FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Practico 2 - Análisis Numérico I

[75.12] Análisis Numérico I
Curso Tarela
Primer cuatrimestre de 2021

Alumno 1:	Argel, Ignacio
Número de padrón:	104351
Email:	iargel@fi.uba.ar
Alumno 2:	Bareiro, Facundo
Número de padrón:	103807
Email:	fbareiro@fi.uba.ar

Índice

1. Introducción

El presente informe tiene como objetivo principal resolver numéricamente problemas de valores iniciales mediante los métodos de Euler explícito, Euler implícito y Runge Kutta de orden 2. El objetivo principal se llevará a cabo mediante el análisis de la suspensión de un vehículo, que puede ser considerado como un sistema oscilatorio amortiguado, el cual puede modelarse mediante una ecuación diferencial de segundo grado.

2. Desarrollo

2.1. Discretización

El enunciado nos da la siguiente ecuación diferencial:

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

a cual se resuelve hallando y(t).
Sin embargo en este trabajo práctico, se hallará una aproximación numérica de y(t) a traves de distintos métodos.
Para poder hallar la solución numérica, se debe discretizar el problema.
El proceso de discretización para los problemas de valores iniciales se puede dividir en 3 pasos:

2.1.1. Discretizar el tiempo

Para ello, se debe definir:

$$t \to t^n \qquad n = 0, 1, 2, \cdots N$$
$$0 \leq t \leq 5 \qquad t^0, t^N = 5$$
$$h = \Delta t = t^{(n+1)} - t^{(n)}$$

2.1.2. Discretizar el EDO

Como en este caso se tiene una EDO de segundo orden, se deben definir dos nuevas variables:

$$\begin{cases} y(t) \cong y(t^{(n)} \cong u^{(n)}) \\ y'(t) \cong y'(t^{(n)} \cong u'^{(n)}) = v^{(n)} \end{cases}$$

Y por lo tanto quedaría un sistema de dos EDO de primer orden

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

2.1.3. Discretizar las condiciones iniciales

Condición inicial de la posición:

$$\begin{cases} y(a) = \alpha = 0 \\ t^{(0)} = a = 0 \\ u((0)) = \alpha = 0 \end{cases}$$

Y también necesitamos discretizar a la condicion inicial de la primer derivada:

$$\begin{cases} y'(a) = \beta = 0 \\ v((0)) = \beta = 0 \end{cases}$$

Cabe aclarar que alfa y beta fueron tomadas ambas como 0 a conciencia, a modo que la solución numérica tenga una mejor aproximación.

2.2. Obtencion de ecuacion Euler explicito

Para el método de Euler explícito se debe utilizar la siguiente ecuación:

$$u^{(n+1)} = u^{(n)} + hf(u^{(n)}, t^{(n)})$$

Pero como aplicamos a dos EDOs de primer orden ,tendremos:

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

Y por lo tanto las ecuaciones a utilizar serán:

$$\begin{cases} u^{(n+1)} = u^{(n)} + hv^{(n)} \\ v^{(n+1)} = v^{(n)} + h[\frac{k}{m}(c - u^{(n)}) + \frac{\lambda}{m}(c' - v^{(n)})] \end{cases}$$

Donde para el caso sin amortiguación será:

$$\begin{cases} u^{(n+1)} = u^{(n)} + hv^{(n)} \\ v^{(n+1)} = v^{(n)} + h[\frac{k}{m}(c - u^{(n)})] \end{cases}$$

2.3. Obtencion de ecuacion Euler implicito

Para este método, se utiliza la siguiente expresión:

$$u^{(n+1)} = u^{(n)} + hf(u^{(n+1),t^{n+1}}$$

Pero como aplicamos a dos EDOs de primer orden ,tendremos:

$$\begin{cases} f_1(u^{(n+1)}, v^{(n+1)}, t^{(n+1)}) = v^{(n+1)} \\ f_2(u^{(n+1)}, v^{(n+1)}, t^{(n+1)}) = \frac{k}{m}(c - u^{(n+1)}) + \frac{\lambda}{m}(c' - v^{(n+1)}) \end{cases}$$

Y por ende las ecuaciones serán:

$$\begin{cases} u^{(n+1)} = u^{(n)} + hv^{(n+1)} \\ v^{(n+1)} = v^{(n)} + h[\frac{k}{m}(c - u^{(n+1)}) + \frac{\lambda}{m}(c' - v^{(n+1)})] \end{cases}$$

Separando los pasos de cada lado de la igualdad:

$$\implies \begin{cases} u^{(n+1)} - hv^{(n+1)} = u^{(n)} \\ v^{(n+1)} - h[\frac{k}{m}(c - u^{(n+1)}) + \frac{\lambda}{m}(c' - v^{(n+1)})] = v^{(n)} \end{cases}$$

Distribuimos para poder expresar la ecuación de forma matricial:

$$\implies \begin{cases} u^{(n+1)} - hv^{(n+1)} = u^{(n)} \\ v^{(n+1)}(1 + \frac{h\lambda}{m}) + \frac{hk}{m}u^{(n+1)} - h[\frac{kc}{m} - \frac{\lambda c'}{m}] = v^{(n)} \end{cases}$$

La expresión matricial es:

$$\implies \begin{bmatrix} 1 + \frac{h\lambda}{m} & \frac{hk}{m} \\ -h & 1 \end{bmatrix} \cdot \begin{bmatrix} v^{(n+1)} \\ u^{(n+1)} \end{bmatrix} + \begin{bmatrix} h\frac{kc+\lambda c'}{m} \\ u^{(n+1)} \end{bmatrix} = \begin{bmatrix} v^{(n)} \\ u^{(n)} \end{bmatrix}$$

Invertimos la matriz:

$$\implies \begin{bmatrix} \frac{m}{h^2k+hl+m} & \frac{-hk}{h^2k+hl+m} \\ \frac{hm}{h^2k+hl+m} & \frac{hl+m}{h^2k+hl+m} \end{bmatrix}$$

La nueva expresión matricial en función de el paso actual es:

$$\begin{bmatrix} v^{(n+1)} \\ u^{(n+1)} \end{bmatrix} = \begin{bmatrix} \frac{m}{h^2k+hl+m} & \frac{-hk}{h^2k+hl+m} \\ \frac{hm}{h^2k+hl+m} & \frac{hl+m}{h^2k+hl+m} \end{bmatrix} \cdot \begin{bmatrix} v^{(n)} \\ u^{(n)} \end{bmatrix}$$

