Problema 1 (5.0pt)

Se desea implementar un método Runge–Kutta tipo EIN (*explicit–implicit–null*) para integrar sistemas de ecuaciones diferenciales de la forma

$$\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t} = \mathbf{f}(t, \mathbf{y}), \quad \mathbf{y}(t^0) = \mathbf{y}^0.$$

En particular, supongamos que conocemos la solución $\mathbf{y}^n$ en un instante t^n. Para hallar la solución $\mathbf{y}^{n+1}$ en $t^{n+1} = t^n + \Delta t^n$, se resuelven primero s etapas intermedias. La primera etapa viene dada por

$$\begin{aligned} t^{[1]} &= t^n, \\ \mathbf{y}^{[1]} &= \mathbf{y}^n, \\ \mathbf{f}^{[1]} &= \mathbf{f}(t^{[1]}, \mathbf{y}^{[1]}), \\ \widehat{\mathbf{f}}^{[1]} &= \mathbb{J}^n \mathbf{y}^{[1]}, \end{aligned}$$

mientras que las etapas $i = 2, \ldots, s$ vienen dadas por

$$t^{[i]} = t^n + \Delta t^n \, c_i,$$

$$\mathbf{y}^{[i]} = \mathbf{y}^n + \Delta t^n \sum_{j=1}^{i-1} a_{ij} \mathbf{f}^{[j]} + \Delta t^n \sum_{j=1}^{i-1} \widehat{a}_{ij} \widehat{\mathbf{f}}^{[j]} + \Delta t^n \gamma \, \widehat{\mathbf{f}}^{[i]}, \tag{1}$$

$$\mathbf{f}^{[i]} = \mathbf{f}(t^{[i]}, \mathbf{y}^{[i]}),$$

$$\widehat{\mathbf{f}}^{[i]} = \mathbb{J}^n \mathbf{y}^{[i]}. \tag{2}$$

En estas ecuaciones, $\mathbb{J}^n := \partial \mathbf{f}(t^n, \mathbf{y}^n)/\partial \mathbf{y}$ es la matriz jacobiana evaluada en $t = t^n$, $\mathbf{y} = \mathbf{y}^n$. Se supone que $\mathbb{J}^n$ es una matriz llena no simétrica. Una vez resueltas las etapas intermedias, la solución en t^{n+1} viene dada por

$$\mathbf{y}^{n+1} = \mathbf{y}^n + \Delta t^n \sum_{j=1}^{s} b_j \mathbf{f}^{[j]} + \Delta t^n \sum_{j=1}^{s} \widehat{b}_j \widehat{\mathbf{f}}^{[j]}. \tag{3}$$

Los coeficientes a_{ij}, $\widehat{a}_{ij}$, γ, b_j, $\widehat{b}_j$ y c_i vienen dados por el *tablero de Butcher*:

$$\begin{array}{r|ccccc|ccccc} c_1 = 0 & 0 & & & & & 0 & & & & \\ c_2 & a_{2,1} & & & & & \widehat{a}_{2,1} & \gamma & & & \\ c_3 & a_{3,1} & a_{3,2} & & & & \widehat{a}_{3,1} & \widehat{a}_{3,2} & \gamma & & \\ \vdots & \vdots & & \ddots & & & \vdots & & & \ddots & \\ c_s & a_{s,1} & \cdots & \cdots & a_{s,s-1} & & \widehat{a}_{s,1} & \cdots & \cdots & \widehat{a}_{s,s-1} & \gamma \\ \hline & b_1 & \cdots & \cdots & b_{s-1} & b_s & \widehat{b}_1 & \cdots & \cdots & \widehat{b}_{s-1} & \widehat{b}_s \end{array} . \tag{4}$$

Asimismo, como ya sucedía con los Runge–Kutta explícitos e implícitos, t^{n+1} y Δt^n vienen dados por

$$t^{n+1} = \min\{t^n + \Delta t, t_{\text{final}}\}, \quad \Delta t^n = t^{n+1} - t^n,$$

donde Δt es el paso de tiempo objetivo y t_{final} es el instante final de simulación.

