División Académica de Tecnologías de la Información y Diseño

UNIVERSIDAD TECNOLÓGICA EMILIANO ZAPATA DEL ESTADO DE MORELOS

Reporte tarea integradora: Software especializado para la resolución de ecuaciones diferenciales

Integrantes:

- Christian Emmanuel Medina Vergara
- Marco Antonio Vazquez Rayo
- Samuel Gonzalez Garcia

Índice

Índice de Ilustraciones

Introducción

Este programa fue realizado con el fin de ofrecer una herramienta para los usuarios en la solución de ecuaciones diferenciales básicas, donde además de los pasos realizados también se muestra una grafica con los resultados obtenidos, para ello es necesario conocer los valores iniciales, el tamaño del paso (h) y el número de iteraciones.

Interfaz de usuario

El programa cuenta con una interfaz gráfica simple y fácil de usar, el usuario puede ingresar la ecuación diferencial, los valores iniciales de **x** y **y,** el tamaño del paso (h) y el número de iteraciones, además, hay un botón "**Solve**" que inicia el proceso de resolución.

Los valores iniciales se refieren a los valores asignados a las variables independientes y dependientes en el punto de partida de la resolución de la ecuación diferencial.

El tamaño de los pasos (h) se refiere al tamaño del incremento utilizado para avanzar en la variable independiente (x) en cada iteración del método de Euler, en nuestro programa el tamaño del paso se ingresa como un valor numérico.

El número de iteraciones (n) se refiere a la cantidad de veces que se repetirá el proceso de cálculo en el método de Euler, cada iteración implica calcular un nuevo valor de la variable dependiente (y) utilizando el valor actual de la variable independiente (x) y el maño del paso (h), en el programa, el número de iteraciones se ingresa como un valor entero, cuantas más iteraciones se realicen, más puntos se generarán en el gráfico de la ecuación diferencial, lo que proporciona una representación más detallada del comportamiento de la función.

Ilustración 1 Interfaz

Método Euler

El método de Euler es un método numérico que se utiliza para aproximar la solución de una ecuación diferencial ordinaria de primer orden. Esta ecuación se puede escribir en la forma general: $\frac{dy}{dx} = f(x, y)$, donde **y** es la función desconocida y $\mathbf{f(x, y)}$ es una función que define la relación entre las variables **x** e **y**.

El método de Euler se basa en la idea de aproximar la solución de la ecuación diferencial mediante los siguientes pasos:

1. **Definir los valores iniciales:**
   Primero el usuario deberá ingresar los valores iniciales de x(x0) y y(y0), estos valores representan el punto de partida de la solución de la ecuación diferencial.

2. **Definir el tamaño del paso:**
   El tamaño del paso (h) se refiere al tamaño del incremento utilizado para avanzar en la variable independiente **x** en cada iteración del método, un tamaño de paso con un valor bajo proporcionará una mayor precisión en la solución, pero también requerirá más iteraciones y tiempo de cálculo.

3. **Calcular los valores de y para para cada paso:**
   El algoritmo del método de Euler se basa en una aproximación lineal de la solución de la ecuación diferencial, en cada uno de los pasos se utiliza la siguiente formular para calcular el nuevo valor de y:

   $$y[1 + i] + h * f(x[i], y[i])$$

   Donde y[i] es el valor acutal de y, x[i] es el valor acutal de x, y f(x[i], y[i] es el valor de la función f en el punto (x[i], y[i]).

4. **Avanzar en la variable independiente:**
   Después de calcular el numero valor de y, se avanza en la variable independiente x sumando el tamaño del paso: x[i+1] = x[i] + h.

5. **Repetir los pasos 3 y 4 hasta alcanzar el número de iteraciones deseado:**
   El proceso de cálculo se repita el número de veces especificado por el usuario **(n)** o hasta que se alcance un punto de parada definido.

6. **Obtener la solución aproximada:**
   Al finalizar las iteraciones, se obtiene una serie de valores de x e y que aproximadamente resuelven la ecuación diferencial.

   El Método de Euler utiliza una aproximación lineal para resolver ecuaciones diferenciales de primero orden, en cuando se avanzan en incrementos pequeños de la variable independiente x, se calcula el nuevo valor de la variable dependiente y utilizando la relación dada para la ecuación diferencial.

Generación del gráfico

Una vez que se resuelve la ecuación diferencial, el programa genera un gráfico de los valores de **x** y **y**, utiliza la biblioteca **matplotlib** para trazar los puntos en un plano cartesiano donde se muestran los ejes **x** e **y**, además un título descriptivo al gráfico.

Matplotlib es una biblioteca de visualización en Python que nos permite crear graficas con diferentes herramientas para su manipulación, en este caso utilizamos matplotlib para trazar los puntos que representan las soluciones de la ecuación diferencial en el plano cartesiano.

