moodle_02_02_15_11

Alan Coila Bustinza

2022-06-02

```r
library(knitr)     # For knitting document and include_graphics function
library(ggplot2)   # For plotting
library('png')
```

pregunta 1

```r
img1_path <- "p1_2022-06-02_140531.png"
include_graphics(img1_path)
```

La tabla de diferencias divididas de Newton para una cierta función es la siguiente:

x	$f(x)$			
0.0	1.0000			
		1.107		
0.2	1.2214		β	
		1.352		$\frac{0.68125}{3}$
0.4	1.4918		0.74875	
		α		
0.6	1.8221			

```r
x <- c(0,0.2,0.4,0.6)
y <- c(1,1.2214,1.4918,1.8221)

dif_n <- function(x,y){
  k=length(x)
  nk <- function(x,y){
    mat <- matrix(0, length(x), length(x))
    mat[,1] <- y
    return (mat)
  }
  mat <- nk(x,y)
```

```r
  val=1
  for(i in 1:k){
    vect=c()
    for(j in 1:(k-i)){
      if(j>0){
        a=mat[j+1,i]
        b=mat[j,i]
        vect <- c(vect ,(a-b)/(x[j+val]-x[j]))

      }
    }
    val=val+1
    if(i<k){
      mat[,i+1] <- vect[1:k]
    }
  }

return(list(mat,k-1))
}
 # retorna la matriz de diferencias divididas y el grado del polinomio
# cuidado con los resultados 0 q cambian el grado del polinomio!!!!!
dif_n(x,y)
```

```
## [[1]]
##        [,1]   [,2]    [,3]      [,4]
## [1,] 1.0000 1.1070 0.61250 0.2270833
## [2,] 1.2214 1.3520 0.74875        NA
## [3,] 1.4918 1.6515      NA        NA
## [4,] 1.8221     NA      NA        NA
##
## [[2]]
## [1] 3
```

pregunta 2

```r
img1_path <- "p2_2022-06-02_141139.png"
include_graphics(img1_path)
```

Calcula el valor de la integral $\int_0^1 \log(x+5)\, dx$ con el método de Simpson compuesto y partiendo el intervalo en 10 subintervalos iguales.

Respuesta:

```r
fun <- function(x){
  return(log((x+5),10))
}
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)
}
simp(fun,0,1,10)
```

```
## [1] 0.739763
```

pregunta 3

```
img1_path <- "p3_2022-06-02_142149.png"
include_graphics(img1_path)
```

Dada la siguiente función definida a trozos:

$$f(x) = \begin{cases} 4 \cdot x^2 + 5 \cdot x + 4 & \text{si } x \in (-\infty, -1] \\ 3 \cdot x^2 - 4 \cdot x + 3 & \text{si } x \in [5, \infty) \end{cases}$$

Calcula el valor de la función en $x = 3$ usando interpolación lineal.

Respuesta:

```
myPhi <- function(x, n) {
  Phi <- matrix(1, length(x), n + 1)
  for (i in 1:n) {
    Phi[, i + 1] <- x^i # funciones base para el ajuste polinómico, segun el grado
  }
  return(Phi)
}

mylssolve <- function(A, y) {
  AT <- t(A)
  return((solve(AT %*% A)) %*% AT %*% y) # la función solve nos devuelve la inversa de la matriz
}

mypolyfit <- function(x, y, n) {
  Phi <- myPhi(x, n) # construimos la matriz con myPhi
  c <- mylssolve(Phi, y) # resolvemos el sistema de ecuaciones normales con la función mylssolve
  return(c)
```

```
}

myeval <- function(x, c) {
  f <- 0
  for (i in 1:length(c)) {
    f <- f + c[i] * x^(i - 1)
  }
  return(f)
}

# ESCRIBIR LAS FUNCIONES

f1 <- function(x){
  return(4*x**2+5*x+4)
}
f2 <- function(x){
  return(3*x**2-4*x+3)
}
lim1=-1
lim2=5
x=3

xn <- c(lim1,lim2)
yn <- c(f1(lim1),f2(lim2))

coef <- mypolyfit(xn,yn,1)

myeval(x,coef)
```

```
## [1] 39.66667
```

pregunta 4

```
img1_path <- "p4_2022-06-02_142630.png"
include_graphics(img1_path)
```

Un río tiene una anchura de $\frac{25}{36}$ m. En función de la distancia a la orilla, la profundidad del río, medida en sección recta, viene dada por la siguiente tabla:

$$\begin{bmatrix} Distancia & 0 & \frac{1}{36} & \frac{7}{36} & \frac{13}{36} & \frac{1}{2} & \frac{23}{36} & \frac{25}{36} \\ Profundidad & 7.784 & 7.799 & 7.893 & 7.985 & 8.06 & 8.133 & 8.162 \end{bmatrix}$$

Usando la regla del trapecio compuesta, dad una aproximación del area de la sección.

```r
## TRAPCIO PARA DATOS TABULADOS -- COMPLETOS--!!
x4 <- c(0,1/36,7/36,13/36,1/2,23/36,25/36)
y4 <- c(7.784,7.799,7.893,7.985,8.06,8.133,8.162)

trap4=function(x,y,m)
{
  a <- x[1]
  b <- x[length(x)]
  p.area=sum(y[2:(m+1)]+y[1:m])
  p.area=p.area*abs(b-a)/(2*m)
  return(p.area)
}

m=length(x4)-1

trap4(x4,y4,m)
```

```
## [1] 5.537384
```