Taller Interpolación

Sebastian Angarita, Hector Rodriguez, Aldemar Ramirez

5/9/2019

1. Dados los n + 1 puntos distintos (xi, yi) el polinomio interpolante que incluye a todos los puntos es ´unico

Teorema de Aproximación: Supongase que f este definida y sea continua en [a,b]. Para todo E>0, existe un polinomio P(X) con la propieda de que |f(x)-p(x)|<E para todo x perteneciente al rango [a,b].

Teorema de Unicidad: Sea $x_k n+1$ valores distintos(nodos) y sea f una función cuyos valores es esos puntos. Existe un unico P(x) de grado menor o igual a n con identidad: $f(x_k) = P(x_k)$ para todo $k = 0, 1, 2, ..., n$.

Suponemos que hay mas de dos polinomios distintos, P(x) y Q(x) de grado a lo sumo n que verifican $p(x_i) = y$; y $q(x_i) = y$; para $i = 0, 1, ...n$.

Sea el polinomio $r(x) = p(x) - q(x)$ se sabe que para $i = 0, 1..., n$, $r(x_i) = p(x_i) - q(x_i) = 0$. $r(x)$ tendria $n + 1$ raices distintas, ya que x_i es distinto por hipotesis.

Tambien se sabe que $r(x)$ es de grado a lo sumo n, por ser la diferencia de dos polinomios distintos de grado n.

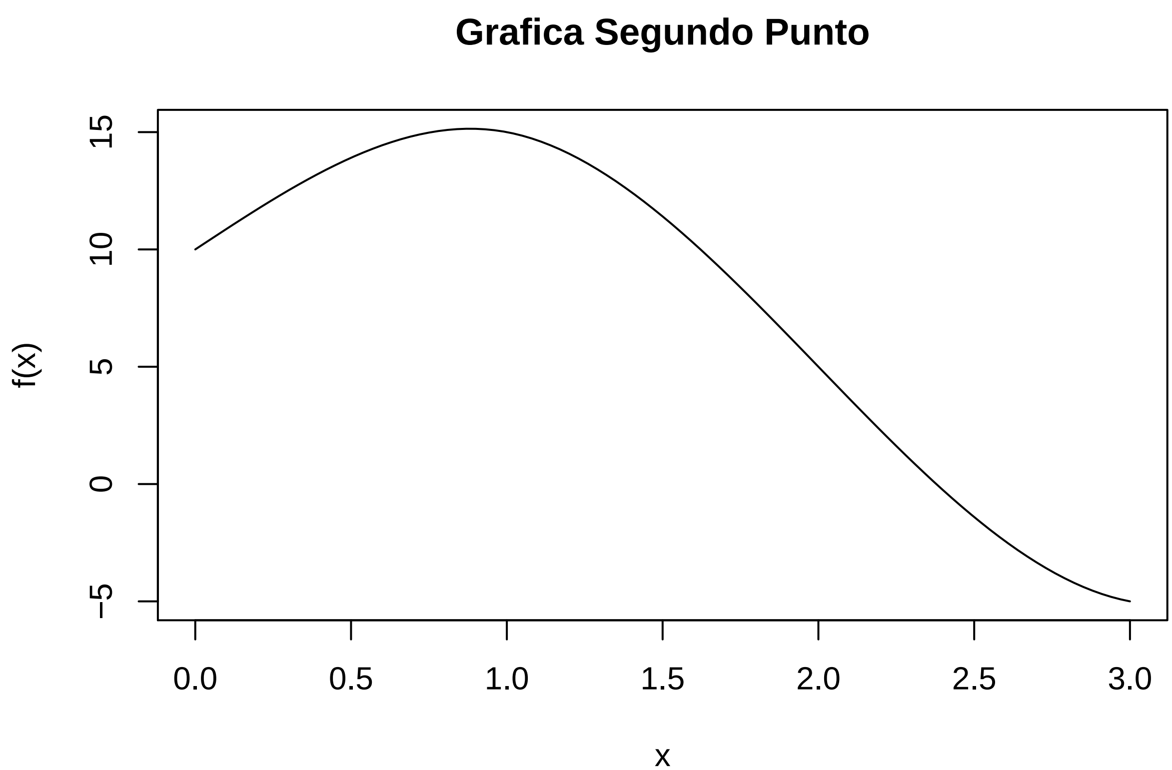
Un polinomio de grado n tiene como maximo n raices distintas o iguales, pero sabemos que $r(x)$ que es de grado a lo sumo n y tiene $n + 1$ raices distintas. Por lo tanto dos raices son iguales y se llega a una contradicción.

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

```r
require(pracma)
```

```
## Loading required package: pracma
```

```r
x = c(0,1,2)
y= c(10,15,5)
xs <- seq(0, 2, by = 1)

pp <- cubicspline(x, y)
ppfun <- function(xs) ppval(pp, xs)

curve(ppfun(x),from=0,to=3,xlab = "x", ylab = "f(x)",
     main = "Grafica Segundo Punto")
```



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

Se hace una Interpolación polinómica por medio del método de diferencias divididas de Newton sabiendo que : .

$f[x_k] = f(x_k), k \in [0, n]$

$f[x_k, x_k + 1)] = \dfrac{f[x_k + 1] - f[x_k]}{x_k + 1 - x_k}, k \in [0, n-1]$

$f[x_k, x_k + 1, ..., x_k + i] = \dfrac{f[x_k + 1, ..., x_k + i] - f[x_k, ..., x_k + i - 1]}{x_k + i - x_k}, k \in [0, n-i]$

```
# Create the data frame.
emp.data <- data.frame(
   "x_k"= c (1:2),
   "fx_k"= c("f[1]=2","f[2]=6"),
  "f[x_k...fx_k1]"= c("f[1,1]=3","f[2,2]=7"),
   "f[x_k...fx_k2]" = c("f[1,1,2]=1","f[2,2,2]=4"),
   "f[x_k...fx_k3]"=c("f[1,1,2,2]=2","-"),
   "f[x_k...fx_k4]"=c("f[1,1,2,2,2]=-1","-"),
   stringsAsFactors = FALSE
)
# Print the data frame.
print(emp.data)
```

