Taller Integración Númerica

Gabriel Gómez, Juan Pablo Méndez, Simón Dávila

Mayo 10 2020

1 Desarrollo De Ejercicios

a) Teniendo en cuenta que en la regla de los trapecios, estime el número mínimo de trapecios para aproximar la integral de f(x) = sen2x dx, con una tolerancia de 0.0001.

b) Dados los siguientes puntos: (0.1,1.8) , (0.2,2.6) , (0.3,3.0) , (0.4,2.8), (0.5,1.9); Utilice la fórmula de Simpson para encontrar una aproximación del área bajo la curva y calculé su error. -Qué resultado se obtendría si utilizamos la regla del trapecio -Utilice la siguiente metodología: a. Primero interpole y encuentre f(x); b. Integre utilizando un método analítico.

Desarrollo de la integral por Simpson con su error:

```
x<-c(0.1,0.2,0.3,0.4,0.5)
y<-c(1.8,2.6,3.0,2.8,1.9)

mat <- matrix(c(0,0,0,0,0,0), nrow=1,ncol = 5, byrow=TRUE)

options(digits = 16)
vec<-c(0)

for(i in 1:length(x)){

  for(j in 0:length(x)){

    vec[j+1]=x[i]^j


  }
  mat<-rbind(mat,c(vec))
}

mat <- mat[1:length(x)+1,]
```

```
coef_pol <- (solve(mat,y))     #el solve solo sirve para matrices cuadradas

cat("Resultado Polinomio: f(x)", coef_pol[5],"x^4 +",coef_pol[4],"X^3 +",coef_pol[3],"X^

f<-function(x){
  return( coef_pol[5]*x^4 +coef_pol[4]*x^3 +coef_pol[3]*x^2 +coef_pol[2]*x +coef_pol[1])
}

simpson <- function(vector_x){

  #h=0.1

  cont=1

  secuencia_simpson=0
  cont=1
  const=4

  for(i in 1:length(vector_x)){

    if(cont==1){
      secuencia_simpson=secuencia_simpson +f(vector_x[cont])
    }
    if(cont > 1 && cont < length(vector_x)){

      if(const==4){
        secuencia_simpson=secuencia_simpson + 4*f(vector_x[cont])
        const=2
      }else{
        secuencia_simpson=secuencia_simpson + 2*f(vector_x[cont])
        const=4
      }
    }
    if(cont==length(vector_x)){
      secuencia_simpson=secuencia_simpson +f(vector_x[cont])
    }
    cont=cont+1
  }

  resultado=secuencia_simpson

  cat("Valor de la integral: ",resultado)
```

```
  return(resultado)
}

valorAprox=simpson(x)
integrate(f,1.6,2)
valorReal=27.44311111111179
errorRelativo= abs((valorReal-valorAprox)/valorAprox)
cat("Error relativo: ",errorRelativo)
```

Desarrollo de la integral por el metodo del trapecio:

```
    trapecio<- function (a,b){

  h = b-a

  trap = h*((f(b)+f(a))/2)

  return(trap)
}

trapecio (1.6,2)
```

c) Con la fórmula de Simpson integrar iterativamente hasta que el error de truncamiento sea menor de 0.0001.

```
options(digits = 16)

f<-function(x){
  return( sqrt(1+cos(x)*cos(x)))
}


simpson <- function(a,b,n){

  h = (b-a)/n

  xn<-c(0)
  cont=1
  xn[cont]=0
  cont=cont+1

  for (i in 1:n) {
    xn[cont]= xn[cont-1]+h
    cont=cont+1
  }
```

```
  secuencia_simpson=0
  cont=1
  const=4

  for(i in 1:length(xn)){

    if(cont==1){
      secuencia_simpson=secuencia_simpson +f(xn[cont])
    }
    if(cont > 1 && cont < length(xn)){

      if(const==4){
        secuencia_simpson=secuencia_simpson + 4*f(xn[cont])
        const=2
      }else{
        secuencia_simpson=secuencia_simpson + 2*f(xn[cont])
        const=4
      }
    }
    if(cont==length(xn)){
      secuencia_simpson=secuencia_simpson +f(xn[cont])
    }
    cont=cont+1
  }

  valor = (b-a)/(3*n)
  resultado=valor*secuencia_simpson

  cat("Valor de la integral: ",resultado)



}

simpson(0,2,4)
```

