# Taller de Simulación

**Práctico 5**
**Métodos de Integración Numérica**

## 1. Ejercicio 1-Ejemplo para regla del trapecio

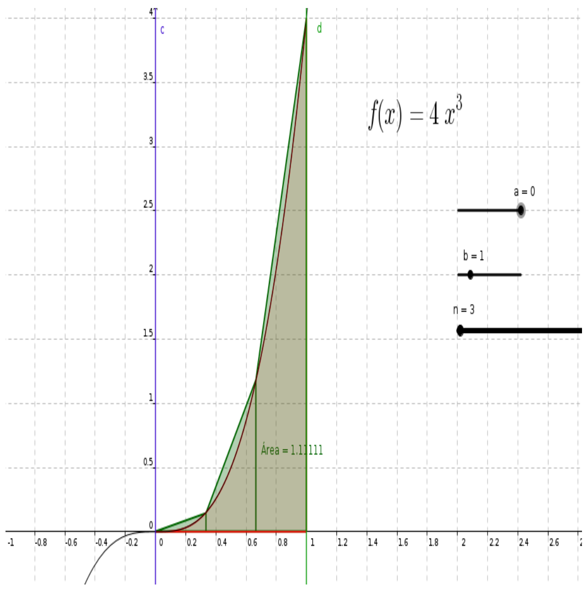

Figura 1: Regla del Trapecio

```
# program spuRs/resources/scripts/trapezoid.r

trapezoid <- function(ftn, a, b, n = 100) {
# numerical integral of ftn from a to b
# using the trapezoid rule with n subdivisions
#
# ftn is a function of a single variable
# we assume a < b and n is a positive integer

h <- (b-a)/n
x.vec <- seq(a, b, by = h)
f.vec <- sapply(x.vec, ftn)
T <- h*(f.vec[1]/2 + sum(f.vec[2:n]) + f.vec[n+1]/2)
return(T)
}

ftn6<-function(x) return(4*x**3)

primitiva<-function(x) return(x**4)

a<-0
b<-1
n<-4
```

```
h<-(a+b)/n
h0<-c(a,0)
h1<-seq(a,b,h)
rango<-seq(a,2*b,0.001)
plot(rango,(ftn6(rango)),type="l",xlim=c(a,b+1),ylim=c(a,ftn6(b)+1),ylab="f(x)")
abline(v=b,col=2)

for (i in 1:n ){
segments(h1[i],ftn6(h1[i]),h1[i+1],ftn6(h1[i+1]))
abline(v=h1[i],col=3,lty=2)
}
trapezoid(ftn6,0,1,n)
(Error=trapezoid(ftn6,0,1,n)-(primitiva(b)-primitiva(a)))
legend(b*(1.1),ftn6(b)+1,paste("Error",Error))
```

Se analiza la convergencia del método

## 2. Ejercicio 2-Regla de Simpson

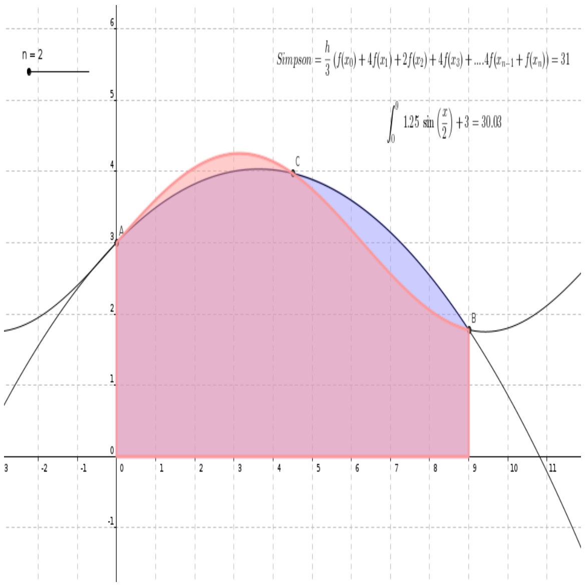

Figura 2: Regla de Simpson

```
#program spuRs/resources/scripts/simpson_n.r

simpson_n <- function(ftn, a, b, n = 100) {
# numerical integral of ftn from a to b
# using Simpson’s rule with n subdivisions
#
# ftn is a function of a single variable
# we assume a < b and n is a positive even integer
n <- max(c(2*(n %/% 2), 4))
h <- (b-a)/n
x.vec1 <- seq(a+h, b-h, by = 2*h)
x.vec2 <- seq(a+2*h, b-2*h, by = 2*h)
f.vec1 <- sapply(x.vec1, ftn)
f.vec2 <- sapply(x.vec2, ftn)
S <- h/3*(ftn(a) + ftn(b) + 4*sum(f.vec1) + 2*sum(f.vec2))
return(S)
}


ftn6<-function(x) return(4*x**3)

#PARA COMPARAR GRAFICAMENTE  SIMPSON CON GEOGEBRA
a<-0
b<-9
n<-4
ftn7<-function(x) return(1.25*sin(x/2)+3)

simpson_n(ftn7,a,b,n)
```

