Modelado y Simulación de Sistemas Dinámicos

Introducción a los Métodos de Integración Numérica de Ecuaciones Diferenciales Ordinarias

1. Introducción

Ernesto Kofman

La gran mayoría de los modelos que se utilizan en las distintas ramas de la Ingeniería tienen como característica distintiva que las variables evolucionan continuamente en el tiempo. Por tal motivo, dichos modelos son habitualmente categorizados como *Sistemas Continuos*.

Entre los Sistemas Continuos encontramos principalmente los modelos que provienen de los distintos dominios de la Física: sistemas mecánicos, termodinámicos, electromagnéticos, hidráulicos, etc. Cuando estos sistemas tienen *parámetros concentrados*, las relaciones matemáticas que vinculan la evolución de las distintas variables conducen a *Sistemas de Ecuaciones Diferenciales Ordinarias*.

Por ejemplo, un sistema-masa–resorte–amortiguamiento muy simple como el de la Figura 1 puede representarse por el siguiente sistema de ecuaciones:

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= \frac{1}{m}[-k\,x_1(t) - b\,x_2(t) + F(t)] \end{aligned} \tag{1}$$

donde la variable $x_1(t)$ representa la posición, $x_2(t)$ representa la velocidad y $F(t)$ es la fuerza de entrada aplicada al sistema. Los parámetros son la masa m, el coeficiente de roce b y la constante del resorte k.

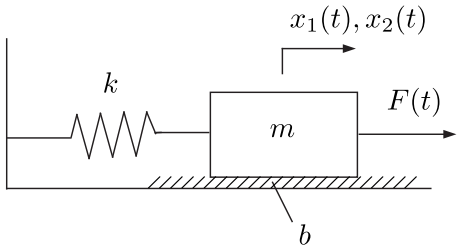

Figura 1: Sistema Masa Resorte

Si queremos determinar como evolucionarán las distintas variables de este sistema a medida que pasa el tiempo, deberemos resolver la Ecuación (1). En este caso, por ejemplo, tomando parámetros $m = b = k = 1$, suponiendo que la entrada es constante $F(t) = 1$, y asumiendo que $x_1(0) = x_2(0) = 0$, obtenemos la siguiente solución:

$$\begin{aligned} x_1(t) &= 1 - \frac{\sqrt{3}}{3}e^{-t/2}\sin\frac{\sqrt{3}}{2}t - e^{-t/2}\cos\frac{\sqrt{3}}{2}t \\ x_2(t) &= \frac{\sqrt{12}}{3}e^{-t/2}\sin\frac{\sqrt{3}}{2}t \end{aligned} \tag{2}$$

para todo $t \geq 0$. La gráfica de esta solución puede verse en la Figura 2

Si bien en este caso fue posible resolver de manera *analítica* la ecuación diferencial y obtener así la evolución de las variables en el tiempo, en general esto no será posible. En presencia de no linealidades, excepto casos muy simples y triviales, no es posible obtener una expresión analítica para la solución de una Ecuación Diferencial Ordinaria.

En los casos lineales, aún siendo posible resolver las ecuaciones (usando la Transformada de Laplace, por ejemplo), es muchas veces engorroso y poco práctico debido a la complejidad de los cálculos y de las expresiones resultantes.

Por todos estos motivos, generalmente se recurre a ditintos métodos que bajo ciertas condiciones permiten obtener soluciones aproximadas, normalmente mediante el uso de computadoras. Estos algoritmos, denominados *métodos de integración numérica* para ecuaciones diferenciales ordinarias constituyen la herramienta básica fundamental para la *Simulación de Sistemas Continuos*.

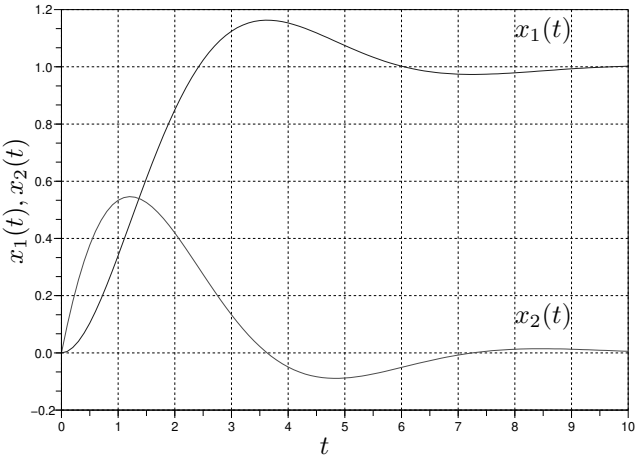

Figura 2: Solución de la Ecuación (1) del Sistema Masa Resorte

2. Principios Básicos de la Aproximación Numérica

En general, nos interesará obtener las trayectorias de un sistema de ecuaciones de la forma

$$\begin{aligned}
\dot{x}_1 &= f_1(x_1, x_2, \ldots, x_n, u_1, \cdots, u_m, t) \\
\dot{x}_2 &= f_2(x_1, x_2, \ldots, x_n, u_1, \cdots, u_m, t) \\
&\vdots \\
\dot{x}_n &= f_n(x_1, x_2, \ldots, x_n, u_1, \cdots, u_m, t)
\end{aligned}$$

donde las x_i son las variables de estado y $u_i(t)$ son las entradas. Este sistema de ecuaciones puede reescribirse en forma vectorial como sigue:

$$\dot{\mathbf{x}}(t) = \hat{\mathbf{f}}(\mathbf{x}(t), \mathbf{u}(t), t)$$

donde definimos $\mathbf{x}(t) \triangleq [x_1, x_2, \ldots, x_n]^T$, $\mathbf{u}(t) \triangleq [u_1, \cdots, u_m]^T$ y $\hat{\mathbf{f}}(\cdot) \triangleq [f_1, \cdots, f_n]^T$.

Para poder calcular una solución numérica, será necesario siempre conocer completamente el vector de trayectorias de entrada $\mathbf{u}(t)$. Asumiendo entonces que este vector es conocido, no tiene sentido expresarlo como una variable. Por este motivo, en la literatura sobre métodos de integración numérica se define

$$\mathbf{f}(\mathbf{x}(t), t) \triangleq \hat{\mathbf{f}}(\mathbf{x}(t), \mathbf{u}(t), t)$$

de manera que el efecto de las entradas queda contenido en la propia definición de la función $\mathbf{f}$.

En virtud de esto, el objetivo de los métodos de integración numérica será encontrar una solución aproximada del sistema de ecuaciones diferenciales:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t) \tag{3}$$

a partir de la condición inicial conocida

$$\mathbf{x}(t_0) = \mathbf{x}_0 \tag{4}$$

2.1. Métodos de Euler

El método más sencillo para resolver la Ec.(3) fue propuesto por Leonhard Euler en 1768. Consiste básicamente en aproximar la derivada por el cociente incremental en la ecuación mencionada:

$$\dot{\mathbf{x}}(t) \approx \frac{\mathbf{x}(t+h) - \mathbf{x}(t)}{h} \approx \mathbf{f}(\mathbf{x}(t), t)$$

de donde despejamos

$$\mathbf{x}(t+h) \approx \mathbf{x}(t) + h\ \mathbf{f}(\mathbf{x}(t), t)$$

Llamando entonces $t_k = t_0 + k\ h$ para $k = 0, 1, \cdots$ y definiendo $\mathbf{x}_k \triangleq \mathbf{x}(t_k)$, resulta la fórmula de *Forward Euler*:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + h\ \mathbf{f}(\mathbf{x}_k, t_k) \tag{5}$$

El parámetro h, que define la distancia entre dos instantes de tiempo donde calculamos la solución, se denomina *paso de integración*. Más adelante veremos que h podrá ser constante o variable según el caso, y estudiaremos formas de elegir su valor adecuado.

El algoritmo de Forward Euler es entonces trivial: dado $\mathbf{x}_0$ y elegido un valor de h, hacemos $k = 0$ y utilizamos la fórmula anterior para calcular $\mathbf{x}_1$. Luego, a partir de este valor de $\mathbf{x}_1$ calculamos $\mathbf{x}_2$ y así sucesivamente.

El algoritmo terminará cuando lleguemos al paso en que $t_k = t_f$, donde el parámetro t_f se denomina *tiempo final* de la simulación. También veremos más adelante como elegir el tiempo final.

Para el sistema masa resorte de la Ec.(1), el método de Forward Euler con paso $h = 0.1$ comenzaría con los siguientes cálculos:

$$\mathbf{x}_0 = \begin{bmatrix}0\\0\end{bmatrix} \Rightarrow \mathbf{f}(\mathbf{x}_0, t_0) = \begin{bmatrix}x_2 = 0\\1 - x_1 - x_2 = 1\end{bmatrix} \Rightarrow \mathbf{x}_1 = \mathbf{x}_0 + h\ \mathbf{f}(\mathbf{x}_0, t_0) = \begin{bmatrix}0 + 0.1 \cdot 0\\0 + 0.1 \cdot 1\end{bmatrix} = \begin{bmatrix}0\\0.1\end{bmatrix}$$

$$\mathbf{x}_1 = \begin{bmatrix}0\\1\end{bmatrix} \Rightarrow \mathbf{f}(\mathbf{x}_1, t_1) = \begin{bmatrix}x_2 = 0.1\\1 - x_1 - x_2 = 1 - 0 - 0.1 = 0.9\end{bmatrix} \Rightarrow \mathbf{x}_2 = \mathbf{x}_1 + h\ \mathbf{f}(\mathbf{x}_1, t_1) = \begin{bmatrix}0 + 0.1 \cdot 0.1\\0.1 + 0.1 \cdot 0.9\end{bmatrix} = \begin{bmatrix}0.01\\0.19\end{bmatrix}$$

Esto es, de acuerdo al método de Forward Euler, $\mathbf{x}(t_1 = 0.1) \approx [0\ 0.1]^T$ y $\mathbf{x}(t_2 = 0.2) \approx [0.01\ 0.19]^T$.

