INTERPOLACIÓN POLINÓMICA DE LAGRANGE

Nota: en todos los programas, 'x' será el vector en el que almacenamos los puntos de soporte; 'a' será el punto en el que queremos interpolar; y 'f' será el vector en el que almacenamos los valores de los puntos de soporte.

1.POR SISTEMA DE ECUACIONES

Este método existe gracias a la propiedad de que solamente hay una función de grado ≤n-1 pasa por todos sus valores. Suponiendo que tenemos un polinomio de grado ≤n-1:

$$p(x) = ax^{n-1} + bx^{n-2} + \cdots + cx + d$$

Sustituyendo todos los puntos del soporte en el polinomio e igualándolos a sus valores correspondientes obtenemos un sistema de ecuaciones, resolviéndolo nos da los coeficientes.

En R, usando el comando 'solve', es muy sencillo obtener los coeficientes del polinomio:

```
p=function(a,x,f){
        n=length(x);d=0
        A=matrix(c(0),nrow=n,ncol=n)          CREACIÓN DE LA MATRIZ QUE SALE DE
        for(i in 1:n){                        SUSTITUIR LOS PUNTOS DE SOPORTE EN
                for(j in 1:n){                EL POLINOMIO.
                        A[i,j]=x[i]**(j-1)
                }
        }
        B=matrix(c(f),nrow=n,ncol=1)          MATRIZ QUE CONTIENE LOS VALORES
        c=solve(A,B)                          GUARDAMOS LA MATRIZ QUE SALE EN 'c'
        for(i in 1:length(c)){                OBTENEMOS EL VALOR INTERPOLADO EN 'd'
                d=d+c[i]*a**(i-1)
        }
        return(d)                             PEDIMOS QUE NOS DEVUELVA EL 'd'
}
```

Ejemplo 1.1:

Usando como entrada de la función x=(1,4,5,6,9,18); f=(2,12,31,42,-12,-16); y a=10, nos da:

```
p(a,x,f)
[1] -59.42081
```

Ejemplo 1.2:

Usamos x=(1,2,4,9); f=(2,4,8,18); y a=7; si te das cuenta, estos son valores de la función f(x)=2x, y para a=7, nos tendría que dar 14; y efectivamente:
p(a,x,f)

[1] 14

2.POR BASES DE LAGRANGE

Puesto que lo que nos interesa es el programa en R y su explicación, no explicaremos por qué de esta forma podemos conseguir el polinomio interpolador, sino que daremos la fórmula y el programa:

$$p(x) = \sum f_i(x) * L_i(x) = \sum (f_i(x) \prod_{i \neq j} \frac{(a - x_i)}{(x_i - x_j)}) \qquad (1 \leq i \leq n), (1 \leq j \leq n)$$

donde L es la base de Lagrange.

Veamos su programa en R:

```
Bases_Lagrange=function(a,x){
n=length(x)
prod=1
for(i in 1:n){
prod[i]=1
for(j in 1:n){
if(i!=j){
prod[i]=prod[i]*(a-x[j])/(x[i]-x[j])}}}
return(prod)}
w=function(Bases_Lagrange,f){
suma=0; n=length(f); L=Bases_Lagrange(a,x)
for(i in 1:n){
suma=suma+f[i]*L[i]}
return(suma)}
```

DEFINIR LA BASE DE LAGRANGE, AUNQUE NO ES NECESARIO DEFINIR 2 FUNCIONES PARA ESTE TIPO DE INTERPOLACIÓN, A VECES, EN OTROS TIPOS DE INTERPOLACIONES, LA BASE DE LAGRANGE PUEDE RESULTAR ÚTIL.

LA FUNCIÓN PARA SACAR EL VALOR INTERPOLADO.
Lo peculiar de esta función es que tiene un argumento que es otra función, cuyo valor lo guardamos en un vector.

NOTA: Las llaves han sido colocadas de esta forma por la simple razón de ahorrar espacio, es más recomendable colocarlos debajo de los procesos.

Ejemplo 2.1:

Usamos las mismas entradas usadas en el ejemplo 1.1: x=(1,4,5,6,9,18); f=(2,12,31,42,-12,-16); y a=10; y según la propiedad de que hay solamente 1 función de grado ≤ n-1, que pasa por todos los valores, la función ha de ser la misma, igual que el valor interpolado:

w(Bases_Lagrange,f)

[1] -59.42081 (COINCIDEN)

Ejemplo 2.2:

Usamos las siguientes entradas: x= (3,4,7,9); f=(27,64,343,729);a=6. Son valores de la función f(x)=x^3; por tanto, para a=6, tendría que salir 216:

w(Bases_Lagrange,f)

[1] 216

3.POR FÓRMULA DE NEWTON/DIFERENCIAS DIVIDIDAS

Igual que la base de Lagrange, NO demostraremos su validez, solo lo introduciremos y analizaremos su programa. La expresión del polinomio es la siguiente:

$$p(x) = f_1 + \sum (f[x_1, x_2, \dots, x_i] * \prod_{j=1}^{i-1} (x - x_j)) \qquad 2 \le i \le n$$

