Programación: el método de bisección

Este texto fue escrito por Egor Maximenko y Maria de los Angeles Isidro Perez.

Objetivos. Entender la idea del método de bisección, programar el método de bisección usando un ciclo `while`, probar la función programada con varios ejemplos.

Requisitos. El teorema del valor intermedio, la sintaxis del ciclo `while`, la sintaxis del operador condicional `if`, programación con funciones, usar funciones como argumentos de otras funciones.

El teorema del valor intermedio (repaso)

1. El primer teorema del valor intermedio (Bernhard Bolzano y Augustin-Louis Cauchy). Sean $a, b \in \mathbb{R}$ tales que $a < b$, sea $f\colon [a, b] \to \mathbb{R}$ una función continua que toma valores de signos opuestos en a y b, esto es,

$$\big(f(a) < 0 \quad \text{y} \quad f(b) > 0\big) \quad \text{o} \quad \big(f(a) \;\underline{\quad}\; 0 \quad \text{y} \quad f(b) \;\underline{\quad}\; 0\big). \tag{1}$$

Entonces existe al menos un punto c en el intervalo (a, b) tal que $f(c)$ ______.

2. Primer teorema del valor intermedio, sin fórmulas.
Si una función $\underbrace{\qquad\qquad\qquad\qquad}_{\text{¿con qué propiedad?}}$ cambia su signo en un intervalo cerrado,

es decir, en los extremos toma valores de signos ______,

entonces esta función $\underbrace{\qquad\qquad\qquad\qquad}_{\text{¿qué hace?}}$ en el intervalo abierto correspondiente.

3. Modelo. La función $f_1(x) \coloneqq x^2 - 2$ es continua en $\mathbb{R}$ (en particular, en $[1, 2]$) y toma valores de signos opuestos en los extremos del intervalo $[1, 2]$:

$$f_1(1) = \underline{\qquad}\;\underbrace{\quad}_{>/<}\;0, \qquad f_1(2) = \underline{\qquad}\;\underbrace{\quad}_{>/<}\;0.$$

Por el teorema del valor ______ existe un punto c en $(1, 2)$ tal que ______.
Dibujemos la gráfica de la función f_1 en un intervalo más amplio que $[1, 2]$:

```
f1 = @(x) x .^ 2 - 2;
xs = linspace(-1, 3, 201)';
plot(xs, f1(xs));
```

4. Ejemplos. Para cada una de las siguientes funciones f_k muestre que f_k toma valores de signos opuestos en los puntos a_k y b_k, además dibuje la gráfica de la función f_k en un intervalo más amplio que $[a_k, b_k]$. Las funciones f_k son continuas en $\mathbb{R}$ y por lo tanto en $[a_k, b_k]$.

$f_2(x) = \operatorname{sen}(x), \quad a_2 = 2,\ b_2 = 4.$

$$f_2(a_2) \approx \boxed{}, \qquad\qquad f_2(b_2) \approx \boxed{}.$$

$f_3(x) = x^5 + x - 1, \quad a_3 = 0,\ b_3 = 1.$

$$f_3(a_3) = \boxed{}, \qquad\qquad f_3(b_3) = \boxed{}.$$

$f_4(x) = e^x + x - 2, \quad a_4 = 0,\ b_4 = 1.$

$$f_4(a_4) = \boxed{}, \qquad\qquad f_4(b_4) \approx \boxed{}.$$

Idea del método de bisección

5. "Divide y vencerás". Sea $f\colon [a, b] \to \mathbb{R}$ una función continua que cumple con (1). Tomamos la aproximación a la raíz c como el punto medio del intervalo $[a, b]$:

$$c = \frac{a + b}{2}.$$

Si $f(c)$ tiene el mismo signo que $f(a)$, entonces f cambia su signo en $[\boxed{}, \boxed{}]$, y tiene que existir una raíz en este intervalo. Pasamos a este intervalo:

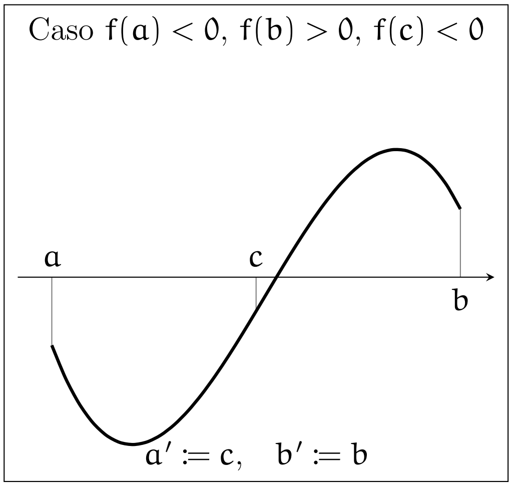
$$a_{\text{nuevo}} \coloneqq \underbrace{\boxed{}}_{a/b/c}, \qquad b_{\text{nuevo}} \coloneqq \underbrace{\boxed{}}_{a/b/c}.$$

