PEC 6 Interpolación, derivación e integración numérica

Alan Coila Bustinza

2022-04-18

1.Integración

Para realizar el cálculo de la opción necesitamos realizar la evaluación de las expresiones $\Phi(d1)$ y $\Phi(d2)$

1.1 Fórmula de Hastings

Ya que usaremos el metodo de Hastings para como referencia para valorar los errores, calcularemos primero la función Φ mediante esta fórmula. Se define por:

$$\widetilde{\Phi}(x) := 1 - \phi(x)(a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5)$$

para $x \geq 0$ y $\widetilde{\Phi}(x) := 1 - \widetilde{\Phi}(-x)$ para cuando $x < 0$. Dentro de la función, la variable t esta representada por la ecuación: $t = \frac{1}{1+\alpha x}$, el resto de los parámetros a_{1-5}, α, los incluiremos dentro de la función auxiliar en R

```
# primero designaremos los parámetros de la función de Hastings que provee el ejercicio
# creamos la función auxiliar te para el parámetro t
te=function(x)
{
  # asignamos el valor de alfa
  alpha=0.2316419
  return(1/(1+alpha*x))
}
# Creamos la función auxiliar phi que corresponde con la ecuación:
```

$$\phi(x) = \frac{1}{\sqrt{2pi}} e^{-\frac{x^2}{2}}$$

```
phi=function(x)
{
  return((1/sqrt(2*pi))*exp(-x^2/2))
}
# ahora creamos la función auxiliar que corresponde a la fórmula de Hastings
PhiHpos=function(x)
{
  # añadimos en las variables los valores de los parámetros proporciónados
  a1=0.319381530
  a2=-0.356563782
  a3=1.781477937
```

```
  a4=-1.821255978
  a5=1.330274429
  t=te(x)
  Phitilde=1-phi(x)*(a1*t+a2*t^2+a3*t^3+a4*t^4+a5*t^5)
  return(Phitilde)
}


# finalmente retornaremos el resultado segun el valor de x que usaremos posteriormente
PhiH=function(x)
{
  if(x>=0) return(PhiHpos(x))
  else return(1-PhiHpos(-x))
}
```

Ahora realizmos las aproximaciónes para el cálculo de la función $\phi(x)$ mediante los distintos metodos planteados

1.2 Regla de los Trapecios

Primero crearemos la función auxiliar trap

```
# esta función recibe 4 paramtros
# f: la función a la que se aplicará la regla
# a: límite inferior del area a aproximar
# b: límite superior del area a aproximar
# m: número de subintervalos qu usaremos
trap=function(f,a,b,m)
{
  # creamos un vector con los subintervalos entre a y b definidos por m paneles que
  # requerira m+1 evaluaciónes del integrando
  x=seq(a,b,length.out=m+1)
  # almacenamos en y los resultados de la función phi evaluada en todos lo x
  y=f(x)
  # realizamos la sumatoria de los valores de tal forma que los valores del medio se sumen
  # dos veces y solo una vez los valores extremos
  p.area=sum((y[2:(m+1)]+y[1:m]))
  # finalmente multiplicamos
  p.area=p.area*abs(b-a)/(2*m)
  return(p.area)
}
# evaluamos la función trap segun diferentes subintervalos
rT=c()
for(i in c(2,10,100,1000)){
    a= trap(phi,0,-1,i)
    rT=c(rT,a)
}
# tenemos los valores de la función trap para un x=1 y 2,10,100 y 100 subintervalos
# vemos que ya con 100 subintervalos nos da un resultado adecuado
rT
```

```
## [1] 0.3362609 0.3411430 0.3413427 0.3413447
```

Tenemos que tener en cuenta que para los metodos de integración por aproximación, el valor de x condicióna que para si:$x \geq 0$, $\Phi(x) = \frac{1}{2} + I(x)$, y si $x < 0$ $\Phi(x) = 1 - \Phi(-x)$, donde $I(x) = \int_0^x \phi(y)dy$.

```
PhiTpos=function(x)
{
  return(0.5+trap(phi,0,x,100))
}
PhiT=function(x)
{
  if(x>=0) return(PhiTpos(x))
  else return(1-PhiTpos(-x))
}
```

