# INSTITUTO TECNOLÓGICO SUPERIOR DEL ORIENTE DEL ESTADO DE HIDALGO


*Evidencia:*

Reporte de Métodos Numéricos:

Interpolación y Regresión

*Nombre Alumno(s):*

Carlos Martinez Gonzalez

Erik Isabel Vera Ortiz

Juan Carlos Sánchez Muñoz

José Luis Yañez Zamora

Jose Sebastian Rodriguez Salgado

Aldo Shaiel Ruiz Martínez.

## 1. Introducción

En el ámbito de la programación y el análisis de datos, los **métodos numéricos** juegan un papel crucial al ofrecer soluciones aproximadas a problemas matemáticos complejos, especialmente cuando los datos disponibles son discretos o no siguen una forma exacta. Los métodos de **interpolación** y **regresión lineal** son dos de los enfoques más utilizados para modelar y predecir relaciones entre variables a partir de datos observacionales. La **interpolación de Lagrange** y **la interpolación de Newton** son técnicas que permiten encontrar polinomios que pasan por un conjunto de puntos de datos, mientras que la **regresión lineal por mínimos cuadrados** busca ajustar una línea recta a los datos minimizando el error cuadrático entre los valores observados y los estimados. Este reporte tiene como objetivo aplicar y comparar estos métodos utilizando **Java** como herramienta de implementación, además de calcular el error asociado en cada caso. Se analizarán tanto la interpolación polinómica como la regresión lineal, mostrando su aplicación práctica en un ejercicio específico, evaluando su precisión y aplicabilidad en el contexto de datos experimentales.

## 2. Métodos de Interpolación

### 2.1. Problema Propuesto

Dados los siguientes puntos:

| X | Y |
|---|---|
| 1 | 2 |
| 3 | 5 |
| 4 | 6 |

Encontrar el valor interpolado para x = 2 usando:

- Método de Lagrange
- Método de Newton (Diferencias Divididas)

### 2.2. Solución con Interpolación de Lagrange

**Formula General:**

$$P(x) = \sum_{i=0}^{n} y_i \cdot L_i(x)$$

**donde:**

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

### Cálculo Manual:

### Para $x_0 = 1$, $y_0 = 2$:

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

### Para $x_1 = 3$, $y_1 = 5$:

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

**Para $x_2 = 4$, $y_2 = 6$:**

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

**Polinomio de Lagrange:**

$$P(2) = 2 \cdot \frac{1}{3} + 5 \cdot 1 + 6 \cdot \left(-\frac{1}{3}\right) = \frac{2}{3} + 5 - 2 = \frac{11}{3} \approx 3.6667$$

## Código en Java:

```
public class LagrangeInterpolation {
    public static double interpolate(double[] x, double[] y, double xVal) {
        double result = 0.0;
        for (int i = 0; i < x.length; i++) {
            double term = y[i];
            for (int j = 0; j < x.length; j++) {
                if (j != i) {
                    term *= (xVal - x[j]) / (x[i] - x[j]);
                }
            }
            result += term;
        }
        return result;
    }

    public static void main(String[] args) {
        double[] x = {1, 3, 4};
        double[] y = {2, 5, 6};
        double xVal = 2;
        System.out.printf("Interpolación Lagrange en x=%.2f: %.4f\n", xVal,
interpolate(x, y, xVal));
    }
}
```

Error:

Valor de Lagrange: PN(2)=3.6¯=11/3

Error: $\left|\frac{11/3 - 11/3}{11/3}\right| = 0$

Error Asociado:
$E(x) = f(n+1)(\xi)/(n+1)! \cdot \Pi(x - x_i)$

```
Salida:

Interpolación Lagrange en x=2.00: 3.6667
```

## 2.3. Solución con Interpolación de Newton

**Tabla de Diferencias Divididas:**

| X | Y | 1° Dif | 2° Dif |
|---|---|---|---|
| 1 | 2 | | |
| 3 | 5 | 1.5 | |
| 4 | 6 | 1 | -0.25 |