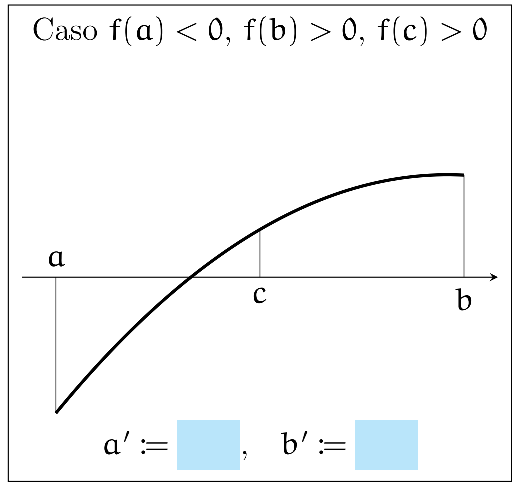
Si $f(c)$ tiene el mismo signo que $f(b)$, entonces f cambia su signo en $[\boxed{}, \boxed{}]$, y tiene que existir una raíz en el último intervalo. Pasamos a este intervalo:

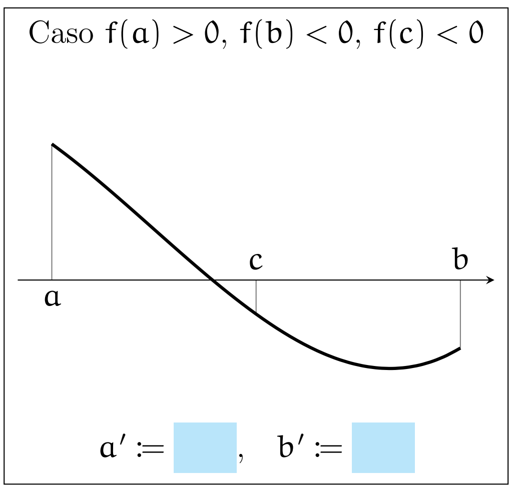
$$a_{\text{nuevo}} \coloneqq \underbrace{\boxed{}}_{a/b/c}, \qquad b_{\text{nuevo}} \coloneqq \underbrace{\boxed{}}_{a/b/c}.$$

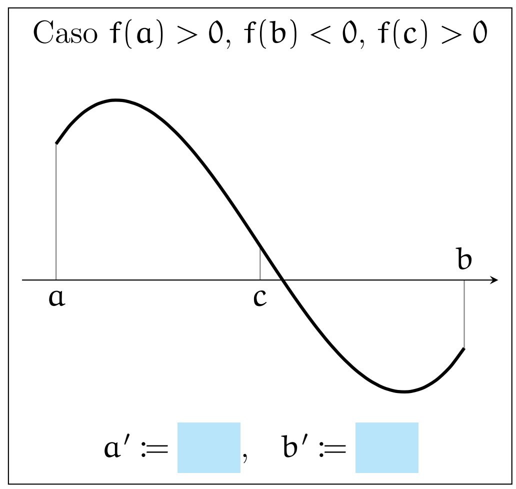
En ambos casos, hemos reducido el intervalo de busqueda a la mitad.

6. Los siguientes dibujos muestran la elección de los extremos nuevos del intervalo, denotados por a' y b', dependiendo de los signos de $f(a)$, $f(b)$ y $f(c)$.









7. Forma breve para escribir la condición que la función tiene signos opuestos en los extremos del intervalo. La condición (1) se puede escribir brevemente de la siguiente manera:

$$f(a)f(b) \underbrace{\qquad}_{>/<} 0. \tag{2}$$

Puede ser que los números $f(a)$ y $f(b)$ son muy pequeños, y el producto $f(a)f(b)$ se representa en la máquina como el cero ("arithmetic underflow"). Para evitar esta situación, en vez de (2) se recomienda usar la siguiente condición que matemáticamente es equivalente a (2), pero es más segura para los cálculos:

$$\operatorname{sign}(f(a))\operatorname{sign}(f(b)) \underbrace{\qquad}_{>/<} 0. \tag{3}$$

Algo termina, algo continúa

8. Terminar cuando hemos captado la raíz en un intervalo pequeño. Si en algún momento nuestro intervalo $[a, b]$ que contiene a la raíz se hizo muy corto, digamos su longitud es menor que un número $\mathtt{xtol}$ dado, entonces podemos terminar los cálculos y regresar cualquier punto de este intervalo.

9. Terminar cuando el valor de la función es demasiado pequeño. Habitualmente calculamos los valores de f aproximadamente. Si $|f(c)|$ es menor que una precisión dada, denotada por $\mathtt{ytol}$, entonces no hay sentido tomar en cuenta el signo de $f(c)$, y lo más natural es terminar los cálculos y regresar c como una aproximación a la raíz.

10. Terminar cuando ya hemos trabajado demasiado. Por seguridad, podemos acotar el número máximo de iteraciones permitidas por un número $\mathtt{smax}$.

