Simulador metodo de Euler para ecuaciones diferenciales

Juan David Palacios Chávez
Ecuaciones diferenciales, grupo 1. 15 de marzo de 2023.

I. Resumen

En el presente documento se busca introducir sobre el método de resolución numérica Euler para ecuaciones diferenciales de primer orden y mostrar su efectividad o margen de error presente por medio de un programa hecho en Python que pueda comparar la gráfica resultante de este método y la gráfica resultante de la función.

II. Marco teórico

- Método de Euler: El método de Euler es un procedimiento utilizado para hallar una resolución numérica a ecuaciones diferenciales ordinarias de primer orden, se considera el método más simple de resolución numérica y nos permite hallar respuesta a ecuaciones con problema de valor inicial, el método de Euler presenta un error correlacionado al tamaño del paso, llamado también delta de x.
- Python: Python es un lenguaje de programación de alto nivel creado por Guido Van Rossum en el año 1991, es un lenguaje multipropósito que se utiliza actualmente principalmente por su potencial en el área de análisis de datos y desarrollo web.
- SymPy: SymPy es una librería de Python que se utiliza para matemáticas simbólicas, la cual nos permite agregar expresiones matemáticas y nos da herramientas para trabajar con las mismas, entre las cuales se encuentran simplificar, resolver integrales, derivadas y además nos permite resolver ecuaciones diferenciales.
- Matplotlib: Matplotlib es una librería para crear visualizaciones estáticas, animadas e interactivas que nos permite visualizar los datos de una mejor forma, permitiéndonos así crear gráficas.

III. Propuesta

Se crea un código en el lenguaje de programación Python para poder visualizar la solución numérica y exacta de una ecuación de la forma y'(x) = f(x, y) en el que el usuario puede escoger la función f(x,y) a evaluar, con esta se muestra la solución general y específica de la función para los puntos (x0, y0) elegidos por el usuario con la librería SymPy y se grafica la solución obtenida por el método de Euler y evaluada por la solución particular del problema con la librería Matplotlib. Para saber más sobre el código utilizado y sus instrucciones de utilización visitar https://github.com/JuanDavidPalaciosCh/SimuladorMetodoEuler.

IV. Funcionamiento

El código fue hecho enteramente en Python, un lenguaje de programación que nos permite con ayuda de librerías externas la creación de gráficas y de cálculos simbólicos que permitieron la elaboración del simulador, el simulador cuenta con 2 partes:

- Clase DiferentialFunction que permite la creación de objetos de la forma y'(x) = f(x, y) y permiten a través de sus métodos la resolución de las ecuaciones, una forma de visualización adaptada al terminal y la graficación que permite comparar el método de Euler con la solución exacta.
- Archivo main.py en el cual se aloja el bucle principal del programa, encargado del funcionamiento y realizar preguntas al usuario para su posterior utilización, como la función f(x, y) y el parámetro delta x.

Para su utilización puedes clonar el código desde el repositorio de GitHub https://github.com/JuanDavidPalaciosCh/SimuladorMetodoEuler (Para más información sobre clonar repositorios visita esta página) y dentro de la carpeta del proyecto abre un terminal, dentro de este escribe:

```
C:\Users\Usuario\CarpetaProyecto> py main.py
```

Con lo cual ya tendrás el simulador funcionando, seguido de esto, el usuario pondrá los parámetros por consola, los cuales son:

- Función f: Una función f(x,y) para la ecuación y'(x) = f(x, y) por ejemplo y(x) * sin(x), la cual evaluará la ecuación y'= y(x) * sin(x). Cuando se utilice la variable 'y' se debe aclarar que esta depende de x colocando 'y(x)'. La función además puede depender tanto solo de x como y(x), por ejemplo 9*cos(10*x) + 2.
- Valor x0: Valor inicial para la función en x.
- Valor y0: Valor inicial para la función en y, que con el valor anterior formarán una condición general de la forma (y(x0), y0).
- Valor delta_x: Valor numérico para su utilización en el método de Euler como salto entre valores, por ejemplo 0,1 (El separador decimal se debe escribir con un '.' es decir 0.1).
- Valor n_max: Este valor determinará hasta donde se graficará la función tanto por el método de Euler

como su función exacta, es decir, si el valor de x0 es 0 y se escoge un valor de 20, la función se graficará hasta x=20, si el valor de x0 es negativo la función se evaluará hasta x0 + n_max, por ejemplo si el valor escogido para x0 es -20 la gráfica irá desde -20 a 0.

Después de ingresar los parámetros el simulador realizará una respuesta con los siguientes componentes:

- Ecuacion diferencial: El programa mostrará la ecuación diferencial escogida por el usuario de la forma y'(x) = f(x, y).
- Solución general: Se mostrará la solución general de la función.
- Solución especifica: Se mostrará la solución específica de la función.
- Gráfica de solución exacta por método de Euler: Se abrirá una pestaña de Matplotlib en la cual se mostrará la gráfica de la solución de la ecuación por el método de Euler de forma punteada en color azul y con la solución exacta en color naranja.

