Harvard Applied Mathematics 205
Further Optimization Methods

Danyun He

November 11, 2021

Two non-derivative-based optimization methods

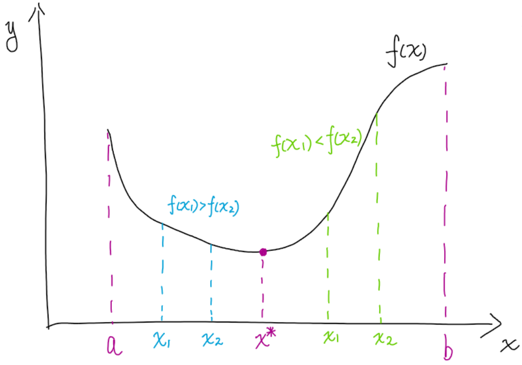
In AM205, we introduce the derivative-based optimization method: Newton's method. What if you don't know the derivative? Here, we introduce two non-derivative-based optimization methods:

1. Golden section search
2. Brent method

Golden section search: Unimodality[1]

A function $f : \mathbb{R} \to \mathbb{R}$ is unimodal on an interval $[a, b]$ if $\exists$ *unique* $x^* \in [a, b]$ such that $f(x^*)$ is the minimum of f on $[a, b]$, and for any $x_1, x_2 \in [a, b]$ with $x_1 < x_2$,

$$x_2 < x^* \implies f(x_1) > f(x_2), \text{ and } x_1 > x^* \implies f(x_1) < f(x_2).$$



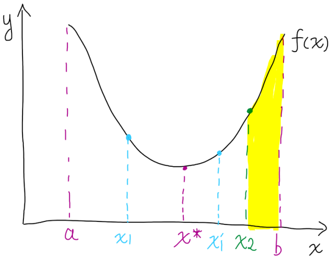
f(x) strictly decreasing for $x \le x^*$

f(x) strictly increasing for $x \ge x^*$

Golden section search: Finding minimum

Suppose $f(x)$ is unimodal on $[a, b]$, pick two points $x_1 < x_2$ in the interval $[a, b]$, and compare function values $f(x_1)$ and $f(x_2)$, then we can discard a sub-interval, either $(x_2, b]$ or $[a, x_1)$, and know that the minimum lies in the remaining interval.

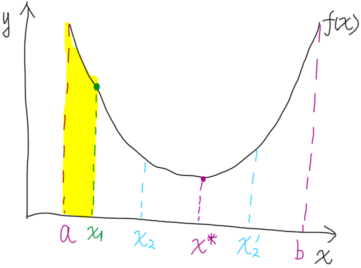
Golden section search: Finding minimum



$f(x_1) < f(x_2)$:
x_2 must be $x_2 > x^*$

otherwise, if $x_2 < x^*$,
then $x_1 < x_2 < x^*$,
lies on the strictly decreasing
side, and so $f(x_1) > f(x_2)$,
contradiction

Discard $(x_2, b]$



$f(x_1) > f(x_2)$
$\Rightarrow x_1 < x^*$

Discard $[a, x_1)$

Golden section search: Optimizing the method

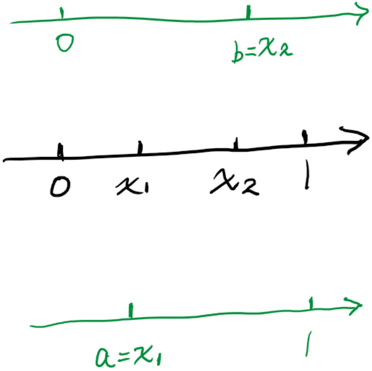
How to achieve maximum efficiency of the method?

1. To make consistent progress in reducing length of the interval containing the minimum, each pair of points should have the same relative positions within the new interval. So we can reduce a fixed fraction of length of an interval at each iteration.