1.3 Metodo de Simpson

Actuaremos de forma similar que con el metodo de los trapecios, generaremos una función auxiliar que exprese el algortimo del metodo de simpson

```
# esta función recibe 4 paramtros
# f: la función a la que se aplicará la regla
# a: límite inferior del area a aproximar
# b: límite superior del area a aproximar
# m: número de subintervalos que usaremos
simp=function(f,a,b,m)
{
  # definiremos los puntos laterales y medio donde evaluaremos la función
  x.ends=seq(a,b,length.out=m+1)
  y.ends=f(x.ends)
  x.mids=(x.ends[2:(m+1)]-x.ends[1:m])/2+x.ends[1:m]
  y.mids=f(x.mids)
  # de tal forma que el area correspondera a la sumade 4 veces los valores intermedios,
  # dos veces los valores laterales y solo añadiremos
  # una vez cada valor extremo
  p.area=sum(y.ends[2:(m+1)]+4*y.mids [1: m ]+y.ends[1:m])
  p.area=p.area * abs(b - a) / (6 * m)
  return(p.area)
}
rS=c()
for(i in c(2,10,100,1000)){
    a= simp(phi,0,1,i)
    rS=c(rS,a)
}
# tenemos los valores de la función simp para un x=1 y 2,10,100 y 100 subintervalos
# vemos que ya con 10 subintervalos nos da un resultado adecuado
rS
```

```
## [1] 0.3413555 0.3413448 0.3413447 0.3413447
```

```
# Al igual que con la regla de los trapecios cumpliremos las condiciónes de los valores de x
PhiSpos=function(x)
{
  return(0.5+simp(phi,0,x,10))
```

```
}
PhiS=function(x)
{
  if(x>=0) return(PhiSpos(x))
  else return(1-PhiSpos(-x))
}
```

1.4 Metodo de Monte Carlo

```
# esta función recibe 4 paramtros
# f: la función a la que se aplicará el metodo
# a: límite inferior del area a aproximar
# b: límite superior del area a aproximar
# m: número de números aleatorios
mcint=function(f,a,b,m)
{
  # para generar los mismos números aleatorios utilizamos la siguiente linea de codigo
  set.seed(5)
  # con la función runif indicamos el valor máximo y minimo para u número m de valores aleatorios
  x=runif(m,min=a,max=b)
  y.hat=f(x)

  area=area <- (b - a) * sum(y.hat) / m
  return(area)
}
rMC=c()
for(i in c(2,10,100,200,400,1000,10000)){
    a= mcint(phi,0,1,i)
    rMC=c(rMC,a)
}
# tenemos los valores de la función simp para un x=1 y 2,10,100,200,300,400,500,1000
# números aleatorios
# vemos que ya con 400 números aleatorios nos da un resultado adecuado
# paradojicamente mayores valores nos dan peores resultados
rMC
```

```
## [1] 0.3532465 0.3344344 0.3375207 0.3390617 0.3415897 0.3408498 0.3410181
```

```
PhiMCpos=function(x)
{
  return(0.5+mcint(phi,0,x,400))
}

PhiMC=function(x)
{
  if(x>=0) return(PhiMCpos(x))
  else return(1-PhiMCpos(-x))
}
```

1.5 cálculo de la call

Ahora realizaremos el cálculo de la call , con cada función que hemos trabajado en la pec, recordamos que este cálculo viene dado por el metodo Black-Scholes :

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

```
# asignamos los parámetros de la opción
S=95
K=95
r=0.001
sigma=0.3
T=0.25
# creamos la función auxiliar para el cálculo de la opción
BScall=function(f)
{
  # recibira como argumento un función(un metodo aproximación a la integración)
  # que se aplicara a d1 y d2
  d1=(log(S/K)+(r+0.5*sigma^2)*(T))/(sigma*sqrt(T))
  d2=d1-sigma*sqrt(T)
  call=S*f(d1)-exp(-r*T)*K*f(d2)
  return(call)
}

vPhiH=BScall(PhiH) #valor con la Phi de Hastings
vPhiT=BScall(PhiT) #valor con la Phi de trapecios
vPhiS=BScall(PhiS) #valor con la Phi de Simpson
vPhiMC=BScall(PhiMC) #valor con la Phi de Monte Carlo
```

1.6 cálculo de los errores

Como se indicó inicialmente, usaremos la fórmula de Hastings como referente para calcular el error cometido por los otros metodos realizados

```
abserrT=abs(vPhiT-vPhiH)
relerrT=abserrT/vPhiH
abserrS=abs(vPhiS-vPhiH)
relerrS=abserrS/vPhiH
abserrMC=abs(vPhiMC-vPhiH)
relerrMC=abserrMC/vPhiH
```

metodo	Error absoluto	Error relativo
Trapecios	1.3503955×10^{-5}	2.3729503×10^{-6}
Simpson	1.3237844×10^{-5}	2.3261885×10^{-6}
Monte-Carlo	1.3870634×10^{-5}	2.437384×10^{-6}