EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE COMITAN

Alumnos:

- Ruedas Velasco Pedro Eduardo__19700073.
- Hernández Méndez Levi Magdiel__19700039.
- Molina Cifuentes Adriel David__19700061.

Docente: Guillen Vera José Flavio.

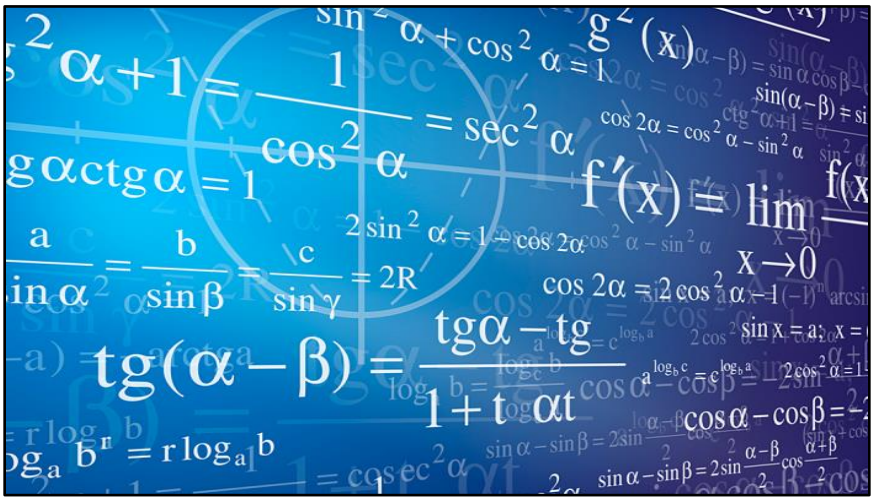
Materia: Métodos Numéricos.

Semestre: Cuarto

Grupo: “A”

Actividad: PA12. Códigos de los programas en el lenguaje seleccionado.

Comitán de Domínguez Chiapas, a 27 de Junio del 2021.



Método de runge kutta en Python.

El código en Python consiste en resolver ecuaciones diferenciales ordinarias no lineales mediante el método de Renge kutta el cual nos proporciona un pequeño margen de error con respecto a la solución del problema y es fácilmente programable.

Lo primero en realizar es introducir la fórmula del método de Runge Kutta y se sabe que el valor inicial de la ecuación es el siguiente:

$$y' = f(t, y), y(t_0) = y_0$$

y ' = f (t , y) , y (t 0) = y 0

Por lo tanto la fórmula de cálculo iterativo de la ecuación es el siguiente:

$$y(t + \Delta t) = y(t) + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) + O(\Delta t^4)$$

$$k_1 = f(y, t)\Delta t$$

$$k_2 = f(y + \frac{1}{2}k_1, t + \frac{1}{2}\Delta t)\Delta t$$

$$k_3 = f(y + \frac{1}{2}k_2, t + \frac{1}{2}\Delta t)\Delta t$$

$$k_4 = f(y + k_3, t + \Delta t)\Delta t$$

y (t + Δ t) = y (t) + 1 6 (k 1 + 2 k 2 + 2 k 3 + k 4) + O (Δ t 4) k 1 = f (y , t) Δ t k 2 = f (y + 1 2 k 1 , t + 1 2 Δ t) Δ t k 3 = f (y + 1 2 k 2 , t + 1 2 Δ t) Δ t k 4 = f (y + k 3 , t + Δ t) Δ t

“Código en Python”:

