ECUACIONES DIFERENCIALES Y MÉTODOS NUMÉRICOS(Curso 2009-2010)

Cuarto Curso de Ingeniero Industrial

Problemas de Valor Inicial y de Contorno para Ecuaciones Diferenciales Ordinarias.

PROBLEMAS DE VALORES INICIALES: MÉTODOS DE UN PASO, MÉTODOS MULTIPASO.

Introducción. Las ecuaciones diferenciales son una de las herramientas matemáticas más usadas en el modelado de problemas técnicos y científicos. Puesto que la gran mayoría de las ecuaciones diferenciales que aparecen en dichos modelos no pueden resolverse analíticamente, la integración numérica de las ecuaciones diferenciales (ordinarias o en derivadas parciales) suele considerarse la parte más importante del Análisis Numérico. En este tema, vamos a estudiar y analizar métodos numéricos para resolver problemas de valores iniciales de sistemas de ecuaciones diferenciales ordinarias.

Cuestiones de Repaso. De modo general, escribiremos los sistemas de ecuaciones diferenciales ordinarias (EDOs) como:

$$\begin{aligned} y_1'(t) &= f_1(t, y_1(t), ..., y_m(t)), \\ \vdots \quad &\vdots \quad \vdots \\ y_m'(t) &= f_m(t, y_1(t), ..., y_m(t)), \end{aligned}$$

donde m es el número de ecuaciones (tantas como incógnitas $y_1(t), \ldots, y_m(t)$), y donde cada f_j es una función de $m+1$ argumentos (aunque podría no depender de alguno de ellos). El sistema anterior suele escribirse en notación más compacta como

$$y'(t) = f(t, y(t)),$$

donde

$$y(t) = \begin{bmatrix} y_1(t) \\ \vdots \\ y_m(t) \end{bmatrix} \in \mathbb{R}^m, \ f(t, y(t)) = \begin{bmatrix} f_1(t, y_1(t), \ldots, y_m(t)) \\ \vdots \\ f_m(t, y_1(t), \ldots, y_m(t)) \end{bmatrix} \in \mathbb{R}^m.$$

En general, y siempre que no de lugar a confusión, suprimiremos la referencia explícita a la dependencia de t.

En buena parte de las aplicaciones, las funciones o campos f son regulares, en el sentido de ser diferenciables con continuidad varias veces. Supondremos en esta lección que f es suficientemente

regular. Recuerde que las correspondientes soluciones y tienen una derivada (respecto de t) más que el número de veces que sea diferenciable f.

Recuerde así mismo que la soluciones pueden no existir para todo t, dependiendo de la naturaleza de f. Aquí nos centraremos exclusivamente en los problemas con soluciones definidas para todo t. Es bien conocido que un sistema de EDOs tiene infinitas soluciones, pero (si f es regular) solamente una que en un instante de tiempo tome un valor determinado. Se entiende por *problema de valor inicial* (PVI) o *problema de Cauchy* en un intervalo $[t_0, t_f]$, al dado por un sistema de EDOs y una condición inicial:

$$\begin{aligned} y'(t) &= f(t, y(t)),\ t \in [t_0, t_f], \\ y(t_0) &= y_0. \end{aligned}$$

Recuerde así mismo las nociones de *dependencia continua* de las condiciones iniciales. Si y y z son dos soluciones

$$y' = f(t, y),\ z' = f(t, z), \quad \text{tales que} \quad y(t), z(t) \in \Omega,\ \forall t \in [t_0, t_f],$$

donde Ω es una región de $\mathbb{R}^m$, y L constante de Lipschitz adecuada en $[t_0, t_f] \times \Omega$, como bien pueda ser una cota de la diferencial de f respecto de y, esto es

$$\|f_y(t, y)\| \leq L,\ \forall t \in [t_0, t_f],\ y \in \Omega, \tag{1}$$

donde

$$f_y = \begin{bmatrix} \frac{\partial f_1}{\partial y_1} & \cdots & \frac{\partial f_1}{\partial y_m} \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial y_1} & \cdots & \frac{\partial f_m}{\partial y_m} \end{bmatrix},$$

entonces se tiene que

$$\|y(t) - z(t)\| \leq e^{L(t-t_0)} \|y(t_0) - z(t_0)\|,\ t \in [t_0, t_f], \tag{2}$$

o dicho de otro modo, si f es suficientemente regular, *a datos iniciales próximos les corresponden soluciones próximas.*

El método de Euler. Como quiera que en la mayor parte de las ecuaciones diferenciales y sistemas de EDOs de interés no se puede obtener una solución expresable en términos de funciones elementales, de antaño se viene utilizando el método de Euler para obtener aproximaciones a una solución, método que le habrán contado en los cursos de ecuaciones diferenciales. Recuerde que es como sigue. Para el problema de Cauchy,

$$\begin{aligned} y'(t) &= f(t, y(t)),\ t \in [t_0, t_f], \\ y(t_0) &= y_0. \end{aligned}$$

se considera una partición

$$t_0 < t_1 < \ldots < t_N = t_f,$$

del intervalo $[t_0, t_f]$, y se obtienen aproximaciones

$$y_n \approx y(t_n),\ n = 1, 2, \ldots, N,$$

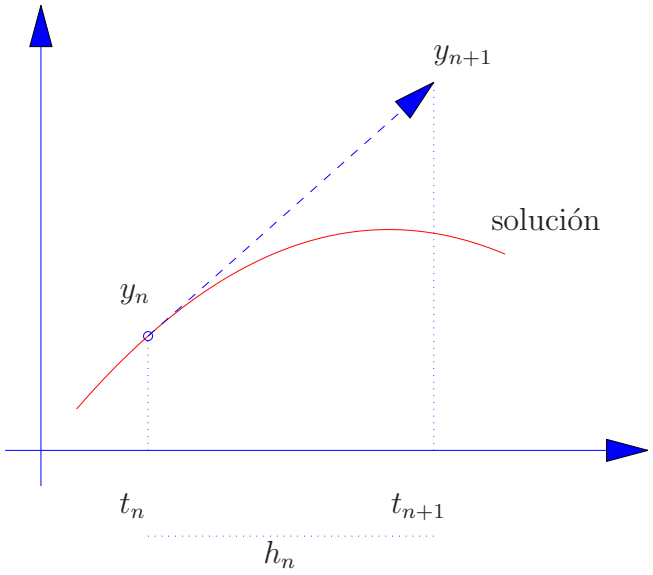
mediante la recurrencia

$$y_{n+1} = y_n + h_n f(t_n, y_n),\ n = 0, 1, \ldots, N-1,$$

donde

$$h_n = t_{n+1} - t_n,\ n = 0, 1, \ldots, N-1.$$

También recordará que la idea del método de Euler es, en cada subintervalo $[t_n, t_{n+1}]$, sustituir la curva solución por su recta tangente, tal y como indicamos en la figura.



El método de Euler converge, en el sentido de que tomando particiones cada vez más finas del intervalo $[t_0, t_f]$, las poligonales que obtenemos uniendo las aproximaciones y_n por segmentos se parecen cada vez más a la gráfica de la solución $y(t)$. De hecho, la correspondiente función lineal a trozos converge a la solución $y(t)$ en cada $t \in [t_0, t_f]$, tal y como se ve en la Fig 1, donde representamos una solución y, y las funciones lineales a trozos que construimos con las aproximaciones del método de Euler tomando particiones uniformes (con todos los incrementos h_j iguales a h) para valores de $h = (t_f - t_0)/4$, $h = (t_f - t_0)/8$ y $h = (t_f - t_0)/16$.

Si llamamos I_h a la función lineal a trozos obtenida a partir de las aproximaciones y_n, esto es:

$$I_h(t) = y_n + \frac{y_{n+1} - y_n}{t_{n+1} - t_n}(t - t_n),\ t \in [t_n, t_{n+1}],\ n = 0, \ldots, N-1,$$

se tiene que

$$\max_{t\in[t_0,t_f]} \|y(t) - I_h(t)\| \leq \frac{e^{L(t_f - t_0)} - 1}{2L} \max_{t\in[t_0,t_f]} \|y''(t)\|\, h, \tag{3}$$

donde

$$h = \max_{n=0,\ldots,N-1} h_n,$$

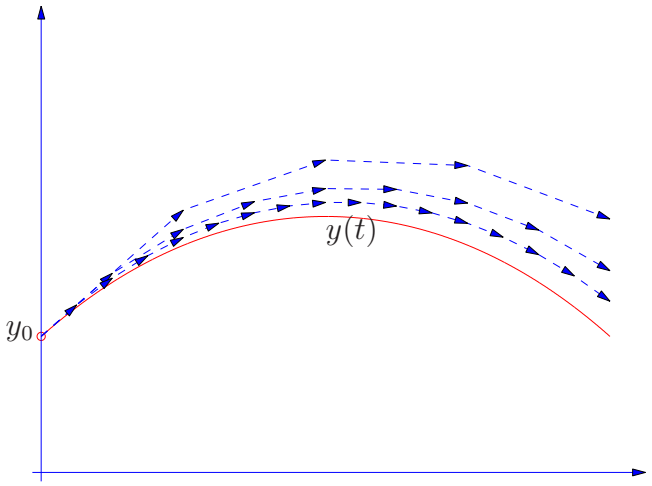


Figura 1: Aproximaciones obtenidas por el método de Euler con $h = (t_f - t_0)/4$, $h/2$ y $h/4$.

y L es una constante de Lipschitz adecuada como la indicada en (1).

En la Fig. 1 se aprecia que, al dividir h por dos, en igual proporción se divide la distancia entre el valor de las poligonales y la solución y.

Métodos numéricos. En general, los métodos numéricos explícitos de un paso para integrar PVI son de la forma:

$$_{n+1} = y_n + h_n \Phi(t_n, y_n, h_n),\ n = 0, 1, \ldots, N-1, \tag{4}$$

donde $\{t_0 < t_1 < \ldots < t_N = t_f\}$ es una partición del intervalo $[t_0, t_f]$. Se denominan así porque la aproximación $y_{n+1} \approx y(t_{n+1})$ sólo depende de la obtenida en el paso previo: y_n.

La obtención de este tipo de fórmulas parte de la igualdad

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} y'(t)\,dt = \int_{t_n}^{t_{n+1}} f(t, y(t))\,dt, \tag{5}$$

y a continuación se aproxima esta integral por alguna fórmula de cuadratura adecuada. Por ejemplo, el método de Euler se obtiene mediante la regla del rectángulo (que no vimos en la lección 5 por tosca):

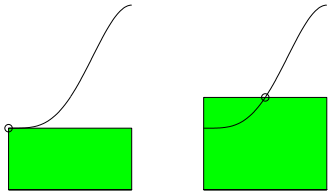
$$\int_{t_n}^{t_{n+1}} f(t, y(t))\,dt \approx \int_{t_n}^{t_{n+1}} f(t_n, y(t_n))\,dt = h_n f(t_n, y(t_n)),$$

en la que se aproxima el integrando por su valor en el extremo inferior del intervalo de integración:

$$f(t, y(t)) \approx f(t_n, y(t_n)),\ t \in [t_n, t_{n+1}].$$

La regla del rectángulo es una fórmula de cuadratura de grado 0 (sólo es exacta para funciones constantes). En cambio, la regla del punto medio es de grado 1 y, por tanto, en general es más

precisa, como podemos apreciar en este dibujo:



Rectángulo P. Medio

De esta forma, una idea para mejorar el método de Euler es utilizar la regla del punto medio, aproximando el integrando en (5) como

$$f(t, y(t)) \approx f(t_{n+1/2}, y(t_{n+1/2})),\ t \in [t_n, t_{n+1}],$$

donde $t_{n+1/2} = t_n + \dfrac{h_n}{2} = \dfrac{t_n + t_{n+1}}{2}$ es el punto medio del intervalo $[t_n, t_{n+1}]$.

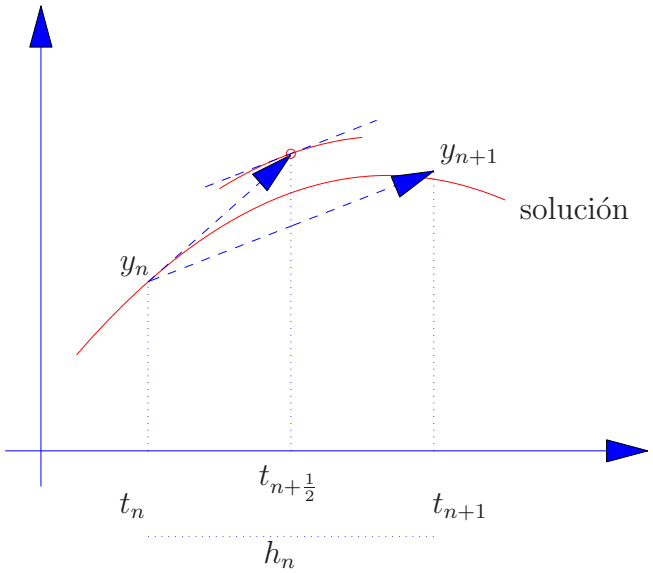
Notemos sin embargo que esto es inviable, puesto que disponemos de la aproximación a la solución $y(t)$ en el punto t_n, pero no de su valor en el punto medio $t_{n+1/2}$. La idea es entonces aproximar $y(t_{n+1/2})$ mediante el método de Euler, partiendo de $y(t_n)$ con un paso de longitud $h_n/2$, esto es:

$$f\left(t_n + \frac{h_n}{2}, y(t_n + \frac{h_n}{2})\right) \approx f\left(t_n + \frac{h_n}{2}, y_n + \frac{h_n}{2} f(t_n, y_n)\right).$$

Así obtenemos el *método de Euler mejorado* (o del punto medio):

$$\begin{aligned} y_{n+1} &= y_n + h_n k_2, \quad \text{donde} \\ k_1 &= f(t_n, y_n), \\ k_2 &= f(t_n + \frac{h_n}{2}, y_n + \frac{h_n}{2} k_1). \end{aligned}$$

Geométricamente, el método de Euler mejorado es análogo al método de Euler, pero ahora la pendiente de la recta es $f(t_{n+1/2}, y_n + \frac{h_n}{2} f(t_n, y_n))$ (para el método de Euler, dicha pendiente es $f(t_n, y_n)$):



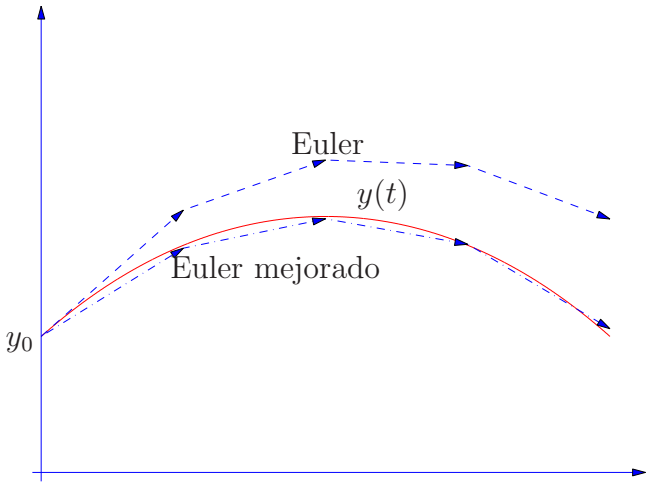


Figura 2: Aproximaciones obtenidas con los métodos de Euler y de Euler mejorado sobre una red uniforme de diámetro $h = (t_f - t_0)/4$.

donde, para el punto auxiliar $Y_1 = y_n + \frac{h_n}{2} f(t_n, y_n)$, hemos representado con un segmento de línea continua la pendiente $f(t_{n+1/2}, \tilde{y}_{n+1/2})$ dada por la ecuación diferencial $y' = f(t, y)$, para que se aprecie que es la misma que la del segmento que une y_n e y_{n+1}. El resultado es un método considerablemente más preciso, tal y como deja de manifiesto la Fig. 2.