Recordatorio: $f[x_1, x_2] = \frac{f(x_2)-f(x_1)}{x_2-x_1}$ $\qquad$ $f[x_1, x_2, x_3] = \frac{f[x_2,x_3]-f[x_1,x_2]}{x_3-x_1}$

Para programar la fórmula, creamos primero una tabla de diferencias divididas, que será una matriz, y aprovecharemos su primera fila para la fórmula:

```
m=function(a,x,f){
suma=f[1];n=length(x)                    ASIGNAMOS AQUÍ QUE LA SUMA ES f[1]
A=matrix(c(0),nrow=n,ncol=n)             CREACIÓN DE LA MATRIZ DE DIFERENCIAS
A[,1]=f                                  DIVIDIDAS
for(j in 2:n){
for(i in 1:(n-j+1)){
A[i,j]=(A[i+1,j-1]-A[i,j-1])/(x[i+j-1]-x[i])}}
d=A[1,]                                  CREAMOS UN VECTOR CON LA PRIMERA FILA
for(i in 2:n){                           APLICAMOS LA FÓRMULA DE NEWTON
prod=1
for(j in 1:(i-1)){
prod=prod*(a-x[j])}
suma=suma+d[i]*prod}
return(suma)}                            PEDIMOS QUE NOS DEVUELVA LA 'suma'
```

Ejemplo 3.1:

Usamos las mismas entrada que los ejemplos 1.1 y 2.1: x=(1,4,5,6,9,18); f=(2,12,31,42,-12,-16); y a=10. Nos tendría que dar el mismo resultado que los anteriores:

m(a,x,f)

[1] -59.42081

Ejemplo 3.2 (apartado a) del ejercicio 3 del 1º parcial 2022-2023):
Usamos los datos x=(-3,-2,-1,0,1,2); f=(cos(π/5)-243, cos(π/5)-32, cos(π/5)-1, cos(π/5), cos(π/5)+1, cos(π/5)+32); a=0,5. La función es f(x)= cos(π/5)+x^5, por tanto, nos tendría que salir $\approx 0{,}840266994$, y efectivamente:

m(a,x,f)

[1] 0.840266994

4.INTERPOLACIÓN POR TRAMOS

Como bien indica su nombre, en este tipo de interpolación, dividiremos los puntos de soporte en pequeños intervalos y a cada intervalo le aplicamos un método de interpolación, que pueden ser distintos o iguales. Este método es útil en cuanto tenemos muchos puntos de soporte, y su polinomio interpolador puede oscilar mucho, de forma que no nos da un resultado coherente.

Los programas no son complicados, lo único que es nuevo es que tenemos que añadir unas cuantas condiciones 'if'. Analizaremos 4 programas en total, que serán respectivamente: interpolación por base de Lagrange de grado 1 y de grado 2; e interpolación por fórmula de Newton con 2 puntos de soporte:

```
#POR TRAMOS BASES LAGRANGE GRADO 1

A=function(a,x){

n=length(x);L=0

if(a<=x[2]&a>=x[1]){                Es posible meter estas dos condiciones en el bucle,

L[1]=(a-x[2])/(x[1]-x[2])           pero es mucho más cómodo definirlas antes del bucle

}else{                              Si no definiéramos estos 'NO' de la condición, nos

L[1]=0}                             puede dar errores

if(a<=x[n]&a>=x[n-1]){

L[n]=(a-x[n-1])/(x[n]-x[n-1])

}else{

L[n]=0}

for(i in 2:(n-1)){                  Definimos con la combinación de un bucle con

if(a<=x[i]&a>=x[i-1]){              una condición la base de Lagrange.
```

```
L[i]=(a-x[i-1])/(x[i]-x[i-1])
}else if(a<=x[i+1]&a>=x[i]){
L[i]=(a-x[i+1])/(x[i]-x[i+1])
}else{
L[i]=0}}
return(L)}                          Pedimos que nos devuelva el vector L
```

Hasta aquí el programa para sacar la base de Lagrange

```
#-------------------------------------------------------------------------------------------------------------------------
u=function(A,a,f){                  Uno de los argumentos es una función, que será usado
suma=0;n=length(f)                  Y 'L' va a ser un vector.
L=A(a,x)                            La entrada 'a' es importante para cuando tengamos
for(i in 1:n){                      más de un punto a interpolar
suma=suma+f[i]*L[i]}                Proceso de interpolación
return(suma)}
#-------------------------------------------------------------------------------------------------------------------------
```

