Universidad de Buenos Aires
Facultad De Ingeniería
Año 2023 - 2do Cuatrimestre

Análisis Numérico I (75.12 – 95.04)

TRABAJO PRÁCTICO
TEMA:Resolución numérica de problemas de valores iniciales
FECHA:21 de Noviembre 2023

INTEGRANTES:

Aramayo Zambrana, Carolina Luna <caramayo@fi.uba.ar>	#106260
Castro Martinez, José Ignacio <jcastrom@fi.uba.ar>	#106957
Buchanan, Felix Tomas <fbuchanan@fi.uba.ar>	#102665

Índice

1. Introducción **3**

2. Análisis inicial **3**

2.1. Ecuación del modelo 4

2.1.1. Lista de variables presentes 4

3. Desarrollo **4**

3.1. Discretizando la ecuación 4

3.1.1. Discretizacion del tiempo 4

3.1.2. Conversión en ecuación de primer grado 4

3.2. Planteando Euler explicito 5

3.3. Planteando Euler implícito 5

3.4. Planteando de Runge Kutta de orden 2 6

3.5. Solución sin amortiguación 7

3.5.1. Solución analítica 7

3.6. Solución con amortiguación 7

3.6.1. Solución c'(t) 7

3.6.2. Búsqueda del k y λ óptimo 7

4. Código **10**

4.1. Referencia al código utilizado 10

4.2. Código de cada método 10

4.2.1. Código de Euler explicito 10

4.2.2. Código de Euler implicito 10

4.2.3. Rugen Kutta orden 2 10

4.3. Métodos para calcular el error 11

5. Resultado de las ejecuciones **12**

5.1. Resultados para $h = 0{,}005$ 12

5.1.1. Resultados para Euler explícito 12

5.1.2. Resultados para Euler implícito 12

5.1.3. Resultados para Runge-Kutta de Orden 2 13

5.2. Resultados para $h = 0{,}01$ 14

5.2.1. Resultados para Euler explícito 14

5.2.2. Resultados para Euler implícito 14

5.2.3. Resultados para Runge-Kutta de Orden 2 15

5.3. Resultados para $h = 0{,}00001$ 16

5.3.1. Resultados para Euler explícito 16

5.3.2. Resultados para Euler implícito 16

5.3.3. Resultados para Runge-Kutta de Orden 2 17

6. Selección del método **17**
6.1. Análisis de la complejidad de los métodos 17
6.1.1. Complejidad de Euler Explicito 18
6.1.2. Complejidad de Euler Implicito 18
6.1.3. Complejidad de Runge Kutta 18
6.2. Análisis del error 18
6.2.1. Análisis del error para h = 0.005 18
6.2.2. Análisis del error para h = 0.01 18
6.2.3. Análisis del error para h = 0.0001 18
6.3. Selección de un método 18
6.4. Calculo del orden de los métodos 19

7. Calculo experimental de la frecuencia **19**

8. Ejecutando Runge Kutta de orden dos con amortiguamiento **19**

9. Conclusiones **20**

1. Introducción

El presente informe tiene el propósito de realizar un análisis sobre un problema de amortiguamiento utilizando métodos numéricos de resolución de ecuaciones diferenciales. Entre los métodos a utilizar se encuentran: Euler explicito, Euler implícito y Runge Jutta de orden 2. El presente problema viene modelado por un sistema físico oscilatorio, el cual viene representado por una ecuación diferencial lineal de segundo grado. A continuación, se presentan algunas claves necesarias para expandir el entendimiento del modelado

2. Análisis inicial

Para empezar, se realiza un análisis del problema de forma física. A continuación se presentan gráficos que representan una base en la modulación del problema

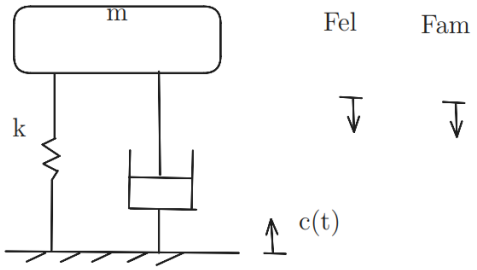

Figura 1: Esquema del modelador del amortiguador

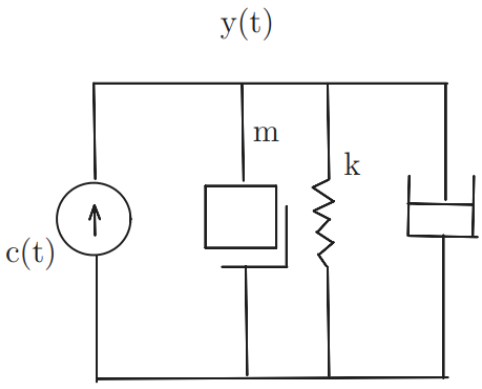

Figura 2: Esquema del circuito mecánico

El problema nos presenta dos situaciones:

1. Por un lado tendremos un oscilador armónico, el cual trataremos como sin amortiguar, con excitación uniforme.
2. Y por otro lado tenemos el oscilador amortiguado que lo trataremos como amortiguador, buscando optimizar la amortiguación del vehículo. Para ello se plantean restricciones sobre el desplazamiento, la aceleración y las oscilaciones de la carrocería.