La Figura 3 muestra el resultado de continuar calculando la solución de la Ec.(1) con este método numérico usando un paso de integración $h = 0.1$ y compara dicha solución con la analítica.

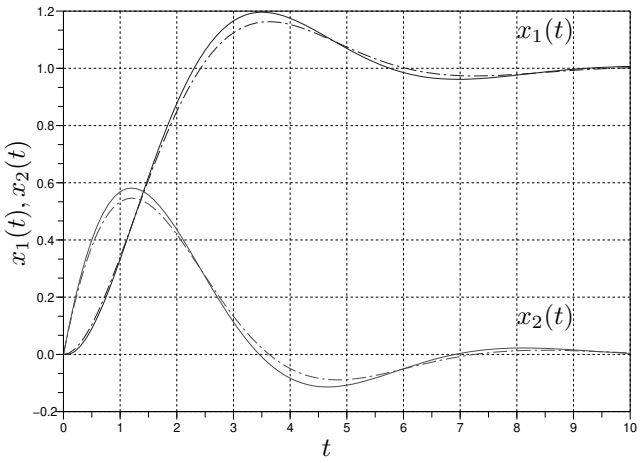

Figura 3: Solución de la Ecuación (1) del Sistema Masa Resorte con Forward Euler (sólida) con $h = 0.1$ y Solución Analítica (punteada)

Como bien puede apreciarse en la gráfica, la solución numérica brindada por el método de Euler se aproxima bastante a la solución analítica. Más adelante analizaremos las causas del error y como se puede mejorar.

Otro método de integración numérica muy conocido, inspirado en el anterior y que recibe el nombre de *Método de Backward Euler*, reemplaza la fórmula de la Ec.(5) por la siguiente:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + h\ \mathbf{f}(\mathbf{x}_{k+1}, t_k) \tag{6}$$

El método de Backward Euler tiene un pequeño inconveniente: de acuerdo a la Ec.(6) para calcular $\mathbf{x}_{k+1}$ necesitamos conocer $\mathbf{x}_{k+1}$.

Naturalmente, conociendo $\mathbf{x}_k$ podemos resolver de alguna forma la Ec.(6) y obtener de allí el valor de $\mathbf{x}_{k+1}$. Por este motivo, se dice que el método de Backward Euler es un *Método Implícito*, ya que para encontrar cada valor debemos resolver una ecuación. En contraposición, el método de Forward Euler es un *Método Explícito* ya que cada valor de la solución numérica queda determinado sin necesidad de resolver ecuaciones algebraicas.

Cuando la función $\mathbf{f}$ es lineal, la resolución de la ecuación implícita (6) es muy sencilla (si bien requiere invertir una matriz). Por el contrario, en el caso no lineal, en general necesitaremos utilizar algún algoritmo iterativo que encuentre la solución para cada instante de tiempo. Normalmente, se utiliza la *iteración de Newton*.

Por lo tanto, la implementación práctica del método de Backward Euler es mucho más compleja y computacionalmente costosa que la de Forward Euler. Más adelante, veremos que ventajas traerá que harán que valga la pena su uso.

La Figura 4 muestra el resultado de calcular la solución con Backward Euler usando un paso de integración $h = 0.1$ y compara dicha solución con la analítica. Como puede observarse, no hay una diferencia perceptible en cuanto a la precisión si se compara con lo que obtuvimos usando Forward Euler.

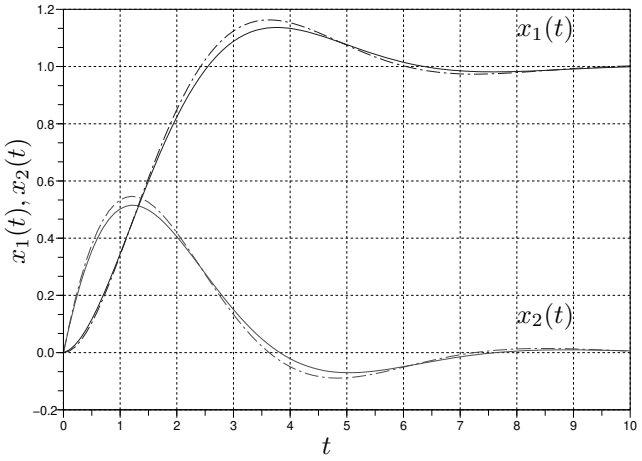

Figura 4: Solución de la Ecuación (1) del Sistema Masa Resorte con Backward Euler (sólida) con $h = 0.1$ y Solución Analítica (punteada)

2.2. Precisión de las Aproximaciones

La Figura 5 muestra el resultado de calcular la solución del sistema Masa Resorte con distintos pasos de integración usando el método de Forward Euler. Como puede observarse,el error de la aproximación crece a medida que aumentamos el paso de integración h.

Trataremos entonces de justificar teóricamente esta observación.

Conocida la solución analítica en un punto $\mathbf{x}(t_k)$, podemos calcula la solución analítica tras un paso de integración de valor h haciendo uso de la expansión en serie de Taylor en torno a t_k:

$$\begin{aligned}\mathbf{x}(t_{k+1}) &= \mathbf{x}(t_k) + h\ \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}(t_k) + \frac{h^2}{2!}\ \frac{\mathrm{d}^2\mathbf{x}}{\mathrm{d}t^2}(t_k) + \frac{h^3}{3!}\ \frac{\mathrm{d}^3\mathbf{x}}{\mathrm{d}t^3}(t_k) + \cdots \\ &= \mathbf{x}(t_k) + h\ \mathbf{f}(\mathbf{x}(t_k), t_k) + \frac{h^2}{2!}\ \frac{\mathrm{d}^2\mathbf{x}}{\mathrm{d}t^2}(t_k) + \frac{h^3}{3!}\ \frac{\mathrm{d}^3\mathbf{x}}{\mathrm{d}t^3}(t_k) + \cdots \\ &= \mathbf{x}(t_k) + h\ \mathbf{f}(\mathbf{x}(t_k), t_k) + o(h^2)\end{aligned}$$

Como podemos ver en esta expresión, el método de Forward Euler utiliza la expansión de Taylor sólo hasta el término de h, ignorando los términos de h^2, h^3, etc. Dado que el método iguala la serie hasta el término de h^1, se dice que es un algoritmo de primer orden.

El error que comete el método de Euler tras el primer paso es del orden de h^2. Esto es, asumiendo que se conoce la solución analítica en t_k el método de Euler cometerá un error proporcional a h^2 en t_{k+1}, es decir, en un paso del algoritmo.

Este error (cometido tras un paso) se denomina *Error Local por Truncamiento*[1], y en general aumentará a medida que se aumente el paso de integración. Este error siempre será proporcional a h^{N+1} donde N es el orden del método de integración.

[1]Se habla de truncamiento ya que se puede pensar que el método *trunca* la serie de Taylor a partir de h^2.

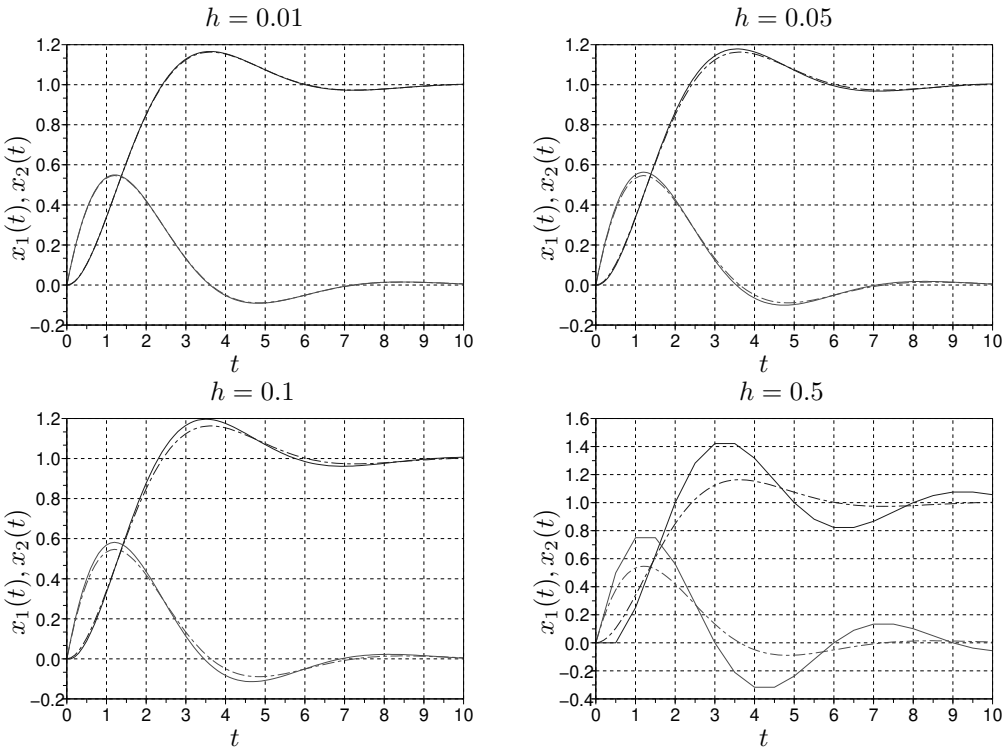

Figura 5: Solución de la Ecuación (1) del Sistema Masa Resorte con Forward Euler (sólida) para distintos pasos de integración y Solución Analítica (punteada)

En realidad, salvo en el primer paso, no conoceremos el valor de la solución analítica en t_k. Por lo tanto, en t_{k+1} habrá un error adicional ya que en cada caso estaremos partiendo de un valor inexacto en t_k. Por lo tanto, la diferencia que observamos entre la solución analítica y la solución numérica en un instante arbitrario de tiempo no es el error local, sino lo que se denomina *Error Global*.

