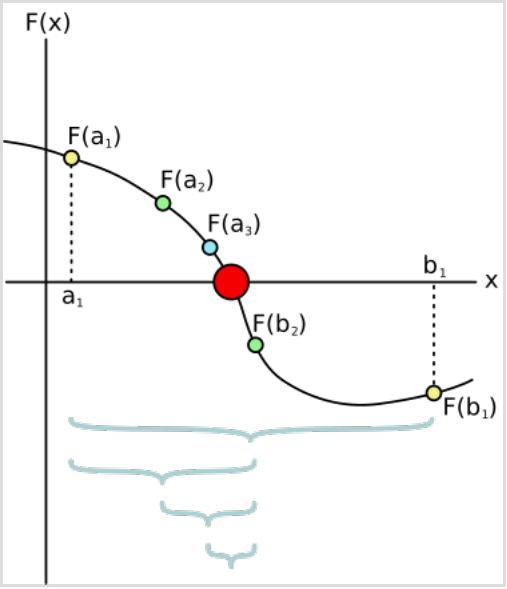
Bisection method

In mathematics, the **bisection method** is a root-finding method that applies to any continuous function for which one knows two values with opposite signs. The method consists of repeatedly bisecting the interval defined by these values and then selecting the subinterval in which the function changes sign, and therefore must contain a root. It is a very simple and robust method, but it is also relatively slow. Because of this, it is often used to obtain a rough approximation to a solution which is then used as a starting point for more rapidly converging methods.[1] The method is also called the **interval halving** method,[2] the **binary search method**,[3] or the **dichotomy method**.[4]

For polynomials, more elaborate methods exist for testing the existence of a root in an interval (Descartes' rule of signs, Sturm's theorem, Budan's theorem). They allow extending the bisection method into efficient algorithms for finding all real roots of a polynomial; see Real-root isolation.



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

The method is applicable for numerically solving the equation $f(x) = 0$ for the real variable x, where f is a continuous function defined on an interval $[a, b]$ and where $f(a)$ and $f(b)$ have opposite signs. In this case a and b are said to bracket a root since, by the intermediate value theorem, the continuous function f must have at least one root in the interval (a, b).

At each step the method divides the interval in two parts/halves by computing the midpoint $c = (a+b) / 2$ of the interval and the value of the function $f(c)$ at that point. If c itself is a root then the process has succeeded and stops. Otherwise, there are now only two possibilities: either $f(a)$ and $f(c)$ have opposite signs and bracket a root, or $f(c)$ and $f(b)$ have opposite signs and bracket a root.[5] The method selects the subinterval that is guaranteed to be a bracket as the new interval to be used in the next step. In this way an interval that contains a zero of f is reduced in width by 50% at each step. The process is continued until the interval is sufficiently small.

Explicitly, if $f(c)=0$ then c may be taken as the solution and the process stops. Otherwise, if $f(a)$ and $f(c)$ have opposite signs, then the method sets c as the new value for b, and if $f(b)$ and $f(c)$ have opposite signs then the method sets c as the new a. In both cases, the new $f(a)$ and $f(b)$ have opposite signs, so the method is applicable to this smaller interval.[6]

Iteration tasks

The input for the method is a continuous function f, an interval $[a, b]$, and the function values $f(a)$ and $f(b)$. The function values are of opposite sign (there is at least one zero crossing within the interval). Each iteration performs these steps:

1. Calculate c, the midpoint of the interval, $c = \frac{a + b}{2}$.
2. Calculate the function value at the midpoint, $f(c)$.
3. If convergence is satisfactory (that is, $c - a$ is sufficiently small, or $|f(c)|$ is sufficiently small), return c and stop iterating.
4. Examine the sign of $f(c)$ and replace either $(a, f(a))$ or $(b, f(b))$ with $(c, f(c))$ so that there is a zero crossing within the new interval.

When implementing the method on a computer, there can be problems with finite precision, so there are often additional convergence tests or limits to the number of iterations. Although f is continuous, finite precision may preclude a function value ever being zero. For example, consider $f(x) = \cos x$; there is no floating-point value approximating $x = \pi/2$ that gives exactly zero. Additionally, the difference between a and b is limited by the floating point precision; i.e., as the difference between a and b decreases, at some point the midpoint of $[a, b]$ will be numerically identical to (within floating point precision of) either a or b.

