False Position Method

Subjects: Others
Contributor: HandWiki Li

In mathematics, the false position method or regula falsi is a very old method for solving an equation in one unknown, that, in modified form, is still in use. In simple terms, the method is the trial and error technique of using test ("false") values for the variable and then adjusting the test value according to the outcome. This is sometimes also referred to as "guess and check". Versions of the method predate the advent of algebra and the use of equations. As an example, consider problem 26 in the Rhind papyrus, which asks for a solution of (written in modern notation) the equation x + x/4 = 15. This is solved by false position, using a technique that predates formally written equations. First, guess that x = 4 to obtain, on the left, 4 + 4/4 = 5. This guess is a good choice since it produces an integer value. However, 4 is not the solution of the original equation, as it gives a value which is three times too small. To compensate, multiply x (currently set to 4) by 3 and substitute again to get 12 + 12/4 = 15, verifying that the solution is x = 12. Modern versions of the technique employ systematic ways of choosing new test values and are concerned with the questions of whether or not an approximation to a solution can be obtained, and if it can, how fast can the approximation be found.

Keywords: regula falsi ; false position ; trial and error

---

1. Two Historical Types

Two basic types of false position method can be distinguished historically, *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. The method 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* = *b*/ *b*′ *x*′.

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. By using a pair of test inputs and the corresponding pair of outputs, the result of this algorithm given by,[1]

$$x = \frac{b_1 x_2 - b_2 x_1}{b_1 - b_2},$$

would be memorized and carried out by rote. Indeed, the rule as given by Robert Recorde in his *Ground of Artes* (c. 1542) is:[1]

Gesse at this woorke as happe doth leade.
By chaunce to truthe you may procede.
And firste woorke by the question,
Although no truthe therein be don.
Suche falsehode is so good a grounde,
That truth by it will soone be founde.
From many bate to many mo,
From to fewe take to fewe also.

With to much ioyne to fewe againe,
To to fewe adde to manye plaine.
In crossewaies multiplye contrary kinde,
All truthe by falsehode for to fynde.

For an affine linear function,

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

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

2. History

The simple false position technique is found in cuneiform tablets from ancient Babylonian mathematics, and in papyri from ancient Egyptian mathematics.[2][3]

Double false position arose in late antiquity as a purely arithmetical algorithm. In the ancient Chinese mathematical text called *The Nine Chapters on the Mathematical Art* (九章算術),[4] 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 conic section. A more typical example is this "joint purchase" problem:

> Now an item is purchased jointly; everyone contributes 8 [coins], the excess is 3; everyone contributes 7, the deficit is 4. Tell: The number of people, the item price, what is each? Answer: 7 people, item price 53.[5]

Between the 9th and 10th centuries, the Egyptian 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-khaṭā'ayn*). The oldest surviving writing on double false position from the Middle East is that of Qusta ibn Luqa (10th century), an 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-khaṭā'ayn* ("reckoning by two errors"). It was used for centuries 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.[6]

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-khaṭā'ayn* method that he had learned from Arab sources.[6] In 1494, Pacioli used the term *el cataym* in his book *Summa de arithmetica*, probably taking the term from Fibonacci. Other European writers would follow Pacioli and sometimes provided a translation into Latin or the vernacular. For instance, Tartaglia translates the latinized version of Pacioli's term into the vernacular "false positions" in 1556.[7] Pacioli's term nearly disappeared in the 16th century European works and the technique went by various names such as "Rule of False", "Rule of Position" and "Rule of False Position". *Regula Falsi* appears as the latinized version of Rule of False as early as 1690.[1]

Several 16th century European authors felt the need to apologize for the name of the method in a science that seeks to find the truth. For instance, in 1568 Humphrey Baker says:[1]

> The Rule of falsehoode is so named not for that it teacheth anye deceyte or falsehoode, but that by fayned numbers taken at all aduentures, it teacheth to finde out the true number that is demaunded, and this of all the vulgar Rules which are in practise) is y^{e} most excellence.

