AI505

Optimization

Bracketing

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

Solutions and Recognizing Them for Smooth Functions

Definition

A point $\boldsymbol{x}^*$ is a global minimizer if $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x}$.

Definition

A point $\boldsymbol{x}^*$ is a local minimizer if there is a neighborhood N of $\boldsymbol{x}^*$ such that $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in N$.

Solutions and Recognizing Them for Smooth Functions

Theorem (Taylor's Theorem)

Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ *is continuously differentiable and that* $\boldsymbol{p} \in \mathbb{R}^n$*. Then we have that*

$$f(\boldsymbol{x} + \boldsymbol{p}) = f(\boldsymbol{x}) + \nabla f(\boldsymbol{x} + t\boldsymbol{p})^T \boldsymbol{p},$$

for some $t \in (0, 1)$*. Moreover, if* f *is twice continuously differentiable, we have that*

$$\nabla f(\boldsymbol{x} + \boldsymbol{p}) = \nabla f(\boldsymbol{x}) + \int_0^1 \nabla^2 f(\boldsymbol{x} + t\boldsymbol{p})\boldsymbol{p} dt,$$

and that

$$f(\boldsymbol{x} + \boldsymbol{p}) = f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T \boldsymbol{p} + \frac{1}{2}\boldsymbol{p}^T \nabla^2 f(\boldsymbol{x} + t\boldsymbol{p})\boldsymbol{p},$$

for some $t \in (0, 1)$*.*

Solutions and Recognizing Them for Smooth Functions

Theorem (First-Order Necessary Conditions)

If $\boldsymbol{x}^*$ is a local minimizer and f is continuously differentiable in an open neighborhood of $\boldsymbol{x}^*$, then $\nabla f(\boldsymbol{x}^*) = 0$.

Definition

We call $\boldsymbol{x}^*$ a stationary point if $\nabla f(\boldsymbol{x}^*) = 0$.

Theorem (Second-Order Necessary Conditions)

If $\boldsymbol{x}^*$ is a local minimizer of f and $\nabla^2 f$ exists and is continuous in an open neighborhood of $\boldsymbol{x}^*$, then $\nabla f(\boldsymbol{x}^*) = 0$ and $\nabla^2 f(\boldsymbol{x}^*)$ is positive semidefinite.

Theorem (Second-Order Sufficient Conditions)

Suppose that $\nabla^2 f$ is continuous in an open neighborhood of $\boldsymbol{x}^*$ and that $\nabla f(\boldsymbol{x}^*) = 0$ and $\nabla^2 f(\boldsymbol{x}^*)$ is positive definite. Then $\boldsymbol{x}^*$ is a strict local minimizer of f.

Theorem

When f is convex, any local minimizer $\boldsymbol{x}^*$ is a global minimizer of f. If in addition f is differentiable, then any stationary point $\boldsymbol{x}^*$ is a global minimizer of f.

Algorithms for Unconstrained Optimization of Smooht Functions

Two main strategies for finding local minima of smooth functions are:

- Line search methods (gradient descent and its variants)
- Trust region methods

A component of both strategies is the ability to find a local minimum of a one-dimensional function, which is the topic of this lecture.

Bracketing: A derivative-free method to identify an interval containing a local minimum and then successively shrinking that interval

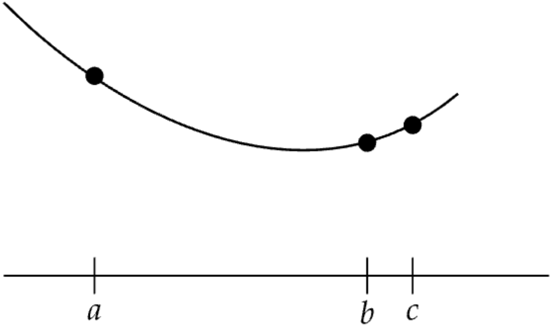
Unimodality

The algorithms in this class assume **unimodality**:

There exists a unique optimizer $\boldsymbol{x}^*$ such that f is monotonically decreasing for $\boldsymbol{x} \leq \boldsymbol{x}^*$ and monotonically increasing for $\boldsymbol{x} \geq \boldsymbol{x}^*$