Puede demostrarse que el máximo valor que alcanza el error global tiene siempre un orden menos que el error local. Esto es, en un método de orden N el error local es proporcional a h^{N+1} mientras que el máximo error global es proporcional a h^N.

En los métodos de Forward y Backward Euler, al ser ambos de primer orden, el máximo error global resulta proporcional al paso h. Por lo tanto, si disminuimos 10000 veces el paso de integración el error máximo disminuirá aproximadamente 10000 veces. Esto es, para realizar una simulación 10000 veces más precisa necesitamos usar un paso 10000 veces más chico, lo que implica realizar 10000 veces más cálculos.

En un método de cuarto orden, en cambio, disminuir 10 veces el paso de integración implica disminuir 10000 veces el error. Entonces, para realizar una simulación 10000 veces más precisa con un método de cuarto orden necesitamos usar un paso 10 veces más chico, lo que sólo implica realizar 10 veces más cálculos.

Esta última observación tiene una implicancia importante: cuando queremos simular con mucha precisión debemos necesariamente utilizar métodos de orden alto. De otra manera, estaremos obligados a disminuir mucho el paso de integración lo que implica realizar muchos pasos y por lo tanto muchos cálculos. Si bien como veremos más adelante los métodos de orden mayor tienen mayor costo computacional por paso, este costo se compensa al utilizar pasos más largos obteniendo mejor precisión.

Si bien los errores de truncamiento son generalmente los más importantes, hay aplicaciones en las que no se pueden ignorar los *Errores de Redondeo*. Estos errores se deben a la representación numérica con un número finito de bits en las computadoras.

Para evitar estos errores siempre se intenta trabajar con representaciones de punto flotante de 64 bits (doble precisión), lo que permite en general despreciar los efectos del redondeo. De todas maneras, en algunas aplicaciones que requieren gran precisión se utilizan métodos de orden muy alto (para dinámica de cuerpos celestes, por ejemplo) y los efectos del redondeo no pueden ignorarse por completo.

Una característica interesante de los errores de redondeo es que, a diferencia de los errores por truncamiento, tienden a aumentar a medida que el paso de integración disminuye. Una explicación simplificada de este fenómeno

puede deducirse de observar que ocurre en la Fórmula de Euler (5) al disminuir h. Cuando sumamos en punto flotante un número grande con uno muy pequeño, debido al redondeo, obtenemos nuevamente el número grande. Si repetimos indefinidamente la operación, seguiremos por lo tanto obteniendo siempre el primer número grande como resultado.

2.3. Estabilidad Numérica

Una pregunta que surge naturalmente tras analizar el error global de un método es si el mismo se sigue acumulando indefinidamente o no a medida que avanza el tiempo de la simulación. Observando nuevamente la Figura 5, vemos que a medida que la solución analítica se acerca a su valor final, la solución numérica tiende al mismo valor.

En otras palabras, en la Figura 5 vemos que la solución numérica preserva la estabilidad y el punto de equilibrio de la solución analítica.

Ahora bien, ¿ocurre esto para cualquier paso de integración?. La Figura 6 muestra que ocurre si seguimos aumentando el paso h.

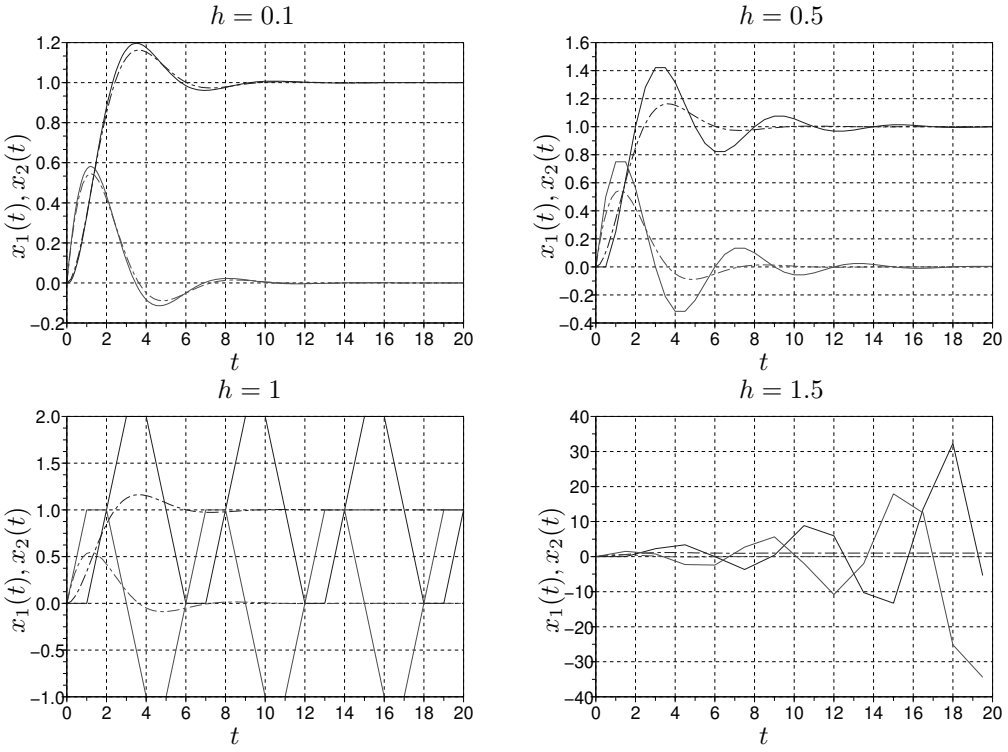

Figura 6: Solución de la Ecuación (1) del Sistema Masa Resorte con Forward Euler (sólida) para distintos pasos de integración y Solución Analítica (punteada)

Evidentemente no siempre se preserva la estabilidad. Para $h = 1$ perdemos la *estabilidad asintótica* y para $h = 1.5$ la solución numérica es claramente inestable.

Veamos que ocurre si, en cambio, utilizamos el método de Backward Euler. La Figura 7 muestra los resultados de este método implícito y como puede observarse, la estabilidad se preserva en todos los casos.

¿Como podemos justificar teóricamente estos resultados?

Si aplicamos el método de Forward Euler a un sistema lineal y estacionario,

$$\dot{\mathbf{x}}(t) = A\ \mathbf{x}(t) + B\ \mathbf{u}(t) \tag{7}$$

resulta,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + h\ (A\ \mathbf{x}_k + B\ \mathbf{u}_k) = (I + h\ A)\ \mathbf{x}_k + h\ B\ \mathbf{u}_k$$

Es decir, la ecuación en diferencias resultante tiene matriz de evolución

$$A_d = (I + h\ A)$$

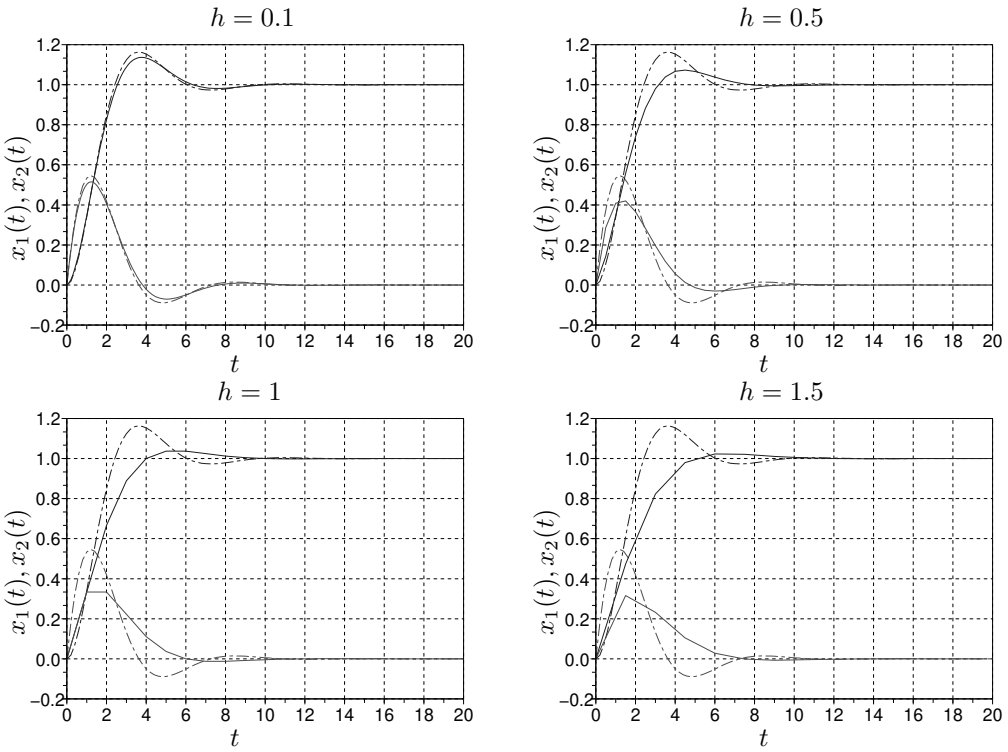

Figura 7: Solución de la Ecuación (1) del Sistema Masa Resorte con Backward Euler (sólida) para distintos pasos de integración y Solución Analítica (punteada)

Los autovalores λ_d de A_d serán las soluciones de

$$\det(\lambda_d\ I - A_d) = \det(\lambda_d\ I - I - h\ A) = 0$$

Si el determinante de una matriz es nulo, también lo es el determinante del producto por h^{-1}, por lo que,

$$\det(h^{-1}\ (\lambda_d - 1)\ I - A) = 0$$

de donde resulta que los autovalores de A_d y los de A se relacionan según:

$$h^{-1}\ (\lambda_d - 1) = \lambda$$

de donde,

$$\lambda_d = \lambda\ h + 1 \tag{8}$$

Es decir, los autovalores de la matriz del sistema discreto aproximado pueden calcularse directamente en función de los autovalores del sistema continuo original.