1. Se importa la biblioteca:

```
5   import matplotlib.pyplot as plt
6
```

Ilustración 2 Importación de la biblioteca

2. **Crear una figura y en un eje:**
   Antes de trazar los puntos, se necesita crear una figura y un eje para el gráfico, esto se hace utilizando la función **plt.subpltis()** de matplotlib, la función devuelve una figura y un objeto de eje que nos permiten configurar y personalizar el grafico.

```
plt.plot(x_vals, y_vals)
plt.xlabel('x')
plt.ylabel('y')
plt.title('Gráfica de la Ecuación Diferencial')
plt.grid(True)
plt.show()
```

Ilustración 3 Creación de la figura y manipulación

3. **Trazar los puntos:**
   Una vez que tenemos el objeto de eje, podemos utilizar sus métodos para trazar los punto en el plano cartesiano, en nuestro caso utilizamos el método **plt.plot ()** para trazar los puntos donde le pasamos dos listas de valores: Una para las coordenadas **X** y otra para las coordenadas **Y**.

```
x_vals, y_vals, steps = euler_method(f, x0, y0, h, n)

plt.plot(x_vals, y_vals)
```

Ilustración 4 Trazar los puntos con los valores de X y Y

4. **Mostrar el gráfico:**
   Ahora para mostrar la grafica utilizamos la función **plt.show()** para mostrar el grafico en una ventana separada, esto abrirá una ventana emergente que muestra el grafico de los puntos trazados en el plano cartesiano.

```
plt.show()
```

Informe de pasos

Después de generar el gráfico, el programa muestra una ventana emergente que presenta los pasos utilizados para resolver la ecuación diferencial cada paso se muestra en el formato de texto, indicando el número de iteración, el valor de x y el valor de y correspondiente.

Con la línea de código **for step in steps: steps_text.insert(tk.END, f"{step} \n")**, se realiza un bucle a través de una lista de pasos (steps) y los inserta en el widget de texto, cada paso se agrega en una nueva línea utilizando **steps_text.insert()**.

```
for step in steps:
    steps_text.insert(tk.END, f"{step}\n")
```

Ilustración 5 Uso de la función steps para insertar los valores del arreglo creado.

Código utilizado

main.py > ...

```python
import numpy as np
import tkinter as tk
from tkinter import ttk
from sympy import lambdify, symbols
import matplotlib.pyplot as plt

def euler_method(f, x0, y0, h, n):
    x = np.zeros(n+1)
    y = np.zeros(n+1)
    x[0] = x0
    y[0] = y0

    steps = [] # Lista para almacenar los pasos utilizados para resolver la ecuación

    for i in range(n):
        y[i+1] = y[i] + h * f(x[i], y[i])
        x[i+1] = x[i] + h

        step = f"Step {i+1}: x = {x[i]:.2f}, y = {y[i]:.4f}"
        steps.append(step)

    return x, y, steps
```

```python
def plot_graph():
    equation = equation_entry.get() # Obtener la ecuación ingresada por el usuario
    x0 = float(x0_entry.get()) # Obtener el valor inicial de x ingresado por el usuario
    y0 = float(y0_entry.get()) # Obtener el valor inicial de y ingresado por el usuario
    h = float(h_entry.get()) # Obtener el tamaño del paso ingresado por el usuario
    n = int(n_entry.get())  # Obtener el número de iteraciones ingresado por el usuario

    try:
        x, y = symbols('x y')
        f = lambdify((x, y), equation) # Convertir la ecuación en una función lambda

        x_vals, y_vals, steps = euler_method(f, x0, y0, h, n)

        plt.plot(x_vals, y_vals)
        plt.xlabel('x')
        plt.ylabel('y')
        plt.title('Gráfica de la Ecuación Diferencial')
        plt.grid(True)
        plt.show()
```

```python
        # Crear una nueva ventana para mostrar los pasos utilizados
        steps_window = tk.Toplevel(root)
        steps_window.title("Pasos para Resolver la Ecuación Diferencial")

        steps_text = tk.Text(steps_window, height=10, width=40, bd=0, font=('Arial', 12))
        steps_text.pack(pady=20)
        steps_text.insert(tk.END, "Pasos utilizados para resolver la ecuación:\n")
        for step in steps:
            steps_text.insert(tk.END, f"{step}\n")

    except:
        error_label.config(text="¡Ecuación inválida! Inténtalo de nuevo.")

# Crear la ventana principal
root = tk.Tk()
root.title("Ingresar Ecuación Diferencial")

# Estilo para los elementos de la interfaz
style = ttk.Style()
style.configure('TLabel', font=('Arial', 12))
style.configure('TButton', font=('Arial', 12))
```

```
# Etiqueta y entrada de texto para ingresar la ecuación
equation_label = ttk.Label(root, text="Ecuación Diferencial:")
equation_label.pack(pady=10)

equation_entry = ttk.Entry(root, font=('Arial', 12))
equation_entry.pack()

# Etiquetas y entradas de texto para los valores iniciales, tamaño del paso y número de iteraciones
x0_label = ttk.Label(root, text="Valor Inicial de x:")
x0_label.pack(pady=5)

x0_entry = ttk.Entry(root, font=('Arial', 12))
x0_entry.pack()

y0_label = ttk.Label(root, text="Valor Inicial de y:")
y0_label.pack(pady=5)

y0_entry = ttk.Entry(root, font=('Arial', 12))
y0_entry.pack()

h_label = ttk.Label(root, text="Tamaño del Paso:")
h_label.pack(pady=5)
```

```
h_entry = ttk.Entry(root, font=('Arial', 12))
h_entry.pack()

n_label = ttk.Label(root, text="Número de Iteraciones:")
n_label.pack(pady=5)

n_entry = ttk.Entry(root, font=('Arial', 12))
n_entry.pack()

# Botón para resolver la ecuación
solve_button = ttk.Button(root, text="Resolver", command=plot_graph)
solve_button.pack(pady=10)

# Etiqueta para mostrar errores
error_label = ttk.Label(root, text="", foreground='red')
error_label.pack(pady=10)

# Mostrar la ventana principal
root.mainloop()
```

