TALLER INTERPOLACIÓN

Álvarez Sánchez, Emanuel
alvarez_emanuel@javeriana.edu.co

Afanador Ochoa, Juan José
j.afanador@javeriana.edu.co

Burgos Melo, Diego Andrés
burgosmd@javeriana.edu.co

Barrera Martinez, Nicolai
barreram.n@javeriana.edu.co

April 2020

1

Dados los n + 1 puntos distintos (xi, yi) el polinomio interpolante que incluye a todos los puntos es unico.
Se desarrolla utilizando los siguientes puntos : (-1,0),(1,3),(2,5),(4,2)

```
# Se desarrolla utilizando los siguientes puntos :
(-1,0),(1,3),(2,5),(4,2)
Matriz <- matrix(c(-1,1,-1,1,1,1,1,1,8,4,2,1,27,9,3,1), nrow=4,
ncol = 4, byrow=TRUE)
b <- c(0,3,5,2)
 #Primero se verifica que tenga solucion la matriz
det(Matriz)
 #Solucion para encontrar los coeficientes del polinomio
coeficientes <- (solve(Matriz,b))
cat("Resultado Polinomio: ", coeficientes[1],"X^3 +",
coeficientes[2],"x^2 +",coeficientes[3],"X +",coeficientes[4])
```

2

Construya un polinomio de grado tres que pase por: (0, 10),(1, 15),(2, 5) y que la tangente sea igual a 1 en x0

```
#Utilizando los puntos:(0,10), (1,15), (2,5)y sabiendo que
f'(0)=1 ya que la tangente debe ser igual a 1 en X0(0)
```

```
matriz <- matrix(c(0,0,0,1,1,1,1,1,8,4,2,1,0,0,1,0),
                 nrow=4,ncol = 4, byrow=TRUE)
vect <- c(10,15,5,1)

det(matriz) #Se verifica que la matriz tenga solucion

Coeficientes <- (solve(matriz,vect)) #Coeficientes del polinomio

print("--Matriz--")
print(matriz)
print("vector de imagen")
print(vect)

print(Coeficientes)
cat("Resultado Polinomio: ", Coeficientes[1],"X^3 +",
    Coeficientes[2],"x^2 +",Coeficientes[3],"X +",Coeficientes[4])
```

3

Construya un polinomio del menor grado que interpole una funcion f(x) en los siguientes datos: f(1) = 2; f(2) = 6; f 0 (1) = 3; f 0 (2) = 7; f 00(2) = 8

```
x <-c(1,1,2,2,2)
funcion <-c(2,3,6,7,4)
coeficientes <-c(0)

coeficientes[1]=funcion[1]
coeficientes[2]=funcion[2]
coeficientes[3]=(4-funcion[2])/(x[3]-x[1])
coeficientes[4]=((funcion[4]-(funcion[3]-funcion[1]))
-(coeficientes[3]))/(x[4]-x[1])

coeficientes[5]= ((funcion[5]- (funcion[4]-(funcion[3]
-funcion[1])))-(coeficientes[4]))/(x[5]-x[1])

print(coeficientes)

cat("P(x):",coeficientes[1],"+",coeficientes[2],
"( x-",x[1],")+",coeficientes[3],"( x-",x[2],")",
x[3],"+",coeficientes[4],"( x-",x[2],")",x[3],"( x-",x[3],")",
coeficientes[5],"( x-",x[2],")",x[4],"( x-",x[3],")",x[5])
```

4

Con la funcion f(x) = lnx construya la interpolacion de diferencias divididas en x0 = 1; x1 = 2 y estime el error en [1, 2]

```
f <- function(x){
  return(log(x))
}

fx <- c(0)
x <- c(0)
cont <- 1

for (i in 1:5){

  imagen <- f(i)
  fx[cont]=imagen
  x[cont]=i
  cont=cont+1
}
cat("Diferencias divididas\n")
tabla = data.frame(x,fx)
print(tabla)

#vector de diferencias divididas

diferencia <- c(0)
cont <- 1
diferencia[cont] = (fx[cont+1]-fx[cont])/(x[cont+1]-x[cont])
cont=cont+1
max = 5
while( cont < max){

  diferencia[cont] = (fx[cont+1]-fx[cont])/(x[cont+1]-x[cont])

  cont=cont+1

}
print(diferencia)

#debido a un x=1.5 se halla el error en [1,2]

lagra <- abs((1.5-x[1])*(1.5-x[2])/factorial(2))

f = expression(log(x))
```

```
Lagrange_Error <-function(f,Grado,sec){
  x=0;
  while (x < Grado){
    f= D(f,'x');
    x=x+1;
  }
  return (sec*abs(eval(f)));
}


cat("Error de Lagrange: ",Lagrange_Error(f,2,lagra))
```