2.4. Runge Kutta orden 2

Para este método se utiliza la siguiente expresión con los pasos corrector y predictor:

$$\begin{cases} u_*^{(n+1)} = u^{(n)} + hf(u^{(n)},t^{(n)} \\ u^{(n+1)} = u^{(n)} + \frac{h}{2}[f(u^{(n)},t^{(n)} + f(u^{(n+1)},t^{(n+1)}] \end{cases}$$

Es decir deben utilizarse estos dos pasos para cada una de las dos ecuaciones diferenciales, por lo tanto tendremos 4 ecuaciones de la siguiente forma:

$$\begin{cases} u_*^{(n+1)} = u^{(n)} + hv^{(n)} \\ v_*^{(n+1)} = v^{(n)} + h\frac{k}{m}(c(t^{(n)}) - u^{(n)}) + \frac{\lambda}{m}(c'(t^{(n)}) - v^{(n)}) \\ u^{(n+1)} = u^{(n)} + \frac{h}{2}(v^{(n)} + v_*^{(n+1)}) \\ v^{(n+1)} = v^{(n)} + \frac{h}{2}(\frac{k}{m}(c(t^{(n)}) - u^{(n)}) + \frac{\lambda}{m}(c'(t^{(n)}) - v^{(n)}) + \frac{k}{m}(c(t^{(n+1)}) - u_*^{(n+1)}) + \frac{\lambda}{m}(c'(t^{(n+1)}) - v_*^{(n+1)})) \end{cases}$$

2.5. Sistema sin amortiguación

2.5.1. Obtención de ecuación analítica

A fin de comparar con los resultados numéricos, mediante la ecuación diferencial dada por el enunciado se obtuvo la solución analítica. El proceso fue una clásica resolución de ecuaciones diferenciales y se llegó a la siguiente expresión

$$y(x) = c + \alpha_2 \sin(\frac{\sqrt{k}t}{\sqrt{m}}) + \alpha_1 \sin(\frac{\sqrt{k}t}{\sqrt{m}})$$

Donde luego reemplazando por los valores iniciales, la ecuación finalmente queda:

$$y(t) = 0{,}1 - 0{,}1\cos(\omega x)$$

2.5.2. Resultados de cada uno de los métodos

Se presentarán a continuación los distintos resultados obtenidos con los distintos métodos: Cabe aclarar que en los gráficos de $y(t)$ la solución analítica se presenta en una linea roja punteada y la solución numérica en una linea azul.

Euler Explicito Gráfico sin amortiguación con $h = 0,005$:

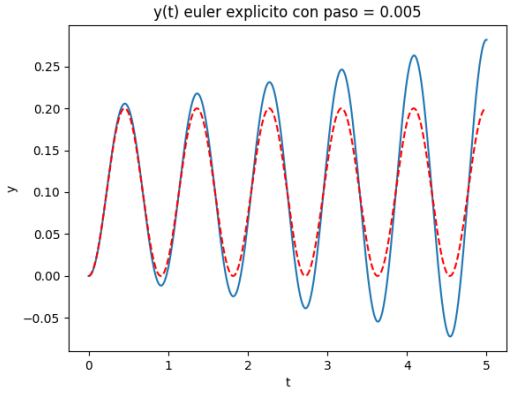

Figura 1: Grafica solucion analitica y solucion Euler explicito con $h = 0,005$.

Gráfico del error sin amortiguación con $h = 0,005$:

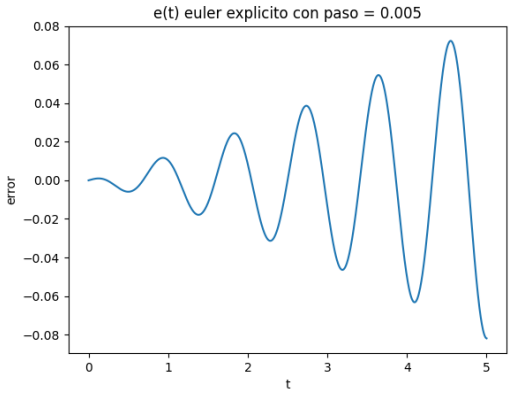

Figura 2: Grafica error $h = 0,005$.

Gráfico sin amortiguación con $h = 0,01$:

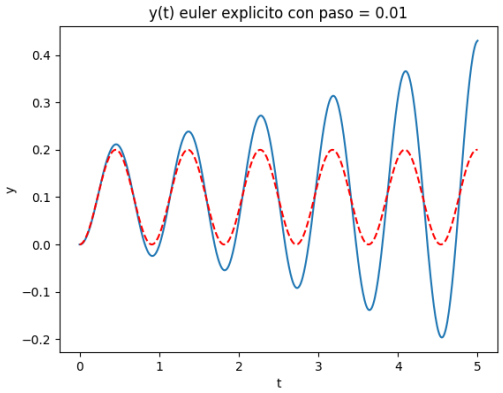

Figura 3: Grafica solucion analitica y solucion Euler explicito con $h = 0,01$.

Gráfico del error sin amortiguación con $h = 0,01$:

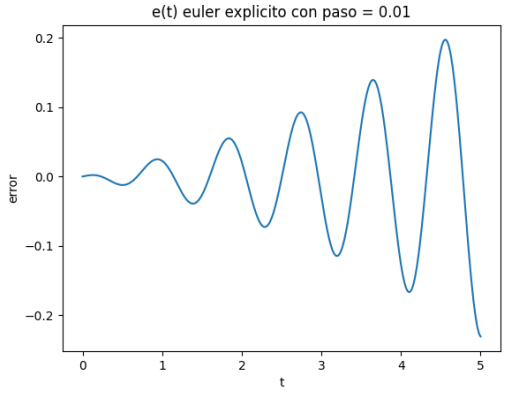

Figura 4: Grafica error $h = 0,01$.

Gráfico sin amortiguación con $h = 0{,}001$:

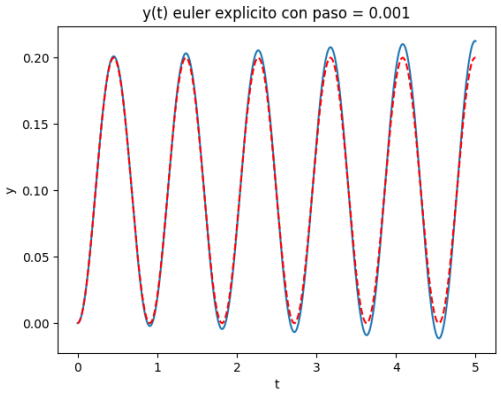

Figura 5: Grafica solucion analitica y solucion Euler explicito con $h = 0,001$.