**Polinomio de Newton:**

$$P(x) = 2 + 1.5(x - 1) - 0.25(x - 1)(x - 3)$$

**Evaluando en x = 2:**

$$P(2) = 2 + 1.5(1) - 0.25(1)(-1) = 3.75$$

## Código en Java:

```
public class NewtonInterpolation {
    public static double interpolate(double[] x, double[] y, double xVal)
{
        double[] coef = y.clone();
        int n = x.length;

        // Cálculo de diferencias divididas
        for (int j = 1; j < n; j++) {
            for (int i = n - 1; i >= j; i--) {
                coef[i] = (coef[i] - coef[i - 1]) / (x[i] - x[i - j]);
            }
        }
        // Evaluación del polinomio
        double result = coef[n - 1];
        for (int i = n - 2; i >= 0; i--) {
            result = result * (xVal - x[i]) + coef[i];
        }
        return result;
    }
    public static void main(String[] args) {
        double[] x = {1, 3, 4};
        double[] y = {2, 5, 6};
        double xVal = 2;
        System.out.printf("Interpolación Newton en x=%.2f: %.4f\n", xVal,
interpolate(x, y, xVal));
    }
}
```

Salida:

```
Interpolación Newton en x=2.00: 3.6667
```

Error:

Valor de Newton: PL(2)=3.6¯=11/3

Error: $\left|\frac{11/3 - 11/3}{11/3}\right| = 0$

Error Asociado:
$E(x) = f[x_0, x_1, ..., x_?, x] \cdot \Pi(x - x_i)$

## 3. Regresión Lineal

### 3.1. Problema Propuesto

Ajustar una recta a los siguientes datos:

| X | Y |
|---|---|
| 1 | 2 |
| 3 | 5 |
| 4 | 6 |

### 3.2. Solución Analítica

Paso 1: Calcular las sumatorias

Tenemos 3 pares de datos, n=3

Calculamos las siguientes sumatorias:

∑x=1+3+4=8

∑y=2+5+6=13

∑xy=(1)(2)+(3)(5)+(4)(6)=2+15+24=41

∑x^2=12+32+42=1+9+16=26

Paso 2: Calcular pendiente (m) y ordenada al origen (b)

Fórmulas:

$$m = \frac{n\sum xy - \sum x \sum y}{n\sum x^2 - (\sum x)^2}$$

$$m = \frac{3(41) - (8)(13)}{3(26) - (8)^2} = \frac{123 - 104}{78 - 64} = \frac{19}{14} \approx 1.357$$

$$b = \frac{\sum y - m\sum x}{n} = \frac{13 - 1.357(8)}{3} = \frac{13 - 10.856}{3} \approx \frac{2.144}{3} \approx 0.715$$

Ecuación ajustada de la recta: y = 1.357x + 0.715

Paso 3: Estimaciones y errores

| X | Y real | Y estimada $\hat{y}$ | Error absoluto $\lvert Y - \hat{y} \rvert$ |
|---|---|---|---|
| 1 | 2 | $1.357(1) + 0.715 = 2.072$ | $\lvert 2 - 2.072 \rvert = 0.072$ |
| 3 | 5 | $1.357(3) + 0.715 = 4.786$ | $\lvert 5 - 4.786 \rvert = 0.214$ |
| 4 | 6 | $1.357(4) + 0.715 = 6.143$ | $\lvert 6 - 6.143 \rvert = 0.143$ |

Ecuación ajustada: y = 1.357x + 0.715 (lo que indica un buen ajuste lineal)

## Código en Java:

```
public class RegresionLineal {

    public static void main(String[] args) {

        double[] x = {1, 3, 4};

        double[] y = {2, 5, 6};

        int n = x.length;

        double sumX = 0, sumY = 0, sumXY = 0, sumX2 = 0;

        for (int i = 0; i < n; i++) {

            sumX += x[i];

            sumY += y[i];

            sumXY += x[i] * y[i];
```

```
            sumX2 += x[i] * x[i];
        }
        double a = (n * sumXY - sumX * sumY) / (n * sumX2 - sumX * sumX);
        double b = (sumY - a * sumX) / n;
        System.out.println("Ecuación ajustada: Y = " + a + "X + " + b);
        System.out.println("\nResultados y errores:");

        for (int i = 0; i < n; i++) {
            double yEstimada = a * x[i] + b;
            double error = Math.abs(y[i] - yEstimada);
            System.out.printf("X=%.1f, Y real=%.1f, Y estimada=%.2f,
Error=%.2f\n",
                              x[i], y[i], yEstimada, error);
        }
    }
}
```

Salida:

Ecuación ajustada: Y = 1.357X + 0.715

## 4. Correlación

### 4.1. Problema Propuesto

Una tienda de electrodomésticos quiere analizar la relación entre el precio de ciertos productos y la cantidad de unidades vendidas en un mes. La tienda tiene datos históricos de tres productos diferentes que se vendieron en diferentes precios y cantidades.

