Computational Macroeconomics

Lecture 8: Notes on Numerical Methods

November 2025

Kanato Nakakuni

Motivation

- This lecture: overview of numerical methods and algorithms commonly used to solve economic models (e.g., function approximation, root finding, optimization).
- You can find built-in functions that perform these methods (e.g., optimize in Optim.jl)
- Why should we understand these methods/algorithms?
  - Every method comes with pros and cons
  - Need to choose an appropriate method tailored to your problem
  - Helps you diagnose why a method might fail when it does not perform well
- For this purpose, we will briefly cover the basic ideas of some important methods.

Reference

- This lecture contents are mainly based on Chapter 10 “Computational tools” in *Macroeconomics* by Marina Azzimonti, Per Krusell, Alisdair McKay, and Toshihiko Mukoyama (link)

Contents

Function approximation

Function approximation

- Often the function of interest has no analytical or parametric form
- We thus discretize state space and represent function on a finite grid (memory is finite)
- But what if we want to:
  - Evaluate the function at a point **not** on the grid?
    $\Rightarrow$ Interpolation
  - Approximate the function using a parametric form?
    $\Rightarrow$Parametric approximation: ie., $\Sigma_{i=1}^{n} a_i \phi_i(x)$, where a_i and ϕ_i are weights and *base function*, e.g., (Chebychev) polynomials (see Projection method in lecture notes 2 as an example of application)

Interpolation

Motivating Example

- Consider a discrete choice of labor supply n, consumption c, and endogenous human capital h in a recursive formulation:

$$V(h) = \max_{c,n \in \{0,0.5,1\}} u(c,n) + \beta V(h')$$

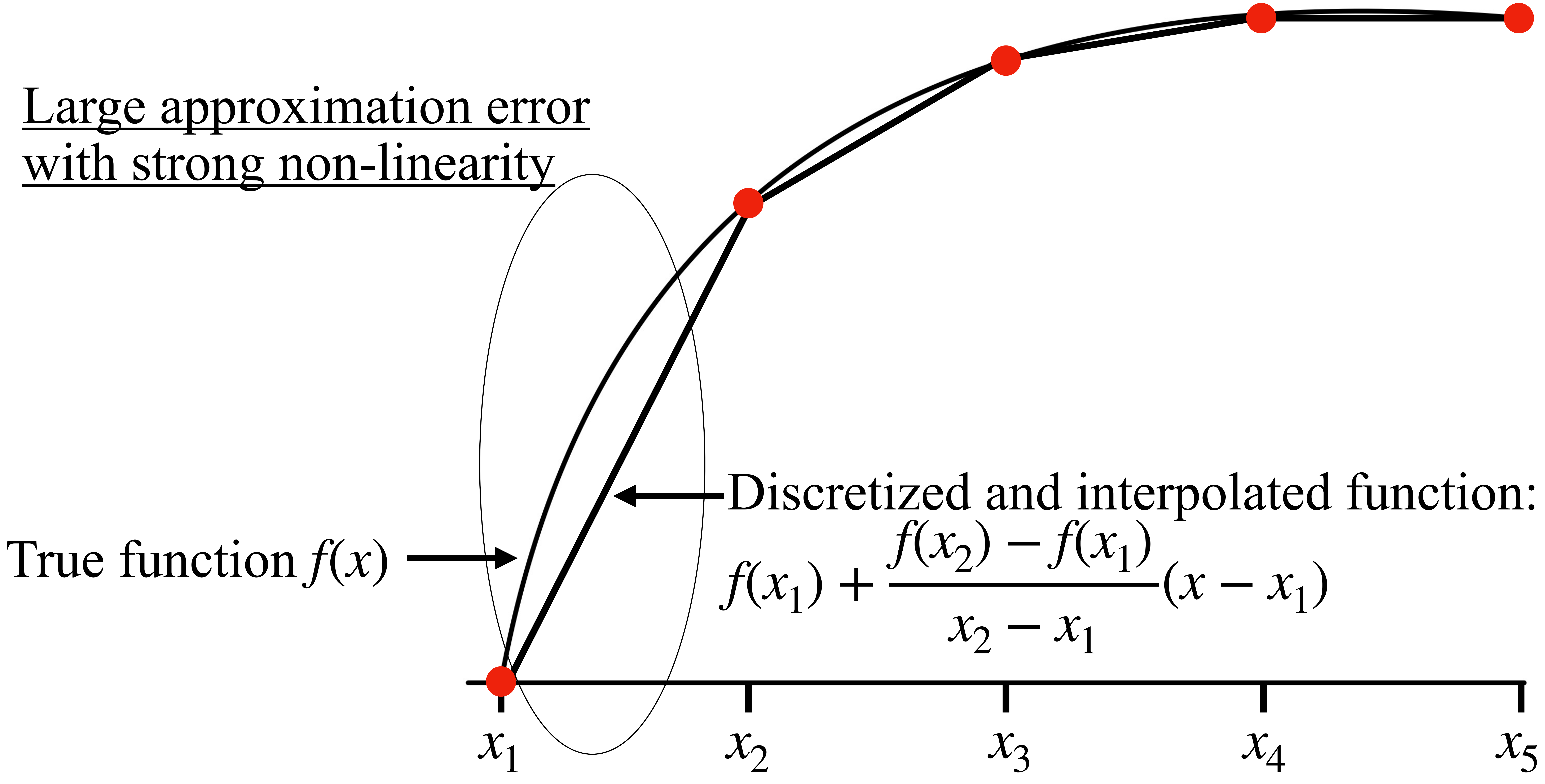
- Subject to $c = nh$, and $h' = f(n) + h$, where $f(\,\cdot\,)$ denotes human capital technology
- Given value function V, this problem is reduced to choosing optimal $n \in \{0,0.5,1\}$
- But $h' = f(n) + h$ may not be a grid point. How to compute $V(h')$?
- We can approximate such $V(h')$ by *interpolating* V

