Análisis Numérico Taller 2 Interpolación

Gabriel Gómez Corredor Juan Pablo Méndez Simón Dávila

26 de Abril 2020

1 Introduccion de Interpolación

En general, el problema de la interpolación consiste en determinar una aproximación f(x) en un punto xi del dominio de f(x), a partir del conjunto (xi,yi) de valores conocidos o en sus vecindades Particularmente, la interpolación polinómica consiste en determinar f(xi) a partir de un polinomio P(x) de interpolación de grado menor o igual que n que pasa por los n + 1 puntos.

2 Desarrollo del taller

1) Dados los n + 1 puntos distintos (xi,yi) el polinomio interpolante que incluye a todos los puntos es único.
Sea f una función continua en [a, b] de la que se conoce el valor que toma en n+1 puntos distintos (nodos).

```
mat <- 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)

print(mat)
print(b)

det(mat) #corroborar que tenga solucion (det != 0)

coef_pol <- (solve(mat,b)) #se encuentran los coeficientes del polinomio

cat("Resultado Polinomio: ", coef_pol[1],"X^3 +",coef_pol[2],"x^2 +",coef_pol[3],"X +",coef_
```

2) Construya un polinomio de gradi tres que pase por: (0,10),(1,15),(2,5) y que la tangente sea igual a 1 en x0.
Para este ejercicio se generó la matriz de forma manual, para ello se realizó el proceso de reemplazar Xi en el polinimo de grado 3 (numero nodos) igualando al otro lado de la ecuacion a Yi (imagenes). Estas imagenes corresponden al vector b para realizar correctamente la función predeterminada de R solve con la matriz.

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

det(mat) #corroborar que tenga solucion (det != 0)

coef_pol <- (solve(mat,b)) #se encuentran los coeficientes del polinomio

print("Matriz resultante ")
print(mat)
print("vector de imagenes")
print(b)

print(coef_pol)

cat("Resultado Polinomio: ", coef_pol[1],"X^3 +",coef_pol[2],"x^2 +",coef_pol[3],"X +",coef_
```

3) Construya un polinomio del menor grado que interpole una función f(x) en los siguientes datos: f(1) = 2; f(2) = 6; fdx(1) = 3; fdx(2) = 7; fdx2(2) = 8. Para encontrar el polinomio interpolante con los anteriores datos, se realizó el metodo de diferencias divididas, para ello, se pusieron los Xi con sus imagenes conrespondientes en dos vectores correspondientes, Después se aplico este metodo de forma manual, utilizando las posiciones de los vectores planteados.

```
x <-c(1,1,2,2,2)
fx <-c(2,3,6,7,4)
coef_pol <-c(0)

coef_pol[1]=fx[1]
coef_pol[2]=fx[2]
coef_pol[3]=(4-fx[2])/(x[3]-x[1])
coef_pol[4]=((fx[4]-(fx[3]-fx[1]))-(coef_pol[3]))/(x[4]-x[1])
coef_pol[5]= ((fx[5]- (fx[4]-(fx[3]-fx[1])))-(coef_pol[4]))/(x[5]-x[1])

print(coef_pol)

cat("P(x):",coef_pol[1],"+",coef_pol[2],"( x-",x[1],")+",coef_pol[3],"( x-",x[2],")",x[3],"+

cat(" Simplificado -> P(x): -x^4 +8x^3 -20x^2 +23x-8")
```

4) Con la función f(x) = lnx construya la interpolación de diferencias divididas en x0 = 1;x1 = 2 y estime el error en [1,2] Para la construccion de la tabla de diferencias divididas se tomó en cuenta todos los Xi con sus respectivas imagenes, y en un ciclo se calculaba según la formula: xi+1 - xi / f(xi+1) -f(xi). Por último, se calculo el error en el intervalo [1,2], para ello se utilizó el metodo de error de Lagrange

```
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
}

tabla = data.frame(x,fx)
print(tabla)

#crear vector para almacenar resultados de diferencias divididas

dif_divi <- c(0)
cont = 1

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

while( cont < 5){

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

  cont=cont+1

}

print(dif_divi)

