Universidad de Valladolid

Facultad de Ciencias

Trabajo Fin de Grado

Grado en Matemáticas

Métodos de colocación para la integración numérica de ecuaciones diferenciales

Autora: Carmen Mezquita Nieto

Tutora: María Paz Calvo Cabrero

Índice general

Introducción

El objetivo de este Trabajo de Fin de Grado es el estudio de métodos de colocación para la integración numérica de sistemas de ecuaciones diferenciales ordinarias, incluyendo resultados teóricos sobre su orden y estabilidad, y prestando especial atención a los denominados métodos de Radau.

Los métodos de colocación se mencionan brevemente en la asignatura optativa Solución Numérica de Ecuaciones Diferenciales de cuarto curso del Grado en Matemáticas, aunque no se profundiza más allá de la definición y alguna propiedad básica.

En el Capítulo 1 se revisan las definiciones básicas sobre métodos Runge-Kutta, incluyendo la construcción de las condiciones de orden utilizando árboles con raíz.

En el Capítulo 2 se analizan con detalle las simplificaciones que en la teoría del orden de los métodos Runge-Kutta implícitos introducen las llamadas condiciones simplificadoras, se relacionan dichas condiciones simplificadoras con los métodos de colocación y se presentan las distintas familias de métodos de Radau.

En el Capítulo 3 se estudian los distintos conceptos de estabilidad para los métodos Runge-Kutta, y en particular, para los métodos de Radau.

El Capítulo 4 está dedicado a la implementación eficiente con paso variable del método de Radau IIA de orden 5 y a mostrar los resultados obtenidos cuando dicha implementación se utiliza para integrar el oscilador de Van der Pol, un ejemplo bien conocido de problema rígido.

En el Apéndice A se han incluido tablas con las condiciones de orden para los métodos Runge-Kutta, y los árboles con raíz asociados. En el Apéndice B se han demostrado algunas propiedades de los polinomios de Legendre y en el Apéndice C se incluyen las funciones en Matlab con las implementaciones de los métodos de Radau y otras funciones auxiliares que se han utilizado para generar las gráficas que aparecen en la memoria.

Me gustaría agradecerle a mi tutora, la Dra. María Paz Calvo, su apoyo e infinita paciencia durante este curso 2019/2020, tanto en la asignatura que me ha impartido como durante la cuarentena y la escritura de esta memoria, sin ella no lo habría conseguido.

Capítulo 1

Métodos Runge-Kutta

1.1. Definición de un método Runge-Kutta

Consideraremos el siguiente problema de valores iniciales

$$\mathbf{y}'(x) = \mathbf{f}(x, \mathbf{y}(x)), \quad x_0 \leq x \leq x_f, \tag{1.1a}$$

$$\mathbf{y}(x_0) = \mathbf{y}_0, \tag{1.1b}$$

donde $x_0 \in \mathbb{R}$ y $\mathbf{f} : \mathbb{R} \times \mathbb{R}^D \to \mathbb{R}^D$ es suficientemente regular.

Definición 1.1. Sean $s \in \mathbb{N}, a_{ij}, b_i, c_i \in \mathbb{R}, i, j = 1, \ldots, s$. Las ecuaciones

$$\mathbf{k}_i = \mathbf{f}(x_n + c_i h, \mathbf{y}_n + h \sum_{j=1}^{s} a_{ij} \mathbf{k}_j), \quad 1 \leq i \leq s, \tag{1.2a}$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h \sum_{i=1}^{s} b_i \mathbf{k}_i \tag{1.2b}$$

definen un paso en la integración numérica de (1.1) con un método Runge-Kutta de s etapas. Los valores $\mathbf{y}_n$ son aproximaciones a la solución de (1.1) en $x_n = x_0 + nh$, $n = 0, 1, \ldots$

Además:

- Si $a_{ij} = 0$ para todo $i \leq j$, el método es explícito.
- Si $a_{ij} = 0$ para todo $i < j$ y al menos tenemos un $a_{ii} \neq 0$, el método es semiimplícito (también denominado semiexplícito).
- En el resto de casos, tenemos un método Runge-Kutta implícito.

Alternativamente, reescribiendo las ecuaciones de (1.2), se puede definir el método de la siguiente forma

$$\mathbf{Y}_i = \mathbf{y}_n + h \sum_{j=1}^{s} a_{ij} \mathbf{f}(x_n + c_j h, \mathbf{Y}_j), \ 1 \leq i \leq s, \tag{1.3a}$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h \sum_{i=1}^{s} b_i \mathbf{f}(x_n + c_i h, \mathbf{Y}_i) \tag{1.3b}$$

Los $\mathbf{Y}_i, i = 1, \ldots, s$, se denominan *etapas intermedias del método* y son aproximaciones a la solución de (1.1) en $x_n + c_i h$.

Los coeficientes del método se pueden agrupar, tomando

$$\mathcal{A} = (a_{ij})_{1\leq i,j\leq s}, \quad \mathbf{b} = (b_1, \ldots, b_s)^T, \quad \mathbf{c} = (c_1, \ldots, c_s)^T$$

como

$$\begin{array}{c|c} \mathbf{c} & \mathcal{A} \\ \hline & \mathbf{b}^T \end{array}, \tag{1.4}$$

que se denomina *tablero de Butcher*.

1.2. Existencia de una solución numérica para un método Runge-Kutta

Si el método Runge-Kutta es explícito, la solución de (1.2) existe siempre, pues $\mathbf{k}_1 = \mathbf{f}(x_n, \mathbf{y}_n)$, y para $i = 2, \ldots, s$, $\mathbf{k}_i$ depende de los $\mathbf{k}_1, \ldots, \mathbf{k}_{i-1}$ previos. Sin embargo, si el método es implícito, obtenemos un sistema en el que cada $\mathbf{k}_i$ depende de $\mathbf{k}_1, \ldots, \mathbf{k}_s$ implícitamente. Veamos que en este caso también existe una solución para (1.2).

Teorema 1.1. *Sea $\boldsymbol{f} : \mathbb{R} \times \mathbb{R}^D \to \mathbb{R}^D$ una función continua y lipschitziana (respecto de su segundo argumento), con constante de Lipschitz L. Si se cumple que*

$$h < \frac{1}{L \cdot \max\limits_{1\leq i\leq s} \sum\limits_{j=1}^{s} |a_{ij}|}, \tag{1.5}$$

entonces existe una única solución de (1.2)*, que puede ser obtenida mediante iteración de punto fijo. Además, si $\boldsymbol{f}(x, \boldsymbol{y}(x))$ es p veces diferenciable con continuidad, también lo son las $\boldsymbol{k}_i$, $i = 1, \ldots, s$, como funciones de h.*

Demostración. Probamos la existencia por iteración de punto fijo. Por (1.2), tenemos que

$$\mathbf{Y}_i = \mathbf{y}_n + h \sum_{j=1}^{s} a_{ij} \mathbf{f}\left(x_n + c_j h, \mathbf{Y}_j\right), \quad 1 \leq i \leq s.$$

Definimos $\mathcal{Y} = (\mathbf{Y}_1, \ldots, \mathbf{Y}_s)^T$ y $\mathcal{G} : \mathbb{R}^{sD} \to \mathbb{R}^{sD}$, con $\mathcal{G} = (\mathbf{G}_1, \ldots, \mathbf{G}_s)^T$ y

$$\mathbf{G}_i(\mathcal{Y}) = \mathbf{y}_n + h \sum_{j=1}^{s} a_{ij} \mathbf{f}\left(x_n + c_j h, \mathbf{Y}_j\right),$$

para $1 \leq i \leq s$. Si tomamos la norma $||\mathcal{Y}|| = \max\limits_{1\leq i\leq s} ||\mathbf{Y}_i||$, entonces

$$\begin{aligned} ||\mathbf{G}_i(\mathcal{Y}) - \mathbf{G}_i(\mathcal{Z})|| &= ||h \sum_{j=1}^{s} a_{ij} \left(\mathbf{f}(x_n + c_j h, \mathbf{Y}_j) - \mathbf{f}(x_n + c_j h, \mathbf{Z}_j)\right)|| \\ &\leq hL \sum_{j=1}^{s} |a_{ij}| \cdot ||\mathbf{Y}_j - \mathbf{Z}_j|| \\ &\leq hL \max_{1\leq j\leq s} ||\mathbf{Y}_j - \mathbf{Z}_j|| \max_{1\leq i\leq s} \sum_{j=1}^{s} |a_{ij}| \\ &= hL \max_{1\leq i\leq s} \sum_{j=1}^{s} |a_{ij}| \cdot ||\mathcal{Y} - \mathcal{Z}|| \end{aligned}$$

Tomando

$$hL \max_{1\leq i\leq s} \sum_{j=1}^{s} |a_{ij}| < 1,$$

es decir, si

$$h < \frac{1}{L \cdot \max\limits_{1\leq i\leq s} \sum\limits_{j=1}^{s} |a_{ij}|},$$

la aplicación $\mathcal{G} = (\mathbf{G}_1, \ldots, \mathbf{G}_s)^T$ es contractiva para la norma $||\mathcal{Y}|| = \max\limits_{1\leq i\leq s} ||\mathbf{Y}_i||$. Por el principio de la función contractiva, podemos asegurar la convergencia de la iteración de punto fijo y, por tanto, la existencia y unicidad de la solución.

Ahora demostraremos la diferenciabilidad de las funciones $\mathbf{k}_i(h)$, que se deduce del teorema de la función implícita.

Sea $\Phi(h, \mathcal{Y}) = \mathcal{Y} - \mathcal{G}(\mathcal{Y}) = 0$, es decir

$$\begin{pmatrix} \mathbf{Y}_1 - \mathbf{y}_n - h\sum\limits_{j=1}^{s} a_{1j}\mathbf{f}(x_n + c_j h, \mathbf{Y}_j) \\ \vdots \\ \mathbf{Y}_s - \mathbf{y}_n - h\sum\limits_{j=1}^{s} a_{sj}\mathbf{f}(x_n + c_j h, \mathbf{Y}_j) \end{pmatrix} = 0$$

Denotando por ∂_2 a la derivada parcial de una función respecto de su segundo argumento y por $\mathbf{f}_i = \mathbf{f}(x_n + c_i h, \mathbf{Y}_i), 1 \leq i \leq s$, obtenemos que la matriz jacobiana de Φ es

$$\mathbf{D}\Phi(h, \mathcal{Y}) = \begin{pmatrix} I - ha_{11}\partial_2\mathbf{f}_1 & -ha_{12}\partial_2\mathbf{f}_2 & \cdots & -ha_{1s}\partial_2\mathbf{f}_s \\ -ha_{21}\partial_2\mathbf{f}_1 & I - ha_{22}\partial_2\mathbf{f}_2 & \cdots & -ha_{2s}\partial_2\mathbf{f}_s \\ \vdots & \vdots & \ddots & \vdots \\ -ha_{s1}\partial_2\mathbf{f}_1 & -ha_{s2}\partial_2\mathbf{f}_2 & \cdots & -ha_{ss}\partial_2\mathbf{f}_s \end{pmatrix}$$

En $h = 0$, $\mathbf{D}\Phi(0, \mathcal{Y}) = I$ y $\Phi(0, \mathcal{Y}) = 0$, y por tanto, en un entorno de $h = 0$ existe una única función $\mathcal{Y}(h)$ que cumple $\Phi(h, \mathcal{Y}(h)) = 0$. Además, $\mathbf{f}$ es p veces diferenciable con continuidad, y entonces Φ también lo es. El teorema de la función implícita nos asegura que esto también se cumple para $\mathcal{Y}(h)$, y por tanto para $\mathbf{k}_i(h), 1 \leq i \leq s$. □

1.3. Orden de un método Runge-Kutta

Ahora definiremos el orden de un método Runge-Kutta y revisaremos las condiciones que tienen que satisfacer sus coeficientes para que el método tenga un orden determinado.

En esta memoria supondremos además que $\sum_{i=1}^{s} a_{ij} = c_i$. Esta condición hace que el uso de un método Runge-Kutta sobre una ecuación escalar no autónoma $\mathbf{y}' = \mathbf{f}(x, \mathbf{y})$ sea numéricamente equivalente a la aplicación de dicho método sobre el sistema autónomo análogo.

$$\begin{cases} x' = 1, \\ \mathbf{y}' = \mathbf{f}(x, \mathbf{y}). \end{cases} \tag{1.6}$$

Por tanto, estudiaremos sistemas de la siguiente forma

$$\mathbf{y}'(x) = \mathbf{f}(\mathbf{y}(x)), \quad x_0 \leq x \leq x_f, \tag{1.7a}$$

$$\mathbf{y}(x_0) = \mathbf{y}_0. \tag{1.7b}$$

Definición 1.2. Sea $\mathbf{f}$ una función como en (1.1), arbitraria y suficientemente regular. Entonces, un método Runge-Kutta tiene orden p si

$$||\mathbf{y}(x_n + h) - \mathbf{y}_{n+1}|| = \mathcal{O}(h^{p+1}), \quad h \to 0,$$

donde $\mathbf{y}_{n+1}$ es la solución numérica obtenida mediante (1.2). Esto equivale a decir que los desarrollos de Taylor en potencias de h de $\mathbf{y}(x_n + h)$ e $\mathbf{y}_{n+1}(h)$ coinciden hasta los términos de orden p.

Tomamos el sistema (1.7), donde

$$\mathbf{y}(x) = (y^1(x), \dots, y^D(x))^T, \quad \mathbf{y}_n(h) = (y_n^1(h), \dots, y_n^D(h))^T, \quad \mathbf{f}(\mathbf{y}) = (f^1(\mathbf{y}), \dots, f^D(\mathbf{y}))^T.$$

Con esta notación, el desarrollo de Taylor de la componente J-ésima de $\mathbf{y}(x+h)$ sería, para $1 \leq J \leq D$,

$$\begin{aligned} y^J(x+h) = y^J(x) + hf^J + \frac{h^2}{2} \sum_{K=1}^{D} \frac{\partial f^J}{\partial x^K} f^K \\ + \frac{h^3}{6} \left(\sum_{K=1}^{D} \sum_{L=1}^{D} \frac{\partial^2 f^J}{\partial x^K \partial x^L} f^K f^L + \sum_{K=1}^{D} \frac{\partial f^J}{\partial x^K} \sum_{L=1}^{D} \frac{\partial f^K}{\partial x^L} f^L \right) + \mathcal{O}(h^4), \end{aligned}$$

donde la función $\mathbf{f}$ y sus derivadas están evaluadas en $\mathbf{y}(x)$. Reescribiéndolo, tendríamos

$$y^J(x+h) = y^J(x) + hf^J + \frac{h^2}{2} \sum_{K} f_K^J f^K + \frac{h^3}{6} \left(\sum_{K,L} f_{KL}^J f^K f^L + \sum_{K,L} f_K^J f_L^K f^L \right) + \mathcal{O}(h^4).$$

Si se busca un método de orden más alto que 3, la construcción del desarrollo de Taylor de cada y^J se complica. Para simplificar estos resultados, se introducirán los árboles con raíz.

Formalmente, los árboles con raíz se definen como clases de equivalencia dentro del conjunto de árboles etiquetados con raíz, pero para estudiar las condiciones de orden de los métodos Runge-Kutta será suficiente considerar su representación gráfica.

Definición 1.3. Definimos un árbol con raíz como un grafo conexo, sin ciclos, que tiene un nodo principal que denominaremos raíz. Dicho árbol consistirá en ramas que conectarán cada nodo (los *hijos*) con un único nodo del nivel previo (el *padre*). Por tanto, la raíz será el único nodo en el primer nivel. El árbol tendrá orden q si tiene q nodos dispuestos en diferentes alturas.

El árbol formado por un único nodo se denota por τ, y un árbol general se denota por $t = [t_1, \dots, t_m]$, donde $t_1, \dots, t_m$ son los árboles con raíz que se obtienen al suprimir en t su raíz y las m ramas que parten de ella [1].

Definición 1.4. Dado un árbol $t = [t_1, \dots, t_m]$, definimos de manera recurrente las siguientes funciones:

- **Orden**:

$$\rho(t) = 1 + \sum_{i=1}^{m} \rho(t_i), \quad \rho(\tau) = 1.$$

- **Densidad**:

$$\gamma(t) = \rho(t) \prod_{i=1}^{m} \gamma(t_i), \quad \gamma(\tau) = 1.$$

[1] En el Apéndice A se adjuntan los árboles de hasta orden cinco para estudiar el orden de los métodos Runge-Kutta.

- **Diferencial elemental**: $\mathbf{F}(t) = \left(F^1(t), \ldots, F^D(t)\right)^T$, donde $\mathbf{F}(\tau) = \mathbf{f}$, y su componente J-ésima es, para $1 \leq J \leq D$,

$$F^J(t) = \sum_{J_1,\ldots,J_m=1}^{D} \frac{\partial^m f^J}{\partial x^{J_1} \cdots \partial x^{J_m}} F^{J_1}(t_1) \cdots F^{J_m}(t_m).$$

- **Peso elemental en la componente i-ésima**:

$$\Phi_i(t) = \sum_{j_1,\ldots,j_m=1}^{s} a_{ij_1} \Phi_{j_1}(t_1) \cdots a_{ij_m} \Phi_{j_m}(t_m), \quad \Phi_i(\tau) = 1.$$

- **Peso elemental**:

$$\Phi(t) = \sum_{i=1}^{s} b_i \Phi_i(t).$$

Veamos ahora un ejemplo para ilustrar estos conceptos.

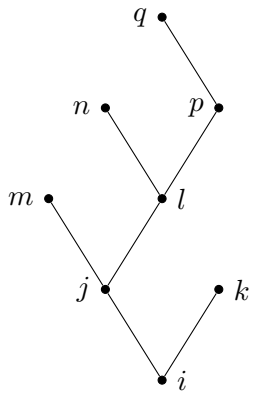


Figura 1.1: Árbol con raíz de 8 nodos.

Ejemplo 1.1. Estudiamos las funciones previamente definidas aplicadas al árbol con raíz de la Figura 1.1:

- $t = [\tau, [\tau, [\tau, [\tau]]]]$,
- $\rho(t) = 8$,
- $\gamma(t) = 8 \cdot 6 \cdot 4 \cdot 2 = 384$,
- $\Phi(t) = \sum_{i,j,k,l,m,n,p,q=1}^{s} b_i a_{ik} a_{ij} a_{jm} a_{jl} a_{ln} a_{lp} a_{pq} = \sum_{i,j,l,p=1}^{s} b_i c_i a_{ij} c_j a_{jl} c_l a_{lp} c_p.$

  Esta última igualdad se obtiene del hecho de que $\sum_{j=1}^{s} a_{ij} = c_i$, para $i = 1, \ldots, s$.

Veamos ahora el resultado sobre el orden de los métodos Runge-Kutta para el cual se ha introducido la teoría de los árboles con raíz.

Teorema 1.2. *Un método Runge-Kutta* (1.2) *tiene orden p si, y sólo si,*

$$\Phi(t) = \frac{1}{\gamma(t)} \tag{1.8}$$

para todos los árboles con raíz t de orden $\leq p$.

Demostración. La demostración se encuentra en [1], Capítulo II.2, Teorema 2.13. □

Capítulo 2

Métodos de colocación

2.1. Métodos implícitos e hipótesis simplificadoras

A partir de ahora solo consideraremos métodos Runge-Kutta implícitos. Los primeros métodos implícitos fueron usados por Cauchy en 1824 insertando el teorema del valor medio en la integral

$$\mathbf{y}(x_{n+1}) = \mathbf{y}(x_n) + \int_{x_n}^{x_{n+1}} \mathbf{f}(x, \mathbf{y}(x))\, dx, \tag{2.1}$$

para obtener el denominado **θ-método**

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}(x_n + \theta h, \mathbf{y}_n + \theta(\mathbf{y}_{n+1} - \mathbf{y}_n)), \tag{2.2}$$

donde $0 \leq \theta \leq 1$ y $h = x_{n+1} - x_n$.

Dependiendo de los valores de θ, obtenemos diferentes métodos:

- Si $\theta = 0$, obtenemos el *método de Euler explícito*, $\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}(x_n, \mathbf{y}_n)$.
- Si $\theta = 1$, obtenemos el *método de Euler implícito*, $\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}(x_{n+1}, \mathbf{y}_{n+1})$.
- Si $\theta = \dfrac{1}{2}$ y elegimos $\mathbf{k}_1 = \dfrac{\mathbf{y}_{n+1} - \mathbf{y}_n}{h}$, obtenemos la *regla implícita del punto medio*,

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{f}(x_n + \frac{h}{2}, \mathbf{y}_n + \frac{h}{2}\mathbf{k}_1), \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + h\mathbf{k}_1. \end{aligned}$$

Veamos un ejemplo de obtención de un método Runge-Kutta implícito.

Ejemplo 2.1. Consideremos ahora la construcción de un método Runge-Kutta implícito de dos etapas con $c_1 = 0$ y el mayor orden posible. Aplicando la teoría de árboles (ver Apéndice A), obtenemos que las condiciones de orden en este caso son

$$\begin{cases} b_1 + b_2 = 1, \\ b_2 c_2 = \frac{1}{2}, \\ b_2 c_2^2 = \frac{1}{3}, \\ b_1 a_{12} c_2 + b_2 a_{22} c_2 = \frac{1}{6}. \end{cases}$$

Dividiendo la tercera ecuación entre la segunda, obtenemos que

$$c_2 = \frac{1/3}{1/2} = \frac{2}{3}.$$

A partir de c_2, podemos hallar b_1 y b_2:

$$b_2 = \frac{1}{2c_2} = \frac{1}{2 \cdot \frac{2}{3}} = \frac{3}{4}, \qquad b_1 = 1 - b_2 = 1 - \frac{3}{4} = \frac{1}{4}.$$

Sustituyendo los valores obtenidos en la cuarta ecuación de las condiciones de orden previamente enunciadas, obtenemos

$$b_1a_{12} + b_2a_{22} = \frac{1}{6c_2} \Rightarrow \frac{1}{4}a_{12} + \frac{3}{4}a_{22} = \frac{1}{6 \cdot \frac{2}{3}} = \frac{1}{4} \Rightarrow a_{12} + 3a_{22} = 1.$$

Además, como sabemos que $c_i = \sum_{j=1}^{s} a_{ij}$, $1 \leq i \leq 2$, obtenemos el siguiente sistema que deben satisfacer los coeficientes a_{ij}:

$$\begin{cases} a_{11} + a_{12} = 0, \\ a_{21} + a_{22} = \frac{2}{3}, \\ a_{12} + a_{22} = 1. \end{cases}$$

Fijando uno de los coeficientes, por ejemplo, $a_{22} = \alpha$, podemos obtener el resto de a_{ij} en función de α:

$$a_{12} = 1 - 3\alpha, \quad a_{11} = 3\alpha - 1, \quad a_{21} = \frac{2 - 3\alpha}{3}.$$

De esta forma, obtenemos una familia uniparamétrica de métodos de 2 etapas y orden 3 con el siguiente tablero de Butcher:

0	$3\alpha - 1$	$1 - 3\alpha$
2/3	$(2 - 3\alpha)/3$	α
	1/4	3/4

Tabla 2.1.

Las ecuaciones para avanzar un paso de longitud h con un método de esta familia son

$$\begin{cases} \mathbf{k}_1 = \mathbf{f}\left(x_n, \mathbf{y}_n + h(3\alpha - 1)(\mathbf{k}_1 - \mathbf{k}_2)\right), \\ \mathbf{k}_2 = \mathbf{f}\left(x_n + \frac{2h}{3}, \mathbf{y}_n + h\left(\frac{2-3\alpha}{3}\mathbf{k}_1 + \alpha\mathbf{k}_2\right)\right), \\ \mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{4}(\mathbf{k}_1 + 3\mathbf{k}_2). \end{cases}$$

Si tomamos $\alpha = 1/3$ obtenemos el método conocido como Radau I, y con $\alpha = 5/12$, obtenemos el método de Radau IA. Como ya se ha indicado previamente, ambos métodos tienen 2 etapas y orden 3.

A la hora de construir métodos Runge-Kutta implícitos de orden alto, será conveniente imponer unas hipótesis (conocidas como *hipótesis simplificadoras*) para facilitar los cálculos necesarios para la obtención de los coeficientes del método, ya que si aumentamos el número de etapas serán complicados de realizar sin ninguna técnica de este tipo. Junto con las condiciones de orden para que el método tenga orden p,

- $A(p)$: $\quad \Phi(t) = \dfrac{1}{\gamma(t)}$ para todos los árboles con raíz t de orden $\leq p$,

consideraremos las siguientes familias de hipótesis simplificadoras:

- $B(p)$: $\quad \phi^{k-1}(t) \equiv \displaystyle\sum_{i=1}^{s} b_i c_i^{k-1} = \frac{1}{k}, \quad k = 1, \ldots, p.$

- $C(\eta)$: $\quad \displaystyle\sum_{j=1}^{s} a_{ij} c_j^{k-1} = \frac{c_i^k}{k}, \quad 1 \leq i \leq s, \quad k = 1, \ldots, \eta.$

- $D(\zeta)$: $\quad \displaystyle\sum_{i=1}^{s} b_i c_i^{k-1} a_{ij} = \frac{b_j(1 - c_j^k)}{k}, \quad 1 \leq j \leq s, \quad k = 1, \ldots, \zeta.$

Veamos que, si se cumplen ciertos conjuntos de hipótesis simplificadoras, entonces se cumplirán también las condiciones de orden $A(p)$ necesarias para que el método tenga orden p. Pero antes, veamos las implicaciones que se siguen de imponer cada conjunto de hipótesis simplificadoras.

- $B(p)$ quiere decir que la fórmula de cuadratura con pesos b_i y abscisas c_i tiene grado de exactitud p. Equivalentemente, esto quiere decir que $A(p)$ se satisface para todos los arbustos $t = [\tau, \ldots, \tau]$ hasta orden p.