Interpolation

- Consider grid points $\mathscr{X} = \{x_1, \ldots, x_n\}$ and function $f(x)$ on $\mathscr{X}$.
- Focus on linear interpolation, the most straightforward method.
  - Another popular method is cubic spline interpolation.
- Based on linear interpolation, the approximated value of the function, $\hat{f}(x)$, is given by:

$$\hat{f}(x) = f(x_i) + \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}(x - x_i)$$

Linear interpolation
Graphical Intuition


Large approximation error
with strong non-linearity
True function $f(x)$
Discretized and interpolated function:
x_1
x_2
x_3
x_4
x_5

$$f(x_1) + \frac{f(x_2) - f(x_1)}{x_2 - x_1}(x - x_1)$$

Linear interpolation (cont'd)

Pros:

- Simple and needs only local info ($f(x_{i+1}), f(x_i)$)

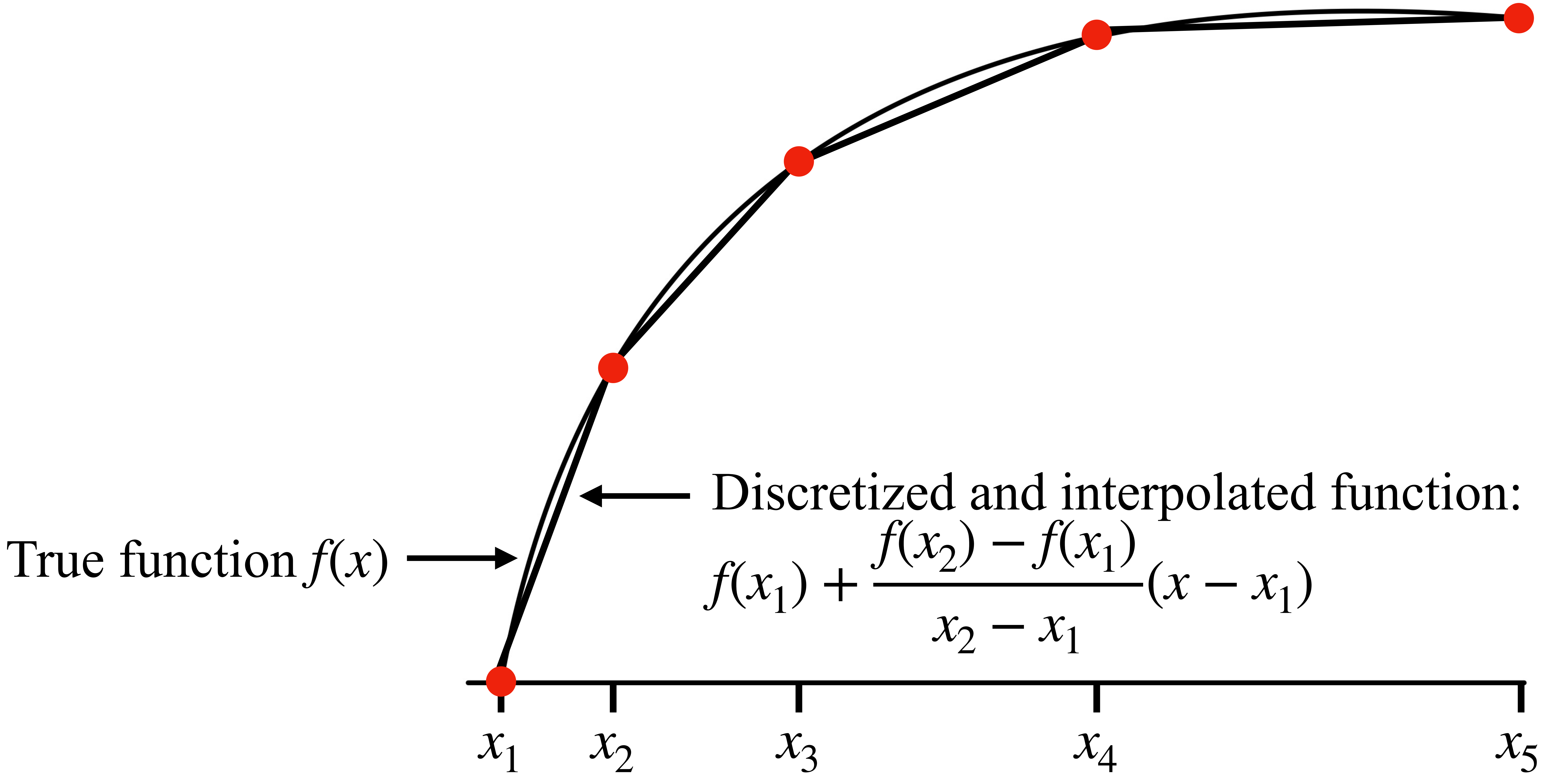
Cons:

- Approximated function is not differentiable at the grid points

- Approximation error can be large if the underlying function is highly non-linear
  - We often see a higher curvature at lower end of function
    – e.g., think about borrowing-constrained households with log utility
  - One way to mitigate the approx. error is to adopt unequally spaced grid points, e.g.,
    $x_i = x_1 + \left(\frac{i-1}{N-1}\right)^{\phi}(x_N - x_1)$ for each $i = 1,\ldots,N$ with some $\phi > 1$ and $x_1 < x_N$

Linear interpolation

Allowing unequally spaced grid points



$$f(x_1) + \frac{f(x_2) - f(x_1)}{x_2 - x_1}(x - x_1)$$

Root finding

Root finding

- We often have to solve for the root of a non-linear equation.
  - Here, we focus on the one-dimensional case: $f(x) = 0$ where $x \in \mathcal{X}$

- One simple approach is a grid search over a discretized space $\{x_1, \ldots, x_n\}$.
  - Works always, but involves a trade-off btw accuracy and speed
  - Need n to be sufficiently large to obtain more accurate solution, taking much time

- Algorithms that are both more accurate and faster.
  - Bisection
  - Newton-Raphson

