False position method

From Wikipedia, the free encyclopedia

The **false position method** or **regula falsi method** is a term for problem-solving methods in arithmetic, algebra, and calculus. In simple terms, these methods begin by attempting to evaluate a problem using test ("false") values for the variables, and then adjust the values accordingly.

Two basic types of false position method can be distinguished, *simple false position* and *double false position*. *Simple false position* is aimed at solving problems involving direct proportion. Such problems can be written algebraically in the form: determine *x* such that

$$ax = b,$$

if *a* and *b* are known. *Double false position* is aimed at solving more difficult problems that can be written algebraically in the form: determine *x* such that

$$f(x) = b,$$

if it is known that

$$f(x_1) = b_1, \qquad f(x_2) = b_2.$$

Double false position is mathematically equivalent to linear interpolation; for an affine linear function,

$$f(x) = ax + c,$$

it provides the exact solution, while for a nonlinear function *f* it provides an approximation that can be successively improved by iteration.

Contents [hide]

1 Arithmetic and algebra
2 Numerical analysis
2.1 Finding the root of the secant
3 Analysis
4 Illinois algorithm
5 Example code
6 See also
7 References
8 Further reading
9 External links

Arithmetic and algebra [edit]

In problems involving arithmetic or algebra, the **false position method** or **regula falsi** is used to refer to basic trial and error methods of solving problems by substituting test values for the unknown quantities. This is sometimes also referred to as "guess and check". Versions of this method predate the advent of algebra and the use of equations.

For simple false position, the method of solving what we would now write as $ax = b$ begins by using a test input value x', and finding the corresponding output value b' by multiplication: $ax' = b'$. The correct answer is then found by proportional adjustment, $x = x' \cdot b \div b'$. This technique is found in cuneiform tablets from ancient Babylonian mathematics, and possibly in papyri from ancient Egyptian mathematics.[1]

Likewise, double false position arose in late antiquity as a purely arithmetical algorithm. It was used mostly to solve what are now called affine linear problems by using a pair of test inputs and the corresponding pair of outputs. This algorithm would be memorized and carried out by rote. In the ancient Chinese mathematical text called *The Nine Chapters on the Mathematical Art* (九章算術), dated from 200 BC to AD 100, most of Chapter 7 was devoted to the algorithm. There, the procedure was justified by concrete arithmetical arguments, then applied creatively to a wide variety of story problems, including one involving what we would call secant lines on a quadratic polynomial. A more typical example is this "joint purchase" problem:

> Now an item is purchased jointly; everyone contributes 8 [coins], the excess is 3; everyone

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
العربية
Català
Deutsch
Español
فارسی
Français
Bahasa Indonesia
Italiano
Nederlands
Norsk bokmål
Norsk nynorsk
Polski
Português
Română
Slovenščina
Suomi
文言
中文
Edit links

contributes 7, the deficit is 4. Tell: The number of people, the item price, what is each? Answer: 7 people, item price 53.[2]

Between the 9th and 10th centuries, the Egyptian Muslim mathematician Abu Kamil wrote a now-lost treatise on the use of double false position, known as the *Book of the Two Errors* (*Kitāb al-khaṭāʾayn*). The oldest surviving writing on double false position from the Middle East is that of Qusta ibn Luqa (10th century), a Christian Arab mathematician from Baalbek, Lebanon. He justified the technique by a formal, Euclidean-style geometric proof. Within the tradition of medieval Muslim mathematics, double false position was known as *hisāb al-khaṭāʾayn* ("reckoning by two errors"). It was used for centuries, especially in the Maghreb, to solve practical problems such as commercial and juridical questions (estate partitions according to rules of Quranic inheritance), as well as purely recreational problems. The algorithm was often memorized with the aid of mnemonics, such as a verse attributed to Ibn al-Yasamin and balance-scale diagrams explained by al-Hassar and Ibn al-Banna, all three being mathematicians of Moroccan origin.[3]

Leonardo of Pisa (Fibonacci) devoted Chapter 13 of his book *Liber Abaci* (AD 1202) to explaining and demonstrating the uses of double false position, terming the method *regulis elchatayn* after the *al-khaṭāʾayn* method that he had learned from Arab sources.[3]

Numerical analysis [edit]

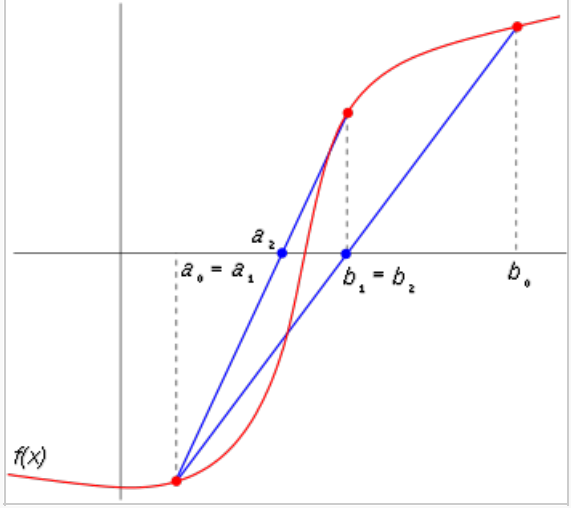
In numerical analysis, double false position became a root-finding algorithm that combines features from the bisection method and the secant method.