2.1. Ecuación del modelo

El enunciado brinda la siguiente ecuación diferencial:

$$y'' = \frac{k}{m}(c - y) + \frac{\lambda}{m}(c' - y') \tag{1}$$

Dicha ecuación representa la aceleración vertical de la carrocería, es decir, el oscilador amortiguado que responde a una excitación dada por la variable c.

2.1.1. Lista de variables presentes

Dada la ecuación diferencial anteriormente dada se proporcionan la infamación asociada a las variables que la componen y las unidades físicas que las representan

1. k constante elástica del muelle [N/m]
2. λ constante de amortiguación [Ns/m]
3. c cota o elevación del terreno [m]
4. y posición de la carrocería [m]
5. c' y y' son derivadas de c e y con respecto al tiempo, es decir, velocidades verticales [m/s]

3. Desarrollo

3.1. Discretizando la ecuación

Para poder resolver la ecuación diferencial dada:

$$y'' = \frac{k}{m}(c - y) + \frac{\lambda}{m}(c' - y') \tag{2}$$

En un principio se trabajara sobre la ecuación en general y posteriormente se realiza el reemplazo con los valores dados del caso particular a analizar.

3.1.1. Discretizacion del tiempo

Es importante primero realizar una discretización de los datos de entrada de la ecuación. En un principio vale que la entrada del problema, la variable t (el tiempo) debe verse como un valor discreto, para poder estimar el valor que toma la ecuación dado un instante en particular. Por lo cual es importante destacar que la unidad t va a estar dada en segundos. Por ello, definimos:

$$h = \Delta t = t_{n+1} - t_n \tag{3}$$

3.1.2. Conversión en ecuación de primer grado

Por otro lado y regresando al ítem anterior para poder discretizar la ecuación (2) y aplicar los métodos anteriormente dichos es necesario convertir la ecuación diferencial de 2do grado en una de primer grado para lo cual se plantea lo siguiente

$$\begin{cases} v(t) = y'(t) \\ v'(t) = \frac{k}{m}(c - y) + \frac{\lambda}{m}(c' - y') \end{cases} \tag{4}$$

Esta nueva función $v(t)$ es simplemente una ayuda para poder cumplir con los requerimientos del método. Una vez planteado esto se puede discretizar la ecuación diferencial

Es importante observar que al discretizar la ecuación lo que se desea conseguir es que los valores aproximados tomen un valor con la menor diferencia posible al valor real que tomaría. Es decir para un tiempo t_n en el continuo queremos que la aproximación tome un valor parecido en el entorno de t

$$\begin{cases} y(t) \approx y(t_n) \\ y'(t) \approx y'(t_n) \end{cases} \tag{5}$$

Por lo tanto, con el sistema anteriormente planteado se tiene lo siguiente:

$$\begin{cases} f_1(v,u,t) = v \\ f_2(v,u,t) = \frac{k}{m}(c-y) + \frac{\lambda}{m}(c'-y') \end{cases} \tag{6}$$

3.2. Planteando Euler explicito

El método de Euler explicito plantea que la solución numérica se encuentre mediante:

$$u_{n+1} = u_n + h \cdot f(u_n, t_n) \tag{7}$$

En este caso es necesario aplicar el método a las dos ecuaciones construidas anteriormente de lo cual se obtiene:

$$\begin{cases} u_{n+1} = u_n + h \cdot v_n \\ v_{n+1} = v_n + h[\frac{k}{m}(c-y) + \frac{\lambda}{m}(c'-y')] \end{cases} \tag{8}$$

También se resalta el caso en el cual la amortiguación es nula:

$$\begin{cases} u_{n+1} = u_n + h \cdot v_n \\ v_{n+1} = v_n + h[\frac{k}{m}(c-y)] \end{cases} \tag{9}$$

3.3. Planteando Euler implícito

El método de Euler implícito plantea que la solución numérica se encuentre mediante:

$$u_{n+1} = u_n + h \cdot f(u_{n+1}, t_{n+1}) \tag{10}$$

Por lo tanto se construyen las ecuaciones:

$$\begin{cases} u_{n+1} = u_n + hv_n \\ v_{n+1} = v_n + h[\frac{k}{m}(c-u_{n+1}) + \frac{\lambda}{m}(c'-v_{n+1})] \end{cases}$$

Despejando u_n y v_n, obtenemos:

$$\begin{cases} u_n &= u_{n+1} - hv_{n+1} \\ v_n &= v_{n+1} - h[\frac{k}{m}(c-u_{n+1}) + \frac{\lambda}{m}(c'-v_{n+1})] \end{cases}$$

Decidimos trabar este método de forma matricial, por lo cual debemos realizar la distributiva:

$$\begin{cases} u_n &= u_{n+1} - hv_{n+1} \\ v_n &= v_{n+1}(1 + \frac{h\lambda}{m}) + \frac{hk}{m}u_{n+1} - h\left(\frac{k}{m}c + \lambda\frac{c'}{m}\right) \end{cases}$$

Pasamos el sistema de ecuaciones a la forma matricial:

$$\begin{bmatrix} 1 & -h \\ \frac{hk}{m} & 1 + \frac{h\lambda}{m} \end{bmatrix} \begin{bmatrix} u_{n+1} \\ v_{n+1} \end{bmatrix} = \begin{bmatrix} u_n \\ v_n + h\left(\frac{k}{m}c + \lambda\frac{c'}{m}\right) \end{bmatrix} \tag{11}$$