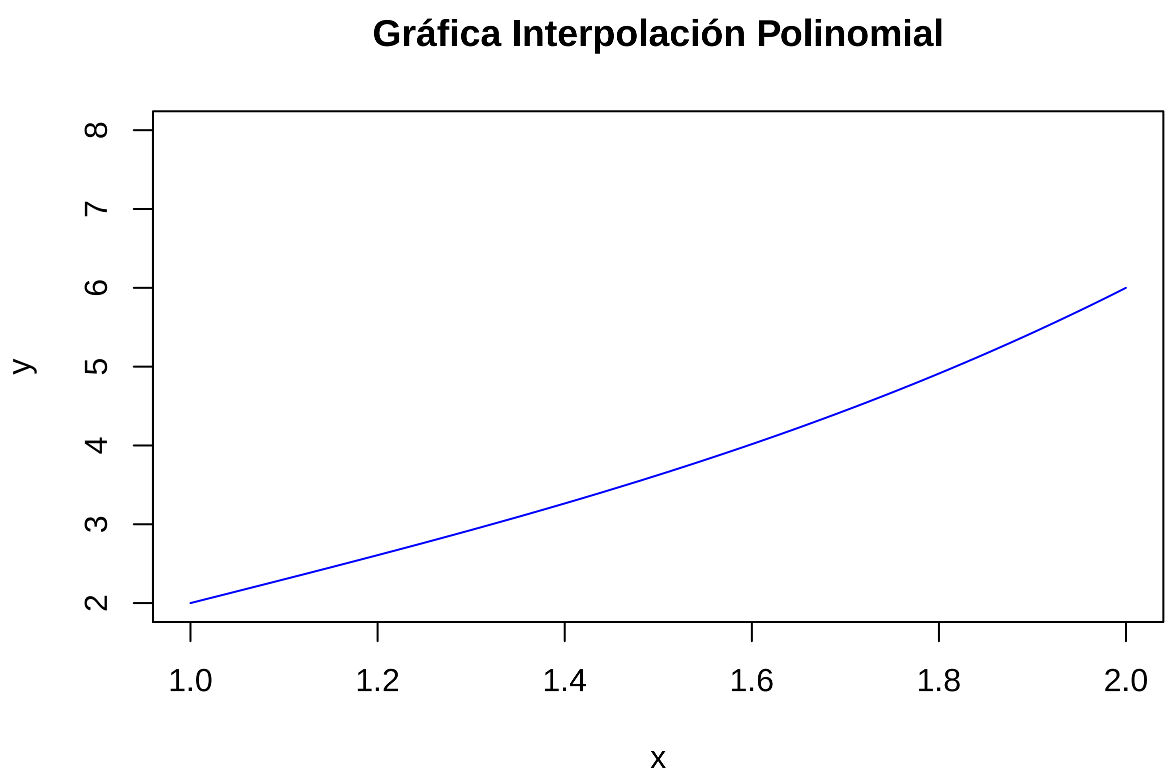
```
##   x_k   fx_k f.x_k...fx_k1. f.x_k...fx_k2. f.x_k...fx_k3.  f.x_k...fx_k4.
## 1   1 f[1]=2       f[1,1]=3     f[1,1,2]=1   f[1,1,2,2]=2 f[1,1,2,2,2]=-1
```

```
## 2   2 f[2]=6       f[2,2]=7     f[2,2,2]=4              -                -
```

Con estos datos y haciendo uso de la fórmula de newton $$se procede a hallar el polinomio requerido : $x^3 - 3x^2 + 6x - 2$

Obteniendo como resultado la siguiente gráfica

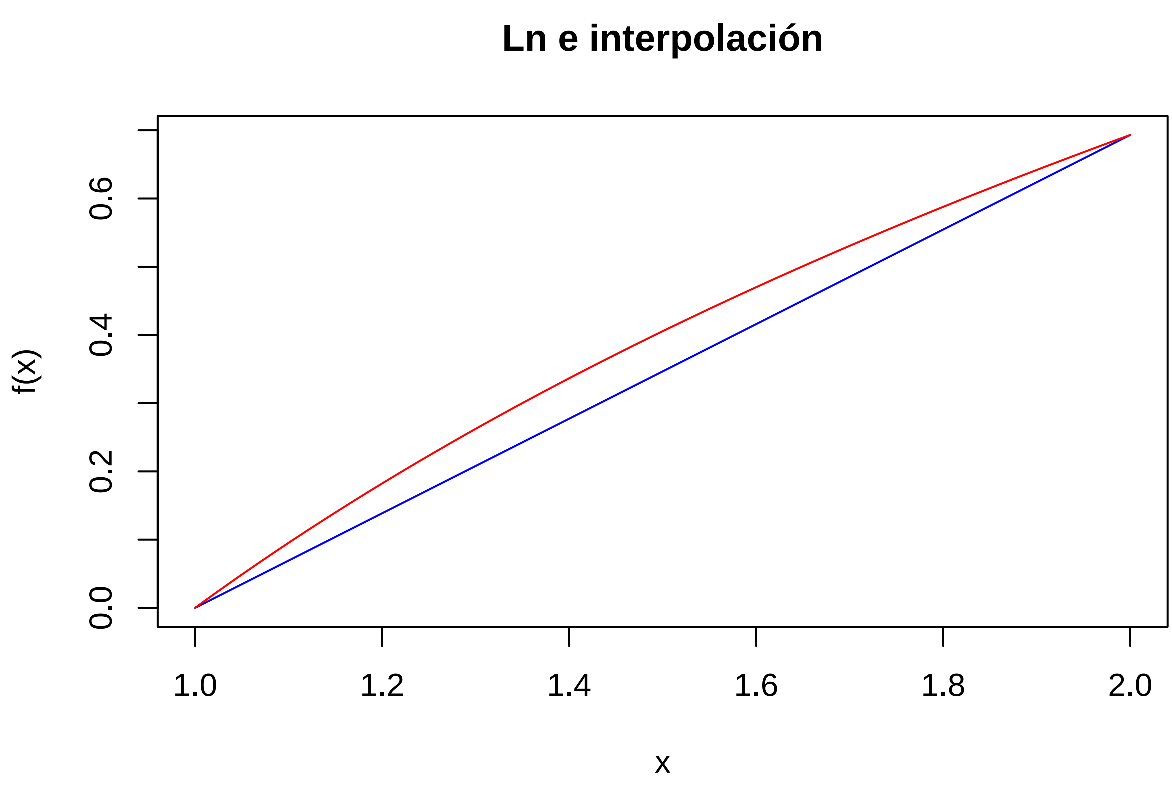
```
y = c (2,8)
x = c(1,2)
fx = function(x) {x^3-3*x^2+6*x-2}
plot(fx,xlab="x",ylab="y",ylim=y,xlim=x,main="Gráfica Interpolación Polinomial",col="blue")
```