Gráfico del error sin amortiguación con $h = 0,001$:

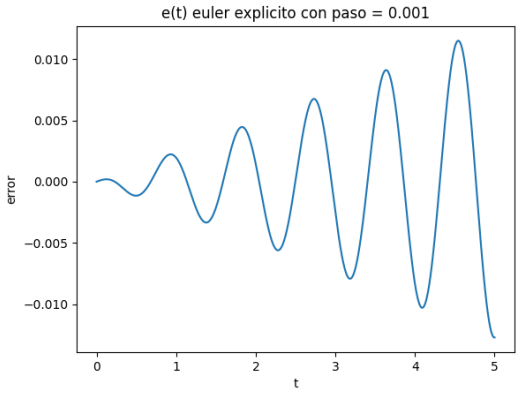

Figura 6: Grafica error $h = 0,001$.

Gráfico sin amortiguación con $h = 0,0005$:

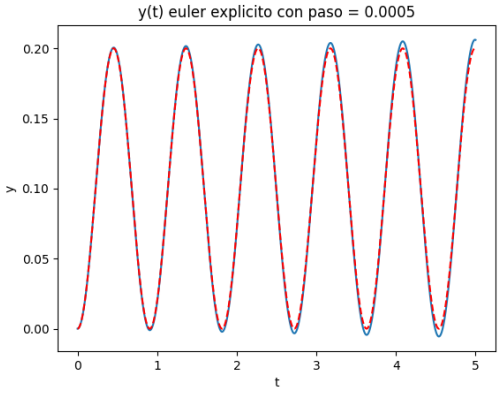

Figura 7: Grafica solucion analitica y solucion Euler explicito con $h = 0,0005$.

Gráfico del error sin amortiguación con $h = 0,0005$:

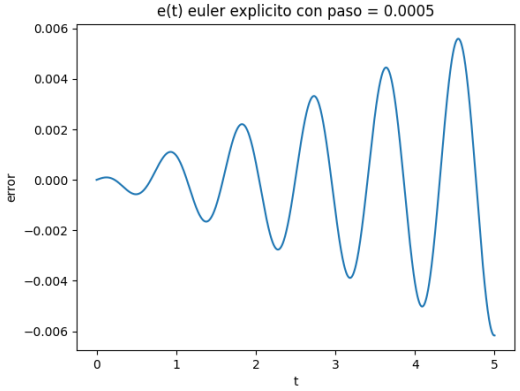

Figura 8: Grafica error $h = 0,0005$.

Euler Implicito Gráfico sin amortiguación con $h = 0,005$:

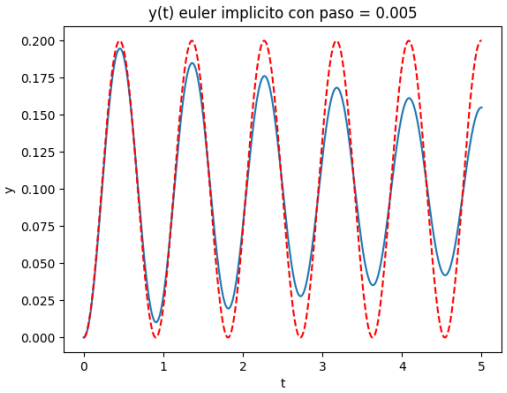

Figura 9: Grafica solucion analitica y solucion Euler explicito con $h = 0,005$.

Gráfico del error sin amortiguación con $h = 0,005$:

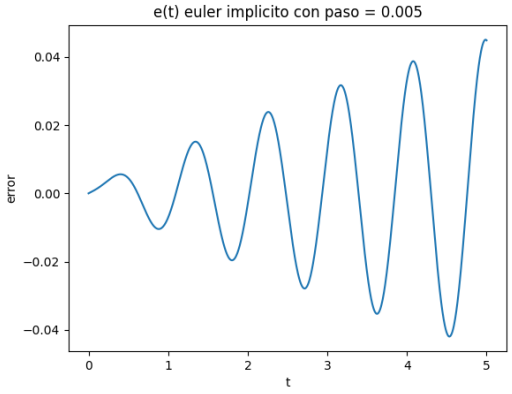

Figura 10: Grafica error $h = 0,005$.

Gráfico sin amortiguación con $h = 0,01$:

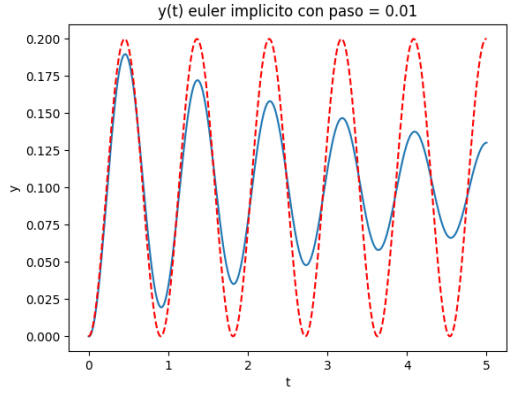

Figura 11: Grafica solucion analitica y solucion Euler explicito con $h = 0,01$.

Gráfico del error sin amortiguación con $h = 0,01$:

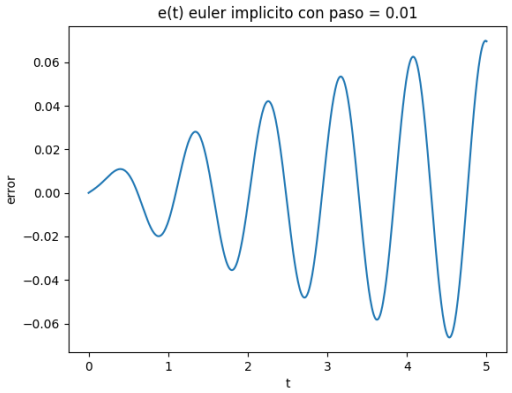

Figura 12: Grafica error $h = 0,01$.

Gráfico sin amortiguación con $h = 0,001$:

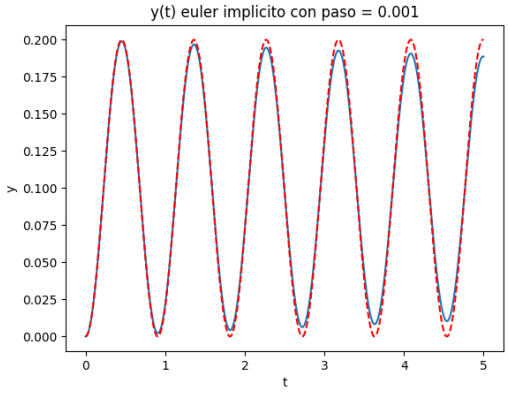

Figura 13: Grafica solucion analitica y solucion Euler explicito con $h = 0,001$.

