PAC6. Interrpolació, derivació i integració numèrica (II)

Amelia Martínez Sequera

13/11/2021

Una opció financera de compra és un contracte entre dues parts que li dóna al seu posseïdor el dret de comprar un determinat actiu financer subjacent (per exemple una acció), el preu actual de la qual (és a dir, t=0) és S_0, en una data futura t=T (anomenat venciment) per un determinat preu K (preu exercici). Si arribat el venciment, el valor del subjacent en el mercat S_T és superior a K, llavors l´opció s´exerceix i s´adquireix l´actiu pagant K, en cas contrari l´opció no s´exerceix. Aquest és l´anomenat preu de liquidació de l´opció, que pot resumir-se matemàticament amb la fòrmula,

$$max(S_T - K, 0)$$

és a dir, el màxim entre $S_T - K$ i 0.

El valor d´aquest contracte, és a dir, el preu que hem de pagar per adquirir el dret d´exercici, es coneix com prima de l´opció i sota el supòsit que els preus des de l´instant t=0 fins a venciment t=T es mouen seguint un determinat model, conegut com model de Black-Scholes, ve donat per:

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

(1)

on

$$d_1 = \frac{ln(S_0/K) + (r + \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}}$$

$$d_2 = d_1 - \sigma\sqrt{T}$$

,

$\Phi(x) = \int_{-\infty}^{x} \phi\,(y)\,dy$ és la funció de distribució.

$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$ és la funció de densitat normal estàndard.

Els paràmetre r i σ són, respectivament, l´anomenat tipus d´interès lliure de risc i la volatilitat de l´actiu subjacent (que podem definir vagament com la variància de l´actiu subjacent).

A més del càlcul de la prima de l´opció, per a una adequada gestió del risc financer, els bancs calculen les denominades delta i gamma de l´opció,

$$\delta = v'(S)$$

(2)

$$\Gamma = v''(S)$$

és a dir, la primera i la segona derivada de v respecte del valor de l´actiu subjacent, respectivament. La delta ens indica com varia el preu de l´opció davant una variació del valor del subjacent, mentre que la gamma ens diu com varia la delta quan varia el preu de l´actiu subjacent. Si derivem v obtenim:

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

(3)

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

(3)

Recordem que a l´activitat anterior vàrem calcular el valor de la delta i la gamma de manera exacta i amb derivació numèrica. En aquesta activitat, posarem en pràctica el concepte d´integració numèrica.

1. Integració

El càlcul del valor de l´opció mitjançant (1), així com el càlcul del la delta en (2) implica l´avaluació de les expressions $\Phi(d_1)$ i $\Phi(d_2)$. Tenint en compte que no existeix una fòrmula exacta per Φ, aquests càlculs s´han de fer usant mètodes numèrics (la funció pnorm de R és en realitat una aproximació de Φ). Per a la propera tasca usarem com a aproximació per Φ l´anomenada fòrmula de Hastings:

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

(4)

per $x \geq 0$

i

$$\widetilde{\Phi}(x) := 1 - \widetilde{\Phi}(-x)$$

quan x < 0.

Els paràmetres venen donats per $t = \frac{1}{1+\alpha x}$; $\alpha = 0.2316419$; $\alpha_1 = 0.319381530$; $\alpha_2 = -0.356563782$; $\alpha_3 = 1.781477937$; $\alpha_4 = -1.821255978$; $\alpha_5 = 1.330274429$

L'error absolut per aquesta aproximació és inferior a 7.5e-8 per tot x, és a dir,

$$|\Phi(x) - \widetilde{\Phi}(x)| < 7.5e - 8$$

per a tot $x \in R$.

Volem explorar mètodes alternatius de càlcul de $\Phi(x)$ i l´impacte que tenen sobre el valor de l´opció en termes d´error absolut i error relatiu, assumint que el valor "exacte" de l´opció serà aquell que s´hagi calculat amb l´aproximació de $\Phi(x)$ donada per la fórmula de Hastings, és a dir, $\widetilde{\Phi}(x)$. Es demana calcular el valor de l´opció en els supòsits en els quals $\Phi(x)$ es calcula mitjançant els mètodes que es detallen en les seccions següents.