Like the bisection method, the false position method starts with two points a_0 and b_0 such that $f(a_0)$ and $f(b_0)$ are of opposite signs, which implies by the intermediate value theorem that the function f has a root in the interval $[a_0, b_0]$, assuming continuity of the function f. The method proceeds by producing a sequence of shrinking intervals $[a_k, b_k]$ that all contain a root of f.

At iteration number k, the number

$$c_k = b_k - f(b_k)\frac{(b_k - a_k)}{f(b_k) - f(a_k)}$$

is computed. As explained below, c_k is the root of the secant line through $(a_k, f(a_k))$ and $(b_k, f(b_k))$. If $f(a_k)$ and $f(c_k)$ have the same sign, then we set $a_{k+1} = c_k$ and $b_{k+1} = b_k$, otherwise we set $a_{k+1} = a_k$ and $b_{k+1} = c_k$. This process is repeated until the root is approximated sufficiently well.



The first two iterations of the false position method. The red curve shows the function f and the blue lines are the secants.

The above formula is also used in the secant method, but the secant method always retains the last two computed points, while the false position method retains two points which certainly bracket a root. On the other hand, the only difference between the false position method and the bisection method is that the latter uses $c_k = (a_k + b_k) / 2$.

Finding the root of the secant [edit]

Given a_k and b_k, we construct the line through the points $(a_k, f(a_k))$ and $(b_k, f(b_k))$, as demonstrated in the picture immediately above. Note that this line is a secant or chord of the graph of the function f. In point-slope form, it can be defined as

$$y - f(b_k) = \frac{f(b_k) - f(a_k)}{b_k - a_k}(x - b_k).$$

We now choose c_k to be the root of this line (substituting for x), and setting $y = 0$ and see that

$$f(b_k) + \frac{f(b_k) - f(a_k)}{b_k - a_k}(c_k - b_k) = 0.$$

Solving this equation gives the above equation for c_k.

$$f(b_k) + \frac{f(b_k) - f(a_k)}{b_k - a_k}(c_k - b_k) = 0$$
$$\frac{f(b_k) - f(a_k)}{b_k - a_k}(c_k - b_k) = -f(b_k)$$
$$(c_k - b_k) = -f(b_k)\frac{b_k - a_k}{f(b_k) - f(a_k)}$$
$$c_k = b_k - f(b_k)\frac{b_k - a_k}{f(b_k) - f(a_k)}$$

Analysis [edit]

If the initial end-points a_0 and b_0 are chosen such that $f(a_0)$ and $f(b_0)$ are of opposite signs, then at each step, one of the end-points will get closer to a root of f. If the second derivative of f is of constant sign (so there is no inflection point) in the interval, then one endpoint (the one where f also has the same sign) will remain fixed for all subsequent iterations while the converging endpoint becomes updated. As a result, unlike the bisection method, the width of the bracket does not tend to zero (unless the zero is at an inflection point around which *sign(f)=-sign(f'')*). As a consequence, the linear approximation to $f(x)$, which is used to pick the false position, does not improve in its quality.

One example of this phenomenon is the function

$$f(x) = 2x^3 - 4x^2 + 3x$$

on the initial bracket [−1,1]. The left end, −1, is never replaced (after the first three iterations, f'' is negative on the interval) and thus the width of the bracket never falls below 1. Hence, the right endpoint approaches 0 at a linear rate (the number of accurate digits grows linearly, with a rate of convergence of 2/3).

For discontinuous functions, this method can only be expected to find a point where the function changes sign (for example at *x=0* for *1/x* or the sign function). In addition to sign changes, it is also possible for the method to converge to a point where the limit of the function is zero, even if the function is undefined (or has another value) at that point (for example at *x=0* for the function given by $f(x)=abs(x)-x^2$ when $x\neq 0$ and by $f(0)=5$, starting with the interval [-0.5, 3.0]). It is mathematically possible with discontinuous functions for the method to fail to converge to a zero limit or sign change, but this is not a problem in practice since it would require an infinite sequence of coincidences for both endpoints to get stuck converging to discontinuities where the sign does not change (for example at $x=\pm 1$ in $f(x)=1/(x-1)^2+1/(x+1)^2$). The method of bisection avoids this hypothetical convergence problem.