Si aproximamos la solución de un sistema estable, será deseable que la aproximación resulte también estable. Para esto, deberá cumplirse para todos los autovalores de A que

$$|\lambda\ h + 1| < 1 \tag{9}$$

lo que sólo ocurrirá para valores pequeños de h. Esta condición de estabilidad del método deberá cumplirse siempre y es fundamental a la hora de elegir el paso de integración.

El sistema masa resorte del ejemplo tiene autovalores $\lambda_{1,2} = -0.5 \pm j\frac{\sqrt{3}}{2}$. Puede verse entonces que la condición de estabilidad de la Ec. (9) se cumple sólo para $h < 1$, lo que corrobora los resultados de la Fig.6.

Si aplicamos en cambio Backward Euler al sistema lineal y estacionario de la Ec.(7), resulta

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{x}_k + h\ (A\ \mathbf{x}_{k+1} + B\ \mathbf{u}_{k+1}) \Rightarrow \\ (I - h\ A)\ \mathbf{x}_{k+1} &= \mathbf{x}_k + h\ B\ \mathbf{u}_{k+1} \\ \mathbf{x}_{k+1} &= (I - h\ A)^{-1}\ [\mathbf{x}_k + h\ B\ \mathbf{u}_{k+1}] \end{aligned}$$

de donde la ecuación en diferencias resultante tiene matriz de evolución

$$A_d = (I - h\ A)^{-1}$$

Los autovalores λ_d de A_d serán las soluciones de

$$\det(\lambda_d\ I - A_d) = \det(\lambda_d\ I - (I - h\ A)^{-1}) = 0$$

Si el determinante de una matriz es nulo, también lo es el determinante del producto por $\frac{(I - h\ A)}{h\ \lambda_d}$ de donde,

$$\det(\lambda_d\ \frac{(I - h\ A)}{h\ \lambda_d} - \frac{I}{h\ \lambda_d}) = \det(I\ (\frac{1}{h} - \frac{1}{h\ \lambda_d}) - A) = 0$$

de donde resulta que los autovalores de A_d y los de A se relacionan según:

$$\frac{1}{h} - \frac{1}{h\ \lambda_d} = \lambda$$

o bien

$$\lambda_d = \frac{1}{1 - \lambda\ h}$$

Por lo tanto, la condición para que el resultado numérico sea estable es que

$$|\frac{1}{1 - \lambda\ h}| < 1 \tag{10}$$

Puede verse fácilmente que cuando $\mathbb{R}\mathrm{e}(\lambda) < 0$, la condición se cumplirá siempre. Esto es, al simular un sistema estable con el método de Backward Euler, la solución numérica siempre resultará estable (hecho que corrobora lo observado en la Fig.7).

Esta propiedad de preservar la estabilidad numérica es la principal ventaja de Backward Euler sobre Forward Euler. Más adelante veremos que esta característica será imprescindible para poder simular adecuadamente ciertos sistemas.

Entre tanto, una desventaja del algoritmo de Backward Euler es que para autovalores inestables $\mathbb{R}\mathrm{e}(\lambda) > 0$ puede resultar (si h es muy grande) $|\lambda_d| < 1$ (estable). Es decir, la simulación de un sistema inestable puede resultar estable, lo cual es indeseado.

Por ejemplo, el sistema inestable

$$\dot{x}(t) = x(t)$$

tiene un autovalor $\lambda = 1$. Por lo tanto, usando cualquier paso $h > 2$, vemos que la condición de la Ec.(10) se cumple lo que indica que la solución numérica será estable.

3. Métodos Monopaso

Como mencionamos antes, los métodos de Forward y Backward Euler que vimos hasta aquí realizan sólo aproximaciones de primer orden. Debido a esto, si se quiere obtener una buena precisión en la simulación, se debe reducir excesivamente el paso de integración lo que implica una cantidad de pasos y de cálculos en general inaceptable.

Para obtener aproximaciones de orden mayor, será necesario utilizar más de una evaluación de la función $\mathbf{f}(\mathbf{x}, t)$ en cada paso. Cuando dichas evaluaciones se realizan de manera tal que para calcular $\mathbf{x}_{k+1}$ sólo se utiliza el valor de $\mathbf{x}_k$, se dice que el algoritmo es *monopaso*. Por el contrario, cuando se utilizan además valores anteriores de la solución ($\mathbf{x}_{k-1}$, $\mathbf{x}_{k-2}$, etc.), se dice que el algoritmo es *multipaso*.

Los métodos monopaso se suelen denominar también *Métodos de Runge–Kutta*, ya que el primero de estos métodos de orden alto fue formulado por Runge y Kutta a finales del siglo XIX.

3.1. Métodos Explícitos de Runge Kutta

Los métodos explícitos de Runge–Kutta realizan varias evaluaciones de la función $\mathbf{f}(\mathbf{x}, t)$ en cercanías del punto $(\mathbf{x}_k, t_k)$ y luego calculan $\mathbf{x}_{k+1}$ realizando una suma pesada de dichas evaluaciones.

Uno de los algoritmos más simple es el denominado *Método de Heun*, inventado por Heun en el año 1900. Dicho método utiliza la fórmula

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{h}{2}\ (\mathbf{k}_1 + \mathbf{k}_2)$$

donde

$$\mathbf{k}_1 = \mathbf{f}(\mathbf{x}_k, t_k)$$
$$\mathbf{k}_2 = \mathbf{f}(\mathbf{x}_k + h\ \mathbf{k}_1, t_k + h)$$

Comparado con Euler, este método realiza una evaluación adicional de la función $\mathbf{f}$ para calcular $\mathbf{k}_2$. El beneficio que se obtiene es que el método de Heun realiza una aproximación de segundo orden y resulta mucho más preciso que Euler.

Para el sistema masa resorte de la Ec.(1), el método de Heun con paso $h = 0.1$ comenzaría con los siguientes cálculos:

$$\mathbf{x}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow$$
$$\mathbf{k}_1 = \mathbf{f}(\mathbf{x}_0, t_0) = \begin{bmatrix} x_2 \\ 1 - x_1 - x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$
$$\mathbf{x}_0 + h\ \mathbf{k}_1 = \begin{bmatrix} 0 + 0.1 \cdot 0 \\ 0 + 0.1 \cdot 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} \Rightarrow$$
$$\mathbf{k}_2 = \mathbf{f}(\mathbf{x}_0 + h\ \mathbf{k}_1, t_0 + h) = \begin{bmatrix} 0.1 \\ 1 - 0 - 0.1 \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.9 \end{bmatrix}$$
$$\mathbf{x}_1 = \mathbf{x}_0 + \frac{h}{2}\ (\mathbf{k}_1 + \mathbf{k}_2) = \begin{bmatrix} 0 + 0.05 \cdot (0 + 0.1) \\ 0 + 0.05 \cdot (1 + 0.9) \end{bmatrix} = \begin{bmatrix} 0.005 \\ 0.095 \end{bmatrix}$$

Otro método, uno de los más utilizados, es el de Runge–Kutta de orden 4 (RK4), desarrollado por Runge y Kutta en 1895:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{h}{6}\ (\mathbf{k}_1 + 2\ \mathbf{k}_2 + 2\ \mathbf{k}_3 + \mathbf{k}_4)$$

donde

$$\mathbf{k}_1 = \mathbf{f}(\mathbf{x}_k, t_k)$$
$$\mathbf{k}_2 = \mathbf{f}(\mathbf{x}_k + h\ \frac{\mathbf{k}_1}{2}, t_k + \frac{h}{2})$$
$$\mathbf{k}_3 = \mathbf{f}(\mathbf{x}_k + h\ \frac{\mathbf{k}_2}{2}, t_k + \frac{h}{2})$$
$$\mathbf{k}_4 = \mathbf{f}(\mathbf{x}_k + h\ \mathbf{k}_3, t_k + h)$$

Este algoritmo, como vemos, utiliza cuatro evaluaciones de la función $\mathbf{f}$ en cada paso. En este caso, el beneficio es que resulta una aproximación de orden 4.

La Tabla 1 resume los resultados de simular el sistema Masa Resorte de la Ec.(1) con los métodos de Euler, Heun y Runge–Kutta de orden 4. En cada caso, se reporta el máximo error obtenido en toda la simulación (esto es, el máximo error global). Puede notarse como en Euler el error disminuye linealmente con el paso, mientras que en Heun lo hace de forma cuadrática y en RK4 lo hace con la cuarta potencia del paso.

Paso	Error Euler	Error Heun	Error RK4
$h = 0.5$	0.298	0.0406	4.8×10^{-4}
$h = 0.1$	0.042	1.47×10^{-3}	6.72×10^{-7}
$h = 0.05$	0.0203	3.6×10^{-4}	4.14×10^{-8}
$h = 0.01$	3.94×10^{-3}	1.42×10^{-5}	6.54×10^{-11}

Tabla 1: Máximo Error Cometido por los Métodos de Euler, Heun y RK4 para Distintos Pasos de Integración con el Sistema Masa Resorte de le Ec.(1)

Hay infinidad de métodos de Runge Kutta explícitos para diversos órdenes (incluyendo métodos de órdenes mayores a 10). De todas maneras, los de orden 1 a 5 son los más utilizados en la práctica.

En lo referente a estabilidad, los métodos de RK tienen características similares a las de Forward Euler. Esto es, preservan la estabilidad para valores no muy grandes del paso de integración h. Por lo tanto, aunque la precisión sea buena, no podremos utilizar pasos muy grandes debido a los límites que impone la estabilidad.

