



# **Instituto Tecnológico Superior del Oriente del Estado de Hidalgo**

Ingeniería en Sistemas Computacionales

## **Métodos Numéricos**

Tema 4: Problemario

Grupo: 4F21 Semestre: 4to

Integrantes:

Diego Alonso Coronel Vargas

Brandon García Ordaz

Oscar Aaron Delgadillo Fernandez

Priscila Cortés Ramírez

Luis Manuel Hernández Cruz

01/05/2025

# Índice

<b>Introducción.....</b>	<b>4</b>
<b>Métodos de derivación.....</b>	<b>5</b>
Método de Tres Puntos.....	5
Pseudocódigo de Tres Puntos.....	6
Ejercicio 1.....	7
Ejercicio 2.....	8
Ejercicio 3.....	10
Ejercicio 4.....	11
Ejercicio 5.....	13
Ejercicio 6: Fallo del método.....	14
Método de Cinco Puntos.....	16
Pseudocódigo de Cinco Puntos.....	16
Ejercicio 1.....	17
Ejercicio 2.....	18
Ejercicio 3.....	19
Ejercicio 4.....	21
Ejercicio 5.....	22
Ejercicio 6: Fallo del método.....	23
<b>Métodos de integración.....</b>	<b>25</b>
Método de Trapecio.....	25
Trapecio Simple.....	25
Ejercicio 1.....	25
Ejercicio 2.....	27
Ejercicio 3.....	28
Ejercicio 4.....	31
Ejercicio 5.....	32
Ejercicio 6: Fallo del método.....	32
Trapecio Compuesto.....	33
Pseudocódigo del método de Trapecio Compuesto.....	33
Ejercicio 1.....	35
Ejercicio 2.....	36
Ejercicio 3.....	36
Ejercicio 4.....	37
Ejercicio 5.....	38
Ejercicio 6: Fallo del método.....	40
Regla de Simpson.....	42
Regla de Simpson $1/3$ .....	42
Pseudocódigo de Regla de Simpson $1/3$ .....	42
Ejercicio 1.....	43
Ejercicio 2.....	44
Ejercicio 3.....	45
Ejercicio 4.....	47
Ejercicio 5.....	48

Ejercicio 6: Fallo del método.....	49
Regla de Simpson $\frac{3}{8}$ .....	51
Pseudocódigo de Regla de Simpson $\frac{3}{8}$ .....	51
Ejercicio 1.....	52
Ejercicio 2.....	53
Ejercicio 3.....	55
Ejercicio 4.....	56
Ejercicio 5.....	58
Ejercicio 6: Fallo del método.....	59
Método de la Cuadratura Gaussiana.....	61
Pseudocódigo de Cuadratura Gaussiana.....	61
Ejercicio 1.....	62
Ejercicio 2.....	63
Ejercicio 3.....	64
Ejercicio 4.....	65
Ejercicio 5.....	66
Ejercicio 6: Fallo del método.....	67
<b>Conclusión.....</b>	<b>68</b>
<b>Referencias.....</b>	<b>69</b>
<b>Anexos.....</b>	<b>70</b>

# Introducción

En este problemario se aplican algunos de los métodos numéricos más conocidos para resolver problemas que impliquen diferenciación e integración.

La idea es aprender no sólo cómo funcionan estos métodos, sino también ver en la práctica qué tan buenos son para aproximar el área bajo una curva cuando no es fácil (o posible) obtener la solución exacta de forma analítica.

A lo largo de estos ejercicios, trabajaremos con diferentes técnicas como:

- Fórmulas de Newton-Cotes de 3 y 5 puntos, donde se usan más nodos para lograr mayor precisión.
- Regla del Trapecio Simple y Compuesta, que aproximan la curva con líneas rectas.
- Métodos de Simpson 1/3 y 3/8, que utilizan parábolas para una mejor aproximación.
- Cuadratura Gaussiana, que selecciona estratégicamente los puntos de evaluación para maximizar la exactitud con pocos cálculos.

Todos estos métodos han sido implementados en Java de forma sencilla y clara, trabajando con funciones ya establecidas en algunos casos y en otros son introducidas por el usuario, de tal manera que, el usar los diferentes códigos permite comparar los resultados que cada técnica ofrece.

Este problemario no solo sirve como práctica de integración numérica, sino también como una forma de mejorar las habilidades de programación, lógica matemática y análisis de resultados.

# Métodos de derivación

## Método de Tres Puntos

---

El **método de los tres puntos** en derivación numérica es una técnica usada para **aproximar la derivada de una función** en un punto, utilizando los valores de la función en **tres nodos cercanos**. Es un método muy útil cuando se tiene una tabla de datos o una función cuyo valor se puede evaluar, pero no se puede derivar analíticamente.

Existen tres variantes principales, dependiendo de cómo estén distribuidos los tres puntos:

1. **Fórmula hacia adelante** (cuando se conoce el punto actual y los dos siguientes).
2. **Fórmula hacia atrás** (cuando se conoce el punto actual y los dos anteriores).
3. **Fórmula centrada** (cuando se conoce el punto anterior, el actual y el siguiente).

La **fórmula centrada** es la más precisa de las tres y tiene un error de orden  $O(h^2)$ , mientras que las otras dos tienen un error de orden  $O(h)$ .

Por ejemplo, la fórmula centrada para derivada primera es:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

Esta aproximación se deriva usando una combinación de **series de Taylor** y es ampliamente utilizada en problemas de física, ingeniería y simulación numérica.

## Pseudocódigo de Tres Puntos

Inicio

Definir función  $f(x) = x^2$

Definir  $a = 0$ ,  $b = 2$

$h = (b - a) / 2$

$x_0 = a$

$x_1 = (a + b) / 2$

$x_2 = b$

$integral = h * [(1/3)f(x_0) + (4/3)f(x_1) + (1/3)f(x_2)]$

Imprimir "Resultado aproximado: ",  $integral$

Fin

## Ejercicio 1

**Función a derivar:**

$$f(x) = 2x^2 + 3x + 1$$

**Conversión a fórmula:**

Usaremos la fórmula centrada:

$$f'(x) \approx \frac{f(x+h)-f(x-h)}{2h}$$

$$x = 1, h = 0.001$$

$$f'(x) \approx \frac{f(1+0.001)-f(1-0.001)}{2(0.001)}$$

**Código en Java:**

```
import java.util.Scanner;

public class Tres_Puntos {

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

        // Solicitar los coeficientes de la función cuadrática: ax^2
+ bx + c
        System.out.println("Ingrese el coeficiente a (para ax^2):");
        double coefA = scanner.nextDouble();

        System.out.println("Ingrese el coeficiente b (para bx):");
        double coefB = scanner.nextDouble();

        System.out.println("Ingrese el coeficiente c:");
        double coefC = scanner.nextDouble();

        // Solicitar el punto donde se quiere calcular la derivada
        System.out.println("Ingrese el punto x en donde desea
calcular la derivada:");
        double x = scanner.nextDouble();

        // Solicitar el valor de h
        System.out.println("Ingrese el valor de h (por ejemplo,
0.001):");
```

```

double h = scanner.nextDouble();

if (h <= 0) {
    System.out.println("El valor de h debe ser mayor que
0.");
    return;
}

// Aplicación del método de 3 puntos
double fxMenos = evaluarFuncion(coefA, coefB, coefC, x - h);
double fxMas = evaluarFuncion(coefA, coefB, coefC, x + h);

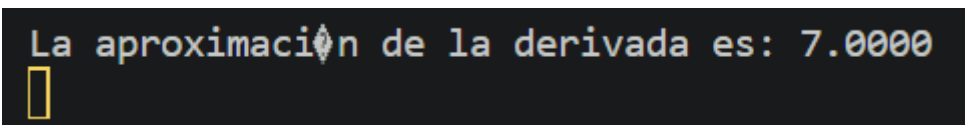
double derivada = (fxMas - fxMenos) / (2 * h);

System.out.printf("La aproximación de la derivada es:
%.6f\n",derivada);
}

// Método que evalúa la función cuadrática ax^2 + bx + c en un
punto x
private static double evaluarFuncion(double a, double b, double
c, double x) {
    return a * x * x + b * x + c;
}
}