Algorithm

The method may be written in pseudocode as follows:[7]

```
input: Function f,
       endpoint values a, b,
       tolerance TOL,
       maximum iterations NMAX
conditions: a < b,
            either f(a) < 0 and f(b) > 0 or f(a) > 0 and f(b) < 0
output: value which differs from a root of f(x) = 0 by less than TOL

N ← 1
while N ≤ NMAX do // limit iterations to prevent infinite loop
    c ← (a + b)/2 // new midpoint
    if f(c) = 0 or (b − a)/2 < TOL then // solution found
        Output(c)
        Stop
    end if
    N ← N + 1 // increment step counter
    if sign(f(c)) = sign(f(a)) then a ← c else b ← c // new interval
end while
Output("Method failed.") // max number of steps exceeded
```

Example: Finding the root of a polynomial

Suppose that the bisection method is used to find a root of the polynomial

$$f(x) = x^3 - x - 2\,.$$

First, two numbers a and b have to be found such that $f(a)$ and $f(b)$ have opposite signs. For the above function, $a = 1$ and $b = 2$ satisfy this criterion, as

$$f(1) = (1)^3 - (1) - 2 = -2$$

and

$$f(2) = (2)^3 - (2) - 2 = +4\,.$$

Because the function is continuous, there must be a root within the interval [1, 2].

In the first iteration, the end points of the interval which brackets the root are $a_1 = 1$ and $b_1 = 2$, so the midpoint is

$$c_1 = \frac{2+1}{2} = 1.5$$

The function value at the midpoint is $f(c_1) = (1.5)^3 - (1.5) - 2 = -0.125$. Because $f(c_1)$ is negative, $a = 1$ is replaced with $a = 1.5$ for the next iteration to ensure that $f(a)$ and $f(b)$ have opposite signs. As this continues, the interval between a and b will become increasingly smaller, converging on the root of the function. See this happen in the table below.

Iteration	a_n	b_n	c_n	$f(c_n)$
1	1	2	1.5	−0.125
2	1.5	2	1.75	1.6093750
3	1.5	1.75	1.625	0.6660156
4	1.5	1.625	1.5625	0.2521973
5	1.5	1.5625	1.5312500	0.0591125
6	1.5	1.5312500	1.5156250	−0.0340538
7	1.5156250	1.5312500	1.5234375	0.0122504
8	1.5156250	1.5234375	1.5195313	−0.0109712
9	1.5195313	1.5234375	1.5214844	0.0006222
10	1.5195313	1.5214844	1.5205078	−0.0051789
11	1.5205078	1.5214844	1.5209961	−0.0022794
12	1.5209961	1.5214844	1.5212402	−0.0008289
13	1.5212402	1.5214844	1.5213623	−0.0001034
14	1.5213623	1.5214844	1.5214233	0.0002594
15	1.5213623	1.5214233	1.5213928	0.0000780

After 13 iterations, it becomes apparent that there is a convergence to about 1.521: a root for the polynomial.

Analysis

The method is guaranteed to converge to a root of f if f is a continuous function on the interval $[a, b]$ and $f(a)$ and $f(b)$ have opposite signs. The absolute error is halved at each step so the method converges linearly. Specifically, if $c_1 = \frac{a+b}{2}$ is the midpoint of the initial interval, and c_n is the midpoint of the interval in the nth step, then the difference between c_n and a solution c is

bounded by[8]

$$|c_n - c| \leq \frac{|b-a|}{2^n}.$$

This formula can be used to determine, in advance, an upper bound on the number of iterations that the bisection method needs to converge to a root to within a certain tolerance. The number n of iterations needed to achieve a required tolerance ε (that is, an error guaranteed to be at most ε), is bounded by

$$n \leq n_{1/2} \equiv \left\lceil \log_2\left(\frac{\epsilon_0}{\epsilon}\right)\right\rceil,$$

where the initial bracket size $\epsilon_0 = |b-a|$ and the required bracket size $\epsilon \leq \epsilon_0$. The main motivation to use the bisection method is that over the set of continuous functions, no other method can guarantee to produce an estimate c_n to the solution c that in the *worst case* has an ϵ absolute error with less than $n_{1/2}$ iterations.[9] This is also true under several common assumptions on function f and the behaviour of the function in the neighbourhood of the root.[9][10]

