

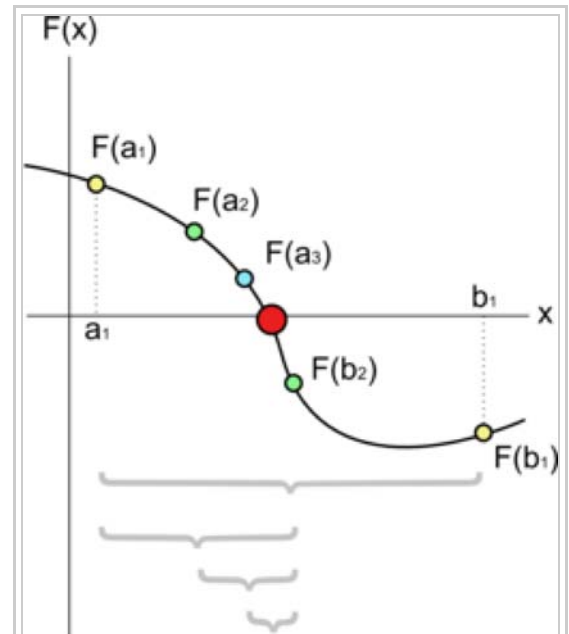
Bisection method

From Wikipedia, the free encyclopedia

In mathematics, the **bisection method** is a root-finding algorithm which repeatedly divides an interval in half and then selects the subinterval in which a root exists. It is a very simple and robust method, but it is also rather slow.

Contents

- 1 The method
- 2 Pseudo-code
- 3 Analysis
- 4 See also
- 5 References
- 6 External links



A few steps of the bisection method applied over the starting range $[a_1; b_1]$. The bigger red dot is the root of the function.

The method

Suppose we want to solve the equation

$$f(x) = 0,$$

where f is a continuous function.

The bisection method starts with two points a and b such that $f(a)$ and $f(b)$ have opposite signs. The intermediate value theorem says that f must have at least one root in the interval $[a, b]$. The method now divides the interval in two by computing $c = (a+b) / 2$. There are now two possibilities: either $f(a)$ and $f(c)$ have opposite signs, or $f(c)$ and $f(b)$ have opposite signs. The bisection algorithm is then applied recursively to the sub-interval where the sign change occurs.

Explicitly, if $f(a)f(c) < 0$, then the method sets b equal to c , and if $f(b)f(c) < 0$, then the method sets a equal to c . In both cases, $f(a)$ and $f(b)$ have again opposite signs, so the method can start again with the points a and b which now lie closer to each other.

Pseudo-code

Here is a representation of the bisection method in Visual Basic code. The variables `left` and `right` correspond to a and b above. The initial `left` and `right` must be chosen so that $f(\text{left})$ and $f(\text{right})$ are
en.wikipedia.org/wiki/Bisection_method

correspond to a and b above. The initial `left` and `right` must be chosen so that $f(\text{left})$ and $f(\text{right})$ are of opposite sign (they 'bracket' a root). The variable `epsilon` specifies how precise the result will be.

```
'Bisection Method
'Start loop
Do While (abs(right - left) > 2*epsilon)

  'Calculate midpoint of domain
  midpoint = (right + left) / 2

  'Find f(midpoint)
  If ((f(left) * f(midpoint)) > 0) Then
    'Throw away left half
    left = midpoint
  Else
    'Throw away right half
    right = midpoint
  End If
Loop
Return (right + left) / 2
```

Analysis

If f is a continuous function on the interval $[a, b]$ and $f(a)f(b) < 0$, then the bisection method converges to a root of f . In fact, the absolute error is halved at each step. Thus, the method converges linearly, which is quite slow. On the positive side, the method is guaranteed to converge if $f(a)$ and $f(b)$ have different signs.

The bisection method gives only a range where the root exists, rather than a single estimate for the root's location. Without using any other information, the best estimate for the location of the root is the midpoint of the range. In that case, the absolute error after n steps is at most

$$\frac{|b - a|}{2^{n+1}}$$

If either endpoint of the interval is used, then the maximum absolute error is

$$\frac{|b - a|}{2^n},$$

the entire length of the interval.

These formulas can be used to find the number of iterations that the bisection method needs to converge to a root within a certain tolerance. For instance, using the second formula for the error, the number of iterations n has to satisfy

$$n > \frac{\log(b - a) - \log \varepsilon}{\log 2}$$

to make sure that the error is smaller than the tolerance ε .

If f has several roots in the interval $[a, b]$, then the bisection method finds the odd-numbered roots with equal, non-zero probability and the even-numbered roots with zero probability. More precisely, suppose that f has $2k + 1$ simple roots $x_1 < x_2 < \dots < x_{2k+1}$ in the interval $[a, b]$ (the number of roots is odd because $f(a)$ and $f(b)$ have opposite signs). Assume that the roots are distributed independently and uniformly in this interval. Then, the probability that the bisection method converges to the root x_i with $i = 1, 2, \dots, 2k + 1$ is zero if i is even and $1 / (k + 1)$ if i is odd (Corliss 1977).

See also

- Binary search algorithm
- Lehmer-Schur algorithm, generalization of the bisection method in the complex plane
- Nested intervals

References

- Burden, Richard L.; Faires, J. Douglas (2000), *Numerical Analysis* (7th ed.), Brooks/Cole, ISBN 978-0-534-38216-2.
- Corliss, George (1977), "Which root does the bisection algorithm find?", *SIAM Review* **19** (2): 325–327, ISSN 1095-7200.
- Kaw, Autar; Kalu, Egwu (2008), *Numerical Methods with Applications* (1st ed.), www.autarkaw.com.

External links

- Bisection Method (<http://twf.mpei.ac.ru/TTHB/1/Bisection.htm>) on Mathcad Application Server.
- Bisection Method (http://numericalmethods.eng.usf.edu/topics/bisection_method.html) Notes, PPT, Mathcad, Maple, Matlab, Mathematica from Holistic Numerical Methods Institute (<http://numericalmethods.eng.usf.edu>)
- Module for the Bisection Method by John H. Mathews (<http://math.fullerton.edu/mathews/n2003/BisectionMod.html>)
- Java Code for Bisection Method by Behzad Torkian (http://www.torkian.info/Site/Research/Entries/2008/2/28_Root-finding_algorithm.html)
- True example of using bisection method in computer programming (<http://isoelectric.ovh.org/>) free program to isoelectric point calculation

Retrieved from "http://en.wikipedia.org/wiki/Bisection_method"

Categories: Root-finding algorithms

-
- This page was last modified on 25 March 2009, at 17:40 (UTC).
 - All text is available under the terms of the GNU Free Documentation License. (See **Copyrights** for details.)

4/11/2009

Bisection method - Wikipedia, the fre...

Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a U.S. registered 501(c)(3) tax-deductible nonprofit charity.