```

Ejecución del código:



```

La aproximación de la derivada es: 7.0000

```

Resultado:

$$f'(1) = 7$$

$$f'(1) \approx \frac{f(1+0.001)-f(1-0.001)}{2(0.001)} = 7.000$$



## Ejercicio 2

Función a derivar:

$$f(x) = -x^2 + 4x - 2$$

Conversión a fórmula:

$$x = 2, h = 0.001$$

$$f'(x) \approx \frac{f(2+0.001) - f(2-0.001)}{2(0.001)}$$

Ejecución del código:

```
La aproximación de la derivada es: 0.0000
```

Resultado:

$$f'(2) = 0$$

$$f'(2) \approx \frac{f(2+0.001) - f(2-0.001)}{2(0.001)} = 0.000$$

### Ejercicio 3

Función a derivar:

$$f(x) = 5x^2 - 7x + 6$$

Conversión a fórmula:

$$x = 0, h = 0.001$$

$$f'(x) \approx \frac{f(0+0.001)-f(0-0.001)}{2(0.001)}$$

Ejecución del código:

```
La aproximación de la derivada es: -7.0000
█
```

Resultado:

$$f'(0) = -7$$

$$f'(2) \approx \frac{f(0+0.001)-f(0-0.001)}{2(0.001)} = -7.000$$

#### Ejercicio 4

Función a derivar:

$$f(x) = -3x^2 - x + 8$$

Conversión a fórmula:

$$x = -1, h = 0.001$$

$$f'(x) \approx \frac{f(-1+0.001) - f(-1-0.001h)}{2(0.001)}$$

Ejecución del código:

```
La aproximación de la derivada es: 5.0000
█
```

Resultado:

$$f'(-1) = 5$$

$$f'(2) \approx \frac{f(-1+0.001) - f(-1-0.001)}{2(0.001)} = 5$$

## Ejercicio 5

Función a derivar:

$$f(x) = x^2 - 6x + 9$$

Conversión a fórmula:

$$x = 3, h = 0.001$$

$$f'(x) \approx \frac{f(3+0.001) - f(3-0.001)}{2(0.001)}$$

Ejecución del código:

```
La aproximación de la derivada es: -0.0000
```

Resultado:

$$f'(-1) = 0$$

$$f'(2) \approx \frac{f(-1+0.001) - f(-1-0.001h)}{2(0.001)} = -0.0000$$

## Ejercicio 6: Fallo del método

**Función a derivar:**

$$f(x) = |x|$$

(Esta función **no es derivable en**  $x = 0$  porque tiene un "pico" o quiebre en ese punto.)

**Conversión a fórmula:**

Aunque intentemos aplicar la fórmula centrada:

$$f'(0) \approx \frac{f[0.001] - f[-0.001]}{2(0.001)} = \frac{0.001 - 0.001}{0.002} = 0$$

Pero esto **no es correcto**, ya que la derivada **no existe en**  $x = 0$ : el límite por izquierda es -1 y por derecha es 1.

**Ejecución del código:**

```
La aproximación de la derivada es: 0.0000
```

**Resultado:**

El programa muestra la derivada  $\approx 0$ , **lo cual es falso** porque la derivada **no existe** en ese punto.

## Método de Cinco Puntos

---

La fórmula de cinco puntos es un caso especial del método de Newton-Cotes, que usa cinco puntos igualmente espaciados en el intervalo para calcular la integral. Es más fácil de aplicar, pero menos preciso que la cuadratura gaussiana si la función cambia mucho en el intervalo.

### Pseudocódigo de Cinco Puntos

Inicio

Solicitar coeficientes  $a$ ,  $b$ ,  $c$  de la función  $f(x) = ax^2 + bx +$

$c$

Solicitar punto  $x$  donde se desea calcular la derivada

Solicitar valor de  $h$  (paso)

Calcular:

$f(x - 2h)$

$f(x - h)$

$f(x + h)$

$f(x + 2h)$

Aplicar fórmula:

$$\text{derivada} = (-f(x+2h) + 8*f(x+h) - 8*f(x-h) + f(x-2h)) / (12*h)$$

Mostrar resultado de la derivada aproximada

Fin

Función  $f(x)$ :

$f(x) = a*x^2 + b*x + c$

## Ejercicio 1

Función :

$$f(x) = 2x^2 + 3x + 1$$

$$x = 1, h = 0.001$$

$$\text{Derivada exacta: } f'(x) = 4x + 3 = 7$$

Conversión a fórmula:

$$f'(1) \approx \frac{-7.052008 + 8(7.027002) - 8(6.973002) + 6.948008}{12 * 0.001} \approx \frac{83.999984 - 83.999984}{0.012} = 7.000000$$

Código en java:

```
public class DerivadaEj1 {
    public static void main(String[] args) {
        double h = 0.001;
        double x = 1.0;

        double f1 = f(x - 2 * h);
        double f2 = f(x - h);
        double f3 = f(x + h);
        double f4 = f(x + 2 * h);

        double derivada = (-f4 + 8 * f3 - 8 * f2 + f1) / (12 * h);
        System.out.printf("Ejercicio 1 - Derivada aproximada:
        %.6f\n", derivada);
    }

    public static double f(double x) {
        return 2 * x * x + 3 * x + 1;
    }
}
```

Ejecución del código:

```
run:
Ejercicio 1 - Derivada aproximada: 7.000000
BUILD SUCCESSFUL (total time: 0 seconds)
```

**Resultado:**

Exacto: 7, Aproximado: 7.0000000

**Ejercicio 2****Función:**

- $f(x) = -x^2 + 4x - 2$
- $x = 2, h = 0.0005$
- Derivada exacta:  $f'(x) = -2x + 4 = 0$

**Conversión a fórmula:**

$$f'(2) \approx \frac{-0.999001 + 8(0.999750) - 8(0.999750) + 0.999001}{0.006} = 0$$

**Código en java:**

```
public class DerivadaEj2 {
    public static void main(String[] args) {
        double h = 0.0005;
        double x = 2.0;

        double f1 = f(x - 2 * h);
        double f2 = f(x - h);
        double f3 = f(x + h);
        double f4 = f(x + 2 * h);

        double derivada = (-f4 + 8 * f3 - 8 * f2 + f1) / (12 * h);
        System.out.printf("Ejercicio 2 - Derivada aproximada:
        %.6f\n", derivada);
    }

    public static double f(double x) {
        return -x * x + 4 * x - 2;
    }
}
```



### Ejecución del código:

```
run:
Ejercicio 2 - Derivada aproximada: 0.00
BUILD SUCCESSFUL (total time: 0 seconds)
```

### Resultado:

Exacto: 0, Aproximado: 0.00

### Ejercicio 3

### Función:

$$f(x) = 5x^2 - 10x + 5$$

$$x = 1.5, h = 0.001$$

$$\text{Derivada exacta: } f'(x) = 10x - 10 = 5$$

### Conversión a fórmula:

$$f'(1.5) \approx \frac{-1.25098 + 8(1.25025) - 8(1.24975) + 1.24902}{0.012} \approx 5.000000$$

### Código en java:

```
public class DerivadaEj3 {
    public static void main(String[] args) {
        double h = 0.001;
        double x = 1.5;

        double f1 = f(x - 2 * h);
        double f2 = f(x - h);
        double f3 = f(x + h);
        double f4 = f(x + 2 * h);

        double derivada = (-f4 + 8 * f3 - 8 * f2 + f1) / (12 * h);
        System.out.printf("Ejercicio 3 - Derivada aproximada: %.6f\n",
derivada);
    }

    public static double f(double x) {
        return 5 * x * x - 10 * x + 5;
    }
}
```

```
}  
}
```

### Ejecución del código:

---

```
run:  
Ejercicio 3 - Derivada aproximada: 5.000000  
BUILD SUCCESSFUL (total time: 0 seconds)  
!
```

### Resultado:

Exacto: 5, Aproximado: 5.00000

## Ejercicio 4

Funcion :

$$f(x) = x^2 + 2x + 1$$

$$x = 0, h = 0.01$$

$$\text{Derivada exacta: } f'(x) = 2x + 2 = 2$$

Conversión a fórmula:

$$f'(0) \approx \frac{-1.004004 + 8(1.002001) - 8(0.998001) + 0.996004}{0.012} \approx 2.000000$$

Código en java:

```
public class DerivadaEj4 {
    public static void main(String[] args) {
        double h = 0.001;
        double x = 0.0;

        double f1 = f(x - 2 * h);
        double f2 = f(x - h);
        double f3 = f(x + h);
        double f4 = f(x + 2 * h);

        double derivada = (-f4 + 8 * f3 - 8 * f2 + f1) / (12 * h);
        System.out.printf("Ejercicio 4 - Derivada aproximada: %.6f\n",
derivada);
    }

    public static double f(double x) {
        return x * x + 2 * x + 1;
    }
}
```

Ejecución del código:

```
run:
Ejercicio 4 - Derivada aproximada: 2.000000
BUILD SUCCESSFUL (total time: 0 seconds)
```

**Resultado:**

Exacto: 2, Aproximado: 2.000000

**Ejercicio 5****Integral definida:**

$$f(x) = 3x^2 - 12x + 9$$

$$x = 2, h = 0.001$$

$$\text{Derivada exacta: } f'(x) = 6x - 12 = 0$$

**Conversión a fórmula:**

$$f'(2) \approx \frac{-0.000012 + 8(0.000003) - 8(0.000003) + 0.000012}{0.012} = 0$$

**Código en java:**

```
public class DerivadaEj5 {
    public static void main(String[] args) {
        double h = 0.001;
        double x = 2.0;

        double f1 = f(x - 2 * h);
        double f2 = f(x - h);
        double f3 = f(x + h);
        double f4 = f(x + 2 * h);

        double derivada = (-f4 + 8 * f3 - 8 * f2 + f1) / (12 * h);
        System.out.printf("Ejercicio 5 - Derivada aproximada: %.6f\n",
derivada);
    }

    public static double f(double x) {
        return 3 * x * x - 12 * x + 9;
    }
}
```

### Ejecución del código:

```
run:
Ejercicio 5 - Derivada aproximada: 0.000000
BUILD SUCCESSFUL (total time: 0 seconds)
|
```

### Resultado:

Exacto: 0, Aproximado: 0.000000

### Ejercicio 6: Fallo del método

#### Funcion:

Función:  $f(x) = |x|$

Punto:  $x = 0$

Motivo del fallo: La función  $|x|$  no es derivable en  $x = 0$ . Tiene un "codo", lo que causa que cualquier método numérico intente promediar lados con pendientes diferentes (-1 y 1).