Finding an Initial Bracket

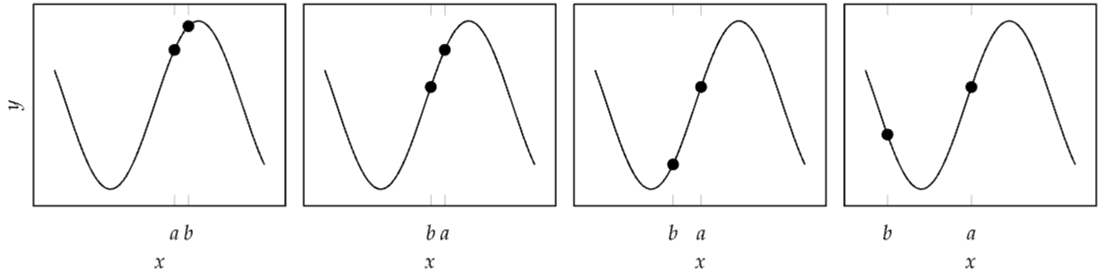
Given a unimodal function, the global minimum is guaranteed to be inside the interval $[a, c]$ if we can find three points $a < b < c$ such that $f(a) > f(b) < f(c)$




```
function bracket_minimum(f, x=0; s=1e-2, k=2.0)
    a, ya = x, f(x)
    b, yb = a + s, f(a + s)
    if yb > ya
        a, b = b, a
        ya, yb = yb, ya
        s = -s
    end
    while true
        c, yc = b + s, f(b + s)
        if yc > yb
            return a < c ? (a, c) : (c, a)
        end
        a, ya, b, yb = b, yb, c, yc
        s *= k
    end
end
```

Finding an Initial Bracket

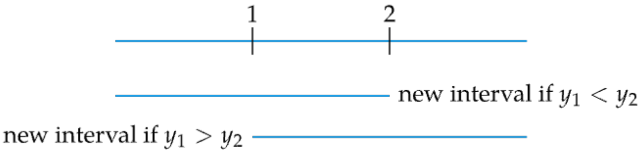
Example of `bracket_minimum` on a function



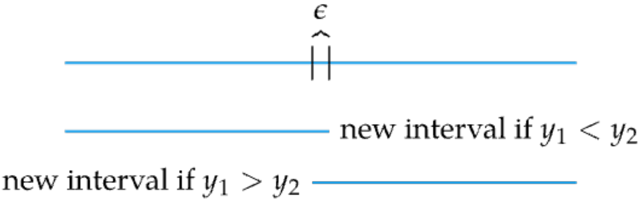
reverses direction between the first and second iteration and expands until a minimum is bracketed in the fourth iteration.

For unimodal functions, when function evaluations are limited, what is the maximal shrinckage we can achieve?

When restricted to only 2 function evaluations (queries) the most we can guarantee to shrink our interval is by just under a factor of 2.

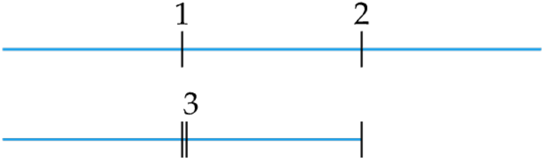


yeilds a factor of 3.



for $\epsilon \to 0$ yields a factor of just less than 2

When restricted to only 3 function evaluations (queries) the most we can guarantee to shrink our interval is by a factor of 3.

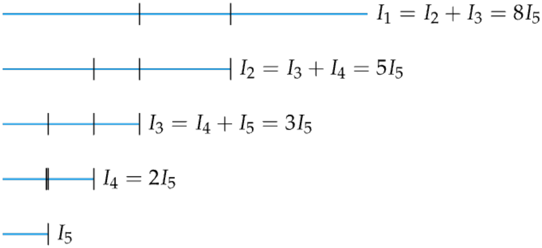


Fibonacci Search

When restricted to n functions evaluations following the previous strategy, we are guaranteed to shrink our interval by a factor of F_{n+1}.

Fibonacci numbers: sum of previous two, $1, 1, 2, 3, 5, 8, 13, \ldots$

$$F_n = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1, 2 \\ F_{n-1} + F_{n-2} & \text{otherwise} \end{cases}$$



The length of every interval constructed can be expressed in terms of the final interval times a Fibonacci number.

- final, smallest interval has length I_n,
- second smallest interval has length $I_{n-1} = F_3 I_n$
- third smallest interval has length $I_{n-2} = F_4 I_n$,

and so forth.