Gráfico del error sin amortiguación con $h = 0,001$:

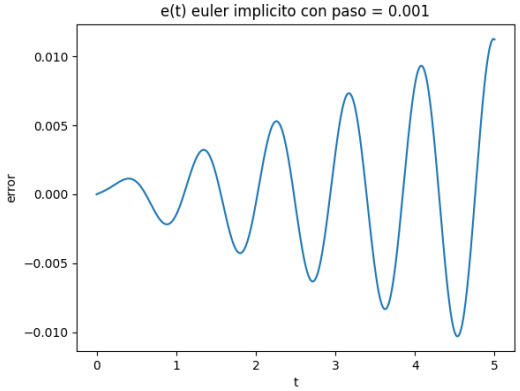

Figura 14: Grafica error $h = 0,001$.

Gráfico sin amortiguación con $h = 0,0005$:

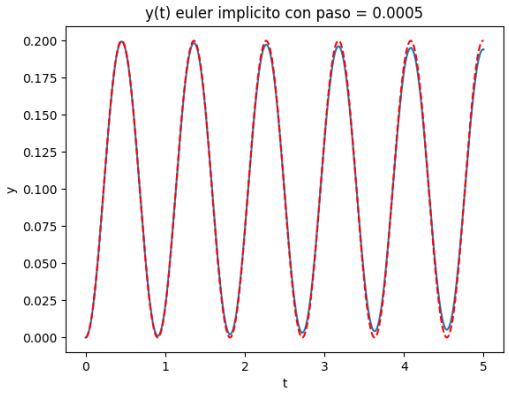

Figura 15: Grafica solucion analitica y solucion Euler explicito con $h = 0,0005$.

Gráfico del error sin amortiguación con $h = 0,0005$:

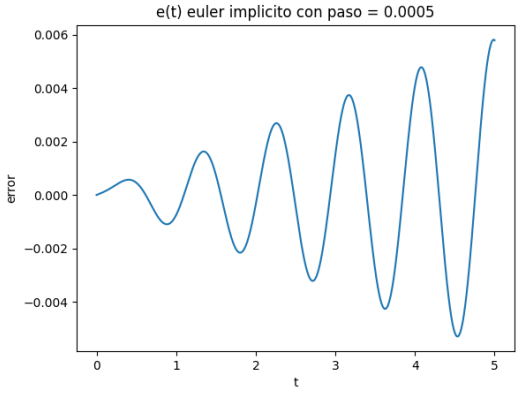

Figura 16: Grafica error $h = 0,0005$.

Runge Kutta orden 2 Gráfico sin amortiguación con $h = 0,005$:

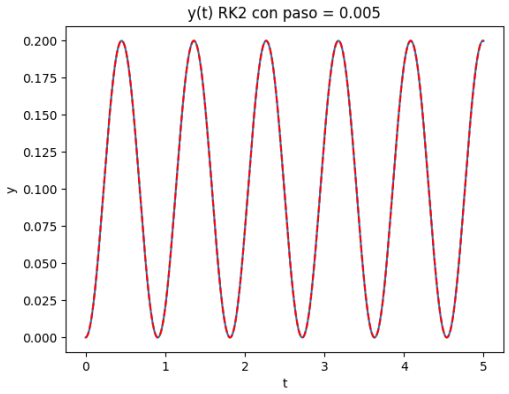

Figura 17: Grafica solucion analitica y solucion Euler explicito con $h = 0,005$.

Gráfico del error sin amortiguación con $h = 0,005$:

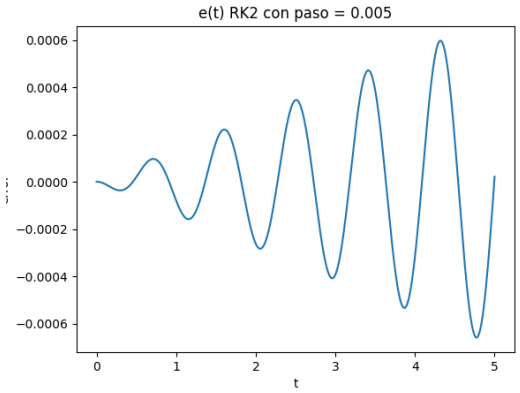

Figura 18: Grafica error $h = 0,005$.

Gráfico sin amortiguación con $h = 0,01$:

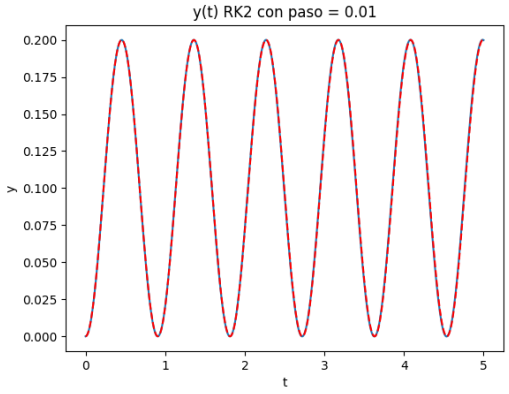

Figura 19: Grafica solucion analitica y solucion Euler explicito con $h = 0,01$.

Gráfico del error sin amortiguación con $h = 0,01$:

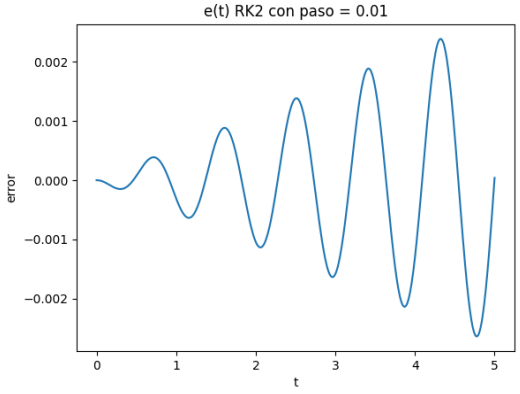

Figura 20: Grafica error $h = 0,01$.