La forma matricial es entonces:

$$\begin{bmatrix} 1 & -h \\ \frac{hk}{m} & 1+\frac{h\lambda}{m} \end{bmatrix} \begin{bmatrix} u_{n+1} \\ v_{n+1} \end{bmatrix} - \begin{bmatrix} 0 \\ h\left(\frac{k}{m}c + \lambda\frac{c'}{m}\right) \end{bmatrix} = \begin{bmatrix} u_n \\ v_n \end{bmatrix} \tag{12}$$

Por lo tanto es necesario hallar la inversa de la matriz para poder realizar un despeje del sistema de ecuaciones. Es importante destacar que el calculo utilizado en el siguiente del paso depende de que la matriz sea no singular

La cual se puede obtener facilmente y es:

$$\begin{bmatrix} \frac{m+h\lambda}{m+h\lambda+h^2k} & \frac{-hk}{m+h\lambda+h^2k} \\ \frac{mh}{m+h\lambda+h^2k} & \frac{m}{m+h\lambda+h^2k} \end{bmatrix} \tag{13}$$

Y por lo tanto podemos obtener la matriz de iteración fácilmente de la siguiente forma:

$$\begin{bmatrix} u_{n+1} \\ v_{n+1} \end{bmatrix} = \begin{bmatrix} \frac{m+h\lambda}{m+h\lambda+h^2k} & \frac{-hk}{m+h\lambda+h^2k} \\ \frac{mh}{m+h\lambda+h^2k} & \frac{m}{m+h\lambda+h^2k} \end{bmatrix} \left(\begin{bmatrix} u_n \\ v_n \end{bmatrix} + \begin{bmatrix} 0 \\ h\left(\frac{k}{m}c + \lambda\frac{c'}{m}\right) \end{bmatrix} \right) \tag{14}$$

Destacando también el caso sin amortiguamiento:

$$\begin{bmatrix} u_{n+1} \\ v_{n+1} \end{bmatrix} = \begin{bmatrix} \frac{m}{m+h^2k} & \frac{-hk}{m+h^2k} \\ \frac{mh}{m+h^2k} & \frac{m}{m+h^2k} \end{bmatrix} \left(\begin{bmatrix} u_n \\ v_n \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{hkc}{m} \end{bmatrix} \right) \tag{15}$$

3.4. Planteando de Runge Kutta de orden 2

El método de RK2 es de la siguiente manera:

$$\begin{cases} q_1 & = hf(u_n, t_n) \\ q_2 & = hf(u_n + q_1, t_{n+1}) \\ u_{n+1} & = u_n + \frac{1}{2}(q_1 + q_2) \end{cases}$$

Tendremos las siguientes ecuaciones:

$$\begin{cases} q_1 = hv_n \\ q_2 = hf(u_n + hv_n, t_{n+1}) \\ u_{n+1} = u_n + \frac{1}{2}(q_1 + q_2) \end{cases} \tag{16}$$

Tenemos a Por lo tanto se construyen las ecuaciones:

$$f(u_{n+1}, t_{n+1}) = [\frac{k}{m}(c - u_{n+1}) + \frac{\lambda}{m}(c' - v_{n+1})] \tag{17}$$

Entonces:

$$f(u_{n+1} + hv_n, t_{n+1}) = [\frac{k}{m}(c - (u_{n+1} + hv_n)) + \frac{\lambda}{m}(c' - v_{n+1})] \tag{18}$$

Por lo tanto:

$$(q_1 + q_2) = hv_n + h([\frac{k}{m}(c - (u_{n+1} + hv_n)) + \frac{\lambda}{m}(c' - v_{n+1})]) \tag{19}$$

Entonces las variables quedan:

$$u_{n+1} = u_n + \frac{1}{2}hv_n + h([\frac{k}{m}(c - (u_{n+1} + hv_n)) + \frac{\lambda}{m}(c' - v_{n+1})]) \tag{20}$$

3.5. Solución sin amortiguación

3.5.1. Solución analítica

Por otro lado con la ayuda de un software especializado se obtiene una solución analítica de la ecuación diferencial con los datos brindados en el enunciado. La ecuación original es:

$$y(t) = c + \alpha_2 \sin\left(\sqrt{\frac{kt}{m}}\right) + \alpha_1 \sin\left(\sqrt{\frac{kt}{m}}\right) \tag{21}$$

Luego, reemplazando por los valores iniciales, la ecuación finalmente queda:

$$c = 0{,}1 \tag{22}$$
$$y(t) = c - c \cdot \cos(\omega t) \tag{23}$$

3.6. Solución con amortiguación

3.6.1. Solución $c'(t)$

Para hallar $c'(t)$, decidimos dividir el perfil del terreno en pequeños pedazos de tiempo. Entonces:

$$c'(t) = \frac{\Delta c}{\Delta t} \tag{24}$$

Por lo tanto, $c'(t)$ en función del tiempo queda como:

$$c'(t) = \begin{cases} 0 & \text{si } 0 \leq c \leq 1 \\ 1 & \text{si } 1 \leq c \leq 1{,}1 \\ 0 & \text{si } 1{,}1 \leq c \leq 1{,}3 \\ -1 & \text{si } 1{,}3 \leq c \leq 1{,}4 \\ 0 & \text{si } 1{,}4 \leq c \leq 5 \end{cases}$$

3.6.2. Búsqueda del k y λ óptimo

Para hallar los valores óptimos, vamos a dejar que una variable varíe y la otra esté constante y viceversa.