#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: ",Lagrange_Error(f,2,lagra))  #se evalua la segunda derivada en 2 por [1,2]
```

5) Utilice la interpolación de splines cúbicos para el problema de la mano y del perrito

```
library(stats)
x = c(00.50, 01.01, 05.85, 07.46, 11.28, 15.20, 18.46, 21.25, 24.15, 25.80, 28.00, 30.80, 30
y = c(02.40, 02.95, 03.86, 05.41, 07.45, 06.30, 04.49, 07.15, 07.05, 05.80, 05.85, 04.50, 02
plot(x,y,main = "Interpolación perrito", asp = 1)
vx1 = c(x[1:4])
vy1 = c(y[1:4])
splines = splinefun(vx1,vy1, method = "fmm")
curve(splines(x), add = TRUE, col = 1, from = vx1[1], to = vx1[length(vx1)])
vx2 = c(x[4:7])
vy2 = c(y[4:7])
splines = splinefun(vx2,vy2, method = "fmm")
curve(splines(x), add = TRUE, col = 1, from = vx2[1], to = vx2[length(vx2)])
vx3 = c(x[7:12])
vy3 = c(y[7:12])
splines = splinefun(vx3,vy3, method = "fmm")
curve(splines(x), add = TRUE, col = 1, from = vx3[1], to = vx3[length(vx3)])
vx4 = c(x[12:13])
vy4 = c(y[12:13])
splines = splinefun(vx4,vy4, method = "fmm")
curve(splines(x), add = TRUE, col = 1, from = vx4[1], to = vx4[length(vx4)])
vx5 = c(x[13:18])
vy5 = c(y[13:18])
splines = splinefun(vx5,vy5, method = "fmm")
curve(splines(x), add = TRUE, col = 1, from = vx5[1], to = vx5[length(vx5)])
vx6 = c(x[18:25])
vy6 = c(y[18:25])
splines = splinefun(vx6,vy6, method = "fmm")
curve(splines(x), add = TRUE, col = 1, from = vx6[1], to = vx6[length(vx6)])
vx7 = c(x[25:26])
vy7 = c(y[25:26])
splines = splinefun(vx7,vy7, method = "fmm")
curve(splines(x), add = TRUE, col = 1, from = vx7[1], to = vx7[length(vx7)])
vx8 = c(x[26:28])
```

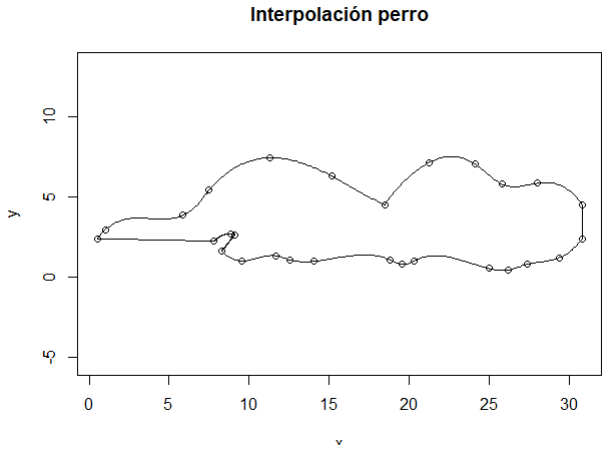

Figure 1: Gráfica Silueta Perro

```
vy8 = c(y[26:28])
splines = splinefun(vx8,vy8, method = "fmm")
curve(splines(x), add = TRUE, col = 1, from = vx8[1], to = vx8[length(vx8)])
vx9 = c(x[28:29])
vy9 = c(y[28:29])
splines = splinefun(vx9,vy9, method = "fmm")
curve(splines(x), add = TRUE, col = 1, from = vx9[1], to = vx9[length(vx9)])


