Taller Interpolacion

Nicolás Camacho PlazasDaniela Cortes Antonio Mateo Florido Sanchez

Primer punto:

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

En un intervalo $[a, b] \in R$ y $x_0, x_1, \ldots, x_n, n+1$ puntos distintos que se encuentran en el intervalo $[a, b]$ existe un **ÚNICO** polinomio de grado n o menor que satisface lo siguiente:

$$P_n(x_i) = f(x_i)$$

$$P_n(x) = \sum_{i=0}^{n} f\,(x_i) L_i(x)$$

Existencia:

$$L_i(x) = \prod_{i=1}^{n} \frac{x - x_j}{x_i - x_j}$$

Si para cada x_i existe un polinomio $L_i(x)$ (este polinomio siempre existe y además es continuo) de grado n, entonces existe un polinomio $P_n(x)$ de grado n como máximo.

Unicidad: Si suposieramos la existencia de dos polinomios distintos $P1_n(x)$ y $P2_n(x)$ de grado menor o igual a n y que son solución ,por que satisfacen $P1_n(xi) = f(x_i)$ y $P2_n(xi) = f(x_i)$,podriamos entonces suponer un nuevo polinomio $R_n(x) = P1_n(x) - P2_n(x)$ que tiene grado menor o igual a n y satisface $R_n(x) = 0$ para cada $i \in x_0, x_1, \ldots, x_n$,. Por tanto y según el teorema fundamental del algebra este polinomio $R_n(x)$ tiene solución y $n + 1$ raices por lo que podríamos concluir que $P1_n(x)$ y $P2_n(x)$ son el mismo : $P1_n(x) = P2_n(x)$

Segundo punto:

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

Tenemos tres puntos por los que queremos conocer el polinomio de grado tres que pasa por estos mismos tres, para reoslver utilizamos la función CubicSpline la cual nos permite enviarle los puntos a interpolar y las derivadas en estos mismos, para que se cumpla la condición que la tangente d x_0 sea igual a 1.

Para la interpolación se utlizaron los intervalos de [0,1] y de [1,2]. El primer polinomio es decir, para ñe primer intervalo dio un resultado de $10x^3 + 8.25x^2 + x - 4.25$ y para el segundo intervalo el polinomio que describe la curva es $15x^3 - 2.50x^2 - 11.75x + 4.25$, en la gráfica se pueden obseravar los puntos inicial y la curva que describe los dos polnomios de interpolación.

```
require(pracma)
```

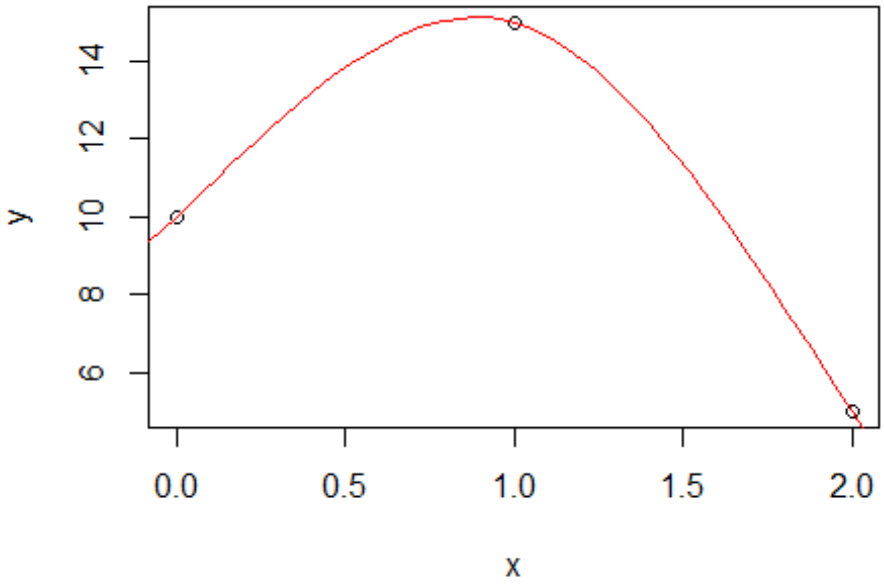
```
## Loading required package: pracma
```

```
library(pracma)

x <- c(0,1,2)
y <- c(10,15,5)

xi <- seq(0,2,by=1)

f <- cubicspline(x, y, xi = NULL, endp2nd = FALSE, der = c(1,1))
ffun <- function(xi) ppval(f,xi)
print(f)
```

```
## $breaks
## [1] 0 1 2
##
## $coefs
##       [,1]   [,2]  [,3] [,4]
## [1,] -4.25   1.00  8.25   10
## [2,]  4.25 -11.75 -2.50   15
##
## $pieces
## [1] 2
##
## $order
## [1] 4
##
## $dim
## [1] 1
##
## attr(,"class")
## [1] "pp"
```

```
plot(x,y, main="Figura 1. Interpolación Spline Cúbico")
plot(ffun, xlim=c(-1,3), add=TRUE, col = rainbow(1), main="Figura 1.
Interpolación Spline Cúbico")
```

