INTERPOLACIÓN MEDIANTE BASES DE LAGRANGE EN R

De una función f(x) se conocen los siguientes valores. Calcula el polinomio interpolador mediante las bases de Lagrange, evalúalo en el punto Sk=1.5 y dibuja la función.

Datos:

Sk	0	1	2	3
F(sk)	4	8	-3	9.5

Variables de entrada

- Puntos de soporte: Sk
- La imagen de los puntos de soporte en f(x): F(Sk)
- Punto del soporte a evaluar el polinomio: 1.5

Programa: lagrange_POL_canonico.R

1. Definir el soporte y la imagen del soporte como vectores concatenados; e introducir el punto del polinomio a evaluar.
2. Llamar a la función lagrange_coefs
   - Toma los valores de soporte y calcula los coeficientes.
   - Obtenemos los coeficientes de los polinomios base Li(x)Li (x) de Lagrange y los guarda en una matriz.
3. Llamar a la función imprime_bases
   - Toma la matriz obtenida en el paso anterior y nos devuelve las bases en forma de polinomios.
4. Definir la función pol_inter_lagrange
   - Toma los valores de la base de lagrange calculadas y las imágenes del soporte, elabora el polinomio interpolador.
5. Imprimir el polinomio en su forma canónica llamando al comando imprime_pol_canonico
6. Utilizar la función evalua_pol_canonico
   - Toma el polinomio y el punto a evaluar (1.5) y nos muestra en la consola el valor aproximado.
7. Graficar el polinomio
   Emplear la función dibujar_canonicos
   - Selecciona los valores del soporte y el polinomio interpolador canónico
8. Definir rango
   - Acotar valor máximo y mínimo de x, es decir los límites de la gráfica.

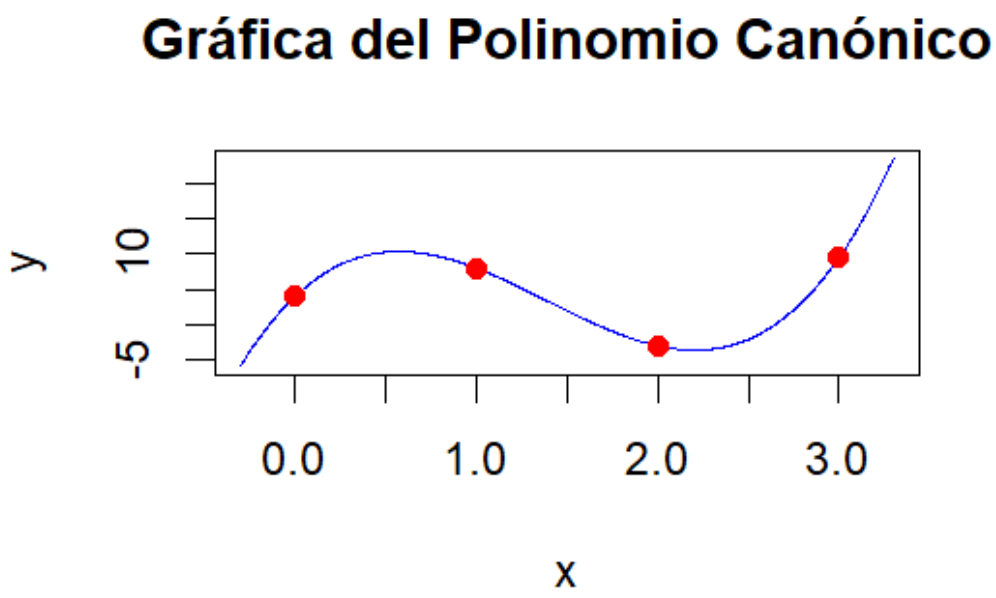
- Crea un vector de 1000puntos entre estos dos valores. El vector tiene mil puntos ya que se considera una extensión necesaria para obtener una curva continua suave definida.