Métodos Runge-Kutta. La idea del método de Euler mejorado (que en realidad se debe a Runge) esto es, utilizar una fórmula de cuadratura en $[t_n, t_{n+1}]$ y calcular (aproximar) los valores del integrando en los nodos mediante el método de Euler utilizado con pasos más pequeños, es la que subyace en buena parte de los Métodos Runge-Kutta. Así, *el método Runge-Kutta clásico* es de la forma,

$$\begin{aligned} y_{n+1} &= y_n + \tfrac{h_n}{6}(k_1 + 2k_2 + 2k_3 + k_4), \\ \\ k_1 &= f(t_n, y_n), \\ k_2 &= f(t_n + \tfrac{h_n}{2}, y_n + \tfrac{h_n}{2} k_1), \\ k_3 &= f(t_n + \tfrac{h_n}{2}, y_n + \tfrac{h_n}{2} k_2), \\ k_4 &= f(t_n + h_n, y_n + \tfrac{h_n}{k}_3). \end{aligned}$$

Para aproximar la integral en (5) este método utiliza la regla de Simpson,

$$\int_{t_n}^{t_{n+1}} f(t, y(t))\, dt \approx \frac{h_n}{6}\Big(f(t_n, y(t_n)) + 4f(t_n + \frac{h_n}{2}, y(t_n + \frac{h_n}{2})) + f(t_{n+1}, y(t_{n+1}))\Big),$$

y aproxima los valores del integrando como

$$f(t_n + \frac{h_n}{2}, y(t_n + \frac{h_n}{2})) \approx \frac{1}{2}(k_2 + k_3), \;\; f(t_{n+1}, y(t_{n+1})) \approx k_4.$$

Otros métodos Runge-Kutta son, por ejemplo, *el método de Heun*,

$$\begin{aligned} y_{n+1} &= y_n + \tfrac{h_n}{4}(k_1 + 3k_3), \\ \\ k_1 &= f(t_n, y_n), \\ k_2 &= f(t_n + \tfrac{h_n}{3}, y_n + \tfrac{h_n}{3}k_1), \\ k_3 &= f(t_n + \tfrac{2}{3}h_n, y_n + \tfrac{2}{3}h_n k_2), \end{aligned}$$

o *la regla de los 3/8*,

$$\begin{aligned} y_{n+1} &= y_n + \tfrac{h_n}{8}(k_1 + 3k_2 + 3k_3 + k_4), \\ \\ k_1 &= f(t_n, y_n), \\ k_2 &= f(t_n + \tfrac{h_n}{3}, y_n + \tfrac{h_n}{3}k_1), \\ k_3 &= f(t_n + \tfrac{h_n}{3}, y_n + \tfrac{h_n}{3}(3k_2 - k_1)), \\ k_4 &= f(t_n + h_n, y_n + h_n(k_1 - k_2 + k3)). \end{aligned}$$

Estos métodos son casos particulares de *métodos Runge-Kutta explícitos de s etapas*, que son de la forma:

$$\begin{aligned} y_{n+1} &= y_n + h_n(b_1 k_1 + \cdots + b_s k_s), \\ \\ k_1 &= f(t_n, y_n), \\ k_2 &= f(t_n + c_2 h_n,\ y_n + h_n a_{21} k_1), \\ k_3 &= f(t_n + c_3 h_n,\ y_n + h_n(a_{31}k_1 + a_{32}k_2)), \\ &\vdots \qquad\quad \vdots \\ k_s &= f(t_n + c_s h_n,\ y_n + h_n(a_{s1}k_1 + \cdots + a_{s,s-1}k_{s-1})). \end{aligned} \tag{6}$$

Así por ejemplo, el método de Euler mejorado es de dos etapas, el de Heun de tres etapas, y el método Runge-Kutta clásico y la regla de los 3/8 son de cuatro etapas. Y el método de Euler de una etapa.

Los coeficientes de un método Runge-Kutta suelen organizarse en el denominado *tablero de Butcher* del mismo, que es de la forma:

$$\begin{array}{c|ccccc} 0 & 0 & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ c_s & a_{s1} & a_{s2} & \dots & a_{s,s-1} & \\ \hline & b_1 & b_2 & \dots & b_{s-1} & b_s \end{array}$$

Así por ejemplo, los tableros del método de Euler, Euler mejorado, y Runge-Kutta clásico son

Euler

0	0
	1

Euler mejorado

0	0	
$\frac{1}{2}$	$\frac{1}{2}$	
	0	1

RK clásico

0	0			
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{1}{2}$	0	$\frac{1}{2}$		
1	0	0	1	
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

Convergencia. Error global y error local

En la Fig. 1 hemos visto cómo al tomar particiones cada vez más finas del intervalo $[t_0, t_f]$, las aproximaciones y_n distan cada vez menos de los verdaderos valores $y(t_n)$ que queremos aproximar.

Dada un problema de Cauchy

$$\begin{aligned} y'(t) &= f(t, y(t)),\ t \in [t_0, t_f], \\ y(t_0) &= y_0. \end{aligned}$$

y un método numérico

$$y_{n+1} = y_n + h_n \Phi(t_n, y_n, h_n),\ n = 0, 1, \ldots, N-1, \tag{7}$$

sobre una partición determinada $t_0 < t_1 < \ldots < t_N = t_f$, llamaremos *error global* o, simplemente, error, a la diferencias

$$y(t_n) - y_n,\ n = 0, 1, \ldots, N,$$

esto es, a la diferencia entre las aproximaciones numéricas y los valores de la solución exacta del PVI sobre los correspondientes puntos que definen la partición de $[t_0, t_f]$.

Se dice entonces que el método (7) es *convergente* si, para todo problema de Cauchy, con f suficientemente regular, se tiene

$$\max_{0 \le n \le N} \|y(t_n) - y_n\| \to 0, \text{ cuando } h = \max_{0 \le n \le N-1} h_n \to 0,$$

y se dice *convergente de orden p* cuando además se verifica que

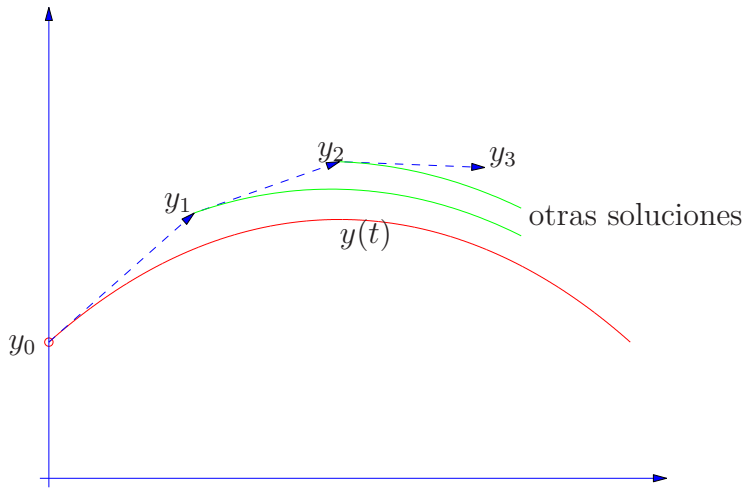
$$\max_{0 \le n \le N} \|y(t_n) - y_n\| = O(h^p).$$

recibiendo el valor p el nombre de *orden de convergencia* del método.

De acuerdo entonces con la cota (3), el método de Euler es de orden $p = 1$. Los órdenes de convergencia de los otros métodos vistos son los siguientes.

- **Método de Euler mejorado**, $p = 2$.
- **Método de Heun**, $p = 3$.
- **Método Runge-Kutta clásico**, $p = 4$.
- **Regla de los tres octavos**, $p = 4$.

Determinar cómo son los errores globales es tarea harto difícil, aunque no tanto acotar su valor. Para entender por qué decaen con h basta fijarse por ejemplo en la Fig. 2 para notar que los valores y_n solo coinciden en general con el valor $y(t_n)$ para el primer punto, $n = 0$. Por los demás puntos y_n pasarán en general otras soluciones (que corresponderán a condiciones iniciales en t_0 distintas de y_0).



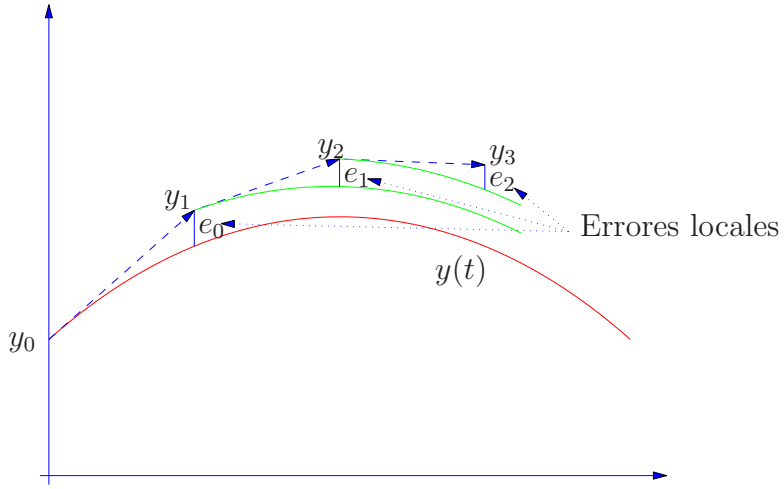
Al dar un paso de un un valor y_n al siguiente y_{n+1}, éste se desvía de la solución $z_n(t)$ de $z' = f(t, z)$ que toma valor y_n en t_n, esto es, la solución del problema

$$\left.\begin{array}{c} z'(t) = f(t, z(t)), \\ z(t_n) = y_n \end{array}\right\}, \ t \in [t_0, t_f]. \tag{8}$$

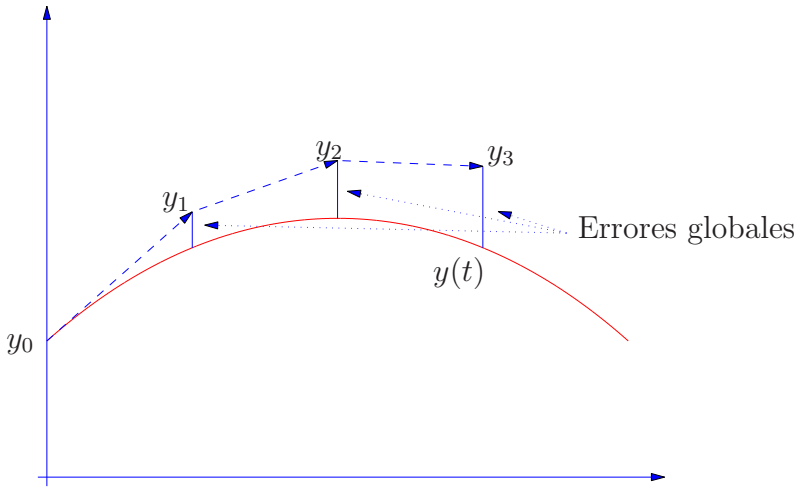
Recibe el nombre de *error local* en t_n la diferencia

$$e_n = z_n(t_{n+1}) - y_{n+1} = z_n(t_{n+1}) - y_n - h_n\Phi(t_n, y_n, h_n)), \ n = 0, 1, \ldots, N-1,$$

esto es, el error cometido en t_{n+1} con respecto a la solución que en t_n coincide con el valor numérico t_n,



que no se deben confundir con los errores globales $y(t_n) - y_n$,



Notando que también se tiene que

$$e_n = z_n(t_{n+1}) - z_{n+1}(t_{n+1}),\ n = 0, \ldots, N-1,$$

esto es, que la diferencia de dos soluciones de la ecuación diferencial en el instante t_{n+1}, en virtud de la dependencia continua respecto de las condiciones iniciales que vimos en (2), la diferencia de estas dos soluciones en otro instante posterior, digamos t_N se podrá acotar en términos de su valor en un instante anterior, tal y como indicamos el la Fig. 3 donde vemos los siguientes

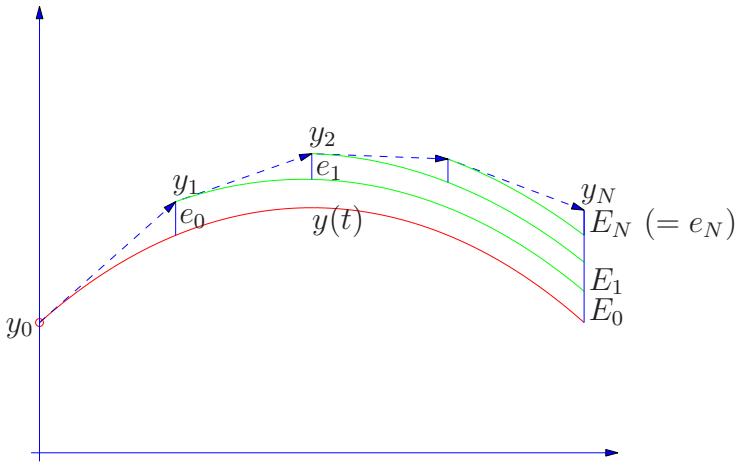


Figura 3: El abanico de Lady Windermere.

hechos.

i) El error global $y(t_N) - y_N$ es

$$y(t_N) - y_N = E_0 + E_1 + \cdots + E_{N-1},$$

ii) Los sumandos E_n son la diferencia de dos soluciones que en el instante t_{n+1} distan e_n, y de acuerdo con (2) serán por tanto,

$$\|E_n\| \leq e^{L(t_N - t_{n+1})} \|e_n\|,\ n = 0, 1, \ldots, N-1,$$

donde L será una constante de Lipschitz de f en una región donde se encuentran las soluciones z_n, $n = 0, \ldots, N-1$.

