



Universidad de Valladolid

FACULTAD DE CIENCIAS

TRABAJO FIN DE GRADO

Grado en Matemáticas

INTEGRADORES EXPONENCIALES PARA ECUACIONES SEMILINEALES

Autora: Clara Georgina Stampa Guilarte

Tutora: María Paz Calvo Cabrero

Junio 2023

*A Alfredo y Patricia, mis padres, por darme la oportunidad, el apoyo y la confianza
para alcanzar mis metas.
A Carlos, mi salvavidas, por recordarme cada día el significado de la amistad.*

Índice general

Lista de figuras	7
Lista de tablas	9
Introducción	11
1. Métodos Runge-Kutta exponenciales explícitos	13
1.1. Métodos Runge-Kutta explícitos	13
1.2. Cuadratura exponencial: problemas lineales	14
1.3. Caso semilineal	21
1.3.1. Condiciones de orden	24
1.3.2. Ejemplos de métodos Runge Kutta exponenciales explícitos	34
2. Implementación y resultados numéricos	41
2.1. Aspectos prácticos de implementación	41
2.2. Un ejemplo lineal	45
2.3. Primer ejemplo semilineal	47
2.4. Integración de una EDP semilineal	49
3. Conclusiones y posibles aplicaciones	55
Bibliografía	59
A. Programas en MATLAB	61
A.1. Programa en MATLAB para el ejemplo lineal	61
A.2. Programa principal para el primer ejemplo semilineal	65
A.3. Programa principal para el segundo ejemplo semilineal	70
A.4. Funciones en MATLAB para los métodos RK exponenciales	73
A.4.1. Método de Euler exponencial (orden 1)	73
A.4.2. Método de orden 2 (I)	74
A.4.3. Método de orden 2 (II)	75
A.4.4. Método de orden 3 (I)	76
A.4.5. Método de orden 3 (II)	77
A.4.6. Método de orden 4	78

Índice de figuras

2.1. Izquierda: fase transitoria de la solución exacta (2.4). Derecha: fase “estacionaria” de la solución exacta (2.4).	45
2.2. Error absoluto en el tiempo $t = \frac{\pi}{2}$ al aproximar la solución del problema (2.3) con $y_0 = 1$ y $c = 100$, en función de la longitud de paso h	46
2.3. Error absoluto en el tiempo $t = 1$ al aproximar la solución del problema (2.8)-(2.9) con $\mathbf{y}_0 = (2, 1)$, $c = 100$ y $\lambda = \frac{1}{2}$, en función de la longitud de paso h	48
2.4. Izquierda: error absoluto en el tiempo $t = 1$ al aproximar la amplitud r del problema (2.10)-(2.11), en función de la longitud de paso h . Derecha: error absoluto en el tiempo $t = 1$ al aproximar la fase θ del problema (2.10)-(2.11), en función de la longitud de paso h	49
2.5. Error absoluto en el tiempo $t = 1$ al aproximar la solución del problema (2.14) discretizado espacialmente con $J = 512$, en función de la longitud de paso h	51
2.6. Error absoluto en el tiempo $t = 1$ al aproximar la solución del problema (2.14) discretizado espacialmente con $J = 512$, en función del tiempo de CPU empleado.	52
2.7. Izquierda: Error absoluto en el tiempo $t = 1$ al aproximar la solución del problema (2.14) discretizado espacialmente con $J = 64$, en función de la longitud de paso h . Derecha: Error absoluto en el tiempo $t = 1$ al aproximar la solución del problema (2.14) discretizado espacialmente con $J = 64$, en función del tiempo de CPU empleado.	53
2.8. Izquierda: Error absoluto en el tiempo $t = 1$ al aproximar la solución del problema (2.14) discretizado espacialmente con $J = 64$, en función de la longitud de paso h . Derecha: Error absoluto en el tiempo $t = 1$ al aproximar la solución del problema (2.14) discretizado espacialmente con $J = 64$, en función del tiempo de CPU empleado.	54

Índice de tablas

1.1. Condiciones de orden no rígidas y árboles de los métodos Runge-Kutta exponenciales.	32
1.2. Condiciones de orden rígidas de los métodos Runge-Kutta exponenciales explícitos.	33

Introducción

En este trabajo, se aborda el estudio de los métodos Runge-Kutta exponenciales explícitos para la integración temporal de problemas semilineales. Es común que estos problemas provengan de la semidiscretización espacial de ecuaciones en derivadas parciales de evolución, como parte de su proceso de resolución utilizando el método de líneas.

El primer capítulo se divide en tres secciones. En la primera, se hace una revisión de los métodos Runge-Kutta explícitos que incluye algunos ejemplos clásicos de éstos.

En la segunda sección, se analizan los métodos exponenciales aplicados a la integración numérica de sistemas diferenciales lineales no homogéneos, junto con el estudio de las condiciones de orden de los mismos. Además, dentro de la misma sección, se muestran ejemplos de métodos de cuadratura exponencial que más tarde serán utilizados en la integración numérica de un problema concreto del segundo capítulo.

La última sección del primer capítulo presenta los métodos Runge-Kutta exponenciales explícitos como herramienta de integración temporal de sistemas diferenciales semilineales. A continuación, se desarrolla la teoría de árboles con el fin de hallar las condiciones de orden no rígidas que deben satisfacer los métodos y se resumen las condiciones de orden rígidas. El capítulo finaliza con una breve muestra de varias familias de métodos Runge-Kutta exponenciales explícitos de distintos órdenes que pueden encontrarse en la literatura y que serán aplicados posteriormente sobre dos de los ejemplos que aparecen en el Capítulo 2.

El segundo capítulo está dedicado a la implementación práctica de los métodos estudiados. Comienza con una primera sección que incluye una breve revisión de los fundamentos teóricos que hay detrás del funcionamiento de la función `phipm.m` para su posterior utilización en los programas de MATLAB que implementan cada método. A continuación, el capítulo tiene otras tres secciones con un ejemplo numérico en cada una de ellas. Cada sección comprende el estudio del ejemplo en cuestión junto con la visualización gráfica de los errores cometidos por los métodos implementados. Esto permite ilustrar el orden de los métodos que ya se había obtenido teóricamente en el capítulo anterior.

La segunda sección del capítulo está dedicada a un ejemplo lineal no homogéneo, de modo que los métodos utilizados para su integración son los que se encuentran en la segunda sección del Capítulo 1. En la tercera sección, se analiza un primer ejemplo semilineal para un sistema formado únicamente por dos ecuaciones diferenciales ordinarias y los métodos implementados son los que se encuentran en la Sección 1.3.2. Por último, la

cuarta sección aborda la integración temporal del problema semilineal resultante de la discretización espacial de la ecuación de Burgers mediante diferencias finitas.

El tercer capítulo concluye el trabajo haciendo una breve mención a distintos modelos matemáticos que describen la evolución de un sistema físico y a los que se pueden aplicar los métodos estudiados a lo largo del trabajo.

En el apéndice se incluyen los programas implementados en MATLAB que se han utilizado para la integración de los problemas planteados en el Capítulo 2.

Capítulo 1

Métodos Runge-Kutta exponenciales explícitos

1.1. Métodos Runge-Kutta explícitos

Empezamos por una breve revisión de los métodos Runge-Kutta explícitos clásicos. Los métodos Runge-Kutta constituyen una familia de métodos numéricos de un único paso para aproximar la solución de problemas de valores iniciales de la forma

$$\mathbf{y}'(t) = \mathbf{F}(t, \mathbf{y}(t)), \quad 0 \leq t \leq T, \quad \text{con } \mathbf{y}(0) = \mathbf{y}_0, \quad (1.1)$$

donde $\mathbf{y}_0 \in \mathbb{R}^d$ y $\mathbf{F} : [0, T] \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ es una función conocida que supondremos suficientemente regular.

Su implementación consiste en que, dada una aproximación \mathbf{y}_n a la solución de (1.1) en un tiempo t_n , generamos una aproximación \mathbf{y}_{n+1} a la solución de (1.1) en $t_{n+1} = t_n + h$ mediante una combinación lineal de evaluaciones de \mathbf{F} en varias aproximaciones a la solución de (1.1) en diferentes puntos del subintervalo $[t_n, t_n + h]$ y repetimos el proceso dando tantos pasos de longitud h como sean necesarios hasta llegar a T (es posible que haya que adaptar la longitud del último paso si h no es un divisor entero de T).

La formulación general de un método Runge-Kutta de s etapas para aproximar la solución \mathbf{y}_{n+1} de (1.1) en un tiempo $t_{n+1} = t_n + h$, conocida una aproximación \mathbf{y}_n de ésta en un tiempo t_n , es la siguiente

$$\begin{aligned} \mathbf{k}_i &= \mathbf{F} \left(t_n + c_i h, \mathbf{y}_n + h \sum_{j=1}^s a_{ij} \mathbf{k}_j \right), \quad 1 \leq i \leq s, \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + h \sum_{i=1}^s b_i \mathbf{k}_i, \end{aligned}$$

donde los coeficientes a_{ij} y b_i , $1 \leq i, j \leq s$, definen el método y

$$c_i = \sum_{j=1}^s a_{ij} \quad \text{para } 1 \leq i \leq s. \quad (1.2)$$

Como formulación alternativa, podemos escribir

$$\begin{aligned} \mathbf{Y}_i &= \mathbf{y}_n + h \sum_{j=1}^s a_{ij} \mathbf{F}(t_n + c_j h, \mathbf{Y}_j), \quad 1 \leq i \leq s, \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + h \sum_{i=1}^s b_i \mathbf{F}(t_n + c_i h, \mathbf{Y}_i). \end{aligned}$$

En general, en las expresiones anteriores, cada \mathbf{k}_i depende de $\mathbf{k}_1, \dots, \mathbf{k}_s$ y avanzar un paso de longitud h con el método requiere la resolución de un sistema de ecuaciones no lineales. Sin embargo, si la matriz $\mathcal{A} = (a_{ij})$ es estrictamente triangular inferior, el método es explícito pues $\mathbf{k}_1 = \mathbf{F}(t_n, \mathbf{y}_n)$ y, para $i \geq 2$, \mathbf{k}_i sólo depende de $\mathbf{k}_1, \dots, \mathbf{k}_{i-1}$.

Algunos ejemplos clásicos de métodos Runge-Kutta explícitos a los que nos referiremos más adelante en este trabajo son los siguientes

- Método de Euler explícito, de orden 1 y con $s = 1$,

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{F}(t_n, \mathbf{y}_n), \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + h\mathbf{k}_1. \end{aligned}$$

- Método de Euler modificado, de orden 2 y con $s = 2$,

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{F}(t_n, \mathbf{y}_n), \\ \mathbf{k}_2 &= \mathbf{F}(t_n + h, \mathbf{y}_n + h\mathbf{k}_1), \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + \frac{h}{2}(\mathbf{k}_1 + \mathbf{k}_2). \end{aligned}$$

- Método Runge-Kutta de orden 4, con $s = 4$,

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{F}(t_n, \mathbf{y}_n), \\ \mathbf{k}_2 &= \mathbf{F}(t_n + \frac{h}{2}, \mathbf{y}_n + \frac{h}{2}\mathbf{k}_1), \\ \mathbf{k}_3 &= \mathbf{F}(t_n + \frac{h}{2}, \mathbf{y}_n + \frac{h}{2}\mathbf{k}_2), \\ \mathbf{k}_4 &= \mathbf{F}(t_n + h, \mathbf{y}_n + h\mathbf{k}_3), \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + \frac{h}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4). \end{aligned}$$

1.2. Cuadratura exponencial: problemas lineales

Vamos a comenzar estudiando la cuadratura exponencial para el caso de problemas diferenciales lineales de la forma

$$\mathbf{y}'(t) + A\mathbf{y}(t) = \mathbf{F}(t), \quad 0 \leq t \leq T, \quad \text{con } \mathbf{y}(0) = \mathbf{y}_0, \quad (1.3)$$

siendo A una matriz real constante de tamaño $d \times d$ y $\mathbf{F} : [0, T] \rightarrow \mathbb{R}^d$ un término fuente dado. Concretamente, vamos a analizar las soluciones aproximadas que pueden obtenerse

mediante los llamados *métodos exponenciales*, con el fin de establecer las condiciones de orden. Para esta sección, fundamentalmente se ha seguido el mismo esquema que en [8].

Sabemos que, conocida la solución exacta de (1.3) en t_n , $n \geq 0$, la solución exacta de (1.3) en $t_{n+1} = t_n + h$ viene dada por la fórmula de variación de las constantes.

$$\mathbf{y}(t_{n+1}) = e^{-hA}\mathbf{y}(t_n) + \int_{t_n}^{t_{n+1}} e^{-(t_n+h-s)A}\mathbf{F}(s)ds.$$

Hacemos el cambio de variable $s = t_n + h\theta$ en la integral y obtenemos

$$\mathbf{y}(t_{n+1}) = e^{-hA}\mathbf{y}(t_n) + h \int_0^1 e^{-h(1-\theta)A}\mathbf{F}(t_n + h\theta)d\theta. \quad (1.4)$$

A continuación, aproximamos $\mathbf{F}(t_n + h\theta)$, visto como función de θ , por su polinomio interpolador de Lagrange en nodos c_1, \dots, c_s contenidos en el intervalo $[0, 1]$. Introduciendo los polinomios de la base de Lagrange

$$l_i(\theta) = \prod_{m=1, m \neq i}^s \frac{\theta - c_m}{c_i - c_m}, \quad 1 \leq i \leq s, \quad (1.5)$$

obtenemos el polinomio buscado

$$\mathbf{P}(t_n + h\theta) = \sum_{i=1}^s \mathbf{F}(t_n + c_i h) l_i(\theta). \quad (1.6)$$

Ahora, si \mathbf{y}_n es una aproximación conocida de $\mathbf{y}(t_n)$, podemos obtener la regla de cuadratura exponencial sustituyendo en la integral de (1.4) la función \mathbf{F} por el polinomio interpolador \mathbf{P} dado por (1.6)

$$\begin{aligned} \mathbf{y}_{n+1} &= e^{-hA}\mathbf{y}_n + h \int_0^1 e^{-h(1-\theta)A} \left[\sum_{i=1}^s \mathbf{F}(t_n + c_i h) l_i(\theta) \right] d\theta \\ &= e^{-hA}\mathbf{y}_n + h \sum_{i=1}^s \left[\int_0^1 e^{-h(1-\theta)A} l_i(\theta) d\theta \right] \mathbf{F}(t_n + c_i h), \end{aligned}$$

que lleva a la regla de cuadratura exponencial

$$\mathbf{y}_{n+1} = e^{-hA}\mathbf{y}_n + h \sum_{i=1}^s b_i(-hA)\mathbf{F}(t_n + c_i h), \quad (1.7)$$

donde

$$b_i(-hA) = \int_0^1 e^{-h(1-\theta)A} l_i(\theta) d\theta, \quad i = 1, \dots, s. \quad (1.8)$$

Notemos que la regla de cuadratura obtenida tiene como pesos integrales de productos de exponenciales matriciales por los polinomios de la base de Lagrange (1.5), en lugar de coeficientes constantes como sucede en las reglas de cuadratura convencionales. Sin embargo, si el problema (1.3) fuese escalar y la matriz A fuese nula, la solución exacta

sería la integral del término fuente y las relaciones (1.7)-(1.8) proporcionarían una fórmula de cuadratura clásica para aproximarla.

A continuación, vamos a hacer un análisis más detallado de las funciones peso $b_i(z)$ que aparecen en (1.7). Para comenzar, como los $\ell_i(\theta)$ son polinomios de grado menor o igual que $s - 1$, podemos reescribirlos en potencias de θ y, por tanto, podemos escribir los $b_i(z)$ como combinaciones lineales de funciones

$$\varphi_k(z) = \int_0^1 e^{(1-\theta)z} \frac{\theta^{k-1}}{(k-1)!} d\theta, \quad k \geq 1. \quad (1.9)$$

Veamos ahora algunas propiedades de estas funciones. En primer lugar, observemos que

$$\varphi_k(0) = \int_0^1 \frac{\theta^{k-1}}{(k-1)!} d\theta = \frac{1}{k!}. \quad (1.10)$$

Para $k = 1$ tenemos que

$$\varphi_1(z) = \int_0^1 e^{(1-\theta)z} d\theta = \frac{e^z - 1}{z}. \quad (1.11)$$

Para $k > 1$, tomando $u = e^{(1-\theta)z}$, $dv = \frac{\theta^{k-1}}{(k-1)!} d\theta$ en (1.9) e integrando por partes, se obtiene

$$\varphi_k(z) = \int_0^1 e^{(1-\theta)z} \frac{\theta^{k-1}}{(k-1)!} d\theta = \frac{1}{k!} + z \int_0^1 e^{(1-\theta)z} \frac{\theta^k}{k!} d\theta = \varphi_k(0) + z\varphi_{k+1}(z),$$

que, partiendo de $\varphi_0(z) = e^z$, da lugar a la siguiente relación de recurrencia para las funciones $\varphi_k(z)$

$$\varphi_{k+1}(z) = \frac{\varphi_k(z) - \varphi_k(0)}{z}, \quad k \geq 0. \quad (1.12)$$

Dado que estamos estudiando el caso en el que A es una matriz real, podemos definir las matrices

$$\varphi_k(-hA) = \int_0^1 e^{-h(1-\theta)A} \frac{\theta^{k-1}}{(k-1)!} d\theta, \quad k \geq 1, \quad (1.13)$$

y la misma recurrencia (1.12) sigue siendo válida.

Lema 1.2.1 *Supongamos que existen constantes C y ω tales que, para $t \geq 0$,*

$$\|e^{-tA}\| \leq Ce^{\omega t}. \quad (1.14)$$

Entonces, las matrices $\varphi_k(-hA)$, $k \geq 1$, están acotadas en \mathbb{R}^d .

Demostración. Tomando normas en ambos lados de la igualdad (1.13), tenemos que

$$\begin{aligned} \|\varphi_k(-hA)\| &= \left\| \int_0^1 e^{-h(1-\theta)A} \frac{\theta^{k-1}}{(k-1)!} d\theta \right\| \\ &\leq \int_0^1 \left\| e^{-h(1-\theta)A} \right\| \frac{\theta^{k-1}}{(k-1)!} d\theta \\ &\leq \int_0^1 Ce^{h(1-\theta)\omega} \frac{\theta^{k-1}}{(k-1)!} d\theta, \quad k \geq 1, \end{aligned}$$

donde hemos utilizado (1.14). Ahora, definimos esta última integral como

$$I_k = \int_0^1 C e^{h(1-\theta)\omega} \frac{\theta^{k-1}}{(k-1)!} d\theta, \quad k \geq 1.$$

Observemos que, integrando por partes, se obtiene que

$$I_k = \frac{C}{-(h\omega)(k-1)!} + \frac{I_{k-1}}{h\omega}, \quad k \geq 1.$$

Ahora, repitiendo esta operación de forma recursiva obtenemos

$$I_k = \frac{C}{(h\omega)^k} \left(e^{h\omega} - \sum_{i=0}^{k-1} \frac{(h\omega)^i}{i!} \right), \quad k \geq 1,$$

y utilizando la expresión del error en el polinomio de Taylor de grado $k-1$ de la función $e^{h\omega}$ en torno a 0, tenemos que

$$I_k = \frac{C}{(h\omega)^k} \frac{e^\xi (h\omega)^k}{k!} \leq \frac{C e^{h\omega}}{k!}, \quad k \geq 1,$$

donde $\xi \in [0, h\omega]$ y concluimos la demostración. □

Veamos a continuación algunos ejemplos concretos de métodos de cuadratura exponencial:

- Para $s = 1$, $\ell_1(\theta) \equiv 1$ y podemos escribir la solución numérica (1.7) como

$$\mathbf{y}_{n+1} = e^{-hA} \mathbf{y}_n + hb_1(-hA) \mathbf{F}(t_n + c_1 h) = e^{-hA} \mathbf{y}_n + h\varphi_1(-hA) \mathbf{F}(t_n + c_1 h).$$

Si ahora tomamos $c_1 = 0$, obtenemos la conocida regla de cuadratura exponencial de Euler

$$\mathbf{y}_{n+1} = e^{-hA} \mathbf{y}_n + h\varphi_1(-hA) \mathbf{F}(t_n), \quad (1.15)$$

y si en lugar de eso, escogemos $c_1 = \frac{1}{2}$, la regla resultante es la regla de cuadratura exponencial del punto medio

$$\mathbf{y}_{n+1} = e^{-hA} \mathbf{y}_n + h\varphi_1(-hA) \mathbf{F}\left(t_n + \frac{h}{2}\right). \quad (1.16)$$

- Para $s = 2$, calculamos las funciones peso a partir de (1.8) y (1.9). Para la primera

$$b_1(z) = \int_0^1 e^{(1-\theta)z} \frac{\theta - c_2}{c_1 - c_2} d\theta = \frac{1}{c_1 - c_2} \varphi_2(z) - \frac{c_2}{c_1 - c_2} \varphi_1(z), \quad (1.17)$$

y de forma análoga, intercambiando los papeles de c_1 y c_2 , obtenemos

$$b_2(z) = \frac{1}{c_2 - c_1} \varphi_2(z) - \frac{c_1}{c_2 - c_1} \varphi_1(z). \quad (1.18)$$

Tenemos entonces

$$\mathbf{y}_{n+1} = e^{-hA}\mathbf{y}_n + h [b_1(-hA)\mathbf{F}(t_n + c_1h) + b_2(-hA)\mathbf{F}(t_n + c_2h)],$$

con $b_1(z)$ y $b_2(z)$ dados por (1.17)-(1.18) y, tomando $c_1 = 0$ y $c_2 = 1$, obtenemos la regla de los trapecios exponencial

$$\begin{aligned} \mathbf{y}_{n+1} &= e^{-hA}\mathbf{y}_n + h [b_1(-hA)\mathbf{F}(t_n) + b_2(-hA)\mathbf{F}(t_n + h)] \\ &= e^{-hA}\mathbf{y}_n + h [(\varphi_1(-hA) - \varphi_2(-hA))\mathbf{F}(t_n) + \varphi_2(-hA)\mathbf{F}(t_n + h)]. \end{aligned} \quad (1.19)$$

