Universidad Nacional del Altiplano
Facultad de Ingeniería Estadística e Informática

E.P. de Ingeniería Estadística e Informática

Docente: Ing. Torres Cruz Fred

Presentado por: Quispe Ito Luz Leidy

Interpolacion

Metodos y Formulas de Alta Precision

¿Que es la Interpolación?

La interpolacion es un metodo matematico para calcular valores intermedios entre puntos conocidos. Se utiliza para rellenar datos faltantes, suavizar datos, predecir valores o crear nuevas series de datos a partir de las existentes.

Introducción

Este documento presenta la resolución aplicada y detallada de los métodos de interpolación más utilizados: Interpolación Lineal, Interpolación de Lagrange, Método de Newton con Diferencias Divididas, Interpolación Cuadrática y Splines Cúbicos. Cada ejercicio incluye solución y código en R

1. Interpolación Lineal

La fórmula general entre dos puntos (x_0, y_0) y (x_1, y_1) es:

$$y = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0)$$

Ejercicio 1

A las 8:00 la temperatura era $20°C$ y a las 12:00 era $28°C$. Estimar la temperatura a las 10:30.

$$y = 20 + \frac{28 - 20}{12 - 8}(10{,}5 - 8) = 25°C.$$

Resultado: $\boxed{25°C}$

```
x0 <- 8; y0 <- 20
x1 <- 12; y1 <- 28
x <- 10.5
y <- y0 + (y1-y0)/(x1-x0)*(x-x0)
y
```

Ejercicio 2

Nivel del agua: a las 6:00 es 3.2 m, a las 9:00 es 4.1 m. Estimar a las 7:30.

$$y = 3{,}2 + \frac{4{,}1 - 3{,}2}{9 - 6}(7{,}5 - 6) = 3{,}5667\ m$$

Resultado: $\boxed{\approx 3{,}57\ m}$

```
x0 <- 6; y0 <- 3.2
x1 <- 9; y1 <- 4.1
x <- 7.5
y <- y0 + (y1-y0)/(x1-x0)*(x-x0)
y
```

Ejercicio 3

Precio por lote: 100 unidades $\rightarrow$ 1200; 200 unidades $\rightarrow$ 2100. Calcular precio para 150 unidades.

$$y = 1200 + \frac{2100 - 1200}{200 - 100}(150 - 100) = 1650.$$

Resultado: $\boxed{1650}$

```
x0 <- 100; y0 <- 1200
x1 <- 200; y1 <- 2100
x <- 150
y <- y0 + (y1-y0)/(x1-x0)*(x-x0)
y
```

2. Interpolación de Lagrange

Fórmula

$$P(x) = \sum_{i=0}^{n} y_i L_i(x), \quad L_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

Ejercicio 1

Datos: $(1, 2)$, $(3, 10)$, $(4, 17)$, $x = 2$

$$L_0(2) = \frac{(2-3)(2-4)}{(1-3)(1-4)} = \frac{2}{6} = \frac{1}{3}$$

$$L_1(2) = \frac{(2-1)(2-4)}{(3-1)(3-4)} = \frac{-2}{-2} = 1$$

$$L_2(2) = \frac{(2-1)(2-3)}{(4-1)(4-3)} = -\frac{1}{3}$$

$$P(2) = 2 \cdot \frac{1}{3} + 10 \cdot 1 + 17 \cdot \left(-\frac{1}{3}\right) = 5$$

Resultado: $P(2) = 5$

```r
x_points <- c(1, 3, 4)
y_points <- c(2, 10, 17)
x <- 2

result <- lagrange_interpol(x_points, y_points, x)
cat("P(2) =", result, "\n")
```