- $C(\eta)$ implica que los árboles como los de la Figura 2.1, con $q \leq \eta$, proporcionan condiciones de orden equivalentes.

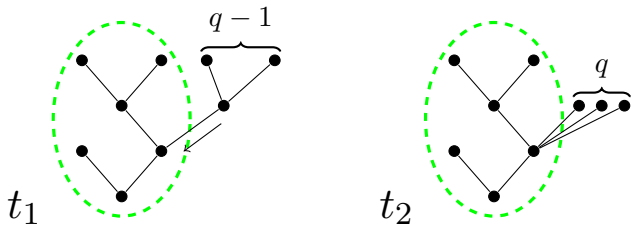


Figura 2.1: Árboles con raíz que corresponden a condiciones de orden equivalentes si se cumplen las hipótesis simplificadoras $C(q)$.

Sean t_1 y t_2 dos árboles con raíz como los descritos en la Figura 2.1, donde la parte que tienen en común t' (encerrada en la figura por las líneas discontinuas verdes) puede ser arbitraria. Si aplicamos $C(\eta)$, obtenemos que

$$\Phi(t_1) = \sum_{i=1}^{s} b_i \Phi_i(t_1) = \sum_{i=1}^{s} b_i \Phi_i(t') \left(\sum_{k=1}^{s} a_{jk} c_k^{q-1} \right) = \sum_{i=1}^{s} b_i \Phi_i(t') \frac{c_j^q}{q} = \frac{1}{q} \sum_{i=1}^{s} b_i \Phi_i(t_2) = \frac{1}{q} \Phi(t_2).$$

Además, es fácil calcular que $\gamma(t_1) = q \cdot \gamma(t_2)$, de donde se obtiene de forma inmediata que las condiciones de orden son equivalentes.

- Si $D(\zeta)$ se cumple, y además $q \leq \zeta$, entonces las condiciones de orden de los dos árboles de la derecha en la Figura 2.2 implican las del árbol de la izquierda.

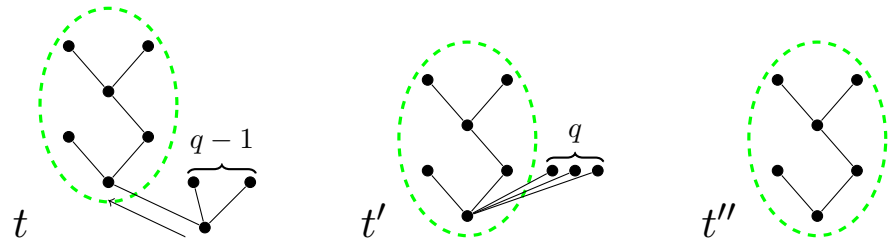


Figura 2.2: Árboles con raíz que corresponden a condiciones de orden equivalentes si se cumplen las hipótesis simplificadoras $D(q)$.

Sean t, t' y t'' tres árboles con raíz como los descritos en la Figura 2.2, donde la parte que tienen en común t'' (encerrada en la figura por las líneas discontinuas verdes) puede ser arbitraria.

Se pueden calcular las siguientes igualdades a partir de las definiciones de orden y densidad:

- $\gamma(t) = \gamma(t') \cdot \rho(t'') \Rightarrow \gamma(t') = \dfrac{\gamma(t)}{\rho(t'')}$,
- $\gamma(t) = \gamma(t'') \cdot \rho(t) \Rightarrow \gamma(t'') = \dfrac{\gamma(t)}{\rho(t)}$,
- $\rho(t) = \rho(t') = \rho(t'') + q$.

Gracias a estas igualdades, podemos concluir que, si $\Phi(t') = \dfrac{1}{\gamma(t')}$ y $\Phi(t'') = \dfrac{1}{\gamma(t'')}$, entonces

$\Phi(t) = \dfrac{1}{\gamma(t)}$:

$$\Phi(t) = \sum_{i=1}^{s} b_i \Phi_i(t) = \sum_{i=1}^{s} b_i c_i^{q-1} \left(\sum_{j=1}^{s} a_{ij} \Phi_j(t'') \right) = \sum_{j=1}^{s} \Phi_j(t'') \left(\sum_{i=1}^{s} b_i c_i^{q-1} a_{ij} \right)$$
$$= \sum_{j=1}^{s} \Phi_j(t'') \cdot \frac{b_j(1-c_j^q)}{q} = \frac{1}{q} \left(\sum_{j=1}^{s} b_j \Phi_j(t'') - \sum_{j=1}^{s} b_j c_j^q \Phi_j(t'') \right) = \frac{1}{q} \left(\Phi(t'') - \Phi(t') \right)$$
$$= \frac{1}{q} \left(\frac{1}{\gamma(t'')} - \frac{1}{\gamma(t')} \right) = \frac{1}{q} \left(\frac{\rho(t)}{\gamma(t)} - \frac{\rho(t'')}{\gamma(t)} \right) = \frac{1}{q} \left(\frac{\rho(t'')+q}{\gamma(t)} - \frac{\rho(t'')}{\gamma(t)} \right) = \frac{1}{\gamma(t)}.$$

Teorema 2.1. *Si $B(p)$, $C(\eta)$ y $D(\zeta)$ se cumplen, con $p \le 2\eta + 2$ y $p \le \zeta + \eta + 1$, entonces el método tiene orden p.*

Demostración. $C(\eta)$ implica que es suficiente considerar árboles $t = [t_1, \ldots, t_m]$ de orden $\le p$, donde los subárboles $t_1, \ldots, t_m$ son iguales a τ o hay exactamente un t_i diferente de τ, es decir, tendría orden $\rho(t_i) \ge \eta + 1$.

Si existiese otro $t_j, j \ne i$, tal que $\rho(t_j) \ge \eta + 1$ se contradiría la hipótesis $p \le 2\eta + 2$.

Para los árboles de orden $< \eta + 1$ aplicamos $C(\eta)$. Si existiese un i tal que $\rho(t_i) \ge \eta + 1$, el número de subárboles τ será $\le \zeta - 1$, pues $p \le \zeta + \eta + 1$, y podemos aplicar $D(\zeta)$.

Por tanto, después de estas reducciones, solo quedan los arbustos, que satisfacen $B(p)$. □

Lema 2.1. *Si se cumple $C(\eta)$, entonces las etapas intermedias*

$$\boldsymbol{Y}_i = \boldsymbol{y}_n + h \sum_{j=1}^{s} a_{ij} \boldsymbol{f}(x_n + c_j h,\, \boldsymbol{Y}_j)\,, \quad 1 \le i \le s \tag{2.3}$$

de un método Runge-Kutta satisfacen

$$\boldsymbol{Y}_i - \boldsymbol{y}\,(x_n + c_i h) = \mathcal{O}(h^{\eta+1}), \tag{2.4}$$

siendo $\boldsymbol{y}(x)$ la solución de (1.1a) *con condición inicial $\boldsymbol{y}(x_n) = \boldsymbol{y}_n$. Es decir, las etapas intermedias son aproximaciones de orden η a la solución en los valores intermedios $x_n + c_i h$, $1 \le i \le s$.*

Demostración. El desarrollo de Taylor de $\mathbf{y}(x_n + c_i h)$ es

$$\mathbf{y}(x_n + c_i h) = \mathbf{y}_n + c_i h \mathbf{y}'(x_n) + \frac{(c_i h)^2}{2} \mathbf{y}''(x_n) + \cdots + \frac{(c_i h)^\eta}{\eta!} \mathbf{y}^{(\eta)}(x_n) + \mathcal{O}(h^{\eta+1})$$
$$= \mathbf{y}_n + h \left(c_i \mathbf{y}'(x_n) + \frac{c_i^2}{2} h \mathbf{y}''(x_n) + \cdots + \frac{c_i^\eta}{\eta} \frac{h^{\eta-1}}{(\eta-1)!} \mathbf{y}^{(\eta)}(x_n) \right) + \mathcal{O}(h^{\eta+1}).$$

Utilizando $C(\eta)$, obtenemos

$$\mathbf{y}(x_n + c_i h) = \mathbf{y}_n + h \left(\sum_{j=1}^{s} a_{ij} \mathbf{y}'(x_n) + h \sum_{j=1}^{s} a_{ij} c_j \mathbf{y}''(x_n) + \cdots + \frac{h^{\eta-1}}{(\eta-1)} \sum_{j=1}^{s} a_{ij} c_j^{\eta-1} \mathbf{y}^{(\eta)}(x_n) \right) + \mathcal{O}(h^{\eta+1})$$
$$= \mathbf{y}_n + h \sum_{j=1}^{s} a_{ij} \left(\mathbf{y}'(x_n) + c_j h \mathbf{y}''(x_n) + \cdots + \frac{(c_j h)^{\eta-1}}{(\eta-1)!} \mathbf{y}^{(\eta)}(x_n) \right) + \mathcal{O}(h^{\eta+1})$$
$$= \mathbf{y}_n + h \sum_{j=1}^{s} a_{ij} \mathbf{y}'(x_n + c_j h) + \mathcal{O}(h^{\eta+1})$$
$$= \mathbf{y}_n + h \sum_{j=1}^{s} a_{ij} \mathbf{f}(x_n + c_j h, \mathbf{y}(x_n + c_j h)) + \mathcal{O}(h^{\eta+1}).$$

De aquí deducimos que

$$\mathbf{Y}_i - \mathbf{y}(x_n + c_i h) = h \sum_{j=1}^{s} a_{ij} \left[\mathbf{f}(x_n + c_j h, \mathbf{Y}_j) - \mathbf{f}(x_n + c_j h, \mathbf{y}(x_n + c_j h)) \right] + \mathcal{O}(h^{\eta+1}).$$

Como $\mathbf{f}$ es una función Lipschitziana con constante de Lipschitz L, tenemos que

$$||\mathbf{Y}_i - \mathbf{y}(x_n + c_i h)|| \leq hL \sum_{j=1}^{s} |a_{ij}| \cdot ||\mathbf{Y}_j - \mathbf{y}(x_n + c_j h)|| + \mathcal{O}(h^{\eta+1}).$$

Sean $\mathcal{M}$ la matriz con coeficientes $(|a_{ij}|)_{1 \leq i,j \leq s}$ e $\mathcal{Y} = (||\mathbf{Y}_1 - \mathbf{y}(x_n + c_1 h)||, \ldots, ||\mathbf{Y}_s - \mathbf{y}(x_n + c_s h)||)^T$. Entonces

$$\mathcal{Y} \leq hL\mathcal{M}\mathcal{Y} + \mathcal{O}(h^{\eta+1}) \Rightarrow \mathcal{Y} - hL\mathcal{M}\mathcal{Y} \leq \mathcal{O}(h^{\eta+1}) \Rightarrow (I - hL\mathcal{M})\mathcal{Y} \leq \mathcal{O}(h^{\eta+1}).$$

Si se cumple la desigualdad estricta (1.5), entonces tendremos que $||hL\mathcal{M}|| < 1$, y por tanto $I - hL\mathcal{M}$ será no singular, con

$$||(I - hL\mathcal{M})^{-1}|| \leq \frac{1}{1 - ||hL\mathcal{M}||}.$$

De aquí podemos concluir que $\mathcal{Y} = \mathcal{O}(h^{\eta+1})$, es decir, $\mathbf{Y}_i - \mathbf{y}(x_n + c_i h) = \mathcal{O}(h^{\eta+1})$, $1 \leq i \leq s$. □

2.2. Métodos de colocación

Definición 2.1. Sea s un entero positivo y $c_1, \ldots, c_s$ números reales distintos (generalmente entre 0 y 1). El correspondiente *polinomio de colocación* $\mathbf{u}(x)$ de grado s en el intervalo $[x_n, x_n + h]$ es el que satisface las igualdades siguientes

$$\mathbf{u}(x_n) = \mathbf{y}_n, \tag{2.5a}$$
$$\mathbf{u}'(x_n + c_i h) = \mathbf{f}(x_n + c_i h, \mathbf{u}(x_n + c_i h)), \quad i = 1, \ldots, s. \tag{2.5b}$$

Se toma entonces como aproximación numérica a la solución de (1.1) en $x_n + h$ la dada por

$$\mathbf{y}_{n+1} = \mathbf{u}(x_n + h), \tag{2.6}$$

es decir, el valor del polinomio de colocación en el extremo derecho del intervalo.

No es necesario que los coeficientes c_i sean distintos, aunque lo supondremos así. En caso de que algunos coincidiesen, aplicando las condiciones de colocación obtendríamos métodos que involucran derivadas de mayor orden.

Veamos que un método de colocación puede interpretarse como la aplicación de un método Runge-Kutta.

Teorema 2.2. *El método de colocación* (2.5) *es equivalente al método Runge-Kutta implícito de s etapas* (1.2) *con coeficientes*

$$a_{ij} = \int_0^{c_i} l_j(t)dt, \quad b_j = \int_0^1 l_j(t)dt, \quad i, j = 1, \ldots, s, \tag{2.7}$$

donde $l_j(t)$ es el polinomio de la base de Lagrange asociado al coeficiente c_j, es decir,

$$l_j(t) = \prod_{k \neq j} \frac{t - c_k}{c_j - c_k}. \tag{2.8}$$

Demostración. Sea $\mathbf{k}_i = \mathbf{u}'(x_n + c_i h)$. Como $\mathbf{u}'(x_n + th)$ es un polinomio de grado menor o igual que $s-1$, coincide con su polinomio interpolador en s nodos distintos, que puede escribirse como

$$\mathbf{u}'(x_n + th) = \sum_{j=1}^{s} \mathbf{k}_j l_j(t).$$

Sabemos que

$$\begin{aligned}\mathbf{u}(x_n + c_i h) &= \mathbf{y}_n + h\int_0^{c_i} \mathbf{u}'(x_n + th)dt = \mathbf{y}_n + h\int_0^{c_i} \sum_{j=1}^{s} \mathbf{k}_j l_j(t)dt \\ &= \mathbf{y}_n + h\sum_{j=1}^{s} \mathbf{k}_j \int_0^{c_i} l_j(t)dt = \mathbf{y}_n + h\sum_{j=1}^{s} a_{ij}\mathbf{k}_j,\end{aligned}$$

donde en la última igualdad se ha usado (2.7). Sustituyendo en (2.5) obtenemos

$$\mathbf{k}_i = \mathbf{u}'(x_n + c_i h) = \mathbf{f}(x_n + c_i h, \mathbf{u}(x_n + c_i h)) = \mathbf{f}(x_n + c_i h, \mathbf{y}_n + h\sum_{j=1}^{s} a_{ij}\mathbf{k}_j).$$

Razonando de forma similar para (2.6), obtenemos

$$\begin{aligned}\mathbf{y}_{n+1} = \mathbf{u}(x_n + h) &= \mathbf{y}_n + h\int_0^{1} \mathbf{u}'(x_n + th)dt = \mathbf{y}_n + h\int_0^{1} \sum_{j=1}^{s} \mathbf{k}_j l_j(t)dt \\ &= \mathbf{y}_n + h\sum_{j=1}^{s} \mathbf{k}_j \int_0^{1} l_j(t)dt = \mathbf{y}_n + h\sum_{j=1}^{s} b_j\mathbf{k}_j.\end{aligned}$$

El método obtenido se corresponde con un método Runge-Kutta implícito de s etapas con los coeficientes definidos por (2.7). □

Como consecuencia, queda probada la existencia y unicidad del polinomio de colocación para valores de h como en (1.7), pues cumple las mismas condiciones que un método Runge-Kutta.

Teorema 2.3. *Un método Runge-Kutta implícito donde todos los coeficientes c_i son diferentes y de orden al menos s es un método de colocación si, y solo si, se cumple $C(s)$.*

Demostración. Si el método Runge-Kutta tiene orden s, entonces se cumple $B(s)$, es decir,

$$\sum_{i=1}^{s} b_i\ c_i^{k-1} = \frac{1}{k} = \int_0^1 t^{k-1}dt, \quad k = 1, \ldots, s,$$

y la fórmula de cuadratura será exacta para polinomios p de grado $\leq s-1$

$$\sum_{i=1}^{s} b_i p(c_i) = \int_0^1 p(t)dt.$$

Si p es un polinomio de la base de polinomios de Lagrange, entonces los coeficientes b_i siempre satisfarán

$$b_i = \int_0^1 l_i(t)dt.$$

Veamos pues que $C(s)$ se cumple si, y sólo si, también se cumple

$$a_{ij} = \int_0^{c_i} l_j(t)dt.$$

($\Rightarrow$) Supongamos que

$$a_{ij} = \int_0^{c_i} l_j(t)dt.$$

Entonces, para todo polinomio p de grado menor o igual que $s-1$ tenemos que

$$\int_0^{c_i} p(t)dt = \int_0^{c_i} \sum_{j=1}^{s} p(c_j) l_j(t)dt = \sum_{j=1}^{s} p(c_j) \int_0^{c_i} l_j(t)dt = \sum_{j=1}^{s} a_{ij} p(c_j),$$

para $1 \leq i \leq s$. En particular, si tomamos $p(t) = t^{k-1}$, $k = 1, \ldots, s$, obtendremos $C(s)$

$$\int_0^{c_i} t^{k-1}dt = \frac{c_i^k}{k} = \sum_{j=1}^{s} a_{ij} c_j^{k-1}, \quad 1 \leq i \leq s.$$

($\Leftarrow$) Si se cumple $C(s)$, entonces tenemos

$$\sum_{j=1}^{s} a_{ij} c_j^{k-1} = \frac{c_i^k}{k}, \quad 1 \leq i \leq s, \quad k = 1, \ldots, s. \tag{2.9}$$

Notamos que, si se cumple (2.9), los coeficientes a_{ij} se obtienen como solución de los s sistemas lineales con s ecuaciones y s incógnitas siguientes

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ c_1 & c_2 & c_3 & \cdots & c_s \\ c_1^2 & c_2^2 & c_3^2 & \cdots & c_s^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_1^{s-1} & c_2^{s-1} & c_3^{s-1} & \cdots & c_s^{s-1} \end{pmatrix} \begin{pmatrix} a_{i1} \\ a_{i2} \\ a_{i3} \\ \vdots \\ a_{is} \end{pmatrix} = \begin{pmatrix} c_i \\ c_i^2/2 \\ c_i^3/3 \\ \vdots \\ c_i^s/s \end{pmatrix}, \tag{2.10}$$

en los que la matriz del sistema es una matriz de Vandermonde fija, y el término independiente varía con i. Como hemos supuesto que los coeficientes c_i son distintos entre sí para $i = 1, \ldots, s$, esta matriz es invertible. Con el cumplimiento de $C(s)$, también se tiene que

$$\sum_{j=1}^{s} a_{ij} c_j^{k-1} = \frac{c_i^k}{k} = \int_0^{c_i} t^{k-1}dt, \quad 1 \leq i \leq s, \quad k = 1, \ldots, s,$$

y, por tanto,

$$\sum_{j=1}^{s} a_{ij} p(c_j) = \int_0^{c_i} p(t)dt, \quad 1 \leq i \leq s, \tag{2.11}$$

para todo polinomio p de grado menor o igual que $s-1$. Si p es un polinomio de la base de polinomios de Lagrange, entonces obtendríamos la expresión que buscábamos para los coeficientes a_{ij}, descrita en (2.7), y el método Runge-Kutta implícito sería un método de colocación.

□

Teorema 2.4. *Sea $M(t) = \prod_{i=1}^{s}(t - c_i)$ y supongamos que M es ortogonal a todos los polinomios de grado menor o igual que $r-1$,*

$$\int_0^1 M(t)t^{q-1}dt = 0, \quad q = 1, \ldots, r, \tag{2.12}$$

Entonces, el método de colocación (2.5) *tiene orden $p = s + r$.*

Demostración. Por el Teorema 2.3, sabemos que será un método de colocación de orden al menos s si, y sólo si, se cumple $C(s)$. Además, si el método tiene orden $p = s + r$, entonces se cumplirá $B(s + r)$. Veamos que esto es cierto.

Sea $M(t) = \prod_{i=1}^{s}(t - c_i)$ de grado s tal que (2.12) se cumple, es decir, que M es ortogonal a polinomios de grado $\leq r - 1$. Por el Teorema 2.2, sabemos que $\int_0^1 p(t)dt = \sum_{j=1}^{s} b_j p(c_j)$, siendo b_j el coeficiente definido por la ecuacion (2.7) y $p(t)$ un polinomio de grado $\leq s + r - 1$. Tomando $p(t) = t^{k-1},\ k = 1, \ldots, s + r$, obtenemos que

$$\int_0^1 t^{k-1}dt = \frac{1}{k} = \sum_{j=1}^{s} b_j c_j^{k-1}.$$

Para poder concluir que el método tiene orden $p = s + r$, necesitamos ver que $D(r)$ se cumple, y por tanto mediante el Teorema 2.1 habremos terminado. Sea V_s la matriz de Vandermonde de tamaño $s \times s$ en (2.10). Definimos los vectores $\mathbf{u}^{(k)} = (u_1^{(k)}, \ldots, u_s^{(k)})^T$ y $\mathbf{v}^{(k)} = (v_1^{(k)}, \ldots, v_s^{(k)})^T$, $k = 1, \ldots, r$, donde la componente j-ésima de cada uno de estos vectores se corresponde con $u_j^{(k)} := \sum_{i=1}^{s} b_i c_i^{k-1} a_{ij}$ y $v_j^{(k)} := b_j(1 - c_j^k)/k$, $j = 1, \ldots, s$. Veamos que al multiplicar ambos vectores por la matriz de Vandermonde V_s el resultado obtenido es el mismo:

$$\begin{aligned}
(V_s \cdot \mathbf{u}^{(k)})_l &= \sum_{j=1}^{s} c_j^{l-1} \sum_{i=1}^{s} b_i c_i^{k-1} a_{ij} = \sum_{i=1}^{s} b_i c_i^{k-1} \sum_{j=1}^{s} a_{ij} c_j^{l-1} = \sum_{i=1}^{s} b_i c_i^{k-1} \cdot \frac{c_i^l}{l} = \frac{1}{l} \sum_{i=1}^{s} b_i c_i^{l+k-1} \\
&= \frac{1}{l(l+k)}, \\
(V_s \cdot \mathbf{v}^{(k)})_l &= \sum_{j=1}^{s} c_j^{l-1} \cdot \frac{b_j(1 - c_j^k)}{k} = \frac{1}{k} \sum_{j=1}^{s} \left(b_j c_j^{l-1} - b_j c_j^{l+k-1} \right) = \frac{1}{k} \left(\sum_{j=1}^{s} b_j c_j^{l-1} - \sum_{j=1}^{s} b_j c_j^{l+k-1} \right) \\
&= \frac{1}{k} \left(\frac{1}{l} - \frac{1}{l+k} \right) = \frac{1}{l(l+k)}.
\end{aligned}$$

Por tanto, $V_s \cdot \mathbf{u}^{(k)} = V_s \cdot \mathbf{v}^{(k)}$. Además, como V_s es invertible, tenemos que $\mathbf{u}^{(k)} = \mathbf{v}^{(k)}$, se cumple $D(r)$ y entonces el orden del método de colocación (2.5) es $p = s + r$. □

Lema 2.2. *Si tenemos un método Runge-Kutta* (1.2) *de s etapas con coeficientes $b_1, \ldots, b_s$ no nulos y $c_1, \ldots, c_s$ distintos, entonces*

(1) $C(s) \wedge B(s + r) \Rightarrow D(r)$,

(2) $D(s) \wedge B(s + r) \Rightarrow C(r)$.

y el método Runge-Kutta tendrá orden $p = s + r$.