Bisection

Algorithm

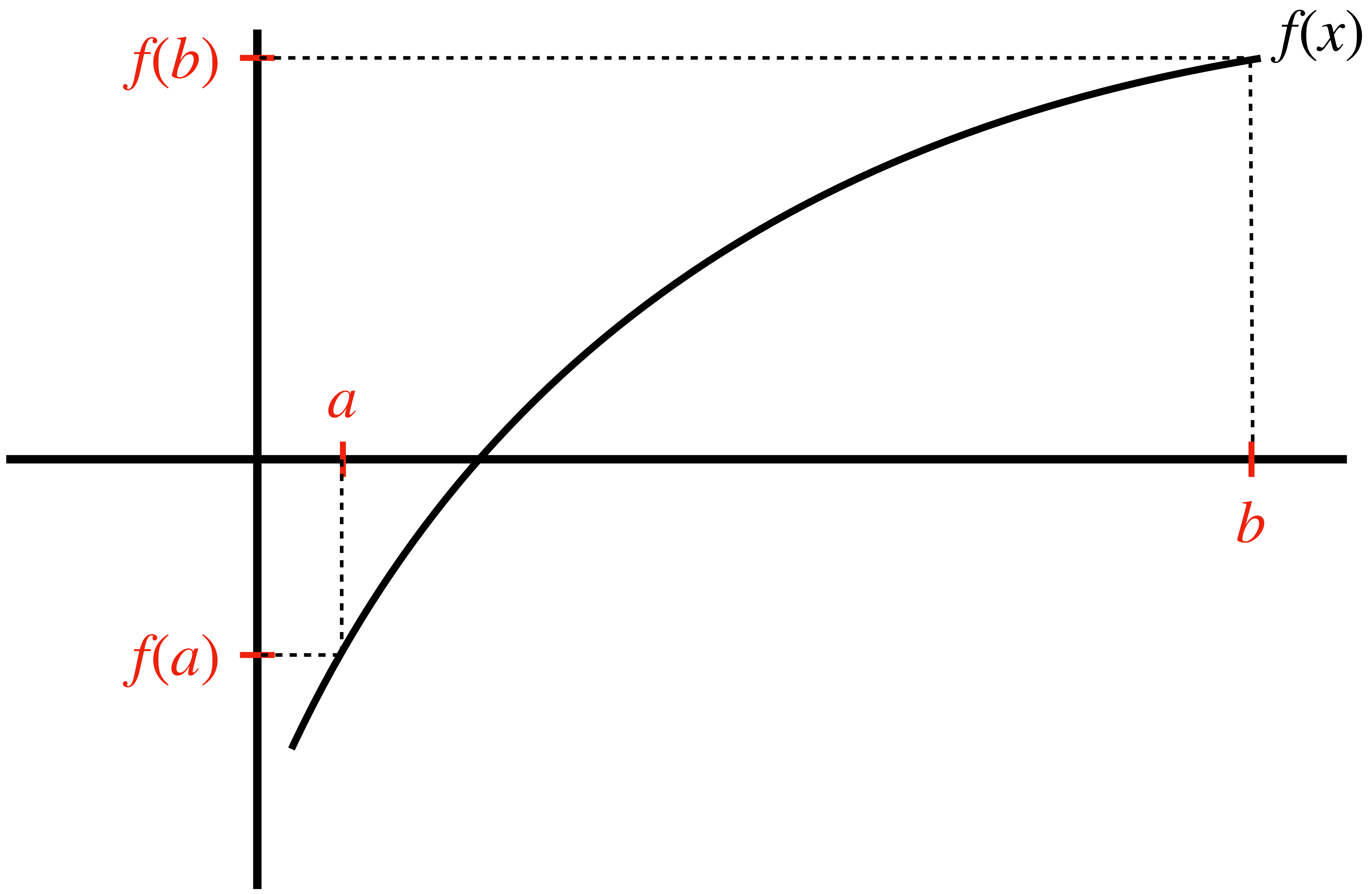
1. *Bracketing*: Find $a \in \mathscr{X}$ and $b \in \mathscr{X}$ where $\text{sign}(a) \neq \text{sign}(b)$.