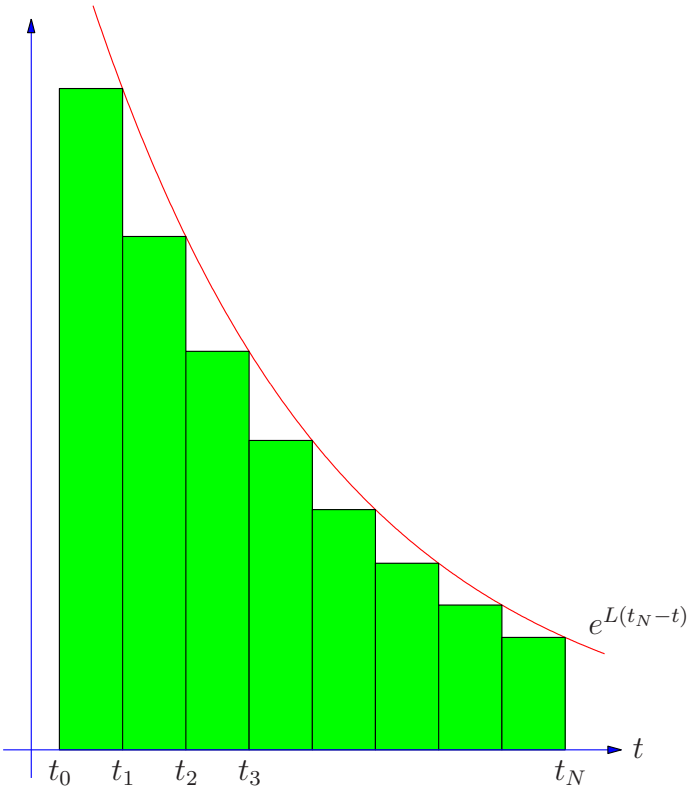
Por tanto, podemos acotar el error global como

$$\|y(t_N) - y_N\| \leq \sum_{n=0}^{n-1} e^{L(t_N - t_{n+1})} \|e_n\| .$$

Dividiendo y multiplicando cada término de esta suma por h_n tenemos que

$$\|y(t_N) - y_N\| \leq \sum_{n=0}^{n-1} e^{L(t_N - t_{n+1})} h_n \| \frac{e_n}{h_n} \| \leq \max_{0 \leq n \leq N-1} \| \frac{e_n}{h_n} \| \sum_{n=0}^{N-1} h_n e^{L(t_N - t_{n+1})}.$$

Observando que esta última suma es una suma inferior de Rieman de la integral de $e^{L(t_N - t)}$, en $[t_0, t_f]$



se tendrá por tanto que

$$\sum_{n=0}^{N-1} h_n e^{L(t_N - t_{n+1})} \leq \int_{t_0}^{t_N} e^{L(t_N - t)} \, dt = \frac{e^{L(t_N - t_0)} - 1}{L},$$

de donde se sigue que el error global en t_N se acota en términos de los errores locales (dividido cada uno por su incremento h_n correspondiente) como

$$\|y(t_N) - y_N\| \leq \frac{e^{L(t_N - t_0)} - 1}{L} \max_{0 \leq n \leq N-1} \| \frac{e_n}{h_n} \|.$$

Dado que en el procedimiento anterior podemos cambiar N por cualquier otro valor $n = 1, 2, \ldots, N-1$, obtenemos que

$$\max_{0\le n\le N} \|y(t_n) - y_n\| \le \frac{e^{L(t_f-t_0)} - 1}{L} \max_{0\le n\le N-1} \|\frac{e_n}{h_n}\|. \tag{9}$$

Vemos pues que *un método es convergente de orden p si sus errores locales son $O(h^{p+1})$*. Cuando los errores locales son $O(h^{p+1})$, se dice que el método es *consistente* de orden p. Para los métodos Runge-Kutta, pues, consistencia y convergencia son conceptos equivalente (no así para otros métodos).

Observando que el error local en t_n será el similar al de t_0 pero cambiando la solución z_n de (8) que satisface $z_n(t_n) = y_n$ por la solución y del PVI que se desea aproximar, basta pues estudiar el error local en t_0 para saber el orden de convergencia del método.

Ejemplo 1. Estudiemos el error local en t_0 del método de Euler.

$$e_0 = y(t_1) - y_1 = y(t_1) - (y_0 + h_0 f(t_0, y_0)) = y(t_1) - (y(t_0) + hy'(t_0)). \tag{10}$$

Por otro lado, calculando el desarrollo de Taylor de $y(t)$ en t_0,

$$y(t_0 + s) = y(t_0) + sy'(t_0) + \frac{s^2}{2} y''(\xi),$$

para algún ξ en $(t_0, t_0 + s)$. Cambiando en la expresión anterior s por h_0, y sustituyendo en (10) tenemos que

$$e_0 = \frac{h_0^2}{2} y''(\xi), \ \Rightarrow \ \|e_0\| = O(h_0^2) = O(h^2).$$

Nota 1. En general los errores locales de los métodos Runge-Kutta no tienen expresiones tan sencillas como la del método de Euler (haga el Ejercicio 7). Denotaremos entonces dichas expresiones genéricamente como $\Psi(y_n)$, esto es

$$e_n = h_n^{p+1} \Psi(y_n) + O(h^{p+2}),$$

entendiendo que cada método tiene una función Ψ diferente.

Nota 2. Cotas del error global más finas que las aquí presentadas, se pueden conseguir con un análisis similar basado en los *errores de truncamiento*. (ver por ejemplo [1], § II.3). El error de truncamiento en t_n es lo que le falta a la solución exacta $y(t)$ del PVI para satisfacer el método numérico. Para el método (7) es por tanto,

$$T_n \equiv y(t_{n+1}) - y(t_n) - h_n \Phi(t_n, y(t_n), h_n),$$

Es como el error local

$$e_n \equiv z_n(t_{n+1}) - y_n - h_n \Phi(t_n, y_n, h_n)),$$

o también,

$$e_n = z_n(t_{n+1}) - z_n(t_n) - h_n \Phi(t_n, z_n(t_n), h_n))$$

pero con $y(t)$ en vez de $z_n(t)$. El error de truncación se calcula del mismo modo que el error local, y de hecho, es el que se estudia para determinar el orden de un método.

Pares encajados de métodos Runge Kutta. Ya en los años 70 estaba claro que es poco práctico tomar todos los incrementos h_n iguales esto es $h_n = h = (t_f - t_0)/N$. La razón es que en la expresión del error local, hay puntos t_n donde $\Psi(y_n)$ toma valores mucho más grandes que en otros, debiéndose en aquellos equilibrar con un valor de h_n más pequeño que en estos segundos.

Hoy día el procedimiento que se ha probado como más eficaz es el de los *pares encajados de métodos Runge-Kutta*. Uno de estos pares consiste en dos métodos Runge-Kutta que comparten coeficientes c y A,

$$\begin{array}{c|c} c & A \\ \hline & b^T \\ \hline & \hat{b}^T \end{array} \text{ con } \begin{array}{c|c} c & A \\ \hline & b^T \end{array} \text{ de orden } p, \text{ y } \begin{array}{c|c} c & A \\ \hline & \hat{b}^T \end{array} \text{ de orden } \hat{p}.$$

Ejemplos de tales pares son por ejemplo el Runge-Kutta-Fehlberg RKF2(3)B, debido a Fehlberg en 1968, de tablero

$$\begin{array}{c|cccc} 0 & 0 & & & \\ \frac{1}{4} & \frac{1}{4} & & & \\ \frac{27}{40} & -\frac{189}{800} & \frac{729}{800} & & \\ 1 & \frac{214}{891} & \frac{1}{33} & \frac{650}{891} & \\ \hline b_i & \frac{214}{891} & \frac{1}{33} & \frac{650}{891} & 0 \\ \hline \hat{b}_i & \frac{533}{2106} & 0 & \frac{800}{1053} & -\frac{1}{78} \end{array}$$

con orden $p = 2$ y $\hat{p} = 3$, y y el DOPRI5(4), debido a Dormad y Prince en 1980 de tablero

$$\begin{array}{c|ccccccc} 0 & 0 & & & & & & \\ \frac{1}{5} & \frac{1}{5} & & & & & & \\ \frac{3}{10} & \frac{3}{40} & \frac{9}{40} & & & & & \\ \frac{4}{5} & \frac{44}{45} & -\frac{56}{15} & \frac{32}{9} & & & & \\ \frac{8}{9} & \frac{19372}{6561} & -\frac{25360}{2187} & \frac{64448}{6561} & -\frac{212}{729} & & & \\ 1 & \frac{9017}{3168} & -\frac{355}{33} & \frac{46732}{5247} & \frac{49}{176} & -\frac{5103}{18656} & & \\ 1 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & \\ \hline b_i & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & 0 \\ \hline \hat{b}_i & \frac{5179}{57600} & 0 & \frac{7571}{16695} & \frac{393}{640} & -\frac{92097}{339200} & \frac{187}{2100} & \frac{1}{40} \end{array}$$

con $p = 5$ y $\hat{p} = 4$.

En un par encajado siempre se utiliza el método de orden p para calcular las aproximaciones (avanzar), y el método de orden $\hat{p}$ para estimar el error local de la manera que mostraremos en breve. Originalmente, como en el par RKF23B, se tomaba $\hat{p} > p$. De esta manera se estimaba

realmente el error que se comete en los cálculos. Hoy día, sin embargo, se toma $p > \hat{p}$, pues se ha comprobado que es una opción más eficaz en la práctica.

Obtener $\hat{y}_{n+1} = y_n + h_n(\hat{b}_1 k_1 + \cdots + \hat{b}_s k_s)$ una vez obtenido y_{n+1} tiene poco costo, pues es combinar linealmente las etapas k_i ya calculadas para obtener y_{n+1}. Pero, como es frecuente hoy día, si $\hat{p} < p$, para los errores locales se tiene que,

$$\begin{aligned} z(t_{n+1}) - y_{n+1} &= \Psi(y_n)(h_n)^{p+1} + O((h_n)^{p+2}, \\ z(t_{n+1}) - \hat{y}_{n+1} &= \hat{\Psi}(y_n)(h_n)^{\hat{p}+1} + O((h_n)^{\hat{p}+2}, \end{aligned}$$

y restando,

$$\hat{y}_{n_1} - y_{n+1} = \hat{\Psi}(y_n)(h_n)^{\hat{p}+1} + O((h_n)^{\hat{p}+2},$$

se obtiene, como término dominante, el error local del método de orden más bajo $\hat{p}$. La norma de esta cantidad

$$\texttt{EST}_n = \|\hat{y}_{n_1} - y_{n+1}\|,$$

se toma como estimación del error local en t_n, y se compara con una tolerancia `TOL`. De ser más grande, no se acepta como válido el valor de y_{n+1} calculado, y se vuelve a calcular con otro valor de h_n. ¿Qué valor? El "razonamiento" es como sigue:

$$\begin{aligned} \text{tenemos que con } h_n, \quad \texttt{EST}_n \approx \|z(t_n + h_n) - \hat{y}_{n+1}(h_n)\| &\approx \|\hat{\Psi}(y_n)\|(h_n)^{\hat{p}+1} > \texttt{TOL}, \\ \text{y queremos que con } h'_n, \quad \texttt{EST}'_n = \|z(t_n + h'_n) - \hat{y}_{n+1}(h'_n)\| &\approx \|\hat{\Psi}(y_n)\|(h'_n)^{\hat{p}+1} = \texttt{TOL}, \end{aligned}$$

deberá ser

$$h'_n = \sqrt[\hat{p}+1]{\frac{\texttt{TOL}}{\|\hat{\Psi}(y_n)\|}} \approx h_n \sqrt[\hat{p}+1]{\frac{\texttt{TOL}}{\|\hat{\Psi}(y_n)\|(h_n)^{\hat{p}+1}}} \approx h_n \sqrt[\hat{p}+1]{\frac{\texttt{TOL}}{\texttt{EST}_n}}.$$

Por tanto, se toma,

$$h'_n = 0.9 h_n \sqrt[\hat{p}+1]{\frac{\texttt{TOL}}{\texttt{EST}_n}},$$

donde el factor 0.9 se coloca por seguridad, puesto que en el razonamiento anterior hemos despreciado los términos de orden superior.

Si por el contrario $\texttt{EST}_\texttt{n} \leq \texttt{TOL}$ el valor de y_{n+1} calculado se da por bueno y se procede a calcular y_{n+2} tras ajustar el paso para que la siguiente estimación $\texttt{EST}_{n+1}$ del error local salga con valor `TOL`, esto es,

$$h_{n+1} = \text{mín}(\ h_{\text{máx}}\ ,\ h_n\texttt{facmax}\ ,\ 0.9 h_n \sqrt[\hat{p}+1]{\frac{\texttt{TOL}}{\texttt{EST}_n}}\ ),$$

donde, también por precaución, se limita el paso h_{n+1} por un h_n máximo, y no se permite que un paso supere `facmax` veces el paso anterior.

También es frecuente en la práctica tomar como estimación una combinación de la estimación de los errores absoluto y relativo

$$\frac{\texttt{EST}_n}{1 + \frac{\texttt{RTOL}}{\texttt{TOL}}\|y_{n+1}\|},$$

o incluso expresiones como la anterior pero para cada una de las componentes de (el vector) $y_{n+1} \in \mathbb{R}^m$.

Ejemplo: Un modelo de reacción química. Tomaremos como ejemplo el sistema

$$\begin{aligned} u' &= A + u^2 v - (B+1)u, \\ v' &= Bu - u^2 v, \end{aligned} \tag{11}$$

conocido como *Brusselator*, que se trata de un modelo simplificado de reacción química. Para $A = 1$ y $B = 3$, este sistema tiene una solución periódica estable, representada en la Fig. 4.

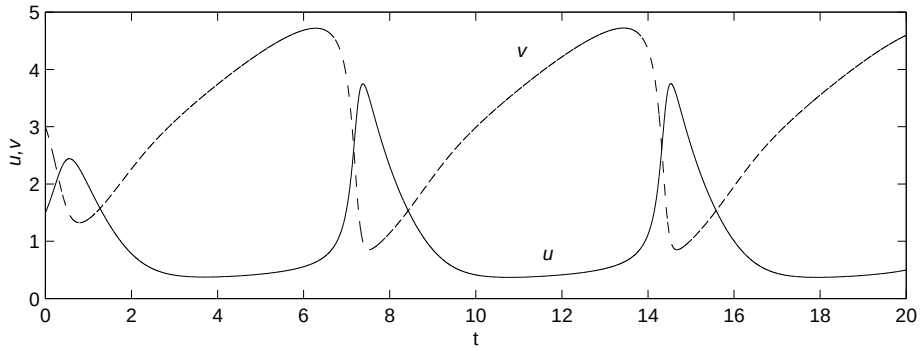


Figura 4: Solución del Brusselator con condiciones iniciales $u(0) = 1.5$ y $v(0) = 3$.

Aproximamos numéricamente dicha solución. En la Fig. 5 mostramos la diferencia que hay

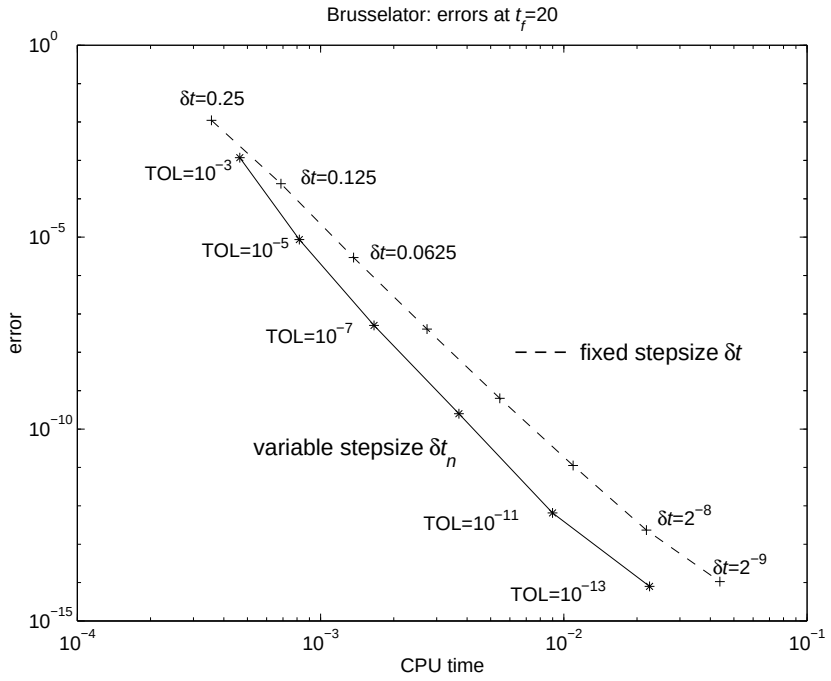


Figura 5: Diagrama de eficiencia para el Brusselator (11) con $u(0) = 1.5$ y $v(0) = 3$. Errores frente a costo para el par encajado DOPRI5(4) y el mismo método sin cambiar de paso

entre usar la técnica de paso variable descrita anteriormente y no usarla, en el caso del método

DOPRI5(4). Se ejecutó este método tomando valores de `TOL` iguales a 10^{-3}, 10^{-5}, 10^{-7}, 10^{-9}, 10^{-11} y 10^{-13}, y, para cada uno de estos cálculos, se midió el tiempo de ejecución (en segundos de CPU) y el *error global* $\|y(20) - y_N\|$ en $t = 20$. Cada par (segundos de CPU, error) aparece marcado en el gráfico con un $*$, y los diversos pares aparecen unidos por un segmento continuo para dar idea de qué hubiera ocurrido de haberse usado valores de `TOL` intermedios a los usados. Observamos que a medida que disminuimos la tolerancia `TOL`, los errores disminuyen, pero el costo aumenta pues el método debe tomar valores de h_n más pequeños para satisfacer el test de error, y por tanto dar más pasos para llegar a $t = 20$. Notamos sin embargo que de usar $\texttt{TOL} = 10^{-3}$ a usar $\texttt{TOL} = 10^{-13}$ el error se divide aproximadamente por 10^{-11}; sin embargo, el costo sólo se multiplica por 50.

