Universidad Alfonso X el Sabio
Martina García González, Lucía Mielgo Torres
Grado en Ingeniería Matemática
AMPLIACIÓN DE LOS MÉTODOS NUMÉRICOS

Taller 8

28 de enero de 2025

1. Introducción

1. Desarrolle un programa que resuelva el siguiente sistema de ecuaciones diferenciales:

$$\frac{dy_1}{dt} = -0{,}5y_1$$

$$\frac{dy_2}{dt} = 4 - 0{,}1y_1 - 0{,}3y_2$$

   Utilizando el método de Euler en el intervalo $t \in [0, 2]$, utilizando tamaños de paso de 1, 0.1 y 0.01. Tome como condiciones iniciales los valores $y_1(0) = 4$ y $y_2(0) = 6$.

2. Realice el mismo problema que el anterior implementando el método de Runge-Kutta de orden 4.

3. Implemente un programa que resuelva el siguiente sistema de ecuaciones diferenciales con un tamaño de paso de 0.001 en el intervalo $[0, 5]$:

$$x' = \frac{1}{2}(45 - x) + \frac{1}{2}(y - x)$$

$$y' = \frac{1}{2}(x - y) + \frac{1}{4}(35 - y) + \frac{1}{4}(z - y) + 20$$

$$z' = \frac{1}{4}(y - z) + \frac{1}{2}(x - z)$$

   Tome como condiciones iniciales $x(0) = 45$, $y(0) = 35$ y $z(0) = 35$.

2. Métodos y modelos matemáticos

Método de Euler

El método de Euler es una técnica numérica para resolver ecuaciones diferenciales ordinarias de la forma:

$$\frac{dy}{dt} = f(t, y) \tag{1}$$

con una condición inicial $y(t_0) = y_0$. El método de Euler utiliza la derivada para aproximar la solución de la ecuación diferencial mediante una secuencia de pasos discretos. A partir de un valor inicial conocido, se calcula el siguiente valor de la función utilizando la fórmula:

$$y_{n+1} = y_n + hf(t_n, y_n) \tag{2}$$

donde:

- h es el tamaño del paso,
- $t_n = t_0 + nh$ es el tiempo en el paso n,
- y_n es la aproximación numérica de la solución en el tiempo t_n.

El método de Euler es explícito, lo que significa que el nuevo valor y_{n+1} se calcula directamente a partir del valor actual y_n y la pendiente $f(t_n, y_n)$. Este método es sencillo de implementar, pero puede ser ineficiente para obtener soluciones precisas cuando el tamaño de paso h es grande, ya que introduce errores numéricos acumulativos. Sin embargo, al reducir el valor de h, se mejora la precisión de la solución a costa de un mayor número de cálculos.

Método de Runge-Kutta de Orden 4

El método de Runge-Kutta de orden 4 (RK4) es un método numérico utilizado para resolver ecuaciones diferenciales ordinarias de la forma $\frac{dy}{dt} = f(t, y)$

Dado un intervalo $[t_0, t_f]$ y un valor inicial $y(t_0) = y_0$, el método RK4 calcula una aproximación de $y(t_f)$ siguiendo estos pasos:

- Definir el tamaño del paso h:

  El intervalo se divide en N pasos, donde h se calcula como:

$$h = \frac{t_f - t_0}{N} \tag{3}$$

- Iterar para cada paso:

  Para cada paso i desde 0 hasta $N - 1$:

  - Calcular el valor actual de t:

$$t_i = t_0 + i \cdot h \tag{4}$$

  - Calcular las pendientes:

$$k_1 = h \cdot f(t_i, y_i) \tag{5}$$

$$k_2 = h \cdot f\left(t_i + \frac{h}{2}, y_i + \frac{k_1}{2}\right) \tag{6}$$

$$k_3 = h \cdot f\left(t_i + \frac{h}{2}, y_i + \frac{k_2}{2}\right) \tag{7}$$

$$k_4 = h \cdot f(t_i + h, y_i + k_3) \tag{8}$$

  - Actualizar el valor de y:

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{9}$$

- Al finalizar la iteración, el valor y_N será la aproximación de la solución $y(t_f)$.