Figura 1. Interpolación Spline Cúbico



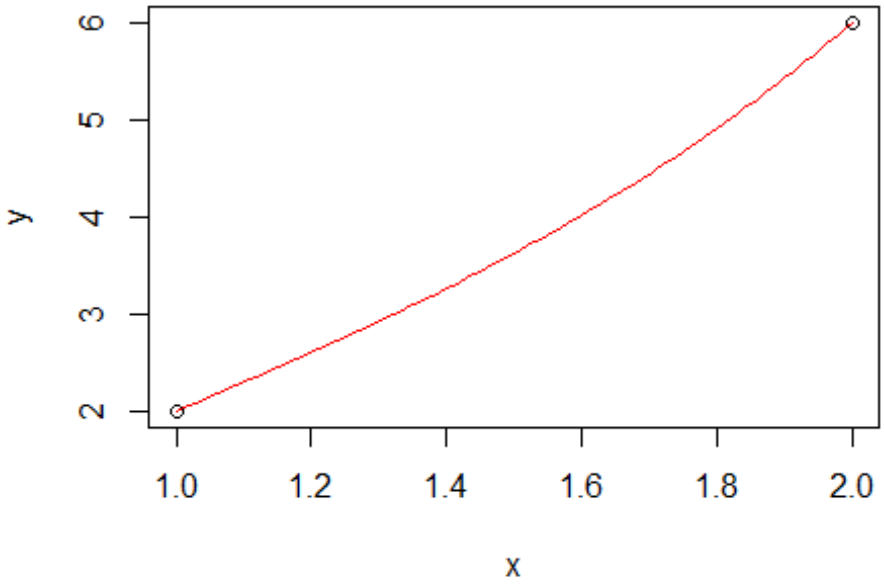
Tercer punto:

Se necesita construir un polinomio que interpole la función $f(x)$ dados los siguientes datos: $f(1) = 2\ f(2) = 6\ f'(1) = 3\ f'(2) = 7\ f'(2) = 8$ De acuerdo a esos datos, se crea la tabla de diferencias divididas: $f(1) = 2\ f'(1) = f[1,1] = 3\ f[1,1,2] = 1$ $f[1,1,2,2] = 2\ f[1,1,2,2,2] = -1\ f(1) = 2\ f[1,2] = 4\ f[1,2,2] = 3\ f[1,2,2,2] = 1$ $f(2) = 6\ f'(2) = f[2,2] = 7\ f[2,2,2] = 4\ f(2) = 6\ f[2,2] = 7\ f(2) = 6$

Ahora ya se puede encontrar el polinomio de interpolación por medio de la fórmula de Newton generalizada: $P(x) = f[1] + f[1,1](x-1) + f[1,1,2](x-1)^2 + f[1,1,2,2](x-1)^2(x-2) + f[1,1,2,2,2](x-1)^2(x-2)^2$ En donde los coeficientes son los elementos de la primera fila de la tabla de diferencias divididas junto con $(x - x_0)+\ldots+(x-x_0)^2(x-x_1)^2\ldots(x-x_n)$. Finalmente, al simplificar queda: $P(x) = x^3 - 3x^2 + 6x - 2$

```
x<-c(1,2)
y<-c(2,6)
plot(x,y,type="p", main = "Figura 2. Polinomio Interpolante ejercicio 3")
p<-function(x) x^3-3*x^2+6*x-2
plot(p,xlim = c(1,2),ylim = c(0,6),type="l",main = "Figura 2. Polinomio
Interpolante ejercicio 3",add=TRUE,col = rainbow(1))
```