d)Utilice la fórmula de la cuadratura de Gauss para aproximar la integral definida [1,2] para f(x)= xe(exp=x) dx.

```
options(digits = 16)

f<- function(t){
  return(t*exp(t))
```

```
}

cuadratura_gauss<-function(){

  m=2 # para encontrar un polinomio de grado <= 2

  xk = sqrt(1/3)   # Encontrar los xn despejando x en los polinomios de Legendre

  #Para encontrar los pesos Ck, como es de grado dos basta con 2 puntos, es decir 2 Xn y

  c1=(2*(1-(xk)^2))/(m^2*(xk)^2)

  c2=(2*(1-(-xk)^2))/(m^2*(-xk)^2)

  ck<-c(c1,c2)

  #Para generar el cambio de variable de t -> x de [1,2] a [-1,1] se tiene en cuenta
  #la formula (t-a)/(b-a) , resolviendo dicha fomrula obtenemos t = (x+1)/2 +1
  # y reemplazamos t en la funcion original, de esta manera quedaria la nueva funcion:

  f_nueva<- function(x){

    return(((x+1)/2)+1*exp((x+1)/2)+1)
  }

  cont=1
  sumatoria_gauss=0

  for (i in 1:length(ck)) {

    sumatoria_gauss=sumatoria_gauss+(ck[cont]*f_nueva(xk))
    cont=cont+1

  }


  cat("Solucion de integral por cuadrantes de Gauss: ",sumatoria_gauss)

}

cuadratura_gauss()
```

e)Utilice la misma fórmula de cuadratura de Gauss, pero particione la integral de la siguiente manera: en un intervalo de [1,1.5] y [1.5,2], mejoró el resultado?

```
options(digits = 16)

cuadratura_gauss_partida<-function(){

  m=2 # para encontrar un polinomio de grado <= 2

  xk = sqrt(1/3)   # Encontrar los xn despejando x en los polinomios de Legendre

  #Para encontrar los pesos Ck, como es de grado dos basta con 2 puntos, es decir 2 Xn y

  c1=(2*(1-(xk)^2))/(m^2*(xk)^2)

  c2=(2*(1-(-xk)^2))/(m^2*(-xk)^2)

  ck<-c(c1,c2)

  #Para generar el cambio de variable de t -> x de [1,1.5] a [-1,1] se tiene en cuenta
  #la formula (t-a)/(b-a) , resolviendo dicha fomrula obtenemos t = (0.5x+0.5)/2 +1
  # y reemplazamos t en la funcion original, de esta manera quedaria la nueva funcion:

  f_nueva<- function(x){

    return(((0.5*x+0.5)/2)+1*exp((0.5*x+0.5)/2)+1)
  }

  cont=1
  sumatoria_gauss=0

  for (i in 1:length(ck)) {

    sumatoria_gauss=sumatoria_gauss+(ck[cont]*f_nueva(xk))
    cont=cont+1

  }

  integral_1=sumatoria_gauss*(1/2)
  cat("integral 1: ",integral_1)

  #Para generar el cambio de variable de t -> x de [1.5,2] a [-1,1] se tiene en cuenta
  #la formula (t-a)/(b-a) , resolviendo dicha fomrula obtenemos t = (0.5x+0.5)/2 +1.5
  # y reemplazamos t en la funcion original, de esta manera quedaria la nueva funcion:
```

```
  f_nueva<- function(x){

    return(((0.5*x+0.5)/2)+1.5*exp((0.5*x+0.5)/2)+1.5)
  }

  cont=1
  sumatoria_gauss=0

  for (i in 1:length(ck)) {

    sumatoria_gauss=sumatoria_gauss+(ck[cont]*f_nueva(xk))
    cont=cont+1

  }

  integral_2=sumatoria_gauss*(1/2)
  cat("integral 2: ",integral_2)

  cat("\n")
  cat("Sumatoria de las dos particiones de integrales: ",integral_1+integral_2)

}

cuadratura_gauss_partida()
```