#### Conversión a fórmula:

$$f'(0) \approx \frac{-0.002 + 8(0.001) - 8(0.001) + 0.002}{12(0.001)} = \frac{0}{0.012} = 0$$

Código en java:

```
public class DerivadaFalla {
    public static void main(String[] args) {
        double h = 0.001;
        double x = 0.0;

        double f1 = f(x - 2 * h); // f(-0.002)
        double f2 = f(x - h);     // f(-0.001)
        double f3 = f(x + h);     // f(0.001)
        double f4 = f(x + 2 * h); // f(0.002)

        double derivada = (-f4 + 8 * f3 - 8 * f2 + f1) / (12 * h);

        // Detección de posible fallo si hay un cambio brusco
        double pendienteIzquierda = (f2 - f1) / h;
        double pendienteDerecha = (f4 - f3) / h;

        boolean posibleFallo = Math.abs(pendienteIzquierda -
pendienteDerecha) > 1e-2;

        System.out.printf("Derivada aproximada en x = %.2f para f(x) = |x|:
%.6f\n", x, derivada);

        if (posibleFallo) {
            System.out.println("⚠ Advertencia: El método puede fallar
aquí.");
            System.out.println("⚠ Posible punto no derivable (cambio
abrupto en la pendiente).");
        } else {
            System.out.println("✅ Método aplicado correctamente.");
        }
    }

    public static double f(double x) {
        return Math.abs(x); // f(x) = |x|
    }
}
```

## Ejecución del código:

```
run:
Derivada aproximada en x = 0.00 para f(x) = |x|: 0.000000
? Advertencia: El método puede fallar aquí.
? Posible punto no derivable (cambio abrupto en la pendiente).
BUILD SUCCESSFUL (total time: 0 seconds)
|
```

## Resultado:

Exacto:0, Aproximado: 0

El método falló aunque el resultado es 0, eso es **incorrecto**, porque:

- A la izquierda de 0, la derivada de  $|x|$  es  $-1$ .
- A la derecha de 0, la derivada de  $|x|$  es  $1$ .
- Pero **en  $x=0$ , la derivada no existe**, y el método simplemente hace un promedio de ambos lados.

# Métodos de integración

## Método de Trapecio

---

El método del trapecio es una técnica de integración numérica que se usa para aproximar el valor de una integral definida. Se basa en dividir el área bajo la curva en trapecios en lugar de rectángulos y calcular su área.

### Trapecio Simple

Este método se aplica sólo a un intervalo  $[a, b]$  y se basa en un solo trapecio que conecta los puntos extremos:

$$\int_a^b f(x) dx \approx \frac{b-a}{2} [f(a) + f(b)]$$

- Aproximación rápida pero poco precisa.
- Se basa en el área de un trapecio.

### Ejercicio 1

**Integral definida:**

$$\int_0^1 (2x + 3) dx$$

**Conversión a fórmula:**

$$\frac{1-0}{2} [f(0) + f(1)] = \frac{1}{2} [3 + 5] = \frac{1}{2} \times 8 = 4$$

**Código en Java:**

```
public class TrapecioSimple_Ejercicio1 {  
    public static double f(double x) {
```



```

        return 2 * x + 3;
    }

    public static double trapecioSimple(double a, double b) {
        return (b - a) / 2.0 * (f(a) + f(b));
    }

    public static void main(String[] args) {
        double a = 0, b = 1;
        double integral = trapecioSimple(a, b);
        System.out.printf("Ejercicio 1 - Resultado aproximado:
%.6f\n", integral);
    }
}

```

#### Ejecución del código:

```

Listening on 51681
User program running
Ejercicio 1 - Resultado aproximado: 4.000000
User program finished

```

#### Resultado:

$$\int_0^1 (2x + 3) dx = 4.000000 u^2$$

## Ejercicio 2

Integral definida:

$$\int_1^2 x^3 dx$$

Conversión a fórmula:

$$\frac{2-1}{2} [f(1) + f(2)] = \frac{1}{2} [1 + 8] = \frac{1}{2} \times 9 = 4.5$$

Código en Java:

```
public class TrapecioSimple_Ejercicio2 {
    public static double f(double x) {
        return Math.pow(x, 3);
    }

    public static double trapecioSimple(double a, double b) {
        return (b - a) / 2.0 * (f(a) + f(b));
    }

    public static void main(String[] args) {
        double a = 1, b = 2;
        double integral = trapecioSimple(a, b);
        System.out.printf("Ejercicio 2 - Resultado aproximado:
%.6f\n", integral);
    }
}
```

### Ejecución del código:

```
Listening on 51792
User program running
Ejercicio 2 - Resultado aproximado: 4.500000
User program finished
```

### Resultado:

$$\int_1^2 x^3 dx = 4.500000 u^2$$

### Ejercicio 3

### Integral definida:

$$\int_0^1 e^x dx$$

### Conversión a fórmula:

$$\frac{1-0}{2} [f(0) + f(1)] = \frac{1}{2} [1 + e] \approx \frac{1}{2} \times (1 + 2.7183) = 1.8592$$

### Código en Java:

```
public class Simpson13 {
    public static double f(double x) {
        return Math.exp(x);
    }

    public static double trapecioSimple(double a, double b) {
        return (b - a) / 2.0 * (f(a) + f(b));
    }

    public static void main(String[] args) {
        double a = 0, b = 1;
    }
}
```

```
        double integral = trapecioSimple(a, b);  
        System.out.printf("Ejercicio 3 - Resultado aproximado:  
%.4f\n", integral);  
    }  
}
```

**Ejecución del código:**

```
Listening on 51803  
User program running  
Ejercicio 3 - Resultado aproximado: 1.8591  
User program finished
```

**Resultado:**

$$\int_0^1 e^x dx = 1.8591 u^2$$

## Código general: segunda solución propuesta

```
import java.util.Scanner;

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

        // Coeficientes de la función cuadrática  $ax^2 + bx + c$ 
        System.out.print("Ingrese el coeficiente a (para  $ax^2$ ): ");
        double coefA = scanner.nextDouble();

        System.out.print("Ingrese el coeficiente b (para  $bx$ ): ");
        double coefB = scanner.nextDouble();

        System.out.print("Ingrese el coeficiente c: ");
        double coefC = scanner.nextDouble();

        // Límites de integración
        System.out.println("Ingrese el límite inferior de integración: ");
        double a = scanner.nextDouble();

        System.out.println("Ingrese el límite superior de integración: ");
        double b = scanner.nextDouble();

        if (a >= b) {
            System.out.println("Error: El límite inferior debe ser menor  
que el límite superior.");
            return;
        }

        // Evaluar la función en los extremos
        double fa = evaluarFuncion(coefA, coefB, coefC, a);
        double fb = evaluarFuncion(coefA, coefB, coefC, b);

        // Aplicar la fórmula del trapecio simple
        double integral = (b - a) / 2.0 * (fa + fb);

        System.out.printf("La aproximación de la integral es: %.6f\n",
            integral);
    }

    // Método que evalúa la función  $ax^2 + bx + c$  en un valor dado de x
    private static double evaluarFuncion(double a, double b, double c,
        double x) {
        return a * x * x + b * x + c;
    }
}
```

### Pseudocódigo del método de Trapecio Simple

Inicio

Leer coeficientes  $a, b, c$

Leer límite inferior a

Leer límite superior b

Si  $a \geq b$  entonces

Mostrar error y finalizar

Evaluar  $f(a)$

Evaluar  $f(b)$

Calcular integral:

```
integral = (b - a) / 2 * (f(a) + f(b))
```

Mostrar resultado de la integral

Fin

## Ejercicio 4

**Integral definida:**

$$\int_3^7 3x + 6 \, dx$$

### Ejecución del código:

```
PS C:\Users\prisc\Desktop\Metodos de integracion\Trapezio_simple> & 'C:\Program Files\Trapezio Simple\TrapezioSimple.exe'
Ingrese el coeficiente a (para ax^2): 0
Ingrese el coeficiente b (para bx): 3
Ingrese el coeficiente c: 6
Ingrese el límite inferior de integración: 3
Ingrese el límite superior de integración: 7
La aproximación de la integral es: 84.000000
PS C:\Users\prisc\Desktop\Metodos de integracion\Trapezio_simple>
```