Figura 2. Polinomio Interpolante ejercicio 3



Cuarto Punto:

Con la función $f(x) = lnx$ se debe construir la interpolación de diferencias ddivididas en x_0 y en x_1 y estimar el error en $[1,2] : x_0 = 1 ; x_1 = 2$ Con dos puntos se genera un polinomio interpolante de a lo sumo primer grado. Este está dado por la siguiente ecuación: $P(x) = f[x_0] + f[x_0, x_1](x - x_0)$ De esta forma entonces el polinomio está dado por: $P(x) = y_0 + \frac{(y_0 - y_1)}{(x_0 - x_1)}(x - x_0)$ Al reemplazar, el polinomio interpolante es: $P(x) = ln(1) + \frac{(ln(1) - ln(2))}{1-2}(x - 1)$ Como tenemos dos puntos (x_0, x_1) con $x_0 < x_1$ el error de la interpolación lineal está dado por: $f(x) - P(x) = \frac{(x-x_0)(x-x1)}{2} f''(\xi(x))$

Entonces el error está acotado por: $|f(x) - P(x)| <= M_2 \frac{(x_1 - x_0)^2}{8}$ por lo tanto $|f(x) - P(x)| <= 1/8$

```
P<-function(x) log(1)+((log(1)-log(2))/(1-2))*(x-1);
print(abs(log(1.5)-P(1.5)))
```

```
## [1] 0.05889152
```

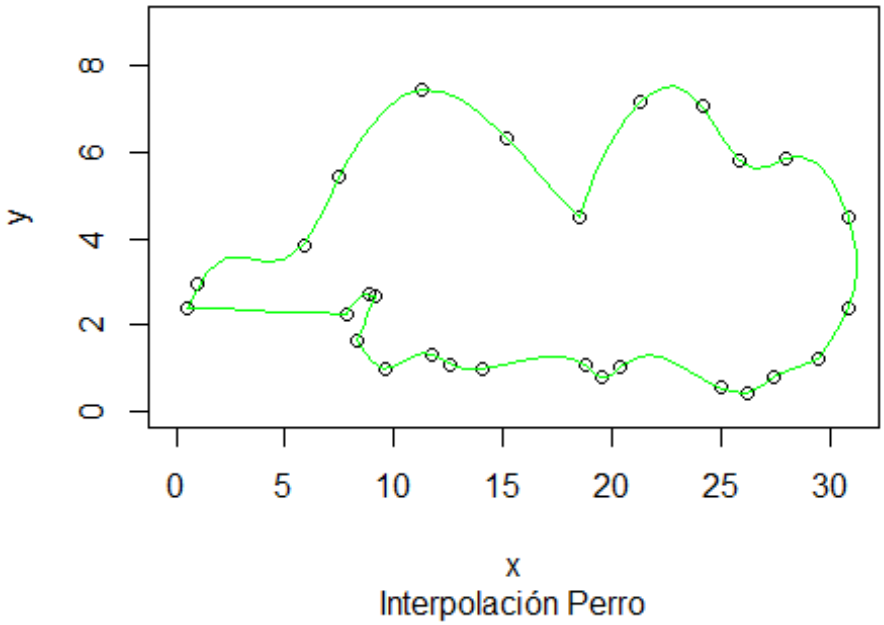
Luego, en el caso de $x = 1.5$ el error es de 0.05889152.