11. Condición de terminación. ¿Hasta qué momento continuar el proceso de bisección? Suponemos que desde el inicio están dados dos números reales positivos $\mathtt{xtol}$, $\mathtt{ytol}$, y un número entero positivo $\mathtt{smax}$. Nuestra función debe pararse en cualquiera de los siguientes tres casos:

- Cuando el número s de las iteraciones realizadas es mayor o igual que $\mathtt{smax}$.
- Cuando $|f(c)| < \mathtt{ytol}$, donde c es la aproximación obtenida de la raíz.
- Cuando hemos encontrado un intervalo de longitud $\leq \mathtt{xtol}$ que contiene una raíz de la función.

Resumen: la condición de terminación es

$$\Big(s \underbrace{\boxed{}}_{\geq/\leq} \mathtt{smax}\Big) \underbrace{\boxed{}}_{\vee/\wedge} \Big(b - a \underbrace{\boxed{}}_{>/<} \mathtt{xtol}\Big) \underbrace{\boxed{}}_{\vee/\wedge} \Big(\text{abs}(\boxed{}) \underbrace{\boxed{}}_{</>} \mathtt{ytol}\Big).$$

12. Condición de continuación. ¿Cuándo se continua el proceso? Cuando todavía no se cumple la condición para terminar. Entonces la condición de continuación se obtiene al

$$\underbrace{\boxed{}}_{\text{repetir/negar/olvidar/ignorar}}$$

la condición de terminación. Utilice la ley de Augustus De $\underbrace{\boxed{}}_{\text{un matemático y lógico británico}}$ para escribir la condición de continuación:

$$\boxed{} \underbrace{\boxed{}}_{\vee/\wedge} \boxed{} \underbrace{\boxed{}}_{\vee/\wedge} \boxed{}$$

13. Operaciones lógicas en el lenguaje Matlab/Octave. Ejecute los siguientes tres comandos uno por uno en el intérprete:

```
true && false && true
true || false || true
true && true && true
```

14. Guardar los valores en los puntos a, b, c. Por lo común, la operación más costosa es la evaluación de la función `f` en un punto dado. Para no evaluar `f` varias veces en el mismo punto, guardaremos $f(a)$, $f(b)$, $f(c)$ en variables especiales `fa`, `fa`, `fb`. Cuando cambiamos a o b, reasignamos también `fa` o `fb`, respectivamente. Antes del ciclo no tenemos c, por eso definimos `fc` como un número más grande que `ytol`, para garantizar que se cumple la condición

$$|f(c)| \underbrace{\boxed{}}_{\geq/\leq} \mathrm{ytol}.$$

15. Programar la función bisec. Completar el código de la siguiente función, guardarla en un archivo `bisec.m`. La función regresa una aproximación de la raíz y el número de los pasos hechos.

```
function [c, s] = bisec(f, a0, b0, xtol, ytol, smax),
   a = ???; fa = f(???);
   b = ???; fb = f(???);
   fc = ytol + 1;
   s = ???;
   while ???,
      c = ???;
      fc = ???;
      if ???,
         a = ???; fa = ???;
      else,
         b = ???; fb = ???;
      end
      s = s + ???;
   end
end
```

16. Pruebas. Programamos una función `testbisec` que aplica la función `bisec` a los ejemplos que vimos en el Ejercicio 4. Se recomienda guardar en un archivo `testbisec.m` las tres siguientes funciones juntas:

```
function [] = testbisec(),
   f1 = @(x) x .^ 2 - 2;
   [c1, s1] = bisec(f1, 1.0, 2.0, 1.0E-10, 1.0E-10, 100);
   showonetest('Ejemplo 1', s1, c1, f1(c1));
   [c2, s2] = bisec(@sin, ???, ???, 1.0E-10, 1.0E-10, 100);
   showonetest('Ejemplo 2', s2, c2, sin(c2));
   f3 = @(x) ???;
   [c3, s3] = bisec(f3, ???);
   showonetest(???);
   [c4, s4] = bisec(@f4, ???);
   showonetest(???);
end

function [] = showonetest(title, s, c, v),
   printf('%s. Despues de %d pasos obtenemos...\n', title, s);
   printf('el punto %.10f donde la funcion vale %.2e.\n\n', c, v);
end

function [y] = f4(x),
   y = ???;
end
```

En el primer argumento de `printf` se recomienda utilizar comillas simples. Después de escribir y guardar `testbisec.m`, en el intérprete de Matlab/Octave es suficiente cambiar la carpeta actual a la carpeta donde están guardados los archivos `bisec.m` y `testbisec.m` (pueden ser útiles los comandos `pwd`, `cd` y `ls`), y ejecutar

```
   testbisec()
```

17. Más ejemplos. Invente algún otro ejemplo y agréguelo en la prueba.

18. Ejercicio. El código de la prueba se puede escribir de manera más elegante y concisa. La llamada de la función `bisec` se puede mover adentro la función `showonetest`; por supuesto, se modifica la lista de los argumentos de `showonetest`. Entonces el código de la función `testbisec` puede ser así:

```
...
f1 = ???;
showonetest('Ejemplo 1', f1, 1.0, 2.0, 1.0E-10, 1.0E-10, 100);
showonetest('Ejemplo 2', @sin, ???);
...
```