3. Resultados

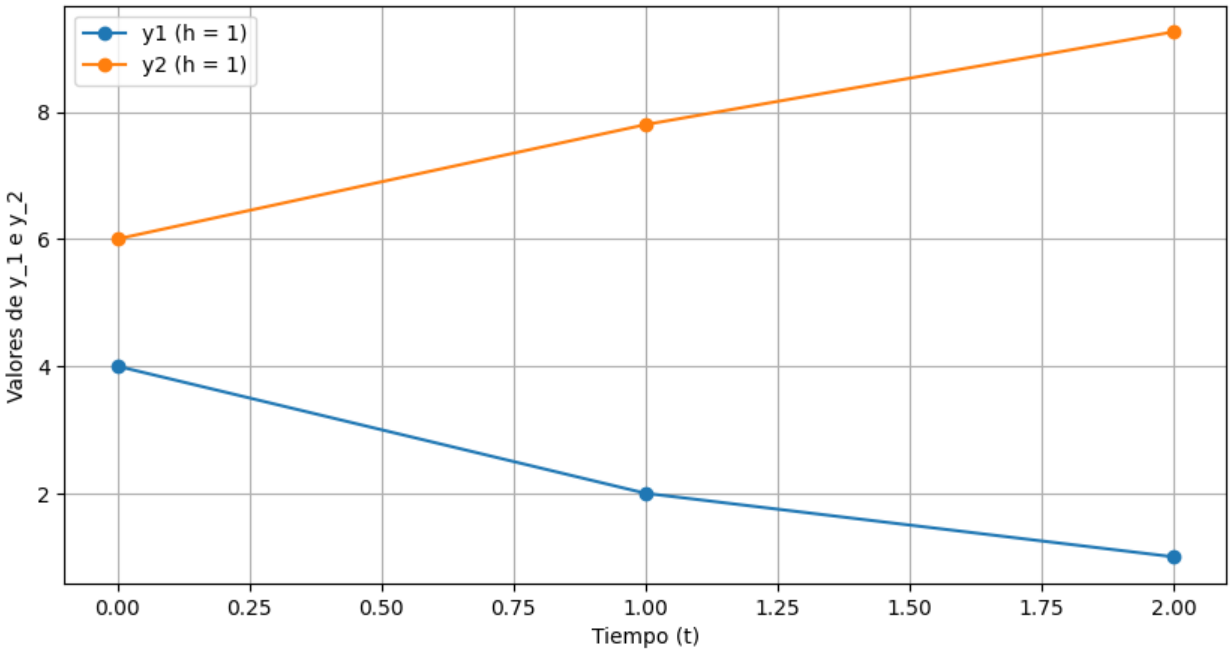

Figura 1: Valores de y_1 e y_2 para h=1

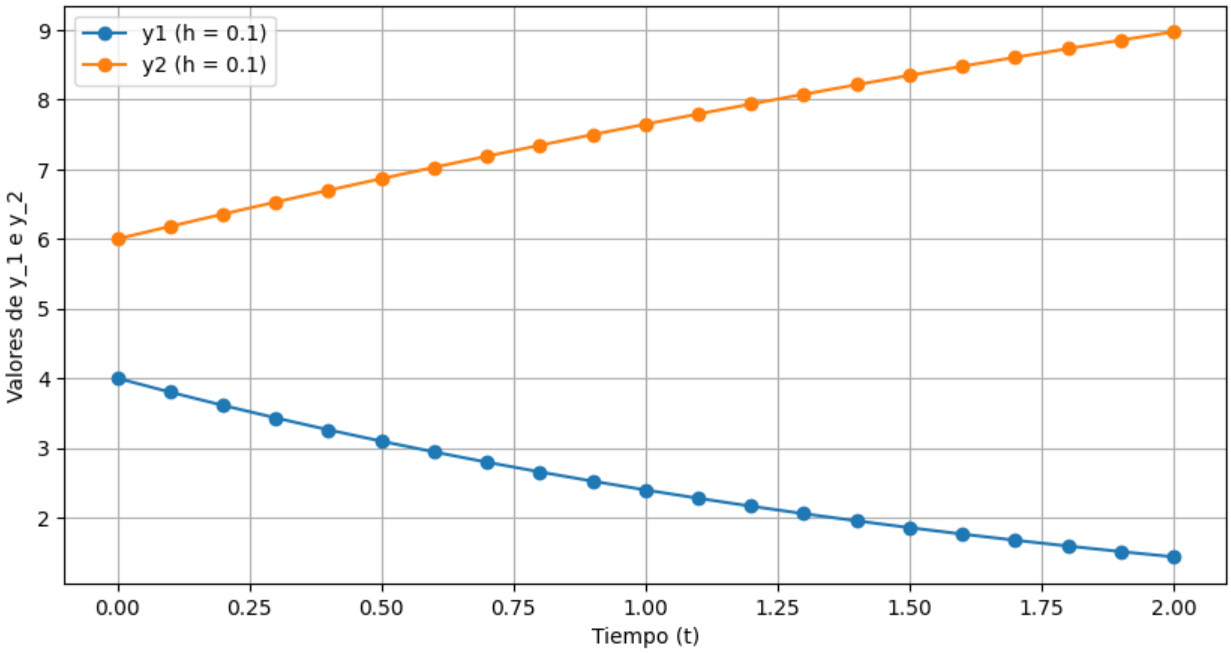

Figura 2: Valores de y_1 e y_2 para h=0.1

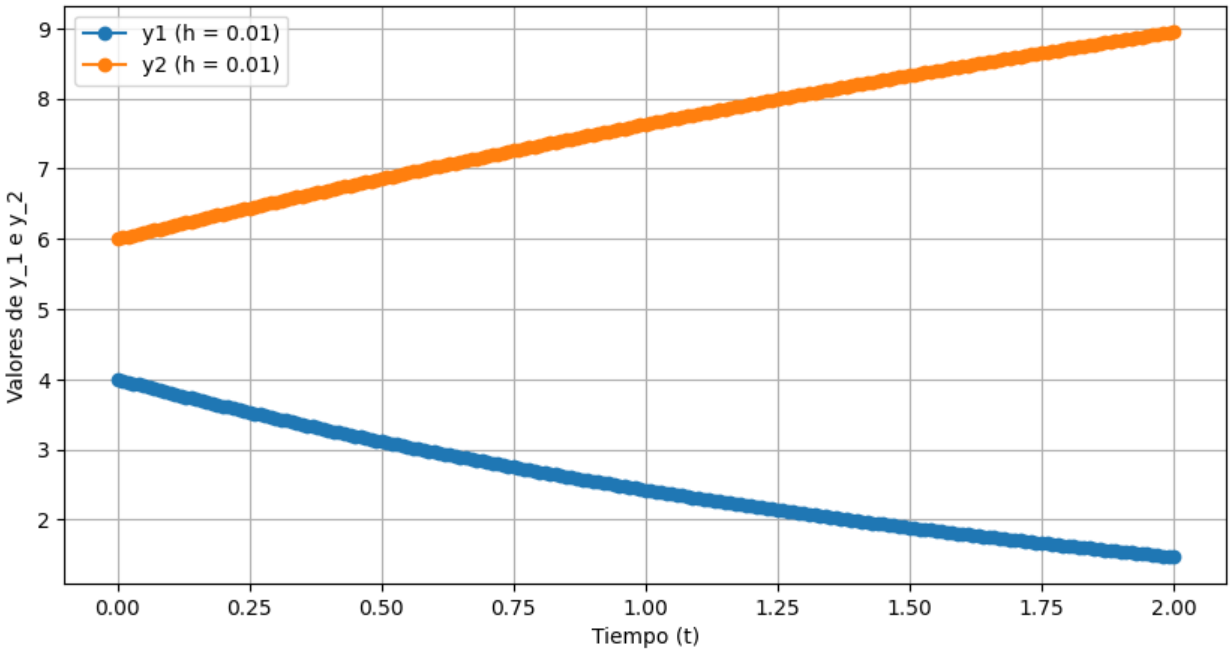

Figura 3: Valores de y_1 e y_2 para h=0.001

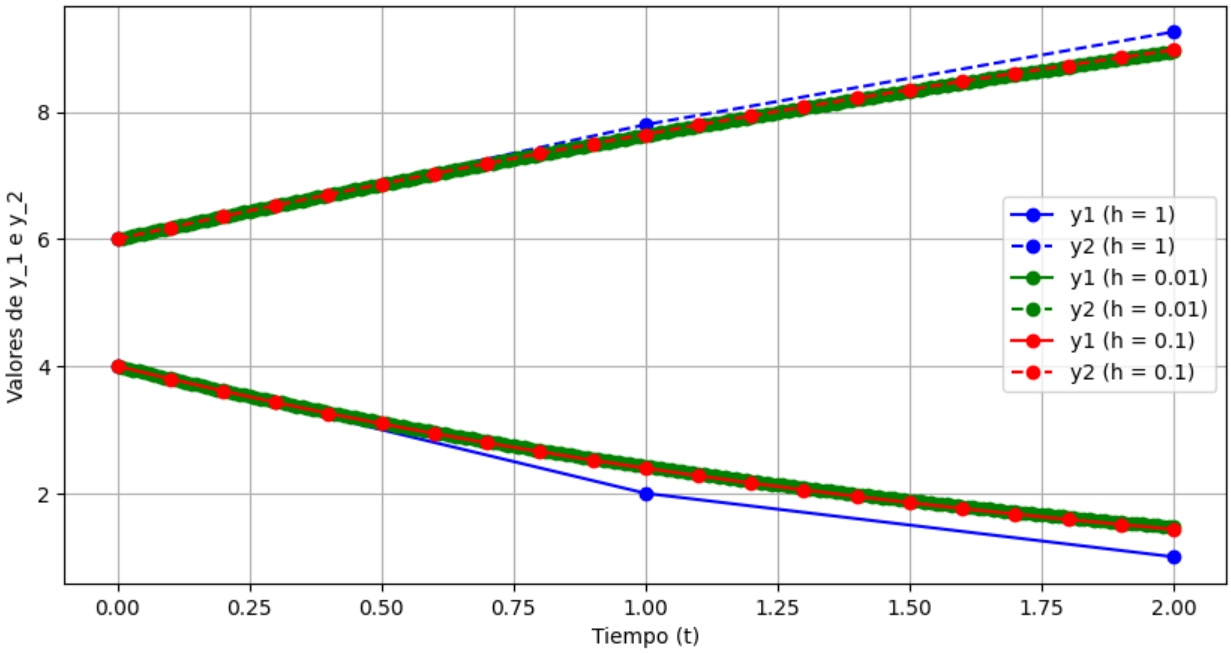

Figura 4: Comparación de los valores de y_1 e y_2 para h=1, h=0.1 y h=0.01