Por lo cual si el usuario escoge los siguientes parámetros:

```
Ingrese la función: y'= y(x) * sin(x)
Ingrese el valor inicial de x: 0
Ingrese el valor inicial de y: 1
Ingrese el valor de x: 0.01
Ingrese el valor de x en la que se
desea graficar: 20
```

Se dará la siguiente respuesta:

```
Ecuación diferencial:
d
--(y(x)) = y(x)sin(x)
dx
Solución general:
          -cos(x)
y(x) = Ce
Solución específica:
                          -cos(x)
y(x) = 2.71828182845905e
```

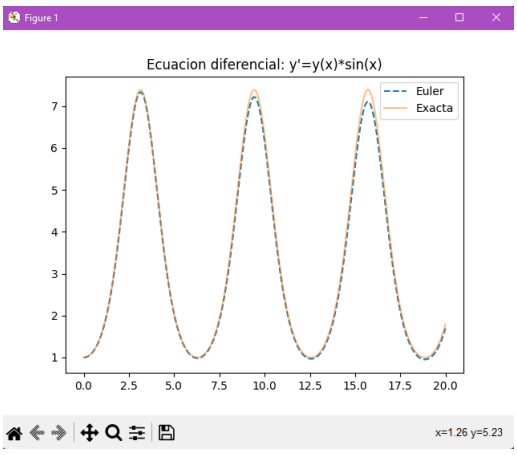

Figura 1: Respuesta de ejemplo para la ecuación y'= y(x) * sin(x)

Para que el código genere esa respuesta se utilizó principalmente la librería SymPy, con la cual para facilitar el código y hacerlo más escalable se creó la clase DiferentialFunction la cual nos permite inicializar un objeto de la siguiente manera:

```
DiferencialFunction(f, x0, y0, delta_x, n_max)
```

En el cual Sympy transforma la entrada por consola del usuario a una función simbólica con la siguiente función:

```
self.f = sympy.parse_expr(f)
```

Siendo self.f la expresión del usuario.

Luego de esto se convierte la función en una ecuación diferencial de la forma y'(x) = f(x, y) de la siguiente forma:

```
sympy.Eq(sympy.diff(y(x)), self.f)
```

Para dar una solución especifica y general a la ecuación con el metodo sympy.dsolve():

```
sympy.dsolve(self.eq, y(x)) # General
sympy.dsolve(self.eq, y(x), ics=ics)# Especifica
```

Nótese que la única diferencia entre la solución general y específica es el parámetro de entrada 'ics' el cual es un diccionario que indica las condiciones iniciales, el cual se le pasa por medio de un diccionario con los valores de (x0,y0) escogidos por el usuario.

Luego el programa se encarga con el método solve_euler de encontrar una lista de valores del eje 'x' y 'y' para graficar la solución de Euler, los valores se encuentran utilizando list comprehesion, una forma de abreviar los bucles for en Python para almacenarlos directamente en una lista.

Para la lista de valores del eje x se utiliza la siguiente fórmula:

$$x_i = x_0 + i * \Delta x$$

Siendo 'i' el número de iteraciones al que llega con un rango dado por la fórmula:

$$n\_max/\Delta x$$

Formulas vistas en el código creando una lista de valores para el eje x de la siguiente forma:

```
x_axis=[self.x0 + (i * self.delta_x) for i in ...
...range(int(self.n_max/self.delta_x))]
```

Para la lista de valores en el eje y se crea primero una lista de valor '0' con el mismo tamaño de la lista para el eje x para posteriormente colocar en su primera posición el valor y0 escogido por el usuario:

```
y_axis = [0 for i in range(len(x_axis))]
```

```
y_axis[0] = self.y0
```

Luego, se itera la lista y_axis con un bucle for para agregar valores con la siguiente ecuación:

$$y_i = y_{i-1} + F(x_{i-1}, y_{i-1}) * \Delta x$$

En el código visto de la siguiente forma:

```
for i in range(len(x_axis) - 1):
    y_axis[i + 1] = (y_axis[i]) +...
    ...function(x_axis[i],...
    ...y_axis[i]) * self.delta_x
```

Ya teniendo el eje 'x' y 'y' para la gráfica evaluada por el método de Euler, se grafican los resultados por medio de la librería Matplotlib y con el método sympy.lambdify() que convierte las funciones simbólicas a una función lambda para el caso de la solución específica:

```
 f_exacta = sympy.lambdify([x], ...
 ...self.solve_specific().rhs, 'numpy')

plt.plot(x_axis, y_axis, "--", label="Euler")

plt.plot(x_axis, [f_exacta(x_axis[i])...
...for i in range(len(x_axis))], ...
...label="Exacta", alpha=0.5)
```

Siendo plt.plot la función de la librería que nos permite graficar los valores en una pestaña como muestra la Figura 1.

V. Conclusiones

Aunque el código puede ser mejorado, nos muestra un funcionamiento importante para comparar la solución de Euler con la solución exacta de la función y como tiene más o menos precisión si variamos el valor de delta x, además nos puede ayudar a encontrar soluciones a ecuaciones diferenciales difíciles de resolver y a encontrar estas soluciones con problemas de valores iniciales rápidamente. Aunque la librería SymPy tiene restricciones y a veces falla a la hora de resolver ecuaciones diferenciales, normalmente tiene un buen rendimiento y calcula las soluciones rápidamente, ayudándonos a hacer un trabajo que nos llevaría bastante tiempo en tan solo unos segundos.