**Resultado:**

$$\int_3^7 3x + 6 \, dx = 84$$

## Ejercicio 5

Integral definida:

$$\int_2^4 x + 8 \, dx$$

Ejecución del código:

```
PS C:\Users\prisc\Desktop\Metodos de integracion\Trapezio_simple> & 'C:\Programas de integracion\Trapezio_simple\bin' 'TrapezioSimple'
Ingrese el coeficiente a (para ax^2): 0
Ingrese el coeficiente b (para bx): 1
Ingrese el coeficiente c: 8
Ingrese el límite inferior de integración: 2
Ingrese el límite superior de integración: 4
La aproximación de la integral es: 22.000000
```

Resultado:

$$\int_2^4 x + 8 \, dx = 22$$

## Ejercicio 6: Fallo del método

Integral definida:

$$\int_3^7 3x^2 + x + 7 \, dx$$

Ejecución del código:

```
PS C:\Users\prisc\Desktop\Metodos de integracion\Trapezio_simple> & 'C:\Programas de integracion\Trapezio_simple\bin' 'TrapezioSimple'
Ingrese el coeficiente a (para ax^2): 3
Ingrese el coeficiente b (para bx): 1
Ingrese el coeficiente c: 7
Ingrese el límite inferior de integración: 3
Ingrese el límite superior de integración: 7
La aproximación de la integral es: 396.000000
```

**Resultado:** El resultado correcto de esta integral es 364, sin embargo, este método solo usa los extremos **a** y **b** para trazar una línea recta que conecta **f(a)** con **f(b)**. Si la curva real entre esos puntos no es casi lineal, entonces la línea recta se aleja bastante de la forma verdadera de la curva, y por eso el área estimada puede ser muy imprecisa, por lo que, el resultado aproximado es 396.

## Trapecio Compuesto

---

Este método mejora la precisión del trapecio simple dividiendo el intervalo [a,b] en n subintervalos iguales, y aplicando la fórmula:

$$\int_a^b f(x) dx \approx \frac{h}{2} \left[ f(x_0) + 2 \sum_{i=1}^{n-1} f(x_i) + f(x_n) \right]$$

donde:

- $h = \frac{b-a}{n}$
- $x_0 = a, x_n = b$
- $x_1, x_2, \dots, x_{n-1}$  son los puntos intermedios

### Pseudocódigo del método de Trapecio Compuesto

Inicio

Leer coeficientes a, b, c

Leer límite inferior limInf

Leer límite superior limSup

Leer número de subintervalos n

Si  $n \leq 0$  o  $\text{limInf} \geq \text{limSup}$  entonces

Mostrar error y finalizar

Calcular  $h = (\text{limSup} - \text{limInf}) / n$

Inicializar suma =  $f(\text{limInf}) + f(\text{limSup})$

Para i desde 1 hasta n-1 hacer

$x = \text{limInf} + i * h$

    suma = suma +  $2 * f(x)$

Fin Para

Calcular integral:

    integral =  $(h / 2) * \text{suma}$

Mostrar resultado de la integral

Fin



## Código general: primera solución propuesta

```
import java.util.Scanner;

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

        // Lectura de los coeficientes de la función cuadrática:  $ax^2 + bx + c$ 
        System.out.println("Ingrese el coeficiente a (para  $ax^2$ ):");
        double coefA = scanner.nextDouble();

        System.out.println("Ingrese el coeficiente b (para  $bx$ ):");
        double coefB = scanner.nextDouble();

        System.out.println("Ingrese el coeficiente c:");
        double coefC = scanner.nextDouble();

        // Lectura de los límites de integración
        System.out.println("Ingrese el límite inferior de integración:");
        double limInf = scanner.nextDouble();

        System.out.println("Ingrese el límite superior de integración:");
        double limSup = scanner.nextDouble();

        // Lectura del número de subintervalos ( $n > 0$ )
        System.out.println("Ingrese el número de subintervalos ( $n > 0$ ):");
        int n = scanner.nextInt();

        if (n <= 0 || limInf >= limSup) {
            System.out.println("Datos inválidos. Asegúrese de que  $n > 0$  y que el límite inferior sea menor al superior.");
            return;
        }

        // Cálculo del ancho del subintervalo
        double h = (limSup - limInf) / n;
        double suma = 0.0;

        // Evaluar  $f(x)$  en los extremos
        suma += evaluarFuncion(coefA, coefB, coefC, limInf);
        suma += evaluarFuncion(coefA, coefB, coefC, limSup);

        // Sumar evaluaciones en los puntos intermedios (multiplicados por 2)
```

```

    for (int i = 1; i < n; i++) {
        double x = limInf + i * h;
        suma += 2 * evaluarFuncion(coefA, coefB, coefC, x);
    }

    // Aproximación de la integral mediante la regla del trapecio
    compuesta
    double integral = (h / 2.0) * suma;
    System.out.printf("La aproximación de la integral es: %.6f\n",
integral);
}

// Método que evalúa la función cuadrática para un valor dado de x
private static double evaluarFuncion(double a, double b, double c,
double x) {
    return a * x * x + b * x + c;
}

```

## Ejercicio 1

Integral definida:

$$\int_0^1 6x^2 + 4x + 12 \, dx$$

Ejecución del código:

```

PS C:\Users\prisc\Desktop\Metodos de integracion\Trapezio> & 'C:\Pr
gracion\Trapezio\bin' 'Trapezio'
Ingrese el coeficiente a (para ax^2): 6
Ingrese el coeficiente b (para bx): 4
Ingrese el coeficiente c: 12
Ingrese el límite inferior de integración: 0
Ingrese el límite superior de integración: 1
Ingrese el número de subintervalos (n > 0): 10
La aproximación de la integral es: 16.010000

```

Resultado:

$$\int_0^1 6x^2 + 4x + 12 \, dx = 16$$

## Ejercicio 2

Integral definida:

$$\int_4^8 x^2 + 13x \, dx$$

Ejecución del código:

```
PS C:\Users\prisc\Desktop\Metodos de integracion\Trapezio> & 'C:\Users\prisc\Desktop\Metodos de integracion\Trapezio\bin\Trapezio'
Ingrese el coeficiente a (para ax^2): 1
Ingrese el coeficiente b (para bx): 13
Ingrese el coeficiente c: 0
Ingrese el límite inferior de integración: 4
Ingrese el límite superior de integración: 8
Ingrese el número de subintervalos (n > 0): 5
La aproximación de la integral es: 461.760000
```

Resultado:

$$\int_4^8 x^2 + 13x \, dx = 461.76$$

## Ejercicio 3

Integral definida:

$$\int_0^9 x^2 + 2x + 1 \, dx$$

Ejecución del código:

```
PS C:\Users\prisc\Desktop\Metodos de integracion\Trapezio> & 'C:\Users\prisc\Desktop\Metodos de integracion\Trapezio\bin\Trapezio'
Ingrese el coeficiente a (para ax^2): 1
Ingrese el coeficiente b (para bx): 2
Ingrese el coeficiente c: 1
Ingrese el límite inferior de integración: 0
Ingrese el límite superior de integración: 9
Ingrese el número de subintervalos (n > 0): 5
La aproximación de la integral es: 337.860000
```

Resultado:

$$\int_0^9 x^2 + 2x + 1 \, dx = 337.86$$

## Segunda solución propuesta: código enfocado

### Ejercicio 4

Integral definida:

$$\int_0^2 (x^2 + 2) dx$$

**Datos:**  $n = 4$  subintervalos  $\rightarrow h = \frac{2-0}{4} = 0.5$

**Puntos:**  $x_0 = 0, x_1 = 0.5, x_2 = 1.0, x_3 = 1.5, x_4 = 2.0$

Conversión a fórmula:

$$\frac{0.5}{2} [f(0) + 2f(0.5) + 2f(1) + 2f(1.5) + f(2)]$$

Código en Java:

```
public class TrapecioCompuesto_Ejercicio4 {
    public static double f(double x) {
        return x * x + 2;
    }

    public static double trapecioCompuesto(double a, double b, int
n) {
        double h = (b - a) / n;
        double sum = f(a) + f(b);

        for (int i = 1; i < n; i++) {
            double xi = a + i * h;
            sum += 2 * f(xi);
        }

        return (h / 2.0) * sum;
    }

    public static void main(String[] args) {
        double a = 0, b = 2;
        int n = 4;
        double integral = trapecioCompuesto(a, b, n);
        System.out.printf("Ejercicio 4 - Resultado aproximado:");
```

```
%.4f\n", integral);
    }
}
```

Ejecución del código:

```
Listening on 52344
User program running
Ejercicio 4 - Resultado aproximado: 6.7500
User program finished
```