Gráfico sin amortiguación con $h = 0,001$:

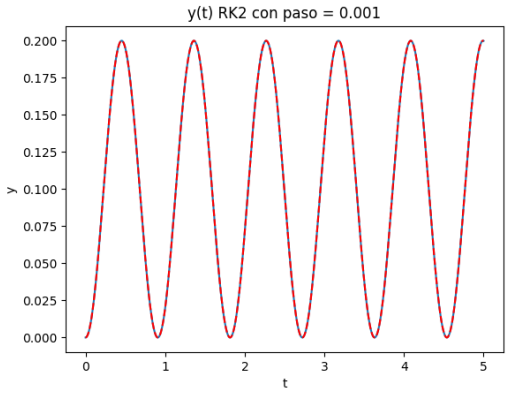

Figura 21: Grafica solucion analitica y solucion Euler explicito con $h = 0,001$.

Gráfico del error sin amortiguación con $h = 0,001$:

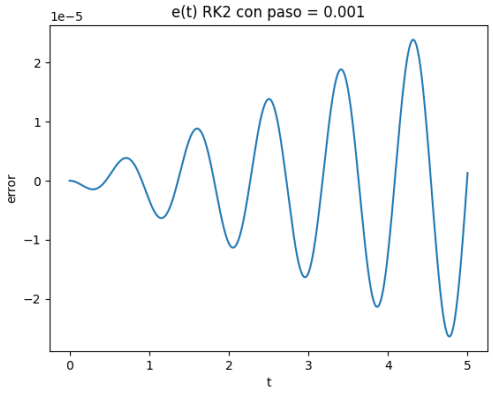

Figura 22: Grafica error $h = 0,001$.

Gráfico sin amortiguación con $h = 0,0005$:

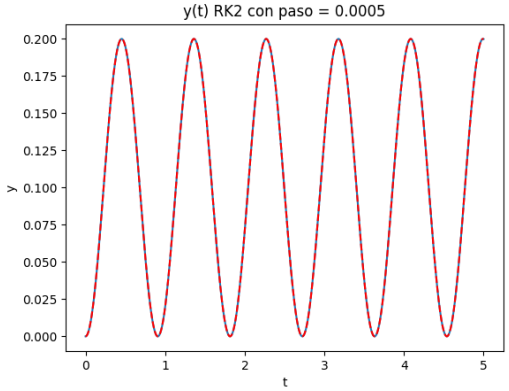

Figura 23: Grafica solucion analitica y solucion Euler explicito con $h = 0,0005$.

Gráfico del error sin amortiguación con $h = 0,0005$:

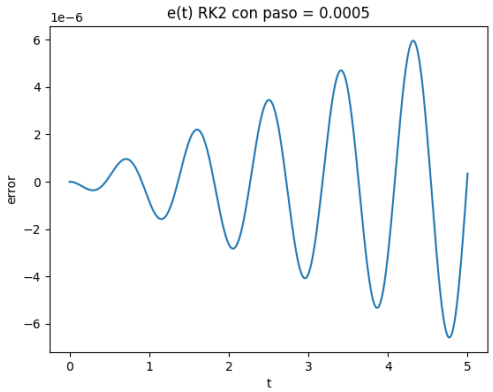

Figura 24: Grafica error $h = 0,0005$.

2.5.3. Obtención de c'

Para la obtencion de la derivada de c se dividió al perfil del terreno en distintos tramos de tiempo y se utilizó la siguiente fórmula

$$c' = \frac{\Delta c}{\Delta t}$$

Y por ende c' en funcion del tiempo quedó como:

$$c' = \begin{cases} 0 & si \quad 0 \leq c \leq 1 \\ 1 & si \quad 1 \leq c \leq 1{,}1 \\ 0 & si \quad 1{,}1 \leq c \leq 1{,}3 \\ -1 & si \quad 1{,}3 \leq c \leq 1{,}4 \\ 0 & si \quad 1{,}4 \leq c \leq 5 \end{cases}$$

2.5.4. Resultados con los datos propuestos

El sistema amortiguado se eligió resolverlo mediante el método de Runge Kutta de orden 2, por razones que serán explicadas en la conclusión del presente informe.
El gráfico utilizando los datos propuestos por el enunciado y con un $h = 0,005$ es el siguiente:

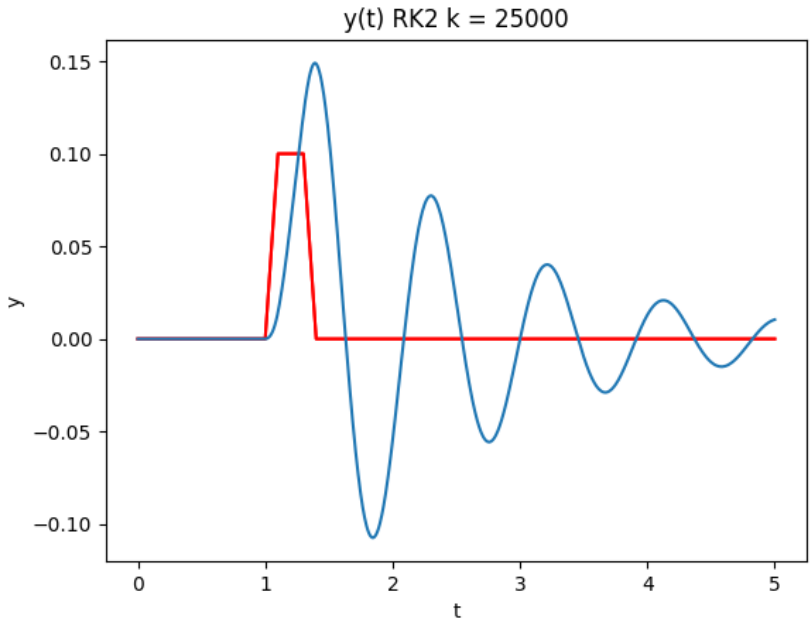

Figura 25: Grafica del sisetma amortiguado con $h = 0,005$.