2. Can we choose the two points x_1, x_2 such that: we can reuse the point in the remaining interval as one of the x_1 or x_2 points in the new interval, so we only need to find another point and compute its function value in the new interval?

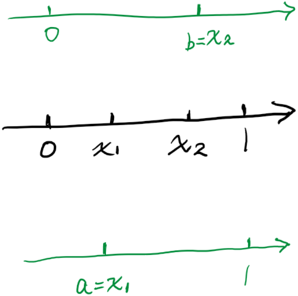
Golden section search: Optimizing the method

1. Decrease the length of the interval by a fixed fraction r at each iteration:



Golden section search: Optimizing the method

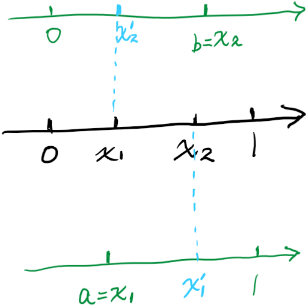
1. Decrease the length of the interval by a fixed fraction r at each iteration:



$$r = \frac{x_2 - 0}{1 - 0} = \frac{1 - x_1}{1 - 0} \implies x_2 = r, x_1 = 1 - r$$

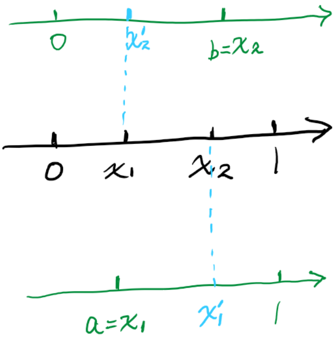
Golden section search: Optimizing the method

2. Reuse the two points x_1 and x_2 in the previous iteration, such that one becomes one of the end points of the new interval, and another one becomes either x_1 or x_2 of the new interval.



Why it must be this way? What equation can you list from here?

Golden section search: Optimizing the method



$\frac{x_1}{1} \neq \frac{x_1}{x_2}$

$\Rightarrow$ So x_1 must be x_2'

$\Rightarrow x_2 = \frac{x_1}{x_2} \Rightarrow x_2^2 = x_1$

$\because x_2 = r, \; x_1 = 1-r$

$\Rightarrow r^2 = 1-r \Rightarrow r^2+r-1=0$

$\Rightarrow r = \frac{-1+\sqrt{5}}{2}$ (neglect $\frac{-1-\sqrt{5}}{2}$ b/c $r=x_2>0$)

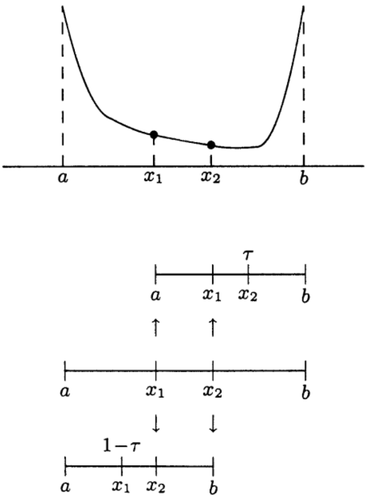
Golden ratio!

$\begin{cases} x_1 = a + (1-r)(b-a) \\ x_2 = a + r(b-a) \end{cases}$ where $r = \frac{-1+\sqrt{5}}{2}$

Golden section search: Algorithm[1]

Algorithm 6.1 Golden Section Search

$\tau = (\sqrt{5} - 1)/2$
$x_1 = a + (1 - \tau)(b - a)$
$f_1 = f(x_1)$
$x_2 = a + \tau(b - a)$
$f_2 = f(x_2)$
while $((b - a) > tol)$ **do**
  if $(f_1 > f_2)$ **then**
    $a = x_1$
    $x_1 = x_2$
    $f_1 = f_2$
    $x_2 = a + \tau(b - a)$
    $f_2 = f(x_2)$
  else
    $b = x_2$
    $x_2 = x_1$
    $f_2 = f_1$
    $x_1 = a + (1 - \tau)(b - a)$
    $f_1 = f(x_1)$
  end
end



Golden section search: Algorithm

Jupyter Notebook

Implement Golden Section Search algorithm

Input parameters:

1. function f;
2. initial interval $[a, b]$;
3. tolerance tol (when interval length $< tol$, program terminates)

For each iteration, print out: **iteration number,** x_1, $f(x_1)$, x_2 **and** $f(x_2)$.