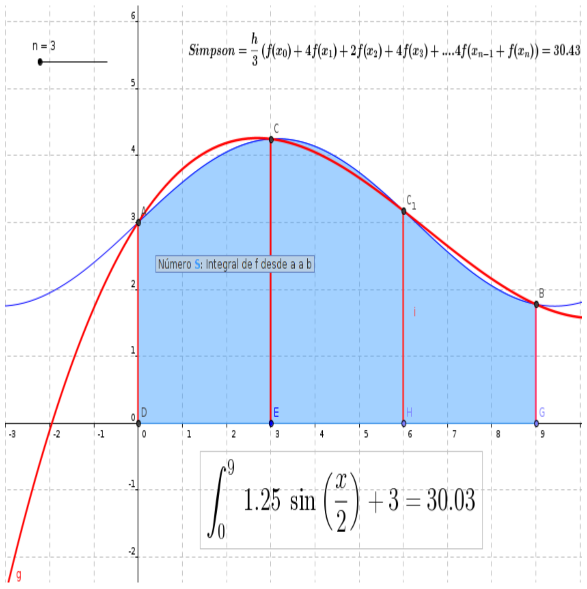

Figura 3: Regla de Simpson n=4

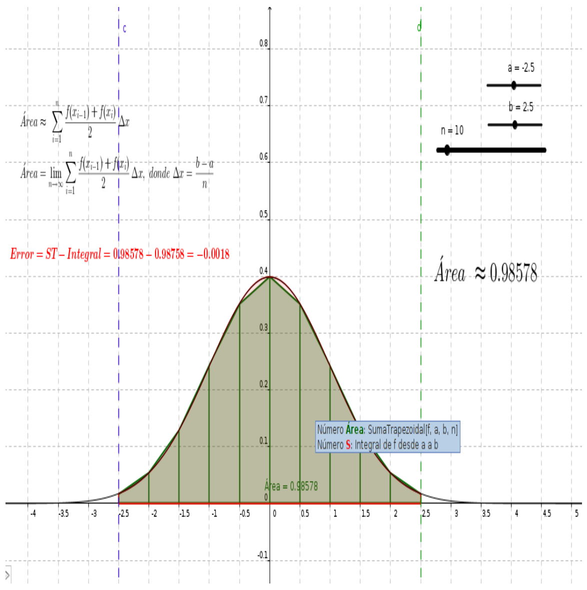

Figura 4: Regla del Trapecio para curva Normal

## 3. Ejercicio 3-Regla de Simpson Tolerancia prefijada

```
# program spuRs/resources/scripts/simpson.r
simpson <- function(ftn, a, b, tol = 1e-8, verbose = FALSE) {
# numerical integral of ftn from a to b
# using Simpson’s rule with tolerance tol
#
# ftn is a function of a single variable and a < b
# if verbose is TRUE then n is printed to the screen
# initialise
n <- 4
h <- (b - a)/4
fx <- sapply(seq(a, b, by = h), ftn)
S <- sum(fx*c(1, 4, 2, 4, 1))*h/3
S.diff <- tol + 1  # ensures we loop at least once

# increase n until S changes by less than tol
while (S.diff > tol) {
# cat(’n =’, n, ’S =’, S, ’\n’)  # diagnostic
S.old <- S
n <- 2*n
h <- h/2
fx[seq(1, n+1, by = 2)] <- fx  # reuse old ftn values
fx[seq(2, n, by = 2)] <- sapply(seq(a+h, b-h, by = 2*h), ftn)
S <- h/3*(fx[1] + fx[n+1] + 4*sum(fx[seq(2, n, by = 2)]) +
2*sum(fx[seq(3, n-1, by = 2)]))
S.diff <- abs(S - S.old)
}
if (verbose) cat(’partition size’, n, ’\n’)
return(S)
}
```