Cabe aclarar que se eligió h=0.005 y no h=0,01 para poder tener aún más precisión en los resultados

2.5.5. Busqueda del k y λ optimo

Como primer método se intentó buscar el k y λ óptimos por separados, haciendo variar uno y el otro dejarlo constante. El objetivo es primero dejar a λ constante y encontrar el valor de k óptimo y luego a ese k óptimo dejarlo constante y variar λ para encontrar el λ óptimo.
Para un mismo λ las máximas compresiones al variar k fueron:

- Máxima compresión $k = 2500$ $\lambda = 750$: -0.09207190446515004

- Máxima compresión $k = 5000$ $\lambda = 750$: -0.09133767691851619
- Máxima compresión $k = 10000$ $\lambda = 750$: -0.08987956118827894
- Máxima compresión $k = 15000$ $\lambda = 750$: -0.088435156706651
- Máxima compresión $k = 25000$ $\lambda = 750$: -0.1073762346245694
- Máxima compresión $k = 40000$ $\lambda = 750$: -0.13021462438185616

Para $k = 15000$ (que fue el valor con el que se obtuvo la menor compresión en módulo) y para distintos valores de λ:

- Máxima compresión $k = 15000$ $\lambda = 500$: -0.09616102721381309
- Máxima compresión $k = 15000$ $\lambda = 750$: -0.088435156706651
- Máxima compresión $k = 15000$ $\lambda = 1000$: -0.08629555422086246
- Máxima compresión $k = 15000$ $\lambda = 1250$: -0.08422388971220598
- Máxima compresión $k = 15000$ $\lambda = 1500$: -0.08221773782697116
- Máxima compresión $k = 15000$ $\lambda = 1750$: -0.08027476511063744
- Máxima compresión $k = 15000$ $\lambda = 2000$: -0.07839272640715833

Ahora, pruebo lo mismo con k=25000 para ver si el método es confiable:

- Máxima compresión $k = 25000$ $\lambda = 500$: -0.12406089058822958
- Máxima compresión $k = 25000$ $\lambda = 750$: -0.1073762346245694
- Máxima compresión $k = 25000$ $\lambda = 1000$: -0.09348699399725394
- Máxima compresión $k = 25000$ $\lambda = 1250$: -0.08184492475436916
- Máxima compresión $k = 25000$ $\lambda = 1500$: -0.07957158883761194
- Máxima compresión $k = 25000$ $\lambda = 1750$: -0.0776921144948735
- Máxima compresión $k = 25000$ $\lambda = 2000$: -0.07587177490575885

Como segundo método se propuso que la relación entre los valores de k y λ proporcionados por el enunciado deba mantenerse como una relación estándar para los k y λ que se busquen a partir de ahora, teniendo en cuenta que estos componentes (muelle y amortiguador) con estos valores tienen una calidad equiparable.
Por lo tanto lo que se hizo fue para valores de λ entre 150 y 12000, avanzando con un paso de 50 y para valores de k entre 2500 y 100000 avanzando con un paso de 1000, se utilizó un algoritmo que probó mediante fuerza bruta cada combinación entre estos valores, encontrando la más cercana a los valores iniciales de manera proporcional hasta encontrar la compresión buscada.
Además uno de los objetivos de este segundo método fue no tener que requerir al mejor muelle y al mejor amortiguador para poder alcanzar las condiciones requeridas.
Los valores obtenidos son $k = 80500$ y $\lambda = 4550$. El gráfico para estos valores es el siguiente:

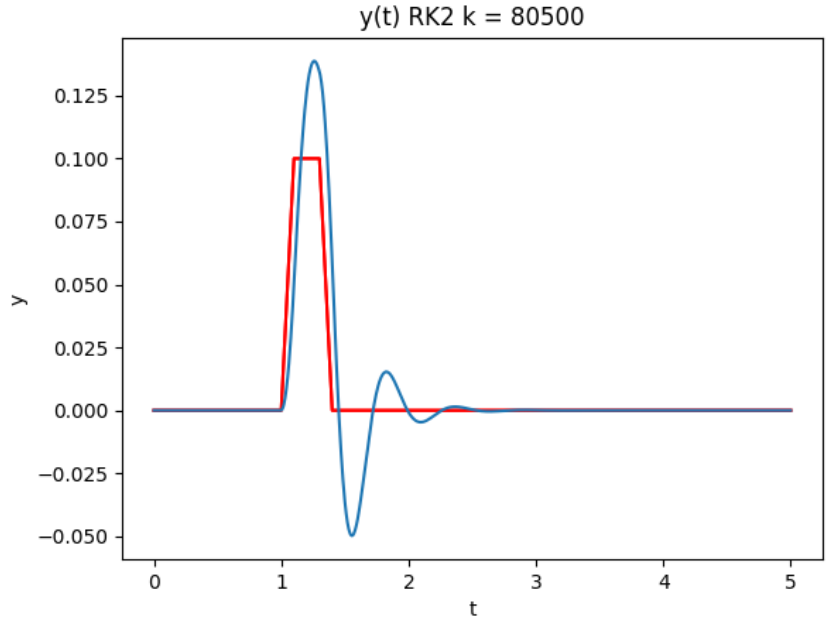

Figura 26: Grafica del sisetma amortiguado con $h = 0{,}005$ y tope de compresión 0,05.

Y el gráfico de la aceleración vertical y'' en función del tiempo para estos mismos valores también es:

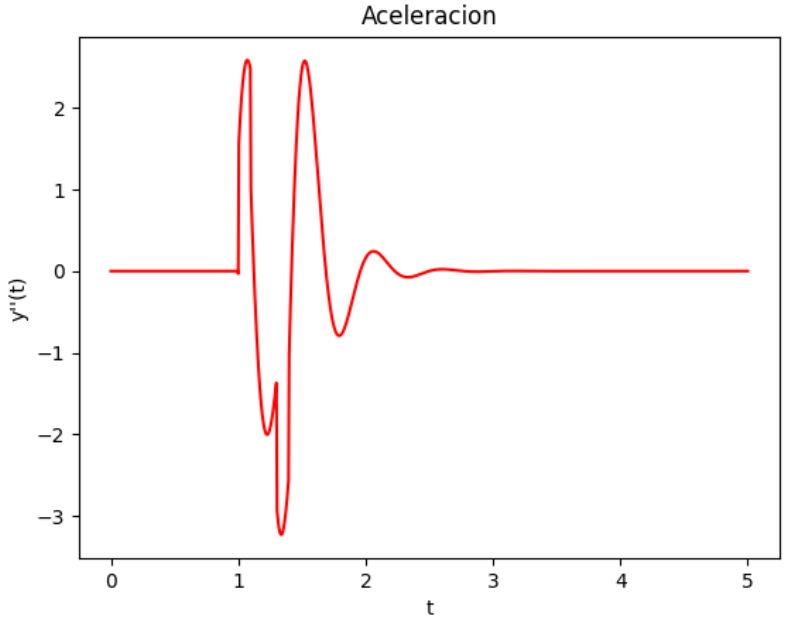

Figura 27: Grafica de la aceleración disminuyendo.

Donde se tienen todos los datos ya que y e y' fueron calculados en cada uno de los métodos porqué $y = u$ e $y' = v$.