Ejercicio 2: Velocidad de objeto

Datos: $(0, 0)$, $(2, 12)$, $(4, 20)$, $x = 3$

$$L_0(3) = \frac{(3-2)(3-4)}{(0-2)(0-4)} = \frac{-1}{8} = -0{,}125$$

$$L_1(3) = \frac{(3-0)(3-4)}{(2-0)(2-4)} = \frac{-3}{-4} = 0{,}75$$

$$L_2(3) = \frac{(3-0)(3-2)}{(4-0)(4-2)} = \frac{3}{8} = 0{,}375$$

$$P(3) = 0 \cdot (-0{,}125) + 12 \cdot 0{,}75 + 20 \cdot 0{,}375 = 16{,}5$$

Resultado: 16.5 m/s

```r
t_points <- c(0, 2, 4)
v_points <- c(0, 12, 20)
t <- 3

velocidad <- lagrange_interpol(t_points, v_points, t)
cat("Velocidad en t=3s:", velocidad, "m/s\n")
```

Ejercicio 3: Temperatura en placa

Datos: $(0, 100)$, $(2, 110)$, $(3, 95)$, $x = 1$

$$L_0(1) = \frac{(1-2)(1-3)}{(0-2)(0-3)} = \frac{2}{6} = \frac{1}{3}$$

$$L_1(1) = \frac{(1-0)(1-3)}{(2-0)(2-3)} = \frac{-2}{-2} = 1$$

$$L_2(1) = \frac{(1-0)(1-2)}{(3-0)(3-2)} = -\frac{1}{3}$$

$$P(1) = 100 \cdot \frac{1}{3} + 110 \cdot 1 + 95 \cdot \left(-\frac{1}{3}\right) = 111{,}67$$

Resultado: 111.67°C

```r
# Para punto (1,2) - asumiendo coordenadas x
x_points <- c(0, 2, 3)
y_points <- c(100, 110, 95)
x <- 1

temp <- lagrange_interpol(x_points, y_points, x)
cat("Temperatura en (1,2):", round(temp, 1), " C \n")
```

3. Divididas de Newton

Fórmula

$$P(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \cdots$$

```r
newton_interpol <- function(x, y, punto_eval){
  n <- length(x)

  # Crear la tabla de diferencias divididas
  tabla <- matrix(0, n, n)
  tabla[,1] <- y
```

```r

  for(j in 2:n){
    for(i in 1:(n-j+1)){
      tabla[i,j] <- (tabla[i+1,j-1] - tabla[i,j-1]) / (x[i+j-1] - x[i])
    }
  }

  # Evaluar el polinomio en punto_eval usando forma de Newton
  resultado <- tabla[1,1]
  prod_term <- 1
  for(j in 2:n){
    prod_term <- prod_term * (punto_eval - x[j-1])
    resultado <- resultado + tabla[1,j] * prod_term
  }

  return(list(result = resultado, tabla = tabla))
}
```

Ejercicio 1

Datos: $(1,1)$, $(2,4)$, $(3,9)$, $x = 2{,}5$

$$f[x_0] = 1, \quad f[x_1] = 4, \quad f[x_2] = 9$$

$$f[x_0, x_1] = \frac{4-1}{2-1} = 3, \quad f[x_1, x_2] = \frac{9-4}{3-2} = 5$$

$$f[x_0, x_1, x_2] = \frac{5-3}{3-1} = 1$$

$$P(2{,}5) = 1 + 3(2{,}5-1) + 1(2{,}5-1)(2{,}5-2) = 6{,}25$$

Resultado: $P(2{,}5) = 6{,}25$

```r
x_points <- c(1, 2, 3)
y_points <- c(1, 4, 9)

resultado <- newton_interpol(x_points, y_points, 2.5)
cat("P(2.5) =", resultado$result, "\n")
cat("Tabla de diferencias:\n")
print(resultado$tabla)
```

Ejercicio 2

Datos: $(0,1)$, $(1,2)$, $(2,0)$, $x = 1{,}5$

$$f[x_0] = 1, \quad f[x_1] = 2, \quad f[x_2] = 0$$

$$f[x_0, x_1] = \frac{2-1}{1-0} = 1, \quad f[x_1, x_2] = \frac{0-2}{2-1} = -2$$
$$f[x_0, x_1, x_2] = \frac{-2-1}{2-0} = -1{,}5$$
$$P(1{,}5) = 1 + 1(1{,}5-0) + (-1{,}5)(1{,}5-0)(1{,}5-1) = 1{,}375$$