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]

```
newtonInterpolacion = function(x, y, a) {
  n = length(x)
  A = matrix(rep(NA, times = n^2), nrow = n, ncol = n)
  A[,1] = y
  for (k in 2:n) {
    A[k:n, k] = (A[k:n, k-1] - A[(k-1):(n-1), k-1] ) / (x[k:n] - x[1:(n-k+1)])
  }
  # Imprimir matriz de diferencias divididas
  print(A)
  # Evaluar
  smds = rep(NA, length = n)
```

```r
  smds[1] = 1 #x = x[1],..., x[n] pues n = length(x)
  for (k in 2:n) {
    smds[k] = (a - x[k-1])*smds[k-1] # hasta x[n-1]
  }
  return(sum(diag(A)*smds) )
}
arithmetic.mean <- function(x) {return(sum(x)/length(x))}

f = function(x) log(x)
x1 = c(1,2)
logaritmo = f(x1)

#Traza de la función Logaritmica
plot(x1,logaritmo,type="l", col="blue",xlab = "x", ylab = "f(x)",
     main = "Ln e interpolación")
#Traza de la recta
curve(log,1,2,add = T, col="red")
```




```r
x2 = seq(1,2,by=0.1)
interpolados = c()

for (i in x2) {
  interpolados = c(interpolados,newtonInterpolacion(x1,logaritmo,i));
}
```

```
##            [,1]      [,2]
## [1,] 0.0000000        NA
## [2,] 0.6931472 0.6931472
##            [,1]      [,2]
## [1,] 0.0000000        NA
## [2,] 0.6931472 0.6931472
##            [,1]      [,2]
## [1,] 0.0000000        NA
## [2,] 0.6931472 0.6931472
##            [,1]      [,2]
## [1,] 0.0000000        NA
## [2,] 0.6931472 0.6931472
##            [,1]      [,2]
## [1,] 0.0000000        NA
## [2,] 0.6931472 0.6931472
##            [,1]      [,2]
## [1,] 0.0000000        NA
## [2,] 0.6931472 0.6931472
##            [,1]      [,2]
## [1,] 0.0000000        NA
## [2,] 0.6931472 0.6931472
##            [,1]      [,2]
## [1,] 0.0000000        NA
## [2,] 0.6931472 0.6931472
##            [,1]      [,2]
## [1,] 0.0000000        NA
## [2,] 0.6931472 0.6931472
##            [,1]      [,2]
## [1,] 0.0000000        NA
## [2,] 0.6931472 0.6931472
##            [,1]      [,2]
## [1,] 0.0000000        NA
## [2,] 0.6931472 0.6931472
```

```
# Tabla de Interpolados
print(interpolados)
```

```
##  [1] 0.00000000 0.06931472 0.13862944 0.20794415 0.27725887 0.34657359
##  [7] 0.41588831 0.48520303 0.55451774 0.62383246 0.69314718
```

```
reales = f(x2)
errores = c()

for (i in 1:length(reales)) {
  errores = c(errores,abs(reales[i]-interpolados[i]))
}

# Tabla de errores
print(errores)
```

```
##  [1] 0.00000000 0.02599546 0.04369212 0.05442011 0.05921336 0.05889152
##  [7] 0.05411532 0.04542522 0.03326892 0.01802142 0.00000000
```

```
#Promedio de error

prom = arithmetic.mean(errores)
print(prom)
```

```
## [1] 0.03573122
```

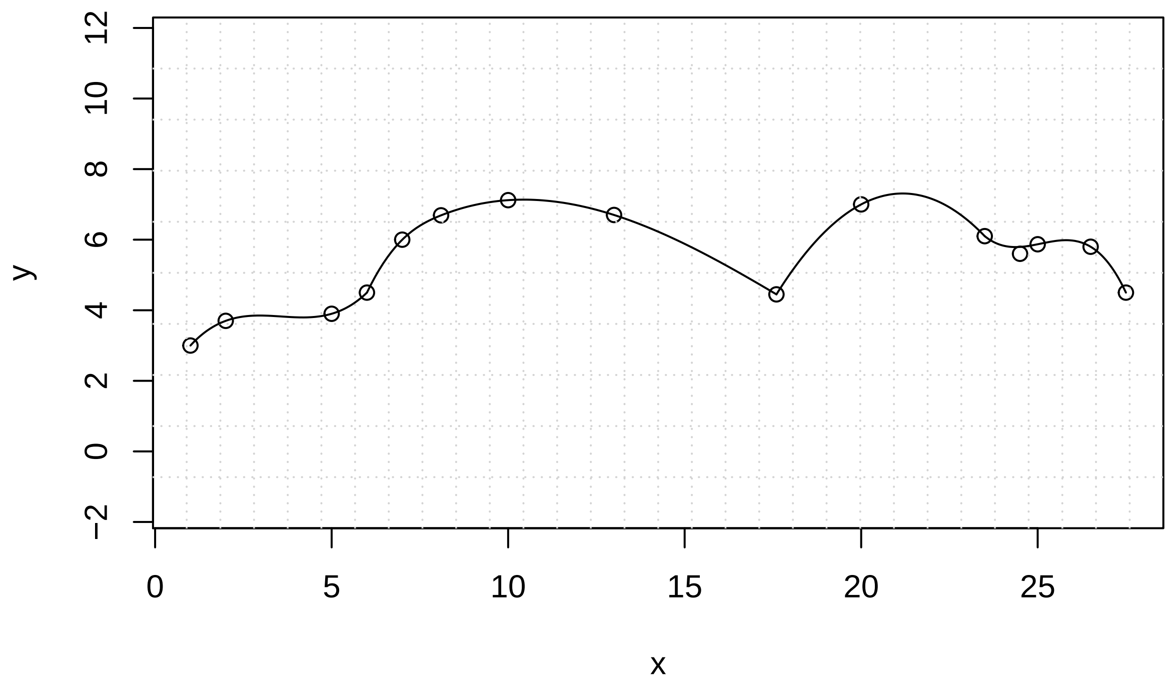
La traza de color roja hace referencia a la función Ln, mientras que la azul se relaciona con la recta $x_0 = 1; x_1 = 2$.