Por ejemplo, para el método de Heun, podemos analizar la estabilidad sobre un sistema escalar de primer orden:

$$\dot{x} = \lambda\ x(t)$$

cuya solución analítica será estable si $\lambda < 0$. Aplicando Heun, tenemos,

$$\begin{aligned} x_{k+1} &= x_k + \frac{h}{2}(k_1 + k_2) \\ &= x_k + \frac{h}{2}(\lambda x_k + \lambda\ (x_k + h\ \lambda x_k)) \\ &= x_k + h\ \lambda x_k + \frac{h^2\lambda^2}{2}\ x_k \\ &= (1 + h\ \lambda + \frac{h^2\lambda^2}{2})\ x_k \\ &= \lambda_d\ x_k \end{aligned}$$

es decir, el autovalor discreto resulta $\lambda_d = 1 + h\ \lambda + \frac{h^2\lambda^2}{2}$, y la condición de estabilidad numérica es entonces

$$|1 + h\ \lambda + \frac{h^2\lambda^2}{2}| < 1$$

que se traduce en dos desigualdades:

$$\begin{cases} 1 + h\ \lambda + \frac{h^2\lambda^2}{2} & < 1 \\ 1 + h\ \lambda + \frac{h^2\lambda^2}{2} & > -1 \end{cases}$$

La primera desigualdad conduce a la condición

$$h < 2|\lambda| \tag{11}$$

y la segunda se cumple siempre que $\lambda < 0$. En conclusión, para un autovalor real negativo, el método de Heun dará un resultado estable cuando el paso sea suficientemente chico de manera que se cumpla la Ec.(11). Este resultado es idéntico al de Forward Euler para autovalores reales negativos.

3.2. Métodos Monopaso Implícitos

Vimos que el método de Backward Euler era un algoritmo implícito cuya principal ventaja es preservar la estabilidad de la solución numérica para cualquier paso de integración. Sin embargo, al igual que Forward Euler, realiza sólo una aproximación de primer orden.

Hay diversos métodos implícitos monopaso de orden mayor que, al igual que Backward Euler, preservan la estabilidad.

Uno de los más utilizados es la *Regla Trapezoidal*, cuya definición es la siguiente:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{h}{2}\ [\mathbf{f}(\mathbf{x}_k, t_k) + \mathbf{f}(\mathbf{x}_{k+1}, t_{k+1})] \tag{12}$$

Este método implícito realiza una aproximación de segundo orden y tiene la propiedad (al menos en sistemas lineales y estacionarios) de que la solución numérica es estable si y sólo si la solución analítica es estable.

Por este motivo, la regla trapezoidal se suele utilizar a menudo para simular eficazmente sistemas muy oscilantes.

Si bien existen numerosos métodos implícitos monopaso, en la práctica no son tan utilizados ya que en general los métodos multipaso implícitos suelen ser más eficientes.

3.3. Algoritmos de Control de Paso

Hasta aquí consideramos siempre el paso h como un parámetro fijo que debe elegirse previo a la simulación. Sin embargo, en muchos casos es posible implementar algoritmos que cambien el paso de integración de forma automática a medida que avanza la simulación.

Los algoritmos de *control de paso* tienen por propósito mantener el error de simulación acotado para lo cual ajustan automáticamente el paso en función del error estimado. La idea es muy simple, en cada paso se hace lo siguiente:

1. Se da un paso con el método de integración elegido calculando $\mathbf{x}_{k+1}$ y cierto paso h.

2. Se estima el error cometido.

3. Si el error es mayor que la tolerancia, se disminuye el paso de integración h y se recalcula $\mathbf{x}_{k+1}$ volviendo al punto 1.

4. Si el error es menor que la tolerancia, se acepta el valor de $\mathbf{x}_{k+1}$ calculado, se incrementa el paso h y se vuelve al punto 1 para calcular $\mathbf{x}_{k+2}$.

Si bien la idea es muy sencilla, de la descripción anterior surgen varias cuestiones:

- ¿Cómo estimamos el error?
- ¿Qué ley utilizamos para recalcular el paso de integración h en cada caso?
- ¿Qué valor de paso de integración usamos inicialmente?

A continuación analizaremos en detalle dichos interrogantes.

3.3.1. Estima del Error

La estima del error de integración se hace utilizando dos métodos de orden diferente. Dado $\mathbf{x}_k$, calculamos, por ejemplo, $\mathbf{x}_{k+1}$ usando un método de orden 4 y luego $\tilde{\mathbf{x}}_{k+1}$ con un método de orden 5. Dado que un método de orden 5 es mucho más preciso que uno de orden 4, podemos suponer que el error en el método de orden 4 es directamente la diferencia entre ambos.

Para ahorrar cálculos, se suelen buscar métodos que compartan evaluaciones comunes en las funciones, de manera que para evaluar $\tilde{\mathbf{x}}_{k+1}$ puedan reutilizarse algunos de los cálculos hechos para calcular $\mathbf{x}_{k+1}$.

Por ejemplo, las siguientes fórmulas calculan el método de Heun (2do orden) y un método de RK de tercer orden:

$$\begin{aligned}
\mathbf{k}_1 &= \mathbf{f}(\mathbf{x}_k, t_k) \\
\mathbf{k}_2 &= \mathbf{f}(\mathbf{x}_k + h\ \mathbf{k}_1, t_k + h) \\
\mathbf{k}_3 &= \mathbf{f}(\mathbf{x}_k + \frac{h}{4}\ \mathbf{k}_1 + \frac{h}{4}\ \mathbf{k}_2, t_k + \frac{h}{2})
\end{aligned}$$

y luego

$$\mathbf{x}_{k+1} = x_k + \frac{h}{2}(\mathbf{k}_1 + \mathbf{k}_2) \tag{13}$$

$$\tilde{\mathbf{x}}_{k+1} = x_k + \frac{h}{6}(\mathbf{k}_1 + \mathbf{k}_2 + 4\mathbf{k}_3) \tag{14}$$

de donde puede estimarse el error como

$$\mathbf{e}_{k+1} \approx \mathbf{x}_{k+1} - \tilde{\mathbf{x}}_{k+1} = \frac{\mathbf{k}_1 + \mathbf{k}_2 - 2\mathbf{k}_3}{3}$$

Este método se denominará RK23, ya que se trata de un algoritmo de Runge–Kutta de orden 2 con estima de error de orden 3. Como puede observarse la Ec.(13) (Heun) y la Ec.(14) (RK3) comparten las etapas de cálculo de $\mathbf{k}_1$ y $\mathbf{k}_2$, lo que ahorra bastante costo computacional.

Uno de los métodos más utilizados en la práctica es un RK45 denominado *Runge–Kutta–Fehlberg*, que utiliza 6 evaluaciones de la función $\mathbf{f}$ para calcular dos soluciones, una de orden 4 y otra de orden 5.

3.3.2. Ajuste del Paso

Como vimos anteriormente, el error local que comete un método de orden N es proporcional a h^{N+1}, esto es,

$$err \approx c\ h^{N+1} \tag{15}$$

donde c es una constante desconocida. Si usamos cierto paso h y obtenemos un error err, la pregunta es, ¿que paso h_0 deberíamos usar para obtener un error tol, donde tol es la tolerancia de error que admitimos?

La respuesta a esta pregunta es lo que nos permitirá ajustar el paso tanto para disminuirlo como para aumentarlo.

Naturalmente, podemos escribir

$$tol \approx c\ h_0^{N+1}$$

Dividiendo miembro a miembro con la Ec.(15) resulta,

$$\frac{tol}{err} \approx \frac{h_0^{N+1}}{h^{N+1}}$$

de donde,

$$h_0 = h\ (tol/err)^{\frac{1}{N+1}}$$

que nos brinda la fórmula para ajustar el paso de acuerdo a la tolerancia admitida, al error estimado y al paso utilizado. Esta última ecuación generalmente se modifica de manera tal que

$$h_0 = 0.8\ h\ (tol/err)^{\frac{1}{N+1}} \tag{16}$$

para asegurar que el paso utilizado no resulte demasiado cercano al límite teórico de precisión y evitar tener que recalcular pasos.

Si simulamos con esta idea el sistema masa resorte de la Ec.(1), utilizando el método de Runge–Kutta–Fehlberg (RK45) que mencionamos antes, obtenemos el resultado que se muestra en la Fig.8.

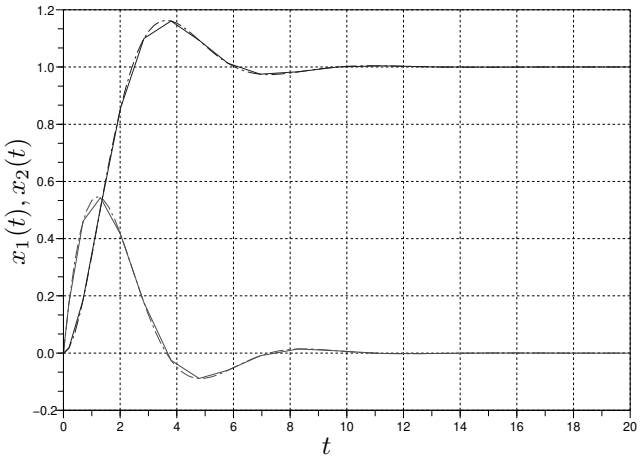

Figura 8: Solución de la Ecuación (1) del Sistema Masa Resorte con Runge–Kutta–Fehlberg (sólida) con $tol = 0.001$ y Solución Analítica (punteada)

El error es inapreciable aunque las curvas difieren ya que la solución numérica tiene muy pocos puntos (hay sólo 25 pasos) y al graficar se perciben las rectas que unen los puntos sucesivos.

La Figura 9 muestra la variación del paso de integración. A medida que el sistema se acerca al punto de equilibrio los pasos se hacen más grandes. Sin embargo, h no puede crecer demasiado debido a las limitaciones de estabilidad del método.