k	λ	Resultado
15000	500	-0.0968098669238602
15000	1000	-0.09522580454339995
15000	1500	-0.09367540519980944
15000	2000	-0.09215786773712362
15000	2500	-0.09067241120438753
15000	3000	-0.08921827432918411
15000	3500	-0.0877947150051363
15000	500	-0.0968098669238602
20000	500	-0.09629776453578828
25000	500	-0.09578725374983617
30000	500	-0.10238501467131668
35000	500	-0.11039128786344855
40000	500	-0.11752925046428406
30000	500	-0.10238501467131668
30000	1000	-0.09371924095547715
30000	1500	-0.09219333678520902
30000	2000	-0.09069982748078406
30000	2500	-0.08923794186578049
30000	3000	-0.08780692822556158
30000	3500	-0.08640605379976532

Cuadro 1: Resultados de la búsqueda del k y λ óptimos.

La búsqueda del k y λ óptimos reveló patrones interesantes en la variación del resultado en función de los parámetros que se encuentran en la siguiente tabla. Al observar los datos, se puede concluir que k y λ tienen un impacto significativo en el resultado del sistema. Aumentar k parece estar asociado con una disminución marcada en el resultado, indicando una sensibilidad a la rigidez del sistema. Por otro lado, un aumento en λ muestra una relación inversa con el resultado, sugiriendo que valores más altos de amortiguamiento están vinculados a una mejora en el comportamiento del sistema.

La interacción entre k y λ también es evidente, especialmente al observar cómo variar λ afecta el resultado para un valor específico de k (por ejemplo, $k = 30000$). Esto sugiere que la selección óptima de λ puede depender del valor específico de k, y viceversa.

Como estrategia adicional, se planteó mantener la relación entre los valores de k y λ proporcionados en el enunciado como una relación estándar para las futuras búsquedas de k y λ. Esta consideración se basa en la premisa de que los componentes, el muelle y el amortiguador, con dichos valores poseen una calidad comparable.

Para implementar este enfoque, se llevó a cabo una exploración sistemática considerando valores de λ en el rango de 1000 a 12000, con incrementos de 50, y valores de k en el rango de 3000 a 100000, con incrementos de 1000. Se empleó un algoritmo de fuerza bruta que evaluó exhaustivamente cada combinación de estos valores, buscando la combinación más cercana a los valores iniciales de manera proporcional para lograr la compresión deseada.

Un objetivo clave de este segundo método fue prescindir de la necesidad de identificar el "mejor"muelle y amortiguador para cumplir con las condiciones requeridas. Los valores resultantes de esta búsqueda fueron $k = 10550$ y $\lambda = 5550$. El gráfico correspondiente a estos valores se presenta a continuación:

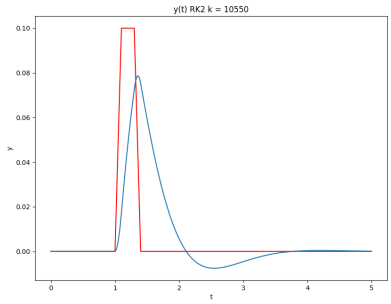

Figura 3: Gráfico del sistema amortiguado con h=0.005 y compresión=0.05

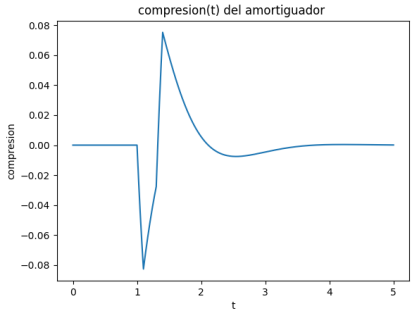

Figura 4: Gráfico de compresión

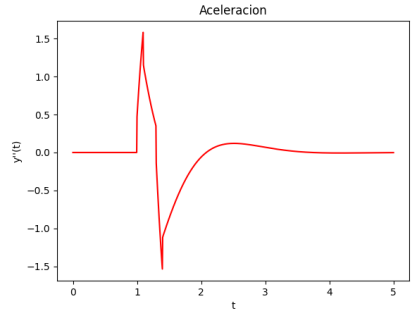

Figura 5: Gráfico de aceleración vertical y" en función del tiempo

4. Código

4.1. Referencia al código utilizado

La resolución del código se encontrara disponible en un notebook de python como también un cripta de mathlab. Se añade al informe el código utilizado de los métodos en cada caso, como también los links a los elementos usados para el desarollo del informe:

Notebook con solución en python

Script en mathlab

4.2. Código de cada método

4.2.1. Código de Euler explicito

```python
def euler_explicito(funcion, intervalo, paso):
    num_puntos = int(intervalo / paso) + 1
    resultados_u = np.zeros(num_puntos, dtype=float)
    resultados_v = np.zeros(num_puntos, dtype=float)
    resultados_u[0] = 0
    resultados_v[0] = 0

    for n in range(num_puntos - 1):
        u_actual = resultados_u[n] + paso * resultados_v[n]
        resultados_u[n+1] = u_actual
        v_actual = resultados_v[n] + paso * funcion(resultados_u[n])
        resultados_v[n+1] = v_actual

    return resultados_u
```