5. Utilice la interpolación de splines cubicos para el problema de la mano y del perrito.

Problema Perrito

```
require(stats)

#Problema:Reconstruir la silueta del perrito utilizando la menor cantidad de puntos para reproducir el
#del contorno completo del perrito sin bigotes,con la información dada

#Coordenadas:
y=c(3,3.7,3.9,4.5,6,6.69,7.12,6.7,4.45,7,6.1,5.6,5.87,5.8,4.5)
x=c(1,2,5,6,7,8.1,10,13,17.6,20,23.5,24.5,25,26.5,27.5)

x1=x[1:4]
y1=y[1:4]

x2=c(x[4:9])
y2=c(y[4:9])

plot(x,y,asp=1)
grid(nx=30,ny = 10)

splinesRunge = splinefun(x1,y1, method="fmm")

curve(splinesRunge(x), add=TRUE, col=1,from = x1[1],to=x1[length(x1)])

splinesRunge = splinefun(x2,y2, method="fmm")


curve(splinesRunge(x), add=TRUE, col=1,from = x2[1],to=x2[length(x2)])

x3=c(x[9:11])
y3=c(y[9:11])

splinesRunge = splinefun(x3,y3, method="fmm")
curve(splinesRunge(x), add=TRUE, col=1,from = x3[1],to=x3[length(x3)])

x4=c(x[11],x[13:15])
y4=c(y[11],y[13:15])

# x4=c(x[11:13])
# y4=c(y[11:13])
splinesRunge = splinefun(x4,y4, method="fmm")
```

```r
curve(splinesRunge(x), add=TRUE, col=1,from = x4[1],to=x4[length(x4)])
```




```r
# x5=c(x[13:15])
# y5=c(y[13:15])
```

Problema Mano

```r
#install.packages("Matrix")#instalar paquete
library(Matrix)
```

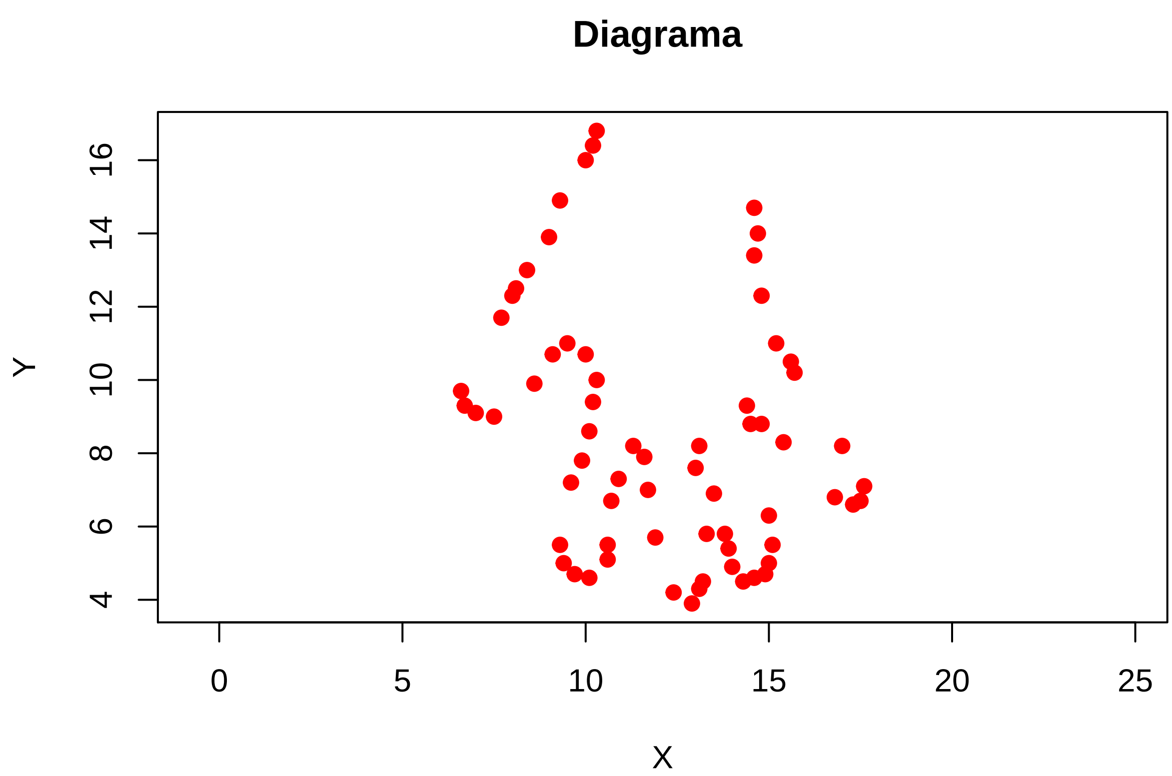
```
##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:pracma':
##
##     expm, lu, tril, triu
```

```r
#install.packages("PolynomF")#instalar paquete
library(PolynomF)
```

```
##
## Attaching package: 'PolynomF'
```

```
## The following object is masked from 'package:pracma':
##
##     integral
```

```r
#interpolacion
##Puntos

x=c(14.6, 14.7, 14.6, 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, 15.0
y=c(14.7, 14.0, 13.4, 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, 6.30


plot(x,y, pch=19, cex=1, col = "red", asp=1,xlab="X", ylab="Y", main="Diagrama ")
```




```r
DatosX = x[1:5]; DatosY = y[1:5]
Ajuste_Polinomio = poly_calc(DatosX,DatosY)
```

```
## Warning in poly_calc(DatosX, DatosY): some duplicated x-points have
## inconsistent y-values
```

```r
Ajuste_Polinomio
```

```
## -420896 + 85191.58*x - 5746.25*x^2 + 129.1667*x^3
```

```r
help("Defunct")
```

```
## starting httpd help server ...
```

```
##  done
```

```r
DatosX1 = x[4:7]; DatosY1 = y[4:7]
Ajuste_Polinomio1 = poly_calc(DatosX1,DatosY1)
Ajuste_Polinomio1
```

```
## 24018.64 - 4697.983*x + 306.5*x^2 - 6.666667*x^3
```

```r
DatosX2 = x[7:9]; DatosY2 = y[7:9]
Ajuste_Polinomio2 = poly_calc(DatosX2,DatosY2)
Ajuste_Polinomio2
```

```
## -7.067881 + 3.536437*x - 0.1551957*x^2
```

```r
DatosX3 = x[9:10]; DatosY3 = y[9:10]
Ajuste_Polinomio3 = poly_calc(DatosX3,DatosY3)
Ajuste_Polinomio3
```

```
## -63.3 + 4*x
```

```r
DatosX4 = x[10:15]; DatosY4 = y[10:15]
Ajuste_Polinomio4 = poly_calc(DatosX4,DatosY4)
Ajuste_Polinomio4
```