Demostración. El apartado (1) se ha visto en la demostración del Teorema 2.4. Para demostrar el apartado (2) se sigue un razonamiento análogo al de (1). Definimos los vectores $\mathbf{u}^{(k)} = (u_1^{(k)}, \ldots, u_s^{(k)})^T$ y $\mathbf{v}^{(k)} = (v_1^{(k)}, \ldots, v_s^{(k)})^T$, $k = 1, \ldots, r$, donde la componente i-ésima de cada uno de estos vectores se corresponde con $u_i^{(k)} := \sum_{j=1}^{s} a_{ij} c_j^{k-1}$ y $v_i^{(k)} := c_i^k/k$, $i = 1, \ldots, s$. Entonces, si denotamos por B a la matriz diagonal cuyos elementos diagonales son las componentes de $\mathbf{b}$, se tiene

$$\begin{aligned}
(V_s \cdot B \cdot \mathbf{u}^{(k)})_l &= \sum_{i=1}^{s} b_i c_i^{l-1} \sum_{j=1}^{s} a_{ij} c_j^{k-1} = \sum_{j=1}^{s} c_j^{k-1} \sum_{i=1}^{s} b_i c_i^{l-1} a_{ij} = \sum_{j=1}^{s} c_j^{k-1} \cdot \frac{b_j(1 - c_j^l)}{l} \\
&= \frac{1}{l} \left(\sum_{j=1}^{s} b_j c_j^{k-1} - \sum_{j=1}^{s} b_j c_j^{l+k-1} \right) = \frac{1}{l} \left(\frac{1}{k} - \frac{1}{l+k} \right) = \frac{1}{k(l+k)}, \\
(V_s \cdot B \cdot \mathbf{v}^{(k)})_l &= \sum_{i=1}^{s} b_i c_i^{l-1} \cdot \frac{c_i^k}{k} = \frac{1}{k} \sum_{i=1}^{s} b_i c_i^{l+k-1} = \frac{1}{k(l+k)}.
\end{aligned}$$

Por tanto, $V_s \cdot B \cdot \mathbf{u}^{(k)} = V_s \cdot B \cdot \mathbf{v}^{(k)}$. Además, como V_s es invertible y los coeficientes $b_1, \ldots, b_s$ son no nulos, tenemos que $\mathbf{u}^{(k)} = \mathbf{v}^{(k)}$ y se cumple $C(r)$. Aplicando el Teorema 2.1, vemos que el método Runge-Kutta tendrá orden $p = s + r$. □

Teorema 2.5. *El polinomio de colocación $\boldsymbol{u}(x)$ da lugar a un método Runge-Kutta implícito de orden s, es decir, para todo $x_n \leq x \leq x_n + h$ tenemos*

$$||\boldsymbol{y}(x) - \boldsymbol{u}(x)|| \leq C \cdot h^{s+1}, \tag{2.13}$$

siendo $\boldsymbol{y}(x)$ la solución exacta de (1.1a) *con condición inicial $\boldsymbol{y}(x_n) = \boldsymbol{y}_n$. Además, para las derivadas de $\boldsymbol{u}(x)$ tenemos*

$$||\boldsymbol{y}^{(k)}(x) - \boldsymbol{u}^{(k)}(x)|| \leq C \cdot h^{s+1-k}, \quad k = 0, \ldots, s. \tag{2.14}$$

Demostración. La solución $\mathbf{y}(x)$ satisface las condiciones de colocación siempre, y particularmente en los puntos de la forma $x_n + c_i h$. De igual forma que en el Teorema 2.2, aplicamos la fórmula de interpolación de Lagrange a $\mathbf{y}'(x)$:

$$\mathbf{y}'(x_n + th) = \sum_{j=1}^{s} \mathbf{f}(x_n + c_j h, \mathbf{y}(x_n + c_j h))\, l_j(t) + h^s R(t, h),$$

donde $R(t, h)$ es una función continuamente diferenciable en las variables t y h, que se corresponde con el resto de la interpolación de Lagrange asociado a $\mathbf{y}'(x)$. Sustrayendo

$$\mathbf{u}(x_n + th) = \mathbf{y}_n + h \int_0^t \mathbf{u}'(x_n + \tau h) d\tau$$

a

$$\mathbf{y}(x_n + th) = \mathbf{y}_n + h \int_0^t \mathbf{y}'(x_n + \tau h) d\tau$$

obtenemos

$$\begin{aligned} \mathbf{y}(x_n + th) - \mathbf{u}(x_n + th) &= h \int_0^t \left(\mathbf{y}'(x_n + \tau h) - \mathbf{u}'(x_n + \tau h)\right) d\tau \\ &= h \sum_{j=1}^{s} \Delta \mathbf{f}_j \cdot \int_0^t l_j(\tau) d\tau + h^{s+1} \int_0^t R(\tau, h) d\tau, \end{aligned}$$

donde

$$\Delta \mathbf{f}_j = \mathbf{f}(x_n + c_j h, \mathbf{y}(x_n + c_j h)) - \mathbf{f}(x_n + c_j h, \mathbf{u}(x_n + c_j h)).$$

Tenemos que

$$\mathbf{y}'(x_n + th) - \mathbf{u}'(x_n + th) = \sum_{j=1}^{s} \Delta \mathbf{f}_j l_j(t) + h^s R(t, h),$$

de donde se obtiene que la derivada de orden k de $\mathbf{y}(x_n + th) - \mathbf{u}(x_n + th)$ respecto de t es

$$h^k \left(\mathbf{y}^{(k)}(x_n + th) - \mathbf{u}^{(k)}(x_n + th)\right) = h \sum_{j=1}^{s} \Delta \mathbf{f}_j \cdot l_j^{(k-1)}(t) + h^{s+1} \frac{\partial^{k-1} R}{\partial t^{k-1}}(t, h).$$

Por el Lema 2.1, sabemos que $\Delta \mathbf{f}_j = \mathcal{O}(h^{s+1})$, y por tanto

$$||\mathbf{y}(x) - \mathbf{u}(x)|| \leq C \cdot h^{s+1}.$$

Además, por la acotación de las derivadas de $R(t, h)$, podemos concluir que

$$||\mathbf{y}^{(k)}(x) - \mathbf{u}^{(k)}(x)|| \leq C \cdot h^{s+1-k}, \quad k = 0, \ldots, s.$$

□

2.3. Familias de métodos de Radau

En 1963, Butcher introdujo métodos Runge-Kutta implícitos basados en la cuadratura de Radau (ver [5]). Los denominó *métodos de Radau* de tipo I o de tipo II dependiendo del polinomio del que fuesen raíces los coeficientes $c_1, \ldots, c_s$:

- Las abscisas del método de Radau I de s etapas son las raíces del polinomio

$$\frac{d^{s-1}}{dx^{s-1}}\left(x^s\,(x-1)^{s-1}\right),$$

- Las abscisas del método de Radau II de s etapas son las raíces del polinomio

$$\frac{d^{s-1}}{dx^{s-1}}\left(x^{s-1}\,(x-1)^{s}\right).$$

Observamos que $c_1 = 0$ para los métodos de Radau I y $c_s = 1$ para los métodos de Radau II.

Si consideramos el polinomio trasladado de Legendre [1] definido en el intervalo $[0, 1]$,

$$P_s^*(x) = \frac{1}{s!}\frac{d^s}{dx^s}\left(x^s\,(x-1)^s\right), \tag{2.15}$$

obtenido tras aplicar un cambio de variable $y = 2x - 1$ al polinomio de Legendre, definido en el intervalo $[-1, 1]$,

$$P_s(y) = \frac{1}{2^s}\frac{1}{s!}\frac{d^s}{dy^s}\left((y+1)^s\,(y-1)^s\right),$$

podemos definir los métodos de Radau I y II como aquellos cuyos coeficientes $c_1, \ldots, c_s$ son los ceros de los polinomios $P_s^*(x) + P_{s-1}^*(x)$ y $P_s^*(x) - P_{s-1}^*(x)$, respectivamente. En efecto,

$$\begin{aligned} P_s^*(x) + P_{s-1}^*(x) &= \frac{1}{s!}\frac{d^s}{dx^s}\left(x^s\,(x-1)^s\right) + \frac{1}{(s-1)!}\frac{d^{s-1}}{dx^{s-1}}\left(x^{s-1}\,(x-1)^{s-1}\right) \\ &= \frac{1}{s!}\cdot s(2x-1)\cdot\frac{d^{s-1}}{dx^{s-1}}\left(x^{s-1}\,(x-1)^{s-1}\right) + \frac{1}{(s-1)!}\frac{d^{s-1}}{dx^{s-1}}\left(x^{s-1}\,(x-1)^{s-1}\right) \\ &= 2xP_{s-1}^*(x), \end{aligned}$$

es decir, $P_s^*(x) + P_{s-1}^*(x)$ tendrá como raíces a los $2s-2$ ceros de $P_{s-1}^*(x)$ y a $c_1 = 0$. Se razona de forma análoga para la expresión $P_s^*(x) - P_{s-1}^*(x)$.

Teorema 2.6. *Los métodos de Radau I y II tienen orden* $2s - 1$.

Demostración.

- Radau I: Por ser un método de colocación basado en la cuadratura de Radau I, se cumplen $B(2s-1)$ y $C(s)$. Por el Lema 2.2, se cumple $D(s-1)$, y aplicando el Teorema 2.1, el orden del método será $2s - 1$.
- Radau II: Análogamente, para la cuadratura de Radau II, se cumplen $B(2s-1)$ y $D(s)$, entonces también $C(s-1)$, y el método tendrá orden $2s - 1$.

□

0	0	0
2/3	1/3	1/3
	1/4	3/4

1/3	1/3	0
1	1	0
	3/4	1/4

Tabla 2.2: Métodos de Radau I y II, respectivamente, con $s = 2, p = 3$.

[1] Las demostraciones de las propiedades sobre los polinomios trasladados de Legendre se incluyen en el Apéndice B.

0	0	0	0	$\frac{4-\sqrt{6}}{10}$	$\frac{24-\sqrt{6}}{120}$	$\frac{24-11\sqrt{6}}{120}$	0
$\frac{6-\sqrt{6}}{10}$	$\frac{9+\sqrt{6}}{75}$	$\frac{24+\sqrt{6}}{120}$	$\frac{168-73\sqrt{6}}{600}$	$\frac{4+\sqrt{6}}{10}$	$\frac{24+11\sqrt{6}}{120}$	$\frac{24+\sqrt{6}}{120}$	0
$\frac{6+\sqrt{6}}{10}$	$\frac{9-\sqrt{6}}{75}$	$\frac{168+73\sqrt{6}}{600}$	$\frac{24-\sqrt{6}}{120}$	1	$\frac{6-\sqrt{6}}{12}$	$\frac{6+\sqrt{6}}{12}$	0
	$\frac{1}{9}$	$\frac{16+\sqrt{6}}{36}$	$\frac{16-\sqrt{6}}{36}$		$\frac{16-\sqrt{6}}{36}$	$\frac{16+\sqrt{6}}{36}$	$\frac{1}{9}$

Tabla 2.3: Métodos de Radau I y II, respectivamente, con $s = 3, p = 5$.

Las Tablas 2.2 y 2.3 muestran los tableros de Butcher de los métodos de Radau I y II de dos y tres etapas, respectivamente. Consideramos ahora un método de un paso de la forma

$$y_{n+1} = y_n + h\Phi(x_n, y_n, h),$$

donde $\Phi(x, y, h)$ es su función incremento. Este puede reescribirse como

$$y_h(x + h) = y_h(x) + h\Phi(x, y_h(x), h),$$

y sustituyendo h por $-h$ se obtiene

$$y_{-h}(x - h) = y_{-h}(x) - h\Phi(x, y_{-h}(x), -h).$$

Ahora, cambiando x por $x + h$, la igualdad anterior se transforma en

$$y_{-h}(x) = y_{-h}(x + h) - h\Phi(x + h, y_{-h}(x + h), -h). \tag{2.16}$$

Esta sería una ecuación implícita para $y_{-h}(x+h)$, que, por el teorema de la función implícita, tendrá solución única para h suficientemente pequeño, y esta se puede escribir de la siguiente manera

$$y_{-h}(x + h) = y_{-h}(x) + h\Phi^*(x, y_{-h}(x), -h), \tag{2.17}$$

para cierta función Φ^*. Si denotamos $A = y_{-h}(x + h)$ y $B = y_{-h}(x)$, podemos definir el concepto de método adjunto.

Definición 2.2. Sea $\Phi(x, y, h)$ la función incremento de un método. Entonces definimos la función incremento de su *método adjunto* $\Phi^*(x, y, h)$ mediante el siguiente par de ecuaciones

$$B = A - h\Phi(x + h, A, -h), \tag{2.18a}$$
$$A = B + h\Phi^*(x, B, h). \tag{2.18b}$$

Las ecuaciones anteriores quieren decir que si A es el resultado de avanzar un paso de longitud h con el método adjunto partiendo de B en x, entonces B es el resultado de avanzar un paso de longitud $-h$ con el método original cuando se parte de A en $x + h$.

Teorema 2.7. *Sea Φ el método Runge-Kutta (1.2) con coeficientes a_{ij}, b_i, c_i, para $i, j = 1, \ldots, s$. Entonces el método adjunto Φ^* es equivalente a un método Runge-Kutta de s etapas con coeficientes*

$$c_i^* = 1 - c_{s+1-i},$$
$$a_{ij}^* = b_{s+1-j} - a_{s+1-i,s+1-j},$$
$$b_i^* = b_{s+1-i}.$$

Demostración. Consideramos el método Runge-Kutta de s etapas definido en (1.2). Sustituyendo h por $-h$,

$$\mathbf{k}_i = \mathbf{f}(x_n - c_i h, \mathbf{y}_n - h\sum_{j=1}^{s} a_{ij}\mathbf{k}_j), \quad 1 \le i \le s,$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n - h\sum_{i=1}^{s} b_i \mathbf{k}_i.$$

Ahora, cambiando x_n por $x_n + h$, la ecuación de los $\mathbf{k}_i$ sería $\mathbf{k}_i = \mathbf{f}(x_n + (1-c_i)h, \mathbf{y}_n - h\sum_{j=1}^{s} a_{ij}\mathbf{k}_j)$, $1 \le i \le s$. Además, $\mathbf{y}_n = \mathbf{y}_{n+1} + h\sum_{i=1}^{s} b_i\mathbf{k}_i$, y por tanto

$$\mathbf{k}_i = \mathbf{f}(x_n + (1-c_i)h, \mathbf{y}_{n+1} + h\sum_{j=1}^{s}(b_j - a_{ij})\mathbf{k}_j), \quad 1 \le i \le s.$$

Si se intercambian $\mathbf{y}_n$ e $\mathbf{y}_{n+1}$, obtenemos que las ecuaciones del método adjunto serían

$$\mathbf{k}_i = \mathbf{f}(x_n + (1-c_i)h, \mathbf{y}_n + h\sum_{j=1}^{s}(b_j - a_{ij})\mathbf{k}_j), \quad 1 \le i \le s,$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\sum_{i=1}^{s} b_i \mathbf{k}_i.$$

Para conservar el orden natural de los coeficientes $c_1, \ldots, c_s$, permutamos los valores $\mathbf{k}_i$ y cambiamos todos los índices i por $s+1-i$:

$$\mathbf{k}_{s+1-i} = \mathbf{f}(x_n + (1-c_{s+1-i})h, \mathbf{y}_n + h\sum_{j=1}^{s}(b_{s+1-j} - a_{s+1-j,s+1-j})\mathbf{k}_{s+1-j}), \quad 1 \le i \le s,$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\sum_{i=1}^{s} b_{s+1-i}\mathbf{k}_{s+1-i}.$$

Tomando como coeficientes del método a los c_i^*, a_{ij}^* y b_i^* definidos en el enunciado del teorema, obtendríamos el método adjunto, cuyo tablero de Butcher es

$$\begin{array}{c|cccc} 1-c_s & b_s - a_{ss} & b_{s-1} - a_{s,s-1} & \cdots & b_1 - a_{s1} \\ 1-c_{s-1} & b_s - a_{s-1,s} & b_{s-1} - a_{s-1,s-1} & \cdots & b_1 - a_{s-1,1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1-c_1 & b_s - a_{1s} & b_{s-1} - a_{1,s-1} & \cdots & b_1 - a_{11} \\ \hline & b_s & b_{s-1} & \cdots & b_1 \end{array}.$$

□

Teorema 2.8. *El adjunto del adjunto de un método, es el método original, es decir,* $\Phi^{**} = \Phi$.

Demostración. Sustituyendo en (2.18) h por $-h$, A por B, B por A y x por $x+h$, vemos que se cumple lo que se quería demostrar. □

Si un método satisface las hipótesis simplificadoras estudiadas en la Sección 2.1, vamos a ver que su adjunto cumple unas condiciones simplificadoras similares. Denotamos pues por $B^*(\eta)$, $C^*(\eta)$ y $D^*(\eta)$ a las condiciones simplificadoras análogas a $B(\eta)$, $C(\eta)$ y $D(\eta)$ cuando se aplican a los coeficientes del método adjunto.

Teorema 2.9. *Sea* η *un entero positivo. Se cumplen las implicaciones siguientes:*

(1) $B(\eta) \Rightarrow B^*(\eta)$,

(2) $B(\eta) \wedge C(\eta) \Rightarrow C^*(\eta)$,

(3) $B(\eta) \wedge D(\eta) \Rightarrow D^*(\eta)$.

Demostración. Sea p un polinomio cualquiera de grado menor que η. Usando la base polinomial estándar, vemos que $B(\eta)$, $C(\eta)$ y $D(\eta)$ equivalen a

(a) $\sum_{j=1}^{s} b_j p(c_j) = \int_0^1 p(x)dx$,

(b) $\sum_{j=1}^{s} a_{ij} p(c_j) = \int_0^{c_i} p(x)dx, \quad i = 1, \ldots, s$,

(c) $\sum_{i=1}^{s} b_i p(c_i) a_{ij} = b_j \int_{c_j}^1 p(x)dx, \quad j = 1, \ldots, s$.

Como $B(\eta)$ se cumple para todo valor $\eta \geq 1$, en particular $\sum_{i=1}^{s} b_i = 1$.

(1) Usando (a), se cumple $B^*(\eta)$, puesto que

$$\sum_{j=1}^{s} b_j p(1 - c_j) = \int_0^1 p(1-x)dx = \int_0^1 p(x)dx.$$

(2) Usando (b), se cumple $C^*(\eta)$, ya que

$$\begin{aligned}\sum_{j=1}^{s} (b_j - a_{ij}) p(1 - c_j) &= \int_0^{1-c_i} p(x)dx = \int_0^1 p(x)dx - \int_0^{c_i} p(1-x)dx \\ &= \int_0^1 p(x)dx - \int_{1-c_i}^1 p(x)dx = \int_0^{1-c_i} p(x)dx.\end{aligned}$$

(3) Usando (c), se cumple $D^*(\eta)$, pues

$$\begin{aligned}\sum_{i=1}^{s} b_i p(1 - c_i)(b_j - a_{ij}) &= b_j \int_{1-c_j}^1 p(x)dx = b_j \left(\int_0^1 p(x)dx - \int_{c_j}^1 p(1-x)dx \right) \\ &= b_j \left(\int_0^1 p(x)dx - \int_0^{1-c_j} p(x)dx \right) = b_j \int_{1-c_j}^1 p(x)dx.\end{aligned}$$

□

Por tanto, considerando los adjuntos de los métodos de Radau II y I, se definen los métodos de Radau IA y IIA, respectivamente. En la Tabla 2.4 se recogen los detalles de la elección de coeficientes b_i, c_i y a_{ij} para estas cuatro familias de métodos.

Nombre	Elección de $\mathbf{b}^T$ y $\mathbf{c}$	Elección de $\mathcal{A}$
Radau I	Cuadratura de Radau I	$C(s)$
Radau IA	Cuadratura de Radau I	Adjunto de Radau II
Radau II	Cuadratura de Radau II	$D(s)$
Radau IIA	Cuadratura de Radau II	Adjunto de Radau I

Tabla 2.4: Elección de coeficientes para las distintas familias de métodos de Radau.

Teorema 2.10. *Los métodos de Radau IA y Radau IIA tienen el mismo orden que los métodos de Radau I y Radau II.*

Demostración. La prueba es análoga a la del Teorema 2.9, usando los resultados del Teorema 2.8. □

De la misma forma que en las Tablas 2.2 y 2.3 se incluyeron los tableros de Butcher para los métodos de Radau I y II de órdenes 3 y 5, se muestran en las Tablas 2.5 y 2.6 los tableros de Butcher para los métodos de Radau IA y IIA de los mismos órdenes.

0	1/4	−1/4
2/3	1/4	5/12
	1/4	3/4

1/3	5/12	−1/12
1	3/4	1/4
	3/4	1/4

Tabla 2.5: Métodos de Radau IA y IIA, respectivamente, con $s = 2, p = 3$.

0	$\frac{1}{9}$	$\frac{-1-\sqrt{6}}{18}$	$\frac{-1+\sqrt{6}}{18}$
$\frac{6-\sqrt{6}}{10}$	$\frac{1}{9}$	$\frac{88+7\sqrt{6}}{360}$	$\frac{88-43\sqrt{6}}{360}$
$\frac{6+\sqrt{6}}{10}$	$\frac{1}{9}$	$\frac{88+43\sqrt{6}}{360}$	$\frac{88-7\sqrt{6}}{360}$
	$\frac{1}{9}$	$\frac{16+\sqrt{6}}{36}$	$\frac{16-\sqrt{6}}{36}$

$\frac{4-\sqrt{6}}{10}$	$\frac{88-7\sqrt{6}}{360}$	$\frac{296-169\sqrt{6}}{1800}$	$\frac{-2+3\sqrt{6}}{225}$
$\frac{4+\sqrt{6}}{10}$	$\frac{296+169\sqrt{6}}{1800}$	$\frac{88+7\sqrt{6}}{360}$	$\frac{-2-3\sqrt{6}}{225}$
1	$\frac{16-\sqrt{6}}{36}$	$\frac{16+\sqrt{6}}{36}$	$\frac{1}{9}$
	$\frac{16-\sqrt{6}}{36}$	$\frac{16+\sqrt{6}}{36}$	$\frac{1}{9}$

Tabla 2.6: Métodos de Radau IA y IIA respectivamente, con $s = 3, p = 5$.

Capítulo 3

Estabilidad de los métodos de Radau

En este capítulo se tratarán las distintas propiedades de estabilidad de los métodos Runge-Kutta implícitos, y en particular las de los métodos de Radau IIA.

3.1. Función de estabilidad

Para el estudio de la estabilidad de los métodos Runge-Kutta, consideramos ahora la ecuación test de Dahlquist

$$y'(x) = \lambda \cdot y(x), \quad Re(\lambda) << 0, \tag{3.1a}$$
$$y(x_0) = y_0. \tag{3.1b}$$

Su solución exacta viene dada por $y(t) = y_0 e^{\lambda t}$, y es claro que $\lim_{t\to\infty} y(t) = 0$. En el estudio de la estabilidad lineal de los métodos Runge Kutta, lo que se busca es comprobar si esta propiedad también se mantiene en la solución numérica, es decir, si $\lim_{n\to\infty} y_n = 0$.

Proposición 3.1. *Si avanzamos un paso de longitud h con un método Runge-Kutta de s etapas* (1.2) *en la integración de* (3.1)*, se cumple que*

$$y_{n+1} = R(h\lambda) y_n,$$

donde $R(z)$, $z = h\lambda \in \mathbb{C}^-$, puede hallarse mediante cualquiera de las siguientes expresiones:

(1) $R(z) = 1 + z\boldsymbol{b}^T (I - z\mathcal{A})^{-1} \boldsymbol{e}$,

(2) $R(z) = \dfrac{det(I - z\mathcal{A} + z\boldsymbol{e}\boldsymbol{b}^T)}{det(I - z\mathcal{A})}$,

donde $\boldsymbol{e} = (1, \ldots, 1) \in \mathbb{R}^s$.

Demostración.

(1) Escribimos en forma matricial el sistema definido por las etapas intermedias del método (1.3a) aplicado a (3.1), obteniendo

$$\begin{pmatrix} Y_1 \\ \vdots \\ Y_s \end{pmatrix} = y_n \cdot \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} + h\lambda \cdot \begin{pmatrix} \sum_{j=1}^s a_{1j} Y_j \\ \vdots \\ \sum_{j=1}^s a_{sj} Y_j \end{pmatrix}. \tag{3.2}$$

Tomando $\mathcal{Y} = (Y_1, \ldots, Y_s)^T$, se puede reescribir dicho sistema como $(I - h\lambda\mathcal{A})\,\mathcal{Y} = \mathbf{e} \cdot y_n$, donde $I - h\lambda\mathcal{A}$ es no singular para h suficientemente pequeño, pues $||h\lambda\mathcal{A}|| < 1$. De aquí se deduce

$\mathcal{Y} = (I - h\lambda\mathcal{A})^{-1}\,\mathbf{e}\cdot y_n$. Sustituyendo este valor en (1.3b), obtenemos

$$\begin{aligned} y_{n+1} &= y_n + h\lambda\sum_{i=1}^{s} b_i Y_i = y_n + h\lambda\mathbf{b}^T\mathcal{Y} = y_n + h\lambda\mathbf{b}^T\,(I - h\lambda\mathcal{A})^{-1}\,\mathbf{e}\cdot y_n \\ &= \left(1 + h\lambda\mathbf{b}^T\,(I - h\lambda\mathcal{A})^{-1}\,\mathbf{e}\right) y_n, \end{aligned}$$

es decir, $y_{n+1} = R(h\lambda)\,y_n$, donde $R(z) = 1 + z\mathbf{b}^T\,(I - z\mathcal{A})^{-1}\,\mathbf{e}$.

(2) Aplicando (1.3) a la ecuación (3.1), obtenemos (3.2) junto con

$$y_{n+1} = y_n + h\lambda\sum_{i=1}^{s} b_i Y_i.$$

Razonando de forma similar al apartado anterior y tomando $z = h\lambda$, se puede escribir como un sistema lineal de tamaño $(s+1)\times(s+1)$

$$\left(\begin{array}{c|c} I - z\mathcal{A} & \mathbf{0} \\ \hline -z\mathbf{b}^T & 1 \end{array}\right)\cdot\begin{pmatrix} \mathcal{Y} \\ y_{n+1} \end{pmatrix} = y_n\begin{pmatrix} \mathbf{e} \\ 1 \end{pmatrix}.$$

Aplicamos la regla de Cramer para hallar la última componente de la solución, y_{n+1}

$$\begin{aligned} y_{n+1} &= \frac{det\left(\begin{array}{c|c} I - z\mathcal{A} & \mathbf{e} \\ \hline -z\mathbf{b}^T & 1 \end{array}\right)}{det\left(\begin{array}{c|c} I - z\mathcal{A} & \mathbf{0} \\ \hline -z\mathbf{b}^T & 1 \end{array}\right)}\cdot y_n = \frac{det\left(\begin{array}{c|c} I - z\mathcal{A} + z\mathbf{e}\mathbf{b}^T & \mathbf{0} \\ \hline -z\mathbf{b}^T & 1 \end{array}\right)}{det\,(I - z\mathcal{A})}\cdot y_n \\ &= \frac{det\left(I - z\mathcal{A} + z\mathbf{e}\mathbf{b}^T\right)}{det\,(I - z\mathcal{A})}\cdot y_n, \end{aligned}$$

es decir, $y_{n+1} = R(h\lambda)\,y_n$, donde $R(z) = \dfrac{det\left(I - z\mathcal{A} + z\mathbf{e}\mathbf{b}^T\right)}{det\,(I - z\mathcal{A})}$.

□

Si el método Runge-Kutta es explícito, la matriz $\mathcal{A}$ de su tablero de Butcher es estrictamente triangular inferior, $R(z)$ es un polinomio de grado $\leq s$ en z y no se satisfaría la condición $|R(z)| < 1$ para todo z, con $Re(z) < 0$. De hecho, los métodos explícitos cumplen que $|R(z)| \geq 1$, $z \to \infty$, y por tanto tienen regiones de estabilidad acotadas.

Para los métodos implícitos, podemos observar que $R(z)$ es una función racional con numerador y denominador polinomios de grados $\leq s$. Esto se denota por

$$R(z) = \frac{P(z)}{Q(z)},$$

donde $deg(P) = k$ y $deg(Q) = j$, con $k, j \leq s$.