Cuestión 1 (0.75pt): *Si se combinan* (1) *y* (2) *se puede hallar una ecuación que determina* $\mathbf{y}^{[i]}$. *¿Cuál es dicha ecuación? ¿Qué características importantes posee? ¿Qué podría hacerse para acelerar su resolución?*

Solución: Si sustituimos (2) en (1), se tiene

$$\mathbf{y}^{[i]} = \mathbf{y}^n + \Delta t^n \sum_{j=1}^{i-1} a_{ij} \mathbf{f}^{[j]} + \Delta t^n \sum_{j=1}^{i-1} \widehat{a}_{ij} \widehat{\mathbf{f}}^{[j]} + \Delta t^n \gamma \, \mathbb{J}^n \mathbf{y}^{[i]},$$

es decir

$$\mathbf{y}^{[i]} - \Delta t^n \gamma\, \mathbb{J}^n \mathbf{y}^{[i]} = \mathbf{y}^n + \Delta t^n \sum_{j=1}^{i-1} a_{ij} \mathbf{f}^{[j]} + \Delta t^n \sum_{j=1}^{i-1} \widehat{a}_{ij} \widehat{\mathbf{f}}^{[j]}.$$

Notemos que el lado izquierdo contiene la incógnita $\mathbf{y}^{[i]}$, mientras que el lado derecho es conocido en la etapa i.

Para despejar correctamente $\mathbf{y}^{[i]}$:

(i) $\mathbf{y}^{[i]} - \Delta t^n \gamma\, \mathbb{J}^n \mathbf{y}^{[i]}$ NO es igual a $(1 - \Delta t^n \gamma\, \mathbb{J}^n)\mathbf{y}^{[i]}$. Las matrices y los escalares NO se pueden sumar. La forma correcta de escribirlo sería $(\mathbb{I} - \Delta t^n \gamma\, \mathbb{J}^n)\mathbf{y}^{[i]}$, siendo $\mathbb{I}$ la matriz identidad.

(ii) La forma correcta de despejar $\mathbf{y}^{[i]}$ NO es

$$\mathbf{y}^{[i]} = \frac{\mathbf{y}^n + \Delta t^n \sum_{j=1}^{i-1} a_{ij} \mathbf{f}^{[j]} + \Delta t^n \sum_{j=1}^{i-1} \widehat{a}_{ij} \widehat{\mathbf{f}}^{[j]}}{\mathbb{I} - \Delta t^n \gamma\, \mathbb{J}^n}.$$

La división por una matriz es una operación inexistente (e inaceptable en tercero de ingeniería). La forma correcta de despejar $\mathbf{y}^{[i]}$ es

$$\mathbf{y}^{[i]} = (\mathbb{I} - \Delta t^n \gamma\, \mathbb{J}^n)^{-1} \left(\mathbf{y}^n + \Delta t^n \sum_{j=1}^{i-1} a_{ij} \mathbf{f}^{[j]} + \Delta t^n \sum_{j=1}^{i-1} \widehat{a}_{ij} \widehat{\mathbf{f}}^{[j]} \right). \tag{5}$$

Recordemos que $(\mathbb{I} - \Delta t^n \gamma\, \mathbb{J}^n)^{-1}$ denota un operador que actúa como la inversa de la matriz $(\mathbb{I} - \Delta t^n \gamma\, \mathbb{J}^n)$. En MATLAB se puede usar el comando barra invertida \, una factorización apropiada de la matriz o algún método iterativo como GMRES.

La ecuación (5) representa un sistema lineal y, además, la matriz es la misma en cada etapa. Por ello, para acelerar los cálculos, se puede factorizar en la primera etapa y utilizar su factorización en las etapas posteriores. Puesto que, según el enunciado, la matriz $\mathbb{J}^n$ es llena y no simétrica, la factorización más adecuada es la LU con permutación de filas. □

La función `[A, Ahat, gamma, b, bhat, c, s] = CoefsRKEIN(metodo)` proporciona los coeficientes del tablero de Butcher, así como el número s de etapas intermedias, a partir del nombre `metodo` del método Runge–Kutta que se desee utilizar.