Consideremos ahora métodos de la forma (1.7) obtenidos a partir de la fórmula de variación de las constantes, pero con pesos $b_i(z)$ arbitrarios, es decir, no dados por (1.8). Con el fin de hallar las condiciones de orden generales de esta familia de métodos, hacemos el desarrollo de Taylor de \mathbf{F} en t_n ($h = 0$), dentro de la ecuación (1.4)

$$\begin{aligned} \mathbf{y}(t_{n+1}) &= e^{-hA}\mathbf{y}(t_n) + h \int_0^1 e^{-h(1-\theta)A}\mathbf{F}(t_n + h\theta)d\theta \\ &= e^{-hA}\mathbf{y}(t_n) + h \sum_{k=1}^{\infty} \int_0^1 e^{-h(1-\theta)A}\mathbf{F}^{(k-1)}(t_n) \frac{(h\theta)^{k-1}}{(k-1)!} d\theta \\ &= e^{-hA}\mathbf{y}(t_n) + h \sum_{k=1}^p h^{k-1} \varphi_k(-hA)\mathbf{F}^{(k-1)}(t_n) \\ &\quad + \int_0^h e^{-(h-\tau)A} \left(\int_0^\tau \frac{(\tau-s)^{p-1}}{(p-1)!} \mathbf{F}^{(p)}(t_n + s) ds \right) d\tau, \end{aligned} \quad (1.20)$$

donde el último sumando corresponde al resto de Taylor en su forma integral. Seguimos el mismo procedimiento para la solución numérica (1.7) y obtenemos

$$\begin{aligned} \mathbf{y}_{n+1} &= e^{-hA}\mathbf{y}_n + h \sum_{i=1}^s b_i(-hA)\mathbf{F}(t_n + c_ih) \\ &= e^{-hA}\mathbf{y}_n + h \sum_{i=1}^s b_i(-hA) \sum_{k=1}^{\infty} \mathbf{F}^{(k-1)}(t_n) \frac{(c_ih)^{k-1}}{(k-1)!} \\ &= e^{-hA}\mathbf{y}_n + h \sum_{i=1}^s b_i(-hA) \sum_{k=1}^p \mathbf{F}^{(k-1)}(t_n) \frac{(c_ih)^{k-1}}{(k-1)!} \\ &\quad + h \sum_{i=1}^s b_i(-hA) \int_0^{c_ih} \frac{(c_ih-s)^{p-1}}{(p-1)!} \mathbf{F}^{(p)}(t_n + s) ds. \end{aligned} \quad (1.21)$$

Introduciendo el error $\mathbf{e}_n = \mathbf{y}_n - \mathbf{y}(t_n)$ para $n \geq 0$ y utilizando (1.20) y (1.21), obtenemos

$$\begin{aligned}
\mathbf{e}_{n+1} &= \mathbf{y}_{n+1} - \mathbf{y}(t_{n+1}) \\
&= e^{-hA} (\mathbf{y}_n - \mathbf{y}(t_n)) + \\
&\quad + \sum_{k=1}^p h^k \left[\sum_{i=1}^s b_i(-hA) \frac{c_i^{k-1}}{(k-1)!} - \varphi_k(-hA) \right] \mathbf{F}^{(k-1)}(t_n) \\
&\quad + h \sum_{i=1}^s b_i(-hA) \int_0^{c_i h} \frac{(c_i h - s)^{p-1}}{(p-1)!} \mathbf{F}^{(p)}(t_n + s) ds \\
&\quad - \int_0^h e^{-(h-\tau)A} \left(\int_0^\tau \frac{(\tau - s)^{p-1}}{(p-1)!} \mathbf{F}^{(p)}(t_n + s) ds \right) d\tau \\
&= e^{-hA} \mathbf{e}_n - \delta_{n+1},
\end{aligned} \tag{1.22}$$

donde

$$\delta_{n+1} = \sum_{k=1}^p h^k \phi_k(-hA) \mathbf{F}^{(k-1)}(t_n) + \delta_{n+1}^{(p)}, \tag{1.23}$$

$$\phi_k(z) = \varphi_k(z) - \sum_{i=1}^s b_i(z) \frac{c_i^{k-1}}{(k-1)!} \text{ para } k \geq 1, \tag{1.24}$$

$$\begin{aligned}
\delta_{n+1}^{(p)} &= \int_0^h e^{-(h-\tau)A} \left(\int_0^\tau \frac{(\tau - s)^{p-1}}{(p-1)!} \mathbf{F}^{(p)}(t_n + s) ds \right) d\tau \\
&\quad - h \sum_{i=1}^s b_i(-hA) \int_0^{c_i h} \frac{(c_i h - s)^{p-1}}{(p-1)!} \mathbf{F}^{(p)}(t_n + s) ds.
\end{aligned} \tag{1.25}$$

Con toda la información desarrollada a lo largo de este apartado, enunciamos el teorema que proporciona las condiciones de orden que buscábamos.

Teorema 1.2.1 Sean $A \in \mathbb{R}^d \times \mathbb{R}^d$ y $\mathbf{F} : [0, T] \rightarrow \mathbb{R}^d$ tal que $\mathbf{F}^{(p)} \in L^1(0, T)$. Consideremos la solución numérica de (1.3) obtenida con una regla de cuadratura exponencial (1.7) con funciones peso $b_i(-hA)$ arbitrarias y uniformemente acotadas para $h \geq 0$. Si el método satisface las condiciones de orden

$$\phi_j(-hA) = 0, \quad j = 1, \dots, p, \tag{1.26}$$

entonces es convergente de orden p , es decir, $\|\mathbf{e}_n\| = O(h^p)$. Más precisamente, existe una constante $K = K(T) > 0$ independiente de h tal que para todo $t_n = nh$ con $0 \leq t_n \leq T$, se verifica la cota de error

$$\|\mathbf{y}_n - \mathbf{y}(t_n)\| \leq K \sum_{j=0}^{n-1} h^p \int_{t_j}^{t_{j+1}} \|\mathbf{F}^{(p)}(\tau)\| d\tau. \tag{1.27}$$

Demostración. Basta seguir los siguientes pasos: partiendo de la expresión (1.22) e introduciendo la condición (1.26) en la definición (1.23), se tiene que

$$\mathbf{e}_n = e^{-hA} \mathbf{e}_{n-1} - \delta_n = e^{-hA} \mathbf{e}_{n-1} - \delta_n^{(p)}.$$

Repitiendo este mismo procedimiento para \mathbf{e}_{n-1} , obtenemos que

$$\mathbf{e}_n = e^{-hA}\mathbf{e}_{n-1} - \delta_n^{(p)} = e^{-hA} \left(e^{-hA}\mathbf{e}_{n-2} - \delta_{n-1}^{(p)} \right) - \delta_n^{(p)},$$

y por inducción se prueba que

$$\mathbf{e}_n = - \sum_{j=1}^n e^{-(n-j)hA} \delta_j^{(p)}.$$

Ahora, tomando normas a ambos lados de esta última igualdad, tenemos que

$$\|\mathbf{e}_n\| \leq \sum_{j=1}^n \left\| e^{-(n-j)hA} \right\| \left\| \delta_j^{(p)} \right\|. \quad (1.28)$$

A continuación, analizamos por separado cada una de las normas que aparecen en (1.28). En primer lugar, suponiendo que se cumple (1.14), tenemos que existen constantes C y ω (que sólo dependen de A) tales que

$$\left\| e^{-(n-j)hA} \right\| \leq C e^{\omega(n-j)h} \leq C e^{\omega T}. \quad (1.29)$$

Por otro lado, por (1.25) tenemos que $\left\| \delta_j^{(p)} \right\| \leq (I) + (II)$, siendo

$$\begin{aligned} (I) &= \int_0^h \left\| e^{-(h-\tau)A} \right\| \left(\int_0^\tau \frac{(\tau-s)^{p-1}}{(p-1)!} \left\| \mathbf{F}^{(p)}(t_{j-1}+s) \right\| ds \right) d\tau, \\ (II) &= h \sum_{i=1}^s \|b_i(-hA)\| \int_0^{c_i h} \frac{(c_i h - s)^{p-1}}{(p-1)!} \left\| \mathbf{F}^{(p)}(t_{j-1}+s) \right\| ds. \end{aligned}$$

Ahora, desarrollamos cada sumando teniendo en cuenta que $0 \leq t_n \leq T$ y suponiendo, de nuevo, que se cumple (1.14). En el primero, tenemos que

$$\begin{aligned} (I) &\leq \int_0^h C e^{\omega h} \left(\int_{t_{j-1}}^{t_j} \frac{h^{p-1}}{(p-1)!} \left\| \mathbf{F}^{(p)}(\theta) \right\| d\theta \right) d\tau \\ &\leq C h e^{\omega h} \frac{h^{p-1}}{(p-1)!} \int_{t_{j-1}}^{t_j} \left\| \mathbf{F}^{(p)}(\theta) \right\| d\theta \\ &\leq C e^{\omega T} \frac{h^p}{(p-1)!} \int_{t_{j-1}}^{t_j} \left\| \mathbf{F}^{(p)}(\theta) \right\| d\theta, \end{aligned}$$

donde hemos hecho el cambio de variable $\theta = s + t_{j-1}$ y, utilizando la positividad del integrando, hemos ampliado el intervalo de integración. Bajo las mismas condiciones y haciendo el mismo cambio de variable en (II), tenemos que

$$\begin{aligned} (II) &\leq hD \sum_{i=1}^s \int_{t_{j-1}}^{t_j} \frac{h^{p-1}}{(p-1)!} \left\| \mathbf{F}^{(p)}(\theta) \right\| d\theta \\ &= sD \frac{h^p}{(p-1)!} \int_{t_{j-1}}^{t_j} \left\| \mathbf{F}^{(p)}(\theta) \right\| d\theta, \end{aligned}$$

donde hemos utilizado que los c_i están en $[0, 1]$ y que existe $D > 0$ tal que $\|b_i(-hA)\| \leq D$ para $i = 1, \dots, s$ pues hemos supuesto que las funciones peso están uniformemente acotadas. Volviendo a $\|\delta_j^{(p)}\|$, obtenemos que

$$\|\delta_j^{(p)}\| \leq (I) + (II) \leq C_1 h^p \int_{t_{j-1}}^{t_j} \|\mathbf{F}^{(p)}(\theta)\| d\theta, \quad (1.30)$$

siendo $C_1 = \frac{C e^{\omega T} + sD}{(p-1)!}$. Finalmente, volviendo a (1.28) y utilizando (1.29) y (1.30), tenemos que

$$\|\mathbf{e}_n\| \leq \sum_{j=1}^n \|e^{-(n-j)hA}\| \|\delta_j^{(p)}\| \leq K \sum_{j=0}^{n-1} h^p \int_{t_j}^{t_{j+1}} \|\mathbf{F}^{(p)}(\theta)\| d\theta,$$

donde $K = C C_1 e^{\omega T}$ es una constante positiva que depende de T pero no de h . Obtenemos así la cota buscada (1.27). A partir de este punto, haciendo tender h hacia 0, es claro que $\|\mathbf{e}_n\| \rightarrow 0$, puesto que tenemos una suma finita de integrales acotadas y, por tanto, el método es convergente de orden p . □

Corolario 1.2.1 *La regla de cuadratura exponencial (1.7)-(1.8) satisface las condiciones de orden (1.26) para $p = s$. Por tanto, dicho método es convergente de orden s .*

Demostración. Si tomamos las funciones peso $b_i(z)$ de la forma (1.8), sabemos que la fórmula de cuadratura exponencial asociada es exacta para polinomios de grado menor o igual que $s-1$ por estar basada en una interpolación polinómica y que los c_i están entre 0 y 1. Teniendo en cuenta la definición de $\varphi_k(z)$ (1.9), tenemos que, para $1 \leq k \leq s$,

$$\varphi_k(z) = \int_0^1 e^{(1-\theta)z} \frac{\theta^{k-1}}{(k-1)!} d\theta = \int_0^1 e^{(1-\theta)z} \sum_{i=1}^s \ell_i(\theta) \frac{c_i^{k-1}}{(k-1)!} d\theta = \sum_{i=1}^s b_i(z) \frac{c_i^{k-1}}{(k-1)!}.$$

Por lo tanto, de la definición (1.24) dada para ϕ_k y de la igualdad anterior, tenemos que

$$\phi_k(-hA) = \varphi_k(-hA) - \sum_{i=1}^s b_i(-hA) \frac{c_i^{k-1}}{(k-1)!} = 0,$$

para $k = 1, \dots, s$. Así pues, la regla de cuadratura exponencial (1.7)-(1.8) verifica las condiciones (1.26) para $p = s$ y, como consecuencia del Teorema 1.2.1, el método es convergente de orden s . □

1.3. Caso semilineal

Pasamos a estudiar el caso que nos ocupa, los problemas semilineales. Para ello, se ha sintetizado la información que aparece al respecto en [2], [7] y [8]. Partimos de un sistema semilineal de ecuaciones diferenciales ordinarias de la forma

$$\mathbf{y}'(t) + \mathbf{A}\mathbf{y}(t) = \mathbf{F}(t, \mathbf{y}(t)), \quad 0 \leq t \leq T, \quad \text{con } \mathbf{y}(0) = \mathbf{y}_0, \quad (1.31)$$

donde $A \in \mathbb{R}^d \times \mathbb{R}^d$ y $\mathbf{F} : [0, T] \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ es un término no lineal.

El formato general de los métodos Runge-Kutta exponenciales es el siguiente: conocida la solución o una aproximación de ésta, \mathbf{y}_n , en un tiempo t_n , la aproximación \mathbf{y}_{n+1} de la solución de (1.31) en $t_n + h = t_{n+1}$ viene dada por

$$\mathbf{Y}_{ni} = \chi_i(-hA)\mathbf{y}_n + h \sum_{j=1}^s a_{ij}(-hA)\mathbf{F}_{nj}, \quad 1 \leq i \leq s, \quad (1.32)$$

$$\mathbf{y}_{n+1} = \chi(-hA)\mathbf{y}_n + h \sum_{i=1}^s b_i(-hA)\mathbf{F}_{ni}, \quad (1.33)$$

donde

$$\mathbf{F}_{nj} = \mathbf{F}(t_n + c_j h, \mathbf{Y}_{nj}), \quad 1 \leq j \leq s, \quad (1.34)$$

y los coeficientes que definen el método son funciones $\chi(z)$, $\chi_i(z)$, $a_{ij}(z)$ y $b_i(z)$, $1 \leq i, j \leq s$, que se obtienen a partir de funciones exponenciales y que están evaluadas en la matriz $-hA$. Por motivos de consistencia, supondremos que $\chi(0) = \chi_i(0) = 1$. Notemos que, si tomamos el límite cuando $A \rightarrow 0$, obtenemos el método Runge-Kutta con coeficientes $b_i = b_i(0)$ y $a_{ij} = a_{ij}(0)$, $1 \leq i, j \leq s$. Tal y como vimos en la introducción de este trabajo, en dicho método se verifica la igualdad (1.2) y, teniendo en cuenta la condición de orden 1 de éste, en adelante supondremos que los métodos (1.32)-(1.34) satisfacen

$$\begin{aligned} \sum_{j=1}^s a_{ij}(0) &= c_i \text{ para } 1 \leq i \leq s, \\ \sum_{j=1}^s b_j(0) &= 1. \end{aligned}$$

Como consecuencia, los métodos son invariantes ante la transformación del problema (1.31) en su forma autónoma y podemos restringirnos al caso en que \mathbf{F} sólo depende de \mathbf{y} .

Proposición 1.3.1 *Un método Runge-Kutta exponencial de la forma (1.32)-(1.34) preserva los equilibrios del problema autónomo si y solo si los coeficientes del método satisfacen*

$$\sum_{j=1}^s b_j(z) = \frac{\chi(z) - 1}{z}, \quad (1.35)$$

$$\sum_{j=1}^s a_{ij}(z) = \frac{\chi_i(z) - 1}{z} \text{ para } 1 \leq i \leq s. \quad (1.36)$$

Demostración. Sea \mathbf{y}^* un equilibrio del sistema autónomo

$$\mathbf{y}'(t) + \mathbf{A}\mathbf{y}(t) = \mathbf{F}(\mathbf{y}(t)),$$

lo que implica que $\mathbf{A}\mathbf{y}^* = \mathbf{F}(\mathbf{y}^*)$.

Supongamos que el método dado por (1.32)-(1.34) conserva los equilibrios, es decir, $\mathbf{y}_n = \mathbf{Y}_{ni} = \mathbf{y}^*$ para $i = 1, \dots, s$ y para $n \geq 0$. Ahora, introduciendo la solución constante \mathbf{y}^* en las ecuaciones (1.32)-(1.34), tenemos que

$$\begin{aligned}\mathbf{y}^* &= \chi_i(-hA)\mathbf{y}^* + h \sum_{j=1}^s a_{ij}(-hA)\mathbf{F}_{nj}, \\ \mathbf{y}^* &= \chi(-hA)\mathbf{y}^* + h \sum_{i=1}^s b_i(-hA)\mathbf{F}_{ni},\end{aligned}$$

con

$$\mathbf{F}_{nj} = A\mathbf{y}^*, \quad 1 \leq j \leq s,$$

de lo que se deduce que

$$\begin{aligned}0 &= \left(\chi_i(-hA) - I + h \sum_{j=1}^s a_{ij}(-hA)A \right) \mathbf{y}^*, \quad 1 \leq i \leq s, \\ 0 &= \left(\chi(-hA) - I + h \sum_{i=1}^s b_i(-hA)A \right) \mathbf{y}^*,\end{aligned}$$

y, como \mathbf{y}^* puede tomar cualquier valor constante, esto debe valer para un sistema arbitrario y obtenemos las condiciones (1.35) y (1.36).

Recíprocamente, supongamos que se verifican las condiciones (1.35) y (1.36) y veamos que, partiendo del equilibrio $\mathbf{y}_n = \mathbf{y}^*$, se verifica $\mathbf{y}_{n+1} = \mathbf{Y}_{ni} = \mathbf{y}^*$ para $i = 1, \dots, s$ y para $n \geq 0$.

Multiplicamos por z en ambos lados de las ecuaciones (1.35) y (1.36) y evaluamos en $-hA$, obteniendo que

$$\begin{aligned}-hA \sum_{j=1}^s a_{ij}(-hA) &= \chi_i(-hA) - I, \quad \text{para } 1 \leq i \leq s, \\ -hA \sum_{j=1}^s b_j(-hA) &= \chi(-hA) - I,\end{aligned}$$

Ahora, multiplicando las ecuaciones resultantes por el vector \mathbf{y}^* y reordenando adecuadamente, obtenemos

$$\begin{aligned}\mathbf{y}^* &= \chi_i(-hA)\mathbf{y}^* + h \sum_{j=1}^s a_{ij}(-hA)A\mathbf{y}^* = \chi_i(-hA)\mathbf{y}^* + h \sum_{j=1}^s a_{ij}(-hA)\mathbf{F}(\mathbf{y}^*), \\ \mathbf{y}^* &= \chi(-hA)\mathbf{y}^* + h \sum_{j=1}^s b_j(-hA)A\mathbf{y}^* = \chi(-hA)\mathbf{y}^* + h \sum_{j=1}^s b_j(-hA)\mathbf{F}(\mathbf{y}^*),\end{aligned}$$

para $1 \leq i \leq s$. Dado que \mathbf{y}^* satisface tanto las ecuaciones (1.32) que definen las etapas intermedias del método, como la ecuación (1.33) que proporciona la siguiente aproximación, teniendo en cuenta la unicidad de solución de las mismas, concluimos que

$\mathbf{y}_{n+1} = \mathbf{Y}_{ni} = \mathbf{y}^*$ para $i = 1, \dots, s$ y para $n \geq 0$. Es decir, el método preserva los equilibrios del problema autónomo si se satisfacen (1.35) y (1.36). \square

De ahora en adelante, supondremos que las funciones coeficientes del método satisfacen las condiciones (1.35) y (1.36) para $i = 1, \dots, s$, de forma que obtenemos una formulación alternativa para los métodos Runge-Kutta exponenciales

$$\begin{aligned} \mathbf{Y}_{ni} &= \chi_i(-hA)\mathbf{y}_n + h \sum_{j=1}^s a_{ij}(-hA)\mathbf{F}_{nj} \\ &= \left(-hA \sum_{j=1}^s a_{ij}(-hA) + I \right) \mathbf{y}_n + h \sum_{j=1}^s a_{ij}(-hA)\mathbf{F}_{nj}, \\ \mathbf{y}_{n+1} &= \chi(-hA)\mathbf{y}_n + h \sum_{i=1}^s b_i(-hA)\mathbf{F}_{ni} \\ &= \left(-hA \sum_{i=1}^s b_i(-hA) + I \right) \mathbf{y}_n + h \sum_{i=1}^s b_i(-hA)\mathbf{F}_{ni}, \end{aligned}$$

con

$$\mathbf{F}_{nj} = \mathbf{F}(\mathbf{Y}_{nj}). \quad (1.37)$$

Es decir,

$$\mathbf{Y}_{ni} = \mathbf{y}_n + h \sum_{j=1}^s a_{ij}(-hA) (\mathbf{F}_{nj} - A\mathbf{y}_n), \quad (1.38)$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h \sum_{i=1}^s b_i(-hA) (\mathbf{F}_{ni} - A\mathbf{y}_n). \quad (1.39)$$

1.3.1. Condiciones de orden

A continuación, vamos a estudiar las condiciones de orden de los métodos Runge-Kutta exponenciales siguiendo el razonamiento de [5] y [7]. De forma intuitiva, compararíamos el desarrollo en serie de Taylor de la solución exacta con el de la solución numérica dada por (1.37)-(1.39). Sin embargo, esto supondría el cálculo de un gran número de derivadas respecto de h y su evaluación en $h = 0$, lo cual puede no resultar una tarea sencilla si buscamos las condiciones para órdenes no tan altos. Veamos a modo de ejemplo cómo obtener las condiciones para los órdenes 1, 2 y 3 a partir de este método.

Supongamos que A no es la matriz nula. Partimos de la forma autónoma del problema (1.31)

$$\mathbf{y}'(t) = \mathbf{F}(\mathbf{y}(t)) - A\mathbf{y}(t). \quad (1.40)$$

El desarrollo de Taylor de la solución exacta $\mathbf{y}(t_{n+1})$ en t_n ($h = 0$) sería como sigue

$$\mathbf{y}(t_{n+1}) = \mathbf{y}(t_n + h) = \mathbf{y}(t_n) + h\mathbf{y}'(t_n) + \frac{h^2}{2}\mathbf{y}''(t_n) + \frac{h^3}{6}\mathbf{y}'''(t_n) + O(h^4),$$

y, desarrollando estas derivadas según (1.40), tenemos que

$$\begin{aligned}
\mathbf{y}(t_{n+1}) &= \mathbf{y} + h(\mathbf{F}(\mathbf{y}) - A\mathbf{y}) + \frac{h^2}{2}(\mathbf{F}'(\mathbf{y})(\mathbf{F}(\mathbf{y}) - A\mathbf{y}) - A(\mathbf{F}(\mathbf{y}) - A\mathbf{y})) \\
&\quad + \frac{h^3}{6}(\mathbf{F}''(\mathbf{y})\{\mathbf{F}(\mathbf{y}), \mathbf{F}(\mathbf{y}) - A\mathbf{y}\} - \mathbf{F}''(\mathbf{y})\{A\mathbf{y}, \mathbf{F}(\mathbf{y}) - A\mathbf{y}\}) \\
&\quad + \mathbf{F}'(\mathbf{y})\mathbf{F}'(\mathbf{y})(\mathbf{F}(\mathbf{y}) - A\mathbf{y}) - \mathbf{F}'(\mathbf{y})A(\mathbf{F}(\mathbf{y}) - A\mathbf{y}) \\
&\quad - A\mathbf{F}'(\mathbf{y})(\mathbf{F}(\mathbf{y}) - A\mathbf{y}) + AA(\mathbf{F}(\mathbf{y}) - A\mathbf{y})) + O(h^4),
\end{aligned} \tag{1.41}$$

donde notemos que hemos obviado la dependencia de \mathbf{y} respecto de t_n con el fin de no saturar la notación. Observemos que, por el mismo motivo, no estamos escribiendo de forma explícita cada una de las derivadas parciales de \mathbf{y} y denotamos por $\mathbf{F}^{(n)}\{\cdot, \dots, \cdot\}$ a la forma multilineal correspondiente a la n -ésima derivada de la función \mathbf{F} . Mantendremos esta notación durante todo el apartado.