NOTA:

$$\Phi(x) = \frac{1}{2} + I(x)$$

si $x \geq 0$ i,

$$\Phi(x) = 1 - \Phi(-x)$$

si x<0,

on $I(x) = \int_0^x \phi\,(y)\,dy$

Fórmula de Hastings

```
#parametres opcio
S=105
K=110
r=0.005
sigma=0.4
T=0.5

#alternatives per calcular Phi
#formula de Hastings
te=function(x)
{
  alpha= 0.2316419
  return(1/(1+alpha*x))
}

phi=function(x)
{
  return((1/sqrt(2*pi))*exp(-x^2/2))
}

PhiHpos=function(x)
{
  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)
}

PhiH=function(x)
{
  if(x>=0) return(PhiHpos(x))
  else return(1-PhiHpos(-x))
}

blackScholes <- function(S, K, r, T, sigma) {
  d1 <- (log(S/K)+(r+0.5*sigma^2)*T)/(sigma*sqrt(T))
  d2 <- d1 - sigma * sqrt(T)

  call = S*PhiH(d1) - K*exp(-r*T)*PhiH(d2)
  return(call)
}

blackScholes(S, K, r, T, sigma)
```

```
## [1] 9.856575
```

1.1 Regla dels trapezis

Heu de donar el valor de l´opció amb $\Phi(x)$ calculada mitjançant la regla dels trapezis. Feu diversos experiments canviant el nombre de subintervals i compareu les diferències entre els valors obtinguts.

```
#regla dels trapezis
trap=function(f,a,b, m)
{
  x=seq(a,b,length.out=m+1)
  y=f(x)
  p.area=sum((y[2:(m+1)]+y[1:m]))
  p.area=p.area*abs(b-a)/(2*m)
  return(p.area)
}

PhiTpos=function(x)
{
  return(0.5+trap(phi,0,x,m))
}

PhiT=function(x)
{
  if(x>=0) return(PhiTpos(x))
  else return(1-PhiTpos(-x))
}
```

1.2 Regla de Simpson

Heu de donar el valor de l´opció amb $\Phi(x)$ calculada mitjançant la regla de Simpson. Feu diversos experiments canviant el nombre de subintervals i compareu les diferències entre els valors obtinguts.

```
#regla de Simpson
simp=function(f,a,b,m){

  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)
  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)
}

PhiSpos=function(x)
{
  return(0.5+simp(phi,0,x,m))
}

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

1.3 Extrapolació de Romberg

Heu de donar el valor de l´opció amb $\Phi(x)$ calculada mitjançant extrapolació de Romberg. Feu diversos experiments canviant el nombre de nivells i compareu les diferències entre els valors obtinguts.

```
#Romberg (si tab=TRUE torna tota la matriu)
romberg=function(f,a,b,m,tab=FALSE){
  R=matrix(NA,nrow=m,ncol=m)
  R[1,1]=trap(f,a,b,m=1)
  for(j in 2:m){
    R[j,1]=trap(f,a,b,m=2^(j-1))
    for (k in 2:j){
      k4=4^(k-1)
      R[j,k]=k4*R[j,k-1]-R[j-1,k-1]
      R[j,k]=R[j,k]/(k4-1)
    }
  }
  if (tab==TRUE)
    return (R)
  return (R[m,m])
```

```
}

PhiRompos=function(x)
{
  return(0.5+romberg(phi,0,x,m,FALSE))
}

PhiRom=function(x)
{
  if(x>=0) return(PhiRompos(x))
  else return(1-PhiRompos(-x))
}
```

1.4 Mètode de Monte Carlo

Heu de donar el valor de l´opció amb $\Phi(x)$ calculada mitjançant el mètode de Monte Carlo fent variar la grandària de la mostra generada per simulació.

```
#Monte Carlo
mcint=function(f,a,b,m){
  set.seed(1255)
  x=runif(m,min=a,max=b)
  y.hat=f(x)
  area= (b-a)*sum(y.hat)/m
  return(area)
}

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

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

Fem servir una mostra de grandària 100:

```
m=100
#preu de la call
BScall=function(S, K, r, T, sigma) {
  d1=(log(S/K)+(r+0.5*sigma^2)*(T))/(sigma*sqrt(T))
  d2=d1-sigma*sqrt(T)
  call=  S*PhiMC(d1) - K*exp(-r*T)*PhiMC(d2)
  return(call)
}
```

```
MC100 <-BScall(S, K, r, T, sigma)
MC100

## [1] 9.829278
```