Ejemplo 3.1. Consideramos el método de Radau I de 2 etapas y orden 3, y su adjunto, el método de Radau IIA. Los tableros de Butcher de ambos métodos están definidos en la Tabla 2.3. Tomando la ecuación de Dahlquist, si se avanza un paso de longitud h con cualquiera de los dos métodos en la integración de (3.1),

$$y_{n+1} = R(h\lambda)y_n,$$

donde la función $R(z)$ viene dada por

- Radau I: $R(z) = \dfrac{1 + \frac{2}{3}z + \frac{1}{6}z^2}{1 - \frac{1}{3}z}$,
- Radau IIA: $R(z) = \dfrac{1 + \frac{1}{3}z}{1 - \frac{2}{3}z + \frac{1}{6}z^2}$.

Teniendo en cuenta la Proposición 3.1, observamos que si $|R(h\lambda)| < 1$ para $Re(\lambda) < 0$, como

$$y_{n+1} = R(h\lambda)y_n \leq R(h\lambda)^2 y_{n-1} \leq \cdots \leq R(h\lambda)^{n+1} y_0,$$

entonces $\lim_{n\to\infty} y_n = 0$, que es la propiedad que queremos obtener.

Notamos entonces que, si $|R(z)| < 1$ para todo $z \in \mathbb{C}$ con $Re(z) < 0$, para cualquier valor de h se tendría la propiedad deseada, con independencia del valor concreto de λ, que aparece en (3.1a). En el caso del método de Radau IIA, la condición $|R(z)| < 1$ se cumple para todo valor de $z \in \mathbb{C}^-$, y por tanto $\lim_{n\to\infty} y_n = 0$. Sin embargo, esto no ocurre con el método de Radau I.

Definición 3.1.

- La función $R(z)$ definida en la Proposición 3.1 se denomina *función de estabilidad* del método.
- Se denomina *región* (o dominio) *de estabilidad del método Runge-Kutta* al conjunto

$$S = \{z \in \mathbb{C} \ : \ |R(z)| \leq 1\}. \tag{3.3}$$

- Un método se dice que es *A-estable* si su región de estabilidad satisface

$$S \supset \mathbb{C}^- = \{z \in \mathbb{C} \ : \ Re(z) \leq 0\}. \tag{3.4}$$

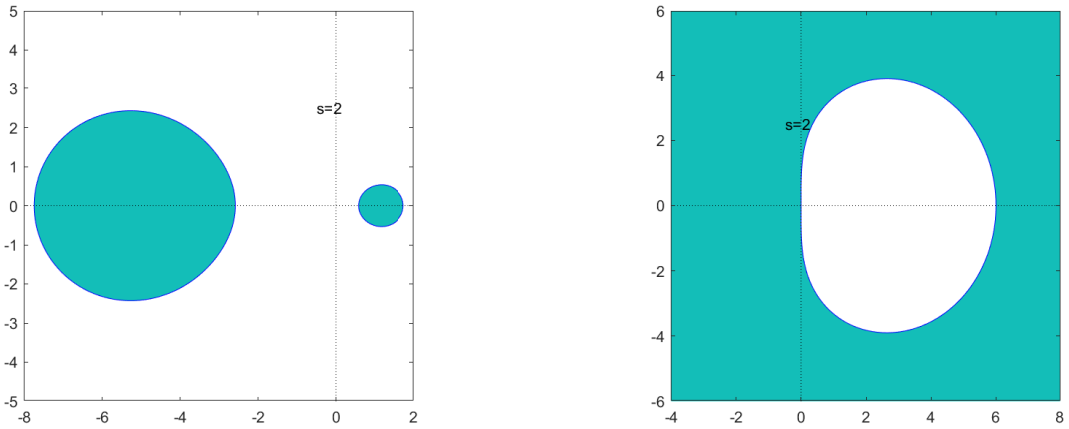


Figura 3.1: Regiones de estabilidad de los métodos de Radau I (izquierda) y Radau IIA de orden 3 (derecha), respectivamente.

En la Tabla 3.1 se recogen las funciones de estabilidad de los métodos de Radau de 2 y 3 etapas y en la Figura 3.1 están representadas las regiones de estabilidad de los métodos Radau I y Radau IIA.

	Método	$R(z)$
(a)	Radau I y Radau II, $s = 2$, $p = 3$	$\dfrac{1 + 2z/3 + z^2/6}{1 - z/3}$
(b)	Radau IA y Radau IIA, $s = 2$, $p = 3$	$\dfrac{1 + z/3}{z - 2z/3 + z^2/6}$
(c)	Radau I y Radau II, $s = 3$, $p = 5$	$\dfrac{1 + 3z/5 + 3z^2/20 + z^3/60}{1 - 2z/5 + z^2/20}$
(d)	Radau IA y Radau IIA, $s = 3$, $p = 5$	$\dfrac{1 + 2z/5 + z^2/20}{1 - 3z/5 + 3z^2/20 - z^3/60}$

Tabla 3.1: Funciones de estabilidad de los métodos de Radau de 2 y 3 etapas.

Proposición 3.2. *Si un método Runge-Kutta de s etapas* (1.2) *tiene orden p, entonces*

$$R(z) = e^z + \mathcal{O}\left(z^{p+1}\right), \quad z \to 0. \tag{3.5}$$

Demostración. Aplicando el método (1.2) a (3.1) con condición inicial $y(x_n) = y_n$, obtenemos que $y_{n+1} = R(h\lambda)\, y_n$ y que $y(x_{n+1}) = y_n\, e^{\lambda x_{n+1}}$. De aquí se deduce que

$$y_{n+1} - y(x_{n+1}) = \left(R(h\lambda) - e^{h\lambda}\right) y_n.$$

Si el método tiene orden p, $y_{n+1} - y(x_{n+1}) = \mathcal{O}(h^{p+1})$ cuando $h \to 0$, y esto equivale a decir que $|R(z) - e^z| = \mathcal{O}\left(z^{p+1}\right)$, y por tanto se cumple (3.5). □

Teorema 3.1. *Un método Runge-Kutta con función de estabilidad $R(z) = \frac{P(z)}{Q(z)}$ es A-estable si, y sólo si, se cumplen las siguientes condiciones:*

- ***(a)*** *Todos los polos de la función R (es decir, todos los ceros del polinomio Q) están en $\mathbb{C}^+$.*
- ***(b)*** *El polinomio $E(y) = |Q(iy)|^2 - |P(iy)|^2 = Q(iy)Q(-iy) - P(iy)P(-iy)$ cumple $E(y) \geq 0$ para todo $y \in \mathbb{R}$.*

Ejemplo 3.2. Consideramos el método de Radau IIA, con $s = 3$, $p = 5$. Su función de estabilidad es

$$R(z) = \frac{P(z)}{Q(z)} = \frac{1 + \frac{2}{5}z + \frac{1}{20}z^2}{1 - \frac{3}{5}z + \frac{3}{20}z^2 - \frac{1}{60}z^3},$$

de donde se obtiene que

$$P(iy) = \left(1 - \frac{1}{20}y^2\right) + i\frac{2}{5}y, \qquad Q(iy) = \left(1 - \frac{3}{20}y^2\right) + i\left(-\frac{3}{5}y + \frac{1}{60}y^3\right),$$

y, por tanto,

$$\begin{aligned} |P(iy)|^2 &= 1 - \frac{1}{10}y^2 + \frac{1}{400}y^4 + \frac{4}{25}y^2 = 1 + \frac{3}{50}y^2 + \frac{1}{400}y^4, \\ |Q(iy)|^2 &= 1 - \frac{3}{10}y^2 + \frac{9}{400}y^4 + \frac{9}{25}y^2 - \frac{1}{50}y^4 + \frac{1}{3600}y^6 = 1 + \frac{3}{50}y^2 + \frac{1}{400}y^4 + \frac{1}{3600}y^6. \end{aligned}$$

Es claro que en este caso $E(y) = y^6/3600 \geq 0$ para todo $y \in \mathbb{R}$.

El polinomio $Q(z)$ tiene una raíz real, $z_1 = 3 - \sqrt[3]{3} + \sqrt[3]{9}$, y un par de raíces complejas,

$$z_2 = 3 + \frac{\sqrt[3]{3}}{2}\left(1 - i\sqrt{3}\right) - \frac{\sqrt[3]{9}}{2}\left(1 + i\sqrt{3}\right), \qquad z_3 = 3 - \frac{\sqrt[3]{9}}{2}\left(1 - i\sqrt{3}\right) + \frac{\sqrt[3]{3}}{2}\left(1 + i\sqrt{3}\right),$$

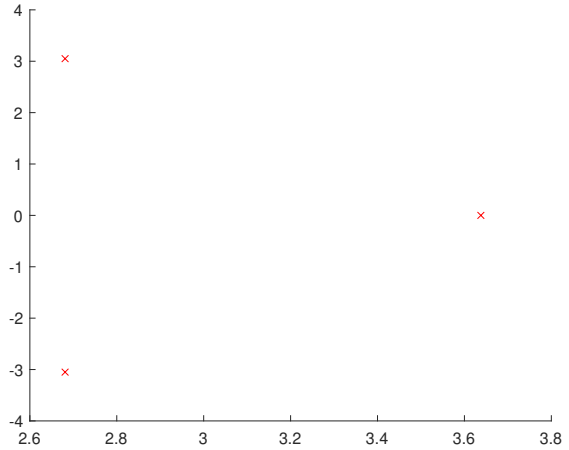


Figura 3.2: Raíces del polinomio $Q(z)$ del Ejemplo 3.2.

que, claramente, están en $\mathbb{C}^+$. Como se cumplen las dos condiciones del Teorema 3.1, entonces el método de Radau IIA de orden 5 es A-estable.

Si en lugar de la ecuación test de Dahlquist tuviésemos un problema de valores iniciales $\mathbf{y}' = M\mathbf{y}$, $\mathbf{y}(x_0) = \mathbf{y}_0$, donde la matriz $M \in \mathbb{R}^{D\times D}$ fuese diagonalizable con todos los autovalores de parte real negativa, la A-estabilidad de un método Runge-Kutta garantizaría que para cualquier longitud de paso la solución numérica $\mathbf{y}_n$ generada con él tendería a $\mathbf{0}$ cuando n tiende a $+\infty$, imitando así el comportamiento de la solución exacta.

En efecto, como M es diagonalizable, existen matrices $P, \Delta \in \mathbb{R}^{D\times D}$ tales que $M = P\Delta P^{-1}$, donde P es invertible y Δ diagonal. Haciendo el cambio de variable $\mathbf{y} = P\mathbf{z}$ y derivando, obtenemos que $\mathbf{y}' = P\mathbf{z}'$ y, como P es invertible, $\mathbf{z}' = P^{-1}\mathbf{y}' = P^{-1}M\mathbf{y} = P^{-1}MP\mathbf{z} = \Delta\mathbf{z}$, y el problema original se convierte en D problemas escalares, a los que se pueden aplicar los teoremas anteriores.

3.2. L-estabilidad

En algunos métodos, la región de estabilidad coincide exactamente con el semiplano negativo $\mathbb{C}^-$. Esta propiedad no es tan deseable como podría parecer, ya que para una función racional $R(z)$ se tiene que

$$\lim_{z\to-\infty} R(z) = \lim_{z\to\infty} R(z) = \lim_{y\to\infty} R(iy).$$

El último límite debe ser igual a 1, pues $|R(iy)| = 1$ para todo $y \in \mathbb{R}$. Esto quiere decir que, para z cercano al eje real, con $Re(z) << 0$, el valor $|R(z)|$ es menor que 1 pero muy próximo a 1. En consecuencia, las componentes rígidas de la solución numérica se amortiguan muy lentamente. Parece pues deseable que $|R(z)|$ sea mucho menor que 1 cuando $z \to \infty$.

Definición 3.2. Un método es L-estable si es A-estable y además

$$\lim_{z\to\infty} R(z) = 0. \tag{3.6}$$

El método de Radau IIA de orden 5 (Tabla 3.1, caso d) es L-estable, pues el grado del denominador es estrictamente mayor que el grado del numerador.

Proposición 3.3. *Si un método Runge-Kutta implícito con matriz $\mathcal{A}$ no singular satisface una de las condiciones siguientes*

$$a_{sj} = b_j, \quad j = 1, \dots, s, \tag{3.7a}$$
$$a_{i1} = b_1, \quad i = 1, \dots, s, \tag{3.7b}$$

entonces $R(\infty) = 0$. Los métodos A-estables que cumplen una de las dos condiciones de (3.7) *son L-estables.*

Demostración. Consideramos la función de estabilidad $R(z) = 1 + z\mathbf{b}^T(I - z\mathcal{A})^{-1}\mathbf{e}$, definida en la Proposición 3.1. Tomando límite cuando $z \to \infty$, tenemos que $R(\infty) = 1 - \mathbf{b}^T\mathcal{A}^{-1}\mathbf{e}$.

Si tenemos que $a_{sj} = b_j$, $j = 1, \dots, s$, la matriz $\mathcal{A}$ cumple que $\mathcal{A}^T\mathbf{e}_s = \mathbf{b}$, donde $\mathbf{e}_s = (0, \dots, 0, 1)^T$. Por tanto,

$$R(\infty) = 1 - \mathbf{b}^T\mathcal{A}^{-1}\mathbf{e} = 1 - \mathbf{e}_s^T\mathcal{A}\mathcal{A}^{-1}\mathbf{e} = 1 - \mathbf{e}_s^T\mathbf{e} = 0.$$

Siguiendo un razonamiento similar, si tenemos que $a_{i1} = b_1$, $i = 1, \dots, s$, la matriz $\mathcal{A}$ cumple que $\mathcal{A}\mathbf{e}_1 = b_1\mathbf{e}$, siendo $\mathbf{e}_1 = (1, 0, \dots, 0)^T$. Por tanto,

$$R(\infty) = 1 - \mathbf{b}^T\mathcal{A}^{-1}\mathbf{e} = 1 - b_1^{-1}\mathbf{b}^T\mathcal{A}^{-1}\mathcal{A}\mathbf{e}_1 = 1 - b_1^{-1}\mathbf{b}\,\mathbf{e}_1 = 0.$$

Si $b_1 = 0$, entonces se tendría $a_{i1} = 0$ para $i = 1, \dots, s$, y la primera etapa del método Runge-Kutta no intervendría, en la solución numérica, por lo que se podría eliminar la primera etapa del método y reescribirlo como un método de $s - 1$ etapas.

Teniendo en cuenta la Definición 3.2, los métodos A-estables que cumplan (3.7a) o (3.7b) serán L-estables. □

Los métodos que satisfacen (3.7a) se denominan *rígidamente estables.*

Proposición 3.4. *Se considera un método Runge-Kutta implícito de s etapas con orden $p \geq s$, matriz $\mathcal{A}$ no singular, nodos $c_1, \dots, c_s$ distintos y abscisas $b_1, \dots, b_s$ no nulas. Entonces,*

(a) *Si se cumple $C(s)$ y $c_s = 1$, entonces se tiene* (3.7a).

(b) *Si se cumple $D(s)$ y $c_1 = 0$, entonces se tiene* (3.7b).

Demostración. Si el método tiene orden p, con $p \geq s$, se cumple $B(p)$, es decir, $\sum_{i=1}^{s} b_i c_i^{k-1} = 1/k$, $k = 1, \dots, p$, y en particular se cumple $B(s)$.

(a) Si se cumple $C(s)$, entonces $\sum_{j=1}^{s} a_{ij} c_j^{k-1} = c_i^k/k$, $1 \leq i \leq s$, $k = 1, \dots, s$. Como $c_s = 1$, tomando $i = s$ tenemos que, para $k = 1, \dots, s$, se cumple que

$$\sum_{j=1}^{s} a_{sj} c_j^{k-1} = \frac{c_s^k}{k} = \frac{1}{k} = \sum_{j=1}^{s} b_j c_j^{k-1} \Rightarrow \sum_{j=1}^{s} c_j^{k-1}(a_{sj} - b_j) = 0 \Rightarrow a_{sj} = b_j, \quad j = 1, \dots, s,$$

pues se obtiene un sistema de tamaño $s \times s$ con la matriz de Vandermonde V_s (2.10), invertible, y cuyas soluciones son los coeficientes $a_{sj} - b_j$, $j = 1, \dots, s$.

(b) Si se cumple $D(s)$, entonces $\sum_{i=1}^{s} b_i c_i^{k-1} a_{ij} = b_j(1 - c_j^k)/k$, $1 \leq j \leq s$, $k = 1, \dots, s$. Como $c_1 = 0$, tomando $j = 1$ tenemos que, para $k = 1, \dots, s$, se cumple que

$$\sum_{i=1}^{s} b_i c_i^{k-1} a_{i1} = \frac{b_1(1 - c_1^k)}{k} = \frac{b_1}{k} = b_1 \sum_{i=1}^{s} b_i c_i^{k-1} = \sum_{i=1}^{s} b_i c_i^{k-1} b_1$$
$$\Rightarrow \sum_{i=1}^{s} b_i c_i^{k-1}(a_{i1} - b_1) = 0 \Rightarrow a_{i1} = b_1, \quad i = 1, \dots, s,$$

pues se obtiene un sistema de tamaño $s \times s$ cuya matriz se corresponde con la matriz de Vandermonde V_s (2.10) multiplicada por la derecha por una matriz diagonal cuyos elementos son los coeficientes b_i, siendo esta matriz invertible, y dicho sistema tiene como solución a los coeficientes $a_{i1} - b_1$, $i = 1, \dots, s$.

□

3.3. Aproximantes de Padé

Teorema 3.2. *La función de estabilidad de un método de colocación basado en las abscisas $c_1, \dots, c_s$ viene dada por la función racional $R(z) = P(z)/Q(z)$, donde $P(z)$ y $Q(z)$ son respectivamente los polinomios*

$$P(z) = M^{(s)}(1) + M^{(s-1)}(1)z + \cdots + M'(1)z^{s-1} + M(1)z^s \tag{3.8a}$$
$$Q(z) = M^{(s)}(0) + M^{(s-1)}(0)z + \cdots + M'(0)z^{s-1} + M(0)z^s \tag{3.8b}$$

con $M(z) = \dfrac{1}{s!} \displaystyle\prod_{i=1}^{s} (x - c_i)$.

Demostración. Supongamos $x_0 = 0$, $y_0 = 1$ y $z = h\lambda$. Sea $u(x)$ el polinomio de colocación de grado s definido en el intervalo $[x_n, x_n + h]$ mediante (2.5). Como $u'(x) - zu(x)$ es un polinomio de grado s que se anula en los nodos de colocación $c_1, \ldots, c_s$, entonces existe una constante K_0 tal que

$$u'(x) - zu(x) = K_0 M(x).$$

Si denotamos por D al operador de diferenciación, entonces

$$u'(x) - zu(x) = (D - z)u(x) = -z\left(1 - \frac{D}{z}\right)u(x),$$

y por tanto

$$u(x) = -\frac{K_0}{z}\left(1 - \frac{D}{z}\right)^{-1} M(x).$$

Si hacemos el desarrollo de $(1 - D/z)^{-1}$ como serie geométrica, obtenemos

$$u(x) = -\frac{K_0}{z}\left(\sum_{j=0}^{\infty} \frac{D^j}{z^j}\right) M(x) = -\frac{K_0}{z}\left(1 + \frac{D}{z} + \frac{D^2}{z^2} + \cdots + \frac{D^s}{z^s}\right) M(x),$$

pues se cumple que $D^j M(x) = M^{(j)}(x) \equiv 0$ para $j > s$. De la igualdad $u(1) = R(z)u(0)$, se deduce

$$\begin{aligned} R(z) = \frac{u(1)}{u(0)} &= \frac{-\frac{K_0}{z}\left(1 + \frac{D}{z} + \frac{D^2}{z^2} + \cdots + \frac{D^s}{z^s}\right) M(1)}{-\frac{K_0}{z}\left(1 + \frac{D}{z} + \frac{D^2}{z^2} + \cdots + \frac{D^s}{z^s}\right) M(0)} \\ &= \frac{M^{(s)}(1) + M^{(s-1)}(1)z + \cdots + M(1)z^s}{M^{(s)}(0) + M^{(s-1)}(0)z + \cdots + M(0)z^s}, \end{aligned}$$

que nos proporciona las expresiones de $P(z)$ y $Q(z)$ que buscábamos. □

Los aproximantes de Padé de una función son las funciones racionales que, fijado el grado del numerador y del denominador, tienen el mayor orden de aproximación posible a dicha función.

Para la función e^z, estos aproximantes óptimos se pueden obtener a partir de las relaciones (3.8) para $P(z)$ y $Q(z)$, utilizando el polinomio $M(x)$ tal que

$$M(x) = \frac{x^k (x-1)^j}{(k+j)!}, \tag{3.9}$$

para el cual $M^{(i)}(0) = 0$ para $i = 0, \ldots, k-1$ y $M^{(i)}(1) = 0$, $i = 0, \ldots, j-1$.

Teorema 3.3. *El aproximante de Padé (k, j) a la función exponencial viene dado por la función racional $R_{kj}(z) = P_{kj}(z)/Q_{kj}(z)$, donde*

$$\begin{aligned} P_{kj}(z) &= 1 + \frac{k}{j+k}z + \frac{k(k-1)}{(j+k)(j+k-1)} \cdot \frac{z^2}{2!} + \cdots + \frac{k(k-1)\cdots 1}{(j+k)\cdots(j+1)} \cdot \frac{z^k}{k!} \\ &= \sum_{l=0}^{k} \binom{k}{l} \frac{(k+j-l)!}{(k+j)!} z^l \\ Q_{kj}(z) &= 1 - \frac{j}{k+j}z + \frac{j(j-1)}{(k+j)(k+j-1)} \cdot \frac{z^2}{2!} + \cdots + (-1)^j \frac{j(j-1)\cdots 1}{(k+j)\cdots(k+1)} \cdot \frac{z^j}{j!} \\ &= P_{jk}(-z), \end{aligned}$$

con error

$$e^z - R_{kj}(z) = (-1)^j \frac{j!k!}{(j+k)!(j+k+1)!} z^{j+k+1} + \mathcal{O}(z^{j+k+2}).$$

$R_{kj}(z)$ es la única aproximación racional a e^z de orden $j + k$ tal que los grados del numerador y denominador son j y k, respectivamente.

Demostración. Por la definición de $M(x)$ en (3.9) y la anulación de sus sucesivas derivadas en 0 y en 1, si sustituimos en las expresiones para $P(z)$ y $Q(z)$ en (3.8), obtenemos las definiciones de los polinomios $P_{kj}(z)$ y $Q_{kj}(z)$ del enunciado, junto con la expresión de su error $e^z - R_{kj}(z)$.

La unicidad viene determinada por el hecho de que $M(x)$ tiene un cero en $x = 0$ de multiplicidad k y un cero en $x = 1$ de multiplicidad j. □

En la Tabla 3.2 se muestran los aproximantes de Padé de la función exponencial para distintas combinaciones de j y k.

$j\backslash k$	$k = 0$	$k = 1$	$k = 2$
$j = 1$	1	$1 + z$	$1 + z + z^2/2$
$j = 2$	$\dfrac{1}{1 - z}$	$\dfrac{1 + z/2}{1 - z/2}$	$\dfrac{1 + 2z/3 + z^2/6}{1 - z/3}$
$j = 3$	$\dfrac{1}{1 - z - z^2/2}$	$\dfrac{1 + z/3}{1 - 2z/3 + z^2/6}$	$\dfrac{1 + z/2 + z^2/12}{1 - z/2 + z^2/12}$
$j = 4$	$\dfrac{1}{1 - z + z^2/2 - z^3/6}$	$\dfrac{1 + z/4}{1 - 3z/4 + z^2/4 - z^3/24}$	$\dfrac{1 + 2z/5 + z^2/20}{1 - 3z/5 + 3z^2/20 - z^3/60}$

Tabla 3.2: Aproximantes de Padé (k, j) a la función exponencial.

Teorema 3.4. *Una aproximación de Padé $R_{kj}(z)$ es A-estable si, y solo si, $k \leq j \leq k + 2$. Todos sus ceros y polos son simples.*

Demostración. La demostración se encuentra en [2], Capítulo IV.4, Teorema 4.12. □

Veamos un resultado que concluirá el estudio de la A-estabilidad de los métodos de Radau IA y IIA.

Teorema 3.5. *Los métodos de Radau IA y IIA de s etapas tienen como función de estabilidad a la subdiagonal de Padé $(s - 1, s)$. Además, ambos métodos son A-estables y L-estables.*

Demostración.

- Radau IA: se tiene que $c_1 = 0$, y se cumplen las hipótesis simplificadoras $D(s)$ y $B(2s - 1)$. Por la Proposición 3.4, se cumple (3.7b), $a_{i1} = b_1$, $i = 1, \ldots, s$.
- Radau IIA: se tiene que $c_s = 1$, y se cumplen las hipótesis simplificadoras $C(s)$ y $B(2s - 1)$. Por la Proposición 3.4, se cumple (3.7a), $a_{sj} = b_j$, $j = 1, \ldots, s$.

Utilizando el apartado (2) de la Proposición 3.1., y sabiendo que se cumple (3.7a) o (3.7b), en la función de estabilidad obtenemos un polinomio de grado $s - 1$ para $P(z)$ y un polinomio de grado s para $Q(z)$. Como el orden de los métodos de Radau es $2s - 1$, entonces los aproximantes de Padé se corresponden con la subdiagonal $(s - 1, s)$.