```
## 23360.57 - 6895.82*x + 808.6899*x^2 - 47.01523*x^3 + 1.352867*x^4 -
## 0.01538416*x^5
```

```r
DatosX5 = x[15:19]; DatosY5 = y[15:19]
Ajuste_Polinomio5 = poly_calc(DatosX5,DatosY5)
```

```
## Warning in poly_calc(DatosX5, DatosY5): some duplicated x-points have
## inconsistent y-values
```

```r
Ajuste_Polinomio5
```

```
## 22452.73 - 4592.857*x + 313.5714*x^2 - 7.142857*x^3
```

```r
DatosX6 = x[19:24]; DatosY6 = y[19:24]
Ajuste_Polinomio6 = poly_calc(DatosX6,DatosY6)
Ajuste_Polinomio6
```

```
## -3342825 + 1186642*x - 168325.4*x^2 + 11927.13*x^3 - 422.1821*x^4 +
## 5.972423*x^5
```

```
DatosX7 = x[24:28]; DatosY7 = y[24:28]
Ajuste_Polinomio7 = poly_calc(DatosX7,DatosY7)
Ajuste_Polinomio7
```

```
## -854143.1 + 251864*x - 27845.16*x^2 + 1367.976*x^3 - 25.19841*x^4
```

```
DatosX8 = x[28:29]; DatosY8 = y[28:29]
Ajuste_Polinomio8 = poly_calc(DatosX8,DatosY8)
Ajuste_Polinomio8
```

```
## 85.6 - 6*x
```

```
DatosX9 = x[29:32]; DatosY9 = y[29:32]
Ajuste_Polinomio9 = poly_calc(DatosX9,DatosY9)
Ajuste_Polinomio9
```

```
## -306738.7 + 70428.12*x - 5390*x^2 + 137.5*x^3
```

```
DatosX10 = x[32:36]; DatosY10 = y[32:36]
Ajuste_Polinomio10 = poly_calc(DatosX10,DatosY10)
Ajuste_Polinomio10
```

```
## 38783.28 - 12287.26*x + 1461.877*x^2 - 77.39744*x^3 + 1.538462*x^4
```

```
DatosX11 = x[36:41]; DatosY11 = y[36:41]
Ajuste_Polinomio11 = poly_calc(DatosX11,DatosY11)
```

```
## Warning in poly_calc(DatosX11, DatosY11): some duplicated x-points have
## inconsistent y-values
```

```
Ajuste_Polinomio11
```

```
## -680729.9 + 245393.8*x - 33165.14*x^2 + 1991.667*x^3 - 44.84127*x^4
```

```
DatosX12 = x[41:45]; DatosY12 = y[41:45]
Ajuste_Polinomio12 = poly_calc(DatosX12,DatosY12)
Ajuste_Polinomio12
```

```
## 85661.05 - 34505.73*x + 5210.386*x^2 - 349.5243*x^3 + 8.788665*x^4
```

```
DatosX13 = x[45:50]; DatosY13 = y[45:50]
Ajuste_Polinomio13 = poly_calc(DatosX13,DatosY13)
Ajuste_Polinomio13
```

```
## 9923776 - 5058218*x + 1030851*x^2 - 104998.5*x^3 + 5345.139*x^4 -
## 108.7963*x^5
```

```
DatosX14 = x[50:52]; DatosY14 = y[50:52]
Ajuste_Polinomio14 = poly_calc(DatosX14,DatosY14)
Ajuste_Polinomio14
```

```
## -104.3704 + 24.33235*x - 1.284314*x^2
```

```
DatosX15 = x[52:55]; DatosY15 = y[52:55]
Ajuste_Polinomio15 = poly_calc(DatosX15,DatosY15)
Ajuste_Polinomio15
```

```
## 31.30742 - 4.481499*x - 0.008373206*x^2 + 0.02791069*x^3
```

```
DatosX16 = x[55:56]; DatosY16 = y[55:56]
Ajuste_Polinomio16 = poly_calc(DatosX16,DatosY16)
Ajuste_Polinomio16
```

```
## 36.1 - 4*x
```

```
DatosX17 = x[56:60]; DatosY17 = y[56:60]
Ajuste_Polinomio17 = poly_calc(DatosX17,DatosY17)
Ajuste_Polinomio17
```

```
## -1980.1 + 1055.725*x - 210.0408*x^2 + 18.557*x^3 - 0.6132756*x^4
```

```
DatosX18 = x[59:62]; DatosY18 = y[59:62]
Ajuste_Polinomio18 = poly_calc(DatosX18,DatosY18)
Ajuste_Polinomio18
```