   $\Rightarrow$ If $f(x)$ is continuous, at least one solution to $f(x) = 0$ exists in the bracket $[a, b]$

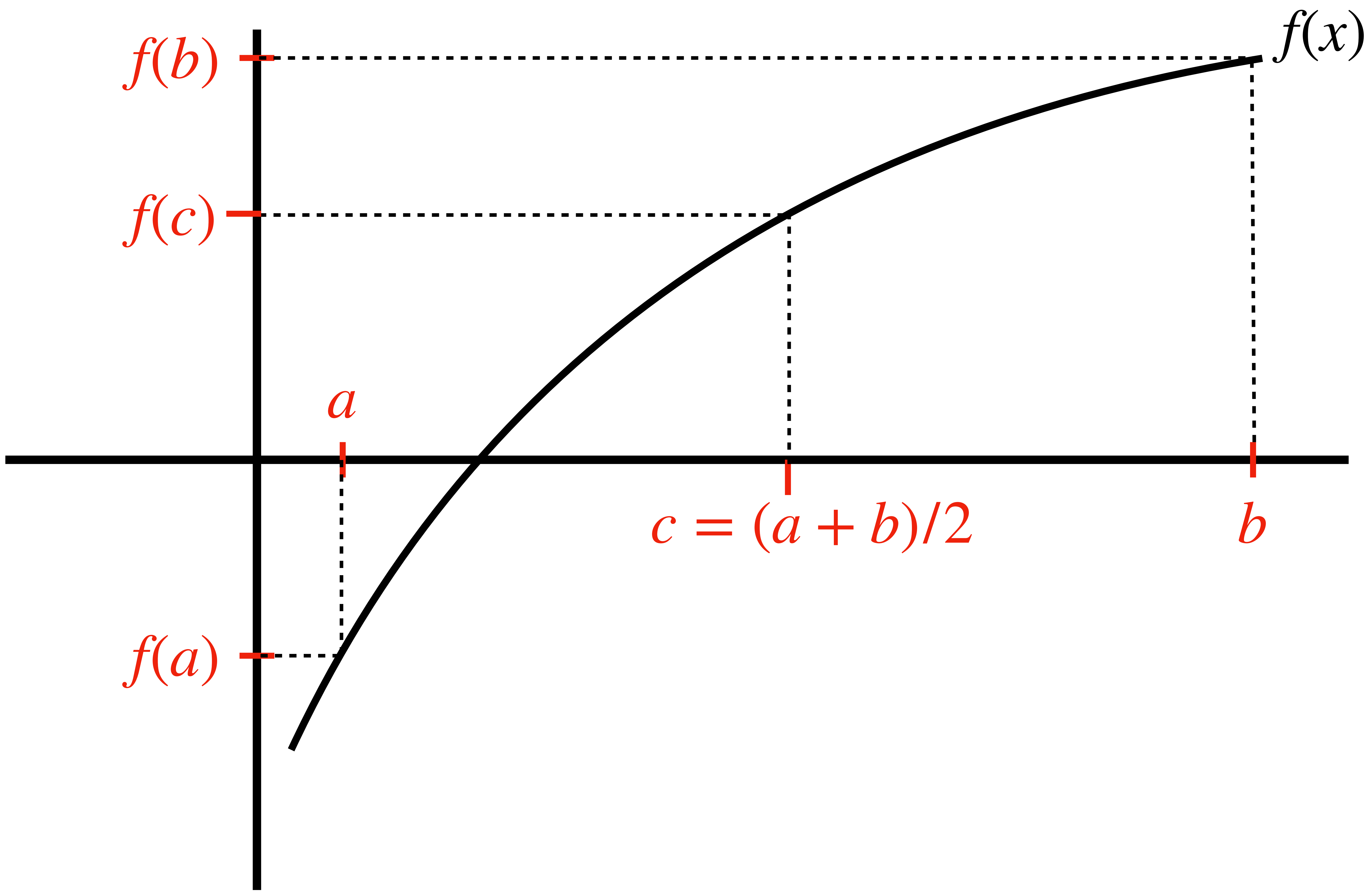
2. *Update*: Let $c = (a + b)/2$. Set a new interval between $\bar{x}$ and c, where $\bar{x} \in \{a, b\}$ and $\text{sign}\, f(c) \neq \text{sign}\, f(\bar{x})$.