La A-estabilidad de ambos métodos viene dada por el Teorema 3.4, con $k = s - 1$ y $j = s$. Por la Proposición 3.3, ambos métodos cumplen además que $R(\infty) = 0$, y podemos concluir que son L-estables. □

3.4. B-estabilidad

Consideramos ahora una ecuación no lineal $\mathbf{y}' = f(x, \mathbf{y})$ tal que se cumple la *condición unilateral de Lipschitz* para la norma euclídea, es decir,

$$\langle \mathbf{f}(x, \mathbf{y}) - \mathbf{f}(x, \mathbf{z}) \, , \, \mathbf{y} - \mathbf{z} \rangle \leq \nu \, ||\mathbf{y} - \mathbf{z}||^2, \tag{3.10}$$

donde ν es la constante unilateral de Lipschitz para la función $\mathbf{f}$.

Lema 3.1. *Sea $\boldsymbol{f}(x,\boldsymbol{y})$ una función continua que satisface* (3.10)*. Entonces, para cualquier par de soluciones $\boldsymbol{y}(x)$, $\boldsymbol{z}(x)$ de $\boldsymbol{y'}=\boldsymbol{f}(x,\boldsymbol{y})$, se cumple que*

$$||\boldsymbol{y}(x)-\boldsymbol{z}(x)|| \leq ||\boldsymbol{y}(x_0)-\boldsymbol{z}(x_0)||\, e^{\nu(x-x_0)}, \quad x \geq x_0. \tag{3.11}$$

Demostración. Sea $m(x)=||\mathbf{y}(x)-\mathbf{z}(x)||^2=\langle \mathbf{y}(x)-\mathbf{z}(x)\,,\,\mathbf{y}(x)-\mathbf{z}(x)\rangle$. Si derivamos $m(x)$, se tiene que

$$\begin{aligned} m'(x) &= 2\,\langle \mathbf{y}'(x)-\mathbf{z}'(x)\,,\,\mathbf{y}(x)-\mathbf{z}(x)\rangle = 2\,\langle \mathbf{f}(x,\mathbf{y}(x))-\mathbf{f}(x,\mathbf{z}(x))\,,\,\mathbf{y}(x)-\mathbf{z}(x)\rangle \\ &\leq 2\nu\,||\mathbf{y}(x)-\mathbf{z}(x)||^2 = 2\nu m(x) \end{aligned}$$

Esta desigualdad se puede resolver, obteniendo $m(x)\leq m(x_0)\,e^{\nu(x-x_0)}$, $x\geq x_0$, que equivale a (3.11). □

Si en (3.10) se toma $\nu \leq 0$, la distancia entre dos soluciones exactas cualesquiera de $\mathbf{y'}=\mathbf{f}(x,\mathbf{y})$ será una función no creciente que dependerá de x. Por tanto, se pretende que la misma propiedad se cumpla en la solución numérica. Consideramos a partir de ahora un método Runge-Kutta implícito de s etapas (1.3).

Definición 3.3. Un método Runge-Kutta es B-estable si la propiedad de contractividad

$$\langle \mathbf{f}(x,\mathbf{y})-\mathbf{f}(x,\mathbf{z})\,,\,\mathbf{y}-\mathbf{z}\rangle \leq 0 \tag{3.12}$$

implica que, para todo valor de $h\geq 0$, se cumple

$$||\mathbf{y}_{n+1}-\widehat{\mathbf{y}}_{n+1}|| \leq ||\mathbf{y}_n-\widehat{\mathbf{y}}_n||, \tag{3.13}$$

donde $\mathbf{y}_{n+1}$ e $\widehat{\mathbf{y}}_{n+1}$ son las soluciones numéricas tras dar un paso de longitud h con valores iniciales $\mathbf{y}_n$ e $\widehat{\mathbf{y}}_n$ respectivamente.

Proposición 3.5. *Si un método Runge-Kutta es B-estable, entonces es A-estable.*

Demostración. Consideramos la ecuación de Dahlquist (3.1). Como $y,\lambda\in\mathbb{C}$, se puede considerar $y=y_1+iy_2$, $\lambda=\alpha+i\beta$, es decir,

$$\begin{aligned} (y_1+iy_2)' = y_1'+iy_2' &= (\alpha+i\beta)\,(y_1+iy_2) = \alpha y_1+i\alpha y_2+i\beta y_1-\beta y_2 \\ &= (\alpha y_1-\beta y_2)+i\,(\beta y_1+\alpha y_2)\,. \end{aligned}$$

Esto equivale a tomar el siguiente sistema en $\mathbb{R}^2$

$$\begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \begin{pmatrix} \alpha & -\beta \\ \beta & \alpha \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = M \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}.$$

Por hipótesis, se tiene que $Re(\lambda)=\alpha << 0$. Si $y=y_1+iy_2$, $z=z_1+iz_2$, entonces

$$\begin{aligned} \langle My-Mz\,,\,y-z\rangle &= \langle M(y-z)\,,\,y-z\rangle \\ &= (\alpha\,(y_1-z_1)-\beta\,(y_2-z_2))\,(y_1-z_1)+(\beta\,(y_1-z_1)+\alpha\,(y_2-z_2))\,(y_2-z_2) \\ &= \alpha\left((y_1-z_1)^2+(y_2-z_2)^2\right)=\alpha\,|y-z|^2\leq 0, \end{aligned}$$

es decir, se cumple la propiedad de contractividad (3.12). Además, como se ha supuesto que el método es B-estable, entonces se cumplirá también (3.13).

Las aproximaciones numéricas y_{n+1} e $\widehat{y}_{n+1}$ satisfacen las igualdades

$$y_{n+1}=R(h\lambda)\,y_n, \qquad \widehat{y}_{n+1}=R(h\lambda)\,\widehat{y}_n.$$

Sustituyéndolo en (3.13),

$$||y_{n+1}-\widehat{y}_{n+1}|| = |R(h\lambda)|\cdot||y_n-\widehat{y}_n|| \leq ||y_n-\widehat{y}_n||,$$

de donde se deduce que $|R(h\lambda)|\leq 1$ para todo h, y el método es A-estable. □

Teorema 3.6. *Si los coeficientes de un método Runge-Kutta* (1.3) *satisfacen*

(i) $b_i \geq 0$, $i = 1, \ldots, s$,

(ii) *La matriz*

$$M = (m_{ij})_{i,j=1}^{s} = (b_i a_{ij} + b_j a_{ji} - b_i b_j)_{i,j=1}^{s} \tag{3.14}$$

es semidefinida positiva,

entonces el método es B-estable.

Demostración. Consideramos las siguientes diferencias:

$$\Delta \mathbf{y}_n = \mathbf{y}_n - \widehat{\mathbf{y}}_n, \quad \Delta \mathbf{y}_{n+1} = \mathbf{y}_{n+1} - \widehat{\mathbf{y}}_{n+1}, \quad \Delta \mathbf{Y}_i = \mathbf{Y}_i - \widehat{\mathbf{Y}}_i,$$

$$\Delta \mathbf{f}_i = h \left(\mathbf{f}(x_n + c_i h, \mathbf{Y}_i) - \mathbf{f}(x_n + c_i h, \widehat{\mathbf{Y}}_i) \right).$$

Restamos las ecuaciones del método Runge-Kutta (1.3) para las aproximaciones numéricas $\mathbf{y}_n$ e $\widehat{\mathbf{y}}_n$, obteniendo

$$\Delta \mathbf{Y}_i = \Delta \mathbf{y}_n + \sum_{j=1}^{s} a_{ij} \Delta \mathbf{f}_j, \quad 1 \leq i \leq s, \tag{3.15a}$$

$$\Delta \mathbf{y}_{n+1} = \Delta \mathbf{y}_n + \sum_{i=1}^{s} b_i \Delta \mathbf{f}_i. \tag{3.15b}$$

Tomando la norma euclídea para $\Delta \mathbf{y}_{n+1}$, se tiene

$$\begin{aligned} ||\Delta \mathbf{y}_{n+1}||^2 &= ||\Delta \mathbf{y}_n||^2 + \langle \Delta \mathbf{y}_n \,,\, \sum_{i=1}^{s} b_i \Delta \mathbf{f}_i \rangle + \langle \sum_{i=1}^{s} b_i \Delta \mathbf{f}_i \,,\, \Delta \mathbf{y}_n \rangle + \langle \sum_{i=1}^{s} b_i \Delta \mathbf{f}_i \,,\, \sum_{j=1}^{s} b_j \Delta \mathbf{f}_j \rangle \\ &= ||\Delta \mathbf{y}_n||^2 + 2 \sum_{i=1}^{s} b_i \langle \Delta \mathbf{f}_i \,,\, \Delta \mathbf{y}_n \rangle + \sum_{i,j=1}^{s} b_i b_j \langle \Delta \mathbf{f}_i \,,\, \Delta \mathbf{f}_j \rangle. \end{aligned}$$

Ahora bien, despejando $\Delta \mathbf{y}_n$ de (3.15a), y haciendo el producto escalar con $\Delta \mathbf{f}_i$, se tiene que

$$\langle \Delta \mathbf{f}_i \,,\, \Delta \mathbf{y}_n \rangle = \langle \Delta \mathbf{f}_i \,,\, \Delta \mathbf{Y}_i \rangle - \sum_{j=1}^{s} a_{ij} \langle \Delta \mathbf{f}_i \,,\, \Delta \mathbf{f}_j \rangle, \quad 1 \leq i \leq s,$$

y sustituyendo en la igualdad anterior, se obtiene que

$$\begin{aligned} ||\Delta y \mathbf{y}_{n+1}||^2 &= ||\Delta \mathbf{y}_n||^2 + 2 \sum_{i=1}^{s} b_i \langle \Delta \mathbf{f}_i \,,\, \Delta \mathbf{Y}_i \rangle - 2 \sum_{i,j=1}^{s} b_i a_{ij} \langle \Delta \mathbf{f}_i \,,\, \Delta \mathbf{f}_j \rangle + \sum_{i,j=1}^{s} b_i b_j \langle \Delta \mathbf{f}_i \,,\, \Delta \mathbf{f}_j \rangle \\ &= ||\Delta \mathbf{y}_n||^2 + 2 \sum_{i=1}^{s} b_i \langle \Delta \mathbf{f}_i \,,\, \Delta \mathbf{Y}_i \rangle - \sum_{i,j=1}^{s} (b_i a_{ij} + b_j a_{ji} - b_i b_j) \langle \Delta \mathbf{f}_i \,,\, \Delta \mathbf{f}_j \rangle \\ &= ||\Delta \mathbf{y}_n||^2 + 2 \sum_{i=1}^{s} b_i \langle \Delta \mathbf{f}_i \,,\, \Delta \mathbf{Y}_i \rangle - \sum_{i,j=1}^{s} m_{ij} \langle \Delta \mathbf{f}_i \,,\, \Delta \mathbf{f}_j \rangle. \end{aligned}$$

Además, $\langle \Delta \mathbf{f}_i \,,\, \Delta \mathbf{Y}_i \rangle = h \langle \mathbf{f}(x_n + c_i h, \mathbf{Y}_i) - \mathbf{f}(x_n + c_i h, \widehat{\mathbf{Y}}_i) \,,\, \mathbf{Y}_i - \widehat{\mathbf{Y}}_i \rangle \leq 0$. Como queremos que el método sea B-estable, faltaría por comprobar que $\sum_{i,j=1}^{s} m_{ij} \langle \Delta \mathbf{f}_i \,,\, \Delta \mathbf{f}_j \rangle \geq 0$. Pero esto es cierto por ser M una matriz semidefinida positiva.

De esta forma, obtenemos que

$$||\Delta \mathbf{y}_{n+1}||^2 = ||\Delta \mathbf{y}_n||^2 + 2 \sum_{i=1}^{s} b_i \langle \Delta \mathbf{f}_i \,,\, \Delta \mathbf{Y}_i \rangle - \sum_{i,j=1}^{s} m_{ij} \langle \Delta \mathbf{f}_i \,,\, \Delta \mathbf{f}_j \rangle \leq ||\Delta \mathbf{y}_n||^2,$$

que es equivalente a la condición de B-estabilidad del método. □

Definición 3.4. Los métodos que cumplen la propiedad definida en el Teorema 3.6 se denominan *algebraicamente estables.*

Teorema 3.7. *Consideramos una fórmula de cuadratura* $(c_i, b_i)_{i=1}^s$ *de orden* p*. Si se cumple que* $p \geq 2s-1$*, entonces* $b_i > 0$*,* $i = 1, \ldots, s$*.*

Demostración. Si el orden de la fórmula de cuadratura es $p \geq 2s-1$, los polinomios de grado $2s-2$ son integrados de forma exacta. Consideramos ahora el polinomio $p_i(t) = \prod_{j \neq i}^s \left(\frac{t-c_j}{c_i-c_j}\right)^2 \geq 0$, $i = 1, \ldots, s$. De esta forma, como $p_i(t)$ tiene grado $2s-2$, se obtiene que

$$\int_0^1 p_i(t)dt = \sum_{j=1}^s b_j\, p_i(c_j) = b_i > 0, \quad i = 1, \ldots, s.$$

□

Por tanto, para los métodos de Radau se cumple que $b_i > 0$, $i = 1, \ldots, s$.

Teorema 3.8. *Los métodos de Radau IA y IIA son algebraicamente estables y, por tanto, son B-estables.*

Demostración. Por el Teorema 3.7, como el orden de los métodos de Radau es $p = 2s-1$, se tiene que $b_i > 0$, $i = 1, \ldots, s$, y por tanto se cumple la condición **(i)** del Teorema 3.6.

Veamos ahora que también se cumple la condición **(ii)**. Consideramos la matriz de Vandermonde V_s y la matriz M definidas en (2.10) y en (3.14), respectivamente. El elemento (k, l) de la matriz producto $V_s \cdot M$ sería

$$(V_s \cdot M)_{kl} = \left(c_1^{k-1}, \ldots, c_s^{k-1}\right) \cdot \begin{pmatrix} b_1 a_{1l} + b_l a_{l1} - b_1 b_l \\ \vdots \\ b_s a_{sl} + b_l a_{ls} - b_s b_l \end{pmatrix} = \sum_{i=1}^s c_i^{k-1} \left(b_i a_{il} + b_l a_{li} - b_i b_l\right).$$

El método de Radau IA cumple las hipótesis simplificadoras $B(2s-1)$, $C(s-1)$ y $D(s)$. Aplicando dichas hipótesis a $V_s \cdot M \cdot V_s^T$, se obtiene que

$$\begin{aligned}
\left(V_s \cdot M \cdot V_s^T\right)_{kl} &= \sum_{i,j=1}^s c_i^{k-1} c_j^{l-1} \left(b_i a_{ij} + b_j a_{ji} - b_i b_j\right) \\
&= \sum_{j=1}^s c_j^{l-1} \left(\sum_{i=1}^s b_i c_i^{k-1} a_{ij}\right) + \sum_{i=1}^s c_i^{k-1} \left(\sum_{j=1}^s b_j c_j^{l-1} a_{ji}\right) - \left(\sum_{i=1}^s b_i c_i^{k-1}\right)\left(\sum_{j=1}^s b_j c_j^{l-1}\right) \\
&= \sum_{j=1}^s c_j^{l-1} \cdot \frac{b_j(1-c_j^k)}{k} + \sum_{i=1}^s c_i^{k-1} \cdot \frac{b_i(1-c_i^l)}{l} - \frac{1}{kl} \cdot \frac{1}{l} \\
&= \frac{1}{k} \sum_{j=1}^s \left(b_j c_j^{l-1} - b_j c_j^{k+l-1}\right) + \frac{1}{l} \sum_{i=1}^s \left(b_i c_i^{k-1} - b_i c_i^{k+l-1}\right) - \frac{1}{kl} \\
&= \frac{1}{k}\left(\frac{1}{l} - \frac{1}{k+l}\right) + \frac{1}{l}\left(\frac{1}{k} - \frac{1}{k+l}\right) - \frac{1}{kl} = \frac{1}{kl} - \frac{1}{k(k+l)} - \frac{1}{l(k+l)} = 0.
\end{aligned}$$

Por tanto, $V_s \cdot M \cdot V_s^T = 0$ y, como V_s es regular, entonces M es la matriz nula. En particular, se cumple la condición (ii) del Teorema 3.6, pues M es semidefinida positiva, y el método de Radau IA es B-estable.

Se razona de forma similar para el método de Radau IIA, utilizando las hipótesis simplificadoras $B(2s-1)$, $C(s)$ y $D(s-1)$. □

Capítulo 4

Implementación del método de Radau IIA

En este capítulo se estudiarán diversos aspectos relacionados con la implementación de un método Runge-Kutta implícito, se abordará la implementación con paso variable del método de Radau IIA de 3 etapas y se mostrarán resultados numéricos obtenidos con dicho método para la integración de las ecuaciones del oscilador de Van der Pol.

4.1. Reformulación del sistema no lineal

Si la dimensión del sistema diferencial $\mathbf{y}'(x) = \mathbf{f}(x, \mathbf{y}(x))$ es D y se utiliza para su integración un método Runge-Kutta implícito de s etapas, en cada paso hay que resolver un sistema no lineal de tamaño $D \times s$ para obtener las etapas intermedias $\mathbf{Y}_1, \ldots, \mathbf{Y}_s$.

Tomamos ahora $\mathbf{Z}_i = \mathbf{Y}_i - \mathbf{y}_n$ para $i = 1, \ldots, s$. Entonces, un paso del método Runge-Kutta (1.3) se puede reescribir como

$$\mathbf{Z}_i = h \sum_{j=1}^{s} a_{ij}\, \mathbf{f}\left(x_n + c_j h, \mathbf{y}_n + \mathbf{Z}_j\right), \quad 1 \leq i \leq s, \tag{4.1a}$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h \sum_{i=1}^{s} b_i\, \mathbf{f}\left(x_n + c_i h, \mathbf{y}_n + \mathbf{Z}_i\right). \tag{4.1b}$$

Si las soluciones $\mathbf{Z}_1, \ldots, \mathbf{Z}_s$ de (4.1a) se conocen, entonces se podrá hallar $\mathbf{y}_{n+1}$ mediante (4.1b), pero esto requiere s evaluaciones adicionales de $\mathbf{f}$. Esto puede evitarse si la matriz $\mathcal{A} = (a_{ij})_{i,j=1}^{s}$ del método Runge-Kutta es no singular. Para ello, escribimos (4.1) en formato matricial

$$\begin{pmatrix} \mathbf{Z}_1 \\ \vdots \\ \mathbf{Z}_s \end{pmatrix} = h \cdot \begin{pmatrix} \sum_{j=1}^{s} a_{1j}\, \mathbf{f}\left(x_n + c_1 h, \mathbf{y}_n + \mathbf{Z}_1\right) \\ \vdots \\ \sum_{j=1}^{s} a_{sj}\, \mathbf{f}\left(x_n + c_s h, \mathbf{y}_n + \mathbf{Z}_s\right) \end{pmatrix} = h \cdot (\mathcal{A} \otimes I) \cdot \begin{pmatrix} \mathbf{f}\left(x_n + c_1 h, \mathbf{y}_n + \mathbf{Z}_1\right) \\ \vdots \\ \mathbf{f}\left(x_n + c_s h, \mathbf{y}_n + \mathbf{Z}_s\right) \end{pmatrix}, \tag{4.2}$$

donde, si I es la matriz identidad de tamaño $D \times D$, $\mathcal{A} \otimes I$ es la matriz

$$\mathcal{A} \otimes I = \begin{pmatrix} a_{11} I & \cdots & a_{1s} I \\ \vdots & \ddots & \vdots \\ a_{s1} I & \cdots & a_{ss} I \end{pmatrix} \in \mathbb{R}^{sD \times sD}.$$

Además, tomando $\mathcal{Z} = (\mathbf{Z}_1, \ldots, \mathbf{Z}_s)^T$ y $\mathbf{F}(\mathcal{Z}) = \left(\mathbf{f}\left(x_n + c_1 h, \mathbf{y}_n + \mathbf{Z}_1\right), \ldots, \mathbf{f}\left(x_n + c_s h, \mathbf{y}_n + \mathbf{Z}_s\right)\right)^T$, el sistema (4.2) se puede reescribir como $\mathcal{Z} = h \cdot (\mathcal{A} \otimes I) \cdot \mathbf{F}(\mathcal{Z})$, es decir,

$$\Phi(h, \mathcal{Z}) = \mathcal{Z} - h \cdot (\mathcal{A} \otimes I) \cdot \mathbf{F}(\mathcal{Z}) = 0.$$

De esta forma, obtenemos que (4.1b) es equivalente a

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \sum_{i=1}^{s} d_i \mathbf{Z}_i, \tag{4.3}$$

donde $\mathbf{d} = (d_1, \ldots, d_s) = \mathbf{b}^T \cdot \mathcal{A}^{-1}$, pero antes es preciso resolver las ecuaciones no lineales $\Phi(h, \mathcal{Z}) = 0$.

4.2. Simplificación de las iteraciones de Newton

Para un sistema diferencial no lineal, (4.1a) se puede resolver mediante la iteración de punto fijo o mediante la iteración de Newton.

La iteración de punto fijo, aplicada a (4.1a), sería

$$\mathbf{Z}_i^{[\nu+1]} = h \sum_{j=1}^{s} a_{ij} \mathbf{f}(x_n + c_j h, \mathbf{y}_n + \mathbf{Z}_j^{[\nu]}), \quad 1 \leq i \leq s, \quad \nu = 0, 1, 2, \ldots,$$

donde $\nu = 0, 1, 2, \ldots$ es el índice de la iteración y $\mathbf{Z}_i^{[0]} = \mathbf{0}$, $1 \leq i \leq s$, es el iterante inicial tomado. El proceso se repetirá mientras se cumpla $||\mathcal{Z}^{[\nu+1]} - \mathcal{Z}^{[\nu]}||_\infty > TOL$, siendo TOL una tolerancia fijada y $\mathcal{Z}^{[\nu]} = (\mathbf{Z}_1^{[\nu]}, \ldots, \mathbf{Z}_s^{[\nu]})^T$ la ν-ésima aproximación a la solución de (4.1a).

El problema de la iteración de punto fijo es que convierte al algoritmo en un método explícito y se pierden las buenas propiedades de estabilidad del método implícito que se está utilizando.

El método de Newton, aplicado a (4.1a), necesita resolver en la iteración ν-ésima un sistema lineal en $\mathbb{R}^{sD}$ cuya matriz es

$$\mathbf{D}\Phi(h, \mathcal{Z}^{[\nu]}) = \begin{pmatrix} I - ha_{11}\partial_2 \mathbf{f}_1^{[\nu]} & -ha_{12}\partial_2 \mathbf{f}_2^{[\nu]} & \cdots & -ha_{1s}\partial_2 \mathbf{f}_s^{[\nu]} \\ -ha_{21}\partial_2 \mathbf{f}_1^{[\nu]} & I - ha_{22}\partial_2 \mathbf{f}_2^{[\nu]} & \cdots & -ha_{2s}\partial_2 \mathbf{f}_s^{[\nu]} \\ \vdots & \vdots & \ddots & \vdots \\ -ha_{s1}\partial_2 \mathbf{f}_1^{[\nu]} & -ha_{s2}\partial_2 \mathbf{f}_2^{[\nu]} & \cdots & -ha_{ss}\partial_2 \mathbf{f}_s^{[\nu]} \end{pmatrix} \in \mathbb{R}^{sD \times sD},$$

donde $\mathbf{f}_i^{[\nu]} = \mathbf{f}(x_n + c_i h, \mathbf{y}_n + \mathbf{Z}_i^{[\nu]}), 1 \leq i \leq s$ y ∂_2 denota la derivada parcial de una función respecto de su segundo argumento.

Para simplificarlo, sustituimos los jacobianos $\partial_2 \mathbf{f}_i^{[\nu]}$, $1 \leq i \leq s$, por una aproximación $J \approx \partial_2 \mathbf{f}(x_n, \mathbf{y}_n)$, obteniendo así que la nueva matriz de nuestro sistema lineal es

$$\mathbf{D}\Phi(h, \mathcal{Z}^{[0]}) = I - h(\mathcal{A} \otimes J) = \begin{pmatrix} I - ha_{11}J & -ha_{12}J & \cdots & -ha_{1s}J \\ -ha_{21}J & I - ha_{22}J & \cdots & -ha_{2s}J \\ \vdots & \vdots & \ddots & \vdots \\ -ha_{s1}J & -ha_{s2}J & \cdots & -ha_{ss}J \end{pmatrix}$$

De esta forma, tomando como iterante inicial $\mathbf{Z}_i^{[0]} = \mathbf{0}$, $1 \leq i \leq s$, las iteraciones de Newton simplificadas se convierten en

$$\mathbf{D}\Phi(h, \mathcal{Z}^{[0]})\, \Delta\mathcal{Z}^{[\nu]} = -\Phi(h, \mathcal{Z}^{[\nu]}) \tag{4.4a}$$

$$\mathcal{Z}^{[\nu+1]} = \mathcal{Z}^{[\nu]} + \Delta\mathcal{Z}^{[\nu]}, \quad \nu = 0, 1, 2, \ldots, \tag{4.4b}$$

donde $\Delta\mathcal{Z}^{[\nu]} = (\Delta\mathbf{Z}_1^{[\nu]}, \ldots, \Delta\mathbf{Z}_s^{[\nu]})^T$, la solución del sistema lineal (4.4), se corresponde con los incrementos, es decir, $\Delta\mathcal{Z}^{[\nu]} = \mathcal{Z}^{[\nu+1]} - \mathcal{Z}^{[\nu]}$.

Cada iteración del método de Newton según (4.4) requiere s evaluaciones de la función $\mathbf{f}$ y hallar la solución de un sistema lineal de dimensión sD. La matriz $\mathbf{D}\Phi(h, \mathcal{Z}^{[0]})$ es siempre la misma para todas las iteraciones de un paso, y bastará con calcularla una única vez, hallando también su factorización LU, lo que reduce notablemente el costo computacional.

Además, se pedirá que el proceso iterativo se repita mientras $||\Delta\mathcal{Z}^{[\nu]}||_\infty > TOL$, donde TOL es una tolerancia fijada.