5

Utilice la interpolacion de splines cubicos para el problema de la mano y del perrito

```
#MANO

library(stats)

x=c(15.6, 15.7, 15.8, 16.2, 16.6, 16.7,
     18.0, 18.6, 18.5, 18.3, 17.8, 14.4,
     15.8, 15.4, 15.5, 16.0, 16.1, 16.0,
     15.9, 15.3, 15.0, 14.9, 14.8, 14.5,
     14.1, 14.0, 14.3, 14.2, 13.9, 13.4,
     12.9, 12.7, 12.6, 12.3, 11.9, 11.7,
     11.6, 11.1, 10.7, 10.4, 10.3, 10.6,
     10.9, 11.1, 11.2, 11.3, 10.10, 9.6,
     8.5, 8.0, 7.7, 7.6, 8.70, 9.00, 9.10,
     9.40, 10.00, 10.30, 11, 11.2, 11.3,
     11.0, 10.50)

y=c(15.45, 14.0, 13.3, 12.0, 11.5, 11.2,
     9.20, 8.10, 7.70, 7.60, 7.80, 9.30,
     9.80, 10.30, 9.80, 7.30, 6.50, 6.00,
     5.70, 5.50, 5.90, 6.40, 6.80, 7.90,
     9.20, 8.60, 6.80, 5.50, 4.90, 5.20,
     6.70, 8.00, 8.90, 9.20, 8.30, 7.70,
     6.50,  5.60, 5.7, 6.0, 6.5, 8.2, 8.8,
     9.60, 10.40, 11.0, 11.7, 10.9, 10.0,
     10.1, 10.3, 10.7, 12.7, 13.3, 13.5,
     14.0, 14.9, 15.3, 16.5 17.4, 17.8,
```

```
    11.7,  12.0)

plot(x,y,main = "Interpolacion mano", asp = 1)
vectorx1 = c(x[1:3])
vectory1 = c(y[1:3])
splines = splinefun(vectorx1,vectory1,
                    method = "fmm")
curve(splines(x), add = TRUE, col = 1,
      from = vectorx1[1],
      to = vectorx1[length(vectorx1)])

vectorx2 = c(x[3:5])
vectory2 = c(y[3:5])
splines = splinefun(vectorx2,vectory2,
                    method = "fmm")
curve(splines(x), add = TRUE, col = 1,
      from = vectorx2[1],
      to = vectorx2[length(vectorx2)])

vectorx3 = c(x[5:8])
vectory3 = c(y[5:8])
splines = splinefun(vectorx3,vectory3,
                    method = "fmm")
curve(splines(x), add = TRUE, col = 1,
      from = vectorx3[1],
      to = vectorx3[length(vectorx3)])

vectorx4 = c(x[8:10])
vectory4 = c(y[8:10])
splines = splinefun(vectorx4,vectory4,
                    method = "fmm")
curve(splines(x), add = TRUE, col = 1,
      from = vectorx4[1],
      to = vectorx4[length(vectorx4)])

vectorx5 = c(x[10:13])
vectory5 = c(y[10:13])
splines = splinefun(vectorx5,vectory5,
                    method = "fmm")
curve(splines(x), add = TRUE, col = 1,
      from = vectorx5[1],
      to = vectorx5[length(vectorx5)])

vectorx6 = c(x[13:14])
vectory6 = c(y[13:14])
splines = splinefun(vectorx6,vectory6,
```

```
                        method = "fmm")
curve(splines(x), add = TRUE, col = 1,
       from = vectorx6[1],
       to = vectorx6[length(vectorx6)])

vectorc7 = c(x[14:17])
vectory7 = c(y[14:17])
splines = splinefun(vectorc7,vectory7,
                        method = "fmm")
curve(splines(x), add = TRUE, col = 1,
       from = vectorc7[1],
       to = vectorc7[length(vectorc7)])

vectorx8 = c(x[17:23])
vectory8 = c(y[17:23])
splines = splinefun(vectorx8,vectory8,
                        method = "fmm")
curve(splines(x), add = TRUE, col = 1,
       from = vectorx8[1],
       to = vectorx8[length(vectorx8)])

vectorx9 = c(x[23:26])
vectory9 = c(y[23:26])
splines = splinefun(vectorx9,vectory9,
                        method = "fmm")
curve(splines(x), add = TRUE, col = 1,
       from = vectorx9[1],
       to = vectorx9[length(vectorx9)])

vectorx10 = c(x[26:27])
vectory10 = c(y[26:27])
splines = splinefun(vectorx10,vectory10,
                        method = "fmm")
curve(splines(x), add = TRUE, col = 1,
       from = vectorx10[1],
       to = vectorx10[length(vectorx10)])

vectorx11 = c(x[27:28])
vectory11 = c(y[27:28])
splines = splinefun(vectorx11,vectory11,
                        method = "fmm")
curve(splines(x), add = TRUE, col = 1,
       from = vectorx11[1], to = vectorx11[length(vectorx11)])

vectorx12 = c(x[28:33])
vectory12 = c(y[28:33])
```

```
splines = splinefun(vectorx12,vectory12,
                    method = "fmm")
curve(splines(x), add = TRUE, col = 1,
      from = vectorx12[1],
      to = vectorx12[length(vectorx12)])

vectorx13 = c(x[33:37])
vectory13 = c(y[33:37])
splines = splinefun(vectorx13,vectory13,
                    method = "fmm")
curve(splines(x), add = TRUE, col = 1,
      from = vectorx13[1],
      to = vectorx13[length(vectorx13)])

vectorx14 = c(x[37:41])
vectory14 = c(y[37:41])
splines = splinefun(vectorx14,vectory14,
                    method = "fmm")
curve(splines(x), add = TRUE, col = 1,
      from = vectorx14[1],
      to = vectorx14[length(vectorx14)])

vectorx15 = c(x[41:46])
vectory15 = c(y[41:46])
splines = splinefun(vectorx15,vectory15,
                    method = "fmm")
curve(splines(x), add = TRUE, col = 1,
      from = vectorx15[1],
      to = vectorx15[length(vectorx15)])

vectorx16 = c(x[46:52])
vectory16 = c(y[46:52])
splines = splinefun(vectorx16,vectory16,
                    method = "fmm")
curve(splines(x), add = TRUE, col = 1,
      from = vectorx16[1],
      to = vectorx16[length(vectorx16)])

vectorx17 = c(x[52:57])
vectory17 = c(y[52:57])
splines = splinefun(vectorx17,vectory17,
                    method = "fmm")
curve(splines(x), add = TRUE, col = 1,
      from = vectorx17[1],
      to = vectorx17[length(vectorx17)])
```

```
vectorx18 = c(x[57:61])
vectory18 = c(y[57:61])
splines = splinefun(vectorx18,vectory18, method = "fmm")
curve(splines(x), add = TRUE, col = 1, from = vectorx18[1], to = vectorx18[lengt
```