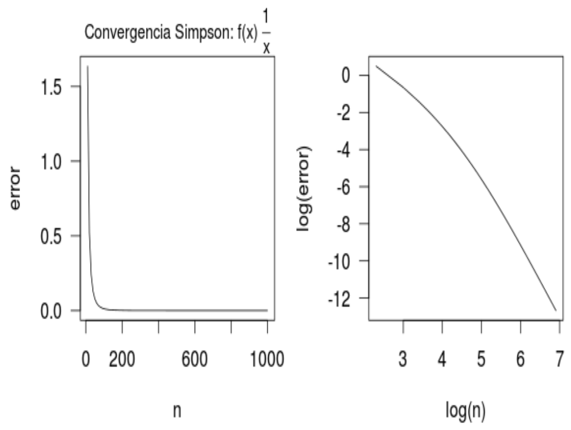

Figura 5: Velocidad de Convergencia para Regla del Simpson

```
# program simpson_test.r
# test the accuracy of Simpson's rule
# using the integral of 1/x from 0.01 to 1
rm(list = ls()) # clear the workspace
source("../scripts/simpson_n.r")
ftn <- function(x) return(1/x)
S <- function(n) simpson_n(ftn, 0.01, 1, n)
n.vec <- seq(10, 1000, by = 10)
S.vec <- sapply(n.vec, S)
opar <- par(mfrow = c(1, 2), pty="s", mar=c(4,4,2,1), las=1)
plot(n.vec, S.vec + log(0.01), type = "l",
xlab = "n", ylab = "error",cex.main=0.9,
main=expression(paste("Convergencia Simpson:
f(x) ",frac(1,x) )))
plot(log(n.vec), log(S.vec + log(0.01)), type = "l",
xlab = "log(n)", ylab = "log(error)")
par(opar)
```

## 4. Ejercicio 4-Análisis de la curva Normal

```
# program spuRs/resources/scripts/Phi.r
# estimate and plot the normal cdf Phi

rm(list = ls()) # clear the workspace
source("../scripts/simpson_n.r")
phi <- function(x) return(exp(-x^2/2)/sqrt(2*pi))
Phi <- function(z) {
if (z < 0) {
return(0.5 - simpson_n(phi, z, 0,n))
} else {
```

```
return(0.5 + simpson_n(phi, 0, z,n))
}
}

a<--5
b<-5
z <- seq(a,b, by = 0.1)
n<-50
phi.z <- sapply(z, phi)
Phi.z <- sapply(z, Phi)
plot(z, Phi.z, type = "l", ylab = "",col=2,
main = expression(paste("Densidad: ",
phi(z), " Distribucion: ",Phi(z))))
lines(z, phi.z,col=4)
legend("topleft",paste("Distribucion aproximada \n Regla Simpson",n))

Phi.z_2<-Phi.z
Phi.z_10<-Phi.z
Phi.z_50<-Phi.z
plot(z,Phi.z_2-Phi.z_10,type="l")
abline(h=0)
abline(v=0)
plot(z,Phi.z_10-Phi.z_50,type="b")

plot(Phi.z_2,Phi.z_10)
```

