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

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

Requisitos. Saber pasar de una EDO de segundo orden a una EDO vectorial de primer orden, haber programado varios métodos de Runge–Kutta (también se pueden usar métodos de Adams–Bashforth), incluso para sistemas de ecuaciones diferenciales, el método de disparo para resolver problemas de frontera lineales, y el método de bisección (o de la regla falsa).

1. La clase de problemas 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 $x_a, x_b \in \mathbb{R}$. Regresaremos los valores de x en n puntos que forman una malla uniforme entre a y b.

2. Ejemplo trigonométrico. Resolvamos el siguiente problema de frontera:

$$x''(t) = -x(t), \qquad x(0) = 0, \qquad x(2) = 1.$$

Programamos la función del lado derecho:

```
function [p] = fbvptrig(t, x, y),
   p = - x;
end
```

3. La función que realiza el método de bisección (repaso). Ya hemos programado una función que realiza el método de bisección. Suponemos que los argumentos son la función cuya raíz estamos buscando, los extremos del intervalo, las "tolerancias" de abscisas y ordenadas, y el número máximo de los pasos. La función regresa una aproximación a la raíz y el número de los pasos hechos.

```
[c, s] = bisec(g, a0, b0, xtol, ytol, smax)
```

4. De la EDO de segundo orden a un sistema de ecuaciones (repaso). Dada una ecuación de segunda orden de la forma

$$x''(t) = f(t, x(t), x'(t)), \tag{1}$$

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

$$\begin{aligned} x'(t) &= y(t), \\ y'(t) &= \end{aligned} \tag{2}$$

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 (2) se escribe como

$$X'(t) = \Big(X_2(t), \Big). \tag{3}$$

Denotamos por $F(t, X(t))$ a la expresión en el lado derecho de (3). 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:

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

5. Programar la función vectorial F en términos de f. Supongamos que f es una función de tres argumentos que se utiliza en la ecuación (1). Entonces la función F se puede definir de la siguiente manera (en la sintaxis de Matlab/Octave):

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

6. Problema auxiliar con valores iniciales. Consideremos la EDO original (1) con condiciones iniciales $x(a) = x_a$, $x'(a) = w$, y denotemos la solución de este problema por x_w. En otras palabras, estamos suponinendo que la función x_w satisface

$$x_w''(t) = f(t, x_w(t), \quad), \qquad x_w(a) = \quad, \quad x_w'(a) = \quad.$$

De manera equivalente, x_w es la primera componente de la función vectorial Φ_w que satisface la ecuación (3):

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

y la condición inicial

$$\Phi_w(a) = \Big(\quad, \quad \Big).$$

Nuestro objetivo es encontrar tal valor del parámetro w que la solución correspondiente satisfaga la condición del extremo derecho:

$$x_w(b) = \quad.$$

Notamos que para cada w la función x_w se puede calcular con alguno de los métodos numéricos para resolver los problemas de Cauchy, por ejemplo, con algún método de Runge–Kutta de orden 4 o 5.

7. Solución del problema de Cauchy vectorial (repaso). Anteriormente hemos programado varios métodos para resolver problemas de Cauchy, incluso en forma vectorial. Suponemos que `onestepmethodvec` es una función con los siguientes argumentos y resultados:

```
[t, v] = onestepmethodvec(F, tmin, tmax, Phi0, onestepformula, n)
```

Vamos a llamar la función `onestepmethodvec` con los siguientes argumentos:

- En vez de `Phi0` pasamos el vector fila que consiste de dos componentes: el valor de la función en el extremo izquierdo y el valor de la derivada en el extremo izquierdo.
- En vez de `F` pasamos el apuntador a la función `F` programada en el Ejercicio 5.
- En vez de `onestepformula` pasamos un apuntador a la función previamente programada `rk41step` o a otra función que realiza un paso de algún método de Runge–Kutta.

Notamos que el resultado `v` es una matriz de tamaño $(n+1) \times 2$. La primera columna contiene los valores de la función x_w, y la segunda columna los valores de x'_w.

8. El valor derecho como una función de la velocidad inicial. Programemos una función que resuelva el problema de Cauchy del Ejercicio 4 con velocidad inicial `w` y regrese el valor en la frontera derecha:

```
function [xright] = rightvalue(F, a, b, n, xa, w),
   [t, v] = onestepmethodvec(F, ???, ???, [???, ???], @rk41step, n);
   xright = v(end, ???);
   #en la etapa de depuracion podemos mostrar la grafica:
   #plot(t, v(:, 1));
end
```

9. El método de disparo para resolver problemas de frontera. Definimos la función

$$g(w) := x_w(b) - x_b,$$

suponiendo que $x_w(b)$ se calcula por la función `rightvalue`. Aplicamos a la función g el método de bisección en el intervalo entre `ya1` y `ya2`. De esta manera encontramos un valor de w tal que $g(w) \approx 0$. Devolvemos el vector de los valores x_w que corresponde al valor encontrado de w:

```
function [t, x] = shootingmethod(f, a, b, xa, xb, ya1, ya2, n),
   F = @(t, U) [U(2), ???];
   g = @(w) rightvalue(F, ???, ???, n, ???, ???) - xb;
   printf('ya1 = %.8f, x_{ya1}(b) - xb = %.2g\n', ya1, g(ya1));
   printf('ya2 = %.8f, x_{ya2}(b) - xb = %.2g\n', ya2, g(ya2));
   [w, s] = bisec(g, ???, ???, 1.0e-12, 1.0e-12, 100);
   printf('Number of bisection steps: %d\n', s);
   printf('w = %.8f, x_w(b) - xb = %.2g\n', w, g(w));
   [t, v] = onestepmethodvec(F, ???, ???, [???, ???], @rk41step, n);
   x = v(:, 1);
end
```

10. Comprobación para el ejemplo trigonométrico. Programamos la solución exacta del Problema 2:

```
function [v] = xtrig0201(t),
   C1 = ???; C2 = ???;
   v = C1 * cos(t) + C2 * sin(t);
end
```

En el método de disparo buscamos la velocidad inicial entre -3 y 3.

```
function [] = test_shooting(),
   1000; a = 0; b = 2; xa = 0; xb = 1; n = 1000;
   [t, x] = shootingmethod(@fbvptrig, a, b, xa, xb, -3, 3, n);
   solexact = xtrig0201(t);
   disp(norm(x - solexact, inf));
   plot(t, x, 'linewidth', 5);
end
```

11. Otros algoritmos para la búsqueda de la raíz. Se recomienda sustituir el método de bisección por el método de la regla falsa o por algún otro método aún más rápido, y comparar el número de los "disparos".

12. Otros ejemplos. Se recomienda buscar y programar un ejemplo más interesante.