Asimismo, se supone que se dispone de una función `[f,df_dy]=fun(t,y,CalcJ)` que calcula $\mathbf{f}(t, \mathbf{y})$ y, si `CalcJ=true`, $\partial \mathbf{f}(t, \mathbf{y})/\partial \mathbf{y}$.

Cuestión 2 (3pt): *Implementar una función* ***`[tv, ym] = RungeKuttaEIN(fun, t0, y0, metodo, Deltat, tf)`*** *que reciba la función* ***`fun`****, el instante inicial t^0, la condición inicial $\mathbf{y}^0$ como vector columna, el nombre* ***`metodo`*** *del método Runge–Kutta EIN que se desea utilizar, el paso de tiempo objetivo Δt y el instante final t_{final} y devuelva:*

- *un vector fila* ***`tv`*** $= [t^0, \ldots, t^N]$ *que recopile los instantes resueltos, excluyendo las etapas intermedias,*
- *una matriz* ***`ym`*** $= [\mathbf{y}^0 | \ldots | \mathbf{y}^N]$ *que contenga, por columnas, la solución correspondiente a cada uno de los instantes en* ***`tv`****.*

Dentro de esta función, deben tomarse los pasos que se estimen oportunos para que el cálculo de $\mathbf{y}^{[i]}$ en las etapas intermedias sea lo más eficiente posible.

Solución: Véase la función `RungeKuttaEIN1`. □

A continuación, se desea resolver el test de Robertson con el método RK-EIN. Recuérdese que la función `EDOs_Robertson_RKI` resuelve este problema con el método Runge–Kutta implícito.

Cuestión 3 (0.25pt): *Implementar una función* ***`EDOs_Robertson_RKEIN(metodo, Deltat, tf)`*** *que resuelva el test de Robertson utilizando la función* ***`RungeKuttaEIN`****. Dibujar la evolución de y_1, y_2, y_3 con el tiempo en tres figuras distintas. Asimismo, mostrar por pantalla el tiempo consumido en la llamada al método Runge–Kutta y el valor de $y_3(t_{\text{final}})$.*

Solución: Véase la función `EDOs_Robertson_RKEIN`. □

Cuestión 4 (0.5pt): *Completar la Tabla 1 con el tiempo de cálculo y el valor de* $y_3(t_{\text{final}})$ *proporcionados por el método Runge–Kutta implícito* ***`RK3I`*** *y el método Runge–Kutta EIN* ***`RK3EIN`****. Con cuatro cifras significativas es suficiente. Tómese* $t_{\text{final}} = 10$*. Es posible que haya que modificar la función* ***`EDOs_Robertson_RKI`*** *para que muestre por pantalla los datos pedidos, pero no es necesario entregar la función modificada.*

Tabla 1: Resultados del test de Robertson.

Δt	`RK3I`: $y_3(t_{\text{final}})$	`RK3I`: tiempo [s]	`RK3EIN`: $y_3(t_{\text{final}})$	`RK3EIN`: tiempo [s]
10^{-2}	0.1586	0.1672	0.1583	0.04900
10^{-3}	0.1586	0.9292	0.1586	0.2424
10^{-4}	0.1586	36.0276	0.1586	1.9269

Cuestión 5 (0.5pt): *¿Se observa alguna diferencia en cuanto a* $y_3(t_{\text{final}})$*? Para el mismo paso de tiempo, ¿qué método es más rápido y a qué se debe?*

Solución: Sí, se observa alguna diferencia para $y_3(t_{\text{final}})$. En particular, este valor es el mismo (con cuatro cifras significativas) para el método implícito, mientras que para el método EIN la convergencia aparece más tarde.