Fibonacci Search Algorithm

For a unimodal function f in the interval $[a, b]$, we want to shrink the interval within n iterations. (At each iteration we want to shrink by a factor ϕ).

$$b_{k+1} - a_{k+1} = \frac{F_{n-k+1}}{F_{n-k+2}}(b_k - a_k)$$

Therefore:

$$\begin{aligned} b_n - a_n &= \frac{F_2}{F_3}(b_{n-1} - a_{n-1}) \\ &= \frac{F_2}{F_3}\frac{F_3}{F_4}\dots\frac{F_n}{F_{n+1}}(b_1 - a_1) \\ &= \frac{1}{F_{n+1}}(b_1 - a_1) \end{aligned}$$

Closed-form expression (Binet's formula):

$$F_n = \frac{\phi^n - (1-\phi)^n}{\sqrt{5}},$$

$\phi = (1+\sqrt{5})/2 \approx 1.61803$ is the golden ratio.

$$\frac{F_{n+1}}{F_n} = \phi\frac{1 - s^{n+1}}{1 - s^n}, \quad s = (1-\sqrt{5})(1+\sqrt{5}) \approx -0.38$$

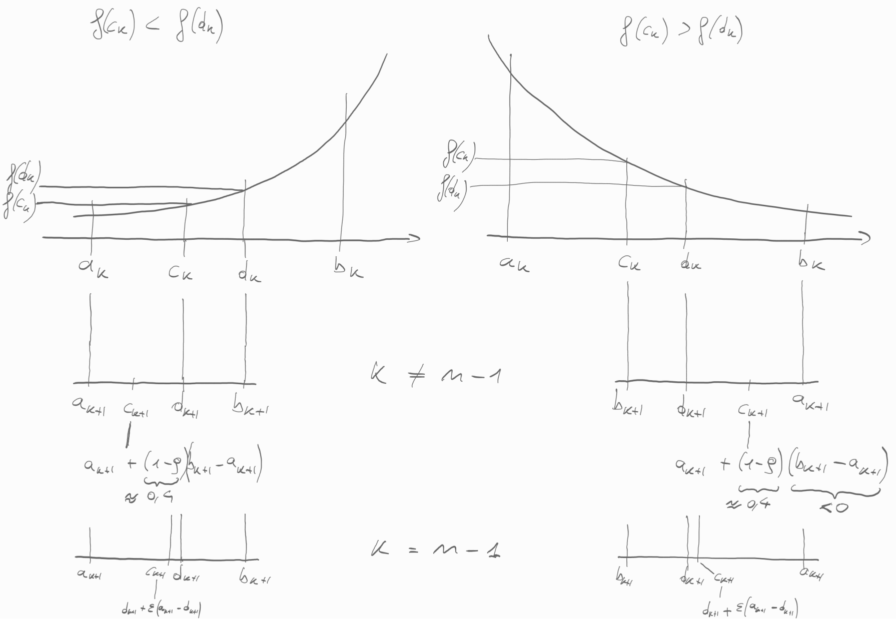
Suppose we have a unimodal function f in the interval $[a, b]$ and a tolerance $\epsilon = 0.01$. Let $k = 1$.

$$\frac{F_n}{F_{n+1}} = \rho_n = \frac{1 - s^n}{\phi(1 - s^{n+1})} \approx 0.6$$

1. $d_k = a_k + \frac{F_{n-k+1}}{F_{n-k+2}}(b_k - a_k)$
2. if $k \neq n - 1$:

   $$c_k = a_k + \left(1 - \frac{F_{n-k+1}}{F_{n-k+2}}\right)(b_k - a_k)$$

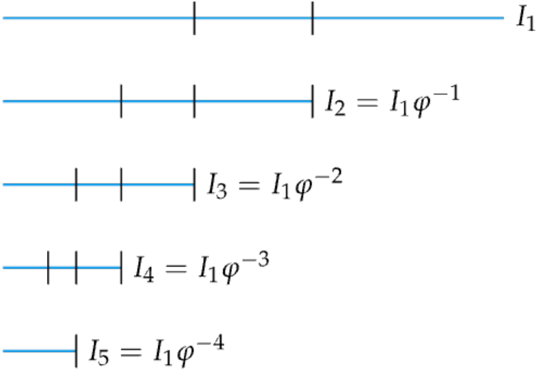
   otherwise: $c_k = d_k + \epsilon(a_k - d_k)$
3. if $f(c_k) < f(d_k)$: $\quad b_{k+1} = d_k, \quad d_{k+1} = c_k, \quad a_{k+1} = a_k$
   otherwise: $a_{k+1} = b_k, \quad b_{k+1} = c_k, \quad d_{k+1} = d_k$
4. $k = k + 1$, if $k = n$ go to step 5, else go to step 2
5. return (a_k, b_k) if $(a_k < b_k)$ else (b_k, a_k)