Fem servir una mostra de grandària 1000:

```
m=1000
#preu de la call
BScall=function(S, K, r, T, sigma) {
  d1=(log(S/K)+(r+0.5*sigma^2)*(T))/(sigma*sqrt(T))
  d2=d1-sigma*sqrt(T)
  call=  S*PhiMC(d1) - K*exp(-r*T)*PhiMC(d2)
  return(call)
}

MC1000 <- BScall(S, K, r, T, sigma)
MC1000

## [1] 9.850335
```

I ara fem servir una mostra de grandària 10000:

```
m=10000
#preu de la call
BScall=function(S, K, r, T, sigma) {
  d1=(log(S/K)+(r+0.5*sigma^2)*(T))/(sigma*sqrt(T))
  d2=d1-sigma*sqrt(T)
  call=  S*PhiMC(d1) - K*exp(-r*T)*PhiMC(d2)
  return(call)
}

MC <- BScall(S, K, r, T, sigma)
MC

## [1] 9.855543
```

Veiem que els resultats milloren a mida que augmenta la grandària de la mostra. Això és perquè aquest algorisme fa servir els m punts aleatoris, uniformement distribuits pel domini d´integració, per calcular l´àrea sota la corba: si el punt cau sota la linia de la funció, es considera dins de l´àrea d´integració, si cau per sobre no. La integral pot aproximar-se fent la mitjana de les mostres de la funció f avaluada en aquests punts.

1.5 Errors absoluts i relatius

Finalment, escriu els errors absoluts i relatius comesos en calcular el valor de l'opció mitjançant els diferents mètodes d'integració per $\Phi(x)$ i expressa els resultats en una (o diverses) taula(s) resum.

Primer farem servir m=5:

```r
m=5
#preu de la call
BScall=function(x){
  d1=(log(S/K)+(r+0.5*sigma^2)*(T))/(sigma*sqrt(T))
  d2=d1-sigma*sqrt(T)
  call=  S*x(d1) - K*exp(-r*T)*x(d2)
  return(call)
}

vPhiH=BScall(PhiH) #valor amb la Phi de Hastings
vPhiT=BScall(PhiT) #valor amb la Phi de trapezis
vPhiS=BScall(PhiS) #valor amb la Phi de Simpson
vPhiRom=BScall(PhiRom) #valor amb la Phi de Romberg
vPhiMC=BScall(PhiMC) #valor amb la Phi de Monte Carlo

#errors
abserrT=abs(vPhiT-vPhiH)
relerrT=abserrT/vPhiH

abserrS=abs(vPhiS-vPhiH)
relerrS=abserrS/vPhiH

abserrRom=abs(vPhiRom-vPhiH)
relerrRom=abserrRom/vPhiH

abserrMC=abs(vPhiMC-vPhiH)
relerrMC=abserrMC/vPhiH
```

```
##        Mètode    valor  e.rel  e.ab
## 1    Hastings 9.856575 0.0000 0.000
## 2    Trapezis 9.855671 0.0001 0.001
## 3     Simpson 9.856586 0.0000 0.000
## 4    Roomberg 9.856586 0.0000 0.000
## 5 Monte Carlo 9.899358 0.0043 0.043
```

Ara m=10

```
##        Mètode    valor  e.rel  e.ab
## 1    Hastings 9.856575 0.0000 0.000
## 2    Trapezis 9.855671 0.0001 0.001
## 3     Simpson 9.856586 0.0000 0.000
## 4    Roomberg 9.856586 0.0000 0.000
## 5 Monte Carlo 9.899358 0.0043 0.043
```

Amb m=5 o m=10 subdivisions o nivells, els mètodes de Simpson i Roomberg aconsegueixen bones aproximacions. La regla dels trapezis també, però fins i tot millora si augmentem les subdivisions a 20, ja que l´error és inversament proporcional a m.

Com hem vist abans, el mètode de Monte Carlo necessita de l´ordre de 10000 punts generats aleatòriament per aconseguir un bon resultat. Això implica un alt cost computacional, però és útil per tractar integrals múltiples.

Per m=20 i 10000 punts a Monte Carlo:

```
##        Mètode    valor e.rel  e.ab
## 1    Hastings 9.856575 0e+00 0.000
## 2    Trapezis 9.856357 0e+00 0.000
## 3     Simpson 9.856586 0e+00 0.000
## 4    Roomberg 9.856586 0e+00 0.000
## 5 Monte Carlo 9.886036 3e-03 0.029
## 6    MC 10000 9.855543 1e-04 0.001
```