3.3.3. Algunos Parámetros de los Algoritmos de Control de Paso

Los parámetros que utilizan los algoritmos de control de paso son generalmente los siguientes:

- *Tolerancia relativa:* Es el error que se admite, expresado como fracción de los valores de las variables. Por ejemplo, cuando se propone una tolerancia relativa de 0.001 (un valor típico), se está imponiendo que el error no debe superar el uno por mil del valor de la solución.
- *Tolerancia absoluta:* Cuando las variables toman valores muy cercanos a cero, la tolerancia relativa impone un error demasiado pequeño que implicaría utilizar pasos exageradamente chicos. Por esto, se establece este parámetro adicional que es el mínimo valor de tolerancia a utilizar.

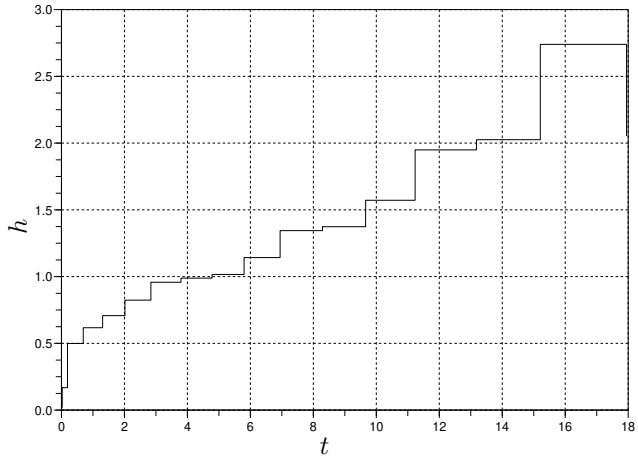

Figura 9: Variación del Paso de Integración en la Solución de la Fig.8

- *Máximo paso de integración:* A veces, puede resultar que los dos métodos usados coincidan muy bien en el valor calculado y el error estimado sea cercano a cero. De acuerdo a la fórmula (16), en tal caso el valor de h resultará muy grande. Por eso, se suele limitar el paso a un valor máximo.
- *Mínimo paso de integración:* En ciertas ocasiones, es posible que no se pueda alcanzar la tolerancia establecida sin utilizar pasos exageradamente pequeños. Esto traería como consecuencia que se trabe la simulación ya que deberían realizarse un número demasiado grande de pasos. Por esto, se suele limitar también por debajo el paso de integración. Este paso mínimo se suele también utilizar como valor inicial del paso.
- *Intervalo de Interpolación:* Al utilizar control de paso, la solución se calcula a intervalos irregulares de tiempo. En muchas aplicaciones, es importante contar con la solución calculada en determinados instantes de tiempo generalmente equiespaciados. Para esto, se suelen utilizar algoritmos de interpolación que calculan la solución en los instantes deseados.

4. Métodos Multipaso

Los métodos monopaso obtienen aproximaciones de orden alto utilizando para esto varias evaluaciones de la función $\mathbf{f}$ en cada paso. Para evitar este costo computacional adicional, se han formulado diversos algoritmos que, en lugar de evaluar repetidamente la función $\mathbf{f}$ en cada paso, utilizan los valores evaluados en pasos anteriores.

Veremos a continuación algunos de estos algoritmos.

4.1. Métodos Multipaso Explícitos

Dentro de los métodos multipaso explícitos encontramos, entre los más utilizados, a los métodos de Adams–Bashforth (AB) y de Adams–Bashforth–Moulton (ABM).

El método de AB de 2do orden, por ejemplo, está dado por la fórmula:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{h}{2}\left(3\mathbf{f}_k - \mathbf{f}_{k-1}\right) \tag{17}$$

donde definimos

$$\mathbf{f}_k \triangleq \mathbf{f}(\mathbf{x}_k, t_k)$$

El método de AB de 4to orden, en tanto, es el que sigue:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{h}{24}\left(55\mathbf{f}_k - 59\mathbf{f}_{k-1} + 37\mathbf{f}_{k-2} - 9\mathbf{f}_{k-3}\right) \tag{18}$$

Como vemos, estos métodos requieren realizar una única evaluación de la función $\mathbf{f}$ en cada paso. El orden alto de aproximación se logra utilizando la información de las evaluaciones anteriores de dicha función.

Un problema de los métodos multipaso es el arranque. La expresión de la Ec.(17) sólo puede utilizarse para calcular la solución a partir de $k = 2$. Para el primer pasos no está definido el valor de $\mathbf{x}_{-1}$ requerido por la fórmula. Similarmente, el algoritmo de AB4 de la Ec.(18) sólo puede usarse a partir de $k = 4$ por la misma razón.

Por este motivo, los primeros pasos de los métodos multipaso deben realizarse con algún método monopaso de, al menos, el mismo orden de precisión. Por ejemplo, para el método de AB4 de la Ec.(18), deberían darse los tres primeros pasos con Runge–Kutta de orden 4.

Veamos por ejemplo, el resultado de utilizar AB2 para simular el sistema masa resorte del ejemplo. Usaremos como valores iniciales $\mathbf{x}_0 = [0\ 0]$ y $\mathbf{x}_1 = [0.005\ 0095]$, este último calculado previamente con el método de Heun. Luego,

$$\mathbf{x}_0 = \begin{bmatrix}0\\0\end{bmatrix} \Rightarrow \mathbf{f}(\mathbf{x}_0, t_0) = \begin{bmatrix}x_2\\1 - x_1 - x_2\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix}$$
$$\mathbf{x}_1 = \begin{bmatrix}0.005\\0.095\end{bmatrix} \Rightarrow \mathbf{f}(\mathbf{x}_1, t_1) = \begin{bmatrix}x_2\\1 - x_1 - x_2\end{bmatrix} = \begin{bmatrix}0.095\\0.9\end{bmatrix}$$
$$\mathbf{x}_2 = \mathbf{x}_1 + \frac{h}{2}(3\ \mathbf{f}_1 - \mathbf{f}_0) = \begin{bmatrix}0.005\\0.095\end{bmatrix} + 0.05\ \left(3\ \begin{bmatrix}0.095\\0.9\end{bmatrix} - \begin{bmatrix}0\\1\end{bmatrix}\right) = \begin{bmatrix}0.01925\\0.18\end{bmatrix}$$

Otro problema de los métodos de Adams–Bashforth es que resultan estables sólo para valores muy pequeños del paso de integración. Esta característica se mejora utilizando los métodos de Adams–Bashforth–Moulton (ABM).

El método de ABM de tercer orden, por ejemplo, está dado por la siguiente fórmula

$$\mathbf{x}^P_{k+1} = \mathbf{x_k} + \frac{h}{12}(23\mathbf{f}_k - 16\mathbf{f}_{k-1} + 5\mathbf{f}_{k-2})$$
$$\mathbf{x}_{k+1} = \mathbf{x_k} + \frac{h}{12}(5\mathbf{f}^P_{k+1} + 8\mathbf{f}_k - \mathbf{f}_{k-1})$$

donde $\mathbf{f}^P_{k+1} = \mathbf{f}(x^P_{k+1}, t_{k+1})$.

Este método, denominado *predictor–corrector*, tiene mejor estabilidad que AB3 a costa de utilizar una evaluación extra de la función $\mathbf{f}$.

4.2. Métodos Multipaso Implícitos

Para obtener soluciones numéricas que preserven la estabilidad independientemente del paso de integración, al igual que en los métodos monopaso, hay que recurrir a los métodos implícitos.

Los métodos implícitos más utilizados en la práctica son los denominados *Backward Difference Formulae* (BDF), que son de tipo multipaso. Por ejemplo, el siguiente es el método de BDF de orden 3:

$$\mathbf{x}_{k+1} = \frac{18}{11}\mathbf{x}_k - \frac{9}{11}\mathbf{x}_{k-1} + \frac{2}{11}\mathbf{x}_{k-2} + \frac{6}{11} \cdot h \cdot \mathbf{f}_{k+1} \tag{19}$$

Este método tiene prácticamente el mismo costo computacional que Backward Euler, ya que la ecuación a resolver es muy similar. Sin embargo, BDF3 es de tercer orden.

Existen métodos de BDF de hasta orden 8, si bien los más utilizados son los de orden 1 (Backward Euler) hasta 5.

4.3. Control de Paso

En los métodos multipaso se puede también controlar el paso de integración de manera similar a la de los métodos monopaso. Sin embargo, las fórmulas de los métodos multipaso son sólo válidas asumiendo paso constante (ya que usan valores anteriores de la solución). Por lo tanto, para cambiar el paso en un método multipaso hay que interpolar los últimos valores de la solución a intervalos regulares correspondientes al nuevo paso de integración. En consecuencia, cambiar el paso tiene un costo adicional en este caso.

Por este motivo, los algoritmos de control de paso en métodos multipaso sólo cambian el paso cuando el cambio de paso es suficientemente grande.

5. Algunas Dificultades

Muchos sistemas dinámicos en la práctica tienen modelos en sistemas de ecuaciones diferenciales que resultan problemáticos para los distintos métodos de integración numérica.

Veremos a continuación distintos casos que nos obligarán a utilizar ciertas técnicas particulares.

5.1. Sistemas Stiff

Consideremos nuevamente el sistema masa resorte de la Ec.(1), pero ahora supongamos que el coeficiente de rozamiento es $b = 100$.

Con este nuevo valor para el parámetro, el sistema tiene matriz de evolución

$$A = \begin{bmatrix} 0 & 1 \\ -1 & -100 \end{bmatrix}$$

cuyos autovalores son $\lambda_1 \approx -0.01$ y $\lambda_2 \approx -100$.