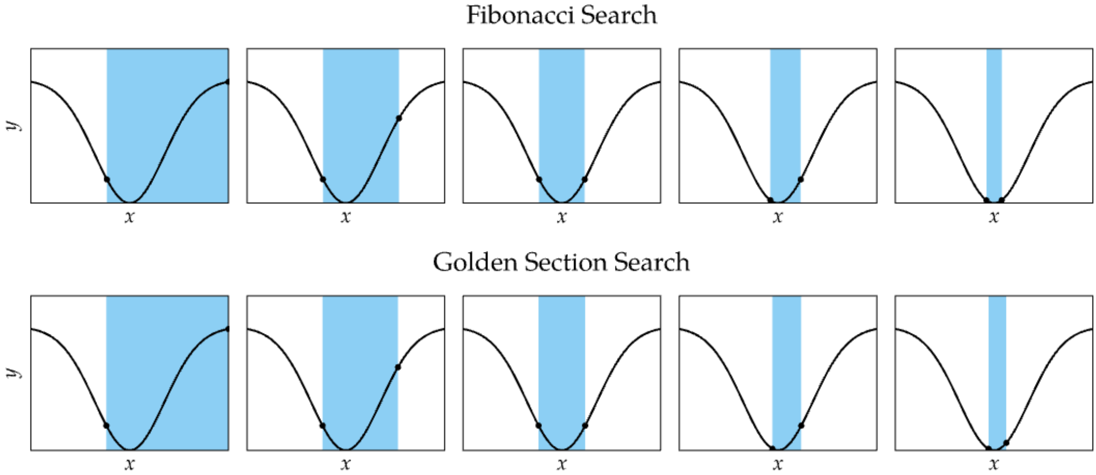
$f(c_k) < f(d_k)$
$f(d_k)$
$f(c_k)$
a_k c_k d_k b_k
$f(c_k) > f(d_k)$
$f(c_k)$
$f(d_k)$
a_k c_k d_k b_k
a_{k+1} c_{k+1} d_{k+1} b_{k+1}
$k \neq n-1$
b_{k+1} d_{k+1} c_{k+1} a_{k+1}
$a_{k+1} + \underbrace{(1-\rho)}_{\approx 0,4}(b_{k+1} - a_{k+1})$
$a_{k+1} + \underbrace{(1-\rho)}_{\approx 0,4}\underbrace{(b_{k+1} - a_{k+1})}_{<0}$
$k = n-1$
a_{k+1} c_{k+1} d_{k+1} b_{k+1}
$d_{k+1} + \varepsilon(a_{k+1} - d_{k+1})$
b_{k+1} d_{k+1} c_{k+1} a_{k+1}
$d_{k+1} + \varepsilon(a_{k+1} - d_{k+1})$

Golden Section Search

$$\lim_{n\to\infty}\frac{F_{n+1}}{F_n} = \lim_{n\to\infty}\frac{1}{\rho_n} = \lim_{n\to\infty}\phi\frac{1-s^{n+1}}{1-s^n} = \phi \approx 1.61803 \qquad \frac{1}{\phi}\approx 0.618$$