Quinto Punto

Perro Interpolado

```
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 , 30.81 , 29.40 , 27.40 , 26.21 , 24.97 , 20.32 , 19.54 ,
18.80 , 14.04 , 12.54
```

```
       , 11.68 , 09.55 , 08.30 , 09.10 , 08.85 , 07.80 , 00.50)

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 , 02.40 , 01.20 , 00.80 , 00.44 , 00.54 , 01.01 , 00.80 ,
01.08 , 00.98 , 01.08
       , 01.33 , 01.00 , 01.64 , 02.65 , 02.70 , 02.24 , 02.40)
segx1=x[1:7]
segx2 = x[7:12]
segx3 = x[12:14]
segx4 = x[14:15]
segx5 = x[15:18]
segx6 = x[18:20]
segx7 = x[20:25]
segx8 = x[25:26]
segx9 = x[26:28]
segx10 = x[28:29]
segy1 = y[1:7]
segy2 = y[7:12]
segy3 = y[12:14]
segy4 = y[14:15]
segy5 = y[15:18]
segy6 = y[18:20]
segy7 = y[20:25]
segy8 = y[25:26]
segy9 = y[26:28]
segy10 = y[28:29]
seg = spline(segy3,segx3)
i = seg$x
seg$x = seg$y
seg$y = i
plot(x, y,sub="Interpolación Perro",xlim=c(0,31),ylim=c(0,9))
lines(spline(segx1, segy1), col = "green",xlim=c(0,31),ylim=c(0,9))
lines(spline(segx2, segy2), col = "green",xlim=c(0,31),ylim=c(0,9))
lines(spline(segx4, segy4), col = "green",xlim=c(0,31),ylim=c(0,9))
lines(spline(segx5, segy5), col = "green",xlim=c(0,31),ylim=c(0,9))
lines(spline(segx6, segy6), col = "green",xlim=c(0,31),ylim=c(0,9))
lines(spline(segx7, segy7), col = "green",xlim=c(0,31),ylim=c(0,9))
lines(spline(segx8, segy8), col = "green",xlim=c(0,31),ylim=c(0,9))
lines(spline(segx9, segy9), col = "green",xlim=c(0,31),ylim=c(0,9))
lines(spline(segx10, segy10), col = "green",xlim=c(0,31),ylim=c(0,9))
lines(seg, col = "green")
```



Mano interpolada

```
library(Matrix)

##
## Attaching package: 'Matrix'

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

library(grid)
library(gridBezier)

#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, 15.1, 15.0, 14.9, 14.6, 14.3, 14.0,
13.9, 13.8, 13.5, 13.1, 13.0, 13.3, 13.2, 13.1, 12.9, 12.4, 11.9, 11.7,
11.6, 11.3, 10.9, 10.7, 10.6, 10.6, 10.1, 9.7, 9.4, 9.3, 9.6, 9.9, 10.1,
10.2, 10.3, 9.10, 8.6, 7.5, 7.0, 6.7, 6.6, 7.70, 8.00, 8.10, 8.40, 9.00,
9.30, 10, 10.2, 10.3, 10.0, 9.50)
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, 5.50, 5.00, 4.70, 4.60, 4.50, 4.90,
5.40, 5.80, 6.90, 8.20, 7.60, 5.80, 4.50, 4.30, 3.90, 4.20, 5.70, 7.00,
7.90, 8.20, 7.30, 6.70, 5.50, 5.10, 4.60, 4.7, 5.0, 5.5, 7.2, 7.8, 8.60,
9.40, 10.0, 10.7, 9.9, 9.0, 9.1, 9.3, 9.7, 11.7, 12.3, 12.5, 13.0, 13.9,
14.9, 16, 16.4, 16.8, 10.7, 11.0)
```

```
x=x/100
y=y/100
print(length(x))

## [1] 67

DatosX = x[1:4]; DatosY = y[1:4]
grid.newpage()
grid.bezier(DatosX, DatosY)
DatosX1 = x[4:7]; DatosY1 = y[4:7]
grid.bezier(DatosX1, DatosY1)
DatosX1 = x[7:10]; DatosY1 = y[7:10]
grid.bezier(DatosX1, DatosY1)
DatosX1 = x[10:13]; DatosY1 = y[10:13]
grid.bezier(DatosX1, DatosY1)
DatosX1 = x[13:16]; DatosY1 = y[13:16]
grid.bezier(DatosX1, DatosY1)
DatosX1 = x[16:19]; DatosY1 = y[16:19]
grid.bezier(DatosX1, DatosY1)
DatosX1 = x[19:22]; DatosY1 = y[19:22]
grid.bezier(DatosX1, DatosY1)
DatosX1 = x[22:25]; DatosY1 = y[22:25]
grid.bezier(DatosX1, DatosY1)
DatosX1 = x[25:28]; DatosY1 = y[25:28]
grid.bezier(DatosX1, DatosY1)
DatosX1 = x[28:31]; DatosY1 = y[28:31]
grid.bezier(DatosX1, DatosY1)
DatosX1 = x[31:34]; DatosY1 = y[31:34]
grid.bezier(DatosX1, DatosY1)
DatosX1 = x[34:37]; DatosY1 = y[34:37]
grid.bezier(DatosX1, DatosY1)
DatosX1 = x[37:40]; DatosY1 = y[37:40]
grid.bezier(DatosX1, DatosY1)
DatosX1 = x[40:43]; DatosY1 = y[40:43]
grid.bezier(DatosX1, DatosY1)
DatosX1 = x[43:46]; DatosY1 = y[43:46]
grid.bezier(DatosX1, DatosY1)
DatosX1 = x[46:49]; DatosY1 = y[46:49]
grid.bezier(DatosX1, DatosY1)
DatosX1 = x[49:52]; DatosY1 = y[49:52]
grid.bezier(DatosX1, DatosY1)
DatosX1 = x[49:52]; DatosY1 = y[49:52]
grid.bezier(DatosX1, DatosY1)
DatosX1 = x[52:55]; DatosY1 = y[52:55]
grid.bezier(DatosX1, DatosY1)
DatosX1 = x[55:58]; DatosY1 = y[55:58]
grid.bezier(DatosX1, DatosY1)
DatosX1 = x[58:61]; DatosY1 = y[58:61]
grid.bezier(DatosX1, DatosY1)
```

```
DatosX1 = x[61:64]; DatosY1 = y[61:64]
grid.bezier(DatosX1, DatosY1)
```