However, despite the bisection method being optimal with respect to worst case performance under absolute error criteria it is sub-optimal with respect to *average performance* under standard assumptions [11][12] as well as *asymptotic performance*.[13] Popular alternatives to the bisection method, such as the secant method, Ridders' method or Brent's method (amongst others), typically perform better since they trade-off worst case performance to achieve higher orders of convergence to the root. And, a strict improvement to the bisection method can be achieved with a higher order of convergence without trading-off worst case performance with the ITP Method.[13][14]

Generalization to higher dimensions

The bisection method has been generalized to multi-dimensional functions. Such methods are called **generalized bisection methods**.[15][16]

Methods based on degree computation

Some of these methods are based on computing the topological degree, which for a bounded region $\Omega \subseteq \mathbb{R}^n$ and a differentiable function $f : \mathbb{R}^n \to \mathbb{R}^n$ is defined as a sum over its roots:

$$\deg(f, \Omega) := \sum_{y \in f^{-1}(0)} \operatorname{sgn} \det(Df(y)),$$

where $Df(y)$ is the Jacobian matrix, $\mathbf{0} = (0, 0, \ldots, 0)^T$, and

$$\operatorname{sgn}(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases}$$

is the sign function.[17] In order for a root to exist, it is sufficient that $\deg(f, \Omega) \neq 0$, and this can be verified using a surface integral over the boundary of Ω.[18]

Characteristic bisection method

The **characteristic bisection method** uses only the signs of a function in different points. Lef f be a function from R^{d} to R^{d}, for some integer $d \geq 2$. A **characteristic polyhedron**[19] (also called an **admissible polygon**)[20] of f is a polytope in R^{d}, having 2^{d} vertices, such that in each vertex $\mathbf{v}$, the combination of signs of $f(\mathbf{v})$ is unique and the topological degree of f on its interior is not zero (a necessary criterion to ensure the existence of a root).[21] For example, for d=2, a characteristic polyhedron of f is a quadrilateral with vertices (say) A,B,C,D, such that:

- $\operatorname{sgn} f(A) = (-,-)$, that is, f_1(A)<0, f_2(A)<0.
- $\operatorname{sgn} f(B) = (-,+)$, that is, f_1(B)<0, f_2(B)>0.
- $\operatorname{sgn} f(C) = (+,-)$, that is, f_1(C)>0, f_2(C)<0.
- $\operatorname{sgn} f(D) = (+,+)$, that is, f_1(D)>0, f_2(D)>0.

A **proper edge** of a characteristic polygon is a edge between a pair of vertices, such that the sign vector differs by only a single sign. In the above example, the proper edges of the characteristic quadrilateral are AB, AC, BD and CD. A **diagonal** is a pair of vertices, such that the sign vector differs by all d signs. In the above example, the diagonals are AD and BC.

At each iteration, the algorithm picks a proper edge of the polyhedron (say, A—B), and computes the signs of f in its mid-point (say, M). Then it proceeds as follows:

- If $\operatorname{sgn} f(M) = \operatorname{sgn}(A)$, then A is replaced by M, and we get a smaller characteristic polyhedron.
- If $\operatorname{sgn} f(M) = \operatorname{sgn}(B)$, then B is replaced by M, and we get a smaller characteristic polyhedron.
- Else, we pick a new proper edge and try again.

Suppose the diameter (= length of longest proper edge) of the original characteristic polyhedron is D. Then, at least $\log_2(D/\varepsilon)$ bisections of edges are required so that the diameter of the remaining polygon will be at most ε.[20]:11, Lemma.4.7 If the topological degree of the initial polyhedron is not zero, then there is a procedure that can choose an edge such that the next polyhedron also has nonzero degree.[21][22]