Golden section search: Exercise 1

Use your implemented algorithm to solve:

min $f(x) = 0.5 - xe^{-x^2}$ in interval $[0, 2]$, with tolerance 0.001.

Before running the code, can we determine in advance the number of iterations needed for the algorithm to terminate?

Golden section search: Exercise 1

Use your implemented algorithm to solve:

min $f(x) = 0.5 - xe^{-x^2}$ in interval $[0, 2]$, with tolerance 0.001.

Before running the code, can we determine in advance the number of iterations needed for the algorithm to terminate?

Yes. Remember that the algorithm shrinks the interval by a fix fraction $r = \frac{\sqrt{5}-1}{2} \approx 0.618$ in each iteration, so

$$0.618^n(2-0) \leq 0.001,$$

$$n = 16 \text{ iterations}$$

Golden section search: Comments[1]

1. Great choice when function is not differentiable or difficult to differentiate.

2. Always converge: safe method.

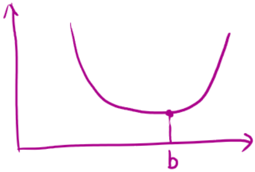
3. Slow convergent rate: linear. Rate of convergence ≈ 0.618.

Golden section search: Comments[1]

4. The method relies on the assumption that the function is unimodal on the starting interval. But often we cannot assume unimodality. Usually, in practice, one finds a suitable starting interval with trial and error: search for three points such that the two outer points have larger function value than the intermediate point. With the starting interval, although the method always converges, there is no guarantee to find the global minimum.

Golden section search: Comments[3]

5. Tolerance limited by rounding error:



For x near minimum b: Taylor expansion

$$f(x) \approx f(b) + \frac{1}{2} f''(b) (x-b)^2$$

$$\sqrt{\frac{2(f(x)-f(b))}{f''(b)}} > |x-b|$$

Golden section search: Comments[3]

5. Tolerance limited by rounding error (continue):

$$\sqrt{\frac{2\,\epsilon\,|f(b)|}{f''(b)}} > |x-b|$$

$$\sqrt{\epsilon\,|f(b)|}\,\sqrt{\frac{2}{f''(b)}} > |x-b|$$

$\therefore$ $|x-b|$ can only be accurate up to $\sqrt{\epsilon}$

that is, if double precision $\epsilon = 10^{-16}$.

we only get $\sqrt{\epsilon} = 10^{-8}$ precision in the interval tolerance with Golden section Search.

Brent's method

Golden-section Search(GSS) can handle function in the worst case. However, it is not fast. For functions that are smooth, nicely parabolic near the minimum, a parabola fitted curve take us to the minimum much faster. The method that uses parabola fitting is called Successive Parabolic Interpolation (SPI). It is faster than GSS but not as safe as it.

Brent's idea is to combines GSS and SPI. Use SPI when it is safe, and switch to GSS when SPI fails. Linear convergence is guaranteed for any function (as good as GSS) and on well-behaved functions convergence is superlinear with order at least 1.325.