Resultado:

$$\int_0^2 (x^2 + 2) dx = 6.7500 u^2$$

### Ejercicio 5

Integral definida:

$$\int_1^3 \sqrt{x^2 + 1} dx$$

Datos:  $n = 4, h = 0.5$

Puntos:  $x_0 = 1.0, x_1 = 1.5, x_2 = 2.0, x_3 = 2.5, x_4 = 3.0$

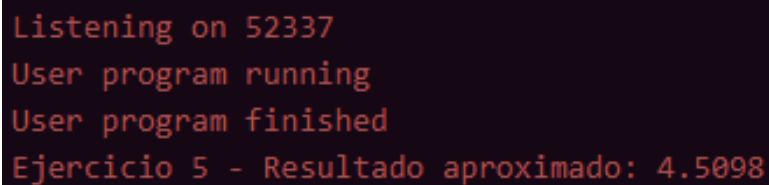
Conversión a fórmula:

$$\frac{0.5}{2} [f(1.0) + 2f(1.5) + 2f(2.0) + 2f(2.5) + f(3.0)]$$

### Código en Java:

```
public class TrapecioCompuesto_Ejercicio5 {  
    public static double f(double x) {  
        return Math.sqrt(x * x + 1);  
    }  
  
    public static double trapecioCompuesto(double a, double b, int  
n) {  
        double h = (b - a) / n;  
        double sum = f(a) + f(b);  
        for (int i = 1; i < n; i++) {  
            double xi = a + i * h;  
            sum += 2 * f(xi);  
        }  
        return (h / 2.0) * sum;  
    }  
  
    public static void main(String[] args) {  
        double a = 1, b = 3;  
        int n = 4;  
        double integral = trapecioCompuesto(a, b, n);  
        System.out.printf("Ejercicio 5 - Resultado aproximado:  
%.6f\n", integral);  
    }  
}
```

### Ejecución del código:



```
Listening on 52337  
User program running  
User program finished  
Ejercicio 5 - Resultado aproximado: 4.5098
```

### Resultado:

$$\int_1^3 \sqrt{x^2 + 1} dx = 4.5098 u^2$$

## Ejercicio 6: Fallo del método

Integral definida:

$$\int_{-1}^1 \frac{1}{x} dx$$

**Problema:** La función tiene una discontinuidad en  $x=0$ , por lo tanto, el método no es aplicable, o puede lanzar error o dar un valor incorrecto.

Conversión a fórmula (inválida):

$$h = \frac{2}{10} = 0.2 \Rightarrow f(0) \text{ no está definido}$$

Código en Java:

```
public class TrapecioCompuesto_Ejercicio6 {
    public static double f(double x) {
        return 1 / x; // Discontinuidad en x = 0
    }

    public static double trapecioCompuesto(double a, double b, int
n) {
        double h = (b - a) / n;
        double sum = f(a) + f(b);
        for (int i = 1; i < n; i++) {
            double xi = a + i * h;
            sum += 2 * f(xi); // Posible división entre cero
        }
        return (h / 2.0) * sum;
    }

    public static void main(String[] args) {
        double a = -1, b = 1;
        int n = 10;
        double integral = trapecioCompuesto(a, b, n);
        System.out.printf("Ejercicio 6 - Resultado aproximado:
%.4f\n", integral);
    }
}
```

Ejecución del código:

```
Listening on 52366  
User program running  
Ejercicio 6 - Resultado aproximado: Infinity  
User program finished
```

Resultado:

$$\int_{-1}^1 \frac{1}{x} dx = \text{Error}$$



## Regla de Simpson

Otra forma de obtener una estimación más exacta de una integral consiste en usar polinomios de grado superior para unir los puntos. Por ejemplo, si hay otro punto a la mitad entre  $f(a)$  y  $f(b)$ , los tres puntos se pueden unir con una parábola. Si hay dos puntos igualmente espaciados entre  $f(a)$  y  $f(b)$ , los cuatro puntos se pueden unir mediante un polinomio de tercer grado. Las fórmulas que resultan de tomar las integrales bajo estos polinomios se conocen como *reglas de Simpson*. (Chapra & Canale, 2015, 479)

### Regla de Simpson 1/3

Este método aproxima una integral definida usando un polinomio cuadrático sobre dos subintervalos. La fórmula para un solo segmento (un tramo con 3 puntos) es:

$$\int_a^b f(x) dx \approx \frac{h}{3} \left[ f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right]$$

donde:

- $h = \frac{b-a}{2}$
- Se evalúa en:  $x_0 = a, x_1 = \frac{a+b}{2}, x_2 = b$

### Pseudocódigo de Regla de Simpson 1/3

Inicio

Definir función  $f(x) = x^2 + 1$

Definir  $a = 0, b = 2$

$h = (b - a) / 2$

$x_0 = a$

$x_1 = a + h$

$x_2 = b$

$integral = (h / 3) * [f(x_0) + 4f(x_1) + f(x_2)]$

Imprimir "Resultado aproximado: ", integral

Fin

## Ejercicio 1

Integral definida:

$$\int_0^2 x^2 + 1 \, dx$$

Conversión a fórmula:

$$a = 0, b = 2, h = 1$$

$$\text{Evaluar en } x_0 = 0, x_1 = 1, x_2 = 2$$

$$\text{Aproximación} = \frac{1}{3} [f(0) + 4f(1) + f(2)]$$

Código en Java:

```
public class Simpson13_Ejercicio1 {

    public static double f(double x) {
        return x * x + 1;
    }

    public static double simpson13(double a, double b) {
        double h = (b - a) / 2.0;
        double x0 = a;
        double x1 = a + h;
        double x2 = b;

        double result = (h / 3.0) * (f(x0) + 4 * f(x1) + f(x2));
        return result;
    }

    public static void main(String[] args) {
        double a = 0;
        double b = 2;

        double integral = simpson13(a, b);
        System.out.printf("Ejercicio 1 - Resultado aproximado:
%.6f\n", integral);
    }
}
```

Ejecución del código:

```
Listening on 51021
User program running
Ejercicio 1 - Resultado aproximado: 4.666667
User program finished
```

Resultado:

$$\int_0^2 x^2 + 1 \, dx = 4.666667 \, u^2$$

$$\frac{1}{3} [f(0) + 4f(1) + f(2)] = 4.666667 \, u^2$$

## Ejercicio 2

Integral definida:

$$\int_1^3 \ln(x^2 + 1) \, dx$$

Conversión a fórmula:

$$a = 1, b = 3, h = 1$$

$$\text{Evaluar en } x_0 = 1, x_1 = 2, x_2 = 3$$

$$\text{Aproximación} = \frac{1}{3} [f(1) + 4f(2) + f(3)]$$

Código en Java:

```
public class Simpson13_Ejercicio2 {

    public static double f(double x) {
        return Math.log(x * x + 1);
    }

    public static double simpson13(double a, double b) {
        double h = (b - a) / 2.0;
        double x0 = a;
        double x1 = a + h;
        double x2 = b;
```

```

        double result = (h / 3.0) * (f(x0) + 4 * f(x1) + f(x2));
        return result;
    }

    public static void main(String[] args) {
        double a = 1;
        double b = 3;

        double integral = simpson13(a, b);
        System.out.printf("Ejercicio 2 - Resultado aproximado:
%.6f\n", integral);
    }
}

```

**Ejecución del código:**

```

Listening on 51376
User program running
Ejercicio 2 - Resultado aproximado: 3.144495
User program finished

```

**Resultado:**

$$\int_1^3 \ln(x^2 + 1) dx = 3.144495 u^2$$

$$\frac{1}{3} [f(1) + 4f(2) + f(3)] = 4.144495 u^2$$

### Ejercicio 3

**Integral definida:**

$$\int_0^2 e^{-x^2} dx$$

**Conversión a fórmula:**

$$a = 0, b = 2, h = 1$$

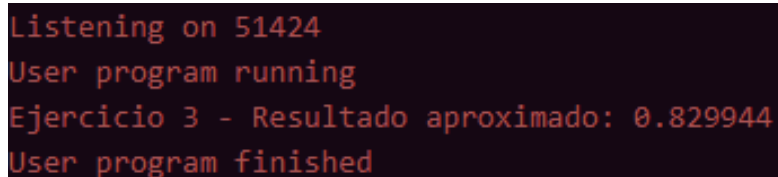
$$\text{Evaluar en } x_0 = 0, x_1 = 1, x_2 = 2$$

$$\text{Aproximación} = \frac{1}{3} [f(0) + 4f(1) + f(2)]$$

### Código en Java:

```
public class Simpson13_Ejercicio3 {  
    public static double f(double x) {  
        return Math.exp(-x * x);  
    }  
  
    public static double simpson13(double a, double b) {  
        double h = (b - a) / 2.0;  
        double x0 = a;  
        double x1 = a + h;  
        double x2 = b;  
        return (h / 3.0) * (f(x0) + 4 * f(x1) + f(x2));  
    }  
  
    public static void main(String[] args) {  
        double a = 0, b = 2;  
        double integral = simpson13(a, b);  
        System.out.printf("Ejercicio 3 - Resultado aproximado:  
%.6f\n", integral);  
    }  
}
```

### Ejecución del código:



```
Listening on 51424  
User program running  
Ejercicio 3 - Resultado aproximado: 0.829944  
User program finished
```

### Resultado:

$$\int_0^2 e^{-x^2} dx = 0.829944 u^2$$
$$\frac{1}{3} [f(0) + 4f(1) + f(2)] = 0.829944 u^2$$