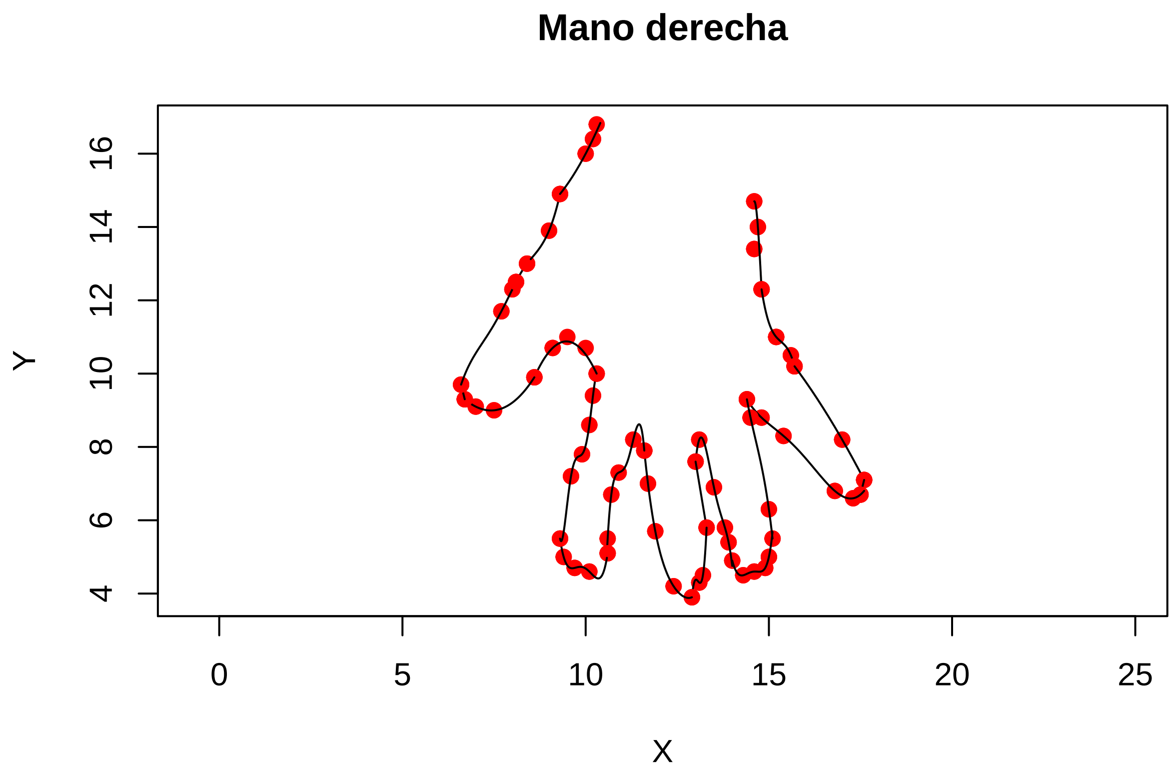
```
## -1147.6 + 405.7222*x - 47.40741*x^2 + 1.851852*x^3
```

```
DatosX19 = x[62:64]; DatosY19 = y[62:64]
Ajuste_Polinomio19 = poly_calc(DatosX19,DatosY19)
Ajuste_Polinomio19
```

```
## 44.57143 - 7.619048*x + 0.4761905*x^2
```

```
#ingresamos los ajustes y los puntos en la grafica
plot(x,y, pch=19, cex=1, col = "red", asp=1,xlab="X", ylab="Y", main="Mano derecha")
points(DatosX,DatosY, pch=19, cex=1, col = "red", asp=1,xlab="X", ylab="Y", main="Mano derecha")
curve(Ajuste_Polinomio,add=T,from =14.6,to =14.8)
points(DatosX1,DatosY1, pch=19, cex=1, col = "red", asp=1,xlab="X", ylab="Y", main="Mano derecha")
curve(Ajuste_Polinomio1,add=T,from =14.8,to =15.7)
points(DatosX2,DatosY2, pch=19, cex=1, col = "red", asp=1,xlab="X", ylab="Y", main="Mano derecha")
curve(Ajuste_Polinomio2,add=T,from =15.7,to =17.6)
points(DatosX3,DatosY3, pch=19, cex=1, col = "red", asp=1,xlab="X", ylab="Y", main="Mano derecha")
curve(Ajuste_Polinomio3,add=T,from =17.5,to =17.6)
points(DatosX4,DatosY4, pch=19, cex=1, col = "red", asp=1,xlab="X", ylab="Y", main="Mano derecha")
curve(Ajuste_Polinomio4,add=T,from =14.4,to =17.6)
points(DatosX5,DatosY5, pch=19, cex=1, col = "red", asp=1,xlab="X", ylab="Y", main="Mano derecha")
```

```r
curve(Ajuste_Polinomio5,add=T,from =14.4,to =15.1)
points(DatosX6,DatosY6, pch=19, cex=1, col = "red", asp=1,xlab="X", ylab="Y", main="Mano derecha")
curve(Ajuste_Polinomio6,add=T,from = 14, to = 15.1)
points(DatosX7,DatosY7, pch=19, cex=1, col = "red", asp=1,xlab="X", ylab="Y", main="Mano derecha")
curve(Ajuste_Polinomio7,add=T,from = 14, to = 13)
points(DatosX8,DatosY8, pch=19, cex=1, col = "red", asp=1,xlab="X", ylab="Y", main="Mano derecha")
curve(Ajuste_Polinomio8,add=T,from = 13.3, to = 13)
points(DatosX9,DatosY9, pch=19, cex=1, col = "red", asp=1,xlab="X", ylab="Y", main="Mano derecha")
curve(Ajuste_Polinomio9,add=T,from = 12.9, to = 13.3)
points(DatosX10,DatosY10, pch=19, cex=1, col = "red", asp=1,xlab="X", ylab="Y", main="Mano derecha")
curve(Ajuste_Polinomio10,add=T,from = 11.6, to = 12.9)
points(DatosX11,DatosY11, pch=19, cex=1, col = "red", asp=1,xlab="X", ylab="Y", main="Mano derecha")
curve(Ajuste_Polinomio11,add=T,from = 10.58, to = 11.6)
points(DatosX12,DatosY12, pch=19, cex=1, col = "red", asp=1,xlab="X", ylab="Y", main="Mano derecha")
curve(Ajuste_Polinomio12,add=T,from = 9.3, to = 10.58)
points(DatosX13,DatosY13, pch=19, cex=1, col = "red", asp=1,xlab="X", ylab="Y", main="Mano derecha")
curve(Ajuste_Polinomio13,add=T,from = 9.3, to = 10.3)
points(DatosX14,DatosY14, pch=19, cex=1, col = "red", asp=1,xlab="X", ylab="Y", main="Mano derecha")
curve(Ajuste_Polinomio14,add=T,from =8.6, to = 10.3)
points(DatosX15,DatosY15, pch=19, cex=1, col = "red", asp=1,xlab="X", ylab="Y", main="Mano derecha")
curve(Ajuste_Polinomio15,add=T,from =6.65, to = 8.6)
points(DatosX16,DatosY16, pch=19, cex=1, col = "red", asp=1,xlab="X", ylab="Y", main="Mano derecha")
curve(Ajuste_Polinomio16,add=T,from =6.6, to = 6.7)
points(DatosX17,DatosY17, pch=19, cex=1, col = "red", asp=1,xlab="X", ylab="Y", main="Mano derecha")
curve(Ajuste_Polinomio17,add=T,from =6.6, to = 8.5)
points(DatosX18,DatosY18, pch=19, cex=1, col = "red", asp=1,xlab="X", ylab="Y", main="Mano derecha")
curve(Ajuste_Polinomio18,add=T,from =8.5, to = 9.3)
points(DatosX19,DatosY19, pch=19, cex=1, col = "red", asp=1,xlab="X", ylab="Y", main="Mano derecha")
curve(Ajuste_Polinomio19,add=T,from =9.3, to = 10.4)
```