# MANO


x=c(14.6, 14.7, 14.8, 15.2, 15.6, 15.7, 17.0, 17.6, 17.5, 17.3, 16.8, 15.4, 14.8, 14.4, 14.5
y=c(14.7, 14.0, 12.3, 11.0, 10.5, 10.2, 8.20, 7.10, 6.70, 6.60, 6.80, 8.30, 8.80, 9.30, 8.80
plot(x,y,main = "Interpolación mano", asp = 1)
vx1 = c(x[1:3])
vy1 = c(y[1:3])
splines = splinefun(vx1,vy1, method = "fmm")
curve(splines(x), add = TRUE, col = 1, from = vx1[1], to = vx1[length(vx1)])
vx2 = c(x[3:5])
vy2 = c(y[3:5])
splines = splinefun(vx2,vy2, method = "fmm")
curve(splines(x), add = TRUE, col = 1, from = vx2[1], to = vx2[length(vx2)])
vx3 = c(x[5:8])
vy3 = c(y[5:8])
splines = splinefun(vx3,vy3, method = "fmm")
curve(splines(x), add = TRUE, col = 1, from = vx3[1], to = vx3[length(vx3)])
vx4 = c(x[8:10])
```

```
vy4 = c(y[8:10])
splines = splinefun(vx4,vy4, method = "fmm")
curve(splines(x), add = TRUE, col = 1, from = vx4[1], to = vx4[length(vx4)])
vx5 = c(x[10:13])
vy5 = c(y[10:13])
splines = splinefun(vx5,vy5, method = "fmm")
curve(splines(x), add = TRUE, col = 1, from = vx5[1], to = vx5[length(vx5)])
vx6 = c(x[13:14])
vy6 = c(y[13:14])
splines = splinefun(vx6,vy6, method = "fmm")
curve(splines(x), add = TRUE, col = 1, from = vx6[1], to = vx6[length(vx6)])
vx7 = c(x[14:17])
vy7 = c(y[14:17])
splines = splinefun(vx7,vy7, method = "fmm")
curve(splines(x), add = TRUE, col = 1, from = vx7[1], to = vx7[length(vx7)])
vx8 = c(x[17:23])
vy8 = c(y[17:23])
splines = splinefun(vx8,vy8, method = "fmm")
curve(splines(x), add = TRUE, col = 1, from = vx8[1], to = vx8[length(vx8)])
vx9 = c(x[23:26])
vy9 = c(y[23:26])
splines = splinefun(vx9,vy9, method = "fmm")
curve(splines(x), add = TRUE, col = 1, from = vx9[1], to = vx9[length(vx9)])
vx10 = c(x[26:27])
vy10 = c(y[26:27])
splines = splinefun(vx10,vy10, method = "fmm")
curve(splines(x), add = TRUE, col = 1, from = vx10[1], to = vx10[length(vx10)])
vx11 = c(x[27:28])
vy11 = c(y[27:28])
splines = splinefun(vx11,vy11, method = "fmm")
curve(splines(x), add = TRUE, col = 1, from = vx11[1], to = vx11[length(vx11)])
vx12 = c(x[28:33])
vy12 = c(y[28:33])
splines = splinefun(vx12,vy12, method = "fmm")
curve(splines(x), add = TRUE, col = 1, from = vx12[1], to = vx12[length(vx12)])
vx13 = c(x[33:37])
vy13 = c(y[33:37])
splines = splinefun(vx13,vy13, method = "fmm")
curve(splines(x), add = TRUE, col = 1, from = vx13[1], to = vx13[length(vx13)])
vx14 = c(x[37:41])
vy14 = c(y[37:41])
splines = splinefun(vx14,vy14, method = "fmm")
curve(splines(x), add = TRUE, col = 1, from = vx14[1], to = vx14[length(vx14)])
vx15 = c(x[41:46])
vy15 = c(y[41:46])
splines = splinefun(vx15,vy15, method = "fmm")
```