Interpolacion.py | Regresion Lineal simple.py | Metodo de RungeKutta.py

Reader Mode

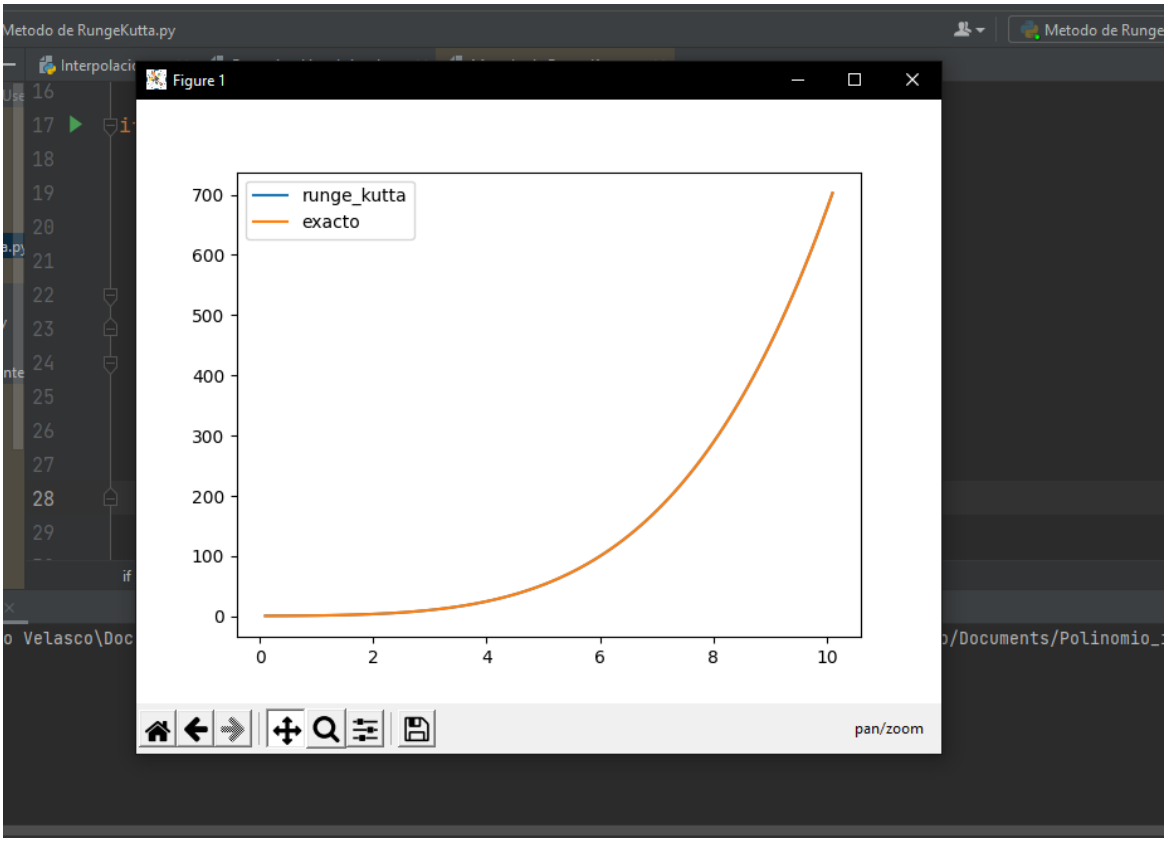
```python
import math
import numpy as np
import matplotlib.pyplot as plt

def runge_kutta(y, x, dx, f):
    """ y es el valor incial de y
        x es el valor inicial de x
        dx es el paso de tiempo en x
        f la derivada de la funcion y(t)
    """
    k1 = dx * f(y, t)
    k2 = dx * f(y + 0.5 * k1, x + 0.5 * dx)
    k3 = dx * f(y + 0.5 * k2, x + 0.5 * dx)
    k4 = dx * f(y + k3, x + dx)
    return y + (k1 + 2 * k2 + 2 * k3 + k4) / 6.

```

Interpolacion.py | Regresion Lineal simple.py | Metodo de RungeKutta.py

Reader Mode

```python
if __name__=='__main__':
    t = 0.
    y = 1.
    dt = .1
    ys, ts = [], []
    def func(y, t):
        return t * math.sqrt(y)
    while t <= 10:
        y = runge_kutta(y, t, dt, func)
        t += dt
        ys.append(y)
        ts.append(t)

    exact = [(t ** 2 + 4) ** 2 / 16. for t in ts]
    plt.plot(ts, ys, label='runge_kutta')
    plt.plot(ts, exact, label='exacto')
    plt.legend()
    plt.show()
    error = np.array(exact) - np.array(ys)
    print("max error {:.5f}".format(max(error)))
```

“Ejecución del Código”:



Método de Taylor:

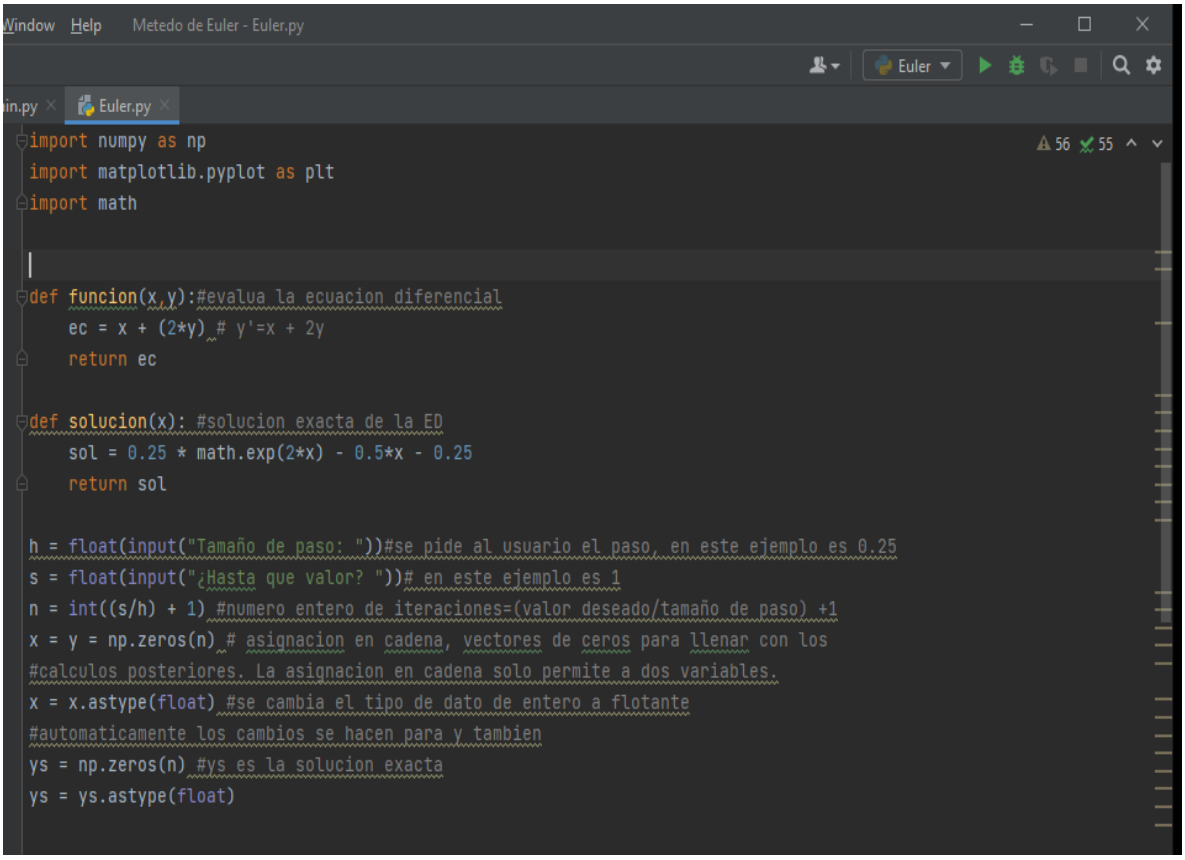
En este programa se resolverá la serie de Taylor, de una forma muy sencilla basándonos en su fórmula simplificada, el programa atraes de la consola de java pide los datos necesarios los cuales son la cantidad de términos y el valor de 'X' gracias a un método que factoriza nos hace más fácil el cálculo de este método y atraes de un cálculo llamamos el método de factorización y lo dividimos entre el resultado del valor de x a x potencia gracias al método Math.pow.

Source History

```java
/*Metodo de Taylor*/
package metodo_serie_taylor;

import java.util.Scanner;

/**
 *
 * @author LeviH KC
 */
public class Metodo_Serie_Taylor {

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

        //Pedir datos de entrada
        System.out.print("Cuantos terminos desea: ");
        int n = leer.nextInt();
        System.out.print("Ingrese el valor de X: ");
        int x = leer.nextInt();

        //Calculo
        double s = 0, t;
        for (int i = 0; i < n; i++) {
            t = Math.pow(x, i) / factorial(i);
            s += t;
        }

        //Mostrar datos
        System.out.printf("f(%d)=%f\n", x, s);
    }

    //Metodo para sacar el factorial
    public static double factorial(int n) {
        double aux = 1;
        for (int i = 2; i <= n; i++) {
            aux *= i;
        }
        return aux;
    }
}
```

Método de Euler en Python.

El código en Python consiste en resolver ecuaciones diferenciales ordinarias de primer orden mediante el método de Euler el cual nos proporciona un pequeño margen de error con respecto a la solución del problema y es fácilmente programable.



```
24
25  #valor inicial x[0] = 0, y[0] = 0
26  for i in np.arange(1,n):
27      #np.arange(1,n) permite a i moverse con h sin importar que es decimal de 1 hasta n
28      y[i] = y[i-1] + h*(funcion(x[i-1],y[i-1])) #calculo de y
29      x[i] = x[i-1] + h #incremento en x
30      ys[i] = solucion(x[i-1])
31  print(x) #impresion de vectores
32  print(y)
33  print(ys)
34  plt.scatter(x,y) #realizacion de grafica de puntos
35  plt.scatter(x,ys, color = 'red') #grafica de solucion exacta que se empalma con la anterior
36  plt.plot(x,y) #realizacion de grafica de lineas (se empalma)
37  plt.plot(x,ys, color = 'red') #se empalma
38  plt.show() #se muestra grafica de las dos soluciones
```

Typo: In word 'cadena'

Replace with 'cadenza' Alt+Mayús+Intro More actions... Alt+Intro