Definimos la función $\mathbf{G}(\mathbf{y}) = \mathbf{F}(\mathbf{y}) - A\mathbf{y}$ de modo que el desarrollo (1.41) equivale a

$$\begin{aligned}
\mathbf{y}(t_{n+1}) &= \mathbf{y} + h\mathbf{G}(\mathbf{y}) + \frac{h^2}{2}(\mathbf{F}'(\mathbf{y})\mathbf{G}(\mathbf{y}) - A\mathbf{G}(\mathbf{y})) \\
&\quad + \frac{h^3}{6}(\mathbf{F}''(\mathbf{y})\{\mathbf{F}(\mathbf{y}), \mathbf{G}(\mathbf{y})\} - \mathbf{F}''(\mathbf{y})\{A\mathbf{y}, \mathbf{G}(\mathbf{y})\} + \mathbf{F}'(\mathbf{y})\mathbf{F}'(\mathbf{y})\mathbf{G}(\mathbf{y}) \\
&\quad - \mathbf{F}'(\mathbf{y})A\mathbf{G}(\mathbf{y}) - A\mathbf{F}'(\mathbf{y})\mathbf{G}(\mathbf{y}) + AAG(\mathbf{y})) + O(h^4).
\end{aligned} \tag{1.42}$$

Ahora, calculamos el desarrollo de Taylor de la solución numérica \mathbf{y}_{n+1} dada por (1.37)-(1.39) en t_n ($h = 0$). Para ello, vamos a considerar los desarrollos en serie de Taylor de las funciones peso $a_{ij}(z)$, $1 \leq i, j \leq s$,

$$a_{ij}(z) = \sum_{k \geq 0} \alpha_{ij}^{(k)} z^k, \tag{1.43}$$

y de las funciones $b_i(z)$, $1 \leq i \leq s$,

$$b_i(z) = \sum_{k \geq 0} \beta_i^{(k)} z^k. \tag{1.44}$$

Observemos que $\mathbf{Y}_{ni} = \mathbf{y}_n$ cuando $h = 0$ y, por tanto, $\mathbf{F}_{ni} = \mathbf{F}(\mathbf{y}_n)$ si $h = 0$. A continuación, calculamos el valor de la primera derivada de \mathbf{F}_{ni} respecto de h evaluada en $h = 0$

$$\left. \frac{d\mathbf{F}_{ni}}{dh} \right|_{h=0} = \mathbf{F}'(\mathbf{y}_n)(\mathbf{F}(\mathbf{y}_n) - A\mathbf{y}_n) \sum_{j=1}^s \alpha_{ij}^{(0)} = \mathbf{F}'(\mathbf{y}_n)\mathbf{G}(\mathbf{y}_n) \sum_{j=1}^s \alpha_{ij}^{(0)},$$

y, para la segunda derivada,

$$\begin{aligned}
\left. \frac{d^2 \mathbf{F}_{ni}}{dh^2} \right|_{h=0} &= \mathbf{F}''(\mathbf{y}_n) \{ \mathbf{F}(\mathbf{y}_n) - A\mathbf{y}_n, \mathbf{F}(\mathbf{y}_n) - A\mathbf{y}_n \} \sum_{j=1}^s \alpha_{ij}^{(0)} \sum_{k=1}^s \alpha_{ik}^{(0)} \\
&\quad - 2\mathbf{F}'(\mathbf{y}_n) A (\mathbf{F}(\mathbf{y}_n) - A\mathbf{y}_n) \sum_{j=1}^s \alpha_{ij}^{(1)} \\
&\quad + 2\mathbf{F}'(\mathbf{y}_n) \mathbf{F}'(\mathbf{y}_n) (\mathbf{F}(\mathbf{y}_n) - A\mathbf{y}_n) \sum_{j=1}^s \alpha_{ij}^{(0)} \sum_{k=1}^s \alpha_{jk}^{(0)} \\
&= \mathbf{F}''(\mathbf{y}_n) \{ \mathbf{G}(\mathbf{y}_n), \mathbf{G}(\mathbf{y}_n) \} \sum_{j,k=1}^s \alpha_{ij}^{(0)} \alpha_{ik}^{(0)} - 2\mathbf{F}'(\mathbf{y}_n) A \mathbf{G}(\mathbf{y}_n) \sum_{j=1}^s \alpha_{ij}^{(1)} \\
&\quad + 2\mathbf{F}'(\mathbf{y}_n) \mathbf{F}'(\mathbf{y}_n) \mathbf{G}(\mathbf{y}_n) \sum_{j,k=1}^s \alpha_{ij}^{(0)} \alpha_{jk}^{(0)}.
\end{aligned}$$

Con esta información, podemos calcular el desarrollo de Taylor de la solución numérica

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h \left. \frac{d\mathbf{y}_{n+1}}{dh} \right|_{h=0} + \frac{h^2}{2} \left. \frac{d^2 \mathbf{y}_{n+1}}{dh^2} \right|_{h=0} + \frac{h^3}{6} \left. \frac{d^3 \mathbf{y}_{n+1}}{dh^3} \right|_{h=0} + O(h^4). \quad (1.45)$$

Ahora, como queremos hallar las condiciones de orden, debemos comparar la expresión anterior con la que obtuvimos para la solución exacta del problema (1.42), de modo que vamos a calcular cada derivada de (1.45) para obtener las condiciones que buscamos.

Comenzamos por la primera derivada

$$\left. \frac{d\mathbf{y}_{n+1}}{dh} \right|_{h=0} = (\mathbf{F}(\mathbf{y}_n) - A\mathbf{y}_n) \sum_{i=1}^s \beta_i^{(0)} = \mathbf{G}(\mathbf{y}_n) \sum_{i=1}^s \beta_i^{(0)},$$

y, comparando con la expresión (1.42), obtenemos la condición de orden 1

$$\sum_{i=1}^s \beta_i^{(0)} = 1.$$

Hacemos lo mismo para la segunda derivada

$$\begin{aligned}
\left. \frac{d^2 \mathbf{y}_{n+1}}{dh^2} \right|_{h=0} &= -2A(\mathbf{F}(\mathbf{y}_n) - A\mathbf{y}_n) \sum_{i=1}^s \beta_i^{(1)} + 2\mathbf{F}'(\mathbf{y}_n) (\mathbf{F}(\mathbf{y}_n) - A\mathbf{y}_n) \sum_{i=1}^s \beta_i^{(0)} \sum_{j=1}^s \alpha_{ij}^{(0)} \\
&= -2A\mathbf{G}(\mathbf{y}_n) \sum_{i=1}^s \beta_i^{(1)} + 2\mathbf{F}'(\mathbf{y}_n) \mathbf{G}(\mathbf{y}_n) \sum_{i,j=1}^s \beta_i^{(0)} \alpha_{ij}^{(0)},
\end{aligned}$$

luego las condiciones de orden 2 son

$$\sum_{i=1}^s \beta_i^{(1)} = \frac{1}{2}, \quad \sum_{i,j=1}^s \beta_i^{(0)} \alpha_{ij}^{(0)} = \frac{1}{2}.$$

Por último, para la tercera derivada

$$\begin{aligned} \left. \frac{d^3 \mathbf{y}_{n+1}}{dh^3} \right|_{h=0} &= 6A^2 \mathbf{G}(\mathbf{y}_n) \sum_{i=1}^s \beta_i^{(2)} - 6A \mathbf{F}'(\mathbf{y}_n) \mathbf{G}(\mathbf{y}_n) \sum_{i,j=1}^s \beta_i^{(1)} \alpha_{ij}^{(0)} \\ &\quad + 3\mathbf{F}''(\mathbf{y}_n) \{ \mathbf{G}(\mathbf{y}_n), \mathbf{G}(\mathbf{y}_n) \} \sum_{i,j,k=1}^s \beta_i^{(0)} \alpha_{ij}^{(0)} \alpha_{ik}^{(0)} \\ &\quad - 6\mathbf{F}'(\mathbf{y}_n) A \mathbf{G}(\mathbf{y}_n) \sum_{i,j=1}^s \beta_i^{(0)} \alpha_{ij}^{(1)} + 6\mathbf{F}'(\mathbf{y}_n) \mathbf{F}'(\mathbf{y}_n) \mathbf{G}(\mathbf{y}_n) \sum_{i,j,k=1}^s \beta_i^{(0)} \alpha_{ij}^{(0)} \alpha_{jk}^{(0)}, \end{aligned}$$

de donde podemos deducir las condiciones de orden 3

$$\begin{aligned} \sum_{i=1}^s \beta_i^{(2)} &= \frac{1}{6}, & \sum_{i,j=1}^s \beta_i^{(1)} \alpha_{ij}^{(0)} &= \frac{1}{6}, & \sum_{i,j=1}^s \beta_i^{(0)} \alpha_{ij}^{(1)} &= \frac{1}{6}, \\ \sum_{i,j,k=1}^s \beta_i^{(0)} \alpha_{ij}^{(0)} \alpha_{ik}^{(0)} &= \frac{1}{3}, & \sum_{i,j,k=1}^s \beta_i^{(0)} \alpha_{ij}^{(0)} \alpha_{jk}^{(0)} &= \frac{1}{6}. \end{aligned}$$

Queda claro que este método resulta ineficiente para el cálculo de las condiciones de orden más alto, pues ya hemos visto que sólo para el tercer orden ya tenemos que realizar un gran número de operaciones y es fácil cometer algún error en el proceso. Para evitar estas complicaciones, vamos a mostrar una forma recursiva para la obtención de las condiciones de orden a través de la representación gráfica de las derivadas haciendo uso de un tipo adecuado de árboles con raíz. Previamente, es necesario definir bien la terminología con la que vamos a trabajar.

Definición 1.3.1 *Un árbol bicolor t con raíz es un grafo conexo sin ciclos formado por nodos que pueden ser blancos o negros, distribuidos en niveles de tal forma que en el nivel inferior hay un único nodo al que denominamos raíz. Exceptuando dicha raíz, cada nodo se conecta con un único nodo de su nivel inferior y puede conectarse o no con uno o más nodos del nivel superior atendiendo a las siguientes reglas:*

1. *Un nodo blanco puede conectarse a lo sumo con un único nodo del nivel superior.*
2. *Todos los nodos finales (los que no están conectados a nodos del nivel superior) son de color negro.*

El orden del grafo se define como el número de nodos que tiene y se denota por $\rho(t)$.

Denotamos por $t = [t_1, \dots, t_m, u_1, \dots, u_n]_K$ a un árbol tal que, al eliminar su raíz, los árboles restantes son $t_1, \dots, t_m, u_1, \dots, u_n$. Los árboles t_i se corresponden con árboles con raíz negra y los árboles u_i serán aquellos cuya raíz es blanca. La raíz de t será negra si $K = N$ o blanca si $K = B$. En particular, denotaremos por τ_N y por τ_B a los árboles formados por una única raíz, negra o blanca, respectivamente.

Definición 1.3.2 *Dado un árbol $t = [t_1, \dots, t_m, u_1, \dots, u_n]_K$, definimos de forma recursiva $\gamma(t)$ como*

$$\gamma(t) = \rho(t) \cdot \gamma(t_1) \cdots \gamma(t_m) \cdot \gamma(u_1) \cdots \gamma(u_n), \quad (1.46)$$

siendo $\gamma(\tau_N) = \gamma(\tau_B) = 1$.

Definición 1.3.3 Dado un árbol t de la forma especificada en la Definición 1.3.1, la diferencial elemental $\Delta(t)(\mathbf{y})$ asociada al problema (1.40) correspondiente a dicho árbol se define recursivamente como

- $\Delta(\tau_N)(\mathbf{y}) = \mathbf{G}(\mathbf{y})$,
- Si $K = N$ y t es de la forma $t = [t_1, \dots, t_m, u_1, \dots, u_n]_N$, entonces

$$\Delta(t)(\mathbf{y}) = \mathbf{F}^{(m+n)}(\mathbf{y})\{\Delta(t_1)(\mathbf{y}), \dots, \Delta(t_m)(\mathbf{y}), \Delta(u_1)(\mathbf{y}), \dots, \Delta(u_n)(\mathbf{y})\}.$$
- Si $K = B$, por construcción, t es un árbol con p nodos blancos seguidos hasta llegar al primer árbol t_1 con raíz negra. Es decir,

$$t = [[\dots [t_1]_B \dots]_B]_B,$$

$\underbrace{\hspace{10em}}_p$

y la diferencial elemental correspondiente es

$$\Delta(t)(\mathbf{y}) = A^p \Delta(t_1)(\mathbf{y}).$$

Definición 1.3.4 Sean (1.43) y (1.44) los desarrollos de Taylor de $a_{ij}(z)$ y $b_i(z)$, respectivamente. Dado un árbol t de la forma especificada en la Definición 1.3.1, el peso elemental $\Phi(t)$ asociado a un método Runge-Kutta exponencial dado por (1.37)-(1.39) correspondiente a dicho árbol se define recursivamente como

- $\Phi(\tau_N) = \sum_{i=1}^s \beta_i^{(0)}$, $\Phi_i(\tau_N) = \sum_{j=1}^s \alpha_{ij}^{(0)}$, $\hat{\Phi}_i(\tau_N) = 1$.
- Si $K = N$ y t es de la forma $t = [t_1, \dots, t_m, u_1, \dots, u_n]_N$, entonces el peso elemental asociado viene dado por

$$\begin{aligned} \Phi(t) &= \sum_{i=1}^s \beta_i^{(0)} \hat{\Phi}_i(t), \\ \Phi_i(t) &= \sum_{j=1}^s \alpha_{ij}^{(0)} \hat{\Phi}_j(t), \\ \hat{\Phi}_i(t) &= \Phi_i(t_1) \cdots \Phi_i(t_m) \Phi_i(u_1) \cdots \Phi_i(u_n). \end{aligned}$$

- Si $K = B$, por construcción, t es un árbol con p nodos blancos seguidos hasta llegar al primer árbol t_1 con raíz negra. Es decir,

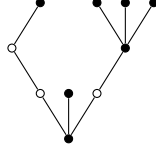
$$t = [[\dots [t_1]_B \dots]_B]_B,$$

$\underbrace{\hspace{10em}}_p$

y el peso elemental correspondiente es

$$\begin{aligned} \Phi(t) &= \sum_{i=1}^s \beta_i^{(p)} \hat{\Phi}_i(t_1), \\ \Phi_i(t) &= \sum_{j=1}^s \alpha_{ij}^{(p)} \hat{\Phi}_j(t_1), \end{aligned}$$

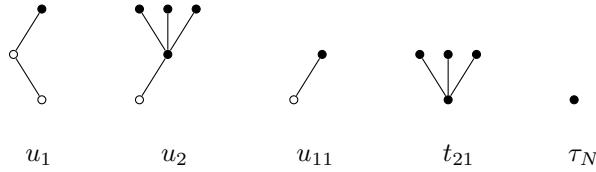
Veamos estos conceptos con ayuda de un ejemplo. Consideramos el siguiente árbol t .



Utilizando nuestra notación,

$$t = [u_1, \tau_N, u_2]_N = [[u_{11}]_B, \tau_N, [t_{21}]_B]_N = [[[\tau_N]_B]_B, \tau_N, [[\tau_N, \tau_N, \tau_N]_N]_B]_N,$$

donde



Como t tiene 10 nodos, tenemos que el orden del árbol es $\rho(t) = 10$. A continuación, calculamos el valor de $\gamma(t)$ según (1.46) eliminando las raíces de cada nivel sucesivamente y obtenemos

$$\begin{aligned} \gamma(t) &= \rho(t) \cdot \gamma(u_1) \cdot \gamma(\tau_N) \cdot \gamma(u_2) \\ &= \rho(t) \cdot \rho(u_1) \cdot \gamma(u_{11}) \cdot \gamma(\tau_N) \cdot \rho(u_2) \cdot \gamma(t_{21}) \\ &= \rho(t) \cdot \rho(u_1) \cdot \rho(u_{11}) \cdot \gamma(\tau_N) \cdot \gamma(\tau_N) \cdot \rho(u_2) \cdot \rho(t_{21}) \cdot \gamma(\tau_N) \cdot \gamma(\tau_N) \cdot \gamma(\tau_N) \\ &= 10 \cdot 3 \cdot 2 \cdot 1 \cdot 1 \cdot 5 \cdot 4 \cdot 1 \cdot 1 \cdot 1 = 1200. \end{aligned}$$

Ahora, calculamos la diferencial elemental asociada a este árbol para el problema (1.40) siguiendo las indicaciones de la Definición 1.3.3 y obtenemos

$$\begin{aligned} \Delta(t)(\mathbf{y}) &= \mathbf{F}'''(\mathbf{y})\{\Delta(u_1)(\mathbf{y}), \Delta(\tau_N)(\mathbf{y}), \Delta(u_2)(\mathbf{y})\} \\ &= \mathbf{F}'''(\mathbf{y})\{A\Delta(u_{11})(\mathbf{y}), \Delta(\tau_N)(\mathbf{y}), A\Delta(t_{21})(\mathbf{y})\} \\ &= \mathbf{F}'''(\mathbf{y})\{A^2\Delta(\tau_N)(\mathbf{y}), \Delta(\tau_N)(\mathbf{y}), A\mathbf{F}'''(\mathbf{y})\{\Delta(\tau_N), \Delta(\tau_N), \Delta(\tau_N)\}\} \\ &= \mathbf{F}'''(\mathbf{y})\{A^2\mathbf{G}(\mathbf{y}), \mathbf{G}(\mathbf{y}), A\mathbf{F}'''(\mathbf{y})\{\mathbf{G}(\mathbf{y}), \mathbf{G}(\mathbf{y}), \mathbf{G}(\mathbf{y})\}\}. \end{aligned}$$

Para terminar con el ejemplo, veamos cuál es el peso elemental correspondiente a este árbol para un método Runge-Kutta exponencial dado por (1.37)-(1.39) siguiendo la recurrencia de la Definición 1.3.4

$$\begin{aligned} \Phi(t) &= \sum_{i=1}^s \beta_i^{(0)} \Phi_i(u_1) \Phi_i(\tau_N) \Phi_i(u_2) = \sum_{i,j,k,l=1}^s \beta_i^{(0)} \alpha_{ij}^{(2)} \hat{\Phi}_j(\tau_N) \alpha_{ik}^{(0)} \alpha_{il}^{(1)} \hat{\Phi}_l(t_{21}) \\ &= \sum_{i,j,k,l=1}^s \beta_i^{(0)} \alpha_{ij}^{(2)} \alpha_{ik}^{(0)} \alpha_{il}^{(1)} \Phi_l(\tau_N) \Phi_l(\tau_N) \Phi_l(\tau_N) \\ &= \sum_{i,j,k,l,m,n,p=1}^s \beta_i^{(0)} \alpha_{ij}^{(2)} \alpha_{ik}^{(0)} \alpha_{il}^{(1)} \alpha_{lm}^{(0)} \alpha_{ln}^{(0)} \alpha_{lp}^{(0)}. \end{aligned}$$

Retomamos nuestro objetivo de hallar las condiciones de orden de un método Runge-Kutta exponencial de la forma (1.37)-(1.39) para aproximar la solución del problema (1.40) mediante el uso de estos últimos conceptos.

Si volvemos al desarrollo de Taylor de la solución exacta (1.42), observamos que podemos hallar cada derivada mediante los árboles especificados. En este caso, los nodos negros se asocian a las derivadas de \mathbf{F} y los nodos blancos, a la matriz A . Partimos de un árbol con un único nodo negro τ_N , asociado a \mathbf{G} y que corresponde a la primera derivada de la solución exacta del problema (1.40). Cada vez que derivamos, tomamos cada uno de los árboles correspondientes a la derivada anterior y añadimos un hijo en cada uno de sus nodos negros por separado, de forma que el nodo padre será negro si no se trataba de un nodo final y será negro o blanco en caso contrario. Es decir, de cada árbol t correspondiente a la derivada anterior obtenemos tantos árboles como la suma de los nodos negros no finales que hay en t más el doble del número de nodos finales de t . Así, podemos hallar cada una de las derivadas de la solución exacta, de forma que éstas serán iguales al sumatorio de las diferenciales elementales de los árboles correspondientes.

Notemos que estamos tomando los nodos finales como negros aunque al derivar en la diferencial elemental asociada tengamos que contar también con ellos como si fueran blancos. Esto se debe a que los árboles que son iguales salvo en el color de los nodos finales nos van a conducir a la misma condición de orden pero al derivar de nuevo necesitamos hacer esa distinción de color.

Podemos ver esto ejemplificado en la Tabla 1.1. Si partimos, por ejemplo, del árbol 6, siguiendo el procedimiento anterior obtendríamos los árboles 11, 16 y 18. Sus diferenciales elementales asociadas también pueden consultarse en la misma tabla.

Por otra parte, retomando el desarrollo de Taylor (1.45) de la solución numérica dada por (1.37)-(1.39), podemos observar que se obtienen las mismas diferenciales elementales en cada derivada que las que encontramos en el caso exacto pero multiplicadas por los pesos elementales asociados a cada árbol.

El siguiente teorema establece las condiciones de orden que buscamos teniendo en cuenta todo lo anterior.

Teorema 1.3.1 *Un método Runge-Kutta exponencial dado por (1.37)-(1.39) es de orden k si y sólo si para cada árbol bicolor t con raíz de orden menor o igual que k generado de la forma descrita en la Definición 1.3.1, se tiene que su peso elemental asociado satisface*

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

La demostración es análoga a la que encontramos en [5], Capítulo II.2, Teorema 2.13, para los métodos Runge-Kutta. Dicha demostración no se incluye en este trabajo para evitar extendernos en exceso en este apartado. No obstante, a continuación se muestra una tabla resumen con todos los árboles hasta orden 4 junto con las diferenciales elementales y las condiciones de orden asociadas a cada uno de ellos.