3. Repeat Step 2 until the two points get close enough (e.g., 10^{-6}).

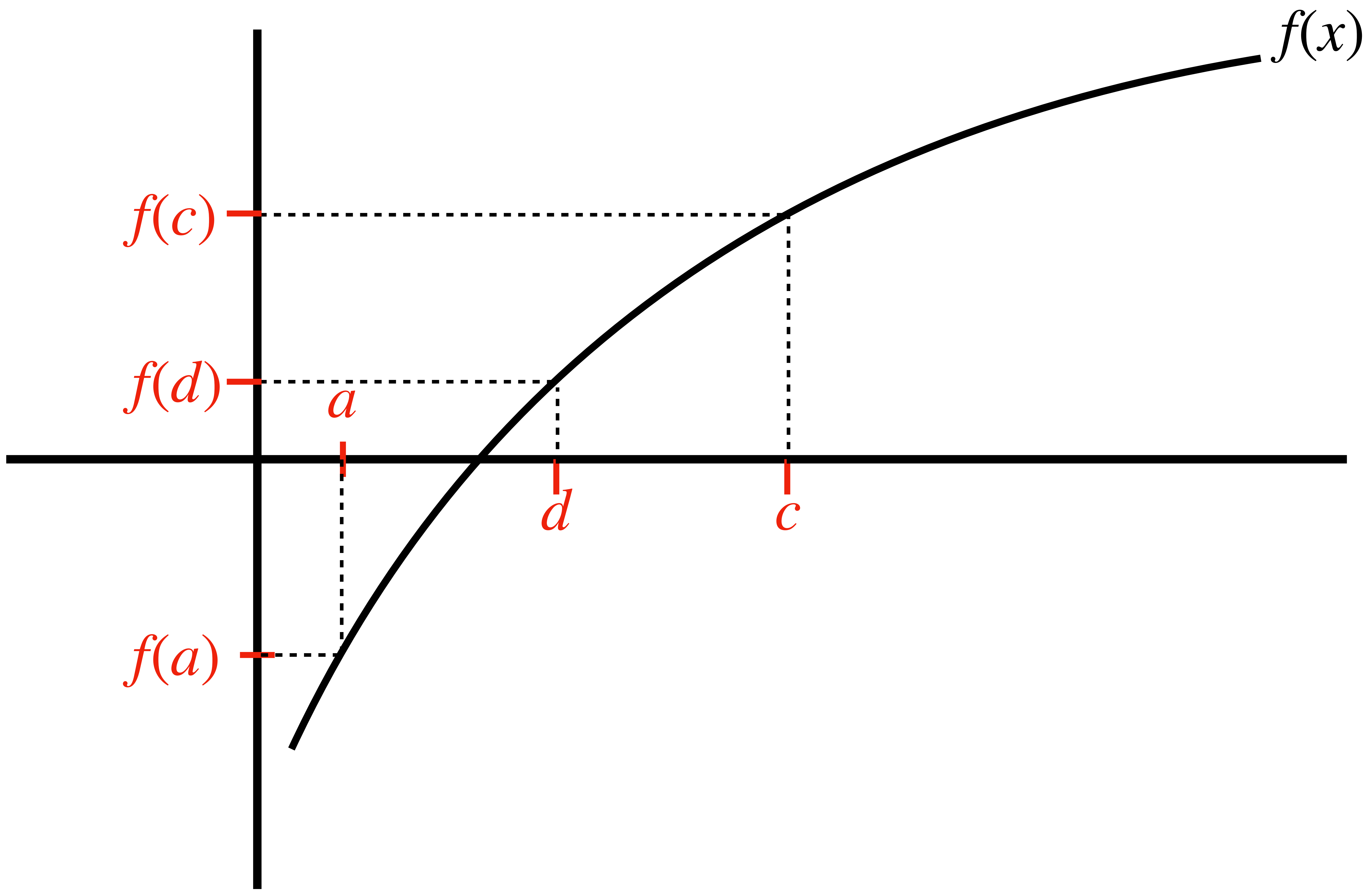
