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
Français
Magyar
日本語
Polski
Русский
中文
Edit links

Golden section search

From Wikipedia, the free encyclopedia

The **golden section search** is a technique for finding the extremum (minimum or maximum) of a strictly unimodal function by successively narrowing the range of values inside which the extremum is known to exist. The technique derives its name from the fact that the algorithm maintains the function values for triples of points whose distances form a golden ratio. The algorithm is the limit of Fibonacci search (also described below) for a large number of function evaluations. Fibonacci search and Golden section search were discovered by Kiefer (1953). (see also Avriel and Wilde (1966)).

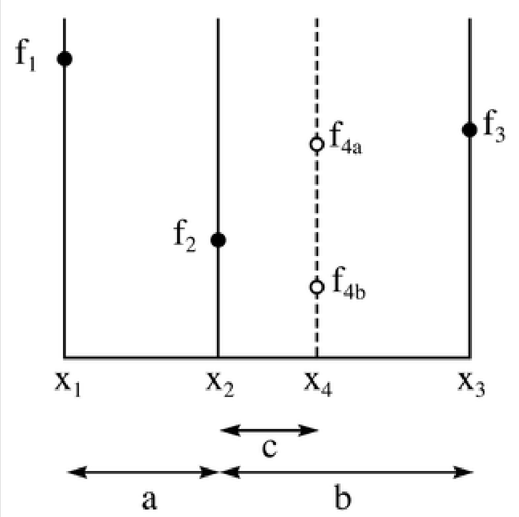


Diagram of a golden section search

Contents [hide]

Basic idea [edit]

The diagram above illustrates a single step in the technique for finding a minimum. The functional values of $f(x)$ are on the vertical axis, and the horizontal axis is the *x* parameter. The value of $f(x)$ has already been evaluated at the three points: x_1, x_2, and x_3. Since f_2 is smaller than either f_1 or f_3, it is clear that a minimum lies inside the interval from x_1 to x_3 (since *f* is unimodal).

The next step in the minimization process is to "probe" the function by evaluating it at a new value of *x*, namely x_4. It is most efficient to choose x_4 somewhere inside the largest interval, i.e. between x_2 and x_3. From the diagram, it is clear that if the function yields f_{4a} then a minimum lies between x_1 and x_4 and the new triplet of points will be x_1, x_2, and x_4. However if the function yields the value f_{4b} then a minimum lies between x_2 and x_3, and the new triplet of points will be x_2, x_4, and x_3. Thus, in either case, we can construct a new narrower search interval that is guaranteed to contain the function's minimum.

Probe point selection [edit]

From the diagram above, it is seen that the new search interval will be either between x_1 and x_4 with a length of *a+c* , or between x_2 and x_3 with a length of *b* . The golden section search requires that these intervals be equal. If they are not, a run of "bad luck" could lead to the wider interval being used many times, thus slowing down the rate of convergence. To ensure that *b* = *a*+*c*, the algorithm should choose $x_4 = x_1 + (x_3 - x_2)$.

However there still remains the question of where x_2 should be placed in relation to x_1 and x_3. The golden section search chooses the spacing between these points in such a way that these points have the same proportion of spacing as the subsequent triple x_1, x_2, x_4 or x_2, x_4, x_3. By maintaining the same proportion of spacing throughout the algorithm, we avoid a situation in which x_2 is very close to x_1 or x_3, and guarantee that the interval width shrinks by the same constant proportion in each step.

Mathematically, to ensure that the spacing after evaluating $f(x_4)$ is proportional to the spacing prior to that evaluation, if $f(x_4)$ is f_{4a} and our new triplet of points is x_1, x_2, and x_4 then we want:

$$\frac{c}{a} = \frac{a}{b}.$$

However, if $f(x_4)$ is f_{4b} and our new triplet of points is x_2, x_4, and x_3 then we want:

$$\frac{c}{(b-c)} = \frac{a}{b}.$$

Eliminating *c* from these two simultaneous equations yields:

$$\left(\frac{b}{a}\right)^2 = \frac{b}{a} + 1$$

or

$$\frac{b}{a} = \varphi$$

where φ is the golden ratio:

$$\varphi = \frac{1+\sqrt{5}}{2} = 1.618033988\ldots$$

The appearance of the golden ratio in the proportional spacing of the evaluation points is how this search algorithm gets its name.

Termination condition [edit]

In addition to a routine for reducing the size of the bracketing of the solution, a complete algorithm must have a termination condition. The one provided in the book *Numerical Recipes in C* is based on testing the gaps among x_1, x_2, x_3 and x_4, terminating when within the relative accuracy bounds:

$$|x_3 - x_1| < \tau(|x_2| + |x_4|)$$

where τ is a tolerance parameter of the algorithm and $|x|$ is the absolute value of x. The check is based on the bracket size relative to its central value, because that relative error in x is approximately proportional to the squared absolute error in $f(x)$ in typical cases. For that same reason, the Numerical Recipes text recommends that $\tau = \sqrt{\epsilon}$ where ϵ is the required absolute precision of $f(x)$.