Nº	Árbol	Orden	Dif. Elem.	Cond. Orden
1		1	G	$\sum_i \beta_i^{(0)} = 1$
2		2	$F'G$	$\sum_{i,j} \beta_i^{(0)} \alpha_{ij}^{(0)} = \frac{1}{2}$
3		2	AG	$\sum_i \beta_i^{(1)} = \frac{1}{2}$
4		3	$F''\{G, G\}$	$\sum_{i,j,k} \beta_i^{(0)} \alpha_{ij}^{(0)} \alpha_{ik}^{(0)} = \frac{1}{3}$
5		3	$F'F'G$	$\sum_{i,j,k} \beta_i^{(0)} \alpha_{ij}^{(0)} \alpha_{jk}^{(0)} = \frac{1}{6}$
6		3	$F'AG$	$\sum_{i,j} \beta_i^{(0)} \alpha_{ij}^{(1)} = \frac{1}{6}$
7		3	$AF'G$	$\sum_{i,j} \beta_i^{(1)} \alpha_{ij}^{(0)} = \frac{1}{6}$
8		3	AAG	$\sum_i \beta_i^{(2)} = \frac{1}{6}$
9		4	$F'''\{G, G, G\}$	$\sum_{i,j,k,l} \beta_i^{(0)} \alpha_{ij}^{(0)} \alpha_{ik}^{(0)} \alpha_{il}^{(0)} = \frac{1}{4}$
10		4	$F''\{F'G, G\}$	$\sum_{i,j,k,l} \beta_i^{(0)} \alpha_{ij}^{(0)} \alpha_{ik}^{(0)} \alpha_{kl}^{(0)} = \frac{1}{8}$
11		4	$F''\{AG, G\}$	$\sum_{i,j,k} \beta_i^{(0)} \alpha_{ij}^{(0)} \alpha_{ik}^{(1)} = \frac{1}{8}$
12		4	$F'F''\{G, G\}$	$\sum_{i,j,k,l} \beta_i^{(0)} \alpha_{ij}^{(0)} \alpha_{jk}^{(0)} \alpha_{jl}^{(0)} = \frac{1}{12}$
13		4	$AF''\{G, G\}$	$\sum_{i,j,k} \beta_i^{(1)} \alpha_{ij}^{(0)} \alpha_{ik}^{(0)} = \frac{1}{12}$

Continuación de la Tabla 1.1








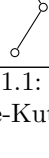
Nº	Árbol	Orden	Dif. Elem.	Cond. Orden
14		4	$F'F'F'G$	$\sum_{i,j,k,l} \beta_i^{(0)} \alpha_{ij}^{(0)} \alpha_{jk}^{(0)} \alpha_{kl}^{(0)} = \frac{1}{24}$
15		4	$F'F'AG$	$\sum_{i,j,k} \beta_i^{(0)} \alpha_{ij}^{(0)} \alpha_{jk}^{(1)} = \frac{1}{24}$
16		4	$F'AF'G$	$\sum_{i,j,k} \beta_i^{(0)} \alpha_{ij}^{(1)} \alpha_{jk}^{(0)} = \frac{1}{24}$
17		4	$AF'F'G$	$\sum_{i,j,k} \beta_i^{(1)} \alpha_{ij}^{(0)} \alpha_{jk}^{(0)} = \frac{1}{24}$
18		4	$F'AAG$	$\sum_{i,j} \beta_i^{(0)} \alpha_{ij}^{(2)} = \frac{1}{24}$
19		4	$AF'AG$	$\sum_{i,j} \beta_i^{(1)} \alpha_{ij}^{(1)} = \frac{1}{24}$
20		4	$AAF'G$	$\sum_{i,j} \beta_i^{(2)} \alpha_{ij}^{(0)} = \frac{1}{24}$
21		4	$AAAG$	$\sum_i \beta_i^{(3)} = \frac{1}{24}$

Tabla 1.1: Condiciones de orden no rígidas y árboles de los métodos Runge-Kutta exponenciales.

A partir de este momento, vamos a considerar los métodos Runge-Kutta exponenciales explícitos. Esto último quiere decir que $c_1 = 0$ y, por tanto, $\chi_1(z) = 1$ y $a_{ij}(z) = 0$ para

$1 \leq i \leq j \leq s$. Además, en este trabajo nos vamos a restringir a aquellos métodos que cumplan

$$\chi(z) = e^z, \quad (1.47)$$

$$\chi_i(z) = e^{c_i z}, \text{ para } i = 1, \dots, s. \quad (1.48)$$

Teniendo en cuenta (1.11), (1.35) y (1.36), observemos que

$$\sum_{j=1}^s b_j(z) = \frac{e^z - 1}{z} = \varphi_1(z),$$

$$\sum_{j=1}^s a_{ij}(z) = \frac{e^{c_i z} - 1}{z} = c_i \varphi_1(c_i z), \quad i = 1, \dots, s.$$

Las condiciones de orden que hemos analizado hasta ahora son las conocidas como *no rígidas*. Todo ello ha sido bajo la suposición de que A es una matriz real y, por tanto, su norma es finita. Sin embargo, tal y como se desarrolla en [7], A podría ser un operador lineal más general cuya discretización espacial, al ir refinando la red, podría dar lugar a que las condiciones de orden no rígidas que hemos visto hasta ahora no sean suficientes para garantizar un comportamiento del error con el orden esperado. Resumimos brevemente en la Tabla 1.2 las condiciones de orden *rígidas* que deben satisfacer los métodos Runge-Kutta exponenciales explícitos. Aquí, J y K denotan matrices en $\mathbb{R}^d \times \mathbb{R}^d$, $\phi_k(z)$ se encuentra definida en (1.24) y definimos $\phi_{k,j}(z)$ como

$$\phi_{k,j}(z) = \varphi_k(c_j z) c_j^k - \sum_{i=1}^{j-1} a_{ji}(z) \frac{c_i^{k-1}}{(k-1)!} \text{ para } k \geq 1, 1 \leq j \leq s.$$

Nº	Orden	Cond. Orden
1	1	$\phi_1(-hA) = 0$
2	2	$\phi_2(-hA) = 0$
3	2	$\phi_{1,i}(-hA) = 0$
4	3	$\phi_3(-hA) = 0$
5	3	$\sum_{i=1}^s b_i(-hA) J \phi_{2,i}(-hA) = 0$
6	4	$\phi_4(-hA) = 0$
7	4	$\sum_{i=1}^s b_i(-hA) J \phi_{3,i}(-hA) = 0$
8	4	$\sum_{i=1}^s b_i(-hA) J \sum_{j=2}^{i-1} a_{ij}(-hA) J \phi_{2,j}(-hA) = 0$
9	4	$\sum_{i=1}^s b_i(-hA) c_i K \phi_{2,i}(-hA) = 0$

Tabla 1.2: Condiciones de orden rígidas de los métodos Runge-Kutta exponenciales explícitos.

El proceso de obtención de dichas condiciones y un análisis de la convergencia de estos métodos está descrito con detalle y puede consultarse en [7], pero no lo incluimos en este trabajo.

Observemos que las condiciones rígidas de la Tabla 1.2 son válidas cuando las funciones χ y χ_i vienen dadas por (1.47) y (1.48), respectivamente, mientras que las condiciones de orden obtenidas mediante árboles en la Tabla 1.1 se pueden utilizar para funciones χ y χ_i arbitrarias.

1.3.2. Ejemplos de métodos Runge Kutta exponenciales explícitos

A continuación, vamos a ver algunos ejemplos de familias de métodos Runge Kutta exponenciales explícitos hasta el orden 4 que han aparecido en la literatura. Concretamente, en este apartado se sigue la Sección 5 de [7].

Representaremos cada método por medio de tableros de Butcher que, al igual que en el caso de los métodos Runge-Kutta clásicos, aportan toda la información necesaria sobre los nodos y las funciones peso del método en cuestión. Se estructuran de la siguiente forma

$$\begin{array}{c|cccc}
 c_1 & 0 & & & \\
 c_2 & a_{21}(z) & 0 & & \\
 \vdots & \vdots & \ddots & \ddots & \\
 c_s & a_{s1}(z) & \cdots & a_{s,s-1}(z) & 0 \\
 \hline
 & b_1(z) & \cdots & b_{s-1}(z) & b_s(z)
 \end{array}$$

Para simplificar el contenido en los tableros de Butcher de los métodos que se presentan en esta sección, denotaremos de forma abreviada las funciones (1.13) de la siguiente manera

$$\begin{aligned}
 \varphi_i &= \varphi_i(-hA), \quad 1 \leq i \leq s, \\
 \varphi_{i,j} &= \varphi_{i,j}(-hA) = \varphi_i(-c_j hA), \quad 1 \leq i \leq j \leq s.
 \end{aligned}$$

Métodos de orden 1

El único método exponencial explícito de orden 1 con una sola etapa ($s = 1$) es el método de Euler exponencial. Para aproximar la solución del problema (1.31), teniendo en cuenta (1.47) y (1.48), dicho método viene dado por (1.32)-(1.34) como

$$\mathbf{y}_{n+1} = e^{-hA} \mathbf{y}_n + h\varphi_1(-hA) \mathbf{F}(t_n, \mathbf{y}_n),$$

pues $\mathbf{Y}_{n1} = \mathbf{y}_n$.

En este caso, el tablero de Butcher correspondiente es el siguiente

$$\begin{array}{c|c}
 0 & \\
 \hline
 & \varphi_1
 \end{array}$$

Podemos ver que efectivamente el método es de orden 1 puesto que se verifica la primera condición de orden 1 de la Tabla 1.2, ya que $b_1(-hA) = \varphi_1(-hA)$. Además,

$$b_1(z) = \varphi_1(z) = \frac{e^z - 1}{z} = \sum_{k=0}^{\infty} \frac{z^{k-1}}{k!} - \frac{1}{z} = 1 + \sum_{k=1}^{\infty} \frac{z^k}{(k+1)!},$$

y vemos que también se verifica la condición de orden 1 no rígida pues $\beta_1^{(0)} = 1$.

Métodos de orden 2

Considerando métodos con dos etapas ($s = 2$), se construye una familia de métodos de orden 2 a partir de las condiciones de orden 1, 2 y 3 que se muestran en la Tabla 1.2

$$\varphi_1(-hA) = b_1(-hA) + b_2(-hA), \quad (1.49)$$

$$\varphi_2(-hA) = c_2 b_2(-hA), \quad (1.50)$$

$$\varphi_1(-c_2 hA) c_2 = a_{21}(-hA). \quad (1.51)$$

De la última ecuación se deduce que

$$a_{21}(-hA) = \varphi_1(-c_2 hA) c_2, \quad (1.52)$$

y de las dos primeras,

$$b_2(-hA) = \frac{1}{c_2} \varphi_2(-hA), \quad (1.53)$$

$$b_1(-hA) = \varphi_1(-hA) - \frac{1}{c_2} \varphi_2(-hA). \quad (1.54)$$

Observemos que, si se cumplen las condiciones rígidas hasta orden 2, también se verifican las no rígidas. Para ello, tendremos en cuenta las propiedades (1.10)-(1.12) y el desarrollo en serie de Taylor de la función exponencial. Comenzamos por hallar los coeficientes del desarrollo en serie de Taylor de la función $b_1(z)$

$$\begin{aligned} b_1(z) &= \varphi_1(z) - \frac{1}{c_2} \varphi_2(z) = \varphi_1(z) - \frac{1}{c_2} \frac{\varphi_1(z) - \varphi_1(0)}{z} = \frac{e^z - 1}{z} - \frac{1}{c_2 z} \left(\frac{e^z - 1}{z} - 1 \right) \\ &= \frac{1}{c_2 z^2} \left(\sum_{k=0}^{\infty} \frac{z^k}{k!} (z c_2 - 1) + z(1 - c_2) + 1 \right) = \sum_{k=0}^{\infty} \left(\frac{1}{(k+1)!} - \frac{1}{(k+2)! c_2} \right) z^k, \end{aligned}$$

de modo que $\beta_1^{(0)} = 1 - \frac{1}{2c_2}$ y $\beta_1^{(1)} = \frac{1}{2} - \frac{1}{6c_2}$.

De la misma manera, calculamos los coeficientes del desarrollo en serie de Taylor de $b_2(z)$

$$\begin{aligned} b_2(z) &= \frac{1}{c_2} \varphi_2(z) = \frac{1}{c_2} \frac{\varphi_1(z) - \varphi_1(0)}{z} = \frac{1}{c_2} \left(\frac{e^z - 1}{z^2} - \frac{1}{z} \right) = \frac{1}{c_2 z^2} \left(\sum_{k=0}^{\infty} \frac{z^k}{k!} - 1 - z \right) \\ &= \sum_{k=0}^{\infty} \frac{z^k}{(k+2)! c_2}, \end{aligned}$$

luego $\beta_2^{(0)} = \frac{1}{2c_2}$ y $\beta_2^{(1)} = \frac{1}{6c_2}$.

Siguiendo el mismo procedimiento para $a_{21}(z)$, tenemos que

$$a_{21}(z) = c_2 \varphi_1(c_2 z) = c_2 \frac{e^{c_2 z} - 1}{c_2 z} = \sum_{k=0}^{\infty} \frac{c_2^k z^{k-1}}{k!} - \frac{1}{z} = \sum_{k=0}^{\infty} \frac{c_2^{k+1}}{(k+1)!} z^k,$$

por lo que $\alpha_{21}^{(0)} = c_2$.

Vemos, por tanto, que se verifican también las condiciones de orden asociadas a los tres primeros árboles de la Tabla 1.1

$$\begin{aligned} \sum_i \beta_i^{(0)} &= \beta_1^{(0)} + \beta_2^{(0)} = 1, \\ \sum_{i,j} \beta_i^{(0)} \alpha_{ij}^{(0)} &= \beta_2^{(0)} \alpha_{21}^{(0)} = \frac{1}{2}, \\ \sum_i \beta_i^{(1)} &= \beta_1^{(1)} + \beta_2^{(1)} = \frac{1}{2}. \end{aligned}$$

Por tanto, hemos comprobado que las condiciones hasta orden 2 rígidas llevan implícitas las condiciones hasta orden 2 no rígidas.

Volviendo a (1.52)-(1.54), concluimos que tenemos una familia de métodos de orden 2 que depende de un único parámetro c_2 y su tablero de Butcher es el siguiente

$$\begin{array}{c|c} 0 & \\ \hline c_2 & c_2 \varphi_{1,2} \\ \hline & \varphi_1 - \frac{1}{c_2} \varphi_2 \quad \frac{1}{c_2} \varphi_2 \end{array} .$$

Consideremos ahora una versión más débil de la condición (1.50) pidiendo que sea cierta sólo para $A \equiv 0$, pero manteniendo la condición de orden no rígida número 2. Es decir,

$$b_2(0)c_2 = \varphi_2(0) = \frac{1}{2}. \tag{1.55}$$

Pese a haber modificado una de las condiciones de orden, tal y como se demuestra en el Teorema 4.7 de [7], bajo determinadas hipótesis, esto no supone una pérdida de orden en el método.

Como $\varphi_1(0) = 1$, se verifica que

$$b_2(0) = \varphi_1(0) \frac{1}{2c_2},$$

luego podemos construir una nueva familia de métodos dependiente del parámetro c_2 y que verifica las condiciones (1.49), (1.51) y (1.55) tomando

$$\begin{aligned} b_2(z) &= \varphi_1(z) \frac{1}{2c_2}, \\ b_1(z) &= \varphi_1(z) \left(1 - \frac{1}{2c_2} \right), \end{aligned}$$

y el tablero de Butcher correspondiente es el siguiente

$$\begin{array}{c|cc} 0 & & \\ c_2 & c_2\varphi_{1,2} & \\ \hline & \left(1 - \frac{1}{2c_2}\right)\varphi_1 & \frac{1}{2c_2}\varphi_1 \end{array} .$$

Notemos que si se escoge $c_2 = \frac{1}{2}$, se tiene que $b_1(z) \equiv 0$.

Métodos de orden 3

Para métodos con tres etapas ($s = 3$), las condiciones de orden rígidas hasta orden 3 dadas en la Tabla 1.2 son las siguientes

$$\begin{aligned} \varphi_1(-hA) &= b_1(-hA) + b_2(-hA) + b_3(-hA), \\ \varphi_2(-hA) &= b_2(-hA)c_2 + b_3(-hA)c_3, \end{aligned} \quad (1.56)$$

$$\begin{aligned} \varphi_1(-c_2hA)c_2 &= a_{21}(-hA), \\ \varphi_1(-c_3hA)c_3 &= a_{31}(-hA) + a_{32}(-hA), \\ 2\varphi_3(-hA) &= b_2(-hA)c_2^2 + b_3(-hA)c_3^2, \end{aligned} \quad (1.57)$$

$$\begin{aligned} 0 &= b_2(-hA)J\varphi_2(-c_2hA)c_2^2 + b_3(-hA)J\varphi_2(-c_3hA)c_3^2 \\ &\quad - b_3(-hA)Ja_{32}(-hA)c_2. \end{aligned} \quad (1.58)$$

Se puede comprobar que estas condiciones implican el cumplimiento de las condiciones no rígidas de orden 3 de forma análoga a como se hizo en el caso de orden 2.

Para que se satisfaga (1.58), podemos tomar o bien

$$b_2 = 0, \quad \varphi_2(-c_3hA)c_3^2 - a_{32}(-hA)c_2 = 0, \quad (1.59)$$

o bien

$$b_2 = \gamma b_3, \quad \gamma (\varphi_2(-c_3hA)c_3^2 - a_{32}(-hA)c_2) + \varphi_2(-c_2hA)c_2^2 = 0, \quad (1.60)$$

con $\gamma \in \mathbb{R}$. Sin embargo, en ambos casos se incumplen las condiciones (1.56) y (1.57). Una posibilidad es sustituir la ecuación (1.57) por una versión más débil de la misma, pidiendo únicamente que sea cierta para $A \equiv 0$, es decir,

$$b_2(0)c_2^2 + b_3(0)c_3^2 = 2\varphi_3(0) = \frac{1}{3}, \quad (1.61)$$

que se corresponde con la condición de orden no rígida número 4 de la Tabla 1.1 y, por el Teorema 4.7 de [7] ya mencionado, bajo ciertas condiciones esto puede no dar lugar a una reducción del orden del método.

Atendiendo a estas condiciones y suponiendo que estamos en el caso (1.59), tenemos que

$$\begin{aligned} b_1 &= \varphi_1 - \frac{1}{c_3}\varphi_2, \quad b_2 = 0, \quad b_3 = \frac{1}{c_3}\varphi_2, \\ a_{21} &= c_2\varphi_{1,2}, \quad a_{31} = c_3\varphi_{1,3} - \frac{c_3^2}{c_2}\varphi_{2,3}, \quad a_{32} = \frac{c_3^2}{c_2}\varphi_{2,3}. \end{aligned}$$

Además, como estamos suponiendo (1.61), deducimos que

$$c_3 = \frac{1}{3\varphi_2(0)} = \frac{2}{3}.$$

Por tanto, tenemos una nueva familia de métodos de orden 3 que depende de un único parámetro y cuyo tablero de Butcher es

$$\begin{array}{c|ccc} 0 & & & \\ c_2 & c_2\varphi_{1,2} & & \\ \frac{2}{3} & \frac{2}{3}\varphi_{1,3} - \frac{4}{9c_2}\varphi_{2,3} & \frac{4}{9c_2}\varphi_{2,3} & \\ \hline & \varphi_1 - \frac{3}{2}\varphi_2 & 0 & \frac{3}{2}\varphi_2 \end{array}.$$

Si ahora consideramos el caso (1.60), entonces

$$\begin{aligned} b_1 &= \varphi_1 - \frac{1+\gamma}{\gamma c_2 + c_3}\varphi_2, & b_2 &= \frac{\gamma}{\gamma c_2 + c_3}\varphi_2, & b_3 &= \frac{1}{\gamma c_2 + c_3}\varphi_2, \\ a_{21} &= c_2\varphi_{1,2}, & a_{31} &= c_3\varphi_{1,3} - \gamma c_2\varphi_{2,2} - \frac{c_3^2}{c_2}\varphi_{2,3}, & a_{32} &= \gamma c_2\varphi_{2,2} + \frac{c_3^2}{c_2}\varphi_{2,3}, \end{aligned}$$

donde, necesariamente, los coeficientes c_2 , c_3 y γ deben verificar la igualdad

$$2(\gamma c_2 + c_3) = 3(\gamma c_2^2 + c_3^2). \quad (1.62)$$

El tablero de Butcher correspondiente a esta familia de métodos de orden 3, que depende de dos parámetros, es el siguiente

$$\begin{array}{c|ccc} 0 & & & \\ c_2 & c_2\varphi_{1,2} & & \\ c_3 & c_3\varphi_{1,3} - \gamma c_2\varphi_{2,2} - \frac{c_3^2}{c_2}\varphi_{2,3} & \gamma c_2\varphi_{2,2} + \frac{c_3^2}{c_2}\varphi_{2,3} & \\ \hline & \varphi_1 - \frac{1+\gamma}{\gamma c_2 + c_3}\varphi_2 & \frac{\gamma}{\gamma c_2 + c_3}\varphi_2 & \frac{1}{\gamma c_2 + c_3}\varphi_2 \end{array}.$$

Métodos de orden 4

Las condiciones rígidas hasta orden 4 de un método explícito con s etapas son

$$\varphi_1(-hA) = \sum_{i=1}^s b_i(-hA), \quad (1.63)$$

$$\varphi_2(-hA) = \sum_{i=2}^s b_i(-hA)c_i, \quad (1.64)$$

$$c_i\varphi_1(-c_i hA) = \sum_{j=1}^{i-1} a_{ij}(-hA), \quad i = 1, \dots, s, \quad (1.65)$$

$$2\varphi_3(-hA) = \sum_{i=2}^s b_i(-hA)c_i^2, \quad (1.66)$$

$$6\varphi_4(-hA) = \sum_{i=2}^s b_i(-hA)c_i^3,$$

$$0 = \sum_{i=2}^s b_i(-hA)J \left(\varphi_2(-c_i hA)c_i^2 - \sum_{j=2}^{i-1} a_{ij}(-hA)c_j \right), \quad (1.67)$$

$$0 = \sum_{i=2}^s b_i(-hA)J \left(\varphi_3(-c_i hA)c_i^3 - \frac{1}{2} \sum_{j=2}^{i-1} a_{ij}(-hA)c_j^2 \right), \quad (1.68)$$

$$0 = \sum_{i=2}^s b_i(-hA)J \sum_{j=2}^{i-1} a_{ij}(-hA)J \left(\varphi_2(-c_j hA)c_j^2 - \sum_{k=2}^{j-1} a_{jk}(-hA)c_k \right), \quad (1.69)$$

$$0 = \sum_{i=2}^s b_i(-hA)c_i K \left(\varphi_2(-c_i hA)c_i^2 - \sum_{j=2}^{i-1} a_{ij}(-hA)c_j \right), \quad (1.70)$$

A continuación, se construye un método de orden 4 con 5 etapas ($s = 5$) con nodos $c_1 = c_4 = 1$ y $c_2 = c_3 = c_5 = \frac{1}{2}$. Para evitar dificultades, siguiendo [7], suponemos que $b_2 \equiv b_3 \equiv 0$, teniendo en cuenta las condiciones (1.63), (1.64) y (1.66) se deduce que

$$b_1 = \varphi_1 - 3\varphi_2 + 4\varphi_3, \quad b_4 = -\varphi_2 + 4\varphi_3, \quad b_5 = 4\varphi_2 - 8\varphi_3.$$

La condición (1.67) implica necesariamente que $\phi_{2,4} = \phi_{2,5} = 0$ y, por tanto, se verifica la condición (1.70). Ahora, la condición (1.69) implica que $a_{42} = \gamma a_{43}$ y $a_{52} = \gamma a_{53}$, donde tomaremos $\gamma = 1$ para facilitar los cálculos. Por tanto, de la misma condición se obtiene que

$$\phi_{2,2} + \phi_{2,3} = 0 = \frac{1}{4}\varphi_{2,2} + \frac{1}{4}\varphi_{2,3} - \frac{1}{2}a_{32},$$

y como $\varphi_{2,2} = \varphi_{2,3}$, tenemos que

$$a_{32} = \varphi_{2,3}.$$

Además, como $\varphi_{2,4} = 0$, se tiene que

$$a_{42} = a_{43} = \varphi_{2,4}.$$

Ahora, de (1.65) para $i = 2, 3$, deducimos que

$$a_{21} = \frac{1}{2}\varphi_{1,2}, \quad a_{31} = \frac{1}{2}\varphi_{1,3} - \varphi_{2,3}.$$

Observemos que es imposible que se verifique la condición (1.68) pues si $\phi_{3,4} = \phi_{3,5} = 0$, entonces no puede ser que $\phi_{2,4} = \phi_{2,5} = 0$ al mismo tiempo, por lo que tomamos una versión más débil de la misma para $A \equiv 0$. A partir de ésta última condición junto con (1.65) para $i = 5$ y $\phi_{2,5} = 0$, se deduce que

$$a_{51} = \frac{1}{2}\varphi_{1,5} - 2a_{52} - a_{54}, \quad a_{52} = a_{53} = \frac{1}{2}\varphi_{2,5} - \varphi_{3,4} + \frac{1}{4}\varphi_{2,4} - \frac{1}{2}\varphi_{3,5}, \quad a_{54} = \frac{1}{4}\varphi_{2,5} - a_{52}.$$

A continuación, se muestra el tablero de Butcher del método de orden 4 con 5 etapas

desarrollado en [7]

$$\begin{array}{c|cccccc}
 0 & & & & & & \\
 \frac{1}{2} & & \frac{1}{2}\varphi_{1,2} & & & & \\
 \frac{1}{2} & & \frac{1}{2}\varphi_{1,3} - \varphi_{2,3} & \varphi_{2,3} & & & \\
 1 & & \varphi_{1,4} - 2\varphi_{2,4} & \varphi_{2,4} & \varphi_{2,4} & & \\
 \frac{1}{2} & \frac{1}{2}\varphi_{1,5} - 2a_{52} - a_{54} & a_{52} & a_{52} & \frac{1}{4}\varphi_{2,5} - a_{52} & & \\
 \hline
 & \varphi_1 - 3\varphi_2 + 4\varphi_3 & 0 & 0 & -\varphi_2 + 4\varphi_3 & 4\varphi_2 - 8\varphi_3 &
 \end{array}$$

donde

$$a_{5,2} = \frac{1}{2}\varphi_{2,5} - \varphi_{3,4} + \frac{1}{4}\varphi_{2,4} - \frac{1}{2}\varphi_{3,5}.$$

Capítulo 2

Implementación y resultados numéricos

2.1. Aspectos prácticos de implementación

En este capítulo, se van a mostrar algunos resultados numéricos para ilustrar el comportamiento de los métodos exponenciales y las ventajas que pueden aportar sobre los métodos Runge-Kutta explícitos. Para ello, se han elaborado en MATLAB los programas necesarios para implementar algunos de los métodos estudiados. Una de las dificultades que presenta la implementación de los métodos exponenciales es el cálculo eficiente del producto de las matrices $\varphi_i(-hA)$ y $\varphi_{ij}(-hA)$ que aparecen en la Sección 1.3.2 por un vector \mathbf{u} dado. Cuando se aplican integradores exponenciales para aproximar la solución de un problema semilineal dado, es habitual que dicho problema se haya obtenido discretizando espacialmente una ecuación en derivadas parciales, de modo que el problema resultante es de la forma (1.31) y la matriz A es dispersa pero de dimensión alta.