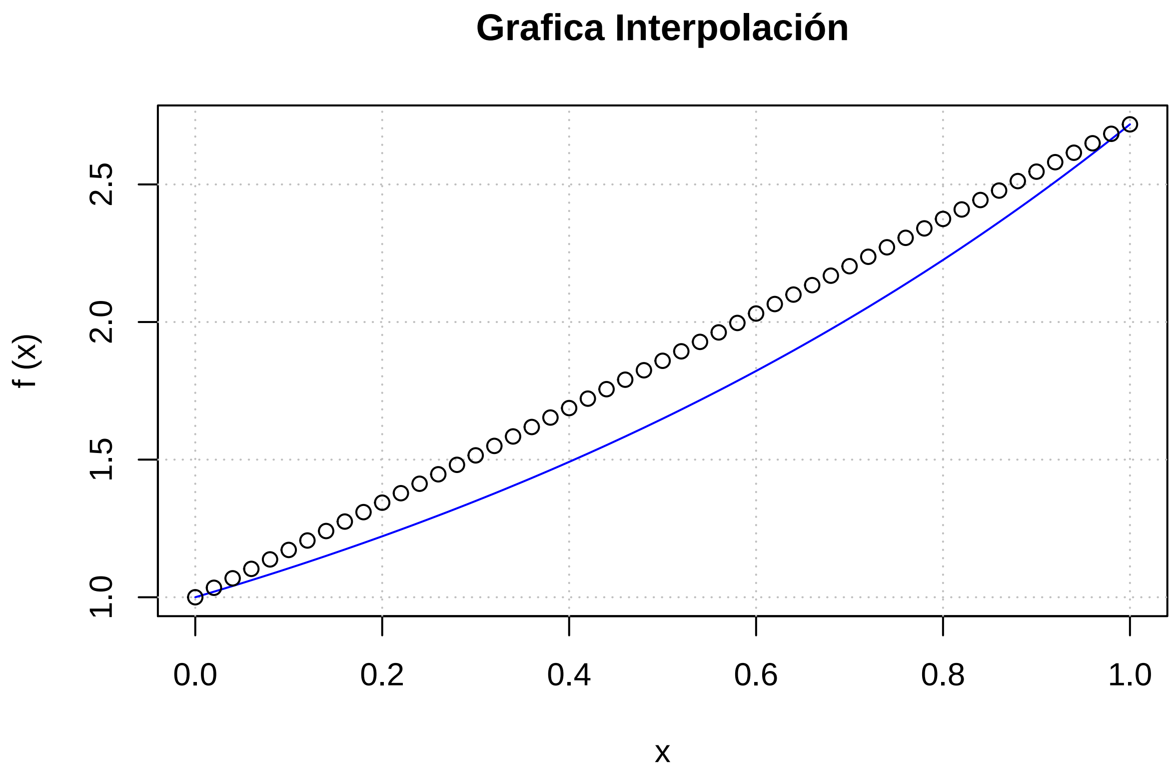


7. Sea f(x) = e^x en el intervalo de [0; 1] utilice el método de lagrange y determine el tamaño del paso que me produzca un error por debajo de 10^{-5}. Es posible utilizar el polinomio de Taylor para interpolar en este caso? Verifique su respuesta

```
library(pracma)
f = function(x) {exp(x)}
x = 0:1;
y = f(x)
xx =linspace(0,1,51)
yy = lagrangeInterp(x,y,xx)
#print(yy) Existe un error debido a la diferencia de longitud entre "x" y "y"
yy = newtonInterp(x,y,xx)
print(yy)
```

```
##  [1] 1.000000 1.034366 1.068731 1.103097 1.137463 1.171828 1.206194
##  [8] 1.240559 1.274925 1.309291 1.343656 1.378022 1.412388 1.446753
## [15] 1.481119 1.515485 1.549850 1.584216 1.618581 1.652947 1.687313
## [22] 1.721678 1.756044 1.790410 1.824775 1.859141 1.893507 1.927872
## [29] 1.962238 1.996603 2.030969 2.065335 2.099700 2.134066 2.168432
## [36] 2.202797 2.237163 2.271529 2.305894 2.340260 2.374625 2.408991
## [43] 2.443357 2.477722 2.512088 2.546454 2.580819 2.615185 2.649551
## [50] 2.683916 2.718282
```

```
ezplot(f, 0, 1, main = "Grafica Interpolación")
points(xx, yy)
```



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

Donde 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. En la práctica se usa la serie truncada para aproximar.

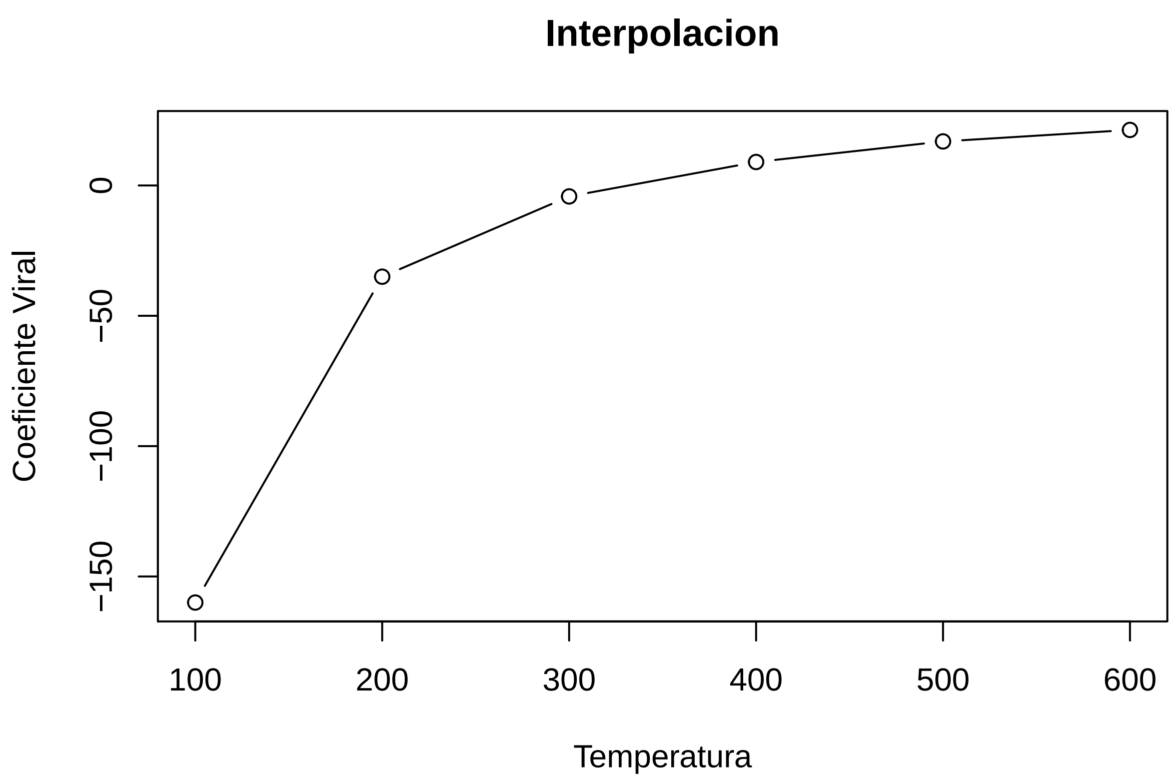
$PV/RT = 1 + B/V$

En la siguiente figura se muestra como se distribuye la variable B a lo largo de la temperatura.

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 interpolación de Lagrange y escriba el polinomio interpolante.
e) Compare su resultado con la serie truncada (modelo teorico), cual aproximacion es mejorpor que?