4.3. Elección del iterante inicial, estimación del error local y cambio de paso

En la iteración de punto fijo se escogió como iterante inicial

$$\mathbf{Z}_i^{[0]} = \mathbf{0}, \quad 1 \leq i \leq s, \tag{4.5}$$

que también se puede utilizar como iterante inicial cuando se emplea la iteración de Newton. Esto es posible, pues la solución exacta del sistema (4.1a) cumple $\mathbf{Z}_i = \mathcal{O}(h)$, $1 \leq i \leq s$, aunque por lo general se pueden hacer mejores elecciones para dicho iterante inicial.

Si nuestro método Runge-Kutta implícito satisface la hipótesis simplificadora $C(\eta)$ para algún $\eta \leq s$, por el Lema 2.1, tenemos que $\mathbf{Y}_i - \mathbf{y}(x_n + c_i h) = \mathcal{O}(h^{\eta+1})$ para $1 \leq i \leq s$, es decir,

$$\mathbf{Z}_i = \mathbf{y}(x_n + c_i h) - \mathbf{y}_n + \mathcal{O}(h^{\eta+1}), \quad 1 \leq i \leq s, \quad \eta \leq s. \tag{4.6}$$

Supongamos ahora que $c_i \neq 0$ para $i = 1, \ldots, s$. En particular, esto ocurre en el método de Radau IIA que se implementará posteriormente. Consideramos el polinomio interpolador $p(t)$ de grado s, tal que $p(0) = \mathbf{0}$ y $p(c_i) = \mathbf{Z}_i$, $1 \leq i \leq s$. Como el error de interpolación es de tamaño $\mathcal{O}(h^{s+1})$, entonces

$$\mathbf{y}(x_n + th) - \mathbf{y}_n - p(t) = \mathcal{O}(h^{\eta+1}).$$

Tomando como iterantes iniciales a

$$\mathbf{Z}_i^{[0]} = p(1 + \tau c_i) + \mathbf{y}_n - \mathbf{y}_{n+1}, \quad 1 \leq i \leq s, \quad \tau = h_n'/h_n, \tag{4.7}$$

donde h_n' es la nueva longitud de paso calculada mediante iteración de Newton, se puede comprobar para el método de Radau IIA de 3 etapas y orden 5 que este converge más rápidamente que tomando (4.5).

Para permitir cambios en la longitud de paso de integración, de modo que el error local se mantenga por debajo de una tolerancia fijada $TOLV$, describiremos un par encajado de métodos Runge-Kutta basado en el método de Radau IIA de 3 etapas y orden 5.

Para esto, usaremos un método de menor orden, pues nuestro método ya tiene un orden óptimo. Este será de la forma

$$\widehat{\mathbf{y}}_{n+1} = \mathbf{y}_n + h\left(\widehat{b}_0\,\mathbf{f}(x_n, \mathbf{y}_n) + \sum_{i=1}^{3} \widehat{b}_i\,\mathbf{f}(x_n + c_i h, \mathbf{Y}_i)\right), \tag{4.8}$$

siendo $\mathbf{Y}_1$, $\mathbf{Y}_2$ e $\mathbf{Y}_3$ las etapas intermedias del método de Radau IIA de 3 etapas y $\widehat{b}_0 \neq 0$ un parámetro por determinar. Siguiendo [2], elegimos el coeficiente $\widehat{b}_0$ de forma que $\widehat{b}_0 = \gamma_0 = \gamma^{-1}$, donde γ es el autovalor real de la matriz $\mathcal{A}^{-1}$.

Entonces, restando (1.3b) de (4.8), se obtiene que

$$\begin{aligned}\widehat{\mathbf{y}}_{n+1} - \mathbf{y}_{n+1} &= \gamma_0 h\mathbf{f}(x_n, \mathbf{y}_n) + h\sum_{i=1}^{3}(\widehat{b}_i - b_i)h\mathbf{f}(x_n + c_i h, \mathbf{Y}_i) \\ &= \gamma_0 h\mathbf{f}(x_n, \mathbf{y}_n) + e_1\mathbf{Z}_1 + e_2\mathbf{Z}_2 + e_3\mathbf{Z}_3,\end{aligned} \tag{4.9}$$

igualdad que se puede usar para la estimación del error local. Como queremos que $\widehat{\mathbf{y}}_{n+1} - \mathbf{y}_{n+1} = \mathcal{O}(h^4)$, los coeficientes e_i, $i = 1, 2, 3$, vienen dados por

$$(e_1, e_2, e_3) = \frac{\gamma_0}{3}(-13 - 7\sqrt{6}, -13 + 7\sqrt{6}, -1).$$

Sin embargo, para $\mathbf{y}'(t) = \lambda\mathbf{y}(t)$ y $h\lambda \to \infty$, $\widehat{\mathbf{y}}_{n+1} - \mathbf{y}_{n+1} \approx \gamma_0 h\lambda\mathbf{y}_n$, que no está acotado. Por tanto, como estamos interesados en utilizar el método para integrar problemas rígidos, en los que λ es negativo de gran valor absoluto, vemos que no podemos usar directamente la diferencia $\widehat{\mathbf{y}}_{n+1} - \mathbf{y}_{n+1}$ para la estimación del error.

Tomaremos pues la siguiente expresión para el error:

$$err = (I - h\gamma_0 J)^{-1}(\widehat{\mathbf{y}}_{n+1} - \mathbf{y}_{n+1}), \tag{4.10}$$

donde hacemos intervenir a la matriz Jacobiana J y necesitamos calcular la factorización LU de la matriz $(I - h\gamma_o J)$. Si tomamos $h \to 0$, entonces $err = \mathcal{O}(h^4)$, y tomando $h\lambda \to \infty$, $err \to -1$. Este comportamiento es bueno cuando $h\lambda$ tiende a infinito, pero todavía es mejorable.

Para el primer paso, y para cada paso rechazado con $||err|| > TOLV$, se puede tomar en lugar de err el nuevo estimativo

$$\widetilde{err} = (I - h\gamma_0 J)^{-1}(\gamma_0 h\mathbf{f}(x_n, \mathbf{y}_n + err) + e_1\mathbf{Z}_1 + e_2\mathbf{Z}_2 + e_3\mathbf{Z}_3), \tag{4.11}$$

que cumple que $\widetilde{err} \to 0$ si $h\lambda \to \infty$, que $\widetilde{err} = \mathcal{O}(h^4)$ para $h \to 0$ y que requiere una evaluación más de $\mathbf{f}$ y la resolución de un sistema lineal adicional, pero con la misma matriz $(I - h\gamma_0 J)$ cuya factorización LU se ha calculado previamente.

Si tenemos que h_n es la longitud del paso actual, y se cumple que $||err|| < TOLV$, entonces tomaremos como nueva longitud de paso para avanzar desde x_{n+1}

$$h_{n+1} = fac \cdot h_n \cdot \left(\frac{||err||}{TOLV}\right)^{-1/4}, \tag{4.12}$$

donde $fac = 0{,}9 \times (2k_{max} + 1)/(2k_{max} + niter)$, siendo k_{max} el número máximo de iteraciones para el método de Newton y $niter$ el número de iteraciones dadas en el paso actual. Además, si h_{n+1} satisface

$$c_1 h_n \leq h_{n+1} \leq c_2 h_n,$$

con $c_1 = 1{,}0$ y $c_2 = 1{,}2$, entonces se mantiene la longitud de paso h_n para el paso siguiente. Si $||err|| > TOLV$, el paso h_n que se ha utilizado es demasiado grande y tras rechazar la aproximación $\mathbf{y}_{n+1}$ calculada se utiliza el lado derecho de (4.12) para calcular una nueva longitud de paso con la que avanzar desde x_n.

Sin embargo, la fórmula (4.12) tiene una desventaja: no es posible reducir la longitud de paso más que fac sin que haya rechazos cuando se quiera disminuir dicha longitud de paso (si $h_{n+1} < fac \cdot h_n$, entonces $||err|| > TOLV$). Denotando por err_{n+1} a la expresión (4.10), con longitud de paso h_n, las nuevas longitudes de paso suelen relacionarse con la fórmula asintótica

$$||err_{n+1}|| = C_n h_n^4. \tag{4.13}$$

En (4.12) se supone que $C_{n+1} \approx C_n$, lo cual no suele ser muy acertado. Para obtener un modelo más preciso, se puede asumir que la función $log(C_n)$ es una función lineal sobre n, es decir, que $log(C_{n+1}) - log(C_n)$ es constante o que se cumple que

$$\frac{C_{n+1}}{C_n} \approx \frac{C_n}{C_{n-1}}. \tag{4.14}$$

Tomando ahora C_n y C_{n-1} a partir de (4.13) y C_{n+1} de la igualdad $1 = C_{n+1}h_{n+1}^4$, insertándolos en (4.14) obtenemos la nueva longitud de paso,

$$h_{n+1} = fac \cdot h_n \cdot \left(\frac{||err_{n+1}||}{TOLV}\right)^{-1/4} \cdot \frac{h_n}{h_{n-1}} \cdot \left(\frac{||err_n||}{||err_{n+1}||}\right)^{1/4}. \tag{4.15}$$

4.4. Resultados numéricos

Para ilustrar el comportamiento de los métodos de Radau IIA consideraremos la ecuación de Van der Pol, que modela el movimiento de un oscilador con un amortiguamiento no lineal. Escrito como sistema de primer orden, las ecuaciones son

$$y_1' = y_2, \tag{4.16a}$$
$$\epsilon \cdot y_2' = \left(1 - y_1^2\right) y_2 - y_1, \tag{4.16b}$$

donde $\mathbf{y}(t) = (y_1(t), y_2(t))^T$, ϵ es una constante real positiva pequeña ($\epsilon \ll 1$), la condición inicial es $\mathbf{y}(0) = (2, 0)^T$, y el intervalo de integración es $[0, 11]$, como en [6].

Estudiaremos este problema para diferentes valores de ϵ, en particular diferenciando el caso $\epsilon = 1$ de $\epsilon < 1$. En la Figura 4.1 se muestra la evolución de las dos componentes de la solución con el tiempo para $\epsilon = 1$, y en la Figura 4.2 para $\epsilon = 0{,}1$ y $\epsilon = 0{,}01$. En las dos figuras se observa un comportamiento casi-periódico de la solución. Además, la amplitud de las oscilaciones de la primera componente es similar para los tres valores del parámetro, pero a medida que se reduce el valor de ϵ, el periodo del oscilador también se reduce, y aumenta la dificultad para integrar numéricamente dicho problema, al producirse cada vez con más frecuencia variaciones grandes de la solución en intervalos de tiempo cada vez más reducidos. Las gráficas de la segunda componente de la solución (la derivada de $y_1(t)$) muestran este hecho pues, para $\epsilon = 1$, dicha componente está acotada en valor absoluto por 3, mientras que para $\epsilon = 0{,}01$ la cota es cercana a 150.

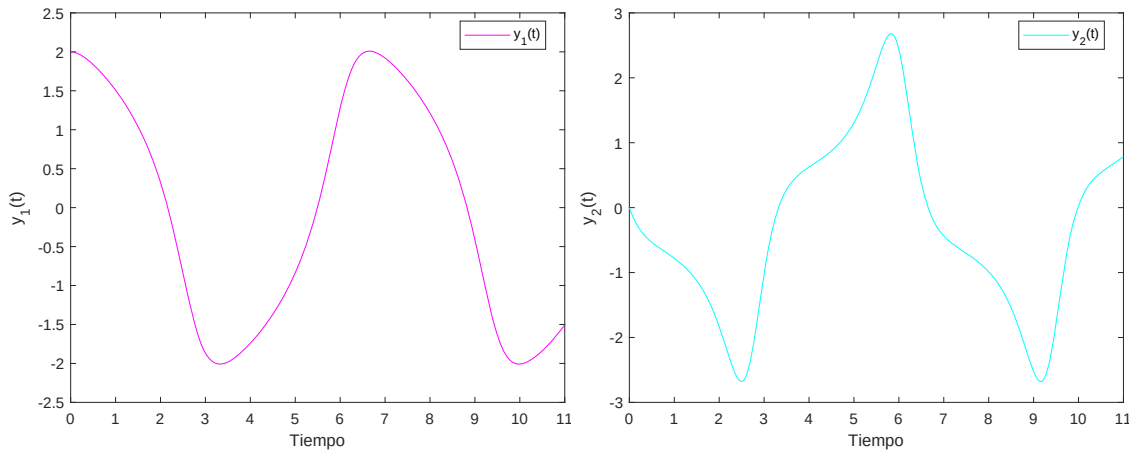


Figura 4.1: Solución del oscilador de Van der Pol frente al tiempo para $\epsilon = 1$.

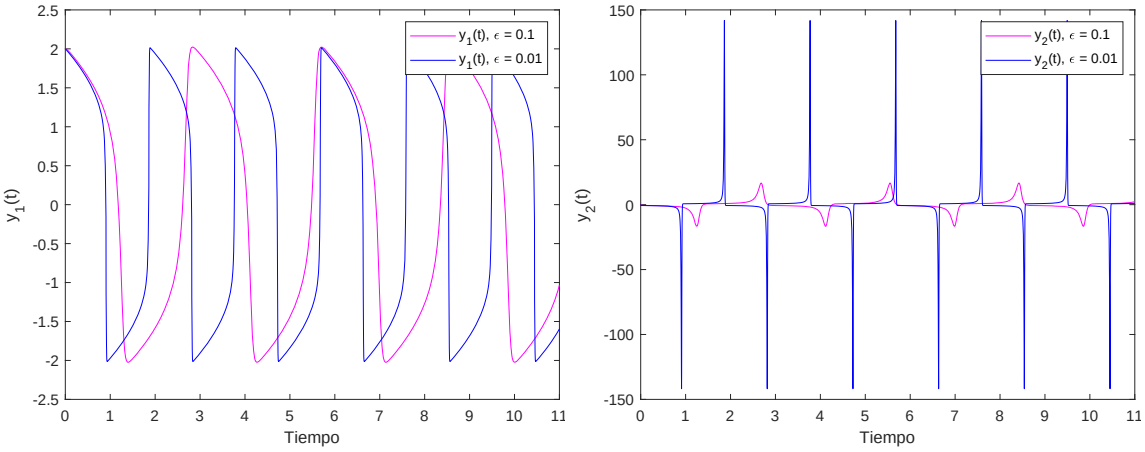


Figura 4.2: Solución del oscilador de Van der Pol frente al tiempo para $\epsilon = 0{,}1$ y $\epsilon = 0{,}01$.

4.4.1. Solución numérica para $\epsilon = 1$

Se han implementado en primer lugar los métodos de Radau IIA de orden 3 y 5 con paso fijo, resolviendo las ecuaciones no lineales en cada paso mediante iteración de punto fijo e iteración de Newton. Se han utilizado las cuatro implementaciones para integrar numéricamente el oscilador de Van der Pol (4.16) con $\epsilon = 1$, valor para el cual el problema no es rígido. Las longitudes de paso utilizadas han sido $h = 1/4, 1/8, \ldots, 1/128$ para el método de orden 3 y $h = 1/4, 1/8, \ldots, 1/64$ para el método de orden 5.

Para dichos métodos, tomamos como tolerancias para detener la resolución numérica de las ecuaciones no lineales $TOL3 = h^3/10$ para el método de orden 3 y $TOL5 = h^5/10$ para el método de orden 5. De este modo, $TOL3 < h^3$ y $TOL5 < h^5$, y por tanto el error del integrador no resulta contaminado por el error resultante al detener la iteración, ya sea la de punto fijo o la de Newton, pues el error global de los métodos de Radau IIA es de $\mathcal{O}(h^p)$ cuando $h \to 0$, y este supera a las tolerancias utilizadas en cada caso.

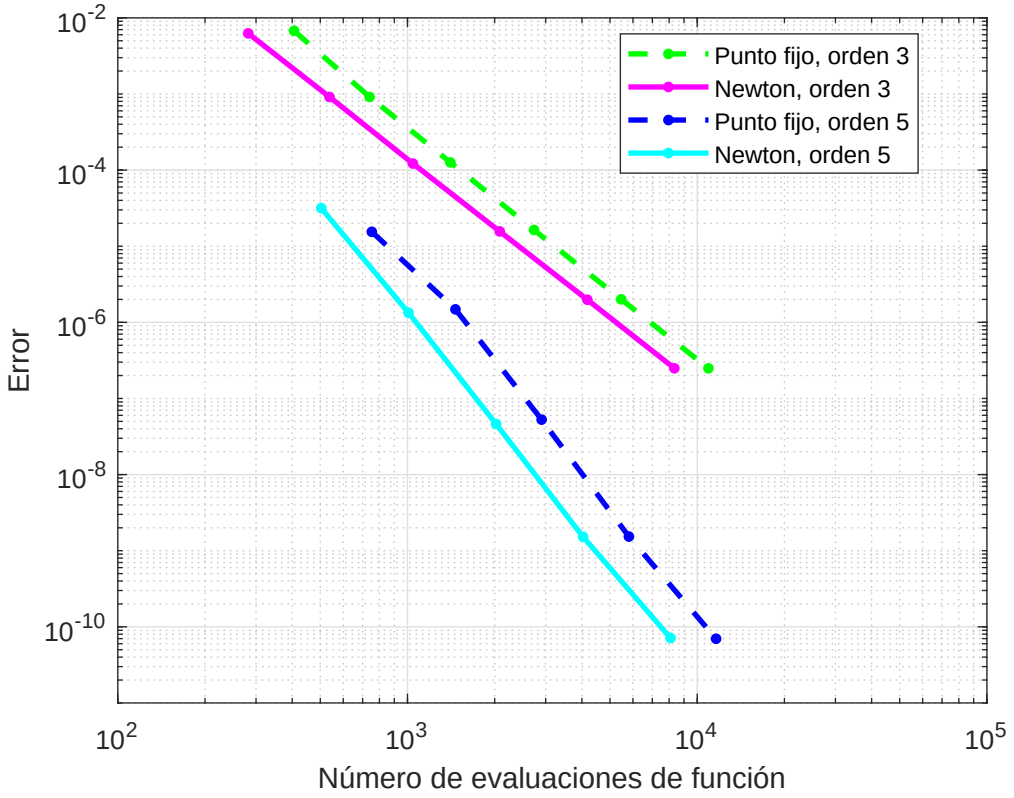


Figura 4.3: Error frente al número de evaluaciones de $\mathbf{f}$ para los métodos de Radau IIA de orden 3 y 5 cuando se utiliza iteración de punto fijo e iteración de Newton.

En la Figura 4.3 se ha representado el error en $T = 11$ (medido en la norma del máximo) frente al número de evaluaciones de función realizadas. Se puede observar que la pendiente de las dos líneas que corresponden a datos generados con el método de orden 3 es muy cercana a -3, mientras que las dos líneas que corresponden a datos generados con el método de orden 5 muestran una pendiente muy próxima a -5, es decir, cuando $h \to 0$, el error global se comporta como $\mathcal{O}(h^3)$ y $\mathcal{O}(h^5)$ para los métodos de Radau IIA de orden 3 y 5, respectivamente. Esto se corresponde con lo esperado, pues si se tiene un método Runge-Kutta de orden p, su error global será de la forma $\mathcal{O}(h^p)$.

Tanto para el método de orden 3 como para el de orden 5, el error global frente al número de evaluaciones de función proporciona un mejor resultado para la iteración de Newton que para la de punto fijo, pues esta última requiere menos evaluaciones de función y proporciona errores del mismo tamaño.

En la Figura 4.4 se ha representado en escala doblemente logarítmica el error frente al tiempo CPU. En este caso, las pendientes de las líneas se mantienen y la iteración de punto fijo para ambos métodos de Radau necesita menos tiempo de cálculo y es más eficiente, pues el método de Newton nos exige la evaluación de la matriz Jacobiana de la función $\mathbf{f}$ en cada paso. Sin embargo, seguimos obteniendo los resultados esperados en ambos casos, errores de tamaño $\mathcal{O}(h^3)$ para el método de orden 3 y $\mathcal{O}(h^5)$ para el de orden 5.

De las cuatro implementaciones consideradas, el método de orden 5 implementado con iteración de punto fijo es la combinación más eficiente para las longitudes de paso utilizadas.

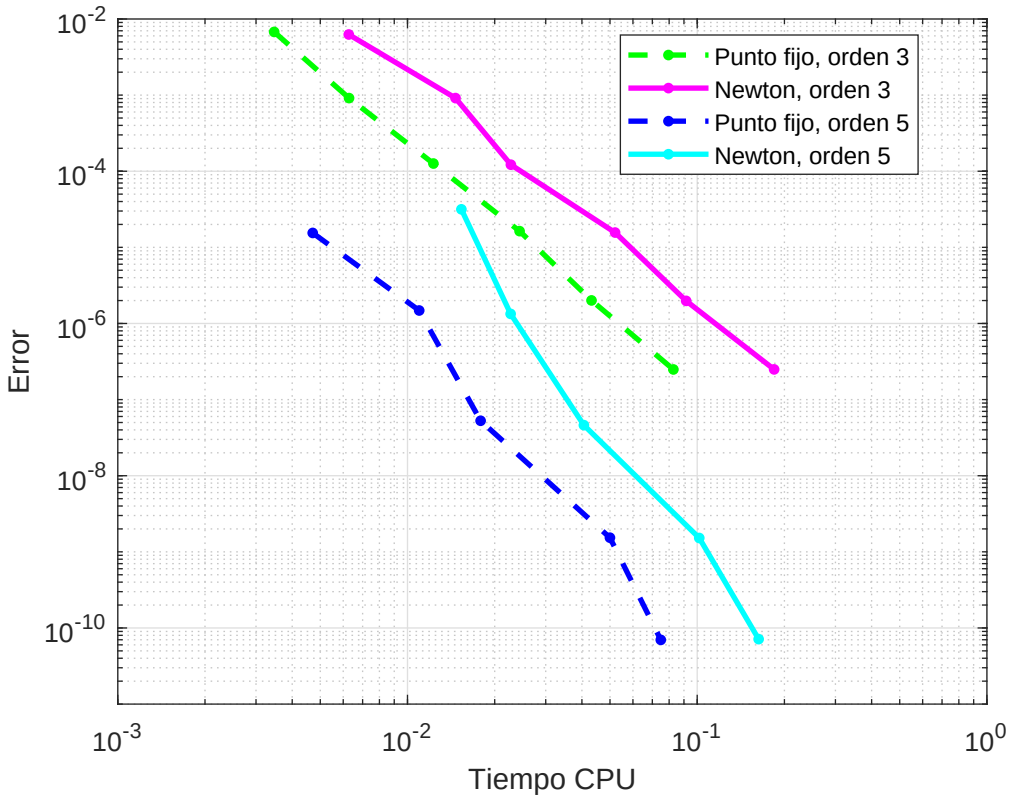


Figura 4.4: Error frente al tiempo CPU de los métodos de Radau IIA de orden 3 y 5 cuando se utiliza iteración de punto fijo e iteración de Newton.

4.4.2. Solución numérica para $\epsilon < 1$

Se ha implementado el método de Radau IIA de orden 5 con paso variable, resolviendo en cada paso las ecuaciones no lineales mediante iteración de Newton (ver [7]). Dicha implementación ha sido utilizada para integrar numéricamente el oscilador de Van der Pol (4.16) para $\epsilon < 1$, pues bajo estas condiciones el problema es rígido y se hace necesario el paso variable. En particular, se han tomado como valores del parámentro ϵ a 0,1, 0,01 y 0,001.

Las tolerancias utilizadas para controlar el error local han sido $TOLV = 10^{-3}, 10^{-4}, \ldots, 10^{-10}$. En cada ejecución se ha tomado como tolerancia para detener la resolución numérica de las ecuaciones no lineales $TOL = TOLV/100$. De este modo, el error del integrador no resulta contaminado por el error resultante al detener la iteración de Newton.

En la Figura 4.5 se muestra en escala semilogarítmica la evolución de la longitud de paso utilizada con el tiempo para $\epsilon = 0{,}001$ (gráfica derecha) y para $\epsilon = 0{,}01$ (gráfica superior izquierda). En ambos casos observamos un comportamiento casi periódico en la elección de h, que se corresponde con el comportamiento casi periódico de la solución exacta y de su derivada. Además, los tiempos en los que la derivada de la solución alcanza los valores máximos y mínimos se corresponden con los tiempos en los que el paso de integración se hace más pequeño.

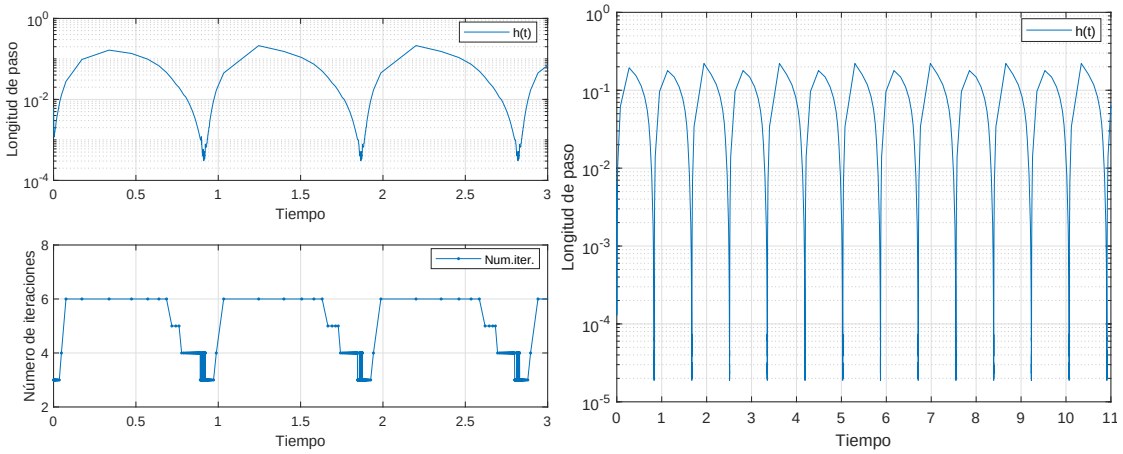


Figura 4.5: Longitud de paso frente al tiempo para $\epsilon = 0{,}01$ y $0{,}001$, respectivamente, junto con el número de iteraciones de cada paso aceptado en la iteración de Newton para $\epsilon = 0{,}01$.