3. Numerical Analysis

The method of false position provides an exact solution for linear functions, but more direct algebraic techniques have supplanted its use for these functions. However, in numerical analysis, double false position became a root-finding algorithm used in iterative numerical approximation techniques.

Many equations, including most of the more complicated ones, can be solved only by iterative numerical approximation. This consists of trial and error, in which various values of the unknown quantity are tried. That trial-and-error may be guided by calculating, at each step of the procedure, a new estimate for the solution. There are many ways to arrive at a calculated-estimate and *regula falsi* provides one of these.

Given an equation, move all of its terms to one side so that it has the form, $f(x) = 0$, where f is some function of the unknown variable x. A value c that satisfies this equation, that is, $f(c) = 0$, is called a *root* or *zero* of the function f and is a solution of the original equation. If f is a continuous function and there exist two points a_0 and b_0 such that $f(a_0)$ and $f(b_0)$ are of opposite signs, then, by the intermediate value theorem, the function f has a root in the interval (a_0, b_0).

There are many root-finding algorithms that can be used to obtain approximations to such a root. One of the most common is Newton's method, but it can fail to find a root under certain circumstances and it may be computationally costly since it requires a computation of the function's derivative. Other methods are needed and one general class of methods are the *two-point bracketing methods*. These methods proceed by producing a sequence of shrinking intervals $[a_k, b_k]$, at the kth step, such that (a_k, b_k) contains a root of f.

3.1. Two-Point Bracketing Methods

These methods start with two x-values, initially found by trial-and-error, at which $f(x)$ has opposite signs. Under the continuity assumption, a root of f is guaranteed to lie between these two values, that is to say, these values "bracket" the root. A point strictly between these two values is then selected and used to create a smaller interval that still brackets a root. If c is the point selected, then the smaller interval goes from c to the endpoint where $f(x)$ has the sign opposite that of $f(c)$. In the improbable case that $f(c) = 0$, a root has been found and the algorithm stops. Otherwise, the procedure is repeated as often as necessary to obtain an approximation to the root to any desired accuracy.

The point selected in any current interval can be thought of as an estimate of the solution. The different variations of this method involve different ways of calculating this solution estimate.

Preserving the bracketing and ensuring that the solution estimates lie in the interior of the bracketing intervals guarantees that the solution estimates will converge toward the solution, a guarantee not available with other root finding methods such as Newton's method or the secant method.

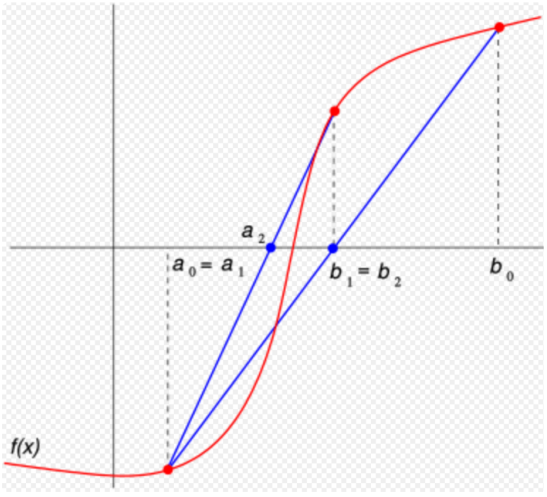
The simplest variation, called the bisection method, calculates the solution estimate as the midpoint of the bracketing interval. That is, if at step k, the current bracketing interval is $[a_k, b_k]$, then the new solution estimate c_k is obtained by,

$$c_k = \frac{a_k + b_k}{2}.$$

This ensures that c_k is between a_k and b_k, thereby guaranteeing convergence toward the solution.

Since the bracketing interval's length is halved at each step, the bisection method's error is, on average, halved with each iteration. The method gains roughly a decimal place of accuracy, for each 3 iterations.

3.2. The *Regula Falsi* (False Position) Method



The first two iterations of the false position method. The red curve shows the function f and the blue lines are the secants.
https://handwiki.org/wiki/index.php?curid=2076972

The convergence rate of the bisection method could possibly be improved by using a different solution estimate.