También aparecen representadas en la Fig. 5 los resultados de ejecutar el método de orden 5 del par DOPRI5(4) con paso fijo (marcados con cruces y unidos por línea de trazos). Se usaron los valores de $h_n = 2^{-2}, 2^{-3}, \ldots, 2^{-9}$. Vemos que efectivamente el método de paso variable es más eficiente que el método de paso fijo (salvo quizá para errores mayores que 10^{-2}) pues seleccionando un nivel de error inferior a 10^{-3} (esto es trazando una línea horizontal en el dibujo), siempre está más a la izquierda el resultado de paso variable (línea continua) que el de paso fijo (línea de trazos), refejando el hecho de que para alcanzar dicho error, el método de paso variable requiere menos costo (tiempo de CPU) que el de paso fijo.

La razón de este fenómeno puede verse en la Fig. , donde se muestran las longitudes de

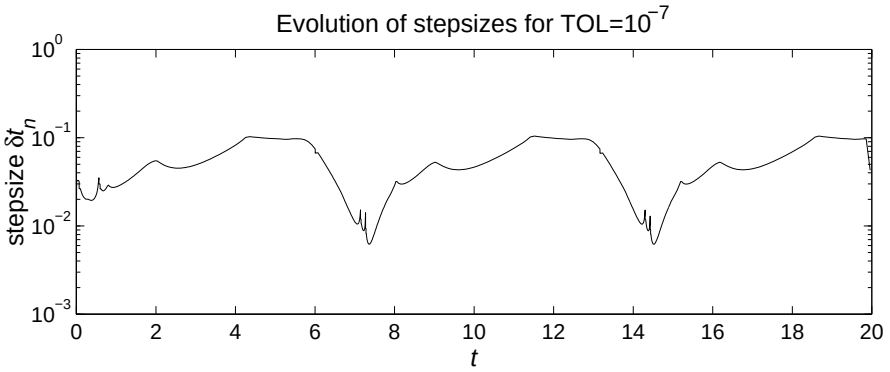


Figura 6: Evolución de las longitudes de paso h_n para $\texttt{TOL} = 10^{-7}$

paso h_n que el método DOPRI5(4) tomo con tolerancia $\texttt{TOL} = 10^{-7}$. Observando a la vez la Fig. 4, vemos que en aquellas partes en las que la solución presenta cambios más acusados y por tanto derivadas (de las que dependen el tamaño de los errores locales) el método toma valores de h_n más pequeños para ajustar el tamaño del error local al valor de la tolerancia `TOL`, mientras que en aquellas partes más lisas aumenta la longitud de paso h_n.

Mostramos por último una comparación de los pares DOPRI5(4) y RKDF23B en la Fig. 7. Puede apreciarse cómo a medida que se demanda mayor precisión el método de orden 5 es cada vez más ventajoso sobre el de orden 2.

Comandos de MATLAB**.** Hay diversos comandos de MATLAB para resolver sistemas de ODEs. Salvo que el PVI que queramos resolver tenga alguna característica específica que aconseje lo contrario, el comando que se recomienda es `ode45`, o en su defecto, `ode23`. El comando `ode45` integra un PVI utilizando el par encajado DOPRI5(4) de Dormand y Prince visto en el apartado

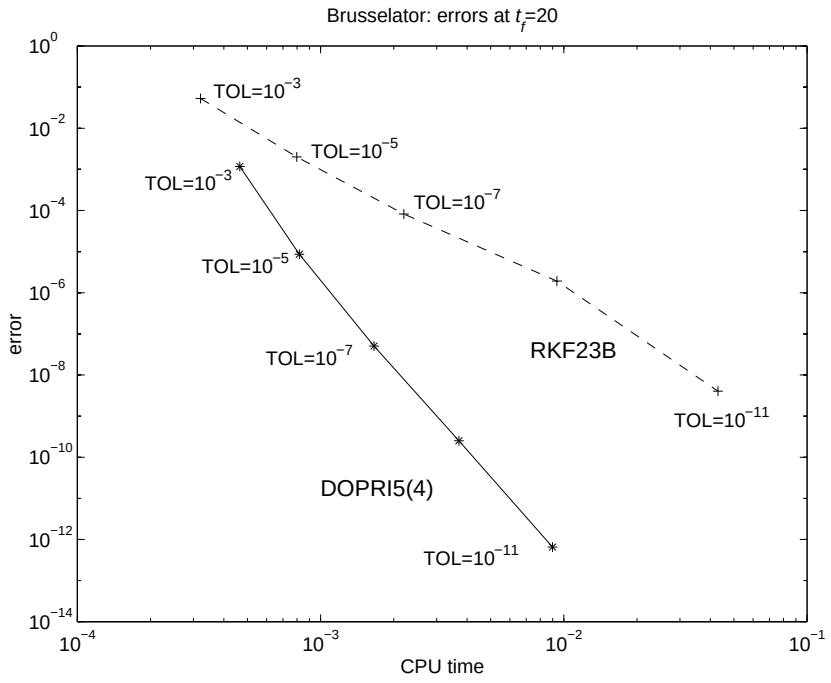


Figura 7: Diagrama de eficiencia para el Brusselator (11) con $u(0) = 1.5$ y $v(0) = 3$. Errores frente a costo para los pares encajados DOPRI5(4) y RKF23B

anterior, esto es, avanza (obtiene las aproximaciones y_n) con un método de orden 5 mientras que el comando `ode23` lo hace con un método de segundo orden debido a Bogacki y Shampine.

Ambos comandos tienen diversas maneras de ejecutarse. De todas ellas aquí nos centraremos en dos. Para un problema de Cauchy

$$\begin{aligned} y'(t) &= f(t, y(t)),\ t \in [t_0, t_f], \\ y(t_0) &= y_0. \end{aligned}$$

se ejecutan del siguiente modo

```
[T,Y]=ode45(@funcion,tiempos,y0)
```

o

```
[T,Y]=ode45(@funcion,tiempos,y0,opciones)
```

donde

`funcion` es el nombre de un fichero que contiene la función de MATLAB que, dados t e y, devuelve el valor $f(t, y)$ del segundo miembro del sistema de EDOs $y' = f(t, y)$.

`tiempos` o bien es el vector $[t_0, t_f]$ (o $[t_0, t_f]^T$) con los extremos del intervalo de integración, o bien es un vector $[t_0, t_1, \ldots, t_l]$ (o $[t_0, t_1, \ldots, t_l]^T$), cuyas componentes constituyen una partición

$$t_0 < t_1 < \ldots < t_l = t_f,$$

de dicho intervalo. En el primer caso, el comando `ode45` devuelve en `T` una partición del intervalo $[t_0, t_f]$ de su propia conveniencia, y en el segundo, el comando `ode45` devuelve en `T` la misma partición que la contenida en el vector `tiempos`.

`y0` vector con la condición inicial y_0 problema de Cauchy.

`opciones` estructura creada con el comando `odeset` donde podemos indicar, entre otros, los valores de TOL y $RTOL$ de las tolerancias para el control del error local que indicamos en la sección anterior.

`T` vector columna cuyo contenido hemos comentado al explicar el argumento `tiempos`.

`Y` matriz de m columnas (tantas como componentes tenga y_0) y tantas filas como componentes tenga el vector `T`, con las aproximaciones (traspuestas, pues se guardan por filas) en los valores de las componentes de `T` a la solución y del problema de Cauchy.

Ejemplo 2. Tomemos el sistema del Brusselator visto al principio de la lección,

$$\begin{array}{rcl} u' & = & A + u^2 v - (B+1)u, \\ v' & = & Bu - u^2 v \end{array} \tag{12}$$

con $A = 1$ y $B = 3$, en el intervalo $[0, 20]$, y condición incial

$$y_0 = \left[\begin{array}{c} u(0) \\ v(0) \end{array} \right] = \left[\begin{array}{c} 1.5 \\ 3 \end{array} \right]. \tag{13}$$

Para poder integrar numéricamente este problema, construimos la función de MATLAB que evalúe el segundo miembro,

```
function f=bruss(t,y)
% segundo miembro del Brusselator
p=y(2)*y(1)^2;
f=[1+p-4*y(1); 3*y(1)-p];
```

Si lo que queremos es una representación gráfica de las dos componentes de la solución, podemos tomar una partición del intervalo $[0, 20]$ de 400 subintervalos,

```
tiempos=[0:0.05:20];
```

calcular una aproximación numérica en los puntos de esta partición a la solución $y(t) = [u(t), v(t)]^T$ de (12), y dibujar la gráfica de ambas componentes u y v,

```
[T,Y]=ode45(@bruss,tiempos,[1.5 3]');
clf;subplot(2,1,1);plot(T,Y(:,1));hold;plot(T,Y(:,2),'--')
xlabel('t');ylabel('u,v')
text(11.5,0.9,'u');text(11,4.5,'v')
```

y obtenemos el gráfico en la Fig. 4 de esta lección (con los comandos `xlabel` e `ylabel` escribimos en el gráfico qué representa cada eje, y el comando `text` escribe las palabras que vayan entre apóstrofes en su tercer argumento en el punto del gráfico cuyas coordenadas indiquemos en los dos primeros argumentos). Si queremos saber con qué tolerancias TOL para el valor absoluto del error local y $RTOL$ para el valor relativo del mísmo ha controlado la longitud de paso el comando `ode45`, podemos averiguar los valores que utiliza por defecto el comando `ode45` ejecutando el comando `odeset` sin argumentos,

```
odeset
          AbsTol: [ positive scalar or vector {1e-6} ]
          RelTol: [ positive scalar {1e-3} ]
     NormControl: [ on | {off} ]
       OutputFcn: [ function ]
       OutputSel: [ vector of integers ]
          Refine: [ positive integer ]
           Stats: [ on | {off} ]
     InitialStep: [ positive scalar ]
         MaxStep: [ positive scalar ]
             BDF: [ on | {off} ]
        MaxOrder: [ 1 | 2 | 3 | 4 | {5} ]
        Jacobian: [ matrix | function ]
        JPattern: [ sparse matrix ]
      Vectorized: [ on | {off} ]
            Mass: [ matrix | function ]
MStateDependence: [ none | weak | strong ]
       MvPattern: [ sparse matrix ]
    MassSingular: [ yes | no | {maybe} ]
    InitialSlope: [ vector ]
          Events: [ function ]
```

que devuelve las posibles opciones que se le pueden dar al comando `ode45` así como los valores que utiliza por defecto (que aparecen indicados entre llaves en caso de haberlos). De este modo vemos que, en los cálculos que hemos realizado, el valor de TOL utilizado es el que figura entre llaves en la opción `AbsTol`, que es 10^{-6}, y el valor de $RTOL$ ha sido de 10^{-3} al ser éste el que figura entre llaves en la opción `RelTol`.

Si quisiésemos ahora obtener las aproximaciones con $TOL = RTOL = 10^{-8}$ podemos crear nuestras propias opciones, también con el comando `odeset`, del siguiente modo

```
misopciones=odeset('AbsTol',1e-8,'RelTol',1e-8,'Stats','on');
```

donde también hemos activado la opción `'Stats'` para que nos informe de cuantos pasos da, cuantos rechaza, etc. Así, si tras ejecutar el comando anterior, ejecutamos ahora,

```
[T08,Y08]=ode45(@bruss,tiempos,[1.5 3]',misopciones);
```

aparte de devolver en $Y08$ las nuevas aproximaciones obtenidas, muestra por pantalla también la siguiente información,

```
336 successful steps
7 failed attempts
2059 function evaluations
0 partial derivatives
0 LU decompositions
0 solutions of linear systems
```

que nos indica que ha intentado 336+7 veces dar un paso, y que en siete ocasiones lo ha tenido que rechazar por ser la estimación mayor que la tolerancia dada.

Note que el número de pasos dado, 336, es inferior al número de componentes del vector `T`, 401, que son los valores de la variable t a los que corresonden los valores de Y. La razón es que `ode45` calcula aproximaciones y_n en instantes t_n que se obtienen según la estrategia de cambio de paso indicada en la sección anterior. Dado que estos valores t_n no coincidirán en general con los valores en el vector `tiempos` (que son los mismos que los de `T`), el comando `ode45` calcula mediante interpolación las aproximaciones que devuelve en `Y`. Esta aproximación es de orden 4 (véase [1], § II.5).

Por último, si quisiésemos calcular una aproximación todavía más precisa, podemos cambiar los valores de las tolerancias del siguiente modo

```
misopciones=odeset(misopciones,'AbsTol',1e-12,'RelTol',1e-12)
```

Note que al poner `misopciones` como primer argumento del comando `odeset` no creamos unas opciones nuevas, sino simplemente *cambiamos* los valores indicados de `AbsTol` y `RelTol`, respetando las demás opciones que hubiésemos fijado anteriormante (si no hubiésemos puesto `misopciones` como primer argumento, habría tomado el valor por defecto de la opción `Stats`, que es `off`.

Si ahora ejecutamos

```
[T12,Y12]=ode45(@bruss,tiempos,[1.5 3]',misopciones);
```

obtenemos unos valores de $y(t)$ más precisos que los anteriores. Podemos hacernos idea del error cometido con las aproximaciones `Y` e `y08` comparándolas con `Y12`. De esta manera, si ejecutamos

```
erru08=max(abs(Y12(:,1)-Y08(:,1))./abs(Y12(:,1)));
errv08=max(abs(Y12(:,2)-Y08(:,2))./abs(Y12(:,2)));
[erru08,errv08]
```

obtenemos

```
ans =
   1.0e-06 *
    0.1817    0.2370
```

errores 20 veces mayores que los valores de las tolerancias usados en el cálculo de `Y08`, y si ejecutamos

```
erru06=max(abs(Y12(:,1)-Y(:,1))./abs(Y12(:,1)));
errv06=max(abs(Y12(:,2)-Y(:,2))./abs(Y12(:,2)));
[erru06,errv06]
```

obtenemos

```
ans =
    0.1817    0.2370
```

valores 100 veces más grandes que el valor de $RTOL = 10^{-3}$ empleado en el cálculo de `Y`.

Este ejemplo aconseja pues no relacionar con demasiada alegría los valores de las tolerancias empleados y el error globlal correspondiente.

Otros métodos: Métodos lineales multipaso. De los llamados métodos lineales multipaso, solamente haremos unos breves comentarios, pues por un lado, el desarrollo en las últimas décadas de los pares encajados de métodos Runge-Kutta han hecho que este tipo de métodos pierdan mucha competitividad y, por otro, un estudio más detallado aporta poco en el orden práctico a lo que se haya podido aprender en la secciones anteriores. A su vez, estos métodos están muy bien documentados en multitud de libros de texto. Debe saber sin embargo que para algunos problemas muy determinados y concretos, presentan ventajas que, a día de hoy, ningún método Runge-Kutta ha conseguido igualar. Lo particular de estas situaciones no justifica, en todo caso, que vayamos más alla de la descripción somera que haremos en esta sección.