La Figura 10 muestra la solución analítica (sólo se grafica x_2 en este caso). En la curva de la izquierda se ve la trayectoria completa, que se asemeja a una evolución de primer orden (correspondiente al autovalor lento $\lambda_1 \approx -0.01$. La curva de la derecha muestra en tanto el inicio de la trayectoria, que también se asemeja a una evolución de primer orden, pero mucho más rápida (correspondiente al autovalor $\lambda_2 \approx -100$).

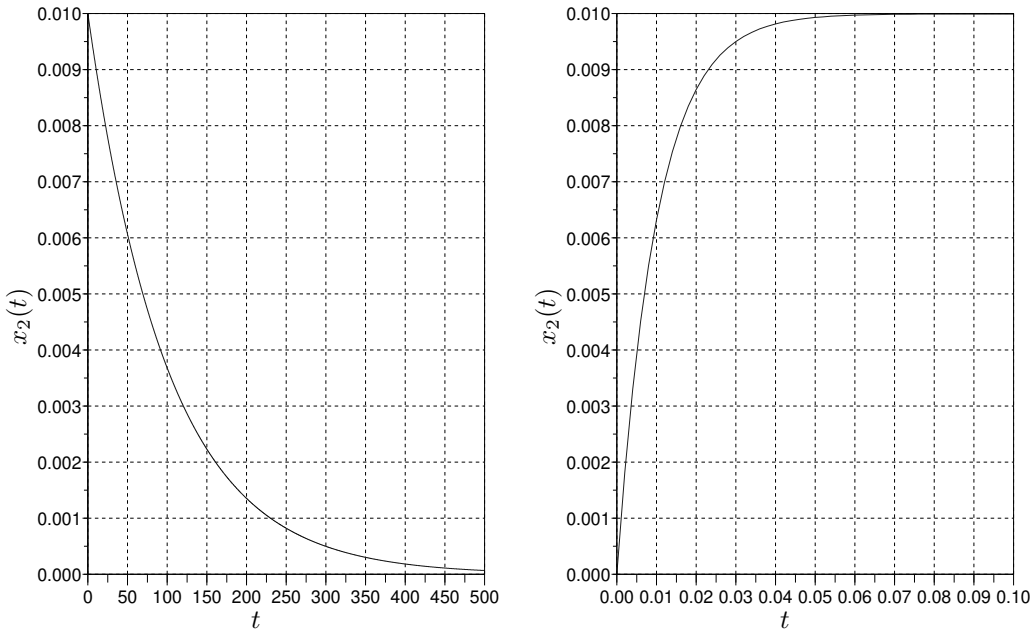

Figura 10: Solución Analítica para x_2 en el Sistema Masa Resorte con $b = 100$.

Evidentemente este sistema posee una dinámica lenta y una rápida. Los sistemas que tienen estas características se denominan *sistemas stiff* o rígidos. Como veremos, esto nos complicará a la hora de simularlo.

Es claro que si queremos simular este sistema usando algún método de integración, debemos hacerlo hasta un tiempo final de al menos $t_f = 500$. Además, si utilizamos el método de Euler, por ejemplo, deberemos usar un paso de integración $h < 0.02$ ya que de otra manera tendremos un resultado inestable según la Ec.(9). Esto implicará que haremos al menos $500/0.2 = 25000$ pasos para completar la simulación, lo que es absurdo dado lo simple del sistema.

La primer idea es entonces usar algún método de paso variable, como por ejemplo, el RK45 que vimos antes. El resultado en este caso no es mucho mejor, ya que el método necesita 13604 pasos para completar la simulación. La Figura 11 muestra la evolución del paso de integración en dicha simulación (se muestra sólo lo que ocurre hasta $t = 10$, luego sigue constante).

Aunque como puede verse en la solución analítica a la izquierda de la Fig.10 la dinámica rápida prácticamente desaparece a partir de $t = 0.1$, el método de RK45 nunca puede aumentar el paso de integración mucho más allá de $h \approx 0.04$. La explicación para esto es sencilla: debido al autovalor rápido $\lambda_2 \approx -100$ con pasos de integración mayores el método de RK4 se torna inestable por lo que el algoritmo obtiene un error grande que lo obliga a disminuir nuevamente el paso.

Este efecto sólo puede evitarse si usamos algoritmos que no se inestabilicen, esto es, si usamos alguno de los métodos implícitos que presentamos.

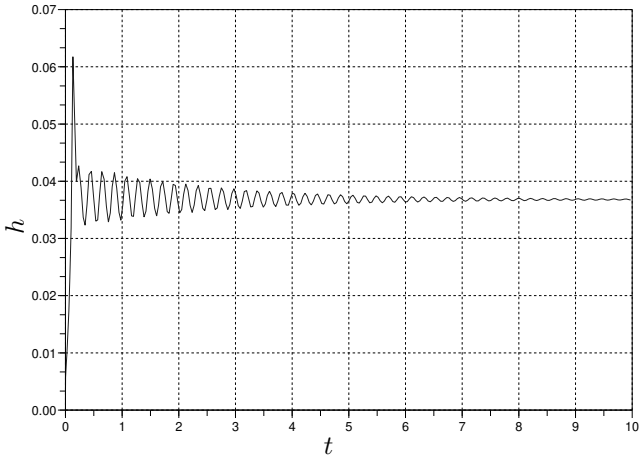

Figura 11: Paso de Integración h en la Simulación de un Sistema Stiff con RK45.

Utilizando por ejemplo, un método implícito de RK de cuarto orden con control de error de orden 5, con la misma tolerancia que en RK45 (0.001), la simulación se completa en 33 sólo pasos y se obtiene una precisión idéntica a la anterior. La Fig.12 muestra la evolución del paso de integración en este caso. El mismo está sólo limitado por la precisión y nunca por la estabilidad.

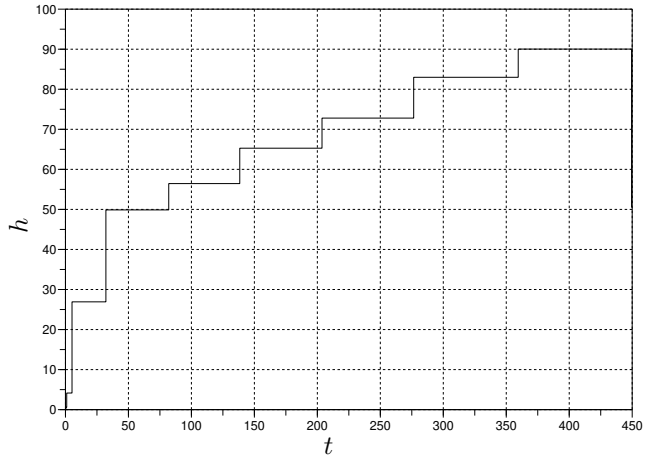

Figura 12: Paso de Integración h en la Simulación de un Sistema Stiff con un método implícito monopaso.

Si bien una computadora moderna puede realizar 13000 pasos en un sistema sencillo en un tiempo casi despreciable, en general los casos stiff son mucho más severos que lo visto aquí. En el simple ejemplo del masa resorte, si hubiéramos tenido $b = 10000$, RK45 necesitaría más de 13 millones de pasos para completar la simulación.

En la práctica, es casi imposible simular sistemas stiff sin recurrir a algoritmos implícitos.

5.2. Sistemas Marginalmente Estables

Muchos sistemas de la práctica tienen nulo o escaso amortiguamiento. En tal caso, las soluciones muestran oscilaciones sostenidas o bien que se amortiguan muy lentamente. Esta característica puede observarse en el ejemplo del sistema Masa Resorte haciendo $b = 0$ o tomando b muy peqeño.

Los sistemas con estas propiedades se denominan *marginalmente estables* y también revisten algunos problemas para la simulación numérica.

En el caso del sistema masa resorte con $b = 0$ tenemos un par de autovalores imaginarios puros conjugados $\lambda_{1,2} = \pm j$, por lo que las soluciones serán periódicas.

La Figura 13 muestra la solución analítica y el resultado de simular el sistema con Forward y Backward Euler usando $h = 0.1$. Como puede observarse en la misma, la solución numérica de Forward Euler es inestable, mientras que la de Backward Euler es asintóticamente estable.

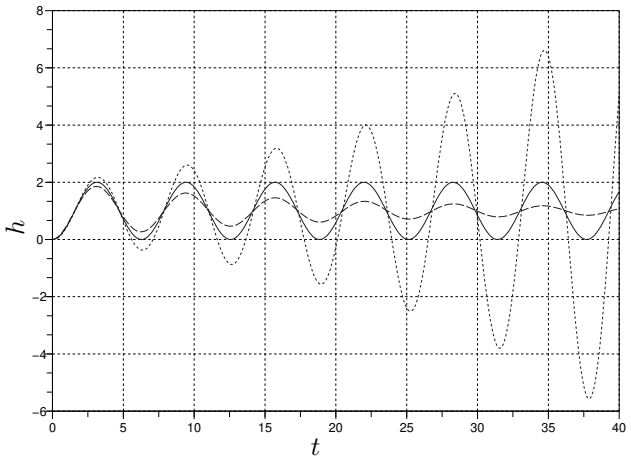

Figura 13: Sistema Masa Resorte con $b = 0$: Solución analítica (sólida), Forward Euler (punteada) y Backward Euler (línea discontinua).

Ninguna de las dos soluciones respeta la característica marginalmente estable original por lo que los resultados son cualitativamente incorrectos. Para obtener buenos resultados hay que utilizar métodos denominados *F–Estables* que preservan también la estabilidad marginal. Uno de estos métodos es la Regla Trapezoidal que presentamos en la Ec.(12).