Los datos son los siguientes:

| Producto | Precio ( x ) | Unidades Vendidas ( y ) |
|---|---|---|
| Producto 1 | 1.00 | 2 |
| Producto 2 | 2.00 | 3 |
| Producto 3 | 3.00 | 5 |

### 4.2. Solución por método de Correlación (Calculos)

*Datos Proporcionados*:

- **Precio (x):** 1, 2, 3
- **Unidades Vendidas (y):** 2, 3, 5

*Paso 1: Cálculo de las sumas necesarias*

Vamos a calcular las sumas requeridas para aplicar las fórmulas.

| | |
|---|---|
| Suma de x (Σx) | $\Sigma x = 1.00 + 2.00 + 3.00 = 6.00$ |
| Suma de y (Σy) | $\Sigma y = 2 + 3 + 5 = 10$ |
| Suma de x y (Σxy) | $\Sigma xy = (1 \times 2) + (2 \times 3) + (3 \times 5) = 2 + 6 + 15 = 23$ |
| Suma de x^2 ($\Sigma x^2$) | $\Sigma x^2 = (1^2) + (2^2) + (3^2) = 1.00 + 4.00 + 9.00 = 14.00$ |
| Suma de y^2 ($\Sigma y^2$) | $\Sigma y^2 = (2^2) + (3^2) + (5^2) = 4 + 9 + 25 = 38$ |

*Paso 2: Aplicamos la fórmula de correlación*

*Fórmula del Coeficiente de Correlación (r):* | *Sustitución de los elementos*

$$r = \frac{n \cdot \Sigma xy - \Sigma x \cdot \Sigma y}{\sqrt{(n \cdot \Sigma x^2 - (\Sigma x)^2) \cdot (n \cdot \Sigma y^2 - (\Sigma y)^2)}} \qquad r = \frac{(3 \cdot 14 - (6)^2) \cdot (3 \cdot 38 - (10)^2)}{3 \cdot 23 - 6 \cdot 10} \approx 0.981$$

## 4.3. Pseudocodigo

```
Inicio
    Leer n  // Número de variables (pares de datos)
Definir dos arreglos:
x[] para los valores de x
y[] para los valores de y
    Para i desde 0 hasta n-1 hacer
        Leer x[i]  // Leer el valor de x[i]
        Leer y[i]  // Leer el valor de y[i]
    FinPara

    Calcular Σx, Σy, Σx², Σy² y Σxy
        Σx = Sumar todos los valores de x
        Σy = Sumar todos los valores de y
        Σx² = Sumar los cuadrados de los valores de x
        Σy² = Sumar los cuadrados de los valores de y
        Σxy = Sumar los productos de x[i] * y[i]

    Calcular el coeficiente de correlación (r):
       r = (n * Σxy - Σx * Σy) / sqrt((n * Σx² - Σx²) * (n * Σy² - Σy²))

    Imprimir el valor de r

Fin
```

## 4.3. Codigo JAVA

```java
import java.util.Scanner;

public class Correlacion {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Ingreso de datos por parte del usuario
        System.out.print("Ingrese el número de puntos de datos: ");
        int n = scanner.nextInt();

        double[] x = new double[n];
        double[] y = new double[n];

        // Ingreso de valores para x y y
        System.out.println("Ingrese los valores de x:");
        for (int i = 0; i < n; i++) {
            x[i] = scanner.nextDouble();
        }
        System.out.println("Ingrese los valores de y:");
        for (int i = 0; i < n; i++) {
            y[i] = scanner.nextDouble();
        }

      // Cálculo de las sumas necesarias
        double sumX = 0, sumY = 0, sumXY = 0, sumX2 = 0, sumY2 = 0;
        for (int i = 0; i < n; i++) {
            sumX += x[i];
            sumY += y[i];
            sumXY += x[i] * y[i];
            sumX2 += x[i] * x[i];
            sumY2 += y[i] * y[i];
        }

      // Cálculo del coeficiente de correlación (r)
      double r = (n * sumXY - sumX * sumY) /
       Math.sqrt((n * sumX2 - sumX * sumX) * (n * sumY2 - sumY * sumY));

  // Imprimir directamente el coeficiente de correlación
      System.out.printf("\nCoeficiente de correlación (r): %.2f\n", r);
    }
}
```

