



EDUCACIÓN

SECRETARÍA DE EDUCACIÓN PÚBLICA



INSTITUTO TECNOLÓGICO DE COMITAN

Alumnos:

- Ruedas Velasco Pedro Eduardo__19700073.
- Hernández Méndez Levi Magdiel__19700039.
- Molina Cifuentes Adriel David__19700061.
- Panti Ordoñez Sergio Ismael__19700065.

Docente: Vera Guillen José Flavio.

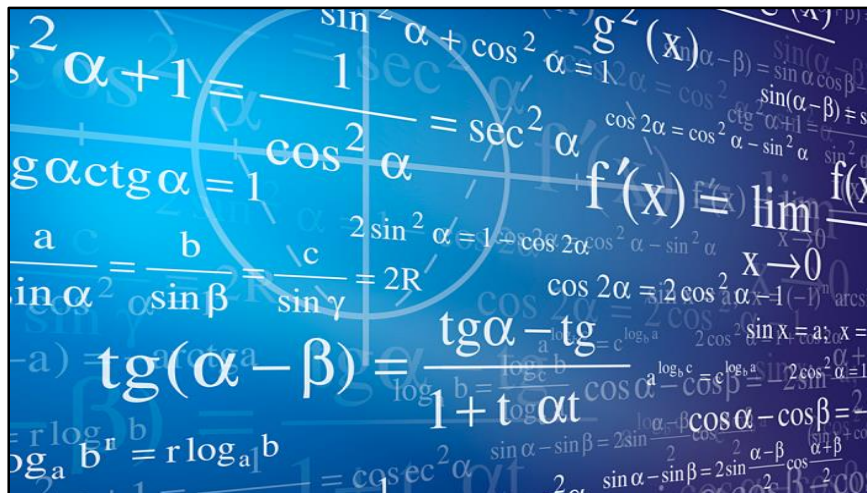
Materia: Métodos Numéricos.

Semestre: Cuarto

Grupo: "A"

Actividad: PA8_Codigo en java- Regla del trapecio y Simpson.

Comitán de Domínguez Chiapas, a 09 de Mayo del 2021.



“Regla del trapecio en Java”.

La regla del trapecio es uno de los métodos más utilizados para calcular aproximaciones numéricas de integrales definidas. Es la primera de las fórmulas cerradas de integración de Newton – Cotes, para el caso cuando el polinomio interpolante es de grado uno.

```
12  *
13  * @author Eduardo Velasco
14  */
15  public class Metodo_Trapecio{
16      float a,b,n,funcion,h,dimen,A=0;
17      float fi,fa,fb,suma=0,resultado,multi;
18      int v;
19      public void trapecio(){
20          System.out.println("Regla del Trapecio");
21          System.out.println("Funcion f(x)= 70/(1+x^2)dx");
22          DecimalFormat dec=new DecimalFormat("#.000");
23          //Se ingresa el total de iteraciones
24          n=Float.parseFloat(JOptionPane.showInputDialog("Digite el total de Iteraciones [n]"));
25          //Introduccion de los limites de la integral
26          b=Float.parseFloat(JOptionPane.showInputDialog("Digite el limite Superior [b]"));
27          a=Float.parseFloat(JOptionPane.showInputDialog("Digite el Limite Inferior [a]"));
28
29          //Calcular el valor de h
30          h=(b-a)/n;
31          A=a;
32          System.out.println("El valor de h es: "+h);
```

```
34      // se calcula el valor de X donde v tomara el "v" tomara el valor de "n" en el arreglo
35      v= (int) n;
36      v=v+1;
37
38      float dimension[]=new float[v];
39      //se le suma "h" al limite inferior "a", desde x1 hasta x(n-1)
40      for (int i = 1; i < (v-1); i++) {
41          a=a+h;
42          dimension[i]=a;
43      }
44      //Se le asigan valores a X0 y Xn
45      dimension[0]=a;
46      dimension[v-1]=b;
47      System.out.println("\n");
48      for (int i = 0; i < v; i++) {
49          System.out.println("El valor de x" +i+ " = " + dec.format(dimension[i]));
50      }
51      //Integral
52      for (int i = 0; i < v; i++) {
53          fi=70/(1+(dimension[i]*dimension[i]));
54          dimension[i]=fi;
55      }
```