Métodos de Adams Son anteriores a los métodos Runge-Kutta y se deben a los británicos Adams, Bashforth (1883, 1885) y Moulton (1926).

Recordemos que dados el PVI

$$\begin{aligned} y'(t) &= f(t, y(t)),\ t \in [t_0, t_f], \\ y(t_0) &= y_0. \end{aligned}$$

y una partición

$$t_0 < t_1 < \ldots < t_N = t_f,$$

del intervalo $[t_0, t_f]$, la solución y satisface la igualdad

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} y'(t)\, dt = \int_{t_n}^{t_{n+1}} f(t, y(t))\, dt. \tag{14}$$

Los métodos Runge-Kutta aproximan esta última integral mediante una fórmula de cuadratura con nodos en el intervalo $[t_0, t_f]$ (y aproximan los valores del integrando con las etapas k_i). Los métodos de Adams aproximan dicha integral con una fórmula de cuadratura basada en los

$$\text{nodos de cuadratura } t_{n-k+1}, \ldots, t_n \text{ o } t_{n-k+1}, \ldots, t_n, t_{n+1}.$$

donde k es *el número de pasos* del método. Para ello, obviamente se requiere haber obtenido (con otro método) esas k aproximaciones anteriores. Si $P^*_{n,k}$ el polinomio de grado $k-1$ que interpola a los k valores del campo $f(t, y)$ en las aproximaciones numéricas $(t_n, y_n), \ldots, (t_{n-k+1}, y_{n-k+1})$, esto es,

$$P^*_{n,k}(t) = \sum_{j=0}^{k-1} \Big(\prod_{l=0}^{j-1} (t - t_{n-l}) \Big) f[t_n, \ldots, t_{n-j}],$$

donde $f[t_n, \ldots, t_{n-j}]$ son las diferencias divididas definidas recurrentemente como

$$\begin{aligned} f[t_{n-j}] &= f(t_{n-j}, y_{n-j}),\ j = 0, \ldots, k-1, \\ f[t_n, \ldots, t_{n-j}] &= \frac{f[t_n, \ldots, t_{n-j+1}] - f[t_{n-1}, \ldots, t_{n-j}]}{t_n - t_{n-j}},\ j = 1, 2, \ldots . \end{aligned}$$

el correspondiente método de k-pasos es entonces

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} P_{n,k}^*(t)\, dt. \tag{15}$$

y se conoce como *método de Adams explícito de k pasos* (más conocido también como método Adams-Bashforth de k pasos, aunque no se deba a Bashforth).

También podemos considerar el polinomio de grado k que interpola k valores del campo $f(t, y)$ en las aproximaciones numéricas $(t_n, y_n), \ldots, (t_{n-k+1}, y_{n-k+1})$ anteriores, junto con el el valor de f en (t_{n+1}, y_{n+1}), aunque no lo conozcamos,

$$P_{n,k}(t) = P_{n,k}^* + \Big(\prod_{l=0}^{k-1}(t - t_{n-l})\Big) f[t_{n+1}, t_n, \ldots, t_{n-k+1}].$$

El correspondiente método numérico será entonces

$$\begin{aligned} y_{n+1} - h_n \beta_{n,k} f(t_{n+1}, y_{n+1}) &= y_n + \int_{t_n}^{t_{n+1}} P_{n,k}^*(t)\, dt \\ &\quad + f^{(0)}[t_{n+1}, t_n, \ldots, t_{n-k+1}] \int_{t_n}^{t_{n+1}} \prod_{l=0}^{k-1}(t - t_{n-l})\, dt, \end{aligned} \tag{16}$$

donde

$$\beta_{n,k} = \frac{1}{h_n} \int_{t_n}^{t_{n+1}} \prod_{l=0}^{k-1} \frac{t - t_{n-l}}{t_{n+1} - t_{n-l}}\, dt,$$

y $f^{(0)}[t_{n+1}, t_n, \ldots, t_{n-k+1}]$ es la diferencia dividida que se obtiene cuando se cambia y_{n+1} por 0. Observe que en (16), la incógnita y_{n+1} aparece a la izquierda del signo igual, y que lo que hay a la derecha solamente depende de los valores conocidos $y_n, \ldots, y_{n-k+1}$. El correspondiente método se conoce como *método de Adams implícito de k pasos*, o también de Adams-Moulton de k pasos.

Observe que para obtener el valor de y_{n+1} hay que resolver la ecuación (o sistema de ecuaciones) dado por la igualdad en (16). En la práctica esto se hace por iteración de punto fijo, procedimiento que convergerá siempre que h_n sea suficientemente pequeño.
Debe saber lo siguiente.

- El método de Adams explícito de un paso es el método de Euler

$$y_{n+1} - y_n = h_n f(t_n, y_n).$$

- El método de Adams implícito de un paso se conoce como *la regla de los trapecios* (igual que la fórmula de cuadratura) y es de la forma

$$y_{n+1} - y_n = \frac{h_n}{2}(f(t_n, y_n) + f(t_{n+1}, y_{n+1})).$$

- El método de Adams explícito de k pasos es convergente de orden k, y el implícito de k pasos de orden $k+1$ (para recordarlo, puede Vd. pensar en los casos $k = 1$).

- En la práctica se suelen emplear ambos métodos juntos, en lo que se conoce como *pares predictor-corrector*. Se utiliza el método explícito para obtener un valor de y^*_{n+1} que se toma como iterante inicial en la iteración de punto fijo necesaria para encontrar el el valor de y_{n+1} del método implícito.

- Existen procedimientos en los pares predictor-corrector para estimar el error local, con lo que generalmente se emplean con paso variable.

- De hecho, se pueden estimar los errores locales correspondientes a los métodos de $k-1$ pasos y de $k+1$ pasos, por lo que los métodos de Adams se emplean en algoritmos de paso y orden variables.

- El comando de MATLAB que utiliza los métodos de Adams con paso y orden variable es `ode113`. Empezando con $k = 1$, puede llegar a utilizar fórmulas de hasta $k = 12$ pasos.

Fórmulas de diferenciación regresiva (BDF). Las fórmulas de diferenciación regresiva (en inglés "backward differentiation formulae" (BDF)), son los métodos multipaso que se utilizan para problemas "stiff" que explicaremos en la sección siguiente. Datan de 1952, y fueron inventadas por Curtiss y Hirschfelder.

Se definen como sigue. Supongamos conocidas las k aproximaciones $y_{n-k+1}, \ldots, y_n$. Para obtener la aproximación y_{n+1} con la fórmula BDF de k pasos, se impone la condición de que el polinomio $Q_{n,k}$ de grado k que interpola a los $k+1$ pares $(t_{n-k+1}, y_{n-k+1}), \ldots, (t_{n+1}, y_{n+1})$ (incluímos el desconocido y_{n+1}) satisfaga la ecuación diferencial en t_{n+1}, esto es

$$Q'_{n,k}(t_{n+1}) = f(t_{n+1}, Q_{n,k}(t_{n+1})) = f(t_{n+1}, y_{n+1}).$$

El polinomio $Q_{n,k}$, escrito en forma de Newton es

$$Q_{n,k}(t) = \sum_{j=0}^{k} \Big(\prod_{l=0}^{j-1} (t - t_{n+1-l}) \Big) y[t_{n+1}, \ldots, t_{n+1-j}],$$

donde $y[t_{n+1}, \ldots, t_{n+1-j}]$ es la diferencia dividida basada en las aproximaciones $y_{n+1}, \ldots, y_{n+1-j}$, esto es

$$\begin{aligned} y[t_{n-j}] &= y_{n-j},\ j = -1, 0, 1, \ldots, k-1, \\ y[t_{n+1}, \ldots, t_{n-j+1}] &= \frac{y[t_{n+1}, \ldots, t_{n-j+2}] - y[t_n, \ldots, t_{n-j+1}]}{t_{n+1} - t_{n-j+1}},\ j = 1, 2, \ldots . \end{aligned}$$

Derivando $Q_{n,k}$, evaluando en t_{n+1}, igualando a $f(t_{n+1}, y_{n+1})$ y multiplicando por $h_n = t_{n+1} - t_n$ obtenemos la expresión de la fórmula BDF de k pasos

$$h_n \sum_{j=1}^{k} \Big(\prod_{l=1}^{j-1} (t - t_{n+1-l}) \Big) y[t_{n+1}, \ldots, t_{n+1-j}] = h_n f(t_{n+1}, y_{n+1}). \tag{17}$$

Si dejamos a un lado del signo igual la incógnita y_{n+1} y al otro el resto, tenemos que el método se escribe como

$$\alpha_{n,k} y_{n+1} - h_n f(t_{n+1}, y_{n+1}) = g_{n,k}(y_n, \ldots, y_{n-1}), \tag{18}$$

donde, obviamente,

$$\alpha_{n,k} = \sum_{j=1}^{k} \Big(\prod_{l=1}^{j-1} \frac{t - t_{n+1-l}}{t_{n+1} - t_{n+1-l}} \Big),$$

y

$$g_{n,k}(y_n, \ldots, y_{n-1}) = h_n \sum_{j=1}^{k} \Big(\prod_{l=1}^{j-1} (t - t_{n+1-l}) \Big) y^{(0)}[t_{n+1}, \ldots, t_{n+1-j}],$$

donde, igual que en el apartado anterior, $y^{(0)}[t_{n+1}, \ldots, t_{n+1-j}]$ denota la diferencia dividida que se obtiene al cambiar y_{n+1} por 0.

Para encontrar y_{n+1} se resuelve la ecuación en (18) siempre *por el método de Newton*. Esto es partiendo de una aproximación inicial $y_{n+1}^{[0]}$ (normalmente obtenida por extrapolación de los valores $y_n, \ldots, y_{n-k+1}$), se obtienen aproximaciones sucesivas $y_{n+1}^{[\mu]}$,

$$y_{n+1}^{[\mu+1]} = y_{n+1}^{[\mu]} + x^{[\mu]}, \ \mu = 0, 1, 2, \ldots,$$

donde $x^{[\mu]}$ es la solución del sistema lineal

$$(\alpha_{n,k} I - h_n f_y) x^{[\mu]} = -(\alpha_{n,k} y_{n+1}^{[\mu]} - h_n f(t_{n+1}, y_{n+1}^{[\mu]}) - g_{n,k}(y_n, \ldots, y_{n-k+1})), \tag{19}$$

siendo f_y la matriz jacobiana

$$f_y = \begin{bmatrix} \dfrac{\partial f_1}{\partial y_1} & \cdots & \dfrac{\partial f_1}{\partial y_m} \\ \vdots & & \vdots \\ \dfrac{\partial f_m}{\partial y_1} & \cdots & \dfrac{\partial f_m}{\partial y_m} \end{bmatrix},$$

evaluada en $(t_{n+1}, y_{n+1}^{[\mu]})$.

Por tanto, para poder utilizar el software que emplea este método, no solo hay que proporcionar el segundo miembro f, sino tambien la matriz diferencial f_y.
Debe saber lo siguiente.

- La fórmula BDF de $k = 1$ paso, es el *método de Euler implícito*

$$y_{n+1} - y_n = h_n f(t_{n+1}, y_{n+1}).$$

- Siempre que $k \leq 6$, la fórmula BDF de k pasos es convergente de orden k.
- Igual que los métodos de Adams, se utilizan en estrategias de orden y paso variables.
- Las fórmulas BDF son en general menos eficientes que los métodos de Adams (y por supuesto, que los pares encajados modernos de métodos RK) para problemas no stiff. Por eso, sólo se emplean en problemas stiff.
- El comando de MATLAB que utiliza estas fórmulas es `ode15s`. En realidad, utiliza unas fórmulas similares a las BDF, debidas a Klpfestein y Reiher en 1971 y 1978, denominadas NDF[1] (del inglés, "numerical differentiation formulae") y supuestamente más eficientes que las fórmulas BDF. Dado que a día de hoy esa mejor eficiencia no es manifiestamente clara, el comando `ode15s` permite utilizar las fórmulas BDF siempre que inicialice la opción `BDF` con el valor `'on'` mediante el comando `odeset`.

Ejemplo 3. (Continuación del Ejemplo 2 de la Sección). Para la función del segundo miembro del Brusselator,

$$f(t,y) = f(u,v) = \begin{bmatrix} f_1(u,v), \\ f_2(u,v) \end{bmatrix} = \begin{bmatrix} A + u^2 v - (B+1)u, \\ Bu - u^2 v \end{bmatrix}, \; y = \begin{bmatrix} u \\ v \end{bmatrix},$$

la matriz jacobiana f_y es

$$f_y = \begin{bmatrix} \frac{\partial f_1}{\partial u} & \frac{\partial f_1}{\partial v} \\ \frac{\partial f_2}{\partial u} & \frac{\partial f_2}{\partial v} \end{bmatrix} = \begin{bmatrix} 2uv - (B+1) & u^2 \\ B - 2uv & -u^2 \end{bmatrix}.$$

Si queremos emplear el comando `ode15s` debemos elaborar una función que dados t e y devuelva esta matriz jacobiana. Para los valores de A y B empleados hasta ahora, $A = 1$ y $B = 3$, dicha función es

```
function J=brussjac(t,y)
% Matriz jacobiana del segundo miembro del Brusselator
p=2*y(2)*y(1);q=y(1)*y(1);
J=[p-4, q; 3-p, -q];
```

Para utilizar las fórmulas BDF, en las correspondientes opciones debemos indicar el nombre del fichero que contiene esta función (en nuestro caso `brussjac.m`) sin la extensión `.m`. Inicializamos las opciones para utilizar `ode15s`,

```
opbdf=odeset('AbsTol',1e-8,'RelTol',1e-8,'Stats','on')
opbdf=odeset(opbdf,'BDF','on','Jacobian',@brussjac)
```

En la primera llamada al comando `odeset` hemos inicializado opciones ya conocidas, y en la segunda hemos indicado que `ode15s` debe utilizar las fórmulas BDF inicializando con `'on'` la opción `'BDF'`, y hemos indicado que la función que calcula la matriz jacobiana lleva por nombre `brussjac.m`.

Ejecutamos

[1] Puede ver una descripción de las fórmulas NDF en [4].

```
[Tb8,Yb8]=ode15s(@bruss,tiempos,[1.5 3]',opbdf)
```

y nos responde

```
901 successful steps
40 failed attempts
1508 function evaluations
1 partial derivatives
123 LU decompositions
1506 solutions of linear systems
```

Podemos observar lo siguiente.

- En los 901+40 pasos que ha dado y ha aceptado o rechazado, ha resuelto 1506 sistemas lineales. Tal y como señalamos anteriormente, para dar cada paso debe resolver un sistema de ecuaciones no lineales por el método de Newton, resolviendo los sistemas lineales en (19). Vemos que en la mitad de los pasos sólo ha efectuado una iteración.

- En los 901+40 pasos y rechazos, sólo ha efectuado 123 descomposiciones LU. Esto es debido a que la matriz en el sistema (19), una vez hecha su factorización LU (que recordemos tiene un costo que crece como el cubo de la dimensión de la matriz), aprovecha dicha descomposición no sólo en otras iteraciones del método de Newton, sino en otros pasos.

- La razón de los ahorros en iteraciones del método de Newton y de factorizaciones LU se debe a que los iterantes iniciales $y_{n+1}^{[0]}$ para el método de Newton obtenidos por extrapolación son tan correctos, que una sola iteración del método de Newton y con una matriz que no es la correcta es suficiente para encontrar el valor correcto de y_{n+1}.