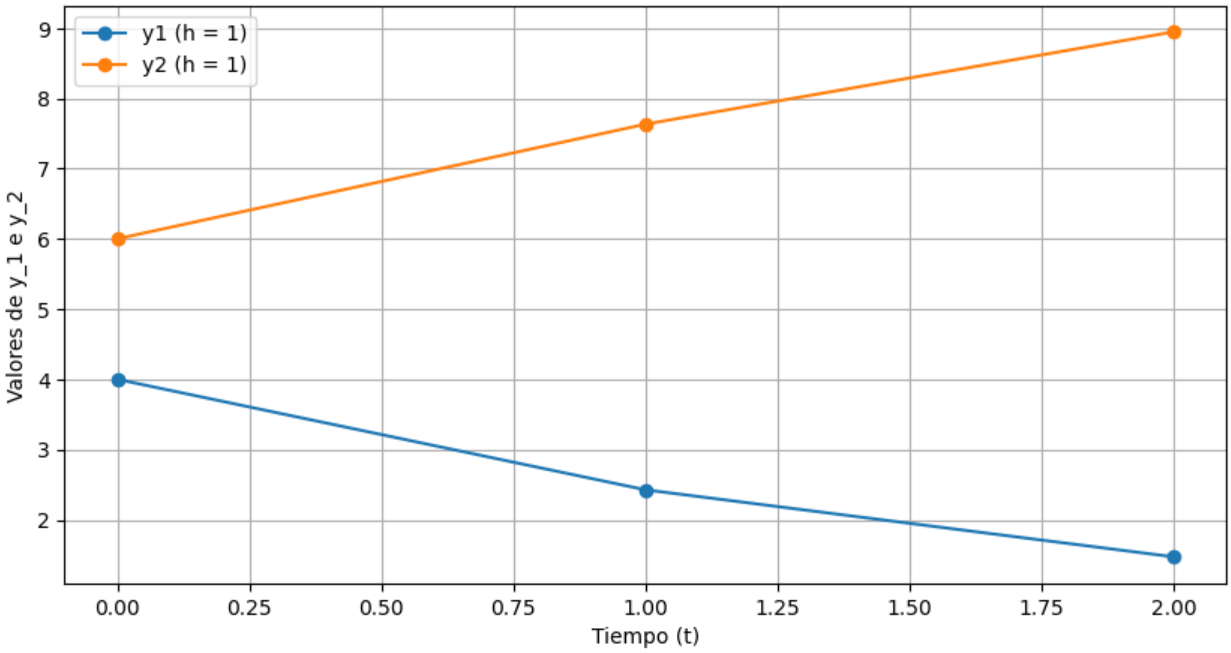

Figura 5: Valores de y_1 e y_2 para h=1

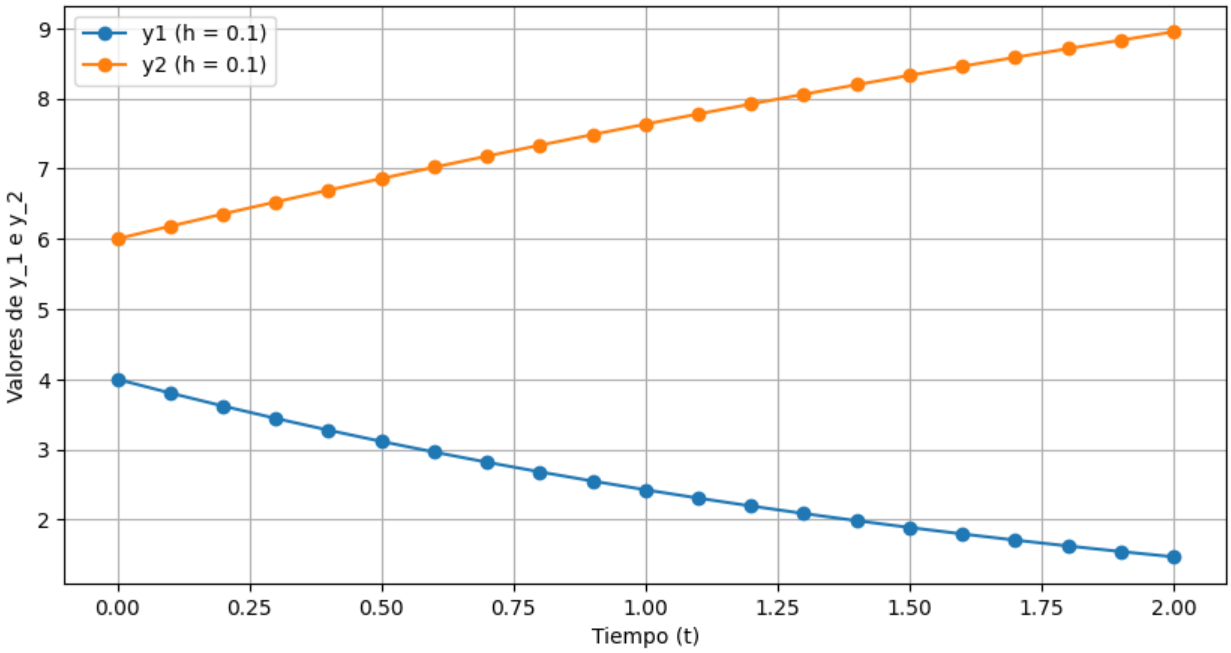

Figura 6: Valores de y_1 e y_2 para h=1

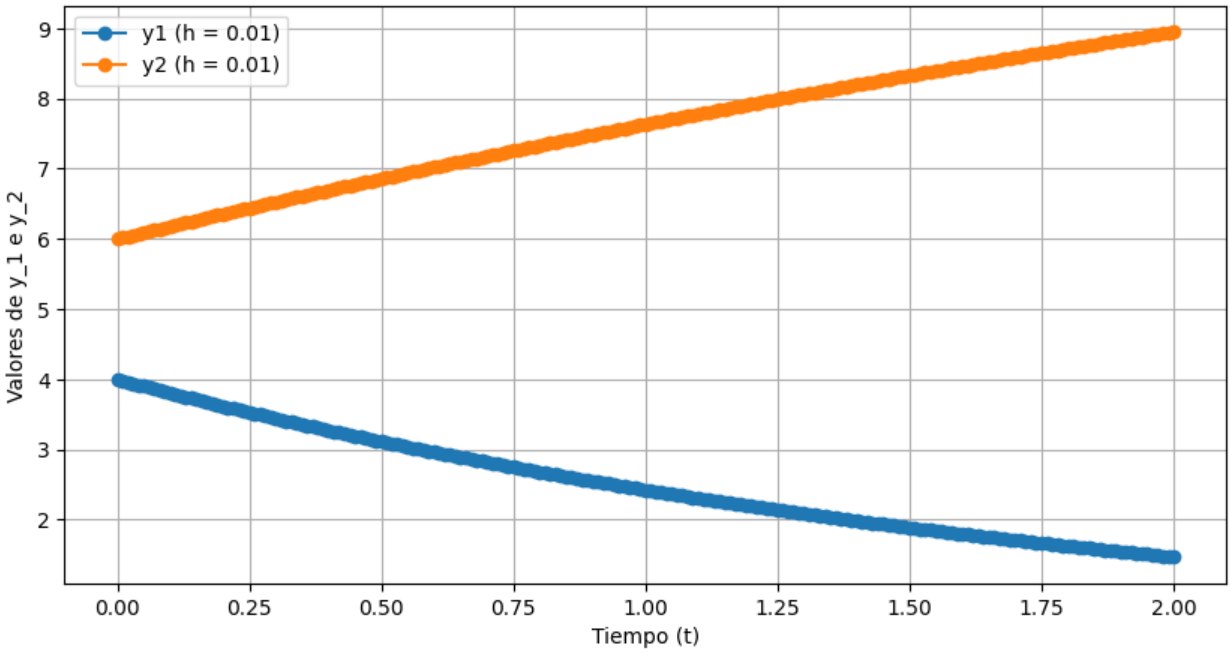

Figura 7: Valores de y_1 e y_2 para h=1

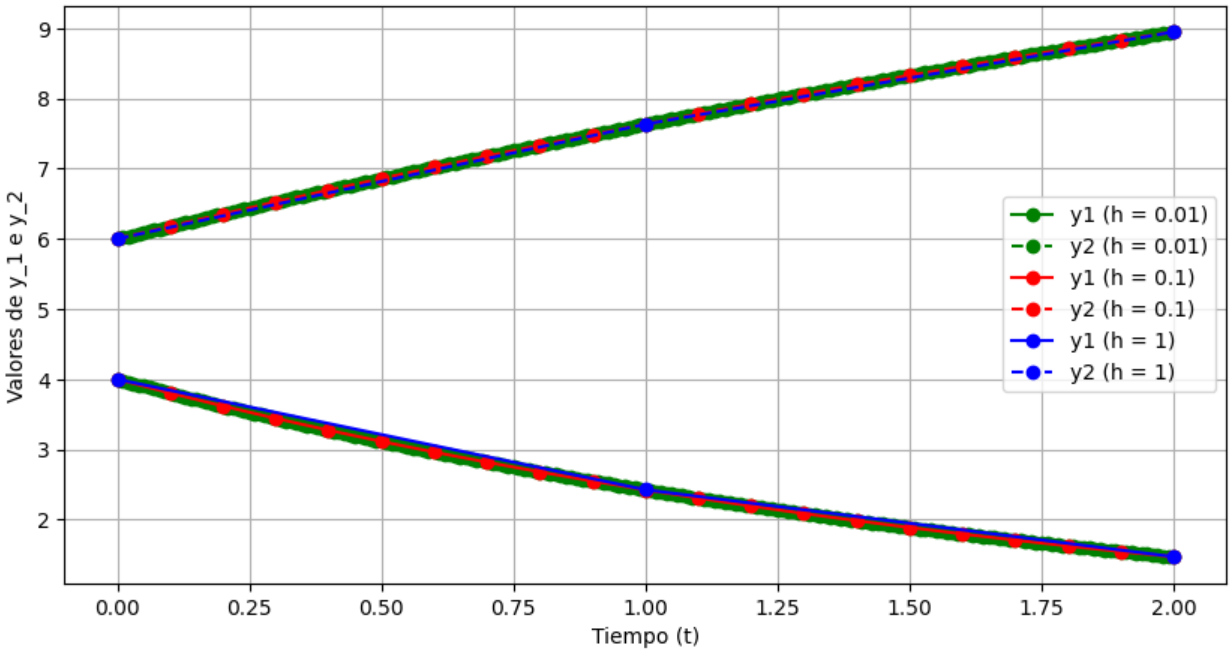

Figura 8: Valores de y_1 e y_2 para h=1

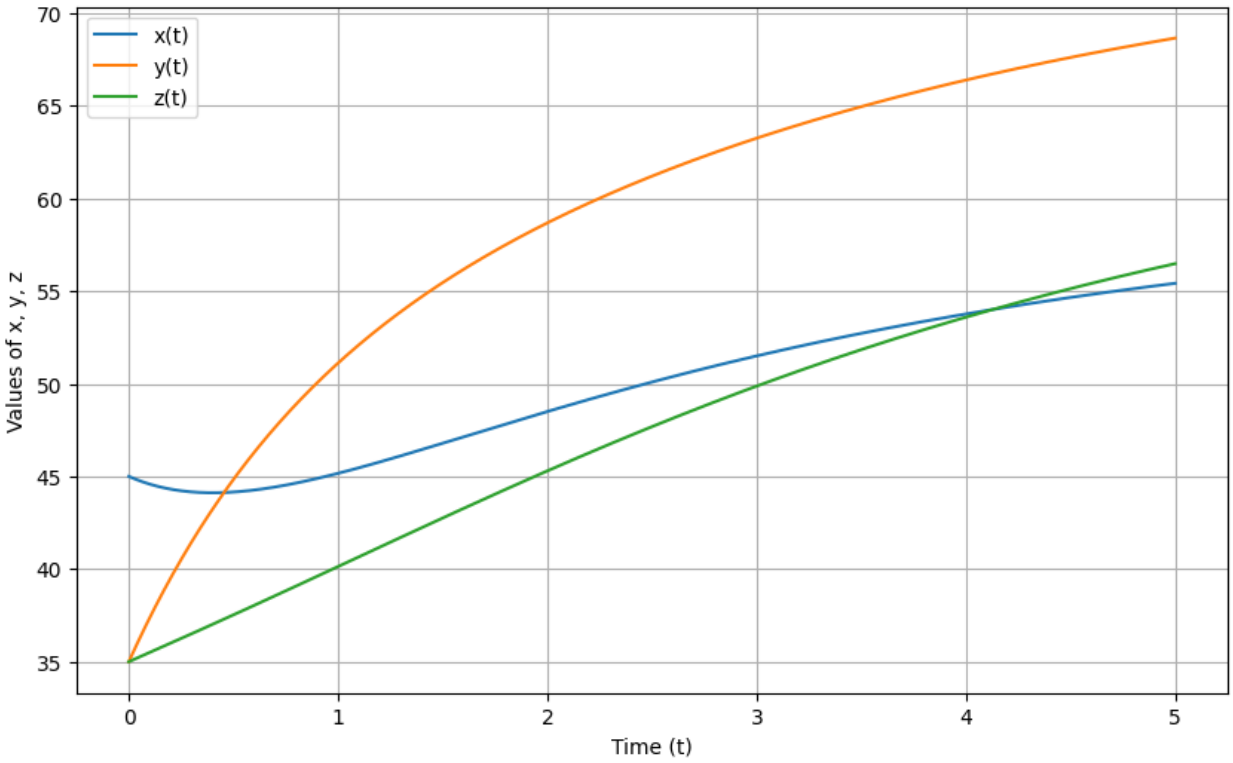

Figura 9: Apartado 3 con Euler