3. Conclusión

3.1. Análisis de cada uno de los métodos para el sistema sin amortiguación

Al observar los resultados que arrojan los gráficos de la sección 2.5 podemos llegar a varias conclusiones. Antes de arrancar el análisis cabe aclarar que el error en todos los gráficos parece aumentar significativamente, sin embargo lo que hay que observar de los mismos es la escala que muestran, porque además como se toma a 0 como valor inicial, es inevitable que el error arranque en 0 y que luego por cuestiones matemáticas aumente.
En principio para Euler explícito vemos que para el $h = 0{,}005$ propuesto por el enunciado, el resultado no aproxima bien a la solución ya que la gráfica de $y(t)$ no es aproximada a la función analítica y tampoco el gráfico del error muestra valores chicos en la escala porque muestra una escala que llega hasta 0,08 el cual no es un error despreciable. Al iniciar la búsqueda del h que permite aproximarse bien a la solución analítica para este método, aunque pareciera algo obvio, no está de más comprobar que un paso mayor al inicialmente propuesto diera un error mayor y así lo fue para $h = 0{,}01$. Ahora si, al disminuir el paso a $h = 0{,}001$ notamos una solución más acorde a la analítica y un error menor ya que ahora la escala del gráfico del error muestra valores menores; sin embargo decidimos seguir disminuyendo el paso una vez más y fue así que para $h = 0{,}0005$ se encontró una escala del error en valores que llegan hasta 0,006 aproximadamente, lo que indica que la aproximación es buena y que este último es el h óptimo.
Pasando al método implícito las observaciones son las mismas que para el explícito, es decir para $h >= 0{,}005$ el error no es admisible y la representación no es adecuada; recién en $h = 0{,}001$ obtenemos resultados aceptables y para $h = 0{,}0005$ se obtiene el h óptimo con un módulo del error máximo que nuevamente ronda los 0,006.
Para Runge Kutta de orden 2 los resultados si fueron notablemente mejores a los últimos dos métodos y era lo esperable ya que el orden de estos era 1 y en RK es de orden 2. Podemos ver que para $h = 0{,}01$, $y(t)$ aproxima perfectamente a la solución teórica y el módulo del error máximo es aproximado a 0,002, es decir un error aún menor al obtenido para los otros dos métodos en el paso $h = 0{,}0005$. Esto último indica una diferencia abismal entre métodos. Por otra parte se podría concluir que $h = 0{,}01$ es el h óptimo para este método, porque si seguimos achicando el paso, el error si va a seguir disminuyendo pero el costo computacional será mayor y sin sentido alguno ya que con $h = 0{,}01$ alcanza y sobra.

3.1.1. Corroboración de la frecuencia experimental

Para corroborar que la frecuencia experimental se corresponde con su expresión analítica basta con mirar los gráficos de cada método en lo que se concluyó que era su h óptimo (en la sección 2.5) y compararlo con la expresión analítica otorgada por el enunciado. En primer lugar al mirar el gráfico se debe averiguar el período T, el cual suponemos como estamos en el h óptimo es el mismo para los 3 métodos y es aproximadamente $T = 0{,}9077$. Ahora utilizamos la fórmula:

$$\omega_{exp} = \frac{2\pi}{T} = 6{,}922$$

Y luego dividimos el w experimental con el w teórico:

$$\frac{\omega_{exp}}{\omega_{teo}} = \omega_{exp}/(\sqrt{\frac{k}{m}}) = 0{,}997$$

Este último resultado nos indica que la relación entre los w es aproximadamente de 1, osea que son casi el mismo valor, por ende se puede corroborar que el ω experimental coincide con el analítico.

3.1.2. Estimación experimental del orden de cada método

Para estimar el orden de los métodos de manera experimental lo que se planteó fue reducir el paso $h = 0{,}01$ a la mitad (osea a h=0,005) y analizar cómo se comporta el error. En teoría para el Euler explícito y para el implícito debería reducirse el error a la mitad porque estamos en presencia de métodos de orden 1 y para el Runge Kutta de orden 2 que como su nombre lo indica, es de orden 2, el error debería reducirse a la cuarta parte. Para poder ver los valores de los errores se tomó el mayor valor (en módulo) que aparece en la escala de los gráficos de la sección 2.5. Luego para ver la relación entre los errores se los dividió y se comparó el resultado con el que se supone que deberían dar. Los resultados fueron:

- EXPLICITO:
  $h = 0{,}01$ $e_1 = 0{,}2$
  $h = 0{,}005$ $e_2 = 0{,}08$
  $\rightarrow \frac{e_2}{e_1} = 0{,}4$
- IMPLICITO:
  $h = 0{,}01$ $e_1 = 0{,}06$
  $h = 0{,}005$ $e_2 = 0{,}04$
  $\rightarrow \frac{e_2}{e_1} = 0{,}66$
- RK:
  $h = 0{,}01$ $e_1 = 0{,}002$
  $h = 0{,}005$ $e_2 = 0{,}0006$
  $\rightarrow \frac{e_2}{e_1} = 0,3$

Para los dos primeros métodos, vemos que su valor es cercano a 0,5 y por lo tanto se puede decir que aproximadamente el error se disminuye a la mitad y que por lo tanto son de orden 1. Para el último método, vemos que el resultado de la división es cercano a 0,25 , por lo tanto podemos afirmar que el error se reduce a la cuarta parte y que por lo tanto el método es de orden 2.

Cabe aclarar que tomar el mayor valor (en módulo) que aparece en la escala de los gráficos quizás no es el método más preciso pero sí nos alcanza para poder obtener una buena aproximación del orden.

3.1.3. Elección del método más conveniente

Para poder avanzar en el informe no fue una difícil decisión elegir el método de Runge Kutta de orden 2 ya que tiene muchas ventajas por sobre los otros 2. En primer lugar la deducción de las fórmulas a utilizar es de la complejidad del método explícito, por lo que no lleva mucho análisis, es simplemente reemplazar; en cambio la deducción de las ecuaciones para el implícito si nos llevó mucho más trabajo, como se expresa en la sección 2.3. A su vez, la fácil deducción de las fórmulas se traduce luego en la complejidad de la programación, la cual será menor al ser más fácil la deducción; por lo tanto aqui se muestra una segunda ventaja de Runge Kutta frente al implícito. Las ventajas de RK que tiene frente al explícito también están a la vista ya que a mayor orden y para un mismo h, el error de truncamiento es menor. Además pudimos ver experimentalmente en la sección 2.5 que RK para llegar a un error del 0,002 sólo le bastó con un $h = 0{,}01$ mientras que los otros 2 métodos para llegar a un error del orden de 0,006 (es decir nisiquiera menor a 0,002) necesitaron un paso de $h = 0{,}0005$, es decir significativamente menor. Esto se traduce también en una gran reducción del costo computacional al usar Runge Kutta. Es por estas razones que se decide avanzar al sistema amortiguado con el método de Runge Kutta