See also

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

References

1. Burden & Faires 1985, p. 31

2. "Interval Halving (Bisection)" (https://web.archive.org/web/20130519092250/http://siber.cankaya.edu.tr/NumericalComputations/ceng375/node32.html). Archived from the original (http://siber.cankaya.edu.tr/NumericalComputations/ceng375/node32.html) on 2013-05-19. Retrieved 2013-11-07.
3. Burden & Faires 1985, p. 28
4. "Dichotomy method - Encyclopedia of Mathematics" (https://www.encyclopediaofmath.org/index.php/Dichotomy_method). *www.encyclopediaofmath.org*. Retrieved 2015-12-21.
5. If the function has the same sign at the endpoints of an interval, the endpoints may or may not bracket roots of the function.
6. Burden & Faires 1985, p. 28 for section
7. Burden & Faires 1985, p. 29. This version recomputes the function values at each iteration rather than carrying them to the next iterations.
8. Burden & Faires 1985, p. 31, Theorem 2.1
9. Sikorski, K. (1982-02-01). "Bisection is optimal" (https://doi.org/10.1007/BF01459080). *Numerische Mathematik*. **40** (1): 111–117. doi:10.1007/BF01459080 (https://doi.org/10.1007%2FBF01459080). ISSN 0945-3245 (https://search.worldcat.org/issn/0945-3245). S2CID 119952605 (https://api.semanticscholar.org/CorpusID:119952605).
10. Sikorski, K (1985-12-01). "Optimal solution of nonlinear equations". *Journal of Complexity*. **1** (2): 197–209. doi:10.1016/0885-064X(85)90011-1 (https://doi.org/10.1016%2F0885-064X%2885%2990011-1). ISSN 0885-064X (https://search.worldcat.org/issn/0885-064X).
11. Graf, Siegfried; Novak, Erich; Papageorgiou, Anargyros (1989-07-01). "Bisection is not optimal on the average" (https://doi.org/10.1007/BF01396051). *Numerische Mathematik*. **55** (4): 481–491. doi:10.1007/BF01396051 (https://doi.org/10.1007%2FBF01396051). ISSN 0945-3245 (https://search.worldcat.org/issn/0945-3245). S2CID 119546369 (https://api.semanticscholar.org/CorpusID:119546369).
12. Novak, Erich (1989-12-01). "Average-case results for zero finding" (https://doi.org/10.1016%2F0885-064X%2889%2990022-8). *Journal of Complexity*. **5** (4): 489–501. doi:10.1016/0885-064X(89)90022-8 (https://doi.org/10.1016%2F0885-064X%2889%2990022-8). ISSN 0885-064X (https://search.worldcat.org/issn/0885-064X).
13. Oliveira, I. F. D.; Takahashi, R. H. C. (2020-12-06). "An Enhancement of the Bisection Method Average Performance Preserving Minmax Optimality" (https://doi.org/10.1145/3423597). *ACM Transactions on Mathematical Software*. **47** (1): 5:1–5:24. doi:10.1145/3423597 (https://doi.org/10.1145%2F3423597). ISSN 0098-3500 (https://search.worldcat.org/issn/0098-3500). S2CID 230586635 (https://api.semanticscholar.org/CorpusID:230586635).
14. Ivo, Oliveira (2020-12-14). "An Improved Bisection Method" (https://link.growkudos.com/1iwxps83474). doi:10.1145/3423597 (https://doi.org/10.1145%2F3423597). S2CID 230586635 (https://api.semanticscholar.org/CorpusID:230586635).
15. Mourrain, B.; Vrahatis, M. N.; Yakoubsohn, J. C. (2002-06-01). "On the Complexity of Isolating Real Roots and Computing with Certainty the Topological Degree" (https://doi.org/10.1006%2Fjcom.2001.0636). *Journal of Complexity*. **18** (2): 612–640. doi:10.1006/jcom.2001.0636 (https://doi.org/10.1006%2Fjcom.2001.0636). ISSN 0885-064X (https://search.worldcat.org/issn/0885-064X).
16. Vrahatis, Michael N. (2020). "Generalizations of the Intermediate Value Theorem for Approximating Fixed Points and Zeros of Continuous Functions" (https://link.springer.com/chapter/10.1007/978-3-030-40616-5_17). In Sergeyev, Yaroslav D.; Kvasov, Dmitri E. (eds.). *Numerical Computations: Theory and Algorithms*. Lecture Notes in Computer Science. Vol. 11974. Cham: Springer International Publishing. pp. 223–238. doi:10.1007/978-3-030-40616-5_17 (https://doi.org/10.1007%2F978-3-030-40616-5_17). ISBN 978-3-030-40616-5. S2CID 211160947 (https://api.semanticscholar.org/CorpusID:211160947).