Resultado: $P(1{,}5) = 1{,}375$

```
x_points <- c(0, 1, 2)
y_points <- c(1, 2, 0)

resultado <- newton_interpol(x_points, y_points, 1.5)
cat("P(1.5) =", resultado$result, "\n")
```

Ejercicio 3: Altura de proyectil

Datos: $(1, 40)$, $(2, 60)$, $(3, 50)$, $x = 2{,}5$

$$f[x_0] = 40, \quad f[x_1] = 60, \quad f[x_2] = 50$$
$$f[x_0, x_1] = \frac{60-40}{2-1} = 20, \quad f[x_1, x_2] = \frac{50-60}{3-2} = -10$$
$$f[x_0, x_1, x_2] = \frac{-10-20}{3-1} = -15$$
$$P(2{,}5) = 40 + 20(2{,}5-1) + (-15)(2{,}5-1)(2{,}5-2) = 58{,}75$$

Resultado: 58.75 m

```
t_points <- c(1, 2, 3)
h_points <- c(40, 60, 50)

resultado <- newton_interpol(t_points, h_points, 2.5)
cat("Altura en t=2.5s:", resultado$result, "m\n")
```

4. Interpolación Cuadrática

Formula

Para tres puntos, buscamos $P(x) = ax^2 + bx + c$ que pase por los puntos. Resolvemos el sistema:

$$\begin{bmatrix} x_0^2 & x_0 & 1 \\ x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}$$

```r
interpolacion_cuadratica <- function(x_points, y_points, x) {
  # Construir matriz del sistema
  A <- matrix(c(
    x_points[1]^2, x_points[1], 1,
    x_points[2]^2, x_points[2], 1,
    x_points[3]^2, x_points[3], 1
  ), nrow = 3, byrow = TRUE)

  b <- matrix(y_points, ncol = 1)

  # Resolver sistema
  coef <- solve(A, b)

  # Evaluar en x
  y <- coef[1]*x^2 + coef[2]*x + coef[3]

  return(list(y = y, coef = coef))
}
```

Ejercicio 1

Datos: $(1,1)$, $(2,8)$, $(3,27)$

$$\begin{cases} a+b+c=1 \\ 4a+2b+c=8 \\ 9a+3b+c=27 \end{cases} \Rightarrow \begin{cases} 3a+b=7 \\ 5a+b=19 \end{cases} \Rightarrow a=6, b=-11, c=6$$

$$P(x) = 6x^2 - 11x + 6, \quad P(2{,}5) = 16$$

Resultado: $P(2{,}5) = 16$

```r
x_points <- c(1, 2, 3)
y_points <- c(1, 8, 27)

resultado <- interpolacion_cuadratica(x_points, y_points, 2.5)
cat("P(2.5) =", resultado$y, "\n")
cat("Coeficientes (a, b, c):", resultado$coef, "\n")
```

Ejercicio 2: Caída libre

Datos: $(0,0)$, $(1,-4{,}9)$, $(2,-19{,}6)$

$$\begin{cases} c=0 \\ a+b+c=-4{,}9 \\ 4a+2b+c=-19{,}6 \end{cases} \Rightarrow \begin{cases} a+b=-4{,}9 \\ 4a+2b=-19{,}6 \end{cases} \Rightarrow a=-4{,}9, b=0$$

$$P(x) = -4{,}9x^2, \quad P(1{,}5) = -11{,}025$$

Resultado: -11.025 m

```r
t_points <- c(0, 1, 2)
y_points <- c(0, -4.9, -19.6)

resultado <- interpolacion_cuadratica(t_points, y_points, 1.5)
cat("Posicion en t=1.5s:", resultado$y, "m\n")
```

Ejercicio 3: Circuito eléctrico

Datos: $(1, 5)$, $(2, 11)$, $(3, 19)$

$$\begin{cases} a+b+c=5 \\ 4a+2b+c=11 \\ 9a+3b+c=19 \end{cases} \Rightarrow \begin{cases} 3a+b=6 \\ 5a+b=8 \end{cases} \Rightarrow a=1, b=3, c=1$$

$$P(x) = x^2 + 3x + 1, \quad P(2{,}5) = 14{,}75$$

Resultado: 14.75 V

```r
I_points <- c(1, 2, 3)  # Corriente
V_points <- c(5, 11, 19) # Voltaje

resultado <- interpolacion_cuadratica(I_points, V_points, 2.5)
cat("Voltaje para I=2.5A:", resultado$y, "V\n")
```