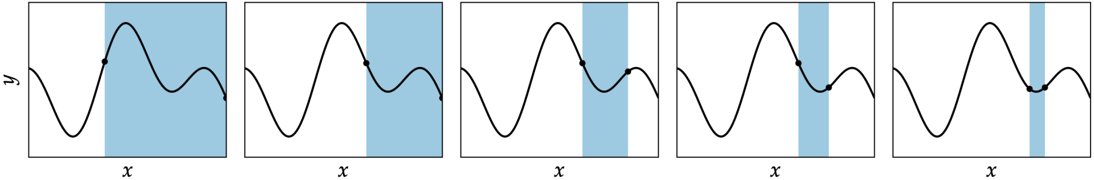


Comparison

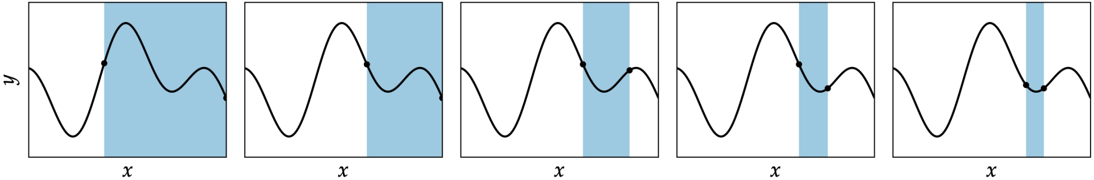


Comparison

Fibonacci Search

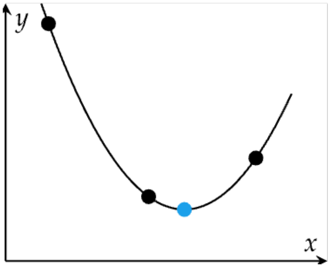


Golden Section Search



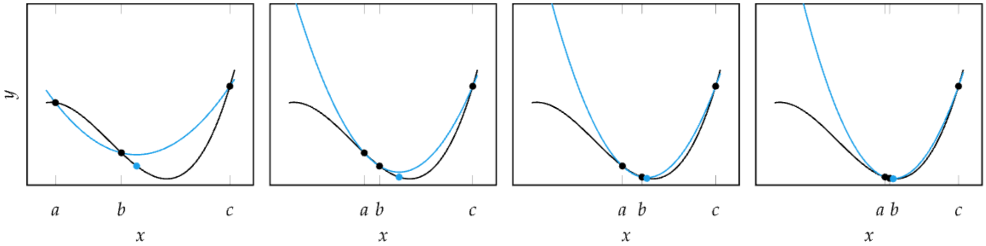
Quadratic Fit Search

- Leverages ability to analytically minimize quadratic functions
- Iteratively fits quadratic function to three bracketing points



Quadratic Fit Search

- If a function is locally nearly quadratic, the minimum can be found after several steps



Using Linear Algebra

- We assume that the variable y is related to $\boldsymbol{x} \in \mathbb{R}^n$ quadratically, so for some constants b_0, b_1, b_2:

$$y = b_0 + b_1 x + b_2 x^2$$

- Given the set of m points $(y_1, x_1,), \ldots, (y_3, x_3)$ in the ideal case, we have that $y_i = b_0 + b_1 x_i + b_2 x_i^2$, for all $i = 1, 2, 3$. In matrix form:

$$\begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

  This can be written as $A\boldsymbol{z} = \boldsymbol{y}$ to emphasize that $\boldsymbol{z}$ are our unknowns and A and $\boldsymbol{y}$ are given.

In Python

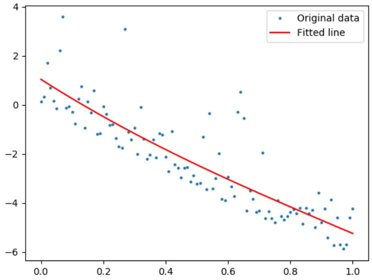
In polynomial regression, the $m \times (n+1)$ matrix A is called a **Vandermonde matrix** (a matrix with entries $a_{ij} = x_i^{n+1-j}$, $j = 1..n+1$).
NumPy's `np.vander()` is a convenient tool for quickly constructing a Vandermonde matrix, given the values x_i, $i = 1..m$, and the number of desired columns $(n+1)$.

```
>>> print(np.vander([2, 3, 5], 2))
[[2 1]                                  # [[2**1, 2**0]
 [3 1]                                  #  [3**1, 3**0]
 [5 1]]                                 #  [5**1, 5**0]]

>>> print(np.vander([2, 3, 5, 4], 3))
[[ 4  2  1]                             # [[2**2, 2**1, 2**0]
 [ 9  3  1]                             #  [3**2, 3**1, 3**0]
 [25  5  1]                             #  [5**2, 5**1, 5**0]
 [16  4  1]]                            #  [4**2, 4**1, 4**0]
```