The *regula falsi* method calculates the new solution estimate as the x-intercept of the line segment joining the endpoints of the function on the current bracketing interval. Essentially, the root is being approximated by replacing the actual function by a line segment on the bracketing interval and then using the classical double false position formula on that line segment.[8]

More precisely, suppose that in the k-th iteration the bracketing interval is (a_k, b_k). Construct the line through the points (a_k, $f(a_k)$) and (b_k, $f(b_k)$), as illustrated. This line is a secant or chord of the graph of the function f. In point-slope form, its equation is given by

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

Now choose c_k to be the x-intercept of this line, that is, the value of x for which $y = 0$, and substitute these values to obtain

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

Solving this equation for c_k gives:

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

This last symmetrical form has a computational advantage:

As a solution is approached, a_k and b_k will be very close together, and nearly always of the same sign. Such a subtraction can lose significant digits. Because $f(b_k)$ and $f(a_k)$ are always of opposite sign the “subtraction” in the numerator of the improved formula is effectively an addition (as is the subtraction in the denominator too).

At iteration number k, the number c_k is calculated as above and then, if $f(a_k)$ and $f(c_k)$ have the same sign, set $a_{k+1} = c_k$ and $b_{k+1} = b_k$, otherwise set $a_{k+1} = a_k$ and $b_{k+1} = c_k$. This process is repeated until the root is approximated sufficiently well.

The above formula is also used in the secant method, but the secant method always retains the last two computed points, and so, while it is slightly faster, it does not preserve bracketing and may not converge.

The fact that *regula falsi* always converges, and has versions that do well at avoiding slowdowns, makes it a good choice when speed is needed. However, its rate of convergence can drop below that of the bisection method.

4. Analysis

Since the initial end-points a_0 and b_0 are chosen such that $f(a_0)$ and $f(b_0)$ are of opposite signs, 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 as rapidly as possible.

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 (it does not change at first and 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 $f(x) = \text{abs}(x) - x^2$ 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

$$f(x) = \frac{1}{(x-1)^2} + \frac{1}{(x+1)^2}.$$

The method of bisection avoids this hypothetical convergence problem.

5. Improvements in *Regula Falsi*

5.1. The LIlinois Algorithm

Though *regula falsi* always converges, usually considerably faster than bisection, there are situations that can slow its convergence—sometimes to a prohibitive degree. That problem isn't unique to *regula falsi*: Other than bisection, *all* of the numerical equation-solving methods can have a slow-convergence or no-convergence problem under some conditions. Sometimes, Newton's method and the secant method *diverge* instead of converging—and often do so under the conditions that slow *regula falsi's* convergence.

But, though *regula falsi* is one of the best methods, and—even in its original un-improved version—would often be the best choice (e.g. when Newton's isn't used because the derivative is prohibitively time-consuming to evaluate, or when Newton's and *Successive-Substitutions* have failed to converge). A number of improvements to *regula falsi* have been proposed, in order to avoid slowdowns under those relatively unusual unfavorable situations.

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 arbitrary, 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.[9]

The above adjustment to *regula falsi* is sometimes called the **Illinois algorithm**.[10][11] Ford (1995) summarizes and analyzes this and other similar superlinear variants of the method of false position.[9]

Put simply:

When the new y-value (that is, $f(c_k)$) has the same sign as the previous one ($f(c_{k-1})$), meaning that the end point of the previous step will be retained, the Illinois version halves the y-value of the retained end point in the next estimate computation.

5.2. Anderson − Björk Algorithm

Suppose that in the k-th iteration the bracketing interval is $[a_k, b_k]$ and that the functional value of the new calculated estimate c_k has the same sign as $f(b_k)$. In this case, the new bracketing interval $[a_{k+1}, b_{k+1}] = [a_k, c_k]$ and the left-hand endpoint has been retained. (So far, that's the same as ordinary Regula Falsi and the Illinois algorithm.)

But, whereas the Illinois algorithm would multiply $f(a_k)$ by 1/2, Anderson − Björk algorithm multiplies it by m, where m has one of the two following values:

$m = 1 - \frac{f(c_k)}{f(b_k)}$ if that value of m is positive,

otherwise, let $m = \frac{1}{2}$.