- El número de pasos, 901, es muy superior al del par encajado DOPRI54 para las mismas tolerancias, que como vimos en la Sección , esto es, el método DOPRI54 puede dar pasos considerablemente mayors que las fórmulas BDF para una misma tolerancia.

El gráfico en la Fig. 8, donde se mide la eficiencia relativa de ambos métodos, muestra que, para alcanzar un nivel de precisión determinado, las fórmulas BDF son un promedio de tres veces más caras que el método DOPRI54. Aunque se trata solamente del ejemplo que venimos tratando, esto es, el problema (12–13) asociado al Brusselator, esta situación es típica y se repite en otros problemas, salvo que éstos, como veremos en la sección siguiente, sean stiff.

Otros métodos. Dado que los puede encontrar en ciertos contextos determinados así como en diversos libros de texto, comentamos aquí algunos métodos lineales multipaso que no pertenecen a las familias descritas en los apartados anteriores. En lo que sigue y por brevedad, denotaremos

$$f_n = f(t_n, y_n),\ n = 0, 1, \ldots N.$$

Para los métodos que siguen a continuación, sólo consideraremos el caso

$$h_n = h,\ n = 0, 1, \ldots, N-1.$$

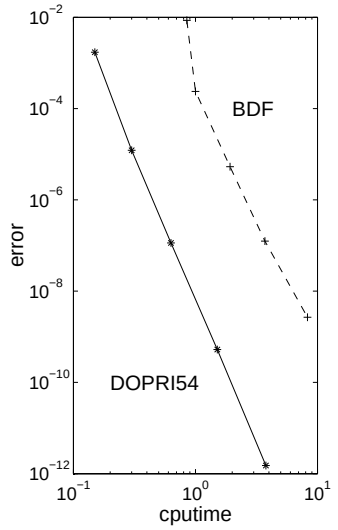


Figura 8: Eficiciencia de los métodos DOPRI54 y fórmulas BDF en problema (12–13)

1. *Regla de Simpson*, Es de la forma

$$y_{n+1} - y_{n-1} = \frac{h}{3}(f_{n+1} + 4f_n + f_{n-1}),\ n = 1, 2, \ldots, N-1,$$

y es convergente de orden 4. Note que debido a su alto orden el valor de arranque y_1 debe calcularse con un orden similar (por ejemplo con un método Runge-Kutta).

Proviene de aplicar la regla de Simpson en la igualdad

$$y(t_n + h) - y(t_n - h) = \int_{t_n-h}^{t_n+h} f(t, y(t))\, dt.$$

2. *Regla del punto medio explícita* o *"Leap-Frog"*. Es de la forma,

$$y_{n+1} - y_{n-1} = 2hf_n,\ n = 1, \ldots, N-1.$$

Es de orden 2. El valor de arranque y_1 se suele aproximar con el método de Euler.

3. *Método centrado para ecuaciones de segundo orden.* Aunque la ecuación (o sistema) de segundo orden

$$y'' = f(t, y),$$

se puede escribir como un sistema de primer orden de doble dimensión,

$$\begin{bmatrix} y' \\ z' \end{bmatrix} = \begin{bmatrix} z \\ f(t, y) \end{bmatrix},$$

y tratarlo como tal con cualquiera de los métodos vistos en las secciones anteriores, existen métodos específicos para sistemas de segundo orden. Aunque quedan fuera del alcance de

este curso, mencionamos aquí uno que aparece frecuentemente en la aproximación numérica de ecuaciones en derivadas parciales hiperbólicas.

$$y_{n+1} - 2y_n + y_{n-1} = h^2 f_n,\ n = 1, \ldots, N-1.$$

Es de orden 2. El valor de arranque y_1 se suele aproximar como

$$y_1 = y_0 + hy_0' + \frac{h^2}{2} f_0.$$

PROBLEMAS STIFF

Reciben este nombre determinados problemas de valor inicial, donde, para una precisión determinada, un método explícito (como los métodos RK vistos hasta ahora) necesitan para integrarlos tomar pasos mucho más pequeños que lo que la regularidad de la solución demanda. A continuación veremos un ejemplo ilustrativo, pero antes una cuestión de nomenclatura. La palabra "stiff" significa rígido en inglés, pero no se ha traducido al referirnos en español a este tipo de problemas. De hecho, se toma como cursi o ignorante hablar de "problemas rígidos".

Quizá, el ejemplo más sencillo para aclarar la diferencia entre problemas stiff y aquellos que no lo son sea el original de Prothero y Robinson, de 1974.

$$y'(t) = \lambda(y(t) - g(t)) + g'(t), \tag{20}$$
$$y(0) = g(0). \tag{21}$$

Las soluciones de la ecuación (20) son de la forma

$$y(t) = \alpha e^{\lambda t} + g(t),\ \alpha \in \mathbb{R},\ \ (\text{o } \alpha \in \mathbb{C}) \tag{22}$$

y la solución del problema (20–21) es

$$y(t) = g(t).$$

Se supone que g tiene infinitas derivadas y que el tamaño de estas es moderado. Cuando $\mathrm{Re}(\lambda) < 0$ y $|\mathrm{Re}(\lambda)|$ es muy grande en relación con $|\mathrm{Im}(\lambda)|$ y el tamaño de las derivadas de g, el problema (20–21) es stiff.

Ejemplo 4. Para $g = \cos(t)$, que es una función cuyas derivadas son del orden de la unidad, consideramos el problema (20–21), con $\lambda = -1$ y $\lambda = -100$. La Fig. 9 muestra el error frente al costo para diversos valores $TOL = RTOL = 10^{-4}, 10^{-6}, \ldots, 10^{-12}$ de los métodos DOPRI54 (en línea continua) y las fórmulas BDF (en línea de trazos) para los casos $\lambda = -1$ (marcado con $*$) y $\lambda = -100$ (marcado con $+$). Son oportunas ciertas observaciones que resumimos a continuación.

1. Para $\lambda = -1$ (gráficas con *), los resultados son similares a los del problema (12–13) asociado al Brusselator que vimos en la Fig. 8 de la sección anterior, esto es, que para alcanzar un niver de precisión determinado, los fórmulas BDf son tres veces más caras que el DOPRI54.

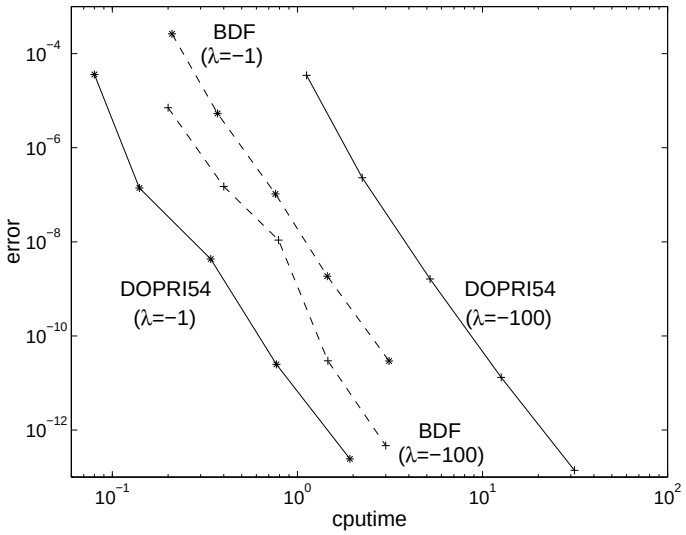


Figura 9: Eficiencia computacional de los métodos DOPRI54 (–) y fórmulas BDF (--) para el problema de Prothero-Robinson (20–21), con $g(t) = \cos(t)$, para $\lambda = -1$ (*), y para $\lambda = -100$ (+).

2. Para $\lambda = -100$, aunque se trata de la misma solución $y(t) = \cos(t)$ del caso $\lambda = -1$, la situación es diametralmente opuesta, resultando ser el método DOPRI54 unas 8 veces más costoso que las fórmulas BDF.

3. Los resultados del DOPRI54 muestran que dada una tolerancia TOL, en ambos problemas obtiene los mismos errores, pero para $\lambda = -100$ se obtienen con 15 veces más de costo.

4. Los resultados de las fórmulas BDF muestran que dada una tolerancia TOL, en ambos problemas tardan lo mismo en integrar la solución, pero para $\lambda = -100$, los errores obtenidos son unas 40 veces más pequeños.

A la vista de estas observaciones, el hecho más destacado es el deterioro tan significativo de la eficiencia del método DOPRI54 al pasar de $\lambda = -1$ a $\lambda = -100$. Este deterioro no puede ser debido a la solución del PVI, pues en ambos casos es la misma, $y(t) = \cos(t)$. Debe ser por tanto debido al PVI (ecuación diferencial junto con su condición inicial).

La razón de este deterioro es que DOPRI54 se ve forzado a tomar pasos muy pequeños para poder integrar el problema. Por ejemplo, para $TOL = 10^{-8}$, la información proporcionada por MATLAB es la siguiente. Para $\lambda = -1$,

```
203 successful steps
7 failed attempts
1261 function evaluations
0 partial derivatives
0 LU decompositions
0 solutions of linear systems
```

y para $\lambda = -100$,

```
3457 successful steps
11 failed attempts
20809 function evaluations
0 partial derivatives
0 LU decompositions
0 solutions of linear systems
```

Vemos que efectivamente el computo total de pasos aceptados y rechazados se multiplica por 16 al pasar de $\lambda = -1$ a $\lambda = -100$, y en igual proporción el número de evaluaciones de función.

No hay otra descripción de lo que es un problema stiff que la que ya hemos dado y repetimos a continuación. Un problema, para una *precisión* determinada y en un *intervalo de tiempo* determinado, es stiff cuando un *método explícito* debe tomar pasos mucho más pequeños que lo que la *regularidad de la solución* demanda. Comentamos las palabras en cursiva.

- Cuando decimos la regularidad de la solución, esto quiere decir que si esa misma solución lo fuese de otro problema, el método tomaría pasos considerablemente más largos manteniendo el nivel de precisión pedido. (Compare si no los pasos que ha necesitado el método DOPRI54 para integrar $y(t) = \cos(t)$ según sea $\lambda = -1$ o $\lambda = -100$).

- Cuando un problema es stiff, *todos* los métodos explícitos sufren la misma pérdida de eficiencia. Algunos métodos implícitos también (como por ejemplo los métodos de Adams implícitos) aunque no otros (como las fórmulas BDF). Por eso se utiliza como criterio para determinar si un problema es stiff el comportamiento en cualquier método explícito.

- No es infrecuente que un problema sea stiff en un intervalo $[t_0, t_f]$ pero deje de serlo a partir de t_0 (o caso más frecuente) no lo sea antes de t_0.

- El carácter stiff de un problema depende de la precisión deseada. Si demandamos mucha precisión (por ejemplo, en un ordenador cuya unidad de redondeo sea del orden de 10^{-36}) cualquier método se ve obligado a tomar pasos pequeños para conseguir errores muy bajos, y los problemas dejan de ser stiff.

Por último, para hacerse una idea de cuál es la dificultad de los problemas stiff, basta considerar en la ecuación (20) la desviación con respecto a la solución particular

$$z(t) = y(t) - g(t).$$

Note que al ser $\mathrm{Re}(\lambda) < 0$, y que tal y como vimos en (22), se tiene que

$$\lim_{t\to\infty} z(t) = 0. \tag{23}$$

Notemos que $z(t)$ satisface la ecuación

$$z' = \lambda z(t).$$

Para el método de Euler explícito (que es el único método Runge-Kutta de una etapa convergente), y sobre una partición uniforme ($h_n = h$, para $n = 1, 2, \dots$) se tiene que

$$z_{n+1} = z_n + \lambda h z_n = (1 + \lambda h) z_n = (1 + \lambda h)^2 z_{n-1} = \ldots = (1 + \lambda h)^{n+1} z_0.$$

Tenemos por tanto que con el metodo de Euler la solución numérica z_n reproduce el comportamiento de la solución de verdad en (23),

$$\lim_{n\to\infty} z_n = 0 \;\Leftrightarrow\; |1+\lambda h| < 1.$$

En el caso de $\lambda \in \mathbb{R}$ (y dado que entonces $\lambda < 0$) tenemos que

$$\lim_{n\to\infty} z_n = 0 \;\Leftrightarrow\; h < \frac{1}{|\lambda|}.$$

Dado que para otros métodos RK la situación es similar, esto explica las diferencias entre $\lambda = -1$ y $\lambda = -100$ en el Ejemplo 4.

Por otro lado para el método de Euler implícito (que es la fórmula BDF de un paso) tenemos que

$$z_{n+1} = z_n + \lambda h z_{n+1}, \text{ esto es, } (1-\lambda h) z_{n+1} = z_n,$$

o, equivalentemente,

$$z_{n+1} = (1-\lambda h)^{-1} z_n = (1-\lambda h)^{-2} z_{n-1} = \ldots = (1-\lambda h)^{-n-1} z_0.$$

Tenemos por tanto que con el método de Euler implícito,

$$\lim_{n\to\infty} z_n = 0 \;\Leftrightarrow\; \frac{1}{|1+\lambda h|} < 1.$$

En el caso de $\lambda \in \mathbb{R}$ y negativo tenemos que

$$\lim_{n\to\infty} z_n = 0 \;\Leftrightarrow\; h > 0,$$

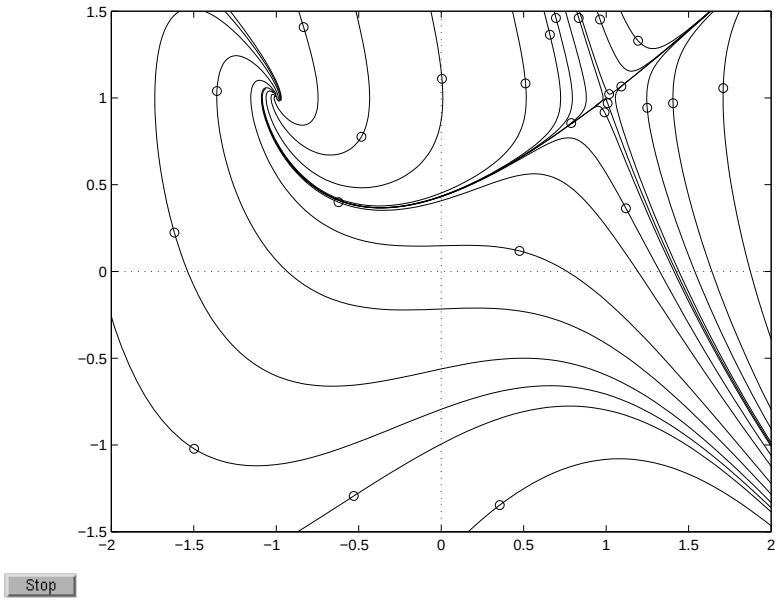
esto es, que para reproducir el comportamiento de la solución $z(t)$ no hay ninguna restricción de longitud de paso, salvo la que venga impuesta por la regularidad de la misma. El caso de otras fórmulas BDF de más pasos es similar al del método de Euler implícito, explicando esto que las fórmulas BDF sean insensibles al tamaño de λ en la ecuación (20).

Puede encontrar más información sobre la naturaleza de los problemas stiff en [3], § 6.1–6.2, así como amplia documentación sobre este tipo de problemas y su tratamiento numérico en [2].

Otras cuestiones de orden práctico. Los comandos de MATLAB para integrar EDOs o sistemas de EDOs, a través de las opciones que les demos mediante el comando `odeset`, pueden llevar a cabo diversas tareas como dibujar mapas de fase, ayudarnos a determinar el periodo de una solución periódica, etc.