En esta sección se presentan los resultados obtenidos al resolver el sistema de ecuaciones diferenciales utilizando los métodos de Euler y Runge-Kutta de orden 4.

Al aplicar el método de Euler con diferentes tamaños de paso ($h = 1$, $h = 0{,}1$ y $h = 0{,}01$), se observa que el método introduce una mayor acumulación de error a medida que el tamaño de paso aumenta, especialmente en intervalos más grandes. Las gráficas muestran cómo un paso más pequeño mejora la precisión, pero incrementa la cantidad de cálculos necesarios.

En contraste, los resultados obtenidos mediante el método de Runge-Kutta de orden 4 son notablemente más precisos y estables. Aun con un tamaño de paso más grande, este método logra aproximaciones que se acercan mejor a la solución exacta, demostrando su ventaja en términos de precisión y eficiencia para problemas donde se requiere un alto grado de exactitud.

Estas diferencias se reflejan en las gráficas generadas, donde se aprecia cómo los métodos responden al variar el tamaño de paso. La comparación entre ambos métodos permite concluir que, para sistemas sensibles a errores acumulativos, el método de Runge-Kutta de orden 4 proporciona mejores resultados en términos de estabilidad y precisión.

4. Discusión

En este taller se aplicaron los métodos de Euler y Runge-Kutta de orden 4 para resolver sistemas de ecuaciones diferenciales, permitiendo evaluar las diferencias entre ambos en términos de precisión y eficiencia computacional. Utilizamos varios tamaños de paso ($h = 1, 0{,}1, 0{,}01$ y $h = 0{,}001$) para observar cómo el tamaño de paso influye en la precisión de la solución.