Bisection: an example



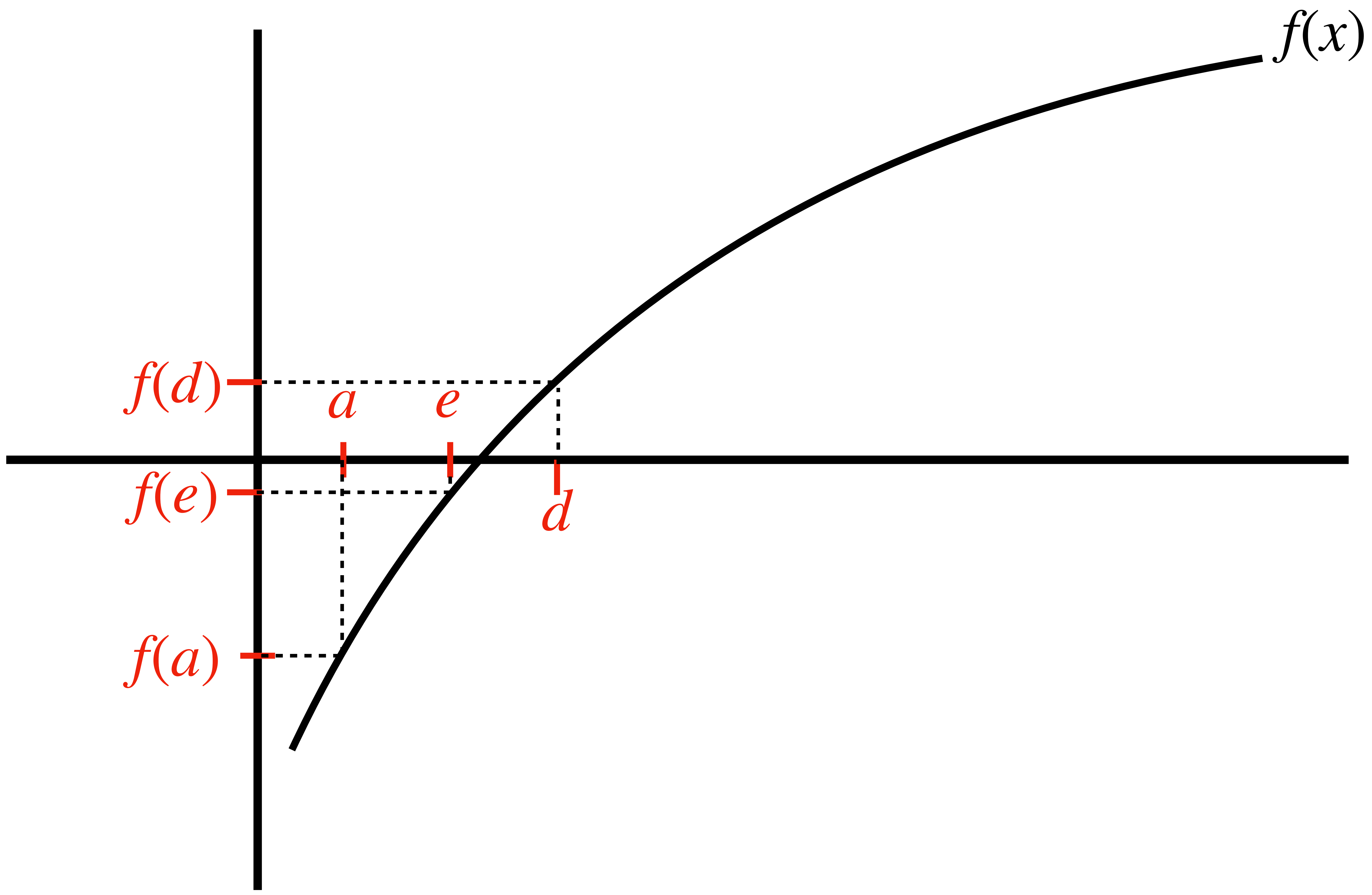
Bisection: an example