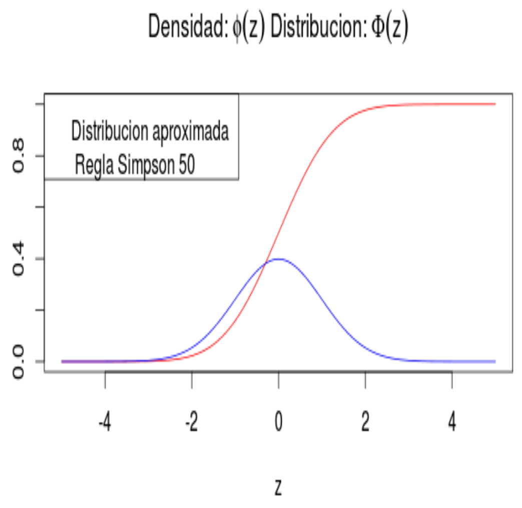

Figura 6: Aproximación Curva Normal

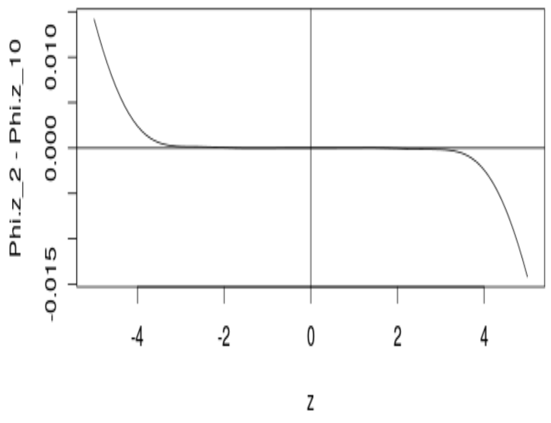

Figura 7: Aproximación Curva Normal:Error para 2 y 10 iteraciones

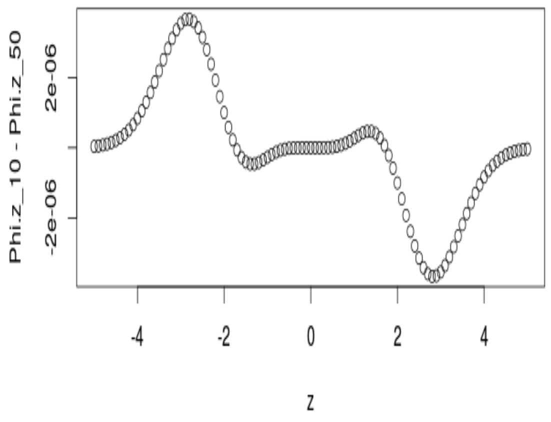

Figura 8: Aproximación Curva Normal: Error para 10 y 50 iteraciones

## 5. Ejercicio 5-Cuadratura adaptativa

```
options(digits = 16)



a<-0;b<-1
k<-4

f4<-function(x) 5*x**4
simpson(f4,a,b,tol=1e-9,verbose = TRUE)
partition size 512
[1] 1.000000000009701


k<-8
f8<-function(x) (k+1)*x**(k)
simpson(f8,a,b,tol=1e-9,verbose = TRUE)
partition size 1024
[1] 1.000000000015279

k<-12
f12<-function(x) (k+1)*x**(k)
simpson(f12,a,b,tol=1e-9,verbose = TRUE)
partition size 2048
[1] 1.000000000005419


#ahora cambiamos la forma de hacer la integracion, teniendo en cuenta que la funcion
#  (k+1)*x**(k) es mucho mas empinada en el intervalo [0.5,1] que en [0,0.5]

# por ejemplo hacerlo separada (adaptativo) es mejor que hacerlo en on intervalo [0,1]

S1<-simpson(f12,0,0.5,tol=5e-10,verbose = TRUE)
S2<-simpson(f12,0.5,1,tol=5e-10,verbose = TRUE)

S1+S2


#ejemplo para
ftn<-function(x) return(1.5*sqrt(x))
simpson(ftn,a,b,tol=0.001,verbose = TRUE)
simpson1<-simpson(ftn,a,b,tol=1e-9,verbose = FALSE)
(tiempo.simpson1<-system.time(simpson(ftn,a,b,tol=1e-9,verbose = FALSE)))
cuadratura1<-quadrature(ftn,a,b,tol=1e-9,trace=FALSE)
(tiempo.cuadratura1<-system.time(quadrature(ftn,a,b,tol=1e-9,trace=FALSE)))
cuadratura1[1]-simpson1
```

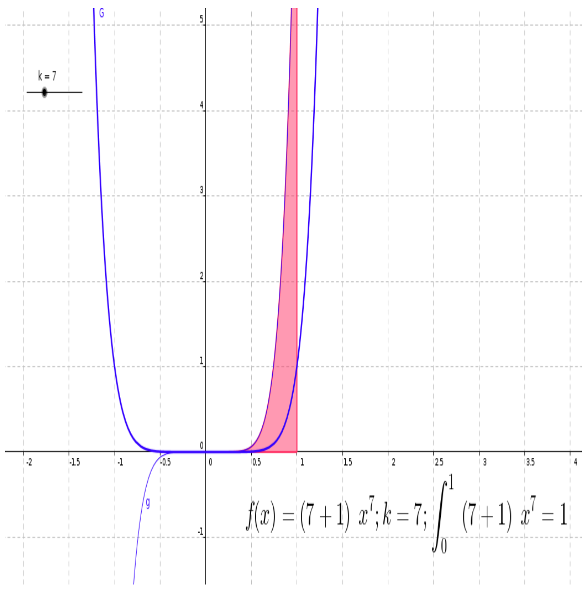

Figura 9: Cuadratura adaptativa

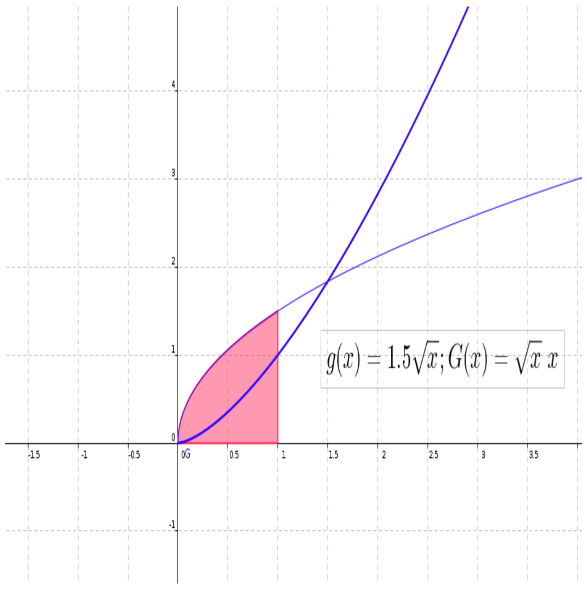

Figura 10: Cuadratura adaptativa