Programación: el método de disparo para EDO de segundo orden lineales con condiciones de frontera

Objetivos. Resolver los problemas con valores de frontera para ecuaciones diferencias ordinarias lineales de segundo orden usando el método de disparo.

Requisitos. Haber programado varios métodos de Runge–Kutta, haber estudiado la teoría del método de disparo.

1. El problema por resolver. Vamos a resolver problemas de la forma

$$x''(t) = f(t, x(t), x'(t)), \quad x(a) = x_a, \quad x(b) = x_b,$$

donde está dada una función f de tres argumentos, algunos números a, b en $\mathbb{R}$ con $a < b$ y algunos números $x_a, x_b \in \mathbb{R}$. Más aún, en este tema estamos suponiendo que f es de la siguiente forma:

$$f(t, u, v) = p(t)v + q(t)u + r(t), \tag{1}$$

pero en el programa trabajaremos directamente con f, sin introducir p, q, r.

2. De la ecuación de segundo orden a un sistema de ecuaciones. Dada una ecuación de segunda orden de la forma

$$x''(t) = f(t, x(t), x'(t)), \tag{2}$$

introducimos la función auxiliar $y(t) \coloneqq x'(t)$ y formamos un sistema de dos ecuaciones de primer orden:

$$\begin{aligned} x'(t) &= y(t), \\ y'(t) &= \qquad\qquad \end{aligned} \tag{3}$$

Notamos que el lado derecho ya no contiene $x'(t)$. En la forma vectorial, si denotamos el vector $(x(t), y(t))$ por $X(t)$, entonces el sistema (3) se escribe como

$$X'(t) = \Big(X_2(t), \qquad\qquad\Big). \tag{4}$$

Denotamos por $F(t, X(t))$ a la expresión en el lado derecho de (4). Formalmente, F es una función de dos argumentos, donde el segundo argumento es vectorial, y esta función se puede definir de la siguiente manera, a través de la función f:

$$F(t, U) = \Big(U_2, \qquad\qquad\Big).$$

3. Programar la función vectorial F en términos de f. Supongamos que f es una función de tres argumentos que se utiliza en la ecuación (2). Entonces la función F se puede definir de la siguiente manera (en la sintaxis de Matlab/Octave):

```
F = @(t, U) [ U(2),                    ];
```

4. Dos problemas auxiliares con valores iniciales. Consideremos la EDO original (2) con condiciones iniciales $x(a) = x_a$, $x'(a) = 0$, y denotemos la solución de este problema por φ. En otras palabras, estamos suponiendo que

$$\varphi''(t) = f(t, \varphi(t), \varphi'(t)), \qquad \varphi(a) = \boxed{}, \quad \varphi'(a) = \boxed{}.$$

De manera equivalente, φ es la primera componente de la función vectorial Φ que satisface la ecuación (4):

$$\Phi'(t) = F(t, \Phi(t)),$$

y la condición inicial

$$\Phi(a) = \Big(\boxed{}, \boxed{}\Big).$$

Además consideremos la EDO original (2) con condiciones iniciales $x(a) = x_a$, $x'(a) = 1$, y denotemos la solución de este problema por ψ. En otras palabras, la función ψ satisface

$$\psi''(t) = \boxed{}, \qquad \psi(a) = \boxed{}, \qquad \psi'(a) = \boxed{}.$$

Vamos a calcular φ y ψ con alguno de los métodos numéricos para resolver problemas de Cauchy, por ejemplo, con algún método de Runge–Kutta de orden 4 o 5.

5. Combinación afín de las dos soluciones auxiliares. Definimos una función nueva x como una combinación afín (combinación lineal con suma de los coeficientes igual a 1) de las funciones φ y ψ:

$$x(t) \coloneqq (1 - \lambda)\varphi(t) + \lambda\psi(t).$$

Estamos suponiendo que f es de la forma Verifique que x satisface la ecuación (2):

$$\begin{aligned}
x''(t) &= (1-\lambda)\varphi''(t) + \lambda\psi''(t) \\
&= (1-\lambda)\big(p(t)\varphi'(t) + q(t)\varphi(t) + r(t)\big) + \lambda\big(\boxed{}\big) \\
&= p(t)\Big((1-\lambda)\varphi'(t) + \lambda\psi'(t)\Big) + q(t)\Big(\boxed{}\Big) \\
&+ \quad \Big(1 - \lambda + \boxed{}\Big) r(t) \\
&= p(t)\boxed{} + q(t)\boxed{} + \boxed{} = f\big(\boxed{}, x(t), \boxed{}\big).
\end{aligned}$$