Puede que le resulte de utilidad la posibilidad de construir mapas de fase. Mediante la opción `OuputFcn` podemos indicarle a MATLAB que realice determinada acción cada vez que calcula un punto. De hecho, MATLAB tiene prevista la función `odephas2` que dibuja las trayectorias según

las va calculado. El siguiente mapa de fase



del sistema

$$\left.\begin{array}{l} x' = 1 - y \\ y' = x^2 - y^2 \end{array}\right\} \tag{24}$$

se obtuvo con la secuencia de comandos

```
% en la opción para la función de output, damos odephas2 para
% que dibuje la curva paramétrica t -> [x(t),y(t)]';
opciones=odeset('AbsTol',1e-6,'RelTol',1e-6,'OutputFcn',@odephas2);
figure(1);clf
% dibujamos los ejes en rojo
plot([-2,2],[0,0],'r:')
hold
plot([0,0],[-1.5 1.5],'r:');

% fijamos los extremos del dibujo
axis([-2 2 -1.5 1.5])

% por cada condición inicial, que avance 5 unidades de tiempo
% y que retroceda cinco unidades de tiempo.
tiempos=[0 5];
tiempos2=[0 -5];
for j=1:25
  % seleccionamos las condiciones inciales con el ratón
  g=ginput(1)
  % marcamos la condición incial en rojo y con un circulito.
  plot(g(1),g(2),'ro')
  % integramos el sistema (dado que la opción OutputFcn tiene
```

```
  % el valor @odephas2, se dibujarán las trayectorias según se
  % calculan
  [T,Y]=ode45(@sistema,tiempos,g,opciones);
  [T,Y]=ode45(@sistema,tiempos2,g,opciones);
end
```

donde la función `sistema` proporciona el segundo miembro del sistema (24):

```
function f=sistema(t,y)
f=[1-y(2);sum(y)*diff(y)];
```

Claramente se aprecian dos equilibrios, uno de ellos un punto de silla, y el otro un punto espiral. Puede continuar el estudio de este ejemplo en el Problema 4

APLICACIÓN A PROBLEMAS DE CONTORNO

Una de las diversas técnicas para resolver problemas de contorno, conocida como método de disparo, se basa en la resolución de PVIs. Recuerde que un problema de contorno consiste en una ecuación diferencial de segundo orden (o un sistema de primer orden con más de una ecuación) junto con condiciones en los extremos de un intervalo $[a, b]$. Ejemplos de tales problemas son los siguientes:

$$\left.\begin{array}{c} y'' + y = 0, \\ y(0) = 0,\ y(1) = 1, \end{array}\right\}, \tag{25}$$

$$\left.\begin{array}{c} y'' + y = \cos(2t), \\ y(0) = 0,\ y(\pi) = 0, \end{array}\right\}, \tag{26}$$

$$\left.\begin{array}{c} y'' + y = \cos(\frac{t}{2}), \\ y(0) = 0,\ y(\pi) = 0, \end{array}\right\}, \tag{27}$$

$$\left.\begin{array}{c} y'' + |y| = 0, \\ y(0) = 0,\ y(1) = \beta, \end{array}\right\}, \tag{28}$$

$$\left.\begin{array}{c} y'' + \lambda e^u = 0, \\ y(0) = 0,\ y(1) = 0, \end{array}\right\}, \tag{29}$$

donde λ es un parámetro positivo, y en general,

$$\left.\begin{array}{c} y'' + f(t, y, y') = r(t), \\ y(a) = \alpha,\ y(b) = \beta, \end{array}\right\}. \tag{30}$$

Los tres primeros son problemas lineales de coeficientes constantes, que son los que se pueden resolver analíticamente. Los dos siguientes son problemas no lineales. Aunque aquí nos centraremos en las condiciones frontera $y(a) = \alpha$, $y(b) = \beta$, se tratan de modo similar problemas con otro tipo de condiciones frontera como $y'(a) = \alpha$, o $\gamma y'(a) + y(a) = \alpha$.

La casuística de los problemas de contorno es bastante más compleja que las de los PVI (en éstos, si el segundo miembro es regular, siempre tienen solución). Así por ejemplo, el problema (25) tiene una única solución que es

$$y(t) = \frac{\text{sen}(t)}{\text{sen}(1)},$$

sin embargo el problema (26) tiene infinitas soluciones de la forma

$$y(t) = \gamma \sin(t) - \frac{1}{3}\cos(2t),\ \gamma \in \mathbb{R},$$

mientras que el problema (27) no tiene solución. A su vez, el problema (28) tiene dos soluciones distintas si $\beta < 0$, una única solución si $\beta = 0$, y ninguna si $\beta > 0$; Por último el problema (27) tiene dos soluciones distintas para $\lambda \in (\lambda, \lambda_0)$ donde $\lambda_0 \approx \sqrt{2}$, una única solución si $\lambda = \lambda_0$ y ninguna si $\lambda > \lambda_0$.

Una casuística más definida presentan los *problemas regulares de contorno*, que son lineales y de la forma,

$$\left.\begin{array}{c} y'' + py' + qy = r, \\ y(a) = \alpha,\ y(b) = \beta, \end{array}\right\} \text{(PRC)},$$

donde $p = p(t)$, $q = q(t)$ y $r = r(t)$ son funciones continuas en todo $[a, b]$. Tratándose de problemas lineales, las soluciones de (PRC), en caso de haberlas, son la suma de una particular y las del problema homogéneo asociado,

$$\left.\begin{array}{c} y'' + py' + qy = 0, \\ y(a) = 0,\ y(b) = 0, \end{array}\right\} \text{(PCH)}.$$

El *Teorema de la alternativa de Fredholm*, asegura (PRC) tiene solución y además es única para todos $\alpha, \beta \in \mathbb{R}$ y para cualquier r continua en $[a, b]$ si y sólo si la única solución de (PCH) es la trivial, $y(t) = 0$ para todo $t \in [a, b]$.

De hecho, se puede disponer todavía de más información a este respecto (véase por ejemplo [5], § 43). Así por ejemplo, si $p = 0$ y (PCH) tiene soluciones no nulas, entonces éstas son todas proporcionales a una dada ϕ, y (PRC), cuando $\alpha = \beta = 0$ tiene solución si y sólo si

$$\int_a^b q(t)\phi(t)r(t)\,dt = 0.$$

(Si $\alpha \neq 0$ ó $\beta \neq 0$ estudiar el problema que queda cuando a y se le resta la recta que en a vale α y β en b).

Para problemas regulares de contorno, el *método de disparo* se entiende fácilmente. Basta observar que las soluciones de la ecuación diferencial

$$y'' + py + qy = r,$$

son una particular y_p

$$y_p'' + py_p' + qy_p = r, \tag{31}$$

más la general de la ecuación homogénea $y_0'' + py_0' + qy_0 = 0$. Si y_p se elige satisfaciendo las condiciones iniciales

$$y_p(a) = \alpha,\ y_p'(a) = 0, \tag{32}$$

bastará sumar a y_p una solución y_0 de la ecuación homogénea satisfaciendo $y_0(0) = 0$ para que no alteremos lo que vale la suma $y_p + y_0$ en $t = a$. Luego consideramos la solución del PVI,

$$\left.\begin{array}{c} y_0'' + py_0' + qy_0 = 0, \\ y_0(a) = 0,\ y_0'(a) = 1 \end{array}\right\}, \tag{33}$$

La combinación lineal

$$y = y_p + \gamma y_0$$

será solución de (PRC) si y sólo si $\beta = y(b) = y_p(b) + \gamma y_0(b)$, y por tanto la solución de (PRC) es

$$y(t) = y_p(t) + \frac{\beta - y_p(b)}{y_0(b)} y_0(t), \tag{34}$$

con y_p solución del PVI (31–32) e y_0 solución del PVI (33).

Nota 3. Observe que para que la solución y en (34) esté bien definida debe ser $y_0(b) \neq 0$. En el caso en que $y_0(b) = 0$, note entonces que, dado que hemos elegido y_0 satisfaciendo la condición inicial $y_0(a) = 0$ en (33), la función y_0 sería solución de (PCH), y además no nula pues también hemos impuesto en (33) la condición inicial $y_0'(a) = 1 \neq 0$. Vemos pues que en la práctica, podemos detectar la presencia de infinitas soluciones comprobando si $y_0(0) = 0$.

Si de usa un comando de MATLAB como `ode45` u `ode113` para encontrar y_p e y_0, recuerde que debe reescribir las ecuaciones diferenciales de segundo orden como sistemas de primer orden,

$$\begin{bmatrix} y_p' \\ v_p' \end{bmatrix} = \begin{bmatrix} v_p \\ r - qy_p - pv_p \end{bmatrix}, \quad \begin{bmatrix} y_0' \\ v_0' \end{bmatrix} = \begin{bmatrix} v_0 \\ -qy_0 - pv_0 \end{bmatrix}.$$

En el caso de *problemas no lineales*, que de forma general se escriben como (30), el método de disparo consiste en resolver la ecuación

$$g(s) = 0,$$

por algún método para resolver ecuaciones no lineales como puede ser el método de bisección o el método de la secante (véase por ejemplo [6], § 9, [7], § 1-4) donde

$$g(s) = y(b) - \beta,$$

e y la solución del PVI

$$\left.\begin{array}{l} y'' + f(t, y, y') = r, \\ y(a) = \alpha,\ y(b) = s. \end{array}\right\}. \tag{35}$$

Para aquél valor de s para el que $g(s) = 0$, la correspondiente solución de (35) es la solución del problema de contorno (30).

Método de diferencias para problemas de contorno en EDO

Dado el problema de contorno lineal

$$(P) \equiv \left\{ \begin{array}{l} y'' + py' + qy = r \\ y(a) = \alpha,\ y(b) = \beta \end{array} \right. ,$$

fijamos un conjunto de $n + 2$ ($n \in \mathbb{N}$) nodos igualmente espaciados

$$t_i = a + ih,\ i = 0, 1, ..., n + 1;\ h = \frac{b - a}{n + 1}.$$

En cada uno de los nodos interiores t_i $(1 \leq i \leq n)$ aproximamos las derivadas primera y segunda mediante las correspondientes fórmulas centradas de tres puntos

$$\begin{aligned} y'(t_i) &= \frac{y(t_{i+1}) - y(t_{i-1})}{2h} + O(h^2) \\ y''(t_i) &= \frac{y(t_{i+1}) - 2y(t_i) + y(t_{i-1})}{h^2} + O(h^2). \end{aligned}$$

La ecuación diferencial en esos puntos se transforma en

$$\frac{y(t_{i+1}) - 2y(t_i) + y(t_{i-1})}{h^2} + p(t_i)\frac{y(t_{i+1}) - y(t_{i-1})}{2h} + q(t_i)y(t_i) = r(t_i) + O(h^2).$$

Imponiendo que las aproximaciones $y_i \approx y(t_i)$ verifiquen la ecuación en diferencias que resulta de despreciar el término de orden $O(h^2)$, llegamos a la fórmula que define el método de diferencias finitas, es decir,

$$\begin{gathered} \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + p_i\frac{y_{i+1} - y_{i-1}}{2h} + q_i y_i = r_i,\ i = 1, ..., n \\ \text{donde } p_i = p(t_i),\ q_i = q(t_i),\ r_i = r(t_i) \end{gathered}$$

Evidentemente, estamos considerando $y_0 = \alpha$ e $y_{n+1} = \beta$.

Reorganizando las ecuaciones anteriores queda

$$(1 - \frac{1}{2}hp_i)y_{i-1} + (-2 + h^2 q_i)y_i + (1 + \frac{1}{2}hp_i)y_{i+1} = h^2 r_i,\ i = 1, ..., n.$$

Si denotamos

$$a_i := -2 + h^2 q_i,\ b_i := 1 + \frac{1}{2}hp_i,\ c_i := 1 - \frac{1}{2}hp_i$$

podemos escribir las ecuaciones como

$$c_i y_{i-1} + a_i y_i + b_i y_{i+1} = h^2 r_i,\ i = 1, ..., n.$$

Esto nos da un sistema tridiagonal de n ecuaciones lineales con n incógnitas $y_1, ..., y_n$. En forma matricial, el sistema es $Ay = d$, donde

$$A = \begin{bmatrix} a_1 & b_1 & & & \\ c_2 & a_2 & b_2 & & \\ & c_3 & a_3 & \ddots & \\ & & \ddots & \ddots & b_{n-1} \\ & & & c_n & a_n \end{bmatrix},\ d = \begin{bmatrix} h^2 r_1 - c_1\alpha \\ h^2 r_2 \\ \vdots \\ h^2 r_{n-1} \\ h^2 r_n - b_n\beta \end{bmatrix}$$

En general, el sistema $Ay = d$ no tiene por qué tener solución. El siguiente teorema nos dice que en condiciones bastante generales y para tamaños de paso h suficientemente pequeños se consigue la compatibilidad.

Teorema. *Sean p, q, y r funciones de clase $C[a, b]$ con $q > 0$. Entonces,*

1. *si* $p \neq 0$ *y el tamaño de paso* h *verifica que* $h < 2/M$, *donde* $M = \max\{|p(t)| : t \in [a,b]\}$, *la matriz del sistema tridiagonal asociado* $Ay = d$ *es de diagonal dominante y, por tanto, dicho sistema tiene solución única.*

2. *si* $p = 0$, *para cualquier tamaño de paso* h, *la matriz del sistema tridiagonal asociado* $Ay = d$ *es simétrica definida positiva y, por tanto, dicho sistema tiene solución única.*

El sistema $Ay = d$ puede calcularse mediante eliminación gaussiana tridiagonal, luego el coste computacional para el calculo del vector y es proporcional al orden de la matriz. Para el caso donde la función p es identicamente nula puede establecerse una acotación relativamente sencilla del error que genera este método. El teorema básicamente indica que el error tiende cuadráticamente a cero respecto al paso.

Teorema. Consideremos que el problema (P) tiene una solución $y \in C^4[a,b]$,, donde $p = 0$ y q, r son funciones de clase $C[a,b]$ con $q > 0$. Si denotamos por $\{y_0(h), ..., y_{n+1}(h)\}$ la correspondiente solución del método de diferencias para el paso $h > 0$, se verifica que

$$|y(t_i) - y_i(h)| \leq \frac{h^2}{24}(b-a)^2 \max\{|y^{4)}(t)| : t \in [a,b]\}.$$

CUESTIONES

Ejercicio 1. Obtener la solución general de la ecuación $y'' + 2y' + y = 1$.

Ejercicio 2. Obtener el paso n-ésimo del método de Euler aplicado a la ecuación

$$\left.\begin{array}{l} y' = t \\ y(0) = A \end{array}\right\}$$

cuando $A > 0$.

Ejercicio 3. Compruebe la convergencia del método $y_{k+1} = y_k + hf_k$, para el problema

$$\left.\begin{array}{l} y' = y, \\ y(0) = 1, \end{array}\right\} \ t \in [0,T].$$

Ejercicio 4. Dado un paso $h > 0$, obtener la forma general de los valores asignados por el método de Euler al problema

$$\left.\begin{array}{l} y' = ty, \\ y(0) = a, \end{array}\right\} \ t \in [0,1].$$

Ejercicio 5. Escriba el tablero del método de Heun.

Ejercicio 6. El método de Runge de orden 3 tiene por tablero

$$\begin{array}{c|cccc} 0 & 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ 1 & 0 & 1 & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{2}{3} & 0 & \frac{1}{6} \end{array}$$

Calcular la expresión de y_{n+1} en función de y_n y de sus cuatro etapas k_1, k_2, k_3, y k_4, y las de éstas en función de y_n y de sí mismas.

Ejercicio 7. Para una ecuación autónoma $y' = f(y)$ (donde f no depende de t), calcular el error local del método de Euler mejorado y comprobar que efectivamente es de orden 2.