In Python

```
A = np.vander(x,4)

coeff = np.linalg.solve(A,y) ## Error!! Why?

B = A.T @ A
z = np.linalg.inv(B) @ A.T @ y

coeff = np.linalg.lstsq(A, y)[0]
np.allclose(z,coeff)

f=np.poly1d(coeff)
plt.plot(x, y, 'o', label='Original data', ↪
    ↪markersize=2)
plt.plot(x, f(x), 'r', label='Fitted line')
plt.legend()
plt.show()
```

ex2.py



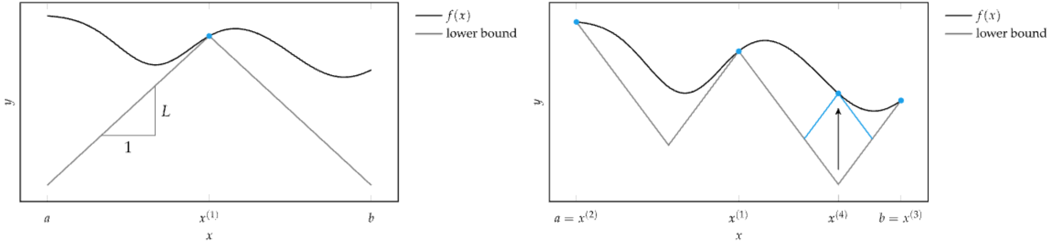
Shubert-Piyavskii Method

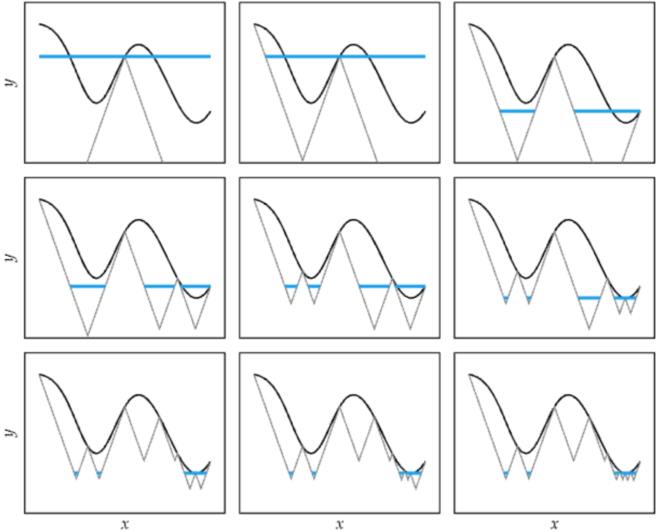
- The Shubert-Piyavskii method is guaranteed to find the global minimum of any bounded function

- but requires that the function be Lipschitz continuous

- A function is **Lipschitz continuous** if there is an upper bound on the magnitude of its derivative. A function f is Lipschitz continuous on $[a, b]$ if there exists an $\ell > 0$ such that:

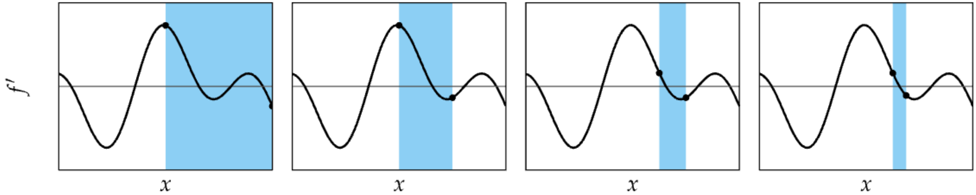
$$|f(x) - f(y)| \leq \ell|x - y|, \quad \forall x, y \in [a, b]$$




y
y
y
x
x
x

Bisection Method

- **Intermediate value theorem**: If f is continuous on $[a, b]$, and there is some $y \in [f(a), f(b)]$, then there exists at least one $x \in [a, b]$, such that $f(x) = y$.
- Used in root-finding methods
- When applied to $f'(x)$, can be used to find minimum of f
- if $\text{sign}(f'(a)) \neq \text{sign}(f'(b))$, or equivalently, $f'(a)f'(b) \leq 0$ then $[a, b]$ is guaranteed to contain a zero.



Bisection method

- Cut the bracketed region $[a, b]$ in half with every iteration
- Evaluate the midpoint $(a + b)/2$
- form a new bracket from the midpoint and whichever side that continues to bracket a zero.
- Terminate after a fixed number of iterations.
- Guaranteed to converge within ϵ of x^* within $\lg_2(|b - a|/\epsilon)$

Summary

- Many optimization methods shrink a bracketing interval, including Fibonacci search, golden section search, and quadratic fit search
- The Shubert-Piyavskii method outputs a set of bracketed intervals containing the global minima, given the Lipschitz constant
- Root-finding methods like the bisection method can be used to find where the derivative of a function is zero