Mostremos que x satisface la condición de frontera izquierda:

$$\begin{aligned} x(a) &= (1-\lambda)\varphi(\square) + \lambda\psi(\square) \\ &= (1-\lambda)\square + \lambda\square = \square. \end{aligned}$$

Finalmente consideremos la situación en la frontera derecha:

$$x(b) = (1-\lambda)\varphi(\square) + \lambda\psi(\square) = \varphi(\square) + \lambda\Big(\psi(b) - \square\Big).$$

Los valores $\varphi(b)$ y $\psi(b)$ son conocidos después de calcular las funciones φ y ψ, pero no tenemos ninguna fórmula simple para estos valores. Elegimos λ de tal manera que se satisfaga la condición de frontera derecha, es decir, $x(b) = x_b$.

$$\lambda = \frac{x_b - \varphi(\square)}{\psi(\square - \square}.$$

Al hallar λ, los valores de x se pueden calcular usando los valores conocidos de φ y ψ:

$$x(t) = \big(1-\lambda\big)\varphi(t) + \square\,\square.$$

6. Ejemplo trigonométrico, la función que representa la ecuación diferencial. Consideremos un ejemplo muy simple:

$$x''(t) = -x(t), \qquad x(0) = -1, \qquad x(\pi/3) = 1.$$

Programamos el lado derecho de la ecuación diferencial como una función `ftrigrhs`:

```
function [dd] = ftrigrhs(t, u, v),
   dd = - ???;
end
```

Por ejemplo, `ftrigrhs(0.2, 5, 3)` debe regresar `-5`.

7. Ejemplo trigonométrico, solución exacta. Resuelva el problema del ejemplo anterior a mano; busque la solución en forma

$$x(t) = C_1 \cos(t) + C_2 \operatorname{sen}(t).$$

Es fácil ver que la ecuación se satisface. Consideramos las condiciones de frontera:

$$\begin{aligned} -1 &= x(0) = C_1\square + C_2\square = \square, \\ 1 &= x(\pi/3) = \ldots \end{aligned}$$

De allí obtenemos $C_1 = \square$, $C_2 = \square$. Programamos la solución exacta:

```
function [v] = xtrigbvp(t),
   C1 = ???; C2 = ???;
   v = C1 * cos(t) + ??? * ???;
end
```

8. Programa. Suponemos que el esquema general de los métodos de un paso está programado como una función `onestepmethodvec`, y un paso de alguno de los métodos de Runge–Kutta está programado como `rk41step`.

```
function [] = games_with_linear_bvp(n),
   a = 0; b = pi/3; xa = -1; xb = 1;
   F = @(t, U) [ U(2), ftrigrhs(t, U(???), U(???)) ];
   # solve the initial value problem with initial values xa, 0:
   phi0 = [???, ???];
   [t, Phi] = onestepmethodvec(F, a, b, phi0, @rk41step, n);
   phi = Phi(:, 1);
   # solve the initial value problem with initial values ???, ???:
   psi0 = [???, ???];
   [t, Psi] = onestepmethodvec(???);
   psi = Psi(:, 1);
   # solve the boundary value problem:
   la = (xb -  phi(end) / (??? - ???);
   v = (1 - la) * phi + v * ???;
   plot(t, phi, t, psi, t, v);
end
```

9. Función que resuelve el problema. Separemos el algoritmo que resuelve el problema:

```
function [t, v] = solve_linear_bvp(f, a, b, xa, xb, n),
   F = @(t, U) [ U(2), f(???, ???, ???) ];
   [t, Phi] = onestepmethodvec(F, a, b, [???, ???], @rk41step, n);
   [t, Psi] = onestepmethodvec(???);
   phi = Phi(:, 1); psi = ???;
   la = ???;
   v = ???;
end
```

Escriba la función que hace la prueba:

```
function [] = test_solve_linear_bvp(),
   a = 0; b = pi/3; xa = -1; xb = 1; n = 1000;
   [t, v] = solve_linear_bvp(@ftrigrhs, ???, ???, ???, ???, ???);
   vexact = xtrigbvp(t);
   disp(norm(v - vexact));
end
```