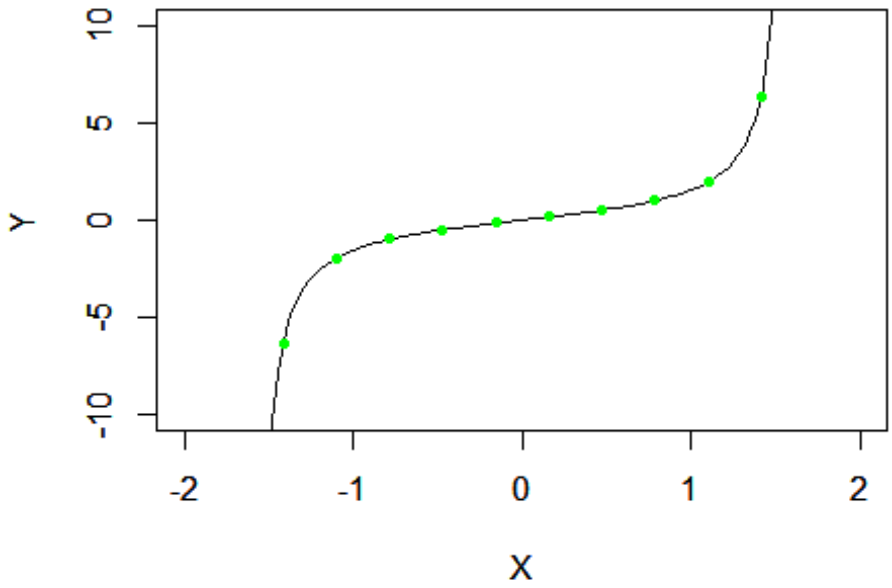
Sexto Punto

```
require(PolynomF)

## Loading required package: PolynomF

##
## Attaching package: 'PolynomF'

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

require(pracma)
tanf=function(x){
  i=1
  resul=c()
  while(i<=length(x)){
    resul[i]=tan(x[i])
    i=i+1
  }
  return(resul)
}
prom=function(errores){
  i=1
  resul=0
  suma=0
  while(i<=length(errores)){
    suma=suma+errores[i]
    i=i+1
  }
  resul=suma/length(errores)
  return (resul)
}
maxIter=10
x=c(1)
y=c(1)
y2=c()
errorA=100000000
errroN=0
ini=1
interpol=poly_calc(x,y)
```

```
i=0
errores=c()
iter=1
while(iter<=maxIter){
  print(iter)
  ini=ini-0.05
  if(ini<=0) break;
  x=seq((-pi/2)*ini,(pi/2)*ini,length=10)
  y=tanf(x)
  interpol=poly_calc(x,y)
  i=1
  while(i<=length(x)){
    y2[i]=interpol(x[i])
    i=i+1

  }
  i=1
  while(i<=length(y)){

    errores[i]=abs((y2[i]-y[i])/y[i])*100
    i=i+1
  }
  errorN=prom(errores)
  print(errorN)
  if (errorA >errorN) {
    errorA=errorN

  }else if(errorA<=errorN) {
    break;
    }

  iter=iter+1
}
```

```
## [1] 1
## [1] 3.778076e-14
## [1] 2
## [1] 1.47001e-12
```

