamma para h=0.1 y errores
vgapprox=findiff2(BScall, 110, 0.1)
abserrgh1 = abs (vg - vgapprox)
relerrgh1 = abserrgh1 /vg

#valor aproximado de la gamma para h=0.001 y errores
vgapprox=findiff2(BScall, 110, 0.001)
abserrgh2 = abs(vg - vgapprox)
relerrgh2 = abserrgh2 /vg

#representacion error absoluto para delta
vec_h=c(0.1,0.01,0.001,0.0001)
index=c(1,2,3,4)
vdapprox_vech=findiff(BScall, 110, vec_h)
error = abs(vd - vdapprox_vech)
plot(index,error,type="b", main="Error absoluto para Delta")

#representacion error absoluto para gamma
vec_h=c(0.1,0.01,0.001,0.0001)
index=c(1,2,3,4)
vdapprox_vech=findiff2(BScall, 110, vec_h)
error = abs(vg - vdapprox_vech)
plot(index,error,type="b", main="Error absoluto para Gamma")
```

Listing 2: R code

2 References

The references consulted are the same as mentioned in the statement. The theoretical material from UOC and the book "Computational Methods for Numerical Analysis with R" were used as references.