Ejercicio 8. Resuelva la ecuación $y' = -y^2$, para todos los valores iniciales posibles.

Ejercicio 9. Comprobar que un método Runge-Kutta es consistente si y sólo si

$$\sum_{i=1}^{s} b_i = 1.$$

Ejercicio 10. Comprobar que un método Runge-Kutta es consistente de orden 2 si y sólo si

$$\sum_{i=1}^{s} b_i = 1, \text{ y } \sum_{i=1}^{s} b_i c_i = \frac{1}{2}$$

Ejercicio 11. Obtener el método de Adams explícito de dos pasos, y particularizar el caso $h_n = h$, para todo n.

Ejercicio 12. Idem con método de Adams implícito de dos pasos.

Ejercicio 13. Compruebe que efectivamente el método de Euler implícito es la fórmula BDF de un paso.

Ejercicio 14. Obtener la fórmula BDF de dos pasos, y particularizar el caso $h_n = h$, para todo n.

Ejercicio 15. Para el problema de valor inicial

$$\left.\begin{array}{l} y' = y, \\ y(0) = 1, \end{array}\right\} \ t \in [0,1]$$

determine explícitamente la solución numérica $y_n(h)$ $(n = 0, 1, \dots)$, proporcionada por el método de Euler para un paso $h > 0$. Compruebe que

$$\lim_{N\to\infty} y_N(\frac{1}{N}) = e$$

e interprete este límite en términos de la convergencia del método de Euler.

Ejercicio 16. Para los problemas de contorno (26) y (27) calcule las soluciones ϕ del problema homogéno asociado, y calcule el valor de la integral

$$\int_0^\pi \phi(t) r(t)\, dt,$$

para $r(t) = \cos(2t)$, y $r(t) = \cos(\frac{1}{2}t)$. ¿Concuerda esto con lo comentado sobre el caso $p = 0$?

Ejercicio 17. Resolver el problema de contorno

$$\begin{cases} y'' - y = 0, \\ y(0) = 0, y(\pi) = 2. \end{cases}$$

Ejercicio 18. Resolver el problema de contorno

$$\begin{cases} y'' + y = t, \\ y(0) = 1, y(\pi) = 0. \end{cases}$$

Comprobar que no se pueden aplicar los teoremas básicos sobre existencia y unicidad de soluciones.

Ejercicio 19. Analizar la existencia y unicidad de soluciones del problema de contorno

$$y'' + \lambda y = \alpha t + \beta b,\ y(0) = y(\pi) = 0.$$

en función de los parámetros λ, α y β.

Ejercicio 20. Determine el sistema tridiagonal asociado, con paso $h > 0$, que resulta de aplicar el método de diferencias al problema de contorno

$$y'' = -ty' + y - 1,\ y(0) = y(1) = 1.$$

Ejercicio 21. Usando el método de diferencias, obtenga la aproximación con tres nodos del problema de contorno

$$y'' = y + 1,\ y(0) = 0, y(1) = -1.$$

Compare el resultado con la solución exacta.

+

PROBLEMAS

Problema 1. Se trata en este problema de familiarizarse con los comandos de MATLAB. Lea atentamente la Sección . Repetiremos algunos de los cálculos allí hechos para las ecuaciones del movimiento de un sólido rígido propuestas por el mismo Euler, $y = f(y)$, que componente a componente son

$$\left.\begin{aligned} y_1' &= \frac{I_2 - I_3}{I_1} y_2 y_3 + b_1/I_1, \\ y_2' &= \frac{I_3 - I_1}{I_2} y_3 y_1 + b_2/I_2, \\ y_3' &= \frac{I_1 - I_2}{I_3} y_1 y_2 + b_3/I_3, \end{aligned}\right\} \tag{36}$$

donde y_1, y_2 e y_3 son las tres componentes de la velocidad angular, I_1, I_2 e I_3 son los momentos principales de inercia del sólido, y b_1, b_2 y b_3 son las tres componentes de un campo externo de momentos.

1. Escriba una función que dados t e y devuelva el segundo miembro de (36), para

$$I_1 = 0.5,\ I_2 = 2,\ I_3 = 3,$$

y con fuerza externa $b = 0$.

2. Utilizando $TOL = RTOL = 10^{-6}$ con el comando `ode45`, integre en $[0, 20]$ el problema $y' = f(y)$, con $y(0) = [1, 0, 0.9]^T$, obteniendo aproximaciones numéricas cada 0.1 unidades de tiempo. Dibuje las tres componentes de la solución y escriba el número de pasos, de rechazos y de evaluaciones de función.

3. Repita los cálculos del apartado anterior anterior utilizando $TOL = RTOL = 10^{-12}$.

4. Utilizando la solución anterior como solución exacta, mida los errores y los tiempos de ejecución del método DOPRI54 (comando `ode45`) y de los métodos de Adams (comando `ode113`), para tolerancias $TOL = RTOL = 10^{-2k}$, para $k = 2, 3, 4, 5, 6$, y elabore un gráfico de eficiencia como el de la Fig. 8. Explique qué método es más eficiente.

Problema 2. Hay diversas cosas que todo estudiante de una carrera técnica o científica debe hacer. Una es programar la eliminación gaussiana, que ya hicimos en su día. Otra programar un método Runge-Kutta. Empezaremos por el método de Euler.

1. Elabore una función de MATLAB que aplique el método de Euler con paso fijo ($h_n = h$ para todo n) a un problema de valor inicial en un intervalo $[t_0, t_f]$. Los argumentos de entrada deben ser el nombre de la función que evalua el segundo miembro de la ecuación $y' = f(t, y)$, los instantes inicial t_0 y final t_f, la condición inicial y el paso h. Los argumentos de salida deben ser un vector T con los tiempos t_n y una matriz Y donde, para cada n, la fila n-ésima contenga y_n^T. Note que los argumentos de salida son como los de los comandos de MATLAB.

2. Aprenda esto: Siempre que se hace un programa para integrar PVI, se debe probar con el problema,

$$\begin{bmatrix} y_1' \\ y_2' \end{bmatrix} = \begin{bmatrix} y_1 \\ -y_2 \end{bmatrix},\ \begin{bmatrix} y_1(0) \\ y_2(0) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

cuya solución es $y(t) = [e^t, e^{-t}]^T$. Hágalo con la función del apartado anterior y obtenga una tabla de errores $\|y(1) - y_N\|$, para $N = 1/h$, y $h = 0.1, 0.01, 0.001, 0.0001$. ¿Es convergente el método programado? ¿De qué orden?

3. Repita los dos apartados anteriores con el método Runge-Kutta clásico. ¿Es convergente el método programado? ¿De qué orden? Explique el comportamiento de los errores.

4. Repita los cálculos del apartado anterior pero con $t_f = 5$ y midiendo el tiempo de ejecución, obteniendo una tabla de costo-error. Obtenga una tabla similar pero integrado con el comando `ode45`, con $TOL = RTOL = 10^{-4}, 10^{-8}, 10^{-12}$ (haga que el ode45 saque valores en $t_0, 2.5, tf$). ¿Qué procedimiento de integración es más eficiente?

Problema 3. En este problema encajaremos un par de orden 3 al método Runge-Kutta clásico. Cosidere el método Runge-kutta de 5 etapas de tablero

$$\begin{array}{c|ccccc} 0 & 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & & \\ 1 & 0 & 0 & 1 & & \\ 1 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{6} \end{array}$$

Observe que las cuatro primeras etapas son las del método Runge-Kutta clásico.

1. Modifique el programa del método Runge-Kutta clásico del problema anterior para que integre con el nuevo método de 5 etapas.

2. Pruebe el método del apartado anterior en con el problema

$$\begin{bmatrix} y_1' \\ y_2' \end{bmatrix} = \begin{bmatrix} y_1 \\ -y_2 \end{bmatrix}, \quad \begin{bmatrix} y_1(0) \\ y_2(0) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

   cuya solución es $y(t) = [e^t, e^{-t}]^T$. Midiendo los errores correspondientes a $h = 0.1$, $h = 0.01$ y $h = 0.001$, determine experimentalmente el orden de método.

3. El método del apartado 1 puede utilizarse para controlar la logitud de paso del método RK clásico en el par encajado de tablero

$$\begin{array}{c|ccccc} 0 & 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & & \\ 1 & 0 & 0 & 1 & & \\ 1 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} & \\ \hline b_i & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} & \\ \hline \hat{b}_i & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{6} \end{array}$$

   Modifique el programa del apartado 1 para que integre con este par encajado. En los argumentos de entrada, cambie la longitud de paso h por las tolerancias TOL y $RTOL$. Tome como longitud de paso inicial $h = TOL^{(1/4)}$. Tome como `facmax` el valor 5, y como `hmax=0.5`. Pruébelo con el problema del apartado 2.

4. Compare la eficiencia de este método con el método DOPRI54 (comando `ode45`) en el problema (12–13) asociado al Brusselator. ¿Qué método es más eficiente?

Problema 4. Lea lo relativo a mapas de fase en la Sección . El sistema autónomo (un sistema $y' = f(t, y)$ es autónomo si f no depende de t, esto es, es de la forma $y' = f(y)$) que vimos en dicha sección, recuerde que es

$$\left.\begin{array}{l} y_1' = 1 - y_2 \\ y_2' = y_1^2 - y_2^2 \end{array}\right\}.$$

1. Ejecute un ciclo similar al detallado en la Sección con 30 condiciones iniciales.

2. Determine (analíticamente) los puntos críticos del sistema, esto es las soluciones de $f(y) = 0$, que corresponden a soluciones constantes en el tiempo. A la vista del mapa de fase obtenido en el apartado anterior, ¿Qué tipo de punto crítico es cada uno?

3. Para cada uno de los equilibrios y_0 del sistema, repita el Apartado 1, con el sistema linealizado en torno a y_0, esto es
$$z' = f_y(y_0)z,$$
donde $f_y(y_0)$ es la matriz jacobiana de f en el punto y_0. ¿Se parecen los mapas de fases de los sistemas linealizados al mapa de fase del Apartado 1?

4. Obtenga la solución analítica de cada uno de los sistemas del apartado anterior, y compárela con la representación gráfica obtenida en dicho apartado.

Problema 5. En este problema trataremos un ejemplo de EDO no lineal de primer orden con propiedades interesantes.

Considere el problema de valores iniciales siguiente

$$y' = y^2(1 - y) \quad y(0) = y_0\,.$$

1. Compruebe que la solución exacta es
$$t = -\frac{1}{y} + \log\left(\frac{y(y_0 - 1)}{(y - 1)y_0}\right)$$
Dibújela con en el intervalo $[0, 2 \cdot 10^4]$ con $y_0 = 10^{-4}$. ¿Qué dificultades encuentra?

2. Discuta el comportamiento de la solución de la EDO anterior de manera cualitativa, calculando los puntos de equilibrio y su estabilidad.

3. Resuelva el PVI con las comandos de MATLAB `ode23`, `ode45` y `ode113` para distintos pasos y tolerancias. Si no conoce el método de integración de estas funciones consulte la ayuda. Preste especial atención al rango de ordenandas en torno a 1 y explique el resultado.

4. ¿Qué orden(es) de MATLAB permite(n) resolver este problema de manera satisfactoria?

Problema 6. Considere la ecuación diferencial

$$y'' = -\lambda y,\ \lambda \in \mathbb{R}.$$

1. Resuelva analíticamente la ecuación anterior. ¿Para qué valores de λ tiene solución única el problema de contorno $y(0) = 3$,e $y(\pi/2) = 7$?

2. Diseñe una función en MATLAB que implemente el método de disparo utilizando la orden `ode45`. Los argumentos de entrada deben ser la correspondiente ecuación, los puntos, las condiciones de contorno y la tolerancia de la orden `ode45`.

3. Resuelva gráficamente el problema de contorno

$$\left.\begin{array}{c} y'' = -\lambda y, \\ y(0) = \alpha,\ y(\pi/2) = \beta, \end{array}\right\} \text{(P)},$$

usando la función del apartado dos con $\alpha = 3$, $\beta = 7$ y $\lambda = 1$, con tolerancias 10^{-4}, 10^{-6}, 10^{-8} y 10^{-10} en el comando `ode45`.

4. Estime el error cometido (en norma euclídea) en el apartado anterior para las distintas tolerancias. Interprete la tabla de resultados obtenida.

5. Utilizando la función del apartados dos, resuelva y represente gráficamente la solución del problema (P) para $\lambda = 4$ con $\alpha = 0, \beta = 1$. ¿Cree que la solución encontrada es realmente la solución del problema?

Problema 7. Considere el problema de contorno

$$\left.\begin{array}{c} y'' = y + \lambda t \\ y(1) = \alpha,\ y(3) = \beta \end{array}\right\} \text{(P)},$$

donde $\lambda \geq 0$

1. Resuelva analíticamente el problema de contorno anterior para los valores de los parámetros $\alpha = 1$ y $\beta = 0$.

2. Diseñe una función en MATLAB que implemente el método de disparo utilizando la orden `ode45` y que proporcione una aproximación de la solución en un mallado del intervalo $[1, 3]$ con un cierto paso. Los argumentos de entrada deben ser la correspondiente ecuación, las condiciones de contorno, el paso y la tolerancia para el comando `ode45`.

3. Utilice la función del apartado dos para estimar la solución del problema (P) con $\lambda = 0$, tolerancia 10^{-4}, 10^{-6}, 10^{-8} y 10^{-10}, $\alpha = sh(1) \simeq 1.17520$, $\beta = sh(3) \simeq 10.01787$, y pasos $h = 0.05, 0.025$. En todos los casos, estime el error cometido en el punto $t = 2$. ¿Qué argumento o argumentos de entrada del método de disparo anterior son los que realmente controlan la precisión de los resultados?

Problema 8. Programe el método de disparo **no lineal** y resuelva las siguientes ecuaciones:

1. $y'' = y^3/2;\ y(1) = -2/3,\ y(2) = -1$. Tome $h = 0.05$ y compare con la solución exacta $y(x) = 2/(x - 4)$.

2. $y'' = y^3 - yy';\ y(1) = 1/2,\ y(2) = 1/3$. Tome $h = 0.1$ y compare con la solución exacta $y(x) = 1/(x + 1)$.

Problema 9. Considere el problema 7.

$$(P) \equiv \begin{cases} y'' = y + \lambda t \\ y(1) = \alpha,\ y(3) = \beta \end{cases} \quad (\lambda \geq 0)$$

1. Diseñe una función de MATLAB que implemente el método de diferencias finitas para resolver un problema de contorno del tipo

$$\begin{cases} y'' = y \\ y(1) = \alpha,\ y(3) = \beta. \end{cases}$$

   Los argumentos de entrada deben ser el tamaño del paso y las condiciones de contorno. Los argumentos de salida deben ser el mallado del intervalo y las estimaciones en los puntos de dicho mallado.

2. Utilice la función del apartado uno para estimar la solución del problema (P) con $\lambda = 0, \alpha = sh(1), \beta = sh(3)$ y con pasos $h = 0.5, 0.25, 0.125$. Para $h = 1$, resuelva el sistema "a mano". En los cuatro casos, estime el error cometido en el punto $t = 2$ y compruebe que dicho error sigue aproximadamente la pauta predicha por la teoría. Modifique la función que implementa el método del disparo de la lección anterior integrando con un el método de Runge Kutta clásico con paso fijo. Para un mismo tamaño de paso ¿qué método proporcionaría resultados más precisos el de disparo o el de diferencias finitas?

3. Estudie el comportamiento del error cuando h tiende a cero. Si es necesario emplee el comando **spdiags** de MATLAB para construir las matrices. Dibuje en escala logarítmica el error frente a h.