5. Splines Cúbicos

Formula

Un spline cúbico es una función suave formada por polinomios de grado 3 en cada intervalo. Para el intervalo $[x_i, x_{i+1}]$:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

Ejercicio 1

Datos: $(1, 2)$, $(2, 4)$, $(3, 1)$

Coeficientes por intervalo:

Intervalo	a_i	b_i	c_i	d_i
$[1,2]$	$a_0 = 2$	$b_0 = \frac{5}{3}$	$c_0 = 0$	$d_0 = -\frac{2}{3}$
$[2,3]$	$a_1 = 4$	$b_1 = -\frac{4}{3}$	$c_1 = -2$	$d_1 = \frac{2}{3}$

Polinomios por tramos:

$$S_0(x) = 2 + \frac{5}{3}(x-1) - \frac{2}{3}(x-1)^3, \quad x \in [1,2]$$
$$S_1(x) = 4 - \frac{4}{3}(x-2) - 2(x-2)^2 + \frac{2}{3}(x-2)^3, \quad x \in [2,3]$$

Evaluación en $x = 2{,}5$ (usa S_1):

$$\begin{aligned} S(2{,}5) &= 4 - \frac{4}{3}(0{,}5) - 2(0{,}5)^2 + \frac{2}{3}(0{,}5)^3 \\ &= 4 - 0{,}667 - 0{,}5 + 0{,}083 = 2{,}916 \end{aligned}$$

$$\boxed{S(2{,}5) = 2{,}916}$$

```r
# EJERCICIO 1: (1,2), (2,4), (3,1)
ejercicio1 <- function() {
  # Definir splines manualmente
  S0 <- function(x) { 2 + (5/3)*(x-1) - (2/3)*(x-1)^3 }
  S1 <- function(x) { 4 - (4/3)*(x-2) - 2*(x-2)^2 + (2/3)*(x-2)^3 }

  x_eval <- 2.5
  resultado <- S1(x_eval)

  cat("=== EJERCICIO 1 ===\n")
  cat("Datos: (1,2), (2,4), (3,1)\n")
  cat("S(2.5) =", resultado, "\n")

  # Verificar con funci n spline de R
  x_points <- c(1, 2, 3)
  y_points <- c(2, 4, 1)
  spline_r <- spline(x_points, y_points, n = 100)
  idx <- which.min(abs(spline_r$x - x_eval))
  cat("Verificaci n con spline() de R:", spline_r$y[idx], "\n")

  return(resultado)
}

result1 <- ejercicio1()
```

Listing 1: Codigo R para Ejercicio 1

Ejercicio 2

Datos: $(0,1)$, $(1,3)$, $(2,0)$

Coeficientes por intervalo:

Intervalo	a_i	b_i	c_i	d_i
$[0,1]$	$a_0 = 1$	$b_0 = \frac{7}{4}$	$c_0 = 0$	$d_0 = -\frac{3}{4}$
$[1,2]$	$a_1 = 3$	$b_1 = -\frac{5}{4}$	$c_1 = -\frac{9}{4}$	$d_1 = \frac{3}{4}$

Polinomios por tramos:

$$S_0(x) = 1 + \frac{7}{4}x - \frac{3}{4}x^3, \quad x \in [0,1]$$
$$S_1(x) = 3 - \frac{5}{4}(x-1) - \frac{9}{4}(x-1)^2 + \frac{3}{4}(x-1)^3, \quad x \in [1,2]$$