```
curve(splines(x), add = TRUE, col = 1, from = vx15[1], to = vx15[length(vx15)])
vx16 = c(x[46:52])
vy16 = c(y[46:52])
splines = splinefun(vx16,vy16, method = "fmm")
curve(splines(x), add = TRUE, col = 1, from = vx16[1], to = vx16[length(vx16)])
vx17 = c(x[52:57])
vy17 = c(y[52:57])
splines = splinefun(vx17,vy17, method = "fmm")
curve(splines(x), add = TRUE, col = 1, from = vx17[1], to = vx17[length(vx17)])
vx18 = c(x[57:61])
vy18 = c(y[57:61])
splines = splinefun(vx18,vy18, method = "fmm")
curve(splines(x), add = TRUE, col = 1, from = vx18[1], to = vx18[length(vx18)])
```

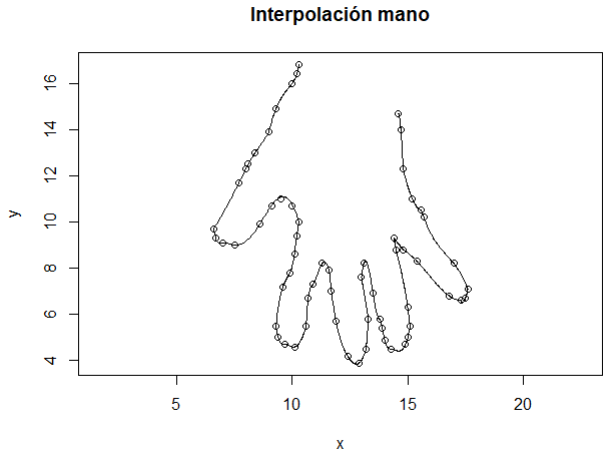

Figure 2: Interpolación Mano

7) Sea f(x) = ex en el intervalo de [0,1] utilice el método de lagrange y determine el tamaño del paso que me produzca un error por debajo de 10e5. Es posible utilizar el polinomio de Taylor para interpolar en este caso? Verifique su respuesta.
Para el desarrollo de este ejercicio se tomó como referencia la funcion ya implementada en el pdf Tec Digital (En referencia). Luego, utlizamos las función predeterminada en R de Taylor para evaluar si esta serie tambien funcionaba como solcuión. Este llamado a la función Taylor nos devuelve los coeficientes del polinomio, para comprobar gráficamente usamos la función curve para poder observar la tendencia de dichos polinomios.

```
f <- function( x ) { exp(x) }
lagrange = function(x,y,a){
  n = length(x)
```

```
  if(a < min(x) || max(x) < a) stop("No está interpolando")
  X = matrix(rep(x, times=n), n, n, byrow=T)
  mN = a - X; diag(mN) = 1
  mD = X - t(X); diag(mD) = 1
  Lnk = apply(mN, 1, prod)/apply(mD, 2, prod)
  sum(y*Lnk)
}
x<-c(0,0.25,0.5,0.75,1)
y<-f(x[1:5])
lagrange(x[1:5],y[1:5],0.5)

require(pracma)
f = function(x) exp(x)
g= function (x) 1/x
polinomio_1= taylor(f, 0, 4)
polinomio_2= taylor(g, 5, 4)
polinomio_3=taylor(g,-5,4)
curve(f, col= "blue", from = -10, to= 10)
curve(polinomio_1[1]*x^(4)+ppolinomio_1[2]*x^(3)+polinomio_1[3]*x^(2)+ppolinomio_1[4]*x+poli
curve(g, col= "purple", from = -10, to= 10)
curve(polinomio_2[1]*x^(4)+polinomio_2[2]*x^(3)+polinomio_2[3]*x^(2)+polinomio_2[4]*x+polinc
curve(polinomio_3[1]*x^(4)+polinomio_3[2]*x^(3)+polinomio_3[3]*x^(2)+polinomio_3[4]*x+polinc
cat(polinomio_1[1],"x^(4)+",polinomio_1[2],"*x^(3)+",polinomio_1[3],"*x^(2)+",polinomio_1[4]
cat(polinomio_2[1],"x^(4)+",polinomio_2[2],"*x^(3)+",polinomio_2[3],"*x^(2)+",polinomio_2[4]
cat(polinomio_3[1],"x^(4)+",polinomio_3[2],"*x^(3)+",polinomio_3[3],"*x^(2)+",polinomio_3[4]
```

Teniendo ya los polinomios, se generó la siguiente grafica con sus curvas correspondientes a la tendencia que toman dichos polinomios:

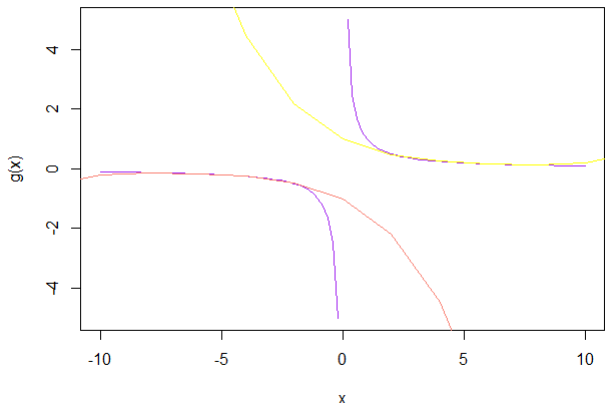

Figure 3: Comportamiento Polinomios

8) Considere el comportamiento de gases no ideales se describe a menudo con la ecuación virial de estado. los siguientes datos para el nitrógeno N2. Donde

T(K)	100	200	300	400	450	500	600
$B(cm^3)/mol$	-160	-35	-4.2	9.0		16.9	21.3

Figure 4: Tabla de Temperaturas

T es la temperatura [K] y B es el segundo coeficiente virial. El comportamiento de gases no ideales se describe a menudo con la ecuación virial de estado.
Donde P es la presión, V el volumen molar del gas, T es la temperatura Kelvin y R es la constante de gas ideal.Los coeficientes B = B(T), C = C(T), son el segundo y tercer coeficiente virial, respectivamente.
a) Determine un polinomio interpolante para este caso.
Para encontrar el polinomio interpolante hallamos la matriz según en el grado del polinomio, en este caso como nos dieron 6 puntos el polinomio era de grado 5. Despúes de hallar esta matriz, se tomo el vector b que corresponde a los Yi, es decir las imagenes. Teniendo una vez estos coeficientes pudimos construir el polinomio interpolante para evaluar P(450).

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

mat <- matrix(c(0,0,0,0,0,0), nrow=1,ncol = 6, 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,]
```