Listing 1: Codigo utilizado

4.2.2. Código de Euler implicito

```python
def euler_implicito(inversa_matriz_a, termino_independiente, intervalo, paso):
    num_puntos = int(intervalo / paso) + 1
    resultados = np.zeros((2, num_puntos))

    for n in range(num_puntos - 1):
        valores_auxiliares = inversa_matriz_a @ resultados[:, n] +
    termino_independiente
        resultados[:, n+1] = valores_auxiliares

    return resultados[1]
```

Listing 2: Codigo utilizado

4.2.3. Rugen Kutta orden 2

```python
    def runge_kuta_orden2(f, intervalo, paso):
    num_puntos = int(intervalo / paso) + 1
    resultados_u = np.zeros(num_puntos, dtype=float)
    resultados_v = np.zeros(num_puntos, dtype=float)
    resultados_u[0] = 0
    resultados_v[0] = 0

    for n in range(num_puntos - 1):
        u_prediccion = resultados_u[n] + paso * resultados_v[n]
        v_prediccion = resultados_v[n] + paso * f(resultados_u[n])

        resultados_u[n+1] = resultados_u[n] + (paso/2) * (resultados_v[n] +
    v_prediccion)
```

```
13          resultados_v[n+1] = resultados_v[n] + (paso/2) * (f(resultados_u[n]) + f(
    u_prediccion))
14
15      return resultados_u
```

Listing 3: Codigo utilizado

4.3. Métodos para calcular el error

Se agregan al informe también las funciones que realizan el calculo del error de cada uno de los métodos. La lógica es muy parecida entre si pero separamos las implementaciones para que sea mas didáctica la lectura dentro del notebook de python

```
1   def error_euler(aproximacion,intervalo,h, analitica):
2   paso = 0
3   for i in range(len(aproximacion)):
4       aproximacion[i] = (analitica(paso) - aproximacion[i])
5       paso+=h
6   return error
7
8
9   def error_rugen_kutta(aproximacion, intervalo, h, analitica):
10      paso = 0
11      error = aproximacion[:]
12      for i in range(len(aproximacion)):
13          error[i] = analitica(paso) - aproximacion[i]
14          paso += h
15      return error
```

Listing 4: Métodos para calcular el error

5. Resultado de las ejecuciones

A continuación se mostrarán los resultados de las ejecuciones de cada método.