Algorithm [edit]

Iterative algorithm [edit]

- Let [a, b] be interval of current bracket. f(a), f(b) would already have been computed earlier.
  $\varphi = (-1+\sqrt{5})/2$.
- Let c = b + φ (a - b), d = a + φ (b - a). If f(c), f(d) not available, compute them.
- If f(c) < f(d) (this is to find min, to find max, just reverse it) then move the data: (b, f(b)) ← (d, f(d)), (d, f(d)) ← (c, f(c)) and update c = b + φ (a - b) and f(c);
- otherwise, move the data: (a, f(a)) ← (c, f(c)), (c, f(c)) ← (d, f(d)) and update d = a + φ (b - a) and f(d).
- At the end of the iteration, [a, c, d, b] bracket the minimum point.

```
# python program for golden section search
gr=(math.sqrt(5)-1)/2
def gss(f,a,b,tol=1e-5):
    '''golden section search
to find the minimum of f on [a,b]
f: a strictly unimodal function on [a,b]

example:
>>> f=lambda x:(x-2)**2
>>> x=gss(f,1,5)
>>> x
2.000009644875678
'''
    c=b-gr*(b-a)
    d=a+gr*(b-a)
    while abs(c-d)>tol:
        fc=f(c);fd=f(d)
        if fc<fd:
            b=d
```

```
            d=c  #fd=fc;fc=f(c)
            c=b-gr*(b-a)
        else:
            a=c
            c=d  #fc=fd;fd=f(d)
            d=a+gr*(b-a)
    return (b+a)/2
```

Recursive algorithm [edit]

```
double phi = (1 + Math.sqrt(5)) / 2;
double resphi = 2 - phi;

// a and c are the current bounds; the minimum is between them.
// b is a center point
// f(x) is some mathematical function elsewhere defined
// a corresponds to x1; b corresponds to x2; c corresponds to x3
// x corresponds to x4
// tau is a tolerance parameter; see above

public double goldenSectionSearch(double a, double b, double c, double tau) {
    double x;
    if (c - b > b - a)
      x = b + resphi * (c - b);
    else
      x = b - resphi * (b - a);
    if (Math.abs(c - a) < tau * (Math.abs(b) + Math.abs(x)))
      return (c + a) / 2;
    assert(f(x) != f(b));
    if (f(x) < f(b)) {
      if (c - b > b - a) return goldenSectionSearch(b, x, c, tau);
      else return goldenSectionSearch(a, x, b, tau);
    }
    else {
      if (c - b > b - a) return goldenSectionSearch(a, b, x, tau);
      else return goldenSectionSearch(x, b, c, tau);
    }
  }
```

To realise the advantage of golden section search, the function $f(x)$ would be implemented with caching, so that in all invocations of `goldenSectionSearch(..)` above, except the first, $f(x_2)$ would have already been evaluated previously — the result of the calculation will be re-used, bypassing the (perhaps expensive) explicit evaluation of the function. Together with a slightly smaller number of recursions, this 50% saving in the number of calls to $f(x)$ is the main algorithmic advantage over Ternary search.

Fibonacci search [edit]

A very similar algorithm can also be used to find the extremum (minimum or maximum) of a sequence of values that has a single local minimum or local maximum. In order to approximate the probe positions of golden section search while probing only integer sequence indices, the variant of the algorithm for this case typically maintains a bracketing of the solution in which the length of the bracketed interval is a Fibonacci number. For this reason, the sequence variant of golden section search is often called *Fibonacci search*.

Fibonacci search was first devised by Kiefer (1953) as a minimax search for the maximum (minimum) of a unimodal function in an interval.

See also [edit]

- Fibonacci search technique
- Brent's method
- Binary search

References [edit]

- Kiefer, J. (1953), "Sequential minimax search for a maximum", *Proceedings of the American Mathematical Society* **4** (3): 502–506, doi:10.2307/2032161, JSTOR 2032161, MR 0055639

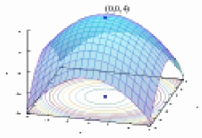
- Avriel, Mordecai; Wilde, Douglass J. (1966), "Optimality proof for the symmetric Fibonacci search technique", *Fibonacci Quarterly* **4**: 265–269, MR 0208812
- Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP (2007), "Section 10.2. Golden Section Search in One Dimension", *Numerical Recipes: The Art of Scientific Computing* (3rd ed.), New York: Cambridge University Press, ISBN 978-0-521-88068-8

v · t · e **Metallic means** [show]

v · t · e **Optimization: Algorithms, methods, and heuristics** [hide]

Unconstrained nonlinear: Methods calling … [hide]

… functions		**Golden section search** · Interpolation methods · Line search · Nelder–Mead method · Successive parabolic interpolation
… and gradients	**Convergence**	Trust region · Wolfe conditions
	Quasi–Newton	BFGS and L-BFGS · DFP · Symmetric rank-one (SR1)
	Other methods	Gauss–Newton · Gradient · Levenberg–Marquardt · Conjugate gradient · Truncated Newton
… and Hessians		Newton's method



Constrained nonlinear [show]

Convex optimization [show]

Combinatorial [show]

Metaheuristics [show]

Categories (Algorithms and methods · Heuristics) · **Software**

Categories: Golden ratio | Fibonacci numbers | Optimization algorithms and methods