Para solventarlo, en los programas elaborados hemos utilizado la función `phipm.m` tomada de [6], que evalúa una combinación lineal de funciones $\varphi_k(tA)$ multiplicando a los vectores correspondientes, es decir, calcula vectores \mathbf{w} dados por

$$\mathbf{w} = \varphi_0(tA)\mathbf{u}_1 + t\varphi_1(tA)\mathbf{u}_2 + t^2\varphi_2(tA)\mathbf{u}_3 + \dots \quad (2.1)$$

Por otra parte, cada paso de los métodos exponenciales desarrollados puede escribirse como una combinación lineal de funciones $\varphi_k(-hA)$ actuando sobre determinados vectores de la forma

$$\mathbf{w} = \varphi_0(-hA)\mathbf{b}_0 + \varphi_1(-hA)\mathbf{b}_1 + \varphi_2(-hA)\mathbf{b}_2 + \dots + \varphi_p(-hA)\mathbf{b}_p, \quad (2.2)$$

donde p es el orden del método en cuestión.

Para poder implementar los métodos, necesitamos entonces calcular varias veces en cada paso expresiones de la forma (2.2) de un modo eficiente y con el menor error posible. Bajo estas circunstancias, la función `phipm.m` hace uso de los subespacios de Krylov para la reducción de matrices de dimensión alta a la hora de evaluar las funciones matriciales,

dando como resultado \mathbf{w} a partir de una relación de recurrencia. Además, hace una estimación del error cometido tras cada aproximación con el subespacio de Krylov y lo utiliza para adaptar la dimensión del mismo durante la integración.

Para dar una visión general del funcionamiento del programa sin entrar en demasiado detalle (véase la referencia [6]), consideramos el problema de calcular $\varphi_p(A)\mathbf{v}$, donde A es una matriz de dimensión $d \times d$ y $\mathbf{v} \in \mathbb{R}^d$. Notemos que se trata de uno de los términos de la suma en (2.2) y que d habitualmente será grande.

Para hallar el vector $\varphi_p(A)\mathbf{v} \in \mathbb{R}^d$, consideramos el subespacio de Krylov K_m de dimensión m generado por $\{\mathbf{v}, A\mathbf{v}, A^2\mathbf{v}, \dots, A^{m-1}\mathbf{v}\}$. Ahora, podemos hallar una base ortonormal $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_m\}$ para este subespacio mediante el método de ortonormalización de Gram-Schmidt. Sea $V_m \in \mathbb{R}^d \times \mathbb{R}^m$ la matriz cuyas columnas son $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_m$ y $H_m = V_m^T A V_m$ la proyección de A sobre el subespacio de Krylov K_m expresada en la base $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_m\}$. Tenemos entonces que $V_m H_m V_m^T$ es la proyección de A en el subespacio K_m en la base canónica de \mathbb{R}^d . Notemos además que H_m es una matriz de Hessenberg superior, es decir, $(H_m)_{ij} = 0$ si $i > j + 1$.

Aproximando $\varphi_p(A)\mathbf{v}$ por $\varphi_p(V_m H_m V_m^T)\mathbf{v}$ y teniendo en cuenta que $V_m^T V_m = I_m$ y $V_m V_m^T \mathbf{v} = \mathbf{v}$, tenemos que $\varphi_p(V_m H_m V_m^T)\mathbf{v} = V_m \varphi_p(H_m) V_m^T \mathbf{v}$ pues las funciones $\varphi_p(z)$ son exponenciales y, para determinados coeficientes μ_l , $l \geq 0$, se puede escribir su desarrollo de Taylor como

$$\varphi_p(V_m H_m V_m^T)\mathbf{v} = \sum_{l=0}^{\infty} \mu_l (V_m H_m V_m^T)^l \mathbf{v} = V_m \left(\sum_{l=0}^{\infty} \mu_l H_m^l \right) V_m^T \mathbf{v}.$$

Además, dado que $\mathbf{v}_1 = \frac{\mathbf{v}}{\|\mathbf{v}\|}$, se verifica $\mathbf{v}_1^T \mathbf{v} = \|\mathbf{v}\|$ y $\mathbf{v}_i^T \mathbf{v} = 0$ para $i = 2, \dots, m$. Como consecuencia, se tiene que $V_m^T \mathbf{v} = \|\mathbf{v}\| \mathbf{e}_1$, donde \mathbf{e}_1 denota el primer vector de la base canónica y, por tanto, obtenemos la aproximación

$$\varphi_p(A)\mathbf{v} \approx \beta V_m \varphi_p(H_m) \mathbf{e}_1, \quad \beta = \|\mathbf{v}\|.$$

Si A no es simétrica, el algoritmo que se emplea para hallar H_m es el algoritmo de Arnoldi, pero en el caso de que A sea simétrica, como H_m es de Hessenberg superior y simétrica, también es tridiagonal y, para aprovechar esa característica, el algoritmo empleado para calcular H_m es el de Lanczos. Ambos algoritmos pueden consultarse en [6] (Algoritmo 1 y Algoritmo 2).

Por tanto, si utilizamos la técnica de los subespacios de Krylov, el problema de calcular $\varphi_p(A)\mathbf{v}$ con A una matriz de dimensión alta $d \times d$ se reduce a hallar $\varphi_p(H_m)\mathbf{e}_1$ con H_m de menor tamaño, $m \times m$. Sin embargo, para calcular $\varphi_p(H_m)$ bajo estas condiciones, normalmente, tenemos que calcular previamente $\varphi_0(H_m), \dots, \varphi_{p-1}(H_m)$. Para evitarlo, se considera la matriz ampliada de dimensión $(m+p) \times (m+p)$

$$\hat{H}_m = \begin{pmatrix} H_m & \mathbf{e}_1 & 0 \\ 0 & 0 & I_{p-1} \\ 0 & 0 & 0 \end{pmatrix},$$

de forma que al calcular $\exp(\hat{H}_m)$, obtenemos el vector $\varphi_p(H_m)\mathbf{e}_1$ en las m primeras componentes de la última columna.

Además, la función `phipm.m` también hace una estimación del error cometido en la aproximación anterior que utiliza como corrector y sirve para adaptar la dimensión del subespacio de Krylov para la siguiente iteración. No vamos a describir la recurrencia en sí pero se hace tomando distintas longitudes de paso dentro del intervalo de tiempo $[0, t]$, donde t es el mismo parámetro que encontramos en (2.1) y también varía en función del error estimado. Esta parte se encuentra bien detallada en las Secciones 3.3 y 3.4 de [6].

Para poder incluir la función `phipm.m` en los programas elaborados, se han reescrito en primer lugar las fórmulas que definen cada método descrito en la Sección 1.3.2 como combinaciones lineales de las funciones φ_k multiplicando a los vectores correspondientes.

El método de Euler exponencial toma la forma

$$\begin{aligned}\mathbf{Y}_{n1} &= \mathbf{y}_n, \\ \mathbf{y}_{n+1} &= e^{-hA}\mathbf{y}_n + h\varphi_1(-hA)\mathbf{F}_{n1},\end{aligned}$$

donde

$$\mathbf{F}_{n1} = \mathbf{F}(\mathbf{Y}_{n1}).$$

En cuanto a los métodos de orden 2, el primero de ellos, al que nos referiremos en lo sucesivo como Método de orden 2 (I), viene dado por

$$\begin{aligned}\mathbf{Y}_{n1} &= \mathbf{y}_n, \\ \mathbf{Y}_{n2} &= \mathbf{y}_n + hc_2\varphi_{1,2}(-hA)\mathbf{G}_{n1}, \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + h\varphi_1(-hA)\mathbf{G}_{n1} + \frac{h}{c_2}\varphi_2(-hA)(\mathbf{F}_{n2} - \mathbf{F}_{n1}).\end{aligned}$$

El segundo (Método de orden 2 (II)) puede escribirse como

$$\begin{aligned}\mathbf{Y}_{n1} &= \mathbf{y}_n, \\ \mathbf{Y}_{n2} &= \mathbf{y}_n + hc_2\varphi_{1,2}(-hA)\mathbf{G}_{n1}, \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + h\varphi_1(-hA)\left(\mathbf{G}_{n1} + \frac{1}{2c_2}(\mathbf{F}_{n2} - \mathbf{F}_{n1})\right).\end{aligned}$$

En ambos casos,

$$\begin{aligned}\mathbf{G}_{n1} &= \mathbf{F}_{n1} - A\mathbf{y}_n, \\ \mathbf{F}_{nj} &= \mathbf{F}(\mathbf{Y}_{nj}), \quad \text{para } j = 1, 2,\end{aligned}$$

y hemos elegido como nodo $c_2 = \frac{1}{2}$, de modo que el método Runge-Kutta subyacente es el método de Runge de orden 2.

Para el primer método de orden 3 estudiado (Método de orden 3 (I)), hemos tomado $c_2 = \frac{1}{3}$ y $c_3 = \frac{2}{3}$, siguiendo el paralelismo con el método Runge-Kutta clásico de Heun y se escribe como

$$\begin{aligned}\mathbf{Y}_{n1} &= \mathbf{y}_n, \\ \mathbf{Y}_{n2} &= \mathbf{y}_n + hc_2\varphi_{1,2}(-hA)\mathbf{G}_{n1}, \\ \mathbf{Y}_{n3} &= \mathbf{y}_n + \frac{2h}{3}\varphi_{1,3}(-hA)\mathbf{G}_{n1} + \frac{4h}{9c_2}\varphi_{2,3}(-hA)(\mathbf{F}_{n2} - \mathbf{F}_{n1}), \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + h\varphi_1(-hA)\mathbf{G}_{n1} + \frac{3h}{2}\varphi_2(-hA)(\mathbf{F}_{n3} - \mathbf{F}_{n1}),\end{aligned}$$

con

$$\begin{aligned}\mathbf{G}_{n1} &= \mathbf{F}_{n1} - \mathbf{A}\mathbf{y}_n, \\ \mathbf{F}_{nj} &= \mathbf{F}(\mathbf{Y}_{nj}), \quad \text{para } j = 1, 2, 3.\end{aligned}$$

En el segundo método de orden 3 (Método de orden 3 (II)), hemos tomado $c_2 = \frac{1}{2}$ y $c_3 = \frac{3}{4}$, de modo que el método Runge-Kutta subyacente es el que está implementado en la función `ode23.m` de MATLAB. Reescribimos el método como

$$\begin{aligned}\mathbf{Y}_{n1} &= \mathbf{y}_n, \\ \mathbf{Y}_{n2} &= \mathbf{y}_n + hc_2\varphi_{1,2}(-hA)\mathbf{G}_{n1}, \\ \mathbf{Y}_{n3} &= \mathbf{y}_n + c_3h\varphi_{1,3}(-hA)\mathbf{G}_{n1} + \frac{c_3^2h}{c_2}\varphi_{2,3}(-hA)(\mathbf{F}_{n2} - \mathbf{F}_{n1}) \\ &\quad + \gamma c_2h\varphi_{2,2}(-hA)(\mathbf{F}_{n2} - \mathbf{F}_{n1}), \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + h\varphi_1(-hA)\mathbf{G}_{n1} + \frac{h}{\gamma c_2 + c_3}\varphi_2(-hA)(\mathbf{F}_{n3} - \mathbf{F}_{n1} + \gamma(\mathbf{F}_{n2} - \mathbf{F}_{n1})),\end{aligned}$$

donde recordemos que el parámetro γ viene dado en función de c_2 y c_3 por (1.62), luego

$$\gamma = \frac{c_3}{c_2} \left(\frac{3c_3 - 2}{2 - 3c_2} \right),$$

y los vectores \mathbf{F}_{nj} y \mathbf{G}_{n1} están definidos igual que para el método anterior.

Por último, el método de orden 4 estudiado en la Sección 1.3.2 se reescribe como

$$\begin{aligned}\mathbf{Y}_{n1} &= \mathbf{y}_n, \\ \mathbf{Y}_{n2} &= \mathbf{y}_n + hc_2\varphi_{1,2}(-hA)\mathbf{G}_{n1}, \\ \mathbf{Y}_{n3} &= \mathbf{y}_n + \frac{h}{2}\varphi_{1,3}(-hA)\mathbf{G}_{n1} + h\varphi_{2,3}(-hA)(\mathbf{F}_{n2} - \mathbf{F}_{n1}), \\ \mathbf{Y}_{n4} &= \mathbf{y}_n + h\varphi_{1,4}(-hA)\mathbf{G}_{n1} + h\varphi_{2,4}(-hA)(-2\mathbf{F}_{n1} + \mathbf{F}_{n2} + \mathbf{F}_{n3}), \\ \mathbf{Y}_{n5} &= \mathbf{y}_n + \frac{h}{2}\varphi_{1,5}(-hA)\mathbf{G}_{n1} + \frac{h}{4}\varphi_{2,5}(-hA)(-3\mathbf{F}_{n1} + 2\mathbf{F}_{n2} + 2\mathbf{F}_{n3} - \mathbf{F}_{n4}) \\ &\quad + \frac{h}{2}\varphi_{3,5}(-hA)(\mathbf{F}_{n1} - \mathbf{F}_{n2} - \mathbf{F}_{n3} + \mathbf{F}_{n4}) \\ &\quad + \frac{h}{4}\varphi_{2,4}(-hA)(-\mathbf{F}_{n1} + \mathbf{F}_{n2} + \mathbf{F}_{n3} - \mathbf{F}_{n4}) \\ &\quad + h\varphi_{3,4}(-hA)(\mathbf{F}_{n1} - \mathbf{F}_{n2} - \mathbf{F}_{n3} + \mathbf{F}_{n4}), \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + h\varphi_1(-hA)\mathbf{G}_{n1} + h\varphi_2(-hA)(-3\mathbf{F}_{n1} - \mathbf{F}_{n4} + 4\mathbf{F}_{n5}) \\ &\quad + h\varphi_3(-hA)(4\mathbf{F}_{n1} + 4\mathbf{F}_{n4} - 8\mathbf{F}_{n5}),\end{aligned}$$

donde

$$\begin{aligned}\mathbf{G}_{n1} &= \mathbf{F}_{n1} - \mathbf{A}\mathbf{y}_n, \\ \mathbf{F}_{nj} &= \mathbf{F}(\mathbf{Y}_{nj}), \quad \text{para } j = 1, \dots, 4.\end{aligned}$$

La notación de las funciones implementadas en MATLAB a la hora de introducir estas expresiones no es exactamente igual que la que se muestra en la memoria puesto que en dichos programas se han aprovechado los términos repetidos para simplificar los cálculos. No incluimos dichas fórmulas en esta parte para aligerar su lectura.

2.2. Un ejemplo lineal

El primer experimento que vamos a estudiar se restringe a un caso lineal para ilustrar el comportamiento de los métodos de cuadratura exponencial descritos en la Sección 1.2. Consideremos el problema unidimensional lineal no homogéneo tomado de [2]

$$\dot{y}(t) + cy(t) = \sin(t), \quad 0 \leq t \leq \frac{\pi}{2}, \quad y(0) = y_0, \quad (2.3)$$

donde $c \gg 1$. Observemos que este problema consta de una parte lineal rígida debida al tamaño de la constante c y de un término fuente de variación lenta, lo que da lugar a un rápido decaimiento inicial de la solución, habitual en sistemas diferenciales rígidos.

Comenzamos por el cálculo de la solución exacta del problema (2.3) en el tiempo t mediante la fórmula de variación de las constantes

$$y(t) = e^{-ct}y_0 + \int_0^t e^{-c(t-\tau)} \sin(\tau) d\tau = y_0 e^{-ct} + \frac{e^{-ct} + c \sin(t) - \cos(t)}{1 + c^2}, \quad (2.4)$$

donde hemos integrado dos veces por partes, la primera de ellas tomando $u = \sin(\tau)$, $dv = e^{-c(t-\tau)} d\tau$ y la segunda con $u = \cos(\tau)$, $dv = \frac{1}{c} e^{-c(t-\tau)} d\tau$.

Notemos que cuando $c \gg 1$, el comportamiento de la solución (2.4) del problema (2.3) se puede ver como la suma de una fase transitoria dada por el término e^{-ct} , que predomina para valores bajos de t pero que rápidamente pasa a ser despreciable frente a un estado “estacionario” (no podemos considerarlo propiamente estacionario pues se produce una lenta variación periódica) que se mantiene en el tiempo, de modo que $y(t) \sim \frac{\sin(t)}{c} - \frac{\cos(t)}{c^2}$. Ambas fases quedan reflejadas en la Figura 2.1. Podemos ver ese rápido decaimiento inicial en la gráfica de la izquierda, de modo que a partir de $t = 0.05$, la solución toma valores muy cercanos al cero. Sin embargo, reduciendo la escala del eje vertical en un par de órdenes de magnitud, a la derecha se puede observar que la solución oscila periódicamente en torno a $y = 0$, con una amplitud aproximada de una centésima.

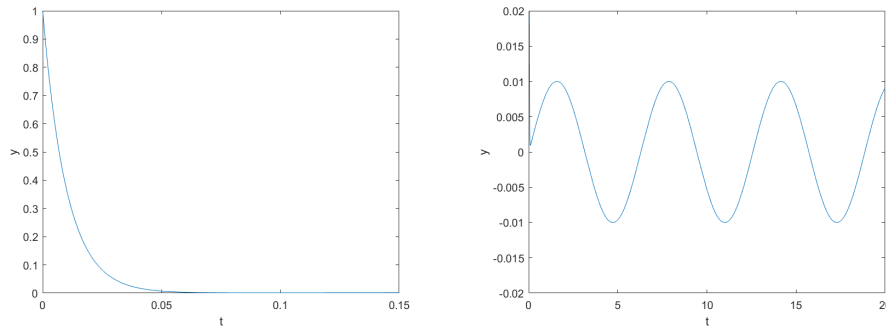


Figura 2.1: Izquierda: fase transitoria de la solución exacta (2.4). Derecha: fase “estacionaria” de la solución exacta (2.4).

Por tanto, a la hora de calcular las aproximaciones numéricas a la solución, será necesario implementar métodos adecuados para ambas componentes de la misma y que, por razones de estabilidad, funcionen para longitudes de paso $h \sim \frac{1}{c}$.

Se han implementado en MATLAB los métodos (1.15), (1.16) y (1.19) estudiados en la Sección 1.2 para el caso lineal no homogéneo. Observemos que, según el Teorema 1.2.1, los dos primeros métodos son, al menos, de orden 1 mientras que la regla de los trapecios exponencial es, al menos, de orden 2.

En este ejemplo concreto, la fórmula de cuadratura exponencial de Euler (1.15) viene dada por

$$y_{n+1} = e^{-hc}y_n + h\varphi_1(-hc)F(t_n), \quad n \geq 0, \quad (2.5)$$

la fórmula de cuadratura exponencial del punto medio (1.16) se escribe como

$$y_{n+1} = e^{-hc}y_n + h\varphi_1(-hc)F\left(t_n + \frac{h}{2}\right), \quad (2.6)$$

y la regla de los trapecios exponencial (1.19) toma la forma

$$y_{n+1} = e^{-hc}y_n + h\varphi_1(-hc)F(t_n) + h\varphi_2(-hc)(F(t_n + h) - F(t_n)). \quad (2.7)$$

Notemos que la fórmula de cada método puede considerarse como una combinación lineal de funciones $\varphi_k(-hc)$ multiplicadas por vectores.