## Ejercicio 4

Integral definida:

$$\int_1^3 \sqrt{x+1} dx$$

Conversión a fórmula:

$$a = 1, b = 3, h = 1$$

$$\text{Evaluar en } x_0 = 1, x_1 = 2, x_2 = 3$$

$$\text{Aproximación} = \frac{1}{3} [f(1) + 4f(2) + f(3)]$$

Código en Java:

```
public class Simpson13 {
    public static double f(double x) {
        return Math.sqrt(x + 1);
    }

    public static double simpson13(double a, double b) {
        double h = (b - a) / 2.0;
        double x0 = a;
        double x1 = a + h;
        double x2 = b;
        return (h / 3.0) * (f(x0) + 4 * f(x1) + f(x2));
    }

    public static void main(String[] args) {
        double a = 1, b = 3;
        double integral = simpson13(a, b);
        System.out.printf("Ejercicio 4 - Resultado aproximado:
        %.6f\n", integral);
    }
}
```

Ejecución del código:

```
Listening on 51473
User program running
User program finished
Ejercicio 4 - Resultado aproximado: 3.447472
```

Resultado:

$$\int_1^3 \sqrt{x+1} dx = 3.447472 u^2$$
$$\frac{1}{3} [f(1) + 4f(2) + f(3)] = 3.447472 u^2$$

### Ejercicio 5

Integral definida:

$$\int_0^2 \frac{1}{1+x^2} dx$$

Conversión a fórmula:

$$a = 0, b = 2, h = 1$$

$$\text{Evaluar en } x_0 = 0, x_1 = 1, x_2 = 2$$

$$\text{Aproximación} = \frac{1}{3} [f(0) + 4f(1) + f(2)]$$

Código en Java:

```
public class Simpson13 {
    public static double f(double x) {
        return 1 / (1 + x * x);
    }

    public static double simpson13(double a, double b) {
```

```

        double h = (b - a) / 2.0;
        double x0 = a;
        double x1 = a + h;
        double x2 = b;
        return (h / 3.0) * (f(x0) + 4 * f(x1) + f(x2));
    }

    public static void main(String[] args) {
        double a = 0, b = 2;
        double integral = simpson13(a, b);
        System.out.printf("Ejercicio 5 - Resultado aproximado:
%.6f\n", integral);
    }
}

```

Ejecución del código:

```

Listening on 51503
User program running
Ejercicio 5 - Resultado aproximado: 1.066667
User program finished

```

Resultado:

$$\int_0^2 \frac{1}{1+x^2} dx = 1.066667 u^2$$

$$\frac{1}{3} [f(0) + 4f(1) + f(2)] = 1.066667 u^2$$

### Ejercicio 6: Fallo del método

Integral definida:

$$\int_0^3 \ln(x) dx$$



### Conversión a fórmula:

$$a = 0, b = 3, h = 1.5$$

$$\text{Evaluar en } x_0 = 0, x_1 = 1.5, x_2 = 3$$

$$\text{Aproximación} = \frac{1.5}{3} [f(0) + 4f(1.5) + f(3)]$$

### Código en Java:

```
public class Simpson13_Ejercicio6 {
    public static double f(double x) {
        return Math.log(x); // Este caso puede fallar por ln(0)
    }

    public static double simpson13(double a, double b) {
        double h = (b - a) / 2.0;
        double x0 = a;
        double x1 = a + h;
        double x2 = b;
        return (h / 3.0) * (f(x0) + 4 * f(x1) + f(x2));
    }

    public static void main(String[] args) {
        double a = 0, b = 3;
        double integral = simpson13(a, b);
        System.out.printf("Ejercicio 6 - Resultado aproximado:
%.6f\n", integral);
    }
}
```

### Ejecución del código:

```
Listening on 51529
User program running
User program finished
Ejercicio 6 - Resultado aproximado: -Infinity
```

### Resultado:

$$\int_0^3 \ln(x) dx = \text{Error}$$

$$\frac{1.5}{3} [f(0) + 4f(1.5) + f(3)] = \text{Error}$$

### Regla de Simpson $\frac{3}{8}$

La regla de Simpson  $\frac{3}{8}$  es un método de integración numérica que aproxima una integral definida usando tres subintervalos iguales (por eso necesita que el número de subintervalos sea múltiplo de 3).

De manera similar a la obtención de la regla del trapecio y Simpson  $\frac{1}{3}$ , es posible ajustar un polinomio de Lagrange de tercer grado a cuatro puntos e integrarlo:

$$I = \int_a^b f(x)dx \equiv \int_a^b f_3(x)dx$$

para obtener

$$I = \frac{3h}{8} [f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)]$$

donde  $h = \frac{b-a}{3}$ . Esta ecuación se llama regla de Simpson  $\frac{3}{8}$  debido a que  $h$  se multiplica por  $\frac{3}{8}$ . Ésta es la tercera fórmula de integración cerrada de Newton-Cotes. (Chapra & Canale, 2015, 484)

### Pseudocódigo de Regla de Simpson $\frac{3}{8}$

Inicio

Definir función  $f(x) = x^3 + 2x^2$

Definir  $a = 0$ ,  $b = 3$

$h = (b - a) / 3$

$x_0 = a$

$x_1 = a + h$

$x_2 = a + 2h$

$x_3 = b$

$integral = (3h / 8) * [f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)]$

Imprimir "Resultado aproximado: ",  $integral$

Fin

## Ejercicio 1

Integral definida:

$$\int_0^3 (x^3 + 2x^2) dx$$

Conversión a fórmula:

$$a = 0, b = 3, h = \frac{3-0}{3} = 1$$

$$\text{Evaluar en } x_0 = 0, x_1 = 1, x_2 = 2, x_3 = 3$$

$$\text{Aproximación} = \frac{3(1)}{8} [f(0) + 3f(1) + 3f(2) + f(3)]$$

Código en Java:

```
public class Simpson38 {

    // f(x):
    public static double f(double x) {
        return Math.pow(x, 3) + 2 * Math.pow(x, 2);
    }

    public static double simpson38(double a, double b) {
        double h = (b - a) / 3.0;
        double x0 = a;
        double x1 = a + h;
        double x2 = a + 2 * h;
        double x3 = b;

        double result = (3 * h / 8.0) * (f(x0) + 3 * f(x1) + 3 *
f(x2) + f(x3));
        return result;
    }

    public static void main(String[] args) {
        double a = 0; // Límite inferior
        double b = 3; // Límite superior

        double integral = simpson38(a, b);
        System.out.printf("Resultado aproximado: %.4f%n", integral);
    }
}
```

Ejecución del código:

```
Listening on 52142
User program running
Resultado aproximado: 38.2500
User program finished
```

Resultado:

$$\int_0^3 (x^3 + 2x^2) dx = 38.2500 u^2$$

$$\frac{3(1)}{8} [f(0) + 3f(1) + 3f(2) + f(3)] = 38.2500 u^2$$

## Ejercicio 2

Integral definida:

$$\int_1^4 \ln(x + 1) dx$$

Conversión a fórmula:

$$a = 1, b = 4, h = \frac{4-1}{3} = 1$$

$$\text{Evaluar en } x_0 = 1, x_1 = 2, x_2 = 3, x_3 = 4$$

$$\text{Aproximación} = \frac{3(1)}{8} [f(1) + 3f(2) + 3f(3) + f(4)]$$

Código en Java:

```
public class Simpson38 {

    // f(x):
    public static double f(double x) {
        return Math.log(x + 1);
    }
}
```

```

public static double simpson38(double a, double b) {
    double h = (b - a) / 3.0;
    double x0 = a;
    double x1 = a + h;
    double x2 = a + 2 * h;
    double x3 = b;

    double result = (3 * h / 8.0) * (f(x0) + 3 * f(x1) + 3 *
f(x2) + f(x3));
    return result;
}

public static void main(String[] args) {
    double a = 1; // Límite inferior
    double b = 4; // Límite superior

    double integral = simpson38(a, b);
    System.out.printf("Resultado aproximado: %.4f\n", integral);
}
}

```

Ejecución del código:

```

Listening on 52258
User program running
Resultado aproximado: 3.6590
User program finished

```

Resultado:

$$\int_1^4 \ln(x + 1) dx = 3.6609 u^2$$

$$\frac{3(1)}{8} [f(1) + 3f(2) + 3f(3) + f(4)] = 3.6590 u^2$$

### Ejercicio 3

Integral definida:

$$\int_0^3 e^{-x^2} dx$$

Conversión a fórmula:

$$a = 0, b = 3, h = \frac{3-0}{3} = 1$$

$$\text{Evaluar en } x_0 = 0, x_1 = 1, x_2 = 2, x_3 = 3$$

$$\text{Aproximación} = \frac{3(1)}{8} [f(0) + 3f(1) + 3f(2) + f(3)]$$

Código en Java:

```
public class Simpson38 {

    // f(x):
    public static double f(double x) {
        return Math.exp(-x * x);
    }

    public static double simpson38(double a, double b) {
        double h = (b - a) / 3.0;
        double x0 = a;
        double x1 = a + h;
        double x2 = a + 2 * h;
        double x3 = b;

        double result = (3 * h / 8.0) * (f(x0) + 3 * f(x1) + 3 *
f(x2) + f(x3));
        return result;
    }

    public static void main(String[] args) {
        double a = 0; // Límite inferior
        double b = 3; // Límite superior

        double integral = simpson38(a, b);
        System.out.printf("Resultado aproximado: %.4f%n", integral);
    }
}
```