5.1. Resultados para $h = 0{,}005$

5.1.1. Resultados para Euler explícito

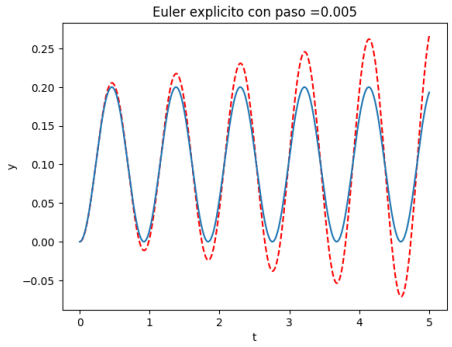

Figura 6: Euler explícito sin amortiguación contra la solución analítica

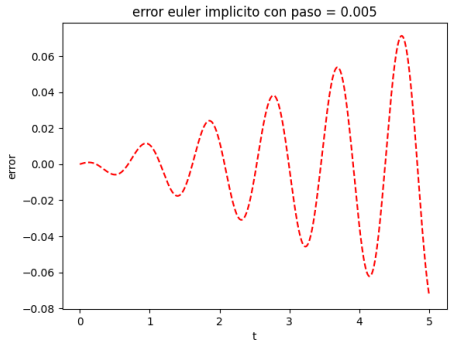

Figura 7: Error de Euler Explícito

5.1.2. Resultados para Euler implícito

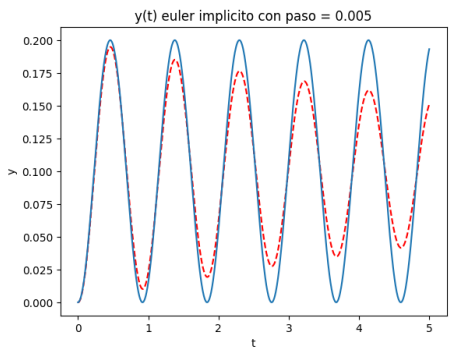

Figura 8: Euler implícito sin amortiguación contra la solución analítica

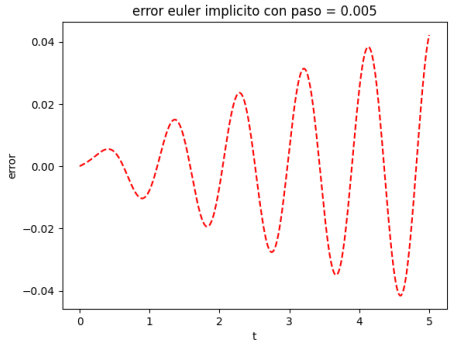

Figura 9: Error de Euler implícito

5.1.3. Resultados para Runge-Kutta de Orden 2

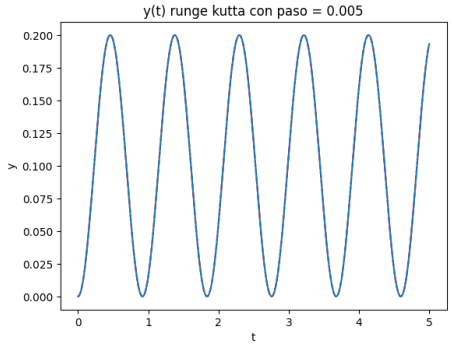

Figura 10: Runge-Kutta sin amortiguación contra la solución analítica

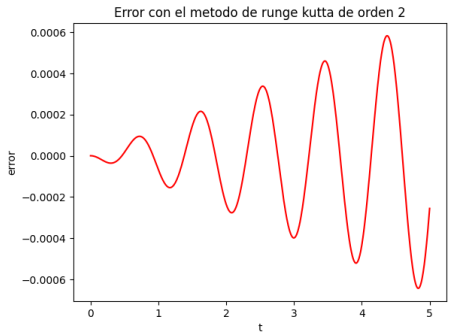

Figura 11: Error de Runge-Kutta de orden 2

5.2. Resultados para $h = 0{,}01$

5.2.1. Resultados para Euler explícito

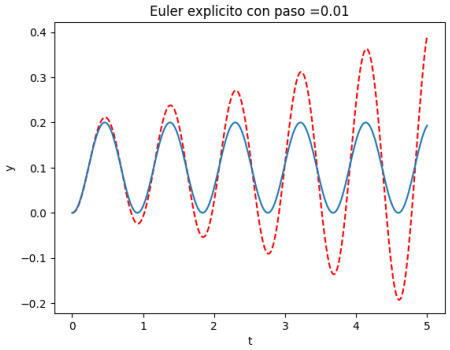

Figura 12: Euler explícito sin amortiguación contra la solución analítica

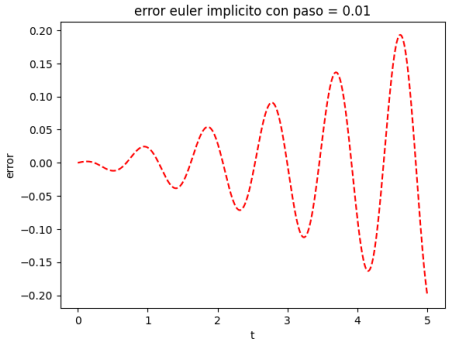

Figura 13: Error de Euler Explícito

5.2.2. Resultados para Euler implícito

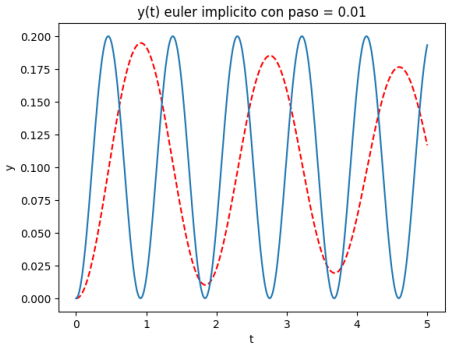

Figura 14: Euler implícito sin amortiguación contra la solución analítica

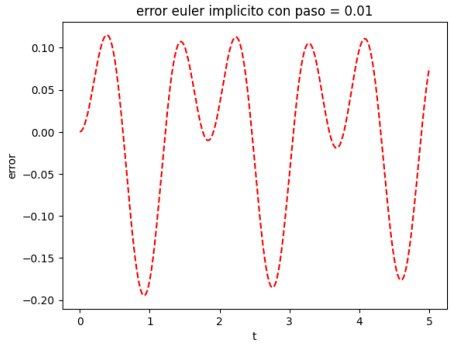

Figura 15: Error de Euler implícito

5.2.3. Resultados para Runge-Kutta de Orden 2

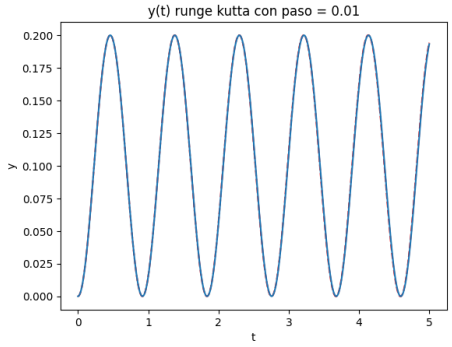

Figura 16: Runge-Kutta sin amortiguación contra la solución analítica

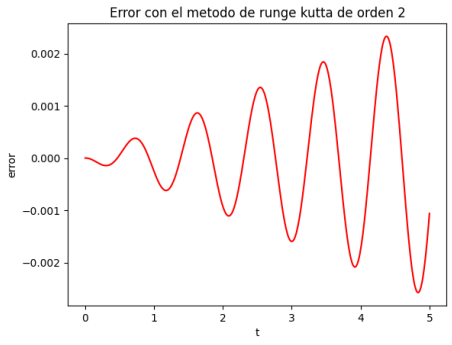

Figura 17: Error de Runge-Kutta de orden 2

5.3. Resultados para $h = 0{,}00001$

5.3.1. Resultados para Euler explícito

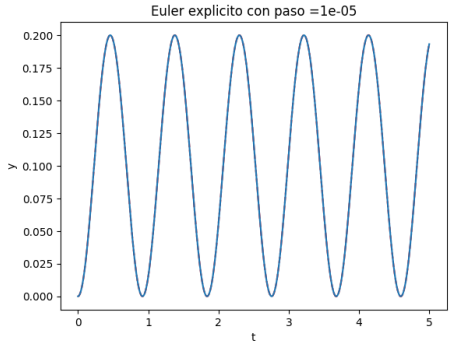

Figura 18: Euler explícito sin amortiguación contra la solución analítica

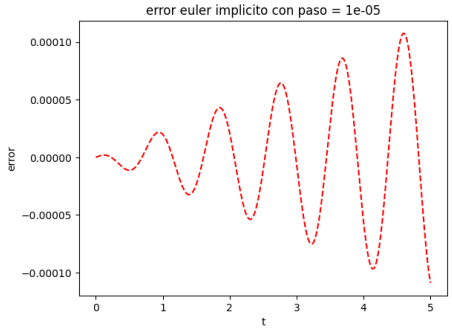

Figura 19: Error de Euler Explícito

5.3.2. Resultados para Euler implícito

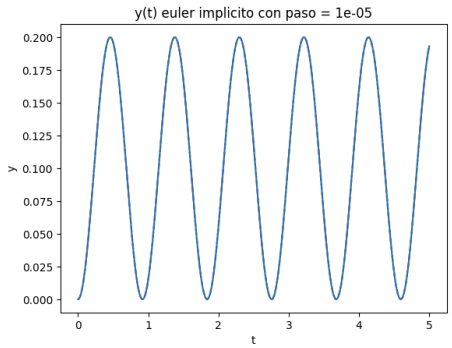

Figura 20: Euler implícito sin amortiguación contra la solución analítica

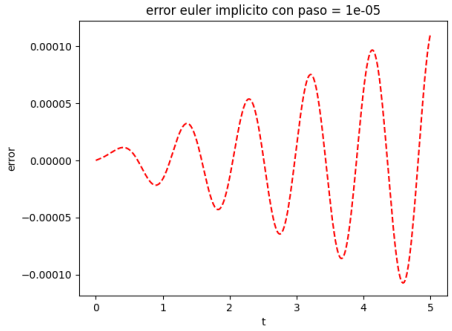

Figura 21: Error de Euler implícito

5.3.3. Resultados para Runge-Kutta de Orden 2

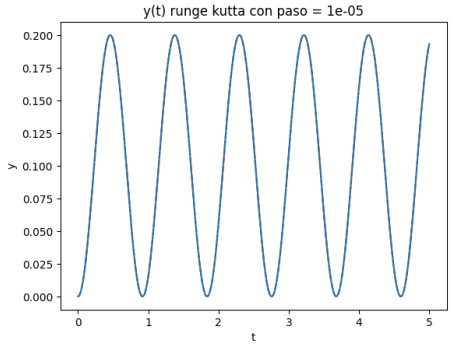

Figura 22: Runge-Kutta sin amortiguación contra la solución analítica

Figura 23: Error de Runge-Kutta de orden 2

6. Selección del método

6.1. Análisis de la complejidad de los métodos

Desde el punto de vista computacional se puede estimar las complejidades y temporales espaciales todos los métodos involucrados de manera teórica. De manera que se puede elegir el método que mejor equilibre precisión y complejidad a la hora de realizar el análisis sobre una ecuación diferencial

6.1.1. Complejidad de Euler Explicito

