Bisection Method

Tamas Kis | kis@stanford.edu | https://github.com/tamaskis

Contents

1 BISECTION METHOD

The **bisection method** can be used to find the root of a univariate function $f(x)$, with no restrictions on the differentiability of f. The basic idea behind the bisection is starting off with some interval $[a, b]$ containing a root, iteratively "shrinking" this interval until it is below some tolerance threshold, and then taking the root to be the midpoint of this interval. The general procedure is as follows:

1. Make an initial guess for the interval $[a, b]$ containing the root.
2. Assume the root, c, is the midpoint of this interval: $c = (a + b)/2$.
3. Evaluate $f(a)$ and $f(c)$.
   (a) If $f(a) < 0$ and $f(c) > 0$ (i.e. they have different sign), we know the true root is contained in the interval $[a, c]$. Therefore, we update our interval so that a remains the same, but b is updated to be c.
   (b) If $f(a)$ and $f(c)$ have the same sign (either both negative or both positive), we know the true root must be contained in the interval $[c, b]$. Therefore, we update our interval so that b remains the same, but a is updated to be c.
4. Repeating steps 2 and 3, the interval $[a, b]$ will keep shrinking. Once the difference $(b - a)$ is small enough, we say that the estimate of the root has **converged** to the true root, within some **tolerance** (which we denote as TOL). Therefore, if we predetermine that, at most, the root must be restricted to an interval of length TOL, we will keep repeating steps 2 and 3 until $(b - a) < \mathrm{TOL}$.

In some cases, the difference $(b - a)$ may never decrease below TOL, or take too long to decrease to below TOL. Therefore, we also define the **maximum number of iterations** ($i_{\max}$) so that the algorithm does not keep iterating forever, or for too long of a time [1, 2].

There are two basic algorithms for implementing the bisection method. The first implementation, shown in Algorithm 1 below, does *not* store the result of each iteration. On the other hand, the second implementation, shown in Algorithm 2, *does* store the result of each iteration. `bisection_method` implements both of these algorithms.

Since Algorithm 2 first needs to preallocate a potentially huge array to store all of the intermediate solutions, Algorithm 1 is significantly faster. Even if $i_{\max}$ (determines size of the preallocated array) is set to be a small number (for example, 10), Algorithm 1 is still faster. The reason we still consider and implement Algorithm 2 is so that convergence studies may be performed.

Algorithm 1:
Bisection method (fast implementation).

Given:
- $f(x)$ - function
- a - lower bound for initial guess of interval with root
- b - upper bound for initial guess of interval with root
- TOL - tolerance
- $i_{\max}$ - maximum number of iterations

Procedure:
1. Initial guess for root.
$$c = \frac{a + b}{2}$$
2. Initialize x_{new} so its scope will not be limited to within the while loop.
$$x_{\text{new}} = 0$$
3. Find the root using the bisection method.

while $((b - a) > \mathrm{TOL})$ **and** $(i < i_{\max})$

(a) Update interval.

if $f(c) = 0$
    Stop
else if $\operatorname{sgn}[f(c)] = \operatorname{sgn}[f(a)]$
    $a = c$
else
    $b = c$
end

(b) Update root estimate.

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

(c) Increment loop index.

$$i = i + 1$$

end

Return:

- root $= c$ - converged root

Algorithm 2:

Bisection method (storing intermediate root estimates).

Given:

- $f(x)$ - function
- a - lower bound for initial guess of interval with root
- b - upper bound for initial guess of interval with root
- TOL - tolerance
- $i_{\max}$ - maximum number of iterations

Procedure:

1. Preallocate $\mathbf{x} \in \mathbb{R}^{i_{\max}}$ to store the estimates of the root at each iteration.
2. Store the initial guess for the root in the first element of $\mathbf{x}$.

$$x_1 = \frac{a+b}{2}$$

3. Find the root using the bisection method.

while $((b - a) > \text{TOL})$ **and** $(i < i_{\max})$

(a) Update interval.

if $f(x_i) = 0$

  Stop

else if $\operatorname{sgn}[f(x_i)] = \operatorname{sgn}[f(a)]$

  $a = x_i$

else

  $b = x_i$

end

(b) Update root estimate.

$$x_{i+1} = \frac{a+b}{2}$$

(c) Increment loop index.

$$i = i + 1$$

end

Return:

- `x` - vector where the first element is the initial guess for the root, the subsequent elements are the intermediate root estimates, and the final element is the converged root

REFERENCES

[1] *Bisection method*. https://en.wikipedia.org/wiki/Bisection_method. (accessed: February 7, 2020).

[2] James Hateley. *Nonlinear Equations*. MATH 3620 Course Reader (Vanderbilt University). 2019.