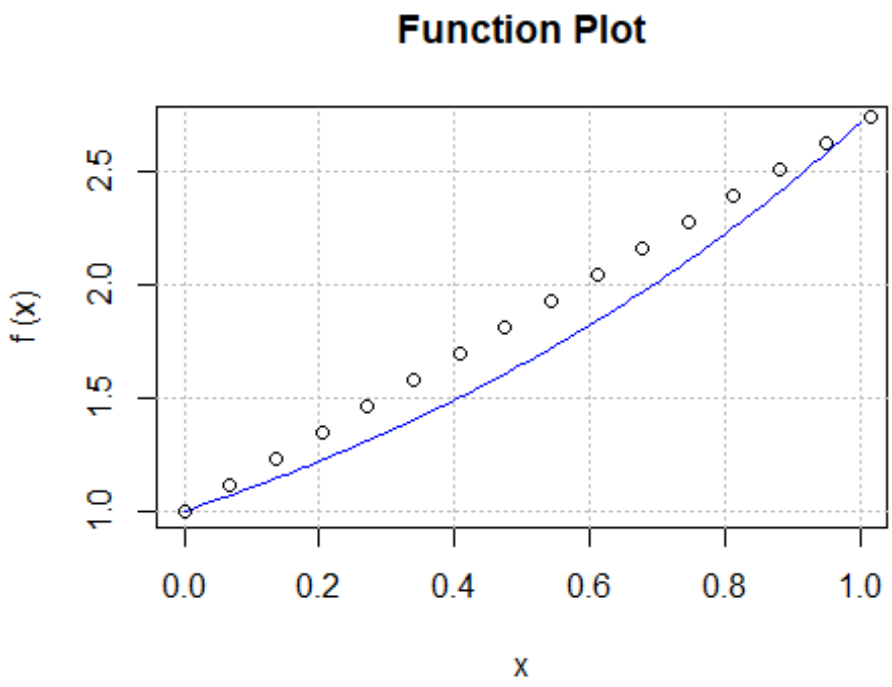
```
x_curve=seq(-pi/2,pi/2,length=100)
y_curve=tan(x_curve)
plot(x_curve,y_curve,type="l",xlim=c(-2,2),ylim=c(-10,10),xlab=
"X",ylab="Y")
par(new=TRUE)
plot(x,y2,pch=20,col="green",xlim=c(-2,2),ylim=c(-10,10),xlab=
"X",ylab="Y",sub=paste("Puntos interpolados para tan(x)","delta optimo
de",ini))
```