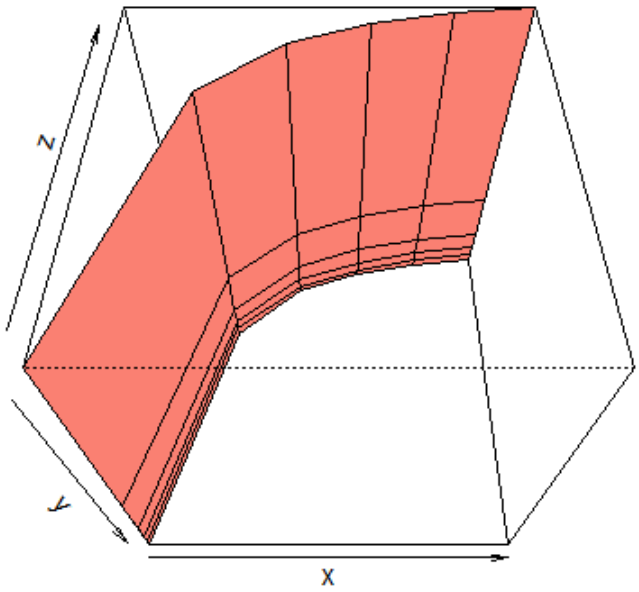

Figure 5: Grafica de Polinomio interpolante

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

cat("Resultado Polinomio: ", coef_pol[6],"X^5 +",coef_pol[5],"x^4 +",coef_pol[4],"X^3 +",c

cat("F(450) = ",coef_pol[6]*(450)^{5}+coef_pol[5]*(450)^{4} +coef_pol[4]*(450)^{3} +coef_p
cat ("el segundo viral es:", coef_pol[2]*(450), "o es: ",coef_pol[2])
cat ("el segundo viral es:", coef_pol[3]*(450)^{2}, "o es: ",coef_pol[3])
```

Por último, graficamos en 3D los puntos dados en el enunciado con el polinomio interpolante resultante, dando como resultado la siguiente grafica:

```
parabola=function(x,y) coef_pol[6]*(x)^{5}+coef_pol[5]*(x)^{4} +coef_pol[4]*(x)^{3} +coef_po
z=outer(x, y, parabola)
persp(x,y,z)
persp(x,y,z,phi = 325,col = "salmon")
```

[1, 2]

References

[1] asignaturas.topografia. *Interpolacion.pdf*. 2018.

[2] Walter Mora F. *Cómo utilizar R en métodos numéricos*. 2016.