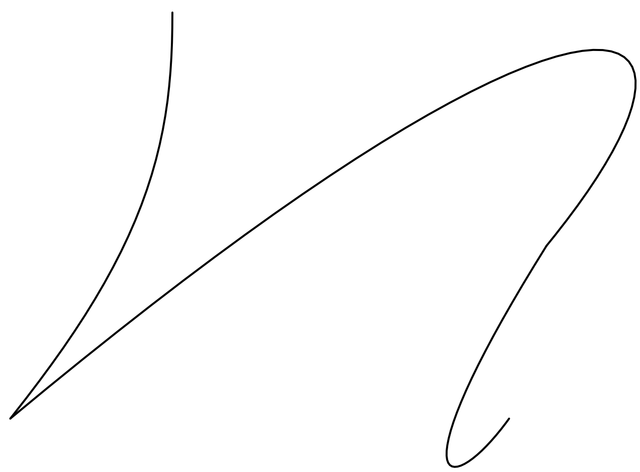
```
require(pracma)
library(pracma)
x = c(100,200,300,400,500,600)
y = c(-160,-35,-4.2,9.0,16.9,21.3)

xi= seq(100,600,by=100)
pp = cubicspline(x,y,der=c(100,700))
```

```r
ppfun =function(xi) ppval(pp, xi)

x1=c(400)
y1=ppfun(x1)

plot(x,y,main="Interpolacion",xlab="Temperatura",ylab="Coeficiente Viral",type="b")
```



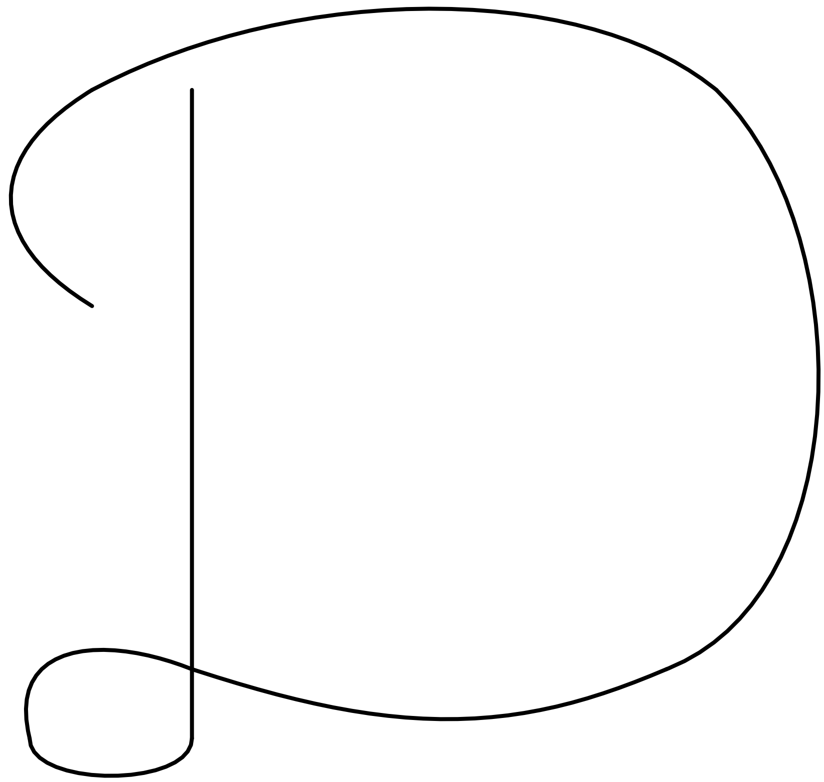
Interpolación de Bezier para letra "n"

```r
library(grid)
library(gridBezier)

x1 <-c(0.23, 0.23 , 0.21 , 0.1 )
y1 <- c(0.57, 0.4 , 0.3 ,0.1)
x2 <- c(0.1,0.6,0.7,0.53)
y2 <- c(0.1,0.7,0.6,0.3)
x3<-c(0.53,0.4,0.45,0.5)
y3<-c(0.3,0,0,0.1)
grid.newpage()
grid.bezier(x1,y1)
grid.bezier(x2,y2)
grid.bezier(x3,y3)
```



Interpolación de Bezier para letra "D"

```r
library(grid)
library(gridBezier)
x <- c(0.25, 0.16,  0.16, 0.25)
y <- c(0.58, 0.66, 0.75, 0.83)
x0 <- c(0.25, 0.42,  0.64, 0.75)
y0 <- c(0.83, 0.96, 0.96, 0.83)
x1 <- c(0.75, 0.87, 0.87, 0.71)
y1 <- c(0.83, 0.66 , 0.25, 0.16)
x2 <- c(0.71, 0.58, 0.5, 0.33)
y2 <- c(0.16, 0.08, 0.08, 0.16)
x3 <- c(0.33, 0.26, 0.18, 0.2)
y3 <- c(0.16, 0.2, 0.2, 0.08)
x4 <- c(0.2, 0.2, 0.33, 0.33)
y4 <- c(0.08, 0.02, 0.02, 0.08)
x5 <- c(0.33, 0.33, 0.33, 0.33)
y5 <- c(0.08, 0.25, 0.75, 0.83)
grid.newpage()
grid.bezier(x, y, gp=gpar(lwd = 2, fill="black"))
grid.bezier(x0, y0, gp=gpar(lwd = 2, fill="black"))
grid.bezier(x1, y1, gp=gpar(lwd = 2, fill="black"))
grid.bezier(x2, y2, gp=gpar(lwd = 2, fill="black"))
grid.bezier(x3, y3, gp=gpar(lwd = 2, fill="black"))
grid.bezier(x4, y4, gp=gpar(lwd = 2, fill="black"))
grid.bezier(x5, y5, gp=gpar(lwd = 2, fill="black"))
```



Interpolación para la letra

```
library(grid)
library(gridBezier)

x1 <- c(0.2, 0.22 , 0.15 ,0.05)
y1 <-c(0.7, 0.5 , 0.35 , 0.1 )

x2 <- c(0.05,0.15,0.35,0.41)
y2 <- c(0.1,0.35,0.58,0.5)

x3<-c(0.41,0.41,0.35,0.31)
y3<-c(0.5,0.4,0.36,0.2)

x4<-c(0.31,0.35,0.5,0.55)
y4<-c(0.2,0.3,0.55,0.44)

x5<-c(0.55,0.5,0.45,0.54)
y5<-c(0.44,0.3,0.2,0.04)

grid.newpage()
grid.bezier(x1,y1)
grid.bezier(x2,y2)
grid.bezier(x3,y3)
grid.bezier(x4,y4)
grid.bezier(x5,y5)
```