3.1.4. k y λ óptimos

En la sección 2.5.5 se intentó buscar el k y λ óptimos mediante 2 métodos distintos. el primero quizás más intuitivo que el segundo. En esta sección se pasará a justificar porque el segundo

método es mejor que el primero. Hablando del primer método, lo primero que se hizo fue dejar λ constante y hacer variar a k, lo que podemos ver con estos resultados fue que el k no modifica sustancialmente las compresiones y que tampoco se ha se ha encontrado una tendencia la cual nos diga que al aumentar o disminuir k (con λ constante) se mejore o empeore la compresión. Sin embargo se ha podido encontar que el menor valor de la compresión fue el de $k = 15000$ asi que el k óptimo se supone que siguiendo esta lógica debería encontrarse por estos órdenes de magnitud. Siguiendo esta última lógica de que el k óptimo debe ser cercano a 15000, se procedió a dejar este valor constante y a variar el λ para encontrar el supuesto λ óptimo. una vez que se hizo esto, los supuestos k óptimos y λ óptimos son 15000 y 2000. Además estos últimos resultados nos muestran que para un mismo k, para obtener la menor compresión necesito aumentar los valores de λ. Todo esto debería parecernos en un principio extraño, porque siguiendo la lógica para poder cumplir con las condiciones solo haría falta tener un muy buen amortiguador y un muelle de una calidad media. Es por eso que para probar que este método tenga sentido, probamos hacer variar el λ pero esta vez con un $k = 25000$, y los resultados no coincidieron con la lógica de este primer método, porque para los λ de 1250 a 2000, el k que se supone que debería ser el óptimo da una máxima compresión mayor (en módulo) y esto entonces invalida nuestro primer método y por lo tanto no es conveniente hallar el k y λ óptimos por este camino. Ahora pasando al segundo método, la realización del mismo se justifica por los resultados obtenidos en el primero, los cuales nos indican que en primer lugar no hay una tendencia si aumentamos o disminuimos k (y dejamos λ constante) en relación con la máxima compresión; en segundo lugar que si dejamos k constante y vamos aumentando λ (es decir dejando de lado la proporción entre ambos dada por el enunciado) logramos obtener una compresión menor a 0,005 pero al fin y al cabo estaríamos necesitando un amortiguador de una muy buena calidad y un muelle que no tenga la misma calidad. Por último cabe aclarar como se dijo en 2.5.5 que la idea de utilizar este método es no tener que requerir al mejor muelle y al mejor amortiguador para poder alcanzar las condiciones requeridas. Los resultados de k y λ óptimos son $k = 80500$ y $\lambda = 4550$ y comprobamos con estos valores como en el gráfico de la aceleración, la misma disminuye a medida que se avanza en el tiempo y también en el anterior gráfico que las oscilaciones se atenúan y que el sistema de amortiguación no se comprime más de 0.05 m.

4. Anexo

Adjuntamos solo las funciones que calculan cada método, no así sus sucesivos llamados e impresiones de gráficos ya que sería innecesario en este contexto y agregaria muchísimo tamaño a este documento.

```
'Euler Explicito'

def metodoDeEulerExplicitoDerivada2(f,intervalo, paso):
    cantidad = intervalo/paso +1
    u=np.zeros(int(cantidad),dtype=float)
    v=np.zeros(int(cantidad),dtype=float)
    u[0]=0
    v[0]=0
    i=0
    n=0
    while n<cantidad-1:
        u_actual = (u[(n)] + paso * v[n])
        u[(n+1)]= u_actual
        v_actual = v[n] + paso * f(u[n])
        v[n+1] = v_actual
        n+=1
        i+=paso
    return u

'Euler Implicito'

def metodoDeEulerImplicitoDerivada2(A_menos1,termino_indep,intervalo, paso):
    cantidad = intervalo/paso +1
    res = np.zeros(int(cantidad*2))
    res.shape=(2,int(cantidad))
    i=0
    n=0
    while n<cantidad-1:
        aux = np.array(A_menos1 @ res[:,n] + termino_indep)
        res[:,(n+1)]= aux
        n+=1
        i+=paso
    return res[1]

'Runge Kutta'

def rungeKuttaO2(f,intervalo, paso):
    cantidad = intervalo/paso +1
    u=np.zeros(int(cantidad),dtype=float)
    v=np.zeros(int(cantidad),dtype=float)
    u[0]=0
    v[0]=0
    i=0
    n=0
    while n<cantidad-1:
        u_actual_predictor = (u[(n)] + paso * v[n])
        v_actual_predictor = (v[n] + paso * f(u[n]))
        u[(n+1)]= (u[n] + (paso/2) *(v[n] + v_actual_predictor) )
        v[n+1] = (v[n] + (paso/2) * (f(u[n]) + f(u_actual_predictor)) )
        n+=1
        i+=paso
    return u

def rungeKuttaO2ConExtras(f,intervalo, paso,extras):
    cantidad = intervalo/paso +1
    u=np.zeros(int(cantidad),dtype=float)
    v=np.zeros(int(cantidad),dtype=float)
    u[0]=0
    v[0]=0
    i=0
    n=0
```

```
62          while n<cantidad-1:
63              u_actual_predictor = (u[(n)] + paso * v[n])
64              v_actual_predictor = (v[n] + paso * f(u[n],extras[0],i,v[n],extras[1]))
65              u[(n+1)]= (u[n] + (paso/2) *(v[n] + v_actual_predictor) )
66              v[n+1] = (v[n] + (paso/2) * (f(u[n],extras[0],i,v[n],extras[1]) + f(
                    u_actual_predictor,extras[0],i+paso,v_actual_predictor,extras[1])))
67              n+=1
68              i+=paso
69          return u,v
70
71      def maximaCompresion(intervalo, h,extras,c,t,k,lam):
72          def f2(y,c,t,y_prim,c_prim):
73              m = 104351/200
74              res =  (k/m)*(c(t)-y) + (lam/m) * (c_prim(t) - y_prim)
75              return res
76
77          paso = 0
78          aproximada,v = rungeKuttaO2ConExtras(f2,intervalo,h,extras)
79          minimo = 0
80          for j in range (int((intervalo/h)+1)):
81              aproximada[j] = (aproximada[j] - float(c(paso)))
82              if aproximada[j] < minimo :
83                  minimo = aproximada[j]
84              paso += h
85          return minimo
```