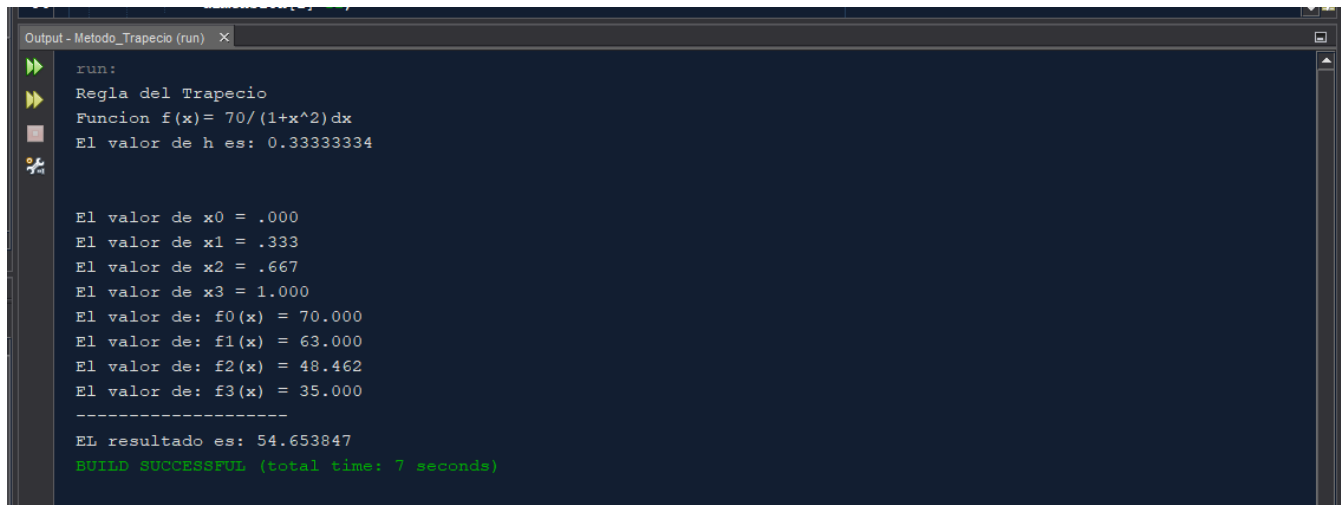
```
55 }
56 fa=dimension[0];
57 fb=dimension[v-1];
58 // Se recorren todos los valores de f(x)
59 for (int i = 0; i < v; i++) {
60     System.out.println("El valor de: f"+i +"(x) = "+dec.format(dimension[i]));
61 }
62 //multiplicaos por 2 desde x1 x(n-1)
63 for (int i = 1; i < (v-1); i++) {
64     multi=dimension[i]*2;
65
66     // se suman los resultados
67     suma=suma+multi;
68 }
69 System.out.println("-----");
70 //Muestra los resultados
71 resultado=((h/2)* (suma+fa+fb));
72 System.out.println("EL resultado es: "+resultado);
73 }
74 public static void main(String[] args) {
75     Metodo_Trapecio trapecio=new Metodo_Trapecio();
76     trapecio.trapecio();
77 }
```

Ejecución del programa.

No. Iteraciones: 3

Límite superior: 1

Límite Inferior: 0



```
Output - Metodo_Trapecio (run) X
run:
Regla del Trapecio
Funcion f(x)= 70/(1+x^2)dx
El valor de h es: 0.33333334

El valor de x0 = .000
El valor de x1 = .333
El valor de x2 = .667
El valor de x3 = 1.000
El valor de: f0(x) = 70.000
El valor de: f1(x) = 63.000
El valor de: f2(x) = 48.462
El valor de: f3(x) = 35.000
-----
EL resultado es: 54.653847
BUILD SUCCESSFUL (total time: 7 seconds)
```

“Regla de Simpson en Java”.

El Método de Simpson sustituye a la curva $y=f(x)$ por una serie de arcos contiguos, cada uno de estos arcos es un arco de parábola de eje vertical. Esto nos lleva a aproximar el área bajo la curva mediante la suma de las áreas bajo cada arco de parábola y es un método de integración numérica que se utiliza para obtener la aproximación de la integral.

```
Simpson.java x
Source History
1 package javaapplication23;
2 import java.util.Scanner;
3
4 /**
5  *
6  * @author Levi Magdiel
7  */
8 public class Simpson {
9     public static Scanner leer = new Scanner(System.in);
10
11     //1.- Definimos una variable de tipo real
12     public double f(double x){
13         return (x*x);
14     }
15
16     //2.- Definimos la regla de Simpson
17     public double Simpson(double a, double b){
18         return ((a-b)/6)*(f(a)+4*f((a+b)/2)+f(b));
19     }
20 }
```

```
Simpson.java x
Source History
19 }
20
21 public static void main(String args[]) {
22     Simpson s = new Simpson();
23     int a,b;
24     System.out.println("limites a: ");
25     a=leer.nextInt();
26     System.out.println("limites de b: ");
27     b=leer.nextInt();
28     System.out.println("El area es: " + s.Simpson(a, b));
29 }
30
31 }
32 }
```



- Ejecución

JavaApplication23 - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

The screenshot shows the NetBeans IDE interface. The main editor window displays the source code of a Java file named `Simpson.java`. The code implements a Simpson's rule integration function. The `main` method is also visible. The `Output` window at the bottom shows the execution results.

```
//2.- Definimos la regla de Simpson
public double Simpson(double a, double b){
    return ((a-b)/6)*(f(a)+4*f((a+b)/2)+f(b));
}

public static void main(String args[]) {
    Simpson s = new Simpson();
}
```

run:
limites a:
1
limites de b:
2
El area es: -2.333333333333333
BUILD SUCCESSFUL (total time: 16 seconds)