#Base de Lagrange de grado 2: necesitamos un **número impar de puntos de soporte**

```
B=function(a,x){
L=0;n=length(x)
if(a<=x[3]&a>=x[1]){                Ídem grado 1. Pero en este caso es muchísimo
L[1]=(a-x[2])*(a-x[3])/((x[1]-x[2])*(x[1]-x[3]))   más cómodo definir estas condiciones fuera
}else{                              del bucle
L[1]=0}
if(a<=x[n]&a>=x[n-2]){
L[n]=(a-x[n-2])*(a-x[n-1])/((x[n]-x[n-1])*(x[n]-x[n-2]))
}else{
L[n]=0}
for(i in seq(2,(n-1),2)){           Este bucle con un 'seq' sirve para definir las
if(a<=x[i+1]&a>=x[i-1]){            bases en los puntos pares, ya que se definen
L[i]=(a-x[i-1])*(a-x[i+1])/((x[i]-x[i-1])*(x[i]-x[i+1]))   de forma diferente que los impares
}else{
L[i]=0 }}
```

```
if(length(x)>3){                Esta condición hace que el programa sirva para 3 puntos
for(i in seq(3,(n-2),2)){       Definición de las bases de los x impares, que son diferentes
if(a<=x[i]&a>=x[i-2]){          que los x pares: hay 2 condiciones dentro del bucle.
L[i]=(a-x[i-2])*(a-x[i-1])/((x[i]-x[i-2])*(x[i]-x[i-1]))
}else if(a<=x[i+2]&a>=x[i]){
L[i]=(a-x[i+1])*(a-x[i+2])/((x[i]-x[i+1])*(x[i]-x[i+2]))
}else{
L[i]=0}}}
return(L)}                      Pedimos que nos devuelva el vector 'L'
```

Hasta aquí la definición de la base de Lagrange

```
#-----------------------------------------------------------------------------------------------------------------------
uu=function(B,a,f){             Uno de los argumentos es un función
suma=0;n=length(f)              Ídem grado 1
V=B(a,x)
for(i in 1:n){
suma=suma+f[i]*V[i]}            Proceso de interpolación
return(suma)}
#-----------------------------------------------------------------------------------------------------------------------
#Por tramos con diferencias divididas/fórmula de Newton: 2 puntos de soporte:
AA=function(a,x,f){             Con 2 puntos de soporte con la fórmula de Newton es
n=length(x)                     relativamente sencillo, ya que no tenemos que definir la
for(i in 1:(n-1)){              tabla de las diferencias divididas para facilitar el cálculo.
if(a<=x[i+1]&a>=x[i]){          Elegir los puntos de soporte para la interpolación.
u=f[i]+(f[i+1]-f[i])/(x[i+1]-x[i])}} Proceso de interpolación
return(u)}                      Le pedimos que nos devuelva el valor interpolado
```

Ejemplo 4.1:

En este ejemplo vamos a usar unas mismas entradas y probamos los tres programas de interpolación por tramos; y recordemos que la interpolación por tramos de base de Lagrange de grado 1 tiene que salir lo mismo que con la fórmula de Newton con 2 puntos de soporte.

```
a=10; x=c(1,4,5,6,9,18,20); f=c(2,12,31,42,-12,-16,23)
```

#Base de Lagrange grado 1 (2 puntos soporte):
A(a,x)

[1] 0.0000000 0.0000000 0.0000000 0.0000000 0.8888889 0.1111111 0.0000000

> u(A,f)

[1] -12.44444

#Base de Lagrange grado 2 (3 puntos soporte):

B(a,x)

[1] 0.0000000 0.0000000 0.0000000 0.0000000 0.8080808 0.5555556

[7] -0.3636364

> uu(B,f)

[1] -26.94949

#Fórmula Newton (2 puntos de soportes):

AA(a,x,f)

[1] -12.44444 (Coincide)

Ejemplo 4.2:

Un biotecnólogo está cultivando la bacteria E. Coli, que quiere ver para cuánto tiempo todas las bacterias mueren. Para ello toma dato de la población cada 6 horas; y justamente para en la última toma de datos del 2º día, vio que ya no había bacterias. Con los datos obtenidos quiere averiguar cuántas bacterias hay 2h después de cada 2ª toma de datos del día 1 y 2 mediante una interpolación (sin contar la situación inicial):

x(horas)=(0,6,12,18,24,30,36,42,48)

f(población bacterias en millones)=(0.00002,0.4,4,8.2,19.2,90.2,60.5,12,0)

puntos en los que interpolamos: a=14; b=38

Solo usaremos la fórmula de Newton; y base de Lagrange de grado 1 y 2 por tramos para contrastar las diferencias entre una interpolación contando con todos los puntos de soporte y una contando solamente unos puntos de los de soporte:

#Fórmula de Newton

m(a,x,f)

[1] 12.33113

> m(b,x,f)

[1] 26.50727

#Por partes base Lagrange grado 1

u(A,a,f)

[1] 5.4

> u(A,b,f)

[1] 44.33333

#Por partes base Lagrange grado 2

uu(B,a,f)

[1] 4.644444

> uu(B,b,f)

[1] 40.27778

En este ejemplo concreto, como todos los valores son positivos, NO nos aparecen valores negativos, sin embargo, en el caso de que sí aparecieran, por ejemplo, en la con la fórmula de Newton, no podremos usar ese resultado como el valor interpolado, ya que una población negativa no tiene sentido ninguno; y en este caso escogeríamos el resultado de la fórmula de Newton, ya que no nos da un resultado ilógico, y el grado es mayor, por tanto, su exactitud.