Salida:

```
Ingrese el número de puntos de datos: 3
Ingrese los valores de x:
1.0
2.0
3.0
Ingrese los valores de y:
2.0
3.0
5.0

Coeficiente de correlación (r): 0.98
```

# 5.Regresión Lineal por Mínimos Cuadrados

## 5.1. Problema Propuesto

Ajustar una recta a los siguientes datos:

| X | Y |
|---|---|
| 1 | 2 |
| 3 | 5 |
| 4 | 6 |

## 5.2. Solución Analítica

Calcular sumatorias:
$\Sigma x = 6, \Sigma y = 10, \Sigma xy = 23, \Sigma x^2 = 14$

Pendiente (m) y ordenada (b):
$m = (3 \cdot 23 - 6 \cdot 10)/(3 \cdot 14 - 36) = 1.5$
$b = (10 - 1.5 \cdot 6)/3 \approx 0.333$

Ecuación de la recta:
$y = 1.5x + 0.333$

## Código en Java:

```
public class LinearRegression {
    public static void main(String[] args) {
        double[] x = {1, 2, 3};
        double[] y = {2, 3, 5};
        int n = x.length;
        double sumX = 0, sumY = 0, sumXY = 0, sumX2 = 0;
        for (int i = 0; i < n; i++) {
            sumX += x[i];
            sumY += y[i];
            sumXY += x[i] * y[i];
            sumX2 += x[i] * x[i];
        }
        double m = (n * sumXY - sumX * sumY) / (n * sumX2 - sumX * sumX);
        double b = (sumY - m * sumX) / n;
        System.out.printf("Recta de regresión: y = %.2fx + %.2f\n", m,
b);
    }
}
```

```
Salida:

Recta de regresión: y = 1.50x + 0.33
```

Error:

Valor de Mínimo cuadrado:

Error: $\left|\frac{3.4286 - 3.6667}{3.6667}\right| = \left|\frac{-0.2381}{3.6667}\right| \approx 0.0649$

Coeficiente de Correlación (r):
$r = (n\Sigma xy - \Sigma x\Sigma y) / \text{sqrt}[(n\Sigma x^2 - (\Sigma x)^2)(n\Sigma y^2 - (\Sigma y)^2)]$

## 6. Conclusiones

En este reporte, hemos aplicado y analizado los métodos de **Interpolación de Lagrange**, **Interpolación de Newton** y **Regresión Lineal por Mínimos Cuadrados** para resolver un problema común de ajuste de datos, utilizando Java como lenguaje de implementación. La **interpolación de Lagrange** y la **interpolación de Newton** son técnicas útiles para encontrar polinomios que se ajustan exactamente a los datos, siendo la segunda más eficiente en términos computacionales para grandes volúmenes de datos. Por otro lado, la **regresión lineal** se mostró como un enfoque más simplificado y efectivo cuando se busca modelar una relación lineal entre las variables, minimizando el error cuadrático. A lo largo del proceso, se calculó el **coeficiente de correlación** (r) y se evaluó el **error asociado** a cada método para comprender mejor su precisión y efectividad. La implementación de estos métodos numéricos en Java no solo demuestra su aplicabilidad práctica en el análisis de datos, sino también resalta la importancia de elegir la técnica adecuada dependiendo de la naturaleza de los datos y el objetivo del análisis. Con todo, estos métodos son fundamentales en el análisis de datos en múltiples disciplinas, desde la ingeniería hasta las ciencias sociales, y su comprensión y aplicación en problemas del mundo real proporciona herramientas poderosas para la toma de decisiones informadas y precisas.

## 7. Referencias


Chapra, S. C. (2018). *Métodos numéricos para ingenieros* (7a ed.). McGraw-Hill Education.

Burden, R. L., & Faires, J. D. (2010). *Numerical analysis* (9th ed.). Brooks/Cole.

Kincaid, D. R., & Cheney, W. (2002). *Numerical analysis: Mathematics of scientific computing* (3rd ed.). Brooks/Cole.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). *Numerical recipes: The art of scientific computing* (3rd ed.). Cambridge University Press.

## 8. Evidencia Colaborativa