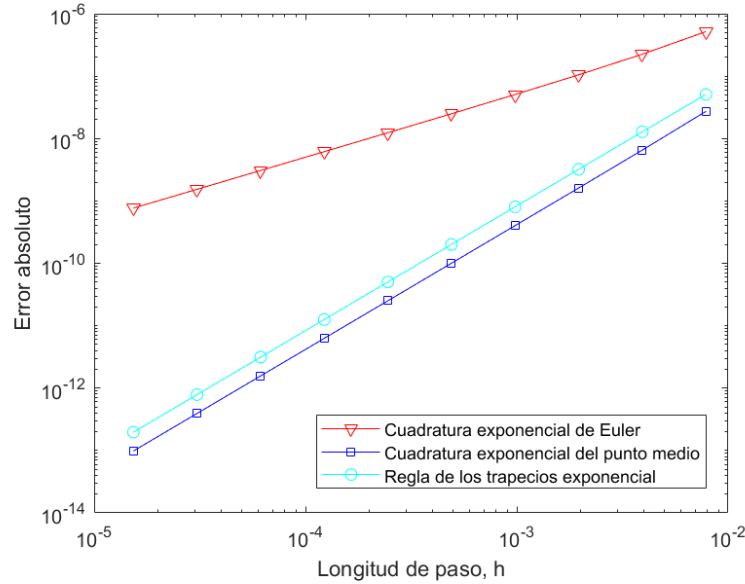


Figura 2.2: Error absoluto en el tiempo $t = \frac{\pi}{2}$ al aproximar la solución del problema (2.3) con $y_0 = 1$ y $c = 100$, en función de la longitud de paso h .

Para visualizar el orden de los métodos mencionados, hemos considerado, como en [2], el problema (2.3) con $y_0 = 1$ y $c = 100$, y se ha elaborado un programa en MATLAB (véase el Apéndice A.1) que calcula tanto la solución exacta del problema dada por (2.4) en el tiempo $t = \frac{\pi}{2}$, como sus aproximaciones numéricas calculadas con los métodos (2.5)-(2.7). Dicho programa muestra gráficamente el error absoluto cometido por cada uno de los métodos en función de la longitud de paso h que se utilice y lo representa en escala doblemente logarítmica, de forma que la pendiente de las líneas obtenidas coincide con el orden del método utilizado. El resultado que se encuentra en la Figura 2.2 ha sido obtenido al utilizar longitudes de paso $h = \frac{\pi}{400}, \frac{\pi}{800}, \frac{\pi}{1600}, \dots, \frac{\pi}{204800}$.

El programa implementado hace uso de la función `phipm.m` de [6] mencionada anteriormente, con tolerancia 10^{-7} , que es la que se toma por defecto en funciones similares y en este ejemplo da buenos resultados.

Dado que el problema concreto que estamos estudiando es escalar, esta función podría considerarse innecesaria pues el coste computacional al calcular la exponencial de un escalar no es importante. Sin embargo, los programas implementados se han realizado con el fin de poder utilizarse para problemas de mayor dimensión, lo que conlleva el cálculo de productos de matrices exponenciales por vectores y el coste aumenta.

La pendiente de la línea que representa el error absoluto correspondiente a la cuadratura exponencial de Euler es 1, coincidiendo con el orden esperado al aplicar el Teorema 1.2.1. Lo mismo ocurre para la regla de los trapecios exponencial que, tal y como se había mencionado, tiene orden 2. Sin embargo, la pendiente de la línea que corresponde al error absoluto cometido al aplicar la cuadratura exponencial del punto medio es 2, indicando que el método es de segundo orden. Esto no supone una contradicción para el Teorema 1.2.1 pues éste da unas condiciones suficientes pero no necesarias para que los métodos considerados tengan un cierto orden. Notemos, por otra parte, que para $A = 0$ se trata de las fórmulas de cuadratura clásicas y es conocido que la regla del punto medio es exacta para polinomios de grado menor o igual que 1, igual que la regla de los trapecios.

2.3. Primer ejemplo semilineal

Consideremos ahora el problema autónomo formado por las siguientes ecuaciones diferenciales ordinarias

$$\dot{u} = -v(1 - \lambda r^2) + cu(1 - r^2), \quad u(0) = u_0, \quad (2.8)$$

$$\dot{v} = u(1 - \lambda r^2) + cv(1 - r^2), \quad v(0) = v_0, \quad (2.9)$$

donde $r^2 = u^2 + v^2$, λ es un parámetro real y $c > 0$. Pasando a coordenadas polares, el sistema equivale a

$$\dot{r} = cr(1 - r^2), \quad (2.10)$$

$$\dot{\theta} = 1 - \lambda r^2, \quad (2.11)$$

siendo $u = r \cos \theta$ y $v = r \sin \theta$. Ahora, teniendo en cuenta que (2.10) es una ecuación diferencial de variables separadas, haciendo el cambio de variable $z = r^2$ y descomponiendo

en fracciones simples, se deduce que

$$r^2(t) = \frac{r_0^2}{r_0^2(1 - e^{-2ct}) + e^{-2ct}}, \quad (2.12)$$

con $r_0 = \sqrt{u(0)^2 + v(0)^2}$. Ahora, teniendo en cuenta (2.12) y llevándolo a (2.11), hacemos el cambio de variable $z = r_0^2(1 - e^{-2ct}) + e^{-2ct}$ y descomponemos de nuevo en fracciones simples para obtener

$$\theta(t) = \theta_0 + (1 - \lambda)t - \frac{\lambda}{2c} \log(r_0^2(1 - e^{-2ct}) + e^{-2ct}), \quad (2.13)$$

donde $\theta_0 = \arccos(u(0)/r(0))$.

Veamos ahora cómo implementar los métodos estudiados en la Sección 1.3.2 para resolver (2.8)-(2.9). Tenemos un sistema de la forma (1.31) con

$$\mathbf{y}(t) = \begin{pmatrix} u(t) \\ v(t) \end{pmatrix}, \quad A = \begin{pmatrix} -c & 1 \\ -1 & -c \end{pmatrix},$$

$$\mathbf{F}(t) = \begin{pmatrix} (\lambda v(t) - cu(t))r^2(t) \\ -(\lambda u(t) + cv(t))r^2(t) \end{pmatrix} = \|\mathbf{y}(t)\|^2 \begin{pmatrix} -c & \lambda \\ -\lambda & -c \end{pmatrix} \mathbf{y}(t),$$

siendo $\|\cdot\|$ la norma euclídea y A una matriz invertible.

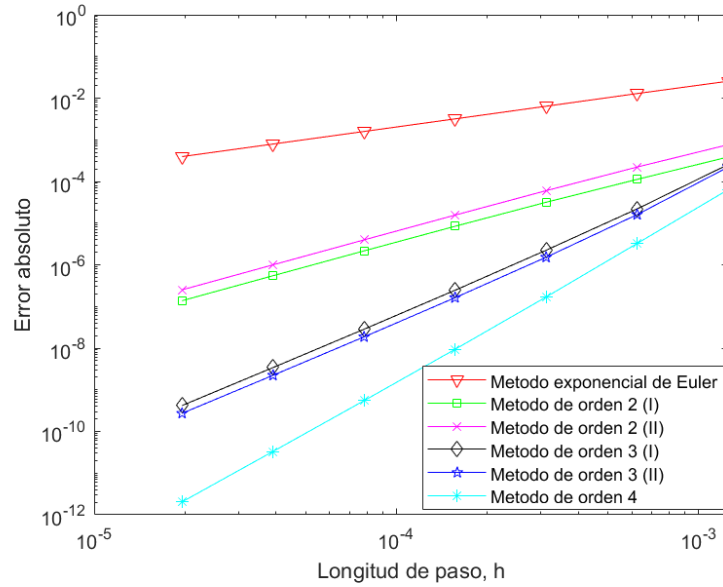


Figura 2.3: Error absoluto en el tiempo $t = 1$ al aproximar la solución del problema (2.8)-(2.9) con $\mathbf{y}_0 = (2, 1)$, $c = 100$ y $\lambda = \frac{1}{2}$, en función de la longitud de paso h .

Con esta notación, se pueden aplicar directamente los métodos Runge-Kutta exponenciales dados por (1.37)-(1.39) estudiados en la Sección 1.3.2. Hemos programado cada uno

de ellos en MATLAB (véase el Apéndice A.2) utilizando de nuevo la función `phipm.m` que se ha explicado de la Sección 2.1.

Igual que en [2], hemos tomado $c = 100$, $\lambda = \frac{1}{2}$, condiciones iniciales $u(0) = 2$ y $v(0) = 1$ y se ha integrado con los distintos métodos hasta $t = 1$. En dicho tiempo se ha medido el error comparando las aproximaciones numéricas calculadas con la solución exacta que se obtiene utilizando (2.12)-(2.13) y pasando de coordenadas polares a coordenadas cartesianas. Los resultados cuando el error se mide con la norma infinito se muestran en la Figura 2.3, donde las pendientes de las líneas obtenidas coinciden con el orden de cada uno de los métodos analizados.

Además, siguiendo [2], en las gráficas de la Figura 2.4 se muestra por separado el error cometido por cada método en la aproximación a la amplitud r (izquierda) y a la fase θ (derecha). Observemos que seguimos obteniendo las mismas pendientes que en la Figura 2.3, pero los errores en la amplitud son alrededor de cuatro órdenes de magnitud menores que los que se cometen al aproximar la fase (el rango de valores del eje vertical no es el mismo en las dos gráficas).

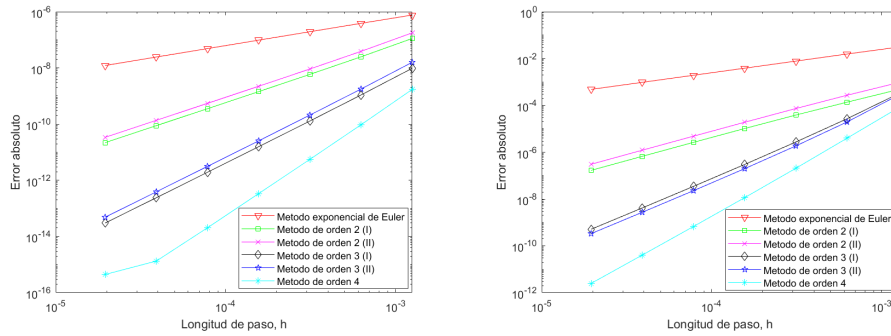


Figura 2.4: Izquierda: error absoluto en el tiempo $t = 1$ al aproximar la amplitud r del problema (2.10)-(2.11), en función de la longitud de paso h . Derecha: error absoluto en el tiempo $t = 1$ al aproximar la fase θ del problema (2.10)-(2.11), en función de la longitud de paso h .

2.4. Integración de una ecuación en derivadas parciales semilineal

El último ejemplo que vamos a estudiar es la ecuación de Burgers

$$\frac{\partial y}{\partial t}(x, t) = \frac{\partial^2 y}{\partial x^2}(x, t) - y(x, t) \frac{\partial y}{\partial x}(x, t) + \Phi(x, t), \quad x \in [0, 1], \quad t \in [0, 1], \quad (2.14)$$

con condiciones frontera de tipo Dirichlet homogéneas. Al igual que en [1], el término $\Phi(x, t)$ se ha elegido de forma que la solución exacta del problema sea

$$y(x, t) = \frac{ax(1-x)}{1 + (10t-3)^2}, \quad (2.15)$$

siendo a un parámetro real que, como en [1], hemos tomado $a = 110$. Por tanto, el término $\Phi(x, t)$ viene dado por

$$\begin{aligned}\Phi(x, t) &= \frac{1}{1 + (10t - 3)^2} \left(2a + \frac{ax(1-x)}{1 + (10t - 3)^2} (a(1-2x) - 20(10t - 3)) \right) \\ &= \frac{2a + a(1-2x)y(x, t) - 20(10t - 3)y(x, t)}{1 + (10t - 3)^2}.\end{aligned}$$

A diferencia de los ejemplos que hemos analizado hasta el momento, en este caso existe una dependencia espacial de la solución $y(x, t)$. Por tanto, para poder utilizar los métodos Runge-Kutta exponenciales desarrollados, previamente se debe llevar a cabo una discretización espacial del problema.

Para ello, definimos una red uniforme de puntos $x_j = j\Delta x$, $j = 1, \dots, J-1$, con $\Delta x = \frac{1}{J}$ y, sobre esta red, denotamos por $Y_j(t)$, $j = 1, \dots, J-1$, a cada una de las funciones que aproximan a la solución en los nodos de la red espacial $Y_j(t) \simeq y(x_j, t)$.

En cada punto de la red x_j , se aproximan las derivadas espaciales que aparecen en (2.14) mediante diferencias finitas centradas de segundo orden

$$\begin{aligned}\frac{\partial y}{\partial x}(x_j, t) &= \frac{y(x_{j+1}, t) - y(x_{j-1}, t)}{2\Delta x} + O(\Delta x)^2, \\ \frac{\partial^2 y}{\partial x^2}(x_j, t) &= \frac{y(x_{j+1}, t) - 2y(x_j, t) + y(x_{j-1}, t)}{\Delta x^2} + O(\Delta x)^2,\end{aligned}$$

y reemplazando en las expresiones anteriores $y(x_{j-1}, t)$, $y(x_j, t)$, $y(x_{j+1}, t)$ por $Y_{j-1}(t)$, $Y_j(t)$, $Y_{j+1}(t)$, respectivamente, e introduciendo estas aproximaciones en (2.14), para $j = 1, \dots, J-1$, tenemos que

$$\frac{d}{dt}Y_j(t) = \frac{Y_{j+1}(t) - 2Y_j(t) + Y_{j-1}(t)}{\Delta x^2} - Y_j(t) \frac{Y_{j+1}(t) - Y_{j-1}(t)}{2\Delta x} + \Phi(x_j, t). \quad (2.16)$$

Notemos que en (2.16) aparecen $Y_0(t)$ e $Y_J(t)$ que, teniendo en cuenta que las condiciones frontera del problema son de tipo Dirichlet homogéneas, es decir, $y(0, t) = y(1, t) = 0$, son funciones idénticamente nulas.

Ahora, como la solución exacta es un polinomio en x de grado 2, no va a haber error espacial en nuestras aproximaciones. Esto supone una ventaja importante a la hora de analizar nuestros experimentos pues no va a haber ningún error que contamine aquel que es producido por los integradores temporales que hemos desarrollado.

Observemos que hemos obtenido un sistema semilineal de ecuaciones diferenciales ordinarias de la forma (1.31) con

$$A = J^2 \begin{pmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix},$$

con término no lineal $\mathbf{F}(t, \mathbf{Y}(t)) = [F_1(t, \mathbf{Y}(t)), \dots, F_{J-1}(t, \mathbf{Y}(t))]^T$,

$$F_j(t, \mathbf{Y}(t)) = \frac{J}{2} Y_j(t) (Y_{j-1}(t) - Y_{j+1}(t)) + \frac{2a + Y_j(t) (a(1 - 2x_j) - 20(10t - 3))}{1 + (10t - 3)^2}.$$

De nuevo, hay que tener en cuenta que $Y_0(t) = Y_J(t) \equiv 0$. La j -ésima componente del vector inicial se obtiene a partir de (2.15) evaluando en $t = 0$ y resulta

$$Y_j(0) = \frac{ax_j(1 - x_j)}{10},$$

En estas condiciones, podemos usar los métodos semilineales estudiados en la Sección 1.3.2 para aproximar la solución del problema (2.14) tras su discretización espacial y, siguiendo [1], comenzaremos tomando $J = 512$. Para ello, se ha elaborado un programa en MATLAB (véase el Apéndice A.3) que, igual que en el ejemplo anterior, representa gráficamente el error absoluto cometido por cada uno de los métodos al aproximar la solución del problema en $t = 1$ en función de la longitud de paso elegida h , con valores $h = 2^{-s}$ con $s = 9, \dots, 15$. Los resultados obtenidos pueden observarse en la Figura 2.5 y, al igual que en los ejemplos anteriores, la pendiente de las líneas resultantes coincide con el orden esperado del método salvo para los puntos asociados a los valores más pequeños de h con el método de orden 4, que producen errores tan pequeños que pueden considerarse como errores de redondeo.

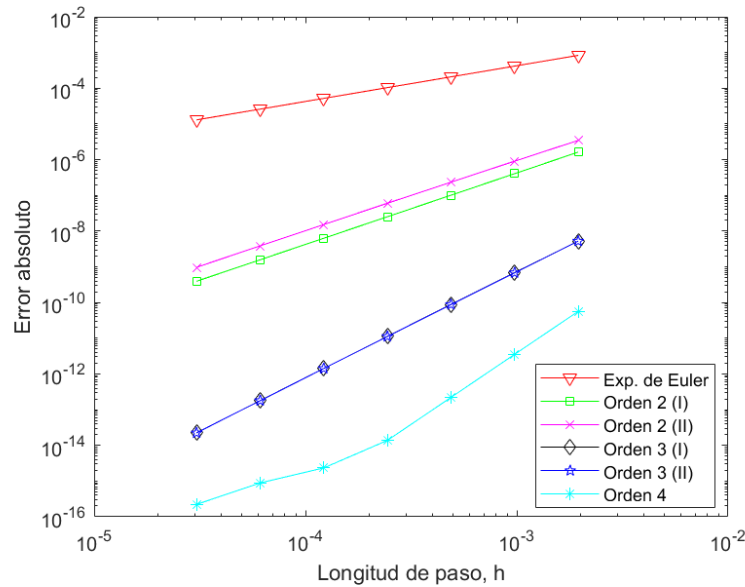


Figura 2.5: Error absoluto en el tiempo $t = 1$ al aproximar la solución del problema (2.14) discretizado espacialmente con $J = 512$, en función de la longitud de paso h .

Además, en el mismo script también se ha implementado la representación gráfica de los errores absolutos al aproximar la solución en el tiempo $t = 1$ en función del tiempo de CPU empleado para ello. Los resultados se encuentran en la Figura 2.6 y podemos observar que cuanto menor es el error generado por cada método, mayor es el tiempo de ejecución necesario, verificándose que las pendientes de las líneas obtenidas para cada método son aproximadamente las inversas de las pendientes que se obtuvieron en la Figura 2.5. Pese a ello, en todos los casos se han conseguido los resultados en tiempos razonables.

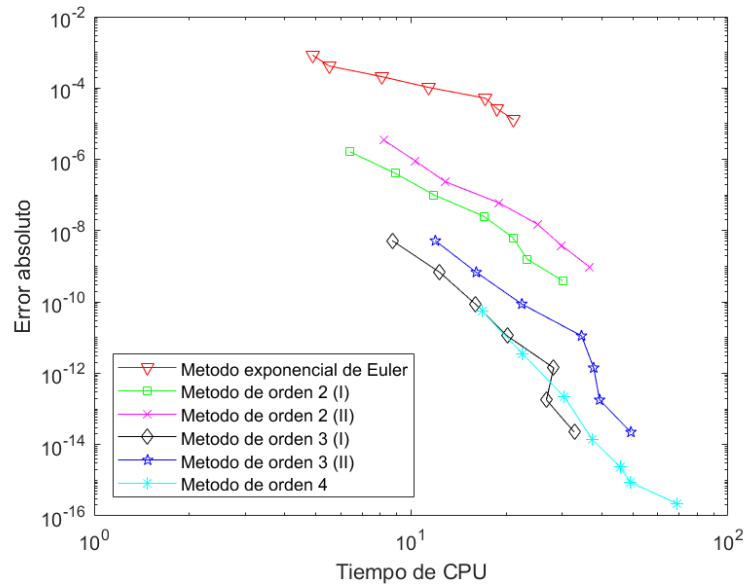


Figura 2.6: Error absoluto en el tiempo $t = 1$ al aproximar la solución del problema (2.14) discretizado espacialmente con $J = 512$, en función del tiempo de CPU empleado.

Observemos que un mayor orden del método va ligado a errores menores, pero también a un tiempo de ejecución mayor. Dentro de los métodos de orden 2, a la vista de las gráficas obtenidas observamos que el Método de orden 2 (II) produce errores menores con tiempos de CPU menores que el (I), por lo que para este ejemplo concreto resulta más adecuado.

Por el mismo motivo, el Método de orden 3 (I) es más eficiente que el (II) y, además, sigue la misma tendencia que el método de orden 4, llegando en algún caso a errores menores para el mismo número de pasos N .

En este caso estamos trabajando con matrices 511×511 , lo cual pone de manifiesto la importancia de calcular los productos de matrices exponenciales por vectores de una forma eficiente. En los casos anteriores, trabajamos ejemplos en los que era menos necesario el uso de la función `hipm.m`, pero para situaciones en las que la dimensión de la matriz A cobra mayor importancia, el costo computacional se ve reducido de forma notable

usando esta función. Además, recalquemos que en este ejemplo, la función `phipm.m` ha aprovechado la simetría de la matriz A pero también ha requerido una menor tolerancia (de orden 10^{-14}) para su correcto funcionamiento.

Ahora, si modificamos la red espacial y tomamos $J = 64$, para las mismas longitudes de paso h que antes, podemos observar que el tamaño de los errores se mantiene pero el tiempo de CPU necesario para la ejecución de cada método se ve reducido, es decir, obtenemos gráficas muy similares a las anteriores pero con una traslación horizontal en el tiempo de cálculo. Dicho resultado era esperable pues, como se ha comentado antes, la semidiscretización realizada no produce ningún error en espacio y la dimensión del problema se ha reducido considerablemente. Las gráficas obtenidas se muestran en la Figura 2.7.

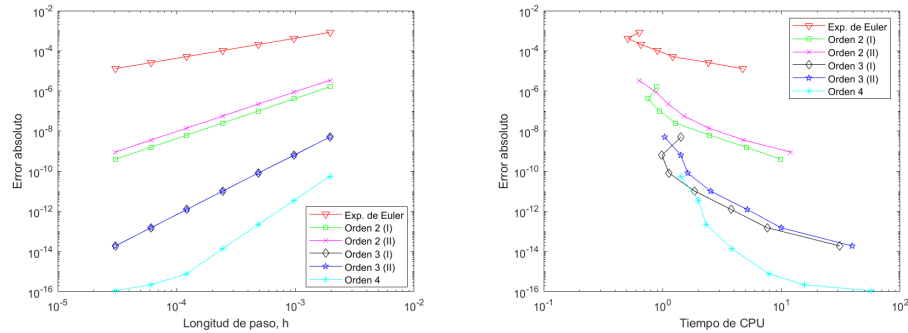


Figura 2.7: Izquierda: Error absoluto en el tiempo $t = 1$ al aproximar la solución del problema (2.14) discretizado espacialmente con $J = 64$, en función de la longitud de paso h . Derecha: Error absoluto en el tiempo $t = 1$ al aproximar la solución del problema (2.14) discretizado espacialmente con $J = 64$, en función del tiempo de CPU empleado.

Notemos que hemos podido utilizar las mismas longitudes de paso h para ambos valores de J y que los métodos exponenciales utilizados han mostrado el orden esperado. En un método Runge-Kutta explícito clásico, si J aumenta, es necesario disminuir la longitud de paso h por cuestiones de estabilidad. Esto se debe a que los autovalores de la matriz $-A$ son proporcionales a $-J^2$ y éstos tienden a $-\infty$ cuando J tiende a infinito. A su vez, para que la longitud de paso h de un método Runge-Kutta explícito sea adecuada, es necesario que el producto de h por el radio espectral de A esté dentro de la región de estabilidad de dicho método. Es decir, h debe ser proporcional a $\Delta x^2 = \frac{1}{J^2}$. Sin embargo, los métodos exponenciales explícitos que se han utilizado satisfacen unas condiciones que a los métodos clásicos no se les han exigido: las condiciones de orden rígidas. Como consecuencia, los métodos exponenciales tienen una región de estabilidad mayor y podemos permitirnos tomar las mismas longitudes de paso h para redes espaciales más finas.