El polinomio obtenido es: `4.0000 + 24.3333x^1-26.7500x^2 + 6.4167x^3`

Programa en R:

```r
cat("\014")
rm(list=ls())

soporte<-c(0,1,2,3)
fs<-c(4,8,-3,9.5)
puntos_evaluacion <-1.5

lagrange_coefs <- function(soporte) {
  n <- length(soporte) - 1
  coefs <- matrix(0, nrow = n + 1, ncol = n + 1)

  for (i in 0:n) {
    Li <- c(1)
    for (j in 0:n) {
      if (i != j) {
        # Multiplicar por (x - soporte[j+1])
        Li <- c(0, Li) - c(Li * soporte[j+1], 0)
        # Dividir por (soporte[i+1] - soporte[j+1])
        Li <- Li / (soporte[i+1] - soporte[j+1])
      }
    }

    coefs[i+1,] <- Li
  }

  return(coefs)
}
matriz_bases_Lagrange <- lagrange_coefs(soporte)

imprime_bases <-function(matriz_bases_Lagrange){
  for (j in 1: nrow(matriz_bases_Lagrange)){
    pol <- matriz_bases_Lagrange[j,]
    # Imprimir la ecuación del polinomio
    cat(sprintf("L_%d=",j-1))
    for (i in 1:length(pol)) {
      if (i > 1 && pol[i]>0) cat(" + ")
      cat(sprintf("%.4f", pol[i]))
      if (i > 1) cat(sprintf("x^%d", i-1))
    }
    cat("\n")
  }
}
```

```r

imprime_bases(matriz_bases_Lagrange)

pol_inter_lagrange <-function(matrizLs,fs){
  #---Calcula el polinomio interpolador a partir de las funciones de la base de Lagrange
  # Inicializo un polinomio del orden que indique la matriz
  pol_interp <- numeric(ncol(matrizLs))

  for (j in 1: nrow(matrizLs)){
    pol_interp <- pol_interp + matrizLs[j,]*fs[j]
  }
  return(pol_interp)
}

imprime_pol_canonico <- function(pol){
  for (i in 1:length(pol)) {
    if (i > 1 && pol[i]>=0) cat(" + ")
    cat(sprintf("%.4f", pol[i]))
    if (i > 1) cat(sprintf("x^%d", i-1))
  }
  cat("\n")
}

polinomio_interp <- pol_inter_lagrange(matriz_bases_Lagrange, fs)
imprime_pol_canonico(polinomio_interp)

evalua_pol_canonico <- function(pol, x_eval) {
  p <- function(x) {
    resultado <- pol[1]
    for (i in 2:length(pol)) {  # Agregado el paréntesis de cierre
      resultado <- resultado + pol[i] * x^(i - 1)
    }
    return(resultado)
  }

  return(sapply(x_eval, p))
}

valores_evaluados <- evalua_pol_canonico(polinomio_interp, puntos_evaluacion)
print(valores_evaluados)

dibujar_canonicos <- function(x_orig, coeficientes) {
  # Crear una función para el polinomio
  polinomio <- function(x) {
    suma <- 0
    for (i in 1:length(coeficientes)) {
      suma <- suma + coeficientes[i] * x^(i-1)
    }
    return(suma)
  }

  # Determinar el rango para x
  x_min <- min(x_orig)
  x_max <- max(x_orig)
  rango <- x_max - x_min

  # Crear un vector de valores x para una curva suave
  x <- seq(x_min - 0.1*rango, x_max + 0.1*rango, length.out = 1000)

  # Calcular los valores y correspondientes
  y <- sapply(x, polinomio)

  # Crear la gráfica
  plot(x, y, type = "l", col = "blue",
       main = "Gráfica del Polinomio Canónico",
       xlab = "x", ylab = "y",
       xlim = c(min(x), max(x)), ylim = range(y))

  # Añadir puntos para el soporte original
  points(x_orig, sapply(x_orig, polinomio), col = "red", pch = 19)
}
dibujar_canonicos(soporte, polinomio_interp)
```


Gráfica del Polinomio Canónico
y
-5
10
0.0
1.0
2.0
3.0
x