Puntos interpolados para tan(x) delta optimo de 0.9

Séptimo Punto

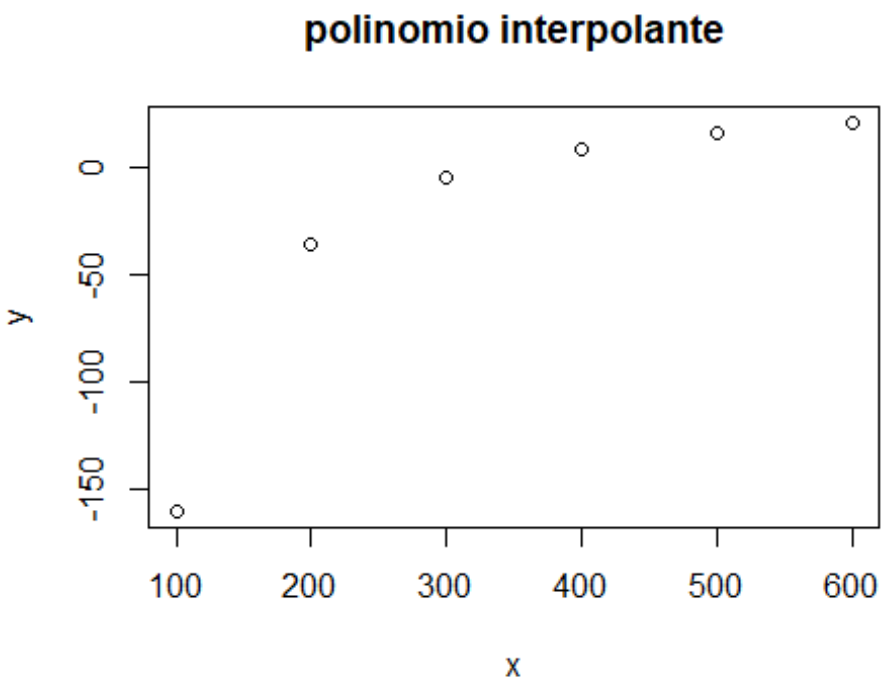
```
require(pracma)
library(pracma)
fp <- function(x) exp(x)
x <- 0:1; y <- fp(x)
xx <- linspace(0, 4, 60)
yy <- lagrangeInterp(x, y, xx)
print(yy)

## <0 x 0 matrix>

yy <- newtonInterp(x, y, xx)
print(yy)

##  [1] 1.000000 1.116494 1.232987 1.349481 1.465975 1.582468 1.698962
##  [8] 1.815456 1.931949 2.048443 2.164937 2.281431 2.397924 2.514418
## [15] 2.630912 2.747405 2.863899 2.980393 3.096886 3.213380 3.329874
## [22] 3.446367 3.562861 3.679355 3.795848 3.912342 4.028836 4.145329
## [29] 4.261823 4.378317 4.494810 4.611304 4.727798 4.844292 4.960785
## [36] 5.077279 5.193773 5.310266 5.426760 5.543254 5.659747 5.776241
## [43] 5.892735 6.009228 6.125722 6.242216 6.358709 6.475203 6.591697
## [50] 6.708190 6.824684 6.941178 7.057672 7.174165 7.290659 7.407153
## [57] 7.523646 7.640140 7.756634 7.873127

## Not run:
ezplot(fp, 0, 1)
points(xx, yy)
```




```
## End(Not run)
```

Octavo Punto

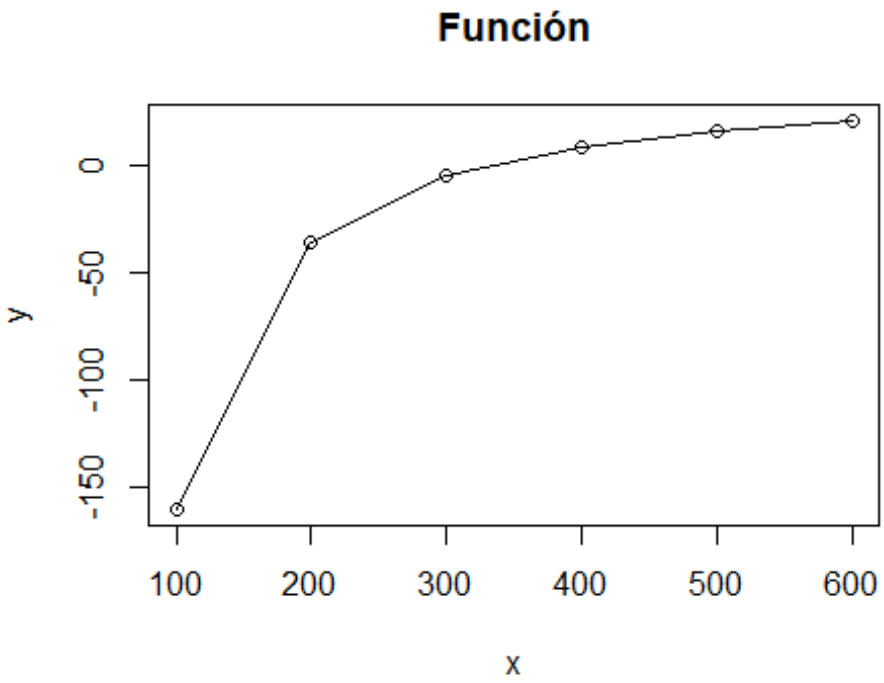
```
require(pracma)
library(pracma)

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)

f <- cubicspline(x, y, xi = NULL, endp2nd = FALSE, der = c(100,700))
ffun <- function(xi) ppval(f,xi)
x1<- c(400)
y1<-ffun(x1)
plot(x,y, main=" polinomio interpolante ")
plot(ffun, xlim=c(-1,3), add=TRUE, col = rainbow(1), main="Interpolación
")
```




```
print("Segundo y tercer coeficiente virial a 450K, es: 13.5")
```

```
## [1] "Segundo y tercer coeficiente virial a 450K, es: 13.5"
```