Ejecución del código:

```
Listening on 52338
User program running
Resultado aproximado: 0.8095
User program finished
```

Resultado:

$$\int_0^3 e^{-x^2} dx = 0.8862 u^2$$

$$\frac{3(1)}{8} [f(0) + 3f(1) + 3f(2) + f(3)] = 0.8095 u^2$$

#### Ejercicio 4

Integral definida:

$$\int_2^5 \sqrt{x} dx$$

Conversión a fórmula:

$$a = 2, b = 5, h = \frac{5-2}{3} = 1$$

$$\text{Evaluar en } x_0 = 2, x_1 = 3, x_2 = 4, x_3 = 5$$

$$\text{Aproximación} = \frac{3(1)}{8} [f(2) + 3f(3) + 3f(4) + f(5)]$$

Código en Java:

```
public class Simpson38 {

    // f(x):
    public static double f(double x) {
        return Math.sqrt(x);
    }
}
```

```

public static double simpson38(double a, double b) {
    double h = (b - a) / 3.0;
    double x0 = a;
    double x1 = a + h;
    double x2 = a + 2 * h;
    double x3 = b;

    double result = (3 * h / 8.0) * (f(x0) + 3 * f(x1) + 3 *
f(x2) + f(x3));
    return result;
}

public static void main(String[] args) {
    double a = 2; // Límite inferior
    double b = 5; // Límite superior

    double integral = simpson38(a, b);
    System.out.printf("Resultado aproximado: %.4f\n", integral);
}
}

```

Ejecución del código:

```

Listening on 52382
User program running
User program finished
Resultado aproximado: 5.5674

```

Resultado:

$$\int_2^5 \sqrt{x} dx = 5.5679 u^2$$

$$\frac{3(1)}{8} [f(2) + 3f(3) + 3f(4) + f(5)] = 5.5674 u^2$$



## Ejercicio 5

Integral definida:

$$\int_0^3 \frac{1}{1+x^2} dx$$

Conversión a fórmula:

$$a = 0, b = 3, h = \frac{3-0}{3} = 1$$

$$\text{Evaluar en } x_0 = 0, x_1 = 1, x_2 = 2, x_3 = 3$$

$$\text{Aproximación} = \frac{3(1)}{8} [f(0) + 3f(1) + 3f(2) + f(3)]$$

Código en Java:

```
public class Simpson38 {

    // f(x):
    public static double f(double x) {
        return 1 / (1 + x * x);
    }

    public static double simpson38(double a, double b) {
        double h = (b - a) / 3.0;
        double x0 = a;
        double x1 = a + h;
        double x2 = a + 2 * h;
        double x3 = b;

        double result = (3 * h / 8.0) * (f(x0) + 3 * f(x1) + 3 *
f(x2) + f(x3));
        return result;
    }

    public static void main(String[] args) {
        double a = 0; // Límite inferior
        double b = 3; // Límite superior

        double integral = simpson38(a, b);
        System.out.printf("Resultado aproximado: %.4f%n", integral);
    }
}
```

Ejecución del código:

```
Listening on 52416
User program running
Resultado aproximado: 1.2000
User program finished
```

Resultado:

$$\int_0^3 \frac{1}{1+x^2} dx = 1.2490 u^2$$

$$\frac{3(1)}{8} [f(0) + 3f(1) + 3f(2) + f(3)] = 1.2000 u^2$$

### Ejercicio 6: Fallo del método

Integral definida:

$$\int_0^2 \cos(x) dx$$

Conversión a fórmula:

$a = 0$ ,  $b = 2$ ,  $h = \frac{2-0}{3} = \frac{2}{3}$ . No se cumple la condición de que el número de subintervalos sea múltiplo de 3 si se quiere aplicar más tramos.

Código en Java:

```
public class Simpson38 {

    // f(x):
    public static double f(double x) {
        return Math.cos(x);
    }
}
```

```

public static double simpson38(double a, double b) {
    double h = (b - a) / 3.0;
    double x0 = a;
    double x1 = a + h;
    double x2 = a + 2 * h;
    double x3 = b;

    double result = (3 * h / 8.0) * (f(x0) + 3 * f(x1) + 3 *
f(x2) + f(x3));
    return result;
}

public static void main(String[] args) {
    double a = 0; // Límite inferior
    double b = 2; // Límite superior

    if ((b - a) % 3 != 0) {
        System.out.println("Error: El intervalo no es divisible
en 3 subintervalos.");
    } else {
        double integral = simpson38(a, b);
        System.out.printf("Resultado aproximado: %.4f%n",
integral);
    }
}
}

```

Ejecución del código:

```

Listening on 52455
User program running
User program finished
Error: El intervalo no es divisible en 3 subintervalos.

```

Resultado:

Error: el intervalo no es divisible en 3 subintervalos.

## Método de la Cuadratura Gaussiana

La cuadratura gaussiana es un método de integración numérica que aproxima una integral definida usando puntos especiales (llamados nodos de Gauss) y sus respectivos pesos. Estos puntos no están equidistantes como en otros métodos, y se eligen para maximizar la precisión. Por ejemplo, usando 2 o 3 puntos se pueden obtener resultados muy exactos para muchos polinomios.

### Pseudocódigo de Cuadratura Gaussiana

Inicio

```
Definir a = 0, b = 1
Definir x = [-1/√3, 1/√3] // Puntos de Gauss
Definir w = [1, 1]         // Pesos
integral = 0
```

Para i desde 0 hasta 1 hacer

```
xi = ((b - a)/2) * x[i] + (a + b)/2
fx = xi2 + 2*xi + 1 // f(x)
integral = integral + w[i] * fx
```

Fin Para

```
integral = integral * (b - a)/2
Imprimir "Resultado aproximado: ", integral
```

Fin

## Ejercicio 1

Integral definida:

$$\int_0^1 (x^2 + 2x + 1) dx$$

Conversión a fórmula:

$$x_i = \frac{(1-0)}{2} t_i + \frac{0+1}{2}, \quad t_i = \pm \frac{1}{\sqrt{3}}, \quad w_i = 1$$

$$\int_0^1 f(x) dx \approx \frac{1}{2} [f(x_1) + f(x_2)]$$

Código en java:

```
public class cuadraturaGaussiana1 {  
    public static void main(String[] args) {  
        double a = 0, b = 1;  
        double[] x = {-1/Math.sqrt(3), 1/Math.sqrt(3)};  
        double[] w = {1, 1};  
        double integral = 0;  
        for (int i = 0; i < 2; i++) {  
            double xi = ((b - a) / 2) * x[i] + (a + b) / 2;  
            double fx = xi*xi + 2*xi + 1;  
            integral += w[i] * fx;  
        }  
        integral *= (b - a) / 2;  
        System.out.printf("Resultado aproximado: %.4f\n",  
integral);  
    }  
}
```

Ejecución del código:

```
PS C:\Users\oscar> & 'C:\Prog  
6' '-cp' 'C:\Users\oscar\AppData  
Resultado aproximado: 2.3333
```

Resultado:

Exacto: 2.3333, Aproximado: 2.3333

## Ejercicio 2

Integral definida:

$$\int_{-1}^2 (3x^3 - 2x^2 + x + 5) dx$$

Conversión a fórmula:

$$x_i = \frac{(2 - (-1))}{2} t_i + \frac{-1 + 2}{2}, \quad t_i = \pm \frac{1}{\sqrt{3}}$$

$$\int_{-1}^2 f(x) dx \approx \frac{3}{2} [f(x_1) + f(x_2)]$$

Código en java:

```
public class cuadraturaGaussiana2 {
    public static void main(String[] args) {
        double a = -1, b = 2;
        double[] x = {-1/Math.sqrt(3), 1/Math.sqrt(3)};
        double[] w = {1, 1};
        double integral = 0;
        for (int i = 0; i < 2; i++) {
            double xi = ((b - a) / 2) * x[i] + (a + b) / 2;
            double fx = 3*Math.pow(xi,3) - 2*Math.pow(xi,2) + xi + 5;
            integral += w[i] * fx;
        }
        integral *= (b - a) / 2;
        System.out.printf("Resultado aproximado: %.4f\n", integral);
    }
}
```

Ejecución del código:

```
PS C:\Users\oscar> & 'C:\Program Files\Java\jdk-7' '-cp' 'C:\Users\oscar\AppData\Local\Temp' 'C:\Users\oscar\src\ejercicio2\CuadraturaGaussiana2.java'
Resultado aproximado: 21.7500
```

Resultado:

Exacto: 21.7500, Aproximado: 21.7500

### Ejercicio 3

Integral definida:

$$\int_0^2 (4x^4 - x^2 + 2) dx$$

Conversión a fórmula:

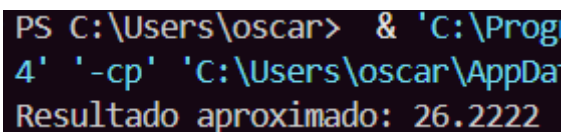
$$x_i = \frac{(2-0)}{2}t_i + \frac{0+2}{2}, \quad t_i = \pm \frac{1}{\sqrt{3}}$$

$$\int_0^2 f(x) dx \approx 1 \cdot [f(x_1) + f(x_2)]$$

Código en java:

```
public class cuadraturaGaussiana3 {  
    public static void main(String[] args) {  
        double a = 0, b = 2;  
        double[] x = {-1/Math.sqrt(3), 1/Math.sqrt(3)};  
        double[] w = {1, 1};  
        double integral = 0;  
        for (int i = 0; i < 2; i++) {  
            double xi = ((b - a) / 2) * x[i] + (a + b) / 2;  
            double fx = 4*Math.pow(xi,4) - Math.pow(xi,2) + 2;  
            integral += w[i] * fx;  
        }  
        integral *= (b - a) / 2;  
        System.out.printf("Resultado aproximado: %.4f\n", integral);  
    }  
}
```

Ejecución del código:



```
PS C:\Users\oscar> & 'C:\Prog4' '-cp' 'C:\Users\oscar\AppData...'  
Resultado aproximado: 26.2222
```

Resultado:

Exacto: 26.93333, Aproximado: 26.22222

## Ejercicio 4

Integral definida:

$$\int_1^3 (2x + 1) dx$$

Conversión a fórmula:

$$x_i = \frac{(3-1)}{2} t_i + \frac{1+3}{2}, \quad t_i = \pm \frac{1}{\sqrt{3}}$$

$$\int_1^3 f(x) dx \approx 1 \cdot [f(x_1) + f(x_2)]$$

Código en java:

```
public class cuadraturaGaussiana4 {  
    public static void main(String[] args) {  
        double a = 1, b = 3;  
        double[] x = {-1/Math.sqrt(3), 1/Math.sqrt(3)};  
        double[] w = {1, 1};  
        double integral = 0;  
        for (int i = 0; i < 2; i++) {  
            double xi = ((b - a) / 2) * x[i] + (a + b) / 2;  
            double fx = 2*xi + 1;  
            integral += w[i] * fx;  
        }  
        integral *= (b - a) / 2;  
        System.out.printf("Resultado aproximado: %.4f\n", integral);  
    }  
}
```

Ejecución del código:

```
PS C:\Users\oscar> & 'C:\Program Files\Java\jdk-5\bin\java.exe' '-cp' 'C:\Users\oscar\AppData\Local\Temp\1\j2se-5.0.50-windows-i586-jre.exe' 'cuadraturaGaussiana4' 1 3  
Resultado aproximado: 10.0000
```

Resultado:

Exacto: 10, Aproximado: 10.0



## Ejercicio 5

Integral definida:

$$\int_{-2}^0 (x^3 + x^2 + x + 1) dx$$

Conversión a fórmula:

$$x_i = \frac{(0 - (-2))}{2} t_i + \frac{-2 + 0}{2}, \quad t_i = \pm \frac{1}{\sqrt{3}}$$

$$\int_{-2}^0 f(x) dx \approx 1 \cdot [f(x_1) + f(x_2)]$$

Código en java:

```
public class cuadraturaGaussiana5 {  
    public static void main(String[] args) {  
        double a = -2, b = 0;  
        double[] x = {-1/Math.sqrt(3), 1/Math.sqrt(3)};  
        double[] w = {1, 1};  
        double integral = 0;  
        for (int i = 0; i < 2; i++) {  
            double xi = ((b - a) / 2) * x[i] + (a + b) / 2;  
            double fx = Math.pow(xi,3) + Math.pow(xi,2) + xi + 1;  
            integral += w[i] * fx;  
        }  
        integral *= (b - a) / 2;  
        System.out.printf("Resultado aproximado: %.4f\n", integral);  
    }  
}
```

Ejecución del código:

```
PS C:\Users\oscar> & 'C:\Prog  
2' '-cp' 'C:\Users\oscar\AppData  
Resultado aproximado: -1.3333
```

Resultado:

Exacto: -1.33333, Aproximado: -1.33333

## Ejercicio 6: Fallo del método

Integral definida:

$$\int_{-1}^1 (x^6 - 4x^5 + x^3 - x + 1) dx$$

Conversión a fórmula:

$$x_i = \frac{(1 - (-1))}{2} t_i + \frac{-1 + 1}{2}, \quad t_i = \pm \frac{1}{\sqrt{3}}$$

$$\int_{-1}^1 f(x) dx \approx 1 \cdot [f(x_1) + f(x_2)]$$

Código en java:

```
public class cuadraturaGaussiana6 {  
    public static void main(String[] args) {  
        double a = -1, b = 1;  
        double[] x = {-1/Math.sqrt(3), 1/Math.sqrt(3)};  
        double[] w = {1, 1};  
        double integral = 0;  
        for (int i = 0; i < 2; i++) {  
            double xi = ((b - a) / 2) * x[i] + (a + b) / 2;  
            double fx = Math.pow(xi,6) - 4*Math.pow(xi,5) +  
Math.pow(xi,3) - xi + 1;  
            integral += w[i] * fx;  
        }  
        integral *= (b - a) / 2;  
        System.out.printf("Resultado del gauss (fallido): %.4f\n",  
integral);  
    }  
}
```

Ejecución del código:

```
PS C:\Users\oscar> & 'C:\Program File  
9' '-cp' 'C:\Users\oscar\AppData\Local  
Resultado del gauss (fallido): 2.0741
```

Resultado:

Exacto: 2.28571, Aproximado: 2.07407

El método falló porque el polinomio es de grado 6, y el método de 2 puntos no lo puede integrar con precisión.

## Conclusión

Al aplicar y al comparar los distintos métodos numéricos aplicados a la resolución de derivadas e integrales, se evidencian diferencias importantes en cuanto a precisión, rapidez de convergencia y eficiencia computacional. En el caso de la derivación, las fórmulas de cinco puntos presentan una mayor precisión que las de tres puntos, ya que consideran más información alrededor del punto de interés, reduciendo el error de truncamiento y mejorando la aproximación. Para la integración, el método del trapecio es el más simple, pero también el que muestra menor precisión si se compara con los métodos de Simpson, que al aproximar la función mediante polinomios cuadráticos o cúbicos (en el caso de Simpson  $1/3$  y  $3/8$ , respectivamente) mejoran la exactitud con relativamente poco esfuerzo adicional. No obstante, la cuadratura gaussiana se destaca como el método con mayor velocidad de convergencia y menor error relativo, ya que no requiere dividir el intervalo en partes iguales, sino que evalúa la función en puntos óptimos (raíces de polinomios ortogonales) con pesos que maximizan la exactitud del resultado. En las aplicaciones prácticas realizadas con funciones polinómicas y suaves, se comprobó que la cuadratura gaussiana alcanzó el valor exacto o muy cercano con solo dos puntos, mientras que métodos como Simpson o el trapecio requerían dividir el intervalo en más segmentos para obtener errores similares. Esto confirma que, si bien todos los métodos cumplen con su propósito bajo ciertas condiciones, la elección adecuada depende del tipo de función, del grado de exactitud deseado y de la disponibilidad de recursos computacionales, siendo la cuadratura gaussiana la opción preferida cuando se busca alta precisión con pocos cálculos.

## Referencias

Chapra, S. C., & Canale, R. P. (2015). *Métodos numéricos para ingenieros* (7.<sup>a</sup> ed.). McGraw-Hill Education.

Burden, R. L., & Faires, J. D. (2011). *Numerical Analysis* (9th ed.). Brooks/Cole Cengage Learning.

Atkinson, K. E. (1989). *An Introduction to Numerical Analysis* (2nd ed.). John Wiley & Sons.

## Anexos

Para los candidatos:
Método de tres puntos Método de cinco puntos Método de trapecio Regla de simpson Método de cuadratura gaussiana
Y las tareas:
Diego Alonso Oscar Aaron Brandon Prisila Luis Manuel
Hemos realizado la siguiente asignación:
<b>Regla de simpson:</b> Prisila
<b>Método de tres puntos:</b> Luis Manuel
<b>Método de cinco puntos:</b> Diego Alonso
<b>Método de trapecio:</b> Brandon
<b>Método de cuadratura gaussiana:</b> Oscar Aaron

Asignación de temas a cada integrante del equipo

### Gráfico Gantt

Actividades	Sábado 26/04	Domingo o 27/04	Lunes 28/04	Martes 29/04	Miércoles 30/04	Jueves 01/05
Método Tres Puntos					Luis Manuel	Luis Manuel
Método de Cinco Puntos	Diego Alonso	Diego Alonso				
Método de Trapecio		Brandon	Brandon			
Regla de Simpson					Priscila	Priscila
Método de Cuadratura Gaussiana			Oscar	Oscar		