6

Sea f(x) = tanx utilice la particion de la forma xi = k para implementar una interpolacion para n=10 puntos y encuentre el valor que minimice el error

7

Sea f(x) = e x en el intervalo de [0, 1] utilice el metodo de lagrange y determine el tama˜no del paso que me produzca un error por debajo de 105 . Es posible utilizar el polinomio de Taylor para interpolar en este caso? Verifique su respuesta

```
fx <- function(x){
  return(exp(1)^x)
}
fx = expression(exp(1)^x)
#Se halla la primer derivada con respecto al grado
primera_Derivada = D(fx,'x')
print(primera_Derivada)
#Se halla la segunda derivada con respecto a grado
segunda_Derivada = D(primera_Derivada, 'x')
print(segunda_Derivada)

evaluar_segunda_Derivada <- function(x) {
  eval(segunda_Derivada)
}

fx<-expression(exp(1)^x)

Derivar <-function(fx,Grado){
  x=0;
  while (x < Grado){
    fx= D(fx,'x');
    x=x+1;
  }
  return (fx);
}
```

```
resul<-Derivar(fx,5);
```

8

Considere el comportamiento de gases no ideales se describe a menudo con la ecuacion virial de estado. los siguientes datos para el nitrogeno N2
a) Determine un polinomio interpolante para este caso
b) Utilizando el resultado anterior calcule el segundo y tercer coeficiente virial a 450K.
c) Grafique los puntos y el polinomio que ajusta
d) Utilice la interpolacion de Lagrange y escriba el polinomio interpolante
e) Compare su resultado con la serie truncada (modelo teorico), cual aproximacion es mejor por que?

```
x<-c(100,200,300,400,500,600)
y<-c(-160,-35,-4.2,9.0,16.9,21.3)

matriz <- matrix(c(0,0,0,0,0,0), nrow=1,ncol = 6, byrow=TRUE)

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

largo <-length(x)
for(i in 1:largo){
  for(j in 0:largo){
    vecto[j+1]=x[i]^j

  }
  matriz<-rbind(matriz,c(vecto))
}
matriz <- matriz[1:length(x)+1,]

coeficientes <- (solve(matriz,y)) #Se hallan los coeficientes

cat("Polinomio: ",
    coeficientes[6],"X^5 +",
    coeficientes[5],"x^4 +",
    coeficientes[4],"X^3 +",
    coeficientes[3],"X^2 +",
    coeficientes[2],"X +",
    coeficientes[1])
```

```
cat("F(450) = ",coeficientes[6]*(450)^{5}+
       coeficientes[5]*(450)^{4} +
       coeficientes[4]*(450)^{3} +
       coeficientes[3]*(450)^{2} +
       coeficientes[2]*(450)
      +coeficientes[1])

cat ("segundo viral es:", coeficientes[2]*(450),
    "o es: ",coeficientes[2])

cat ("tercer viral es:", coeficientes[3]*(450)^{2},
    "o es: ",coeficientes[3])



para=function(x,y)
coeficientes[6]*(x)^{5}+
coeficientes[5]*(x)^{4}+
coeficientes[4]*(x)^{3}+
coeficientes[3]*(x)^{2}+
coeficientes[2]*(x)+
coeficientes[1]
z=outer(x, y, para)
persp(x,y,z,phi = 325,col = "red")
```