Resultados

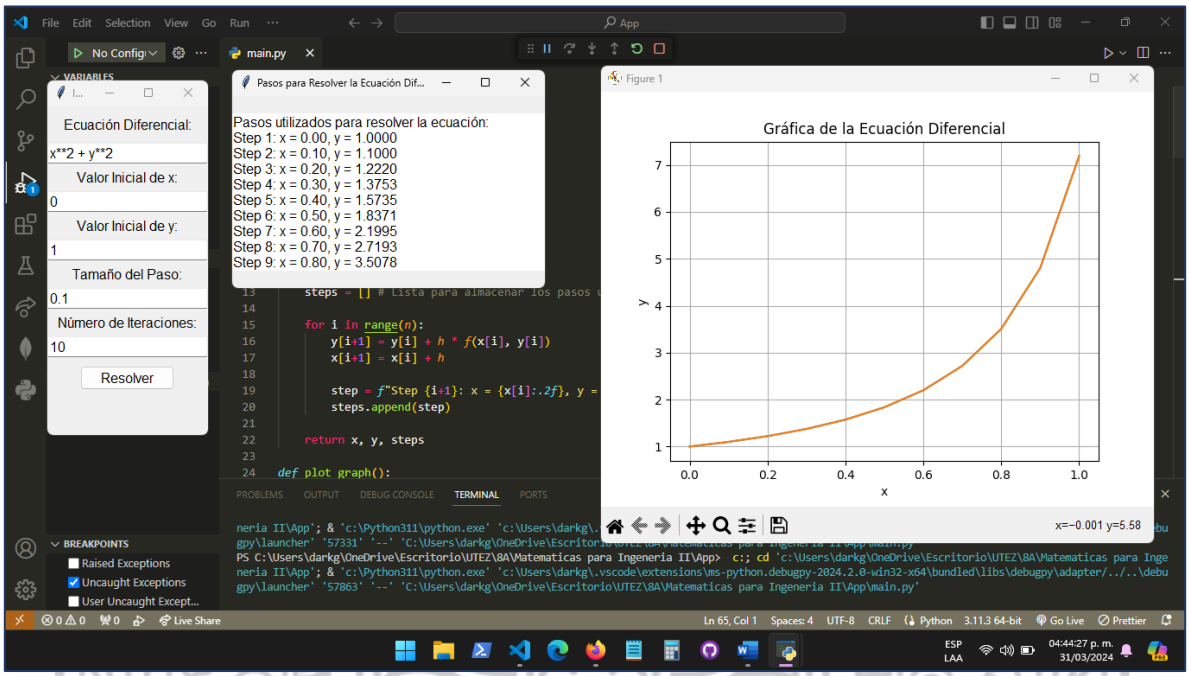

Ilustración 6 Resultado de la solución de la ecuación diferencial.

Bibliografías

- Solución numérica de sistemas de ecuaciones diferenciales ordinarias con condiciones iniciales y de ecuaciones diferenciales ordinarias de orden superior, por Jesús Javier C. R, Miguel Eduardo G. C, Víctor Damián P. M, (2019), recuperado el (31/03/2024) de https://www.ingenieria.unam.mx/pinilla/PE105117/pdfs/tema5/5-123_soluciones_edo.pdf
- Chapra, S., y Canale, R. (2015). Métodos numéricos para ingenieros (M. Hill, Ed, (2016), recuperado el (29/03/2024) de http://artemisa.unicauca.edu.co/~cardila/Chapra.pdf
- Métodos numéricos para las ecuaciones diferenciales, Jose S. Cánovas Peña, (18 de diciembre de 2009) recuperado el (29/03/2024) de https://www.dmae.upct.es/~jose/metodos/numer4.pdf

GitHub

- https://github.com/ChristianEMV/Project_CalculatorD.E.git