También se observa que mientras para $\epsilon = 0{,}01$ las longitudes de paso varían entre 3×10^{-4} y 2×10^{-1}, cuando $\epsilon = 0{,}001$, se mantiene la longitud de paso máxima, pero la mínima se reduce hasta 2×10^{-5}, como corresponde a una mayor dificultad del problema.

Por último, en la gráfica inferior izquierda se han representado la evolución del número de iteraciones de Newton con el tiempo. Se observa el mismo carácter casi periódico de las longitudes de paso, y cómo cuando la longitud de paso es más pequeña el número de iteraciones necesarias también disminuye.

En la Figura 4.6 se ha representado en escala doblemente logarítmica el error en $T = 11$ frente al número de evaluaciones de función realizadas para los tres valores del parámetro $\epsilon < 1$. Se puede comprobar que, para dichos valores de ϵ, la pendiente obtenida mediante iteración de Newton para el método de Radau IIA de orden 5 es próxima a -5, y por tanto, cuando $h \to 0$, el error global se comporta como $\mathcal{O}(h^5)$, como se esperaba.

Observamos que el número de evaluaciones realizadas es mayor cuando ϵ es más cercano a 0, puesto que las longitudes de paso mínimas son más pequeñas y el número de pasos es mayor.

En la Figura 4.7 se ha representado en escala doblemente logarítmica el error frente al tiempo CPU. Al igual que en la Figura 4.6, para las tolerancias más pequeñas obtenemos mejores aproximaciones para los valores de ϵ menores, pero el costo computacional es más elevado. Las pendientes obtenidas se mantienen, son cercanas a -5 en los tres casos considerados, y obtenemos que el comportamiento del error global es $\mathcal{O}(h^5)$ si $h \to 0$, que es lo esperado.

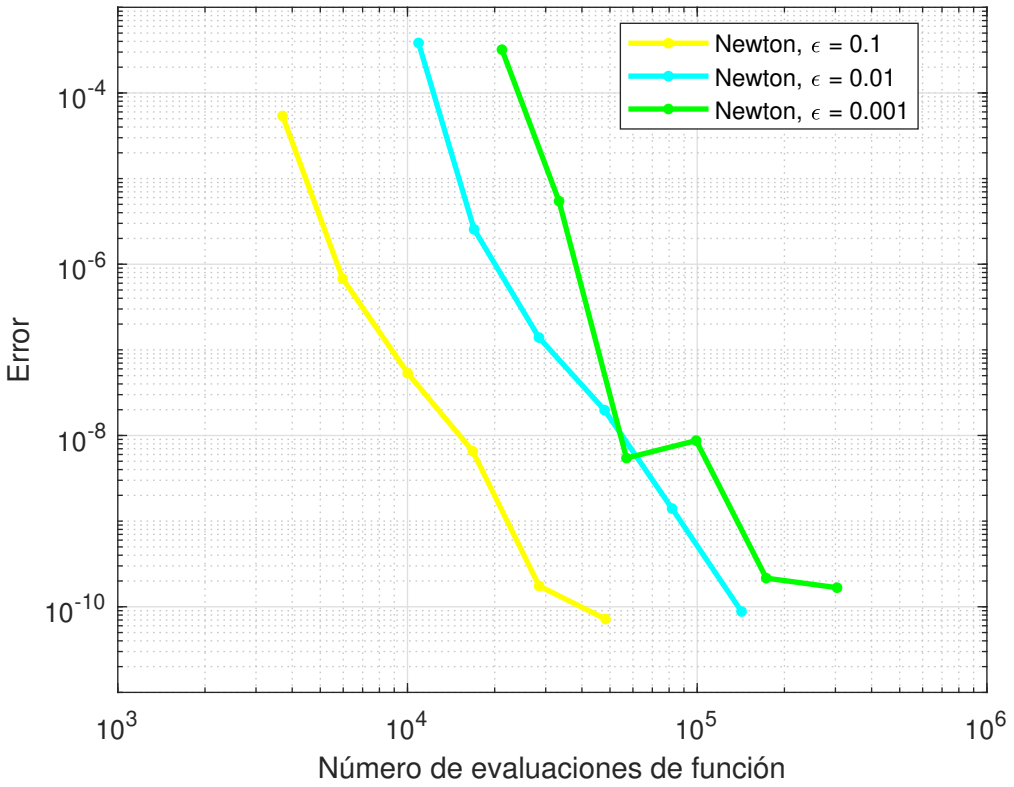


Figura 4.6: Error frente al de evaluaciones de $\mathbf{f}$ para el método de Radau IIA de orden 5 cuando se utiliza iteración de Newton con paso variable para $\epsilon = 0{,}1, 0{,}01, 0{,}001$.

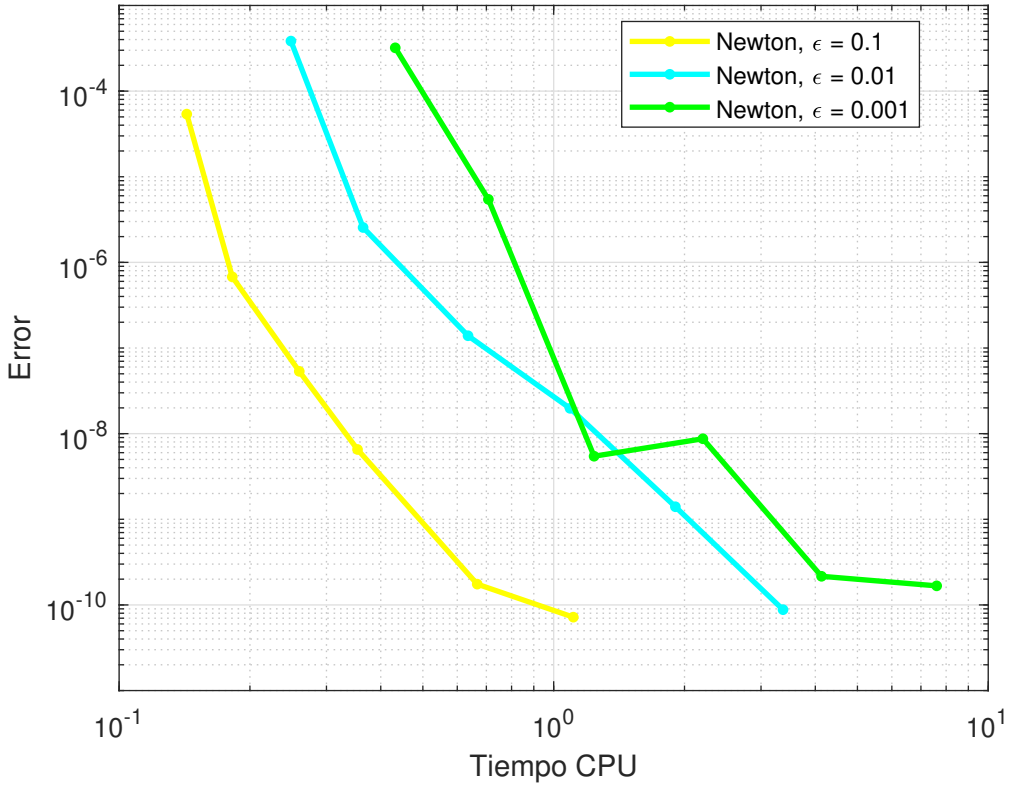


Figura 4.7: Error frente al tiempo CPU del método de Radau IIA de orden 5 cuando se utiliza iteración de Newton con paso variable para $\epsilon = 0{,}1, 0{,}01, 0{,}001$.

Bibliografía

[1] E. Hairer, S.P. Nørsett and G. Wanner, *Solving Ordinary Differential Equations I. Nonstiff Problems.* Second revised edition. Springer Series in Computational Mathematics 8. Springer-Verlag, Berlin Heidelberg GmbH (1987).

[2] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems.* Second revised edition. Springer Series in Computational Mathematics 14. Springer-Verlag, Berlin Heidelberg GmbH (1996).

[3] J.C. Butcher, *Numerical Methods for Ordinary Differential Equations.* Second edition. Wiley (2008).

[4] J.C. Butcher, *Implicit Runge-Kutta Processes.* Math. Comp. (1962), p. 50-64.

[5] J.C. Butcher, *Integration Processes Based on Radau Quadrature Formulas.* Math. Comp. (1963), p. 233-244.

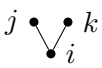
[6] E. Hairer and G. Wanner, *Stiff differential equations solved by Radau methods.* Journal of Computational and Applied Mathematics (1999), p. 93-111.

[7] E. Hairer and G. Wanner, *Radau methods.* Encyclopedia of Applied and Computational Mathematics (2011).

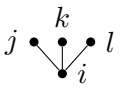
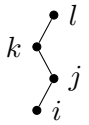
Apéndice A

Teoría de árboles

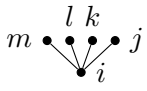
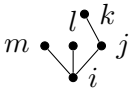
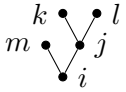
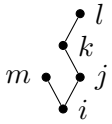
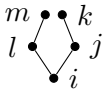
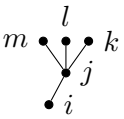
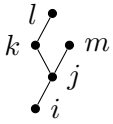
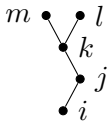
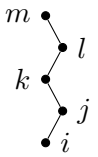
En este apéndice se muestran las tablas de condiciones de hasta orden 5 para los métodos Runge-Kutta generales, mencionadas en el Capítulo 1.

$\rho(t)$	t	árbol	$\gamma(t)$	$\Phi(t)$
1	τ		1	$\sum_{i=1}^{s} b_i$
2	t_{21}		2	$\sum_{i,j=1}^{s} b_i a_{ij} = \sum_{i=1}^{s} b_i c_i$
3	t_{31}		3	$\sum_{i,j,k=1}^{s} b_i a_{ij} a_{ik} = \sum_{i=1}^{s} b_i c_i^2$
	t_{32}		6	$\sum_{i,j,k=1}^{s} b_i a_{ij} a_{jk} = \sum_{i,j=1}^{s} b_i a_{ij} c_j$

Cuadro A.1: Condiciones de orden 1, 2 y 3.

$\rho(t)$	t	árbol	$\gamma(t)$	$\Phi(t)$
4	t_{41}		4	$\sum_{i,j,k,l=1}^{s} b_i a_{ij} a_{ik} a_{il} = \sum_{i=1}^{s} b_i c_i^3$
	t_{42}		8	$\sum_{i,j,k=1}^{s} b_i a_{ij} a_{jk} a_{il} = \sum_{i,j=1}^{s} b_i c_i a_{ij} c_j$
	t_{43}		12	$\sum_{i,j,k,l=1}^{s} b_i a_{ij} a_{jk} a_{jl} = \sum_{i,j=1}^{s} b_i a_{ij} c_j^2$
	t_{44}		24	$\sum_{i,j,k,l=1}^{s} b_i a_{ij} a_{jk} a_{kl} = \sum_{i,j,k=1}^{s} b_i a_{ij} a_{jk} c_k$

Cuadro A.2: Condiciones de orden 4.

$\rho(t)$	t	árbol	$\gamma(t)$	$\Phi(t)$
5	t_{51}		5	$\sum_{i,j,k,l,m=1}^{s} b_i a_{ij} a_{ik} a_{il} a_{im} = \sum_{i=1}^{s} b_i c_i^4$
	t_{52}		10	$\sum_{i,j,k,l,m=1}^{s} b_i a_{ij} a_{jk} a_{il} a_{im} = \sum_{i,j=1}^{s} b_i c_i^2 a_{ij} c_j$
	t_{53}		15	$\sum_{i,j,k,l,m=1}^{s} b_i a_{ij} a_{jk} a_{jl} a_{im} = \sum_{i,j=1}^{s} b_i c_i a_{ij} c_j^2$
	t_{54}		30	$\sum_{i,j,k,l,m=1}^{s} b_i a_{ij} a_{jk} a_{kl} a_{im} = \sum_{i,j,k=1}^{s} b_i c_i a_{ij} a_{jk} c_k$
	t_{55}		20	$\sum_{i,j,k,l,m=1}^{s} b_i a_{ij} a_{jk} a_{il} a_{lm} = \sum_{i,j,l=1}^{s} b_i a_{ij} c_j a_{il} c_l$
	t_{56}		20	$\sum_{i,j,k,l,m=1}^{s} b_i a_{ij} a_{jk} a_{jl} a_{jm} = \sum_{i,j=1}^{s} b_i a_{ij} c_j^3$
	t_{57}		40	$\sum_{i,j,k,l,m=1}^{s} b_i a_{ij} a_{jk} a_{kl} a_{jm} = \sum_{i,j,k=1}^{s} b_i a_{ij} c_j a_{jk} c_k$
	t_{58}		60	$\sum_{i,j,k,l,m=1}^{s} b_i a_{ij} a_{jk} a_{kl} a_{km} = \sum_{i,j,k=1}^{s} b_i a_{ij} a_{jk} c_k^2$
	t_{59}		120	$\sum_{i,j,k,l,m=1}^{s} b_i a_{ij} a_{jk} a_{kl} a_{lm} = \sum_{i,j,k,l=1}^{s} b_i a_{ij} a_{jk} a_{kl} c_l$

Cuadro A.3: Condiciones de orden 5.

Apéndice B

Polinomios de Legendre

En este apéndice se demostrarán algunas propiedades sobre los polinomios trasladados de Legendre, mencionadas en el Capítulo 2.

Recordemos que un polinomio trasladado de Legendre definido en el intervalo $[0, 1]$ se corresponde con la denominada *fórmula de Rodrigues*,

$$P_s^*(x) = \frac{1}{s!}\frac{d^s}{dx^s}\left(x^s\,(x-1)^s\right). \tag{B.1}$$

Este polinomio se obtiene tras aplicar el cambio de variable $y = 2x - 1$ al polinomio de Legendre en el intervalo $[-1, 1]$,

$$P_s(y) = \frac{1}{2^s}\frac{1}{s!}\frac{d^s}{dy^s}\left((y+1)^s\,(y-1)^s\right).$$

Esta afirmación es sencilla de comprobar:

$$\begin{aligned} P_s(2x-1) &= \frac{1}{2^s}\frac{1}{s!}\frac{d^s}{d(2x-1)^s}\left((2x-1+1)^s\,(2x-1-1)^s\right) = \frac{1}{2^s}\frac{1}{s!}\frac{d^s}{d(2x-1)^s}\left((2x)^s\,(2x-2)^s\right) \\ &= \frac{1}{s!}\frac{d^s}{dx^s}\left(x^s\,(x-1)^s\right) = P_s^*(x). \end{aligned}$$

Los polinomios de Legendre $P_s(y)$ también se pueden definir mediante la *ecuación diferencial de Legendre*,

$$\left(\left(1-y^2\right)P_s(y)'\right)' + s(s+1)P_s(y) = 0.$$

Aplicando el cambio de variable previo $y = 2x - 1$, la ecuación para los polinomios trasladados de Legendre $P_s^*(x)$ será

$$\begin{aligned} 0 &= \left((1-2x+1)(1+2x-1)P_s(2x-1)'\cdot 2\right)' + s(s+1)P_s(2x-1) \\ &= \left(2\cdot(1-x)\cdot 2x\cdot 2P_s^*(x)'\right)' + s(s+1)P_s^*(x), \end{aligned}$$

es decir, el polinomio $P_s^*(x)$ será solución de la ecuación diferencial

$$8\left((x-x^2)P_s^*(x)'\right)' + s(s+1)P_s^*(x) = 0. \tag{B.2}$$

Para demostrar algunas propiedades de los polinomios trasladados de Legendre se usará la *fórmula recursiva de Bonet*, que es la siguiente:

$$P_s^*(x) = \frac{2s-1}{s}\,(2x-1)\,P_{s-1}^*(x) - \frac{s-1}{s}P_{s-2}^*(x), \quad s = 2, 3, 4, \ldots \tag{B.3}$$

Los primeros polinomios trasladados de Legendre obtenidos son:

$$\begin{aligned}
P_0^*(x) &= 1,\\
P_1^*(x) &= 2x-1,\\
P_2^*(x) &= 6x^2-6x+1,\\
P_3^*(x) &= 20x^3-30x^2+12x-1,\\
P_4^*(x) &= 70x^4-140x^3+90x^2-20x+1,\\
P_5^*(x) &= 252x^5-630x^4+560x^3-210x^2+30x-1,\\
&\vdots \qquad \vdots
\end{aligned}$$

Veamos ahora algunas propiedades de estos polinomios.

Propiedad B.1.

$$\int_0^1 P_s^*(x)P_t^*(x)dx = 0, \quad s \neq t, \quad s,t = 0,1,2,\ldots$$

Demostración. Sean $P_s^*(x)$ y $P_t^*(x)$ dos polinomios de Legendre tales que ambos son soluciones de la ecuación diferencial (B.2), es decir,

$$8\left((x-x^2)P_s^*(x)'\right)' + s(s+1)P_s^*(x) = 0, \qquad 8\left((x-x^2)P_t^*(x)'\right)' + t(t+1)P_t^*(x) = 0.$$

Multiplicando ambas ecuaciones por $P_t^*(x)$ y $P_s^*(x)$ respectivamente, se obtiene

$$8\left((x-x^2)P_s^*(x)'\right)' P_t^*(x) + s(s+1)P_s^*(x)P_t^*(x) = 0,$$

$$8\left((x-x^2)P_t^*(x)'\right)' P_s^*(x) + t(t+1)P_t^*(x)P_s^*(x) = 0.$$

Restando la segunda ecuación de la primera, e integrando el resultado, tenemos que

$$\begin{aligned}
0 = 8\int_0^1 &\left(\left((x-x^2)P_s^*(x)'\right)' P_t^*(x) - \left((x-x^2)P_t^*(x)'\right)' P_s^*(x)\right)dx\\
&- ((s+1)-t(t+1))\int_0^1 P_s^*(x)P_t^*(x)dx.
\end{aligned}$$

Mediante integración por partes, se puede comprobar que la primera integral es cero, pues el término $x-x^2$ se anula en 0 y en 1:

$$\begin{aligned}
&\int_0^1 \left(\left((x-x^2)P_s^*(x)'\right)' P_t^*(x) - \left((x-x^2)P_t^*(x)'\right)' P_s^*(x)\right)dx =\\
&[P_t^*(x)(x-x^2)P_s^*(x)']_0^1 - [P_s^*(x)(x-x^2)P_t^*(x)']_0^1\\
&-\int_0^1 (x-x^2)P_s^*(x)'P_t^*(x)'dx + \int_0^1 (x-x^2)P_t^*(x)'P_s^*(x)'dx = 0
\end{aligned}$$

Por tanto, se obtiene que $(s(s+1)-t(t+1))\int_0^1 P_s^*(x)P_t^*(x)dx = 0$. Es claro que, si $s \neq t$, se tiene la propiedad pedida,

$$\int_0^1 P_s^*(x)P_t^*(x)dx = 0.$$

□

Propiedad B.2.

$$\int_0^1 P_s^*(x)^2 dx = \frac{1}{2s+1}, \quad s = 0,1,2,\ldots$$

Demostración. Veámoslo por inducción.

- $s = 0$:

$$\int_0^1 P_0^*(x)^2 dx = \int_0^1 1^2\, dx = 1 = \frac{1}{2\cdot 0+1}.$$

- $s = 1$:

$$\int_0^1 P_1^*(x)^2\, dx = \int_0^1 (2x-1)^2\ dx = \int_0^1 \left(4x^2-4x+1\right) dx = \frac{1}{3} = \frac{1}{2\cdot 1+1}.$$

Supongámoslo cierto para $s-1$, es decir, que se cumple $\int_0^1 P_{s-1}^*(x)^2\, dx = \frac{1}{2s-1}$.

Sustituimos la fórmula recursiva de Bonet en la integral y, usando la Propiedad B.1 para s y $s-2$, obtenemos

$$\begin{aligned}\int_0^1 P_s^*(x)^2\, dx &= \int_0^1 P_s^*(x)\cdot\left(\frac{2s-1}{s}\,(2x-1)\,P_{s-1}^*(x) - \frac{s-1}{s}P_{s-2}^*(x)\right) dx\\ &= \frac{2s-1}{s}\int_0^1 (2x-1)\,P_s^*(x)P_{s-1}^*(x)dx - \frac{s-1}{s}\int_0^1 P_s^*(x)P_{s-2}^*(x)dx\\ &= \frac{2s-1}{s}\int_0^1 (2x-1)\,P_s^*(x)P_{s-1}^*(x)dx.\end{aligned}$$

Usamos ahora la fórmula recursiva de Bonet para el índice $s+1$ en lugar de s, es decir,

$$\begin{aligned}P_{s+1}^*(x) &= \frac{2\cdot(s+1)-1}{s+1}\,(2x-1)\,P_s^*(x) - \frac{s+1-1}{s+1}P_{s-1}^*(x)\\ &= \frac{2s+1}{s+1}\,(2x-1)\,P_s^*(x) - \frac{s}{s+1}P_{s-1}^*(x), \quad s = 1,2,3,\ldots\end{aligned}$$

Despejamos el valor $(2x-1)P_s^*(x)$:

$$(2x-1)P_s^*(x) = \frac{s+1}{2s+1}\left(P_{s+1}^*(x) + \frac{s}{s+1}P_{s-1}^*(x)\right)$$

Sustituimos el valor obtenido en la integral anterior y volvemos a usar la Propiedad B.1 para los valores $s+1$ y $s-1$, usando la hipótesis de inducción para concluir:

$$\begin{aligned}\int_0^1 P_s^*(x)^2\, dx &= \frac{2s-1}{s}\int_0^1 (2x-1)\,P_s^*(x)P_{s-1}^*(x)dx\\ &= \frac{2s-1}{s}\cdot\frac{s+1}{2s+1}\int_0^1 \left(P_{s+1}^*(x) + \frac{s}{s+1}P_{s-1}^*(x)\right)P_{s-1}^*(x)dx\\ &= \frac{2s-1}{s}\cdot\frac{s+1}{2s+1}\int_0^1 P_{s+1}^*(x)P_{s-1}^*(x)dx + \frac{2s-1}{s}\cdot\frac{s+1}{2s+1}\cdot\frac{s}{s+1}\int_0^1 P_{s-1}^*(x)^2\, dx\\ &= \frac{2s-1}{2s+1}\int_0^1 P_{s-1}^*(x)^2\, dx = \frac{2s-1}{2s+1}\cdot\frac{1}{2s-1} = \frac{1}{2s+1}.\end{aligned}$$

□

Propiedad B.3. $P_s^*(1) = 1, \quad s = 0,1,2,\ldots$

Demostración. Veámoslo por inducción.

- $s = 0$: $P_0^*(x) = 1 \Rightarrow P_0^*(1) = 1$.
- $s = 1$: $P_1^*(x) = 2x-1 \Rightarrow P_1^*(1) = 2\cdot 1 - 1 = 1$.

Supongámoslo cierto para $s-1$ y $s-2$, es decir, que se cumple $P^*_{s-1}(1)=1$ y $P^*_{s-2}(x)=1$. Tomando $x=1$ en la fórmula recursiva de Bonet (B.3), se obtiene que

$$P^*_s(1)=\frac{2s-1}{s}(2\cdot 1-1)P^*_{s-1}(1)-\frac{s-1}{s}P^*_{s-2}(1)=\frac{2s-1}{s}-\frac{s-1}{s}=1.$$

Por tanto, por inducción, se tiene que $P^*_s(1)=1$, para $s=0,1,2,\ldots$ □

Propiedad B.4. $P^*_s(1-x)=(-1)^sP^*_s(x), \quad s=0,1,2,\ldots$

Demostración. Veámoslo por inducción.

- $s=0$: $P^*_0(x)=1 \Rightarrow P^*_0(1-x)=1=P^*_0(x)$.
- $s=1$: $P^*_1(x)=2x-1 \Rightarrow P^*_1(1-x)=2\cdot(1-x)-1=1-2x=-P^*_1(x)$.

Supongámoslo cierto para $s-1$ y $s-2$, es decir, que se cumple $P^*_{s-1}(1-x)=(-1)^{s-1}P^*_{s-1}(x)$ y $P^*_{s-2}(1-x)=(-1)^{s-2}P^*_{s-2}(x)$. Tomando $1-x$ en lugar de x en la fórmula recursiva de Bonet (B.3), se obtiene que

$$\begin{aligned}
P^*_s(1-x) &= \frac{2s-1}{s}\left(2\cdot(1-x)-1\right)P^*_{s-1}(1-x)-\frac{s-1}{s}P^*_{s-2}(1-x)\\
&= \frac{2s-1}{s}(1-2x)\cdot(-1)^{s-1}P^*_{s-1}(x)-\frac{s-1}{s}\cdot(-1)^{s-2}P^*_{s-2}(x)\\
&= \frac{2s-1}{s}(2x-1)\cdot(-1)^{s}P^*_{s-1}(x)-\frac{s-1}{s}\cdot(-1)^{s}P^*_{s-2}(x)\\
&= (-1)^sP^*_s(x)
\end{aligned}$$

Por tanto, por inducción, se tiene que $P^*_s(1-x)=(-1)^sP^*_s(x)$, para $s=0,1,2,\ldots$ □

Propiedad B.5. *$P^*_s(x)$ tiene s ceros reales distintos en el intervalo $(0,1)$ para $s=0,1,2,\ldots$*

Demostración. Se razonará por reducción al absurdo. Consideramos que el polinomio de Legendre $P^*_s(x)$ puede descomponerse en un producto de dos polinomios distintos $Q(x)$ y $R(x)$, donde $gr(Q(x))=m<s$ y $gr(R(x))=s-m$. Suponemos además que $R(x)$ no tiene raíces en el intervalo $(0,1)$. Entonces, se cumple que

$$\int_0^1 P^*_s(x)Q(x)dx=\int_0^1 Q(x)^2R(x)dx=0,$$

lo cual es absurdo, pues el integrando es no nulo y con signo constante. □

Apéndice C

Programas de Matlab

En este Apéndice se incluye el código de las distintas funciones de Matlab programadas para estudiar los métodos de Radau, en particular la implementación con paso variable del método de Radau IIA de 3 etapas y orden 5.