Evaluación en $x = 1{,}5$ (usa S_1):

$$\begin{aligned} S(1{,}5) &= 3 - \frac{5}{4}(0{,}5) - \frac{9}{4}(0{,}5)^2 + \frac{3}{4}(0{,}5)^3 \\ &= 3 - 1{,}25 - 1{,}125 + 0{,}1875 = 0{,}8125 \end{aligned}$$

$$\boxed{S(1{,}5) = 0{,}8125}$$

```r
# EJERCICIO 2: (0,1), (1,3), (2,0)
ejercicio2 <- function() {
  # Definir splines manualmente
  S0 <- function(x) { 1 + (7/4)*x - (3/4)*x^3 }
  S1 <- function(x) { 3 - (5/4)*(x-1) - (9/4)*(x-1)^2 + (3/4)*(x-1)^3 }

  x_eval <- 1.5
  resultado <- S1(x_eval)

  cat("\n=== EJERCICIO 2 ===\n")
  cat("Datos: (0,1), (1,3), (2,0)\n")
  cat("S(1.5) =", resultado, "\n")

  # Verificar con funci n spline de R
  x_points <- c(0, 1, 2)
  y_points <- c(1, 3, 0)
  spline_r <- spline(x_points, y_points, n = 100)
  idx <- which.min(abs(spline_r$x - x_eval))
  cat("Verificaci n con spline() de R:", spline_r$y[idx], "\n")

  return(resultado)
}

result2 <- ejercicio2()
```

Listing 2: Codigo R para Ejercicio 2

Ejercicio 3

Datos: $(2,5)$, $(3,2)$, $(5,6)$

Coeficientes por intervalo:

Intervalo	a_i	b_i	c_i	d_i
$[2,3]$	$a_0 = 5$	$b_0 = -\frac{10}{3}$	$c_0 = 0$	$d_0 = \frac{1}{3}$
$[3,5]$	$a_1 = 2$	$b_1 = \frac{11}{6}$	$c_1 = \frac{3}{2}$	$d_1 = -\frac{1}{6}$

Polinomios por tramos:

$$S_0(x) = 5 - \frac{10}{3}(x-2) + \frac{1}{3}(x-2)^3, \quad x \in [2,3]$$
$$S_1(x) = 2 + \frac{11}{6}(x-3) + \frac{3}{2}(x-3)^2 - \frac{1}{6}(x-3)^3, \quad x \in [3,5]$$

Evaluación en $x = 4$ (usa S_1):

$$\begin{aligned} S(4) &= 2 + \frac{11}{6}(1) + \frac{3}{2}(1)^2 - \frac{1}{6}(1)^3 \\ &= 2 + 1{,}833 + 1{,}5 - 0{,}167 = 5{,}166 \end{aligned}$$

$$\boxed{S(4) = 5{,}166}$$

6. Error de Interpolación

Fórmula General

$$|f(x) - P_n(x)| \leq \frac{M_{n+1}}{(n+1)!} \left| \prod_{i=0}^{n} (x - x_i) \right|$$

donde:

- $M_{n+1} = \max\limits_{\xi \in [a,b]} |f^{(n+1)}(\xi)|$
- n es el grado del polinomio interpolante
- x_i son los nodos de interpolación

Ejercicio 1: Error para $\ln(x)$

Datos: $x_0 = 1$, $x_1 = 2$, $x_2 = 3$, $x = 1{,}5$, $f(x) = \ln(x)$

Solución:

$$f'''(x) = \frac{2}{x^3}$$

$$M_3 = \max_{[1,3]} |f'''(x)| = \frac{2}{1^3} = 2$$

$$\prod_{i=0}^{2} |1{,}5 - x_i| = |1{,}5 - 1| \cdot |1{,}5 - 2| \cdot |1{,}5 - 3|$$

$$= 0{,}5 \cdot 0{,}5 \cdot 1{,}5 = 0{,}375$$

$$\text{Error} \leq \frac{2}{3!} \cdot 0{,}375 = \frac{2}{6} \cdot 0{,}375 = 0{,}125$$