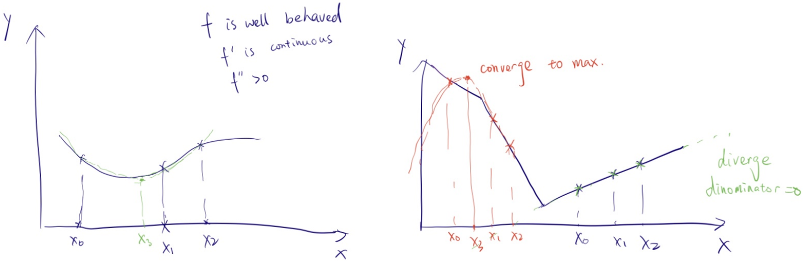
Successive Parabolic Interpolation (SPI)

1. Start with three initial guesses x_0, x_1, x_2
2. Draw the parabola that interpolates the three points, take the minimum point as x_3 where

$$x_3 = x_2 + \frac{1}{2}\frac{(x_1 - x_2)^2[f(x_2) + f(x_0)] - (x_0 - x_2)^2[f(x_1) - f(x_2)]}{(x_1 - x_2)[f(x_2) - f(x_0)] + (x_0 - x_2)[f(x_1) - f(x_2)]}$$

3. Repeat 2 with three lowest points among x_0, x_1, x_2, x_3 until reach terminal condition

SPI



With arbitrary starting points, it might diverge or converge to the maximum.

Brent's method

Brent's method keep track of six function points: a, b, u, v, w, x

a: left bound

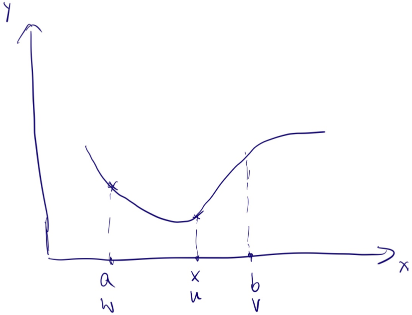
b: right bound

x: the least value f

w: second least value f

v: previous value of w

u: the most recent evaluated point



Brent's method: Algorithm

Given a unimodal $f(x)$ and bound $[a, b]$

1. Initialize $v = w = x = a + (\frac{3-\sqrt{5}}{2})(b - a)$
2. Find the next point u via SPI with x, w, v if it is well behaved, otherwise, use GSS.
3. Evaluate $f(u), f(x)$. Update u, v, w, x, a, b accordingly. Note when $f(u) <= f(x)$, check if $u >= x$, then $a = x$, else $b = x$. When $f(u) > f(x)$, check if $u < x$, then $a = u$, else $b = u$.
4. Repeat 2,3 until $b - a < Tol$, then return $\frac{a+b}{2}$

SPI: well behaved?

Recall that

$$x_3 = x_2 + \frac{1}{2}\frac{(x_1 - x_2)^2[f(x_2) + f(x_0)] - (x_0 - x_2)^2[f(x_1) - f(x_2)]}{(x_1 - x_2)[f(x_2) - f(x_0)] + (x_0 - x_2)[f(x_1) - f(x_2)]}$$

We find u in the form of

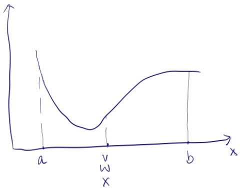
$$u = x + \frac{p}{q}$$

SPI is well behaved if

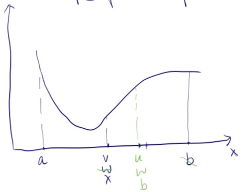
- $q \neq 0$
- $u \in [a, b]$
- The movement is smaller than the half of the last step $|\frac{p}{q}| < \frac{1}{2}|e|$, e is the previous $\frac{p}{q}$, to ensure it is converging
- $|e|$ is greater than some tolerance

Brent's method

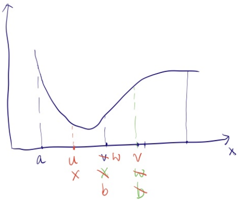
(1) Init
$v = w = x = a + \left(\frac{3-\sqrt{5}}{2}\right)(b-a)$



(2) v, w, x are the same points
$\Rightarrow$ SPI fails, compute u use GSS



(3) v, x are the same points
$\Rightarrow$ SPI fails, compute u use GSS



(4) unique x, v, w, SPI well behaved

Brent's method: Exercise 2

Use the library functions for Brent's method (MATLAB: fminbnd; Python: scipy.optimize.minimize_scalar) to solve the problem in exercise 1: min $f(x) = 0.5 - xe^{-x^2}$ in interval $[0, 2]$, with tolerance 0.001.