For simple roots, Anderson-Björk was the clear winner in Galdino's numerical tests.[12][13]

For multiple roots, no method was much faster than bisection. In fact, the only methods that were as fast as bisection were three new methods introduced by Galdino. But even they were only a little faster than bisection.

6. Practical Considerations

When solving one equation, or just a few, using a computer, the bisection method is an adequate choice. Although bisection isn't as fast as the other methods—when they're at their best and don't have a problem—bisection nevertheless is guaranteed to converge at a useful rate, roughly halving the error with each iteration – gaining roughly a decimal place of accuracy with every 3 iterations.

For manual calculation, by calculator, one tends to want to use faster methods, and they usually, but not always, converge faster than bisection. But a computer, even using bisection, will solve an equation, to the desired accuracy, so rapidly that there's no need to try to save time by using a less reliable method—and every method is less reliable than bisection.

An exception would be if the computer program had to solve equations very many times during its run. Then the time saved by the faster methods could be significant.

Then, a program could start with Newton's method, and, if Newton's isn't converging, switch to *regula falsi*, maybe in one of its improved versions, such as the Illinois or Anderson-Bjőrk versions. Or, if even that isn't converging as well as bisection would, switch to bisection, which always converges at a useful, if not spectacular, rate.

When the change in y has become very small, and x is also changing very little, then Newton's method most likely will not run into trouble, and will converge. So, under those favorable conditions, one could switch to Newton's method if one wanted the error to be very small and wanted very fast convergence.

7. Example Code

This example program, written in the C programming language, includes the **Illinois Algorithm.** To find the positive number x where $\cos(x) = x^3$, the equation is transformed into a root-finding form $f(x) = \cos(x) - x^3 = 0$.

```
#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

References

1. Smith, D. E. (1958) [1925], History of Mathematics, II, Dover, pp. 437−441, ISBN 978-0-486-20430-7
2. Jean-Luc Chabert, ed., A History of Algorithms: From the Pebble to the Microchip (Berlin: Springer, 1999), pp. 86-91.
3. Katz, Victor J. (1998), A History of Mathematics (2nd ed.), Addison Wesley Longman, p. 15, ISBN 978-0-321-01618-8
4. Joseph Needham (1 January 1959). Science and Civilisation in China: Volume 3, Mathematics and the Sciences of the Heavens and the Earth. Cambridge University Press. pp. 147–. ISBN 978-0-521-05801-8. https://books.google.com/books?id=jfQ9E0u4pLAC&pg=PA147#v=onepage&q&f=false.
5. 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.
6. 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 "Archived copy". Archived from the original on 2014-05-16. https://web.archive.org/web/20140516012137/http://www.ub.edu/islamsci/Schwartz.pdf. Retrieved 2012-06-08.
7. General Trattato, I, Venice, 1556, p. fol. 238, v, "Regola Helcataym (vocabulo Arabo) che in nostra lingua vuol dire delle false Positioni"

8. Conte, S. D. (1965), Elementary Numerical Analysis / an algorithmic approach, McGraw-Hill, p. 40
9. Ford, J. A. (1995), Improved Algorithms of Illinois-type for the Numerical Solution of Nonlinear Equations, Technical Report, University of Essex Press, CSM-257
10. Dahlquist, Germund; Björck, Åke (2003) [1974]. Numerical Methods. Dover. pp. 231–232. ISBN 978-0486428079. https://books.google.com/books?id=armfeHpJIwAC&pg=PA232.
11. 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. https://dx.doi.org/10.1007%2FBF01934364
12. Galdino, Sérgio (2011). "A family of regula falsi root-finding methods". Proceedings of 2011 World Congress on Engineering and Technology 1. http://cstm.cnki.net/stmt/TitleBrowse/KnowledgeNet/IEEE201110004133?db=STMI8515. Retrieved 9 September 2016.
13. Galdino, Sérgio (2011). A family of regula falsi root-finding methods. http://sergiogaldino.pbworks.com/w/file/fetch/66011429/0130-1943543. Retrieved 11 July 2017.