17. Polymilis, C.; Servizi, G.; Turchetti, G.; Skokos, Ch.; Vrahatis, M. N. (May 2003). "LOCATING PERIODIC ORBITS BY TOPOLOGICAL DEGREE THEORY". *Libration Point Orbits and Applications*: 665–676. arXiv:nlin/0211044 (https://arxiv.org/abs/nlin/0211044). doi:10.1142/9789812704849_0031 (https://doi.org/10.1142%2F9789812704849_0031).
18. Kearfott, Baker (1979-06-01). "An efficient degree-computation method for a generalized method of bisection" (https://doi.org/10.1007/BF01404868). *Numerische Mathematik*. **32** (2): 109–127. doi:10.1007/BF01404868 (https://doi.org/10.1007%2FBF01404868). ISSN 0945-3245 (https://search.worldcat.org/issn/0945-3245). S2CID 122058552 (https://api.semanticscholar.org/CorpusID:122058552).
19. Vrahatis, Michael N. (1995-06-01). "An Efficient Method for Locating and Computing Periodic Orbits of Nonlinear Mappings" (https://www.sciencedirect.com/science/article/pii/S0021999185711199). *Journal of Computational Physics*. **119** (1): 105–119. Bibcode:1995JCoPh.119..105V (https://ui.adsabs.harvard.edu/abs/1995JCoPh.119..105V). doi:10.1006/jcph.1995.1119 (https://doi.org/10.1006%2Fjcph.1995.1119). ISSN 0021-9991 (https://search.worldcat.org/issn/0021-9991).
20. Vrahatis, M. N.; Iordanidis, K. I. (1986-03-01). "A rapid Generalized Method of Bisection for solving Systems of Non-linear Equations" (https://doi.org/10.1007/BF01389620). *Numerische Mathematik*. **49** (2): 123–138. doi:10.1007/BF01389620 (https://doi.org/10.1007%2FBF01389620). ISSN 0945-3245 (https://search.worldcat.org/issn/0945-3245). S2CID 121771945 (https://api.semanticscholar.org/CorpusID:121771945).
21. Vrahatis, M.N.; Perdiou, A.E.; Kalantonis, V.S.; Perdios, E.A.; Papadakis, K.; Prosmiti, R.; Farantos, S.C. (July 2001). "Application of the Characteristic Bisection Method for locating and computing periodic orbits in molecular systems". *Computer Physics Communications*. **138** (1): 53–68. doi:10.1016/S0010-4655(01)00190-4 (https://doi.org/10.1016%2FS0010-4655%2801%2900190-4).
22. Vrahatis, Michael N. (December 1988). "Solving systems of nonlinear equations using the nonzero value of the topological degree". *ACM Transactions on Mathematical Software*. **14** (4): 312–329. doi:10.1145/50063.214384 (https://doi.org/10.1145%2F50063.214384).

- Burden, Richard L.; Faires, J. Douglas (1985), "2.1 The Bisection Algorithm", *Numerical Analysis* (https://archive.org/details/numericalanalys00burd) (3rd ed.), PWS Publishers, ISBN 0-87150-857-5

Further reading

- Corliss, George (1977), "Which root does the bisection algorithm find?", *SIAM Review*, **19** (2): 325–327, doi:10.1137/1019044 (https://doi.org/10.1137%2F1019044), ISSN 1095-7200 (https://search.worldcat.org/issn/1095-7200)
- Kaw, Autar; Kalu, Egwu (2008), *Numerical Methods with Applications* (https://web.archive.org/web/20090413123941/http://numericalmethods.eng.usf.edu/topics/textbook_index.html) (1st ed.), archived from the original (http://numericalmethods.eng.usf.edu/topics/textbook_index.html) on 2009-04-13

External links

- Weisstein, Eric W. "Bisection" (https://mathworld.wolfram.com/Bisection.html). *MathWorld*.
- Bisection Method (https://web.archive.org/web/20060901073129/http://numericalmethods.eng.usf.edu/topics/bisection_method.html) Notes, PPT, Mathcad, Maple, Matlab, Mathematica from Holistic Numerical Methods Institute (https://web.archive.org/web/20060906070428/http://numericalmethods.eng.usf.edu/)