Para el mismo paso de tiempo, se observa que el método EIN es mucho menos costoso. Esto se debe a que el método implícito requiere resolver iterativamente un sistema no lineal en cada etapa, mientras que el método EIN sólo debe resolver un sistema lineal en cada etapa. □

Problema 2 (5.0pt)

Se desea estudiar la deformación que experimenta el ala de un avión al atravesar una zona de turbulencias. Para ello, se supone que el ala es una viga en voladizo de longitud $L = 1$, como se muestra en la Fig. 1(a). Bajo ciertas hipótesis, el desplazamiento vertical hacia arriba u viene dado por[1]

$$u(x) = \int_0^L K(x, X)\, p(X)\, \mathrm{d}X, \quad 0 \le x \le L, \tag{6}$$

donde

$$K(x, X) = \begin{cases} \dfrac{x^2 X}{2} - \dfrac{x^3}{6}, & x \le X, \\ \dfrac{x X^2}{2} - \dfrac{X^3}{6}, & x > X, \end{cases} \tag{7}$$

es la función núcleo y p es la distribución de fuerza hacia arriba. En este caso, la turbulencia se modela por

$$p(X) = \lambda \frac{\sin(\pi(0.75 - X))}{1 + u^2(X)}, \quad 0 \le X \le L, \tag{8}$$

siendo λ un parámetro conocido que mide la intensidad de la misma.

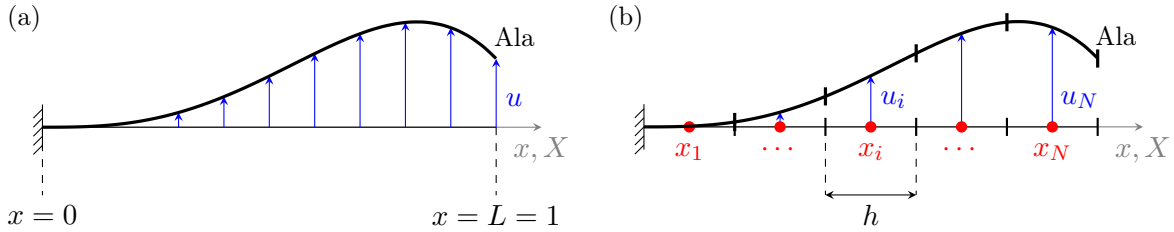

Figura 1: (a) Ala deformada y campo de desplazamientos. (b) Discretización numérica.

Cuestión 6 (0.25pt): *Conviene recordar en lo que sigue que la regla del punto medio establece que*

$$\int_a^b g(X)\, \mathrm{d}X \approx g\left(\frac{a+b}{2}\right) w_0,$$

donde $w_0 = b - a$ *(completar la expresión anterior en función de* a *y* b*).*

Para resolver el sistema (6)-(8), sustituimos primero (8) en (6), de tal forma que queda la siguiente ecuación integral para u:

$$u(x) = \int_0^L K(x, X) \qquad . \tag{9}$$

A continuación, se divide el ala en N paneles de longitud h (véase Fig. 1(b)). Llamamos $x_1, \ldots, x_N$ a las coordenadas de los puntos medios de cada panel, $u_1, \ldots, u_N$ a los valores de u en dichos nodos, y $\mathbf{u} := [u_1, \ldots, u_N]^{\mathrm{T}}$. La ecuación (9) se particulariza para $x = x_1, \ldots, x_N$ y la integral en el lado derecho se aproxima por la regla del punto medio a trozos. En ese caso, se obtiene el sistema no lineal

$$f_i(\mathbf{u}) := u_i - \sum_{k=1}^{N} K(x_i, \quad) \qquad = 0, \quad i = 1, \ldots, N. \tag{10}$$

La matriz jacobiana asociada es

$$J_{ij}(\mathbf{u}) := \frac{\partial f_i(\mathbf{u})}{\partial u_j} = \qquad , \quad i, j = 1, \ldots, N. \tag{11}$$

[1] Todas las magnitudes están adimensionalizadas con la longitud del ala y con la rigidez a flexión de la misma.