5.3. Sistemas con Discontinuidades

La siguiente ecuación es un modelo muy simple de una pelotita que rebota contra el piso:

$$\begin{aligned}\dot{x}(t) =& v(t)\\ \dot{v}(t) =& -g - s_w(t) \cdot \frac{1}{m}(k \cdot x(t) + b \cdot v(t))\end{aligned}$$

donde

$$s_w = \begin{cases} 0 & \text{si } x(t) > 0 \\ 1 & \text{en otro caso} \end{cases}$$

En este modelo, estamos considerando que cuando $x(t) > 0$ la pelotita está en el aire y responde a una ecuación de caída libre ($s_w = 0$). Cuando la pelotita entra en contacto con el piso, en cambio, sigue un modelo *masa–resorte–amortiguador*, lo que produce el rebote.

En este caso, consideramos parámetros $m = 1$, $b = 30$, $k = 1 \times 10^6$ y $g = 9.81$.

Decidimos simular este sistema utilizando el método de RK4, durante 5 segundos, a partir de la condición inicial $x(0) = 1$, $v(0) = 0$. Esta situación corresponde a soltar la pelotita desde 1 metro de distancia del piso. Comenzamos a tener resultados *decentes* con un paso de integración $h = 0.002$. En la Fig 14 se muestran los resultados de la simulación con pasos $h = 0.002$, $h = 0.001$ y $h = 0.0005$.

Evidentemente, no podemos confiar en los resultados de esta simulación. La del paso más pequeño parecería más precisa, pero no hay manera de asegurarlo por el momento.

Mirando el comienzo de la simulación, no hay error apreciable hasta el primer pique. Evidentemente, el problema tiene que ver con la discontinuidad.

Veamos que ocurre en tanto si utilizamos un método de paso variable. En la Fig 15 se pueden apreciar los resultados con RK23 utilizando las tolerancias relativas 10^{-3}, 10^{-4} y 10^{-5}. De nuevo, los resultados son desalentadores. Si bien

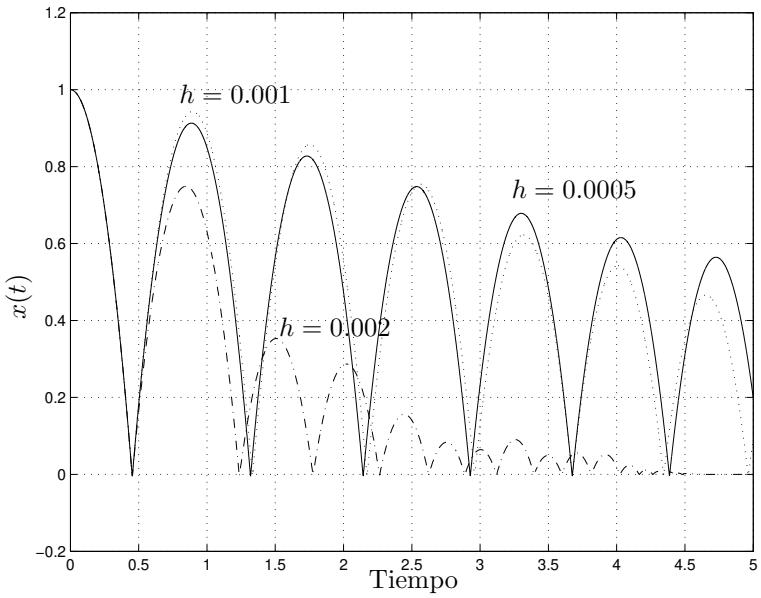

Figura 14: Simulación con RK4 de la pelotita rebotando.

algunos piques se resuelven *bien* (al menos el método hace lo mismo con las tres tolerancias), en otros piques el error se torna inaceptable.

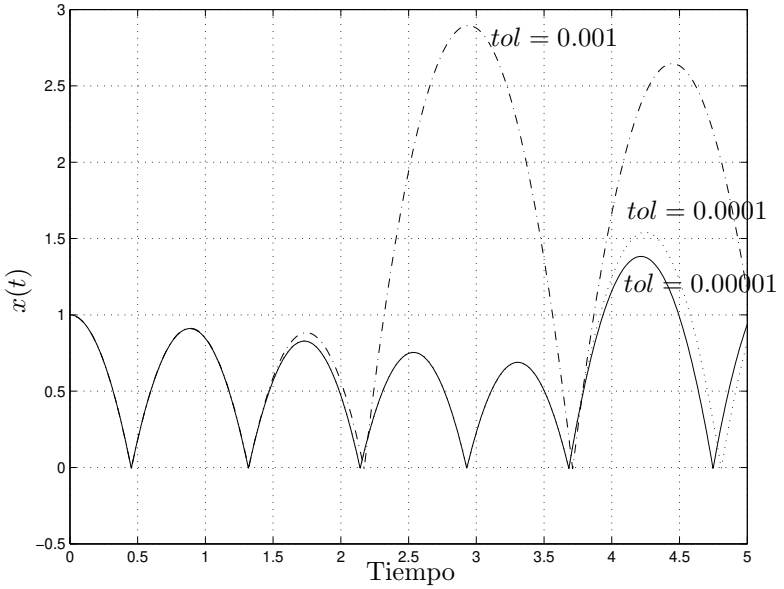

Figura 15: Simulación con RK23 de la pelotita rebotando.

Tanto al usar paso fijo como al usar paso variable el problema tiene que ver con que se dieron pasos que atravesaron la discontinuidad. Es decir, en t_k teníamos $x(t_k) > 0$ y en t_{k+1} resultó $x(t_{k+1}) < 0$. Como la función $\mathbf{f}$ resulta discontinua entre t_k y t_{k+1}, se produce un error muy grande.

La manera de corregir estos errores es evitando la situación anterior. Para esto, si nos encontramos con que $x(t_{k+1}) < 0$ hay que volver atrás y buscar el punto donde se produce la discontinuidad, esto es, el valor de $\tilde{t}$ para el cual $x(\tilde{t}) = 0$. Una vez detectado este punto (para lo cual puede ser necesario iterar), se debe realizar un paso de valor $h = \tilde{t} - t_k$ (para llegar a dicho punto) y desde este nuevo punto reiniciar la simulación.

Los problemas de detección y tratamiento de discontinuidades son cruciales para simular correctamente sistemas de electrónica de potencia. En estos sistemas, generalmente, las discontinuidades ocurren a frecuencias muy grandes por lo que el uso de algoritmos eficientes para tratar con las discontinuides es imprescindible.

6. Recapitulación

En estas notas realizamos una introducción a los métodos de integración numérica de ecuaciones diferenciales ordinarias. En primer lugar, tras una breve motivación sobre la necesidad de utilizar dichos algoritmos, presentamos los métodos de Forward y Backward Euler. Luego, analizamos dos de las propiedades fundamentales que caracterizan a los distintos métodos de integración: el orden de la aproximación y la estabilidad numérica.

Con respecto al orden de aproximación, introdujimos los conceptos de error local y global. Vimos, a través de la expansión en serie de Taylor, que el error local es aproximadamente proporcional a h^{N+1} siendo N el orden del método y mencionamos que el máximo error global resulta aproximadamente proporcional a h^N. En cuanto a la estabilidad numérica, determinamos las condiciones bajo las cuales los métodos de Forward y Backward Euler preservan la estabilidad analítica. Vimos particularmente que el método de Backward Euler siempre da soluciones estables cuando el sistem original (lineal y estacionario) es estable, mientras que en Forward Euler la estabilidad numérica resulta de utilizar un paso de integración pequeño.

Tras esto, presentamos la familia de los algoritmos *monopaso* (también llamados métodos de Runge–Kutta). Estos métodos se caracterizan por conseguir órdenes de precisión altos en base a realizar varias evaluaciones de la función $\mathbf{f}(\mathbf{x}, t)$ en cada paso. Vimos, para estos algoritmos, como se puede variar el paso de integración de manera automática para controlar el error.

Luego, en forma breve, presentamos la familia de algoritmos *multipaso*. Estos métodos, para conseguir órdenes de precisión altos sin realizar múltiples evaluaciones de la función $\mathbf{f}(\mathbf{x}, t)$ en cada paso, utilizan los valores de las evaluaciones de dicha función en pasos anteriores.

Finalmente, analizamos tres clases de problemas que suelen aparecer en la práctica y que ofrecen dificultades a los algoritmos de integración numérica: los sistemas stiff (donde necesitamos usar métodos implícitos que preserven la estabilidad numérica), los sistemas marginalmente estables (que requieren utilizar algoritmos especiales como la regla trapezoidal) y por último los sistemas discontinuos (que requieren de técnicas especiales de detección y tratamiento de discontinuidades).

7. Lecturas para Profundización

Los contenidos de este apunte están principalmente basados en las notas de clases de la materia de doctorado *Simulación de Sistemas Continuos* [Kof06], que a su vez están basadas principalmente en el libro [CK06] (disponible en la biblioteca de la Facultad y de la Escuela de Ingeniería Electrónica).

Por otro lado, los libros [HNW00] y [HW91] son referencia obligada en la disciplina para quienes tengan interés en las propiedades más matemáticas de los algoritmos.

Finalmente, una colección muy importante de algoritmos y programas para implementar diversos métodos numéricos (no sólo para ecuaciones diferenciales, sino para problemas más generales) puede encontrarse en el libro clásico [PTVF07].

Referencias

[CK06] François Cellier and Ernesto Kofman. *Continuous System Simulation*. Springer, New York, 2006.

[HNW00] Ernst Hairer, Syvert Nørsett, and Gerhard Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer, Berlin, Germany, 2^{nd} edition, 2000.

[HW91] Ernst Hairer and Gerhard Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems.* Springer, Berlin, 1991.

[Kof06] Ernesto Kofman. Simulación de Sistemas Continuos. Notas de Clase. Departamento de Control, FCEIA, UNR. Disponible online en `http://www.fceia.unr.edu.ar/control/ssc/notas_ssc.pdf`, 2006.

[PTVF07] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, New York, 3^{rd} edition, 2007.