Illinois algorithm [edit]

While it is a misunderstanding to think that the method of false position is a good method, it is equally a mistake to think that it is unsalvageable. The failure mode is easy to detect (the same end-point is retained twice in a row) and easily remedied by next picking a modified false position, such as

$$c_k = \frac{\frac{1}{2}f(b_k)a_k - f(a_k)b_k}{\frac{1}{2}f(b_k) - f(a_k)}$$

or

$$c_k = \frac{f(b_k)a_k - \frac{1}{2}f(a_k)b_k}{f(b_k) - \frac{1}{2}f(a_k)}$$

down-weighting one of the endpoint values to force the next c_k to occur on that side of the function. The factor of 2 above looks like a hack, but it guarantees superlinear convergence (asymptotically, the algorithm will perform two regular steps after any modified step, and has order of convergence 1.442). There are other ways to pick the rescaling which give even better superlinear convergence rates.[4]

The above adjustment to *regula falsi* is sometimes called the **Illinois algorithm**.[5][6] Ford (1995) summarizes and analyzes this and other similar superlinear variants of the method of false position.[4]

Example code [edit]

This example programme, written in the C programming language, has been written for clarity instead of efficiency. It was designed to solve the same problem as solved by the Newton's method and secant method code: to find the positive number x where $\cos(x) = x^3$. This problem is transformed into a root-finding problem of the form $f(x) = \cos(x) - x^3 = 0$.

```c
#include <stdio.h>
#include <math.h>

double f(double x)
{
   return cos(x) - x*x*x;
}
/* s,t: endpoints of an interval where we search
   e: half of upper bound for relative error
   m: maximal number of iterations */
double FalsiMethod(double s, double t, double e, int m)
{
   double r,fr;
   int n, side=0;
   /* starting values at endpoints of interval */
   double fs = f(s);
   double ft = f(t);

   for (n = 0; n < m; n++)
   {

       r = (fs*t - ft*s) / (fs - ft);
       if (fabs(t-s) < e*fabs(t+s)) break;
       fr = f(r);

       if (fr * ft > 0)
       {
         /* fr and ft have same sign, copy r to t */
         t = r; ft = fr;
         if (side==-1) fs /= 2;
         side = -1;
       }
       else if (fs * fr > 0)
       {
         /* fr and fs have same sign, copy r to s */
         s = r;  fs = fr;
         if (side==+1) ft /= 2;
         side = +1;
       }
       else
       {
         /* fr * f_ very small (looks like zero) */
         break;
       }
    }
    return r;
}

int main(void)
{
    printf("%0.15f\n", FalsiMethod(0, 1, 5E-15, 100));
    return 0;
}
```

After running this code, the final answer is approximately 0.865474033101614

See also [edit]

- Ridders' method, another root-finding method based on the false position method
- Brent's method
- Secant method

References [edit]

1. ^ Jean-Luc Chabert, ed., *A History of Algorithms: From the Pebble to the Microchip* (Berlin: Springer, 1999), pp. 86-91.
2. ^ Shen Kangshen, John N. Crossley and Anthony W.-C. Lun, 1999. *The Nine Chapters on the Mathematical Art: Companion and Commentary*. Oxford: Oxford University Press, p. 358.
3. ^ [a] [b] Schwartz, R. K. (2004). *Issues in the Origin and Development of Hisab al-Khata'ayn (Calculation by Double*

False Position). Eighth North African Meeting on the History of Arab Mathematics. Radès, Tunisia. Available online at: http://facstaff.uindy.edu/~oaks/Biblio/COMHISMA8paper.doc and http://www.ub.edu/islamsci/Schwartz.pdf

4. ^ [a] [b] Ford, J. A. (1995), *Improved Algorithms of Illinois-type for the Numerical Solution of Nonlinear Equations*, Technical Report, University of Essex Press, CSM-257
5. ^ Dahlquist, Germund; Björck, Åke (2003) [1974]. *Numerical Methods*. Dover. pp. 231–232. ISBN 978-0486428079.
6. ^ Dowell, M.; Jarratt, P. (1971). "A modified regula falsi method for computing the root of an equation". *BIT* **11** (2): 168–174. doi:10.1007/BF01934364.

Further reading [edit]

- Richard L. Burden, J. Douglas Faires (2000). *Numerical Analysis*, 7th ed. Brooks/Cole. ISBN 0-534-38216-9.
- L.E. Sigler (2002). *Fibonacci's Liber Abaci, Leonardo Pisano's Book of Calculation*. Springer-Verlag, New York. ISBN 0-387-40737-5.

External links [edit]

- The Regula Falsi Method by John H. Mathews

Categories: Root-finding algorithms