How many number of iterations they use? Which one is faster?

Take home exercise 1

min $f(x) = e^{-x} - cos(x)$ in interval $[0, 1]$, with tolerance 0.01.

1. Prove $f(x)$ is unimodal on interval $[0, 1]$
2. Prove there is a unique global minimizer in $(0, 1)$
3. Calculate by hand the number of iterations needed to reach the tolerance
4. Run your golden section search algorithm and report the results $x_1, f(x_1), x_2, f(x_2)$ for each iteration
5. Run Brent's method and report the number of iterations needed. Compare the result with golden section search

Take home exercise 2

$$\min \; f(x) = sin(\frac{1}{x}) + cos(\frac{1}{\sqrt(x)})$$

Use your favorite method, find out how many minima does $f(x)$ have in interval $[0.005, 0.2]$.

Report: submit pdf version of your jupyter notebook, including in-class exercises 1&2, and take-home exercises 1&2. You are welcome to submit as a group. Remember to list the names of group members!

Reference

[1] Heath, M. T. (2002). Scientific computing: An introductory survey. Boston: McGraw-Hill.

[2] Xu, Huifu. MATH3016: OPTIMIZATION notes. http://web.tecnico.ulisboa.pt/mcasquilho/compute/com/,Fibonacci/pdfHXu_ch1.pdf.

[3] William H. Press ... [and others]. Numerical Recipes in C : the Art of Scientific Computing. Cambridge [Cambridgeshire] ; New York :Cambridge University Press, 1992.

[4] Brent, R.P. 1973, Algorithms for Minimization without Derivatives (Englewood Cliffs, NJ: Prentice- Hall); reprinted 2002 (New York: Dover), Chapter 5.[1]

Brent-Dekker's method: a root finding technique

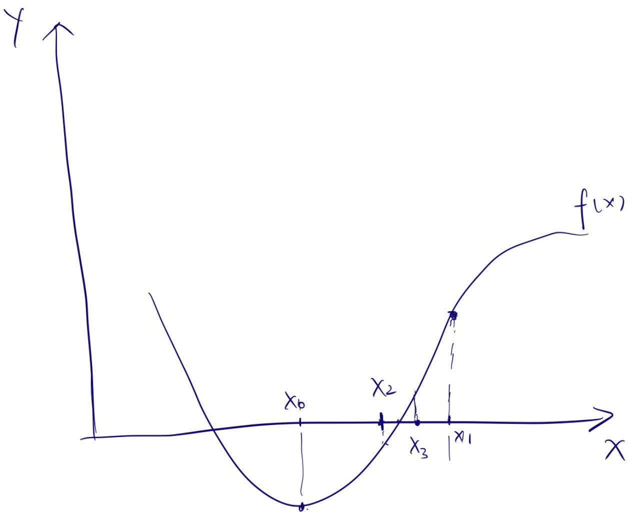
Brent's minimization method is analogous to Brent-Dekker method, which is a root finding method combining the bisection method and secant method. It's as fast as secant method and guarantee convergence as bisection method.
Root finding problem: find $x \in R$ such that $f(x) = 0$.

Bisection method

1. Initialize with two guesses x_1, x_2 such that $f(x_1)f(x_2) < 0$.
2. Calculate midpoint of x_1 and x_2, $m = \frac{x_1+x_2}{2}$. Check if $f(m) < Tol$, if not, check if $f(x_1)f(m) < 0$, then $newx_2 = m$, else $newx_1 = m$.
3. Repeat 2 until find some m that $|f(m)| < Tol$.