Cuestión 7 (0.75pt): *Completar las ecuaciones* (9), (10) *y* (11). *No es necesario sustituir la expresión* (7) *para* K *en estas ecuaciones.*

Solución: Si sustituimos (8) en (6), queda

$$u(x) = \int_0^L K(x, X)\lambda \frac{\sin(\pi(0.75 - X))}{1 + u^2(X)}\,\mathrm{d}X.$$

Notemos que x denota un punto fijo del espacio, mientras que la variable de integración es X. Si particularizamos la ecuación anterior para $x = x_1, \ldots, x_N$, y tenemos en cuenta que $u_i = u(x_i)$, queda

$$u_i = \int_0^L K(x_i, X)\lambda \frac{\sin(\pi(0.75 - X))}{1 + u^2(X)}\,\mathrm{d}X, \quad i = 1, \ldots, N.$$

Recordemos que la regla del punto medio a trozos establece (utilizando de nuevo la notación de la Fig. 1(b)) que

$$\int_0^L g(X)\,\mathrm{d}X \approx \sum_{k=1}^{N} g(x_k)h.$$

Por tanto, si aplicamos esta regla, se tiene

$$u_i = \sum_{k=1}^{N} K(x_i, x_k)\lambda \frac{\sin(\pi(0.75 - x_k))}{1 + u_k^2}\,h, \quad i = 1, \ldots, N.$$

Es decir, hay que resolver el sistema no lineal dado por

$$f_i(\mathbf{u}) := u_i - \sum_{k=1}^{N} K(x_i, x_k)\lambda \frac{\sin(\pi(0.75 - x_k))}{1 + u_k^2}\,h = 0, \quad i = 1, \ldots, N.$$

La matriz jacobiana asociada es

$$\begin{aligned} J_{ij} := \frac{\partial f_i(\mathbf{u})}{\partial u_j} &= \delta_{ij} + \lambda h \sum_{k=1}^{N} K(x_i, x_k) \frac{\sin(\pi(0.75 - x_k))}{(1 + u_k^2)^2} 2u_k\delta_{kj}, \\ &= \delta_{ij} + \lambda h\, K(x_i, x_j)\, \frac{\sin(\pi(0.75 - x_j))}{(1 + u_j^2)^2} 2u_j, \qquad i, j = 1, \ldots, N. \end{aligned}$$

□

Cuestión 8 (3pt): *Implementar una función* `[f,J] = AlaTorbellino(lambda, xv, uv, CalcJ)` *que, recibidos* λ, `xv` $= [x_1, \ldots, x_N]^{\mathrm{T}}$, `uv` $= \mathbf{u}$ *y la variable booleana* `CalcJ`, *calcula el residuo* $\mathbf{f}(\mathbf{u})$. *Además, si* `CalcJ=true`, *devuelve la matriz jacobiana,* `J` $= \mathbb{J}(\mathbf{u})$; *en caso contrario,* `J` $=$ `NaN`.

Solución: Véase la función `AlaTorbellino1`. □

De ahora en adelante, utilícese la función codificada `AlaTorbellino1`.

A continuación, se desea implementar una función `Problema2(N)` que, recibido N, realice los siguientes pasos:

1. Definir el parámetro $\lambda = 10$ y el vector `xv` $= [x_1, \ldots, x_N]^{\mathrm{T}}$.
2. Resolver el sistema (10) mediante el método de Newton–Raphson. Tómese el vector nulo como condición inicial (no olvidar definirlo como vector columna para evitar problemas con la función `NewtonRaphson1`), una tolerancia 10^{-8} y un máximo de 100 iteraciones.
3. Representar los desplazamientos u del ala frente a la coordenada x.

4. Mostrar por pantalla el desplazamiento máximo $u^{\text{máx}} := \text{máx}_i\, |u_i|$ (puede utilizar el comando `max`).

Cuestión 9 (0.5pt): *Implementar la función* `Problema2(N)`.

Solución: Véase la función `Problema2`. □

Cuestión 10 (0.5pt): *Indicar el desplazamiento máximo obtenido según el valor de N en la Tabla 2. Con escribir cinco cifras decimales es suficiente.*

Tabla 2: Desplazamiento máximo del ala.

N	$u^{\text{máx}}$
40	0.03238
80	0.03231
160	0.03229