Insistimos en recalcar la ausencia de error espacial en este problema concreto, que nos permite observar que el tamaño de los errores temporales puede ser muy pequeño. Si tuviéramos que hacer este mismo estudio sobre un problema con error espacial, aunque se tomaran valores de h muy bajos, se verían mayoritariamente los errores espaciales y no

podríamos apreciar bien los errores temporales, que son los que se deben analizar para evaluar los integradores temporales sobre los que nos centramos en este trabajo.

Para terminar, hemos probado el método clásico Runge-Kutta explícito de orden 4 sobre este problema para comparar sus resultados con los obtenidos mediante los métodos exponenciales. Lo que se ha observado es que en el rango de valores de h en los que el método Runge-Kutta es estable, los errores obtenidos son del mismo orden que los que se han dado para los métodos exponenciales y, además, el tiempo de CPU es considerablemente menor que los que hemos necesitado para el cálculo de los productos de matrices exponenciales por vectores. Las gráficas correspondientes se encuentran en la Figura 2.8.

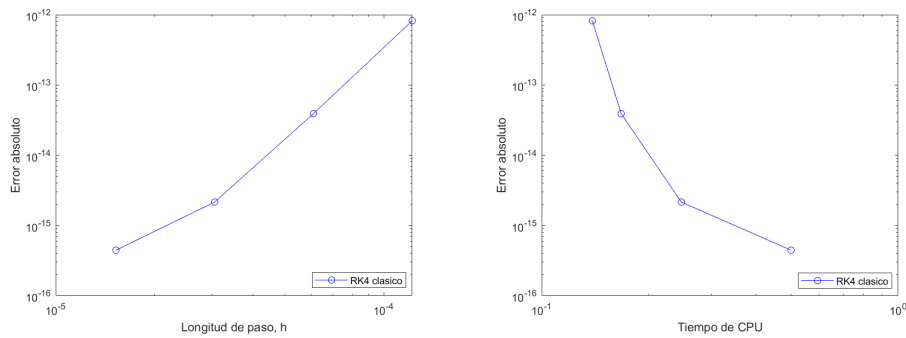


Figura 2.8: Izquierda: Error absoluto en el tiempo $t = 1$ al aproximar la solución del problema (2.14) discretizado espacialmente con $J = 64$, en función de la longitud de paso h . Derecha: Error absoluto en el tiempo $t = 1$ al aproximar la solución del problema (2.14) discretizado espacialmente con $J = 64$, en función del tiempo de CPU empleado.

Podría pensarse entonces que los métodos clásicos son preferibles frente a los exponenciales, sin embargo, la cuestión radica en la estabilidad. Con $J = 64$, se ha necesitado que h fuese menor o igual que 2^{-13} para que el método clásico funcionase correctamente y no se ha podido probar para $J = 512$, pues necesita valores tan pequeños de h que no ha sido factible. Ahora, si volvemos sobre los métodos exponenciales, vemos que con $J = 512$ hemos podido utilizar longitudes de paso h con valores de hasta 2^{-9} pero podríamos haberlos escogido incluso más grandes, en particular con los métodos de orden más alto.

En el problema semidiscretizado que hemos estudiado en este apartado, ya se ha mencionado que no hay error proveniente de la discretización espacial, pero es común que sí que exista ese error en otros ejemplos y, cuanto más fina sea la red de nodos que se cree, menor será éste. Es decir, es posible que la elección de J repercuta en el error y no podamos tomar una cantidad tan baja de nodos como para que el método clásico funcione correctamente. De ahí viene la importancia de los métodos exponenciales, pues aunque necesiten más tiempo de CPU y los errores obtenidos puedan ser similares a los que tendríamos con un método clásico, nos permiten abordar problemas que, por cuestiones de estabilidad, serían inabarcables para los otros.

Capítulo 3

Conclusiones y posibles aplicaciones

En este último capítulo, concluimos el trabajo haciendo un breve resumen de problemas reales a los que se pueden aplicar los integradores exponenciales. El objetivo del apartado no es la visualización de los resultados obtenidos, sino dar una idea de algunas aplicaciones que se han encontrado en la literatura. Generalmente, se tratará de problemas para los que hay una o varias ecuaciones de evolución en las que aparece el operador laplaciano y, tras su semidiscretización espacial, se tiene un término no lineal.

Un primer ejemplo de aplicación de los métodos exponenciales explícitos es la evolución del negativo de una película fotográfica. Tal y como se explica en [4], es posible considerar un modelo macroscópico que describa este proceso químico mediante la evolución de las distintas funciones de densidad que intervienen en el mismo. Sin entrar en explicaciones sobre el significado de cada una de las variables que aparecen, el problema es el siguiente

$$\begin{aligned}\frac{\partial R}{\partial t} &= D_R \Delta R - f_{dev}(R, P, S, P^*)E(x), \quad (x = (x_1, x_2, x_3)), \\ \frac{\partial T}{\partial t} &= D_T \Delta T + f_{dev}(R, P, S, P^*)E(x) - k_1 TC, \\ \frac{\partial C}{\partial t} &= -k_1 TC, \\ \frac{\partial Y}{\partial t} &= k_1 TC, \\ \frac{\partial P}{\partial t} &= D_P \Delta P + k_1 TC - k_2 PS + k_3 P^*, \\ \frac{\partial P^*}{\partial t} &= k_2 PS - k_3 P^*, \\ \frac{\partial S}{\partial t} &= -k_4 S^{-1/2} f_{dev}(R, P, S, P^*)E(x) - k_2 PS + k_3 P^*,\end{aligned}$$

donde $E(x)$ es una función no negativa dada; D_R , D_T , D_P y k_i , $i = 1, 2, 3$, son constantes positivas y la función f_{dev} no es conocida exactamente, por lo que debe adecuarse según los resultados experimentales.

En [4], se hace una simplificación del problema, considerando una reacción en la que sólo intervienen dos reactivos, A y B , y se toma $f_{dev} = \gamma$, constante. Se tienen entonces las ecuaciones

$$\begin{aligned}\frac{\partial A}{\partial t} &= D\Delta A - kAB + \gamma E(x), \\ \frac{\partial B}{\partial t} &= -kAB.\end{aligned}$$

Considerando el caso unidimensional, la primera de las ecuaciones tiene la forma

$$\frac{\partial A}{\partial t} = D \frac{\partial^2 A}{\partial x^2} A - kAB + \gamma E(x), \quad 0 \leq x \leq L, \quad t > 0.$$

Las condiciones iniciales del problema son

$$\begin{aligned}A(x, 0) &= A_0(x), \quad 0 \leq x \leq L \quad (A_0 \geq 0), \\ B(x, 0) &= B_0(x), \quad 0 \leq x \leq L \quad (B_0 \geq 0),\end{aligned}$$

y, para $t > 0$, las condiciones frontera son de tipo Dirichlet homogéneas

$$\begin{aligned}A(0, t) &= 0, \\ A(L, t) &= 0.\end{aligned}$$

Pese a que en [4] no se aborda el problema utilizando los métodos expuestos en este trabajo, se trata de dos ecuaciones en derivadas parciales con un laplaciano de forma que, tras la discretización espacial, obtendremos un sistema semilineal de ecuaciones diferenciales ordinarias y podría resolverse numéricamente mediante integradores exponenciales.

Otro ejemplo de aplicación es el modelo matemático que describe la calidad del aire y cuyas funciones incógnita son las distintas concentraciones de sustancias químicas concretas. Así pues, el siguiente modelo tiene como objetivo la predicción de la evolución de la contaminación del aire según los cambios meteorológicos y de las fuentes contaminantes (véase [3]).

El modelo general que rige la contaminación existente en un espacio tridimensional Ω se obtiene a partir de las ecuaciones de conservación de la masa de cada una de las sustancias químicas de interés. Se tiene entonces un sistema parabólico no lineal de ecuaciones en derivadas parciales que viene dado por

$$\frac{\partial c_i}{\partial t} + \nabla \cdot (\vec{u}c_i) = \nabla \cdot (K\nabla c_i) + f_i(c_1, \dots, c_p), \quad 1 \leq i \leq p,$$

con $x \in \Omega$ y $t > 0$. En estas ecuaciones, el vector \vec{u} describe el campo de la velocidad del aire, K es una matriz de difusión y f_i es la tasa de variación de la concentración de la especie i . Las concentraciones de cada especie química vienen dadas por c_i y son las incógnitas del problema a tratar.

Así, añadiendo unas condiciones iniciales y bajo las condiciones frontera pertinentes, este sistema puede resolverse numéricamente tras su discretización espacial utilizando los métodos exponenciales estudiados en este trabajo. Como se ha mencionado al comienzo del capítulo, el objetivo de este apartado no es la evaluación de los métodos por lo que

se puede consultar el Capítulo 11 de [3], donde se analiza el problema con más detalle a través de métodos alternativos a los que forman este trabajo.

En general, dentro de los estudios relacionados con la creación de modelos matemáticos para la predicción de la evolución de un sistema, se pueden encontrar problemas que, pese a describir realidades que no parecen tener ninguna relación entre sí, pueden tratarse con los métodos descritos. En aquellos casos en los que el objeto de análisis es, por ejemplo, la evolución de una reacción química o el desarrollo de un sistema regido por una ecuación del calor, es común llegar a un problema en el que la discretización espacial conduce a un sistema de ecuaciones diferenciales ordinarias que puede resolverse numéricamente con integradores exponenciales.

A modo de conclusión, en este trabajo hemos sintetizado la información disponible sobre los métodos Runge-Kutta exponenciales explícitos, tratando de comprender su relevancia frente otros tipos de métodos numéricos por su estabilidad y precisión. Además, las implementaciones alternativas al cálculo de matrices exponenciales sin necesidad de obtener las mismas explícitamente hacen que, aunque su coste computacional es alto, éste se vea reducido considerablemente. Si también tenemos en cuenta el amplio rango de aplicaciones para las que se pueden utilizar estos métodos, pese a que existen maneras alternativas de resolución, es entendible que haya un interés cada vez mayor en su desarrollo.

Bibliografía

- [1] M.P. Calvo y A.M. Portillo *Variable step implementation of ETD methods for semilinear problems*, Applied Mathematics and Computation 196 (2008), pp. 627-637.
- [2] S. M. Cox y P. C. Matthews, *Exponential Time Differencing for Stiff Systems*, Journal of Computational Physics 176 (2002), pp. 430-455.
- [3] A. Friedman, *Mathematics in industrial problems, Part 3*, Springer (1990), pp. 116-125.
- [4] A. Friedman, W. Littman, *Industrial Mathematics: A Course in Solving Real-World Problems*, SIAM (1994), pp. 69-86.
- [5] E. Hairer, S. P. Norsett y G. Wanner, *Solving ordinary differential equations I: nonstiff problems*, Springer (1993).
- [6] J. Niesen y W. M. Wright, *Algorithm 919: A Krylov Subspace Algorithm for Evaluating the φ -Functions Appearing in Exponential Integrators*, ACM Transactions on Mathematical Software 38(3), Article 22 (2012).
- [7] A. Ostermann y M. Hochbruck, *Explicit exponential Runge-Kutta methods for semilinear parabolic problems*, SIAM Journal on Numerical Analysis 43 (2005), pp. 1069-1090.
- [8] A. Ostermann y M. Hochbruck, *Exponential integrators*, Acta Numérica 19 (2010), pp. 209-286.

Apéndice A

Programas en MATLAB

A.1. Programa en MATLAB para el ejemplo lineal

En este script se encuentra tanto el programa principal para generar las gráficas de la Sección 2.2 como las funciones que se han elaborado para aproximar la solución del problema (2.3) mediante los métodos desarrollados en la Sección 1.2:

- La cuadratura exponencial de Euler (1.15) en la función `euler_lineal(y0, N, h, c, F)`.
- La cuadratura exponencial del punto medio (1.16) en `pmedio_lineal(y0, N, h, c, F)`.
- La regla de los trapecios exponencial (1.19) en `trapecios_lineal(y0, N, h, c, F)`.

Como se ha explicado a lo largo del trabajo, dichas funciones utilizan la función `phipm.m` tomada de la referencia [6].

```
1 % CASO LINEAL
2
3 % Comenzamos por definir los parametros de entrada de
4   nuestras funciones y que definen el problema a estudiar.
5 y0 = 1;
6 c = 100;
7 F = @(t) sin(t);
8
9 % Primero, mostramos graficamente la solucion exacta del
10  problema
11 ysol = @(t) (y0*exp(-t*c)+(exp(-t*c)+c*sin(t)-cos(t))/(1+c
12   ^2));
13 figure('Name', 'Solucion exacta');
14 fplot(ysol, [0,0.15])
15 xlabel('t')
16 ylabel('y')
```

```
14 pause
15
16 figure('Name', 'Solucion exacta');
17 fplot(ysol, [0,20])
18 xlabel('t')
19 ylabel('y')
20 axis([0,20,-0.02,0.02])
21 pause
22
23 % Ahora, calculamos los errores absolutos y relativos que se
      cometen para distintas longitudes de paso h dentro de
      cada uno de los metodos lineales estudiados en el tiempo
      t = pi/2, donde Nexp es el numero de experimentos a
      realizar.
24 Nexp = 10;
25 N = zeros(1,Nexp);
26 N(1) = 200;
27 for i = 2:Nexp
28     N(i) = 2*N(i-1);
29 end
30 H = (pi/2)./N;
31
32 % Creamos tres vectores con los valores de las
      aproximaciones en t = pi/2, uno por cada metodo.
33 yn_euler = zeros(1,Nexp);
34 yn_pmedio = zeros(1,Nexp);
35 yn_trapecios = zeros(1,Nexp);
36 for i = 1:Nexp
37     clear yn
38     [yn] = euler_lineal(y0, N(i), H(i), c, F);
39     yn_euler(i) = yn(N(i)+1);
40     clear yn
41     [yn] = pmedio_lineal(y0, N(i), H(i), c, F);
42     yn_pmedio(i) = yn(N(i)+1);
43     clear yn
44     [yn] = trapecios_lineal(y0, N(i), H(i), c, F);
45     yn_trapecios(i) = yn(N(i)+1);
46 end
47
48 % El valor que toma la solucion exacta en t = pi/2 es
49 t = pi/2;
50 clear y
51 y = ysol(t);
52
53 % El error absoluto cometido en cada caso es:
54 e_euler = abs(yn_euler-y);
55 e_pmedio = abs(yn_pmedio-y);
```

```
56 e_trapecios = abs(yn_trapecios-y);
57
58 % Representamos graficamente el error cometido por cada
    metodo en el tiempo t = pi/2 en funcion de la longitud de
    paso h.
59 figure('Name','Errores caso lineal');
60 loglog(H, e_euler, 'rv-')
61 xlabel('Longitud de paso, h')
62 ylabel('Error absoluto')
63 pause
64 hold on
65 loglog(H, e_pmedio, 'bs-')
66 pause
67 loglog(H, e_trapecios, 'co-')
68 legend('Cuadratura exponencial de Euler', 'Cuadratura
    exponencial del punto medio', 'Regla de los trapecios
    exponencial')
69 pause
70
71
72 % Las funciones que se han utilizado para realizar las
    correspondientes aproximaciones con cada metodo son las
    siguientes:
73
74 % CUADRATURA EXPONENCIAL DE EULER
75
76 function [yn] = euler_lineal(y0, N, h, c, F)
77
78 % Los parametros de entrada de esta funcion son el valor
    inicial del problema y0, el numero de iteraciones que
    queremos realizar N, la longitud de paso deseada h, el
    valor que toma la constante c del problema y el termino
    fuente F del problema, que debera ser introducido en el
    siguiente formato: @(t) F(t).
79
80 % El output sera una matriz Lx(N+1) yn cuyas columnas son
    las aproximaciones de la solucion en cada tiempo tn = nh,
    n<N+1, dadas por la cuadratura exponencial de Euler,
    siendo L la dimension de la solucion y del problema.
81
82 L = length(y0);
83 yn = zeros(L,N+1);
84 yn(:,1) = y0;
85 t = 0;
86 tol = 1e-7;
87 symm = 'false';
88
```

```

89 for n = 2:N+1
90     y = yn(:,n-1);
91     u = [y, F(t)];
92     [yn(:,n), ~] = phipm(h, -c, u, tol, symm);
93     t = t + h;
94 end
95 end
96
97 % CUADRATURA EXPONENCIAL DEL PUNTO MEDIO
98
99 function [yn] = pmedio_lineal(y0, N, h, c, F)
100
101 % Los parametros de entrada de esta funcion son el valor
    inicial del problema y0, el numero de iteraciones que
    queremos realizar N, la longitud de paso deseada h, el
    valor que toma la constante c del problema y el termino
    fuente F del problema, que debera ser introducido en el
    siguiente formato: @(t) F(t).
102
103 % El output sera una matriz Lx(N+1) yn cuyas columnas son
    las aproximaciones de la solucion en cada tiempo tn = nh,
    n<N+1, dadas por la cuadratura exponencial del punto
    medio, siendo L la dimension de la solucion y del
    problema.
104
105 L = length(y0);
106 yn = zeros(L,N+1);
107 yn(:,1) = y0;
108 t = 0;
109 tol = 1e-7;
110 symm = 'false';
111 h_2 = 0.5*h;
112
113 for n = 2:N+1
114     y = yn(:,n-1);
115     u = [y, F(t+h_2)];
116     [yn(:,n), ~] = phipm(h, -c, u, tol, symm);
117     t = t + h;
118 end
119 end
120
121 % REGLA DE LOS TRAPECIOS EXPONENCIAL
122
123 function [yn] = trapecios_lineal(y0, N, h, c, F)
124
125 % Los parametros de entrada de esta funcion son el valor
    inicial del problema y0, el numero de iteraciones que

```



```

126     queremos realizar N, la longitud de paso deseada h, el
127     valor que toma la constante c del problema y el termino
        fuente F del problema, que debera ser introducido en el
        siguiente formato: @(t) F(t).
128
129     % El output sera una matriz Lx(N+1) yn cuyas columnas son
        las aproximaciones de la solucion en cada tiempo tn = nh,
        n<N+1, dadas por la regla de los trapecios exponencial,
        siendo L la dimension de la solucion y del problema.
128
129     L = length(y0);
130     yn = zeros(L,N+1);
131     yn(:,1) = y0;
132     t = 0;
133     tol = 1e-7;
134     symm = 'false';
135     h_1 = 1/h;
136
137     for n = 2:N+1
138         y = yn(:,n-1);
139         u = [y, F(t), h_1*(F(t+h)-F(t))];
140         [yn(:,n), ~] = phipm(h, -c, u, tol, symm);
141         t = t + h;
142     end
143     end

```

A.2. Programa principal en MATLAB para el primer ejemplo semilineal

A continuación se encuentra el script con el programa principal elaborado para el análisis del problema (2.8)-(2.9) desarrollado en la Sección 2.3. Las funciones que aparecen para aproximar la solución del problema mediante los métodos Runge-Kutta exponenciales estudiados en la Sección 1.3.2 se encuentran en el Apéndice A.4.

```

1  % CASO SEMILINEAL 1
2
3  % Comenzamos por definir los parametros de entrada de
        nuestras funciones y que definen el problema a estudiar.
4  y0 = [2;1];
5  c = 100;
6  A = [-c,1;-1,-c];
7  lambda = 0.5;
8  F = @(~,y) (y(1)^2+y(2)^2)*[-c,lambda;-lambda,-c]*y;
9
10 % Ahora, calculamos los errores absolutos que se cometen
        para distintas longitudes de paso h dentro de cada uno de

```

```

    los metodos en el tiempo t = 1, donde Nexp es el numero
    de experimentos a realizar.
11 Nexp = 7;
12 N = zeros(1,Nexp);
13 N(1) = 800;
14 for i = 2:Nexp
15     N(i) = 2*N(i-1);
16 end
17 H = 1./N;
18
19 % Creamos matrices con los valores de las aproximaciones en
    t=1, una por cada metodo.
20 yn_eu = zeros(2,Nexp);
21 yn_21 = zeros(2,Nexp);
22 yn_22 = zeros(2,Nexp);
23 yn_31 = zeros(2,Nexp);
24 yn_32 = zeros(2,Nexp);
25 yn_4 = zeros(2,Nexp);
26 % Estas matrices muestran las aproximaciones a la solucion
    en coordenadas polares de modo que la primera fila
    corresponde a la amplitud r y la segunda es la fase theta
27 polares_eu = zeros(2,Nexp);
28 polares_21 = zeros(2,Nexp);
29 polares_22 = zeros(2,Nexp);
30 polares_31 = zeros(2,Nexp);
31 polares_32 = zeros(2,Nexp);
32 polares_4 = zeros(2,Nexp);
33
34 for i = 1:Nexp
35     clear yn;
36     [yn] = euler(y0, N(i), H(i), A, F);
37     yn_eu(:,i) = yn(:,N(i)+1);
38     polares_eu(1,i) = sqrt(dot(yn_eu(:,i),yn_eu(:,i)));
39     polares_eu(2,i) = acos(yn_eu(1,i)/polares_eu(1,i));
40     clear yn;
41     c2 = 0.5;
42     [yn] = orden2_1(y0, N(i), H(i), A, F, c2);
43     yn_21(:,i) = yn(:,N(i)+1);
44     polares_21(1,i) = sqrt(dot(yn_21(:,i),yn_21(:,i)));
45     polares_21(2,i) = acos(yn_21(1,i)/polares_21(1,i));
46     clear yn;
47     [yn] = orden2_2(y0, N(i), H(i), A, F, c2);
48     yn_22(:,i) = yn(:,N(i)+1);
49     polares_22(1,i) = sqrt(dot(yn_22(:,i),yn_22(:,i)));
50     polares_22(2,i) = acos(yn_22(1,i)/polares_22(1,i));
51     clear yn;
52     c2 = 1/3;

```

```

53     [yn] = orden3_1(y0, N(i), H(i), A, F, c2);
54     yn_31(:,i) = yn(:,N(i)+1);
55     polares_31(1,i) = sqrt(dot(yn_31(:,i),yn_31(:,i)));
56     polares_31(2,i) = acos(yn_31(1,i)/polares_31(1,i));
57     clear yn;
58     c2 = 0.5;
59     c3 = 0.75;
60     [yn] = orden3_2(y0, N(i), H(i), A, F, c2, c3);
61     yn_32(:,i) = yn(:,N(i)+1);
62     polares_32(1,i) = sqrt(dot(yn_32(:,i),yn_32(:,i)));
63     polares_32(2,i) = acos(yn_32(1,i)/polares_32(1,i));
64     clear yn;
65     [yn] = orden4(y0, N(i), H(i), A, F);
66     yn_4(:,i) = yn(:,N(i)+1);
67     polares_4(1,i) = sqrt(dot(yn_4(:,i),yn_4(:,i)));
68     polares_4(2,i) = acos(yn_4(1,i)/polares_4(1,i));
69 end
70
71 % El valor que toma la solucion exacta en t = 1 es
72 t = 1;
73 clear y;
74 y = zeros(2,1);
75 e2c = exp(-2*c);
76 r2_0 = y0(1)^2+y0(2)^2;
77 theta0 = acos(y0(1)/sqrt(r2_0));
78 r2 = r2_0/(r2_0*(1-e2c)+e2c);
79 theta = theta0 + (1-lambda)-lambda/(2*c)*log(r2_0*(1-e2c)+
    e2c);
80 y(1) = sqrt(r2)*cos(theta);
81 y(2) = sqrt(r2)*sin(theta);
82 polares = [sqrt(r2); theta];
83
84 % El error absoluto cometido en cada caso es:
85 e_eu = zeros(1,Nexp);
86 e_21 = zeros(1,Nexp);
87 e_22 = zeros(1,Nexp);
88 e_31 = zeros(1,Nexp);
89 e_32 = zeros(1,Nexp);
90 e_4 = zeros(1,Nexp);
91
92 for n = 1:Nexp
93     e_eu(n) = norm(yn_eu(:,n)-y, "inf");
94     e_21(n) = norm(yn_21(:,n)-y, "inf");
95     e_22(n) = norm(yn_22(:,n)-y, "inf");
96     e_31(n) = norm(yn_31(:,n)-y, "inf");
97     e_32(n) = norm(yn_32(:,n)-y, "inf");
98     e_4(n) = norm(yn_4(:,n)-y, "inf");