C.1. Regiones de estabilidad

Los dos programas siguientes generan las regiones de estabilidad de los métodos de Radau I y IIA de 2 etapas y orden 3, empleados en el Ejemplo 3.1. Estas dos funciones se denominan `R3estabilidad.m` y `RA3estabilidad.m`, y se han utilizado para realizar la Figura 3.1.

```
function [] = R3estabilidad()
    clf
    x = [-8:0.01:2];
    m = length(x);

    y = [-5:0.01:5]';
    n = length(y);

    z = ones(n,1)*x + sqrt(-1)*y*ones(1,m);
    R3 = abs(-3.5-0.5*z-13.5./(2*z-6));

    figure(1)
    contourf(x,y,-R3,[-1 -1],'b')
    text(-0.5,2.5,'s=2')
    axis('square')
    axis([-8 2 -5 5])
    hold on
    plot([-8 2],[0 0],'k:',[0 0],[-5 5],'k:')
    print -depsc est_R3.eps
end
```

```
function [] = RA3estabilidad()
    clf
    x = [-4:0.01:8];
    m = length(x);

    y = [-6:0.01:6]';
    n = length(y);

    z = ones(n,1)*x + sqrt(-1)*y*ones(1,m);
    RA3 = abs((1+(1/3)*z)./(1-(2/3)*z+(1/6)*z.^2));

    figure(1)
    contourf(x,y,-RA3,[-1 -1],'b')
    text(-0.5,2.5,'s=2')
```

```
    axis('square')
    axis([-4 8 -6 6])
    hold on
    plot([-4 8],[0 0],'k:',[0 0],[-6 6],'k:')
    print -depsc est_RA3.eps
end
```

C.2. Oscilador de Van der Pol

Se incluyen aquí las funciones auxiliares utilizadas para la integración del oscilador de Van der Pol.

Las funciones `VanderPol.m` y `VanderPol_jacobian.m` tienen como valores de entrada al tiempo t y al vector de posiciones $\mathbf{y}(t) = [y_1(t), y_2(t)]^T$, y devuelven el lado derecho del sistema diferencial (4.13) y su matriz Jacobiana, respectivamente. La función `VanderPol_inicial.m` no necesita ningún valor de entrada.

Como valores de salida, la función `VanderPol.m` proporcionará el lado derecho del sistema diferencial $\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}(t))$, la función `VanderPol_initial.m` proporcionará la condición inicial $\mathbf{y}(0) = \mathbf{y}_0$ y la función `VanderPol_jacobian.m` calculará el jacobiano de $\mathbf{f}$.

```
function f = VanderPol(t,y)
    global epsilon
    f = [y(2); (1/epsilon)*((1-y(1)^2)*y(2)-y(1))];
end
```

```
function y0 = VanderPol_initial()
    y0 = [2; 0];
end
```

```
function J = VanderPol_jacobian(t,y)
    global epsilon
    J = [0, 1;
        (1/epsilon)*(-2*y(1)*y(2)-1), (1/epsilon)*(1-y(1)^2)];
end
```

C.3. Métodos de Radau IIA

Se recogen aquí las funciones que proporcionan los coeficientes de los métodos de Radau IIA de órdenes 3 y 5, vistos con anterioridad en la Tabla 2.5 y la Tabla 2.6, respectivamente.

```
function [A,b,c] = RadauIIA_ord3()
    A = [5/12, -1/12; 3/4, 1/4];
    b = [3/4, 1/4];
    c = [1/3; 1];
end
```

```
function [A,b,c] = RadauIIA_ord5()
    r6 = sqrt(6);
    A = [(88-7*r6)/360, (296-169*r6)/1800, (-2+3*r6)/225; ...
         (296+169*r6)/1800, (88+7*r6)/360, (-2-3*r6)/225; ...
         (16-r6)/36, (16+r6)/36, 1/9];
    b = [(16-r6)/36, (16+r6)/36, 1/9];
    c = [(4-r6)/10; (4+r6)/10; 1];
end
```

C.4. Iteración de punto fijo

La función `FixedPoint.m` implementa con paso fijo los métodos de Radau IIA de órdenes 3 y 5, utilizando iteración de punto fijo para resolver en cada paso las ecuaciones no lineales.

Como valores de entrada tenemos a los siguientes elementos:

- t_0, t_F: tiempos inicial y final.
- y_0: vector de posiciones inicial como vector columna.
- h: longitud del paso inicial de integración.
- TOL: tolerancia para controlar el bucle en el que se realiza la iteración de punto fijo.
- $odefun$: nombre de la función que evalúa el lado derecho del sistema diferencial y lo guarda en una columna f, cuya primera línea debe ser de la forma `function f=odefun(t,y)`.
- p: orden del método considerado.
- $nmaxiter$: número máximo de iteraciones para controlar el bucle en el que se realiza la iteración de punto fijo.

Como valores de salida tenemos a los siguientes elementos:

- tt: vector columna con los tiempos en los que se obtiene aproximación a la solución.
- yy: matriz que guarda la solución numérica calculada en dichos tiempos, de manera que en cada fila están las componentes de la aproximación numérica correspondientes a un tiempo dado.
- $time$: tiempo que tarda el programa en ejecutar el bucle *while*.
- $stats$: vector columna de tres componentes.
  - $nstep$: número de pasos dados en el bucle *while*.
  - $nfun$: número de evaluaciones de función realizadas en el bucle *while*.
  - $Niter/nstep$: número de iteraciones realizadas en el bucle *while* entre el número de pasos dados en dicho bucle.

```
function [tt,yy,time,stats] = FixedPoint(t0,tF,y0,h,TOL,odefun,p,nmaxiter)
    global epsilon

    D = length(y0);
    t = t0; tt = t0;
    y = y0; yy = y0';
    nstep = 0; nfun = 0;
    Niter = 0;

    switch p
        case 3 %p=3
            [A,b,c] = RadauIIA_ord3();
        case 5 %p=5
            [A,b,c] = RadauIIA_ord5();
    end

    s = length(b);
    d = b/A;
    F = zeros(D,s);

    tic
```

```
    while t < tF
        if t+h > tF
            h = tF - t;
        end

        nstep = nstep + 1;
        Zaux = ones(D,s);
        Z = zeros(D,s);
        niter = 0;

        while (max(max(abs(Z-Zaux)))>TOL && niter<=nmaxiter)
            for j = 1:s
                F(:,j) = odefun(t+c(j)*h,y+Z(:,j));
                nfun = nfun + 1;
            end

            Zaux = Z;
            niter = niter + 1;

            for j = 1:s
                Z(:,j) = h*(A(j,:)*F')';
            end
        end

        Niter = Niter + niter;

        y = y + Z*d';
        t = t + h;

        yy = [yy; y'];
        tt = [tt; t];
    end

    time = toc;
    stats = [nstep; nfun; Niter/nstep];
end
```

C.5. Iteración de Newton

La función `Newton.m` implementa con paso fijo los métodos de Radau IIA de órdenes 3 y 5, utilizando iteración de Newton para resolver en cada paso las ecuaciones no lineales.

Los valores de entrada son los mismos que en `FixedPoint.m`, junto con *jacobian*, función que calculará el jacobiano del sistema diferencial a integrar. Los valores de salida se mantienen iguales a los de la función `FixedPoint.m`.

```
function [tt,yy,time,stats] = Newton(t0,tF,y0,h,TOL,odefun,jacobian,p,nmaxiter)
    global epsilon

    D = length(y0);
    t = t0; tt = t0;
    y = y0; yy = y0';
    nstep = 0; nfun = 0;
    Niter = 0;

    switch p
        case 3 %p=3
            [A,b,c] = RadauIIA_ord3();
        case 5 %p=5
            [A,b,c] = RadauIIA_ord5();
    end
```

```
    s = length(b);
    d = b/A;
    F = zeros(D*s,1);
    Faux = zeros(D*s,1);

    tic

    while t < tF
        if t+h > tF
            h = tF - t;
        end

        nstep = nstep + 1;
        Z = zeros(D*s,1);
        DeltaZ = TOL + 1;
        niter = 0;

        Id = eye(D*s);
        AtimesJ = kron(A,jacobian(t,y));
        DPhi = Id - h*AtimesJ;
        [L,U,P] = lu(DPhi);

        while (max(abs(DeltaZ))>TOL && niter<=nmaxiter)
            for i = 1:s
                F(D*(i-1)+1:D*i,1) = odefun(t+c(i)*h,y+Z(D*(i-1)+1:D*i,1));
                nfun = nfun + 1;
            end

            AF = zeros(D*s,1);
            niter = niter + 1;

            for i = 1:s
                for j = 1:s
                    AF(D*(i-1)+1:D*i) = AF(D*(i-1)+1:D*i) + A(i,j)*F(D*(j-1)+1:D*j);
                end
                Faux(D*(i-1)+1:D*i) = Z(D*(i-1)+1:D*i) - h*AF(D*(i-1)+1:D*i);
            end

            DeltaZ = U\(L\(P*Faux));
            Z = Z - DeltaZ;
        end

        Niter = Niter + niter;

        Z = reshape(Z,[D,s]);
        y = y + Z*d';
        t = t + h;

        yy = [yy; y'];
        tt = [tt; t];
    end

    time = toc;
    stats = [nstep; nfun; Niter/nstep];
end
```

C.6. Iteración de Newton con paso variable

La función `Newton_var.m` implementa con paso variable el método de Radau IIA de 3 etapas y orden 5, utilizando iteración de Newton para resolver en cada paso las ecuaciones no lineales.

Tomamos como valores de entrada los mismos que para la función `Newton.m`. Como valores de salida, además de los que se generan en las funciones anteriores se genera también un vector *hh* que

contiene las longitudes de paso utilizadas en la integración, un vector *nnit* que contiene el número de iteraciones dadas en cada paso aceptado, y exceptuando el vector *stats*, que tiene dos entradas más:

- *accept*: número de pasos que han cumplido que $nerrn < TOL$.
- *reject*: número de pasos que no han cumplido dicha condición.

```
function [tt,yy,hh,nnit,time,stats] = Newton_var(t0,tF,y0,h,TOL,odefun,jacobian,p,nmaxiter)
    global epsilon
    % nmaxiter = 5 o 10

    D = length(y0);
    t = t0; tt = t0;
    y = y0; yy = y0';
    hh = h; nnit = [];
    nstep = 0; nfun = 0;
    Niter = 0;
    accept = 0; reject = 0;
    TOLN = TOL/100;

    [A,b,c] = RadauIIA_ord5(); % p=5
    s = length(b); % s=3
    d = b/A;
    F = zeros(D*s,1);
    Faux = zeros(D*s,1);

    gamma = 3.637834252744497; % autovalor real de la matriz A^{-1}
    e = (1/(3*gamma))*[-13-7*sqrt(6) -13+7*sqrt(6) -1];

    tic

    while t < tF
        if t+h > tF
            h = tF - t;
        end

        nstep = nstep + 1;
        Z = zeros(D*s,1);
        DeltaZ = TOLN + 1;
        niter = 0;

        Id = eye(D*s);
        J = jacobian(t,y);
        AtimesJ = kron(A,J);
        DPhi = Id - h*AtimesJ;
        [L,U,P] = lu(DPhi);

        while (max(abs(DeltaZ))>TOLN && niter<=nmaxiter)
            for i = 1:s
                F(D*(i-1)+1:D*i,1) = odefun(t+c(i)*h,y+Z(D*(i-1)+1:D*i,1));
                nfun = nfun + 1;
            end

            AF = zeros(D*s,1);
            niter = niter + 1;

            for i = 1:s
                for j = 1:s
                    AF(D*(i-1)+1:D*i) = AF(D*(i-1)+1:D*i) + A(i,j)*F(D*(j-1)+1:D*j);
                end
                Faux(D*(i-1)+1:D*i) = Z(D*(i-1)+1:D*i) - h*AF(D*(i-1)+1:D*i);
            end

            DeltaZ = U\(L\(P*Faux));
```

```
            Z = Z - DeltaZ;
        end

        Niter = Niter + niter;

        Z = reshape(Z,[D,s]);
        dif = (1/gamma)*h*odefun(t,y) + Z*e';
        errn = (eye(2)-h*(1/gamma)*J)\dif;

        fac = 0.9*(2*nmaxiter+1)/(2*nmaxiter+niter);
        nerrn = norm(errn);
        haux = fac*h*(nerrn/TOL)^(-1/4);

        if nerrn<TOL
            % aceptamos el paso, calculamos la siguiente aproximacion del
            % metodo y el nuevo paso
            y = y + Z*d';
            t = t + h;

            yy = [yy; y'];
            tt = [tt; t];
            hh = [hh; h];
            nnit=[nnit; niter];  %mp

            if (haux<h || haux>1.2*h)
                h = haux;
            end
            accept = accept + 1;
        else % nerrn>=TOL
            % no aceptamos el paso, calculamos el nuevo
            h = haux;
            reject = reject + 1;
        end
    end

    time = toc;
    stats = [nstep; nfun; Niter/nstep; accept; reject];
end
```

C.7. Gráficas

La función `graphics.m` llama a las funciones previas y genera las siguientes gráficas:

- La primera gráfica representa la evolución con el tiempo de la componente $y_1(t)$ del oscilador de Van der Pol para $\epsilon = 1$, donde el eje de abscisas se corresponde con el tiempo t y el eje de ordenadas con la componente $y_1(t)$.
- De forma similar, la segunda gráfica representa la evolución con el tiempo de la componente $y_2(t)$ para $\epsilon = 1$.
- La tercera gráfica muestra el error frente al número de evaluaciones de función realizadas según el método de Radau IIA y el proceso iterativo elegidos.
- La cuarta gráfica muestra el error frente al tiempo CPU de dichos métodos.

```
function [] = graphics()
    global epsilon
    % epsilon = 1;

    y0 = VanderPol_initial();
    t0 = 0; tF = 11; h = 0.25;
    % nmaxiter = 50 para paso fijo
```

```
m = 6; H = zeros(m,1);
tol3 = zeros(m,1); TOL3 = (h^3)/10;
tol5 = zeros(m,1); TOL5 = (h^5)/10;

options = odeset('RelTol',2.3e-14,'AbsTol',2.3e-14,'Jacobian',@VanderPol_jacobian);
[texact,yexact] = ode15s(@VanderPol,[t0,tF],y0,options);
nexact = length(texact); exact = yexact(nexact,:);

figure(1)
clf
plot(texact,yexact(:,1),'m-')
xlim([t0 tF])
xlabel('Tiempo')
ylabel('y_1(t)')
legend('y_1(t)','Location','best');
print -depsc realsolution1.eps

figure(2)
clf
plot(texact,yexact(:,2),'c-')
xlim([t0 tF])
xlabel('Tiempo')
ylabel('y_2(t)')
legend('y_2(t)','Location','best');
print -depsc realsolution2.eps

time3fp = zeros(m,1); err3fp = zeros(m,1); nfun3fp = zeros(m,1);
time3n = zeros(m,1); err3n = zeros(m,1); nfun3n = zeros(m,1);
time5fp = zeros(m-1,1); err5fp = zeros(m-1,1); nfun5fp = zeros(m-1,1);
time5n = zeros(m-1,1); err5n = zeros(m-1,1); nfun5n = zeros(m-1,1);

for i = 1:m-1
    [tt,yy,time,stats] = FixedPoint(t0,tF,y0,h,TOL3,@VanderPol,3,50);
    tt3fp = tt; yy3fp = yy;
    time3fp(i) = time; nfun3fp(i) = stats(2);
    n3fp = length(tt3fp); err3fp(i) = norm(yy3fp(n3fp,:)-exact);

    [tt,yy,time,stats] = Newton(t0,tF,y0,h,TOL3,@VanderPol,@VanderPol_jacobian,3,50);
    tt3n = tt; yy3n = yy;
    time3n(i) = time; nfun3n(i) = stats(2);
    n3n = length(tt3n); err3n(i) = norm(yy3n(n3n,:)-exact);

    [tt,yy,time,stats] = FixedPoint(t0,tF,y0,h,TOL5,@VanderPol,5,50);
    tt5fp = tt; yy5fp = yy;
    time5fp(i) = time; nfun5fp(i) = stats(2);
    n5fp = length(tt5fp); err5fp(i) = norm(yy5fp(n5fp,:)-exact);

    stats

    [tt,yy,time,stats] = Newton(t0,tF,y0,h,TOL5,@VanderPol,@VanderPol_jacobian,5,50);
    tt5n = tt; yy5n = yy;
    time5n(i) = time; nfun5n(i) = stats(2);
    n5n = length(tt5n); err5n(i) = norm(yy5n(n5n,:)-exact);

    stats

    tol3(i) = TOL3; TOL3 = TOL3/(2^3);
    tol5(i) = TOL5; TOL5 = TOL5/(2^5);
    H(i) = h; h = h/2.0;
end

% Damos otro paso con el metodo de orden 3
i=i+1;
[tt,yy,time,stats] = FixedPoint(t0,tF,y0,h,TOL3,@VanderPol,3,50);
tt3fp = tt; yy3fp = yy;
time3fp(i) = time; nfun3fp(i) = stats(2);
```

```
    n3fp = length(tt3fp); err3fp(i) = norm(yy3fp(n3fp,:)-exact);

    [tt,yy,time,stats] = Newton(t0,tF,y0,h,TOL3,@VanderPol,@VanderPol_jacobian,3,50);
    tt3n = tt; yy3n = yy;
    time3n(i) = time; nfun3n(i) = stats(2);
    n3n = length(tt3n); err3n(i) = norm(yy3n(n3n,:)-exact);

    figure(3)
    clf
    loglog(nfun3fp,err3fp,'g.--','MarkerSize',12,'LineWidth',2);
    hold on
    loglog(nfun3n,err3n,'m.-','MarkerSize',12,'LineWidth',2);
    hold on
    loglog(nfun5fp,err5fp,'b.--','MarkerSize',12,'LineWidth',2);
    hold on
    loglog(nfun5n,err5n,'c.-','MarkerSize',12,'LineWidth',2);
    hold off
    grid on
    xlim([100 100000])
    ylim([1e-11 1e-2])
    xlabel('Numero de evaluaciones de funcion')
    ylabel('Error')
    legend('Punto fijo, orden 3','Newton, orden 3','Punto fijo, orden 5', ...
        'Newton, orden 5','Location','best');
    print -depsc err_neval.eps

    figure(4)
    clf
    loglog(time3fp,err3fp,'g.--','MarkerSize',12,'LineWidth',2);
    hold on
    loglog(time3n,err3n,'m.-','MarkerSize',12,'LineWidth',2);
    hold on
    loglog(time5fp,err5fp,'b.--','MarkerSize',12,'LineWidth',2);
    hold on
    loglog(time5n,err5n,'c.-','MarkerSize',12,'LineWidth',2);
    hold off
    grid on
    xlim([1e-3 1])
    ylim([1e-11 1e-2])
    xlabel('Tiempo CPU')
    ylabel('Error')
    legend('Punto fijo, orden 3','Newton, orden 3','Punto fijo, orden 5', ...
        'Newton, orden 5','Location','best');
    print -depsc err_time.eps
end
```

La función `graphics_var.m` llama a las funciones previas y representa:

- La longitud de paso frente al tiempo, donde el eje de abscisas se corresponde con el tiempo t y el eje de ordenadas con el vector de longitudes de paso h.
- El número de iteraciones de cada paso aceptado en la función `Newton_var.m` frente al tiempo, donde el eje de abscisas se corresponde con el tiempo t y el eje de ordenadas con el vector de iteraciones $nnit$.

Además, proporciona como valores de salida al tiempo CPU $time5nvar$, a número de evaluaciones de función $nfun5nvar$ y al error $err5nvar$.

```
function [time5nvar,nfun5nvar,err5nvar] = graphics_var()
    global epsilon
    % epsilon < 1;

    y0 = VanderPol_initial();
    t0 = 0; tF = 11; h = 0.01;
    % nmaxiter = 5 o 10 para iteracion de Newton paso variable
```

```
m = 6; H = zeros(m,1);
tol = zeros(m,1); TOL = 1.0e-3;

options = odeset('RelTol',2.3e-14,'AbsTol',2.3e-14,'Jacobian',@VanderPol_jacobian);
[texact,yexact] = ode15s(@VanderPol,[t0,tF],y0,options);
nexact = length(texact); exact = yexact(nexact,:);

time5nvar = zeros(m,1); err5nvar = zeros(m,1); nfun5nvar = zeros(m,1);

for i = 1:m
    [tt,yy,hh,nnit,time,stats] = Newton_var(t0,tF,y0,h,TOL,@VanderPol, ...
        @VanderPol_jacobian,5,5);
    if i==2
        tt5nvar = tt;
        hh5nvar = hh;
        nnit5nvar = nnit;
    end
    time5nvar(i) = time; nfun5nvar(i) = stats(2); yy5nvar = yy;
    n5nvar = length(tt); err5nvar(i) = norm(yy5nvar(n5nvar,:)-exact);

    stats

    if epsilon == 0.1
        figure(1)
    elseif epsilon == 0.01
        figure(2)
    elseif epsilon == 0.001
        figure(3)
    end
    semilogy(tt,hh)
    grid on
    xlim([t0 tF])
    xlabel('Tiempo')
    ylabel('Longitud de paso')
    legend('h(t)','Location','best');
    if i==2
        if epsilon == 0.1
            print -depsc tthh_01.eps
        elseif epsilon == 0.001
            print -depsc tthh_0001.eps
        end
    end

    tol(i) = TOL; TOL = TOL/10;
    H(i) = h; h = h/(10^(1/5));
end

if epsilon == 0.01
    subplot(2,1,1)
    semilogy(tt5nvar,hh5nvar)
    grid on
    xlim([0 3])
    xlabel('Tiempo')
    ylabel('Longitud de paso')
    legend('h(t)','Location','best');

    subplot(2,1,2)
    plot(tt5nvar(1:end-1),nnit5nvar,'.-')
    grid on
    xlim([0 3])
    ylim([2 8])
    xlabel('Tiempo')
    ylabel('Numero de iteraciones')
    legend('Num.iter.','Location','best');

    print -depsc tthh_001.eps
```

```
    end
end
```

La función `main_graphics.m` llama a las funciones `graphics.m` y `graphics_var.m`, y además genera las siguientes gráficas:

- La primera gráfica representa la evolución con el tiempo de la componente $y_1(t)$ del oscilador de Van der Pol para $\epsilon = 0{,}1$ y $\epsilon = 0{,}01$, donde el eje de abscisas se corresponde con el tiempo t y el eje de ordenadas con la componente $y_1(t)$.
- De forma similar, la segunda gráfica representa la evolución con el tiempo de la componente $y_2(t)$ para $\epsilon = 0{,}1$ y $\epsilon = 0{,}01$.
- La tercera gráfica muestra el error frente al número de evaluaciones de función realizadas según el método de Radau IIA y el proceso iterativo elegidos.
- La cuarta gráfica muestra el error frente al tiempo CPU de dichos métodos.

```
function [] = main_graphics()
   global epsilon
   t0 = 0; tF = 11;

   if epsilon==1
       graphics();
   else
       epsilon = 0.1;
       [texact1,yexact1,time1,nfun1,err1] = graphics_var();

       epsilon = 0.01;
       [texact2,yexact2,time2,nfun2,err2] = graphics_var();

       epsilon = 0.001;
       [texact3,yexact3,time3,nfun3,err3] = graphics_var();

       figure(4)
       clf
       plot(texact1,yexact1(:,1),'m-')
       hold on
       plot(texact2,yexact2(:,1),'b-')
       hold off
       xlim([t0 tF])
       xlabel('Tiempo')
       ylabel('y_1(t)')
       legend('y_1(t), \epsilon = 0.1','y_1(t), \epsilon = 0.01','Location', ...
           'northeast');
       print -depsc realsolution1_var.eps

       figure(5)
       clf
       plot(texact1,yexact1(:,2),'m-')
       hold on
       plot(texact2,yexact2(:,2),'b-')
       hold off
       xlim([t0 tF])
       xlabel('Tiempo')
       ylabel('y_2(t)')
       legend('y_2(t), \epsilon = 0.1','y_2(t), \epsilon = 0.01','Location', ...
           'northeast');
       print -depsc realsolution2_var.eps

       figure(6)
       clf
       loglog(nfun1,err1,'y.-','MarkerSize',12,'LineWidth',2);
       hold on
```

```
        loglog(nfun2,err2,'c.-','MarkerSize',12,'LineWidth',2);
        hold on
        loglog(nfun3,err3,'g.-','MarkerSize',12,'LineWidth',2);
        hold off
        grid on
        xlim([1000 1000000])
        ylim([1e-11 1e-3])
        xlabel('Numero de evaluaciones de funcion')
        ylabel('Error')
        legend('Newton, \epsilon = 0.1','Newton, \epsilon = 0.01', ...
            'Newton, \epsilon = 0.001','Location','best');
        print -depsc err_neval_var.eps

        figure(7)
        clf
        loglog(time1,err1,'y.-','MarkerSize',12,'LineWidth',2);
        hold on
        loglog(time2,err2,'c.-','MarkerSize',12,'LineWidth',2);
        hold on
        loglog(time3,err3,'g.-','MarkerSize',12,'LineWidth',2);
        hold off
        grid on
        xlim([1e-1 10])
        ylim([1e-11 1e-3])
        xlabel('Tiempo CPU')
        ylabel('Error')
        legend('Newton, \epsilon = 0.1','Newton, \epsilon = 0.01', ...
            'Newton, \epsilon = 0.001','Location','best');
        print -depsc err_time_var.eps
    end
end
```