En el caso de la ecuación diferencial dada la tanto la complejidad espacial y temporal del método en notación Big O equivale a ser $O(n)$. Temporalmente porque se realiza un ciclo definido y las operaciones internas al mismo todas podrían estimarse algoritmicamente como $O(1)$, a su vez, espacialmente solo requiere guardar los resultados de cada una de esas ejecuciones, por lo tanto se explica que ambas complejidades sean iguales

6.1.2. Complejidad de Euler Implicito

En el caso de la ecuación diferencial dada la tanto la complejidad espacial y temporal del método en notación Big O equivale a ser $O(n)$. En cada paso dada la implementación hecha para el informe se tiene que realizar una multiplicación matricial que se sabe es $O(n^3)$ pero debido que el tamaño de la matriz es constante entonces la complejidad algorítmica y espacial también pueden ser acotadas como el método anterior

6.1.3. Complejidad de Runge Kutta

Si bien este método realiza mas operaciones aritméticas es evidente que la cantidad de tiempo para realizarlas se puede acotar como $O(1)$ y en general el algoritmo también consiste en realizar una iteración sobre el intervalo dado, por lo tanto este algoritmo puede ser acotado espacial y temporalmente como $O(n)$

6.2. Análisis del error

Al haber realizado tres ejecuciones del método se puede apreciar como se comporta el mismo con distintos valores de h.

6.2.1. Análisis del error para h = 0.005

Para este tamaño de h se observa que los métodos en general se comportan de manera esperada. Todos tienen un error menor al 0.06 sin embargo es evidente que al observar las gráficas de comportamiento del error el método de Runge-Kutta de orden dos destaca sobre el resto de los métodos

6.2.2. Análisis del error para h = 0.01

Para este tamaño de h se observa un aumento considerado en los errores de todos los métodos sobretodo en los métodos de Euler que empeoran considerablemente su rendimiento a comparación del anterior valor de h, sin embargo, el método de Runge Kutta de orden dos aunque con un error mayor, sigue teniendo valores aceptables en su error

6.2.3. Análisis del error para h = 0.0001

Para este tamaño de h, los metodos de Euler implicito y Runge Kutta toman la delantera a la hora de disminuir el error, siendo claramente el metodo de Runge Kutta de orden dos el que claramente posee un error risible, se encuentra en la escala de 10^{-9}

6.3. Selección de un método

Debido al análisis anterior debe quedar claro que la selección del método considerando la complejidad algorítmica y espacial como también la precisión del mismo a la hora de modelar la ecuación diferencial numéricamente es el método de Runge Kutta de orden 2