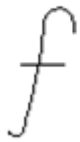
```
xi <- seq(100,600,by=100)
f <- cubicspline(x, y, xi = NULL, endp2nd = FALSE, der = c(100,700))
ffun <- function(xi) ppval(f,xi)
plot(x,y, main=" Función ",type="o")
```



Letras con Interpolacion de Bezier

Letra de Nicolas Camacho

```
library(gridBezier)

x<-c(0.0125,0.02,0.031,0.03)
y<-c(0.015,0,0.01,0.03)
x1<-c(0.03,0.04,0.05,0.05)
y1<-c(0.03,0.06,0.14,0.16)
x2<-c(0.05,0.06,0.08,0.081)
y2<-c(0.16,0.18,0.18,0.142)

grid.newpage()
grid.segments(.025, .1, .07, .1)
grid.bezier(x,y)
grid.bezier(x1,y1)
grid.bezier(x2,y2)
```



Letra de Daniela Cortes

```
library(grid)
library(gridBezier)
x <- c(0.4, 0.47, 0.47, 0.43)
y <- c(0.6, 0.6, 0.62, 0.65)
grid.bezier(x, y)
grid.bezier(x, y,
            gp=gpar(lwd=3, fill="black"),
            arrow=NULL)

x <- c(0.2, 0, 0, 0.4)
y <- c(0.4, 0.4, 0.7, 0.6)
grid.bezier(x, y)
grid.bezier(x, y,
            gp=gpar(lwd=3, col = "gray", fill = "green3"),
            arrow=NULL)
x <- c(0.1, 0.5, 0.5, 0.2)
y <- c(0.2, 0.1, 0.4, 0.4)
grid.bezier(x, y)
grid.bezier(x, y,
            gp=gpar(lwd=3,  col = "gray", fill = "green3"),
            arrow=NULL)

x <- c(0.1, 0, 0, 0.13)
y <- c(0.2, 0.21, 0.27, 0.25)
grid.bezier(x, y)
grid.bezier(x, y,
            gp=gpar(lwd=3, fill="black"),
            arrow=NULL)

x <- c(0.23, 0.03, 0.03, 0.43)
y <- c(0.45, 0.45, 0.75, 0.65)
grid.bezier(x, y)

grid.bezier(x, y,
            gp=gpar(lwd=3, fill="black"),
            arrow=NULL)

x <- c(0.13, 0.53, 0.53, 0.23)
y <- c(0.25, 0.15, 0.45, 0.45)
grid.bezier(x, y)
grid.bezier(x, y,
```

```
            gp=gpar(lwd=3, fill="black"),
            arrow=NULL)
```

Letra de Mateo florido

```
library(grid)
library(gridBezier)

x <- c(0.2, 0, 0, 0.37)
y <- c(0.4, 0.4, 0.7, 0.6)
grid.bezier(x, y)
grid.bezier(x, y,
            gp=gpar(lwd=3, fill="black"),
            arrow=NULL)

x <- c(0.29, 0.42, 0.54, 0.37)
y <- c(0.56, 0.54, 0.56, 0.6)
grid.bezier(x, y)

grid.bezier(x, y,
            gp=gpar(lwd=3, fill="black"),
            arrow=NULL)

x <- c(0.37, 0.25, 0.2, 0.29)
y <- c(0.58, 0.62, 0.6, 0.56)
grid.bezier(x, y)
```

```
grid.bezier(x, y,
            gp=gpar(lwd=3, fill="black"),
            arrow=NULL)



x <- c(0.1, 0.5, 0.5, 0.2)
y <- c(0.2, 0.1, 0.4, 0.4)
grid.bezier(x, y)
grid.bezier(x, y,
            gp=gpar(lwd=3, fill="black"),
            arrow=NULL)

x <- c(0.13, 0, 0, 0.1)
y <- c(0.3, 0.4, 0.25, 0.2)
grid.bezier(x, y)

grid.bezier(x, y,
            gp=gpar(lwd=3, fill="black"),
            arrow=NULL)
```