Resultado: Error $\leq 0{,}125$

Ejercicio 2: Error para e^x

Datos: $x_0 = 0$, $x_1 = 1$, $x_2 = 2$, $x = 0{,}5$, $f(x) = e^x$

Solución:

$$f'''(x) = e^x$$

$$M_3 = \max_{[0,2]} |f'''(x)| = e^2 \approx 7{,}389$$

$$\prod_{i=0}^{2} |0{,}5 - x_i| = |0{,}5 - 0| \cdot |0{,}5 - 1| \cdot |0{,}5 - 2|$$

$$= 0{,}5 \cdot 0{,}5 \cdot 1{,}5 = 0{,}375$$

$$\text{Error} \leq \frac{7{,}389}{6} \cdot 0{,}375 \approx 0{,}462$$

Resultado: Error $\leq 0{,}462$

Ejercicio 3: Error para $\sin(x)$

Datos: $x_0 = 0$, $x_1 = \frac{\pi}{2}$, $x_2 = \pi$, $x = \frac{\pi}{4}$, $f(x) = \sin(x)$

Solución:

$$
\begin{aligned}
f'''(x) &= -\cos(x) \\
M_3 &= \max_{[0,\pi]} |f'''(x)| = 1 \\
\prod_{i=0}^{2} \left|\frac{\pi}{4} - x_i\right| &= \left|\frac{\pi}{4} - 0\right| \cdot \left|\frac{\pi}{4} - \frac{\pi}{2}\right| \cdot \left|\frac{\pi}{4} - \pi\right| \\
&= \frac{\pi}{4} \cdot \frac{\pi}{4} \cdot \frac{3\pi}{4} = \frac{3\pi^3}{64} \approx 1{,}451 \\
\text{Error} &\leq \frac{1}{6} \cdot 1{,}451 \approx 0{,}242
\end{aligned}
$$

Resultado: Error $\leq 0{,}242$

```r
# FUNCI N PARA CALCULAR ERROR DE INTERPOLACION
error_interpolacion <- function(x_points, x, Mn1) {
  n <- length(x_points)
  product <- 1
  for (i in 1:n) {
    product <- product * abs(x - x_points[i])
  }
  error <- (Mn1 / factorial(n)) * product
  return(error)
}

# EJERCICIO 1: Error para ln(x)
x_points1 <- c(1, 2, 3)
x1 <- 1.5
Mn1_1 <- 2  # max|f'''(x)| para ln(x) en [1,3]
error1 <- error_interpolacion(x_points1, x1, Mn1_1)
cat("Ejercicio 1 - Error para ln(x) en x=1.5:", error1, "\n")

# EJERCICIO 2: Error para exp(x)
x_points2 <- c(0, 1, 2)
x2 <- 0.5
Mn1_2 <- exp(2)  # max|f'''(x)| para exp(x) en [0,2]
error2 <- error_interpolacion(x_points2, x2, Mn1_2)
cat("Ejercicio 2 - Error para exp(x) en x=0.5:", round(error2, 4), "\n")

# EJERCICIO 3: Error para sin(x)
x_points3 <- c(0, pi/2, pi)
x3 <- pi/4
Mn1_3 <- 1  # max|f'''(x)| para sin(x) en [0,  ]
error3 <- error_interpolacion(x_points3, x3, Mn1_3)
cat("Ejercicio 3 - Error para sin(x) en   /4:", round(error3, 4), "\n")

# FUNCI N PARA CALCULAR DERIVADAS Y MOSTRAR PASOS
calcular_error_detallado <- function(x_points, x, funcion, intervalo,
    derivada_orden) {
  n <- length(x_points)

  # Calcular producto
```

```r
  producto <- 1
  cat("Producto: ")
  for (i in 1:n) {
    producto <- producto * abs(x - x_points[i])
    if (i < n) {
      cat("|", x, "-", x_points[i], "|    ")
    } else {
      cat("|", x, "-", x_points[i], "| = ", producto, "\n")
    }
  }

  # Calcular error
  error <- (derivada_orden / factorial(n)) * producto
  cat("Error     (", derivada_orden, "/", factorial(n), ")   ", producto,
     "=", error, "\n")

  return(error)
}

# Ejemplo detallado para ln(x)
cat("\n--- C LCULO DETALLADO EJERCICIO 1 ---\n")
calcular_error_detallado(c(1, 2, 3), 1.5, "ln(x)", c(1, 3), 2)
```

Listing 3: Codigo R para Error de Interpolacion