6.4. Calculo del orden de los métodos

Para calcular el orden del método se hace lo siguiente;

$$p = \frac{\ln\left(\frac{\Delta x_{n+2}}{\Delta x_{n+1}}\right)}{\ln\left(\frac{\Delta x_{n+1}}{\Delta x_n}\right)} \tag{25}$$

Donde Δx_{n+i} es la resta entre el error de dos pasos consecutivos. En este caso se toma h = 0.005 para calcular el orden del método y se observan los resultados

- El orden de convergencia de Euler explícito es: 1.091634176201194
- El orden de convergencia de Euler implícito es: 1.0900572934323425
- El orden de convergencia de Runge-Kutta es: 1.0970729275266298

Dentro del notebook de python se encuentra el desarrollo en codigo del calculo ahora mostrado

7. Calculo experimental de la frecuencia

Para obtener la frecuencia experimental del método basta con realizar la búsqueda de los máximos pertenecientes a la función oscilatoria. Mediante el uso de la biblioteca de scipy de utiliza la función findPeaks() que encuentra los datos necesarios.

Una vez obtenido la posición en el arreglo de los picos, se encuentran los valores de *tdiscretizado* para los cuales la función tiene un máximo y diseminar el posible error introducido por la aproximación numérica se promedian las diferencias entre los t_i y se obtiene una frecuencia promedio del método.

Dentro de la notebook se realizan los cálculos para obtener la frecuencia experimental usando los tres métodos disponibles como también el h = 0.005 brindado por el enunciado. A continuación los resultados arrojados por el experimento

- La frecuencia experimental de Euler explicito es: 6.8295492469343335
- La frecuencia experimental de Euler implícito es: 6.8295492469343335
- La frecuencia experimental de Runge Kutta de segundo orden es: 6.838841150671659

Finalmente se agrega a la tabla anterior el calculo de la frecuencia analítica del método:

- La frecuencia analítica es: 6.837233244886601

De lo cual podemos concluir que ambos métodos aproximan con un error muy pequeño la frecuencia

8. Ejecutando Runge Kutta de orden dos con amortiguamiento

Se realiza una aproximación del modelo amortiguado usando Runge Kutta de orden dos y se observa su comportamiento

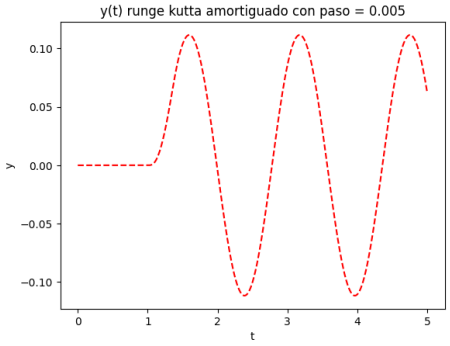

Figura 24: Resultado de aplicar Runge Kutta de orden dos al modelo amortiguado

9. Conclusiones

Los tres módulos analizados para el sistema sin amortiguación tienen un comportamiento similar. El error disminuye al disminuir el paso h. Sin embargo, el método de Runge Kutta de orden 2 es el que tiene mejor comportamiento, ya que tiene un error menor y requiere un paso menor.

Por lo tanto, el método de Runge Kutta de segundo orden es el más conveniente para el sistema sin amortiguación.

- La elección del método adecuad depende de varios factores, como la precisión requerida, el costo computacional y la facilidad de implementación. En general, los métodos de orden superior son más recisos que los de orden inferior. También requieren un paso menor, lo que puede aumentar el costo computacional
- El paso h debe elegirse cuidadosamente para obtener una precisión aceptable. Si el paso es demasiado grande, el error puede ser significativo. Si el paso es demasiado pequeño, el costo computacional puede ser excesivo.
- Es importante verificar la estabilidad de los métodos antes de usarlos. Algunos métodos pueden ser inestables para ciertos valores de los parámetros del sistema.

En el caso específico del problema del amortiguador, se pueden considerar los siguientes puntos adicionales:

- El método de Euler Implícito es más preciso que el método de Euler Explícito para un mismo paso h. Sin embargo, el método implícito puede ser más difícil de implementar, ya que requiere resolver una ecuación no lineal en cada paso.
- El método de Runge Kutta de orden 2 es un buen compromiso entre precisión y costo computacional. Es más preciso que el método de Euler Explícito y más fácil de implementar que el método implícito.

En conclusión, al analizar los módulos para el sistema sin amortiguación, se destaca que, aunque presentan comportamientos similares, el método de Runge Kutta de segundo orden exhibe un menor error y requiere pasos más reducidos, convirtiéndolo en la elección preferida. La decisión del método adecuado depende de factores como la precisión, el costo computacional y la implementación. Métodos de orden superior ofrecen mayor precisión, pero suelen requerir pasos más pequeños, aumentando el costo computacional. La elección cuidadosa del paso "h.es esencial para lograr una precisión aceptable sin aumentar innecesariamente el costo. Además, se subraya la importancia de verificar la estabilidad de los métodos antes de su aplicación. En el contexto específico del problema del amortiguador, se observa que el método de Euler Implícito, aunque más preciso que el de Euler Explícito, puede ser más complejo de implementar debido a la resolución de ecuaciones no lineales. Finalmente, el método de Runge Kutta de segundo orden se destaca como un equilibrio efectivo entre precisión y costo computacional.