1. **Precisión de los métodos:** Los resultados mostraron que el método de Runge-Kutta de orden 4 es notablemente más preciso que el método de Euler para los mismos tamaños de paso, como era de esperarse. A pesar de que el método de Euler es más simple y rápido de implementar, su precisión disminuye considerablemente para pasos grandes, mostrando acumulación de error. En contraste, el método de Runge-Kutta se mantuvo preciso incluso con tamaños de paso moderados.

2. **Influencia del tamaño de paso:** Observamos que, al disminuir el tamaño de paso, ambos métodos aumentaron su precisión. Sin embargo, el método de Euler requiere pasos mucho más pequeños para alcanzar una precisión similar a la obtenida con Runge-Kutta de orden 4 en pasos más grandes. Esto resalta la ventaja de métodos de orden superior en términos de precisión frente a eficiencia computacional.

3. **Estabilidad y convergencia:** Para este tipo de sistemas de ecuaciones, la estabilidad es un factor crítico. El método de Runge-Kutta mostró ser más estable en todos los tamaños de paso, mientras que el método de Euler comenzó a diverger en pasos grandes, demostrando ser menos adecuado para problemas en donde se requiere precisión en intervalos largos o sistemas sensibles.

5. Conclusiones

1. Los métodos de Euler y Runge-Kutta de orden 4 son útiles para resolver sistemas de ecuaciones diferenciales, pero difieren en precisión y estabilidad. Mientras que el método de Euler es fácil de implementar y computacionalmente ligero, el método de Runge-Kutta proporciona mayor precisión y estabilidad, especialmente cuando se necesitan soluciones exactas en intervalos largos.

2. En problemas donde la precisión es crucial y el sistema es sensible a errores, el método de Runge-Kutta de orden 4 es preferible, ya que permite obtener soluciones estables y precisas con tamaños de paso más grandes, reduciendo el costo computacional en comparación con el método de Euler.

3. Se recomienda elegir el método y el tamaño de paso en función de los requisitos de precisión y eficiencia computacional específicos del problema. Para problemas donde la velocidad y simplicidad son esenciales, el método de Euler es adecuado, mientras que para aplicaciones que requieren alta precisión, el método de Runge-Kutta es más adecuado.

6. Referencias

Douglas J. Faires and Richard L. Burden, *Análisis Numérico*, Cengage Learning, 1998.

7. Anexos

Las funciones utilizadas para realizar los ejercicios son las siguientes.

Función del Método de Runge-Kutta

```
def runge_kutta_4(f, t0, tf, y0, N):
    # Calcula el tamaño del paso h basándose en los límites de integración
```

```
h = (tf - t0) / N

# Crea un array de valores de t que va desde t0 hasta tf con N+1 puntos
t_values = np.linspace(t0, tf, N + 1)

# Inicializa un array para almacenar los valores de y, con tamaño N+1
y_values = np.zeros(N + 1)

# Asigna el valor inicial y0 al primer elemento de y_values
y_values[0] = y0

# Itera sobre el número de pasos N
for i in range(N):
    # Obtiene el valor actual de t y y
    t = t_values[i]
    y = y_values[i]

    # Calcula las pendientes k1, k2, k3, y k4 usando la función f
    k1 = h * f(t, y)  # Pendiente en el inicio del intervalo
    k2 = h * f(t + h/2, y + k1/2)  # Pendiente en el medio del intervalo usando k1
    k3 = h * f(t + h/2, y + k2/2)  # Pendiente en el medio del intervalo usando k2
    k4 = h * f(t + h, y + k3)  # Pendiente al final del intervalo usando k3

    # Actualiza el valor de y para el siguiente paso
    y_values[i + 1] = y + (k1 + 2*k2 + 2*k3 + k4) / 6

# Devuelve los arrays de valores de t y de y
return t_values, y_values


def euler_method(h, t_range, y1_0, y2_0):
    # Crea valores de tiempo en el intervalo, con paso h
    t_values = np.arange(t_range[0], t_range[1] + h, h)

    # Inicializa los arreglos para y1 y y2
    y1_values = np.zeros(len(t_values))
```

```
y2_values = np.zeros(len(t_values))

# Asigna condiciones iniciales
y1_values[0] = y1_0
y2_values[0] = y2_0

# Aplica el método de Euler
for i in range(1, len(t_values)):
    y1_values[i] = y1_values[i-1] + h * f1(y1_values[i-1])
    y2_values[i] = y2_values[i-1] + h * f2(y1_values[i-1], y2_values[i-1])

# Devuelve el tiempo y las soluciones de y1 y y2
return t_values, y1_values, y2_values
```