```

```

99 end
100
101
102 % Representamos graficamente el error cometido por cada
    metodo en el tiempo t = 1 en funcion de la longitud de
    paso h.
103 figure('Name','Comparacion de los errores absolutos en
    funcion del paso h');
104 loglog(H, e_eu, 'rv-')
105 xlabel('Longitud de paso, h')
106 ylabel('Error absoluto')
107 pause
108 hold on
109 loglog(H, e_21, 'gs-')
110 pause
111 loglog(H, e_22, 'mx-')
112 pause
113 loglog(H, e_31, 'kd-')
114 pause
115 loglog(H, e_32, 'bp-')
116 pause
117 loglog(H, e_4, 'c*-')
118 legend('Metodo exponencial de Euler','Metodo de orden 2 (I)'
    , 'Metodo de orden 2 (II)', 'Metodo de orden 3 (I)',
    'Metodo de orden 3 (II)', 'Metodo de orden 4')
119 pause
120
121 % Por ultimo, representamos graficamente el error absoluto
    cometido sobre la amplitud r y la fase theta, por separado
122 ep_eu = zeros(2,Nexp);
123 ep_21 = zeros(2,Nexp);
124 ep_22 = zeros(2,Nexp);
125 ep_31 = zeros(2,Nexp);
126 ep_32 = zeros(2,Nexp);
127 ep_4 = zeros(2,Nexp);
128
129 for n = 1:Nexp
130     ep_eu(:,n) = abs(polares_eu(:,n)-polares);
131     ep_21(:,n) = abs(polares_21(:,n)-polares);
132     ep_22(:,n) = abs(polares_22(:,n)-polares);
133     ep_31(:,n) = abs(polares_31(:,n)-polares);
134     ep_32(:,n) = abs(polares_32(:,n)-polares);
135     ep_4(:,n) = abs(polares_4(:,n)-polares);
136 end
137
138 % Representamos graficamente el error cometido por cada
    metodo en el tiempo t = 1 en funcion de la longitud de

```

```

    paso h.
139 figure('Name','Comparacion del error sobre r en funcion del
    paso h');
140 loglog(H, ep_eu(1,:), 'rv-')
141 xlabel('Longitud de paso, h')
142 ylabel('Error absoluto')
143 pause
144 hold on
145 loglog(H, ep_21(1,:), 'gs-')
146 pause
147 loglog(H, ep_22(1,:), 'mx-')
148 pause
149 loglog(H, ep_31(1,:), 'kd-')
150 pause
151 loglog(H, ep_32(1,:), 'bp-')
152 pause
153 loglog(H, ep_4(1,:), 'c*-')
154 legend('Metodo exponencial de Euler','Metodo de orden 2 (I)'
    , 'Metodo de orden 2 (II)', 'Metodo de orden 3 (I)', '
    Metodo de orden 3 (II)', 'Metodo de orden 4')
155 pause
156
157 figure('Name','Comparacion del error sobre theta en funcion
    del paso h');
158 loglog(H, ep_eu(2,:), 'rv-')
159 xlabel('Longitud de paso, h')
160 ylabel('Error absoluto')
161 pause
162 hold on
163 loglog(H, ep_21(2,:), 'gs-')
164 pause
165 loglog(H, ep_22(2,:), 'mx-')
166 pause
167 loglog(H, ep_31(2,:), 'kd-')
168 pause
169 loglog(H, ep_32(2,:), 'bp-')
170 pause
171 loglog(H, ep_4(2,:), 'c*-')
172 legend('Metodo exponencial de Euler','Metodo de orden 2 (I)'
    , 'Metodo de orden 2 (II)', 'Metodo de orden 3 (I)', '
    Metodo de orden 3 (II)', 'Metodo de orden 4')
173 pause

```

A.3. Programa principal en MATLAB para el segundo ejemplo semilineal

En este script se encuentra el programa principal realizado para el análisis del problema (2.14) discretizado espacialmente en la Sección 2.4. Las funciones que aparecen para aproximar la solución del problema mediante los métodos Runge-Kutta exponenciales estudiados en la Sección 1.3.2 se encuentran en el Apéndice A.4.

```

1  % CASO SEMILINEAL 2
2
3  % Comenzamos por definir los parametros de entrada de
4  % nuestras funciones y que definen el problema a estudiar.
5  a = 110;
6  J = 512;
7  x = ((1:J-1)/J)';
8  e = ones(J-1,1);
9  y0 = a*x.*(e-x)/10;
10 A = -(J^2)*spdiags([e -2*e e],[-1:1,J-1,J-1]);
11 I = eye(J-1);
12 MF1 = (J/2)*spdiags([-e zeros(J-1,1) e],[-1:1,J-1,J-1]);
13 F = @(t,y) -(MF1*y).*y + (1/(1+(10*t-3)^2))*(2*a*e-20*(10*t
14 -3)*y+a*(y-2*x.*y));
15
16 % Ahora, calculamos los errores absolutos que se cometen
17 % para distintas longitudes de paso h dentro de cada uno de
18 % los metodos en el tiempo t = 1, donde Nexp es el numero
19 % de experimentos a realizar.
20 Nexp = 7;
21 N = zeros(1,Nexp);
22 N(1) = 2^9;
23 for i = 2:Nexp
24     N(i) = 2*N(i-1);
25 end
26 H = 1./N;
27
28 % Creamos matrices con los valores de las aproximaciones en
29 % t=1, una por cada metodo.
30 yn_eu = zeros(J-1,Nexp);
31 yn_21 = zeros(J-1,Nexp);
32 yn_22 = zeros(J-1,Nexp);
33 yn_31 = zeros(J-1,Nexp);
34 yn_32 = zeros(J-1,Nexp);
35 yn_4 = zeros(J-1,Nexp);
36
37 % El tiempo que tarda cada metodo en ejecutarse para cada h
38 % lo guardamos en la matriz temp, de modo que cada fila
39 % corresponde a cada metodo.

```

```

32 temp = zeros(6,Nexp);
33
34 for i = 1:Nexp
35     clear yn;
36     tic;
37     [yn] = euler(y0, N(i), H(i), A, F);
38     temp(1,i) = toc;
39     yn_eu(:,i) = yn(:,N(i)+1);
40     clear yn;
41     c2 = 0.5;
42     tic;
43     [yn] = orden2_1(y0, N(i), H(i), A, F, c2);
44     temp(2,i) = toc;
45     yn_21(:,i) = yn(:,N(i)+1);
46     clear yn;
47     tic;
48     [yn] = orden2_2(y0, N(i), H(i), A, F, c2);
49     temp(3,i) = toc;
50     yn_22(:,i) = yn(:,N(i)+1);
51     clear yn;
52     c2 = 1/3;
53     tic;
54     [yn] = orden3_1(y0, N(i), H(i), A, F, c2);
55     temp(4,i) = toc;
56     yn_31(:,i) = yn(:,N(i)+1);
57     clear yn;
58     c2 = 0.5;
59     c3 = 0.75;
60     tic;
61     [yn] = orden3_2(y0, N(i), H(i), A, F, c2, c3);
62     temp(5,i) = toc;
63     yn_32(:,i) = yn(:,N(i)+1);
64     clear yn;
65     tic;
66     [yn] = orden4(y0, N(i), H(i), A, F);
67     temp(6,i) = toc;
68     yn_4(:,i) = yn(:,N(i)+1);
69 end
70
71 % El valor que toma la solucion exacta en t = 1 es
72 t = 1;
73 clear y;
74 y = a*x.*(e-x)/(1+(10*t-3)^2);
75
76 % El error absoluto cometido en cada caso es:
77 e_eu = zeros(1,Nexp);
78 e_21 = zeros(1,Nexp);

```

```

79 e_22 = zeros(1,Nexp);
80 e_31 = zeros(1,Nexp);
81 e_32 = zeros(1,Nexp);
82 e_4 = zeros(1,Nexp);
83
84 for n = 1:Nexp
85     e_eu(n) = norm(yn_eu(:,n)-y, "inf");
86     e_21(n) = norm(yn_21(:,n)-y, "inf");
87     e_22(n) = norm(yn_22(:,n)-y, "inf");
88     e_31(n) = norm(yn_31(:,n)-y, "inf");
89     e_32(n) = norm(yn_32(:,n)-y, "inf");
90     e_4(n) = norm(yn_4(:,n)-y, "inf");
91 end
92
93
94 % Representamos graficamente el error cometido por cada
    metodo en el tiempo t = 1 en funcion de la longitud de
    paso h.
95 figure('Name','Comparacion de los errores absolutos en
    funcion del paso h');
96 loglog(H, e_eu, 'rv-')
97 xlabel('Longitud de paso, h')
98 ylabel('Error absoluto')
99 pause
100 hold on
101 loglog(H, e_21, 'gs-')
102 pause
103 loglog(H, e_22, 'mx-')
104 pause
105 loglog(H, e_31, 'kd-')
106 pause
107 loglog(H, e_32, 'bp-')
108 pause
109 loglog(H, e_4, 'c*-')
110 legend('Metodo exponencial de Euler','Metodo de orden 2 (I)'
    , 'Metodo de orden 2 (II)', 'Metodo de orden 3 (I)',
    'Metodo de orden 3 (II)', 'Metodo de orden 4')
111 pause
112
113 % Representamos graficamente el error cometido por cada
    metodo en el tiempo t=1 en funcion del tiempo CPU
114 figure('Name','Comparacion de los errores absolutos en
    funcion del tiempo CPU');
115 loglog(temp(1,:), e_eu, 'rv-')
116 xlabel('Tiempo de CPU')
117 ylabel('Error absoluto')
118 pause

```



```

119 hold on
120 loglog(temp(2,:), e_21, 'gs-')
121 pause
122 loglog(temp(3,:), e_22, 'mx-')
123 pause
124 loglog(temp(4,:), e_31, 'kd-')
125 pause
126 loglog(temp(5,:), e_32, 'bp-')
127 pause
128 loglog(temp(6,:), e_4, 'c*-')
129 legend('Metodo exponencial de Euler','Metodo de orden 2 (I)'
        , 'Metodo de orden 2 (II)', 'Metodo de orden 3 (I)',
        'Metodo de orden 3 (II)', 'Metodo de orden 4')
130 pause

```

A.4. Funciones en MATLAB que implementan los métodos Runge-Kutta exponenciales

A continuación se muestran las funciones elaboradas para la aproximación de la solución de los problemas (2.8)-(2.9) y (2.14), previamente discretizado en espacio, mediante cada uno de los métodos Runge-Kutta exponenciales desarrollados en la Sección 1.3.2. En ellas, se hace uso de la función `phipm.m` tomada de la referencia [6] y la única modificación que ha habido que hacer entre ambos ejemplos ha sido sobre esta función, pues para el primer ejemplo hemos fijado una tolerancia 10^{-7} y se ha indicado que la matriz A no es simétrica, mientras que para el segundo, la tolerancia tomada es 10^{-14} y se ha aprovechado que la matriz A es simétrica. Las funciones que mostramos son las de este último caso.

A.4.1. Método de Euler exponencial (orden 1)

```

1 function [yn] = euler(y0, N, h, A, F)
2 % Los parametros de entrada de esta funcion son el vector
   inicial del problema y0, el numero de iteraciones que
   queremos realizar N, la longitud de paso deseada h, el
   valor que toma la matriz A del problema y el termino no
   lineal F del problema, que debera ser introducido en el
   siguiente formato: @(t,y) F(t,y).
3
4 % El output sera una matriz LxN yn cuyas columnas son las
   aproximaciones de la solucion en cada tiempo tn = nh, n<N
   +1, mediante el metodo de Euler exponencial.
5
6 L = length(y0);
7 yn = zeros(L,N+1);
8 yn(:,1) = y0;

```

```

9  t = 0;
10 tol = 1e-14;
11 symm = 'true';
12
13 for n = 2:N+1
14     y = yn(:,n-1);
15     u = [y, F(t,y)];
16     [yn(:,n), ~] = phipm(h, -A, u, tol, symm);
17     t = t + h;
18 end
19 end

```

A.4.2. Método de orden 2 (I)

```

1  function [yn] = orden2_1(y0, N, h, A, F, c2)
2  % Los parametros de entrada de esta funcion son el vector
   % inicial del problema y0, el numero de iteraciones que
   % queremos realizar N, la longitud de paso deseada h, el
   % valor que toma la matriz A del problema, el termino no
   % lineal F del problema, que debera ser introducido en el
   % siguiente formato: @(t,y) F(t,y) y el nodo c2.
3
4  % El output sera una matriz LxN yn cuyas columnas son las
   % aproximaciones de la solucion en cada tiempo tn = nh, n<N
   % +1, mediante el primer metodo de orden 2 estudiado.
5
6  L = length(y0);
7  yn = zeros(L,N+1);
8  yn(:,1) = y0;
9  t = 0;
10 tol = 1e-14;
11 symm = 'true';
12 hc2 = h*c2;
13 hc2_1 = 1/hc2;
14
15 for n = 2:N+1
16     y = yn(:,n-1);
17     FYn1 = F(t,y);
18     Gn1 = FYn1-A*y;
19     u1 = [zeros(L,1), Gn1];
20     [a21F, ~] = phipm(hc2, -A, u1, tol, symm);
21     Yn2 = y + a21F;
22     FYn2 = F(t+hc2,Yn2);
23     F21 = FYn2-FYn1;
24     u2 = [zeros(L,1), Gn1, hc2_1*F21];
25     [bF, ~] = phipm(h, -A, u2, tol, symm);

```

```

26         yn(:, n) = y + bF;
27         t = t + h;
28     end
29 end

```

A.4.3. Método de orden 2 (II)

```

1  function [yn] = orden2_2(y0, N, h, A, F, c2)
2  % Los parametros de entrada de esta funcion son el vector
   % inicial del problema y0, el numero de iteraciones que
   % queremos realizar N, la longitud de paso deseada h, el
   % valor que toma la matriz A del problema, el termino no
   % lineal F del problema, que debera ser introducido en el
   % siguiente formato: @(t,y) F(t,y) y el nodo c2.
3
4  % El output sera una matriz LxN yn cuyas columnas son las
   % aproximaciones de la solucion en cada tiempo tn = nh, n<N
   % +1, mediante el segundo metodo de orden 2 estudiado.
5
6  L = length(y0);
7  yn = zeros(L,N+1);
8  yn(:,1) = y0;
9  t = 0;
10 tol = 1e-14;
11 symm = 'true';
12 hc2 = h*c2;
13 d2 = 0.5/c2;
14
15 for n = 2:N+1
16     y = yn(:,n-1);
17     FYn1 = F(t,y);
18     Gn1 = FYn1-A*y;
19     u1 = [zeros(L,1), Gn1];
20     [a21F, ~] = phipm(hc2, -A, u1, tol, symm);
21     Yn2 = y + a21F;
22     FYn2 = F(t+hc2,Yn2);
23     F21 = FYn2-FYn1;
24     u2 = [zeros(L,1), Gn1+d2*F21];
25     [bF, ~] = phipm(h, -A, u2, tol, symm);
26     yn(:, n) = y + bF;
27     t = t + h;
28 end
29 end

```

A.4.4. Método de orden 3 (I)

```

1 function [yn] = orden3_1(y0, N, h, A, F, c2)
2 % Los parametros de entrada de esta funcion son el vector
   inicial del problema y0, el numero de iteraciones que
   queremos realizar N, la longitud de paso deseada h, el
   valor que toma la matriz A del problema, el termino no
   lineal F del problema, que debera ser introducido en el
   siguiente formato: @(t,y) F(t,y) y el nodo c2.
3
4 % El output sera una matriz LxN yn cuyas columnas son las
   aproximaciones de la solucion en cada tiempo tn = nh, n<N
   +1, mediante el primer metodo de orden 3 estudiado.
5
6 L = length(y0);
7 yn = zeros(L,N+1);
8 yn(:,1) = y0;
9 t = 0;
10 tol = 1e-14;
11 symm = 'true';
12 hc2 = h*c2;
13 hc2_1 = 1/hc2;
14 h23 = h*2/3;
15 h23_1 = 1/h23;
16
17 for n = 2:N+1
18     y = yn(:,n-1);
19     FYn1 = F(t,y);
20     Gn1 = FYn1-A*y;
21     u1 = [zeros(L,1), Gn1];
22     [a21F, ~] = phipm(hc2, -A, u1, tol, symm);
23     Yn2 = y + a21F;
24     FYn2 = F(t+hc2,Yn2);
25     F21 = FYn2-FYn1;
26     u2 = [zeros(L,1), Gn1, hc2_1*F21];
27     [a3F, ~] = phipm(h23, -A, u2, tol, symm);
28     Yn3 = y + a3F;
29     FYn3 = F(t+h23,Yn3);
30     F31 = FYn3-FYn1;
31     u3 = [zeros(L,1), Gn1, h23_1*F31];
32     [bF, ~] = phipm(h, -A, u3, tol, symm);
33     yn(:, n) = y + bF;
34     t = t + h;
35 end
36 end

```

A.4.5. Método de orden 3 (II)

```

1 function [yn] = orden3_2(y0, N, h, A, F, c2, c3)
2 % Los parametros de entrada de esta funcion son el vector
   inicial del problema y0, el numero de iteraciones que
   queremos realizar N, la longitud de paso deseada h, el
   valor que toma la matriz A del problema, el termino no
   lineal F del problema, que debiera ser introducido en el
   siguiente formato: @(t,y) F(t,y) y los nodos c2 y c3.
3
4 % El output sera una matriz LxN yn cuyas columnas son las
   aproximaciones de la solucion en cada tiempo tn = nh, n<N
   +1, mediante el segundo metodo de orden 3 estudiado.
5
6 L = length(y0);
7 yn = zeros(L,N+1);
8 yn(:,1) = y0;
9 t = 0;
10 gamma = c3*(3*c3-2)/(c2*(2-3*c2));
11 tol = 1e-14;
12 symm = 'true';
13 hc2 = h*c2;
14 hc2_1 = 1/hc2;
15 hc3 = h*c3;
16 d1 = gamma/hc2;
17 d2 = 1/(h*(gamma*c2+c3));
18
19 for n = 2:N+1
20     y = yn(:,n-1);
21     FYn1 = F(t, y);
22     Gn1 = FYn1-A*y;
23     u1 = [zeros(L,1), Gn1];
24     [a21F, ~] = phipm(hc2, -A, u1, tol, symm);
25     Yn2 = y + a21F;
26     FYn2 = F(t+hc2,Yn2);
27     F21 = FYn2-FYn1;
28     u2_1 = [zeros(L,1), Gn1, hc2_1*F21];
29     [a3F_1, ~] = phipm(hc3, -A, u2_1, tol, symm);
30     u2_2 = [zeros(L,1), zeros(L,1), d1*F21];
31     [a3F_2, ~] = phipm(hc2, -A, u2_2, tol, symm);
32     Yn3 = y + a3F_1 + a3F_2;
33     FYn3 = F(t+hc3,Yn3);
34     F31 = FYn3-FYn1;
35     u3 = [zeros(L,1), Gn1, d2*(F31+gamma*F21)];
36     [bF, ~] = phipm(h, -A, u3, tol, symm);
37     yn(:, n) = y + bF;
38     t = t + h;

```

```

39 end
40 end

```

A.4.6. Método de orden 4

```

1 function [yn] = orden4(y0, N, h, A, F)
2 % Los parametros de entrada de esta funcion son el vector
   inicial del problema y0, el numero de iteraciones que
   queremos realizar N, la longitud de paso deseada h, el
   valor que toma la matriz A del problema y el termino no
   lineal F del problema, que debera ser introducido en el
   siguiente formato: @(t,y) F(t,y).
3
4 % El output sera una matriz LxN yn cuyas columnas son las
   aproximaciones de la solucion en cada tiempo tn = nh, n<N
   +1, mediante el metodo de orden 4 estudiado.
5
6 L = length(y0);
7 yn = zeros(L,N+1);
8 yn(:,1) = y0;
9 t = 0;
10 tol = 1e-14;
11 symm = 'true';
12 hc2 = h*0.5;
13 hc22 = 1/(hc2^2);
14 h_1 = 1/h;
15 h_12 = h_1^2;
16 h_14 = 4*h_1;
17 h_025 = 0.25*h_1;
18
19 for n = 2:N+1
20     y = yn(:,n-1);
21     FYn1 = F(t,y);
22     Gn1 = FYn1-A*y;
23     u1 = [zeros(L,1), Gn1];
24     [a21F, ~] = phipm(hc2, -A, u1, tol, symm);
25     Yn2 = y + a21F;
26     FYn2 = F(t+hc2,Yn2);
27     F21 = FYn2-FYn1;
28     u2 = [zeros(L,1), Gn1, h_14*F21];
29     [a3F, ~] = phipm(hc2, -A, u2, tol, symm);
30     Yn3 = y + a3F;
31     FYn3 = F(t+hc2,Yn3);
32     F31 = FYn3-FYn1;
33     u3 = [zeros(L,1), Gn1, h_1*(F21+F31)];
34     [a4F, ~] = phipm(h, -A, u3, tol, symm);

```

A.4. FUNCIONES EN MATLAB PARA LOS MÉTODOS RK EXPONENCIALES 79

```
35     Yn4 = y + a4F;
36     FYn4 = F(t+h,Yn4);
37     F1234 = FYn1-FYn2-FYn3+FYn4;
38     u4_1 = [zeros(L,1), Gn1, h_1*(F21+F31-F1234), hc2*
           F1234];
39     [a5F_1, ~] = phipm(hc2, -A, u4_1, tol, symm);
40     u4_2 = [zeros(L,1), zeros(L,1), h_025*(-F1234), h_12
           *F1234];
41     [a5F_2, ~] = phipm(h, -A, u4_2, tol, symm);
42     Yn5 = y + a5F_1 + a5F_2;
43     FYn5 = F(t+hc2,Yn5);
44     u5 = [zeros(L,1), Gn1, h_1*(-3*FYn1-FYn4+4*FYn5),
           hc22*(FYn1+FYn4-2*FYn5)];
45     [bF, ~] = phipm(h, -A, u5, tol, symm);
46     yn(:, n) = y + bF;
47     t = t + h;
48 end
49 end
```