According to intermediate value theorem, there must exist a root in $[a, b]$. Bisection method is guaranteed to find the solution.
Limitation: slow

Bisection method

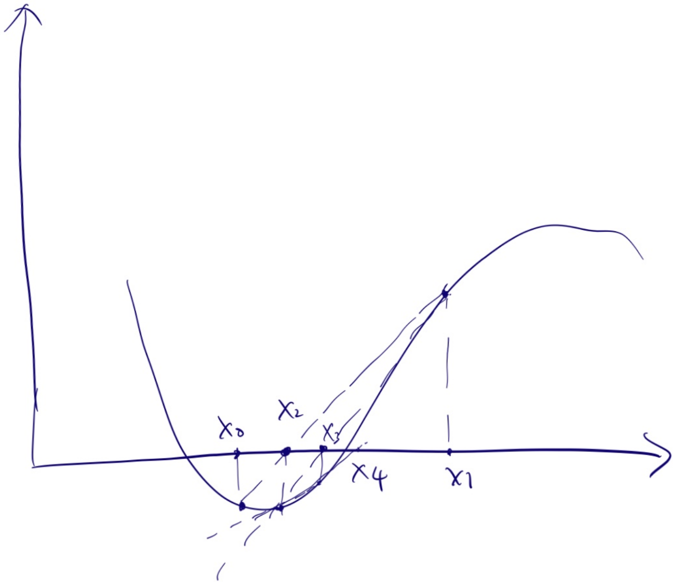


Secant method

1. Initialize with two guesses x_1, x_2
2. Draw a secant using x_1, x_2, take the intersection point with x-axis as x_3
3. Repeat 2 until find x_n such that $|f(x_n)| < Tol$

Secant method is fast, it's doing linear interpolation of the curve, approaching closer and closer to the solution. However, it might diverge ($f(x_i) = f(x_{i+1})$) or divide by zero ($x_i = x_{i+1}$).

Secant method

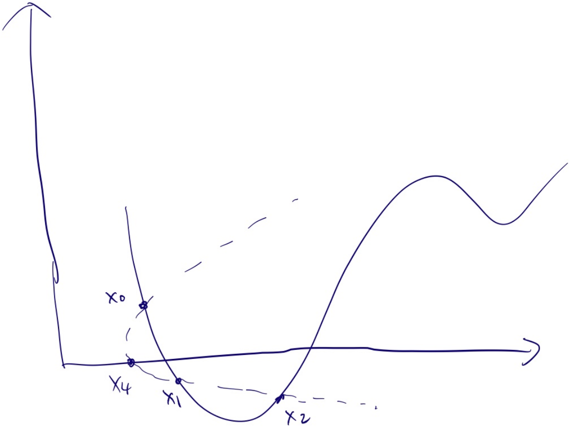


Inverse quadratic interpolation (IQI) method

1. Initialize with three guesses x_1, x_2, x_3
2. Draw sideways quadratic using x_1, x_2, x_3, take the intersection point with x-axis as x_4
3. Repeat 2 until find x_n such that $|f(x_n) < Tol$

IQI method is doing quadratic interpolation of the curve, theoretically, it fits the curve better. It's fast when starting with points close the root. However, it fails when any two values of $f(x_i), f(x_{i+1}), f(x_{i+2})$ match

IQI method



Brent-Dekker method

The idea of Brent-Dekker method is to make use of the bisection method, secant method and IQI method. It maintains the fast speed as secant method while the convergence is guaranteed.

1. Initialize with three guesses a, b, c
2. Compute s using IQI with a, b, c. If IQI doesn't work, compute s with secant method using b, c. If secant method doesn't work, use bisection method. Update a, b, c
3. Repeat 2 until $f(s) < Tol$

python: scipy.optimize.brentq
matlab: fzerotx

Exercise

Make your own function. Run bisection method, secant method and Brent-Dekker method. Compare: how many steps they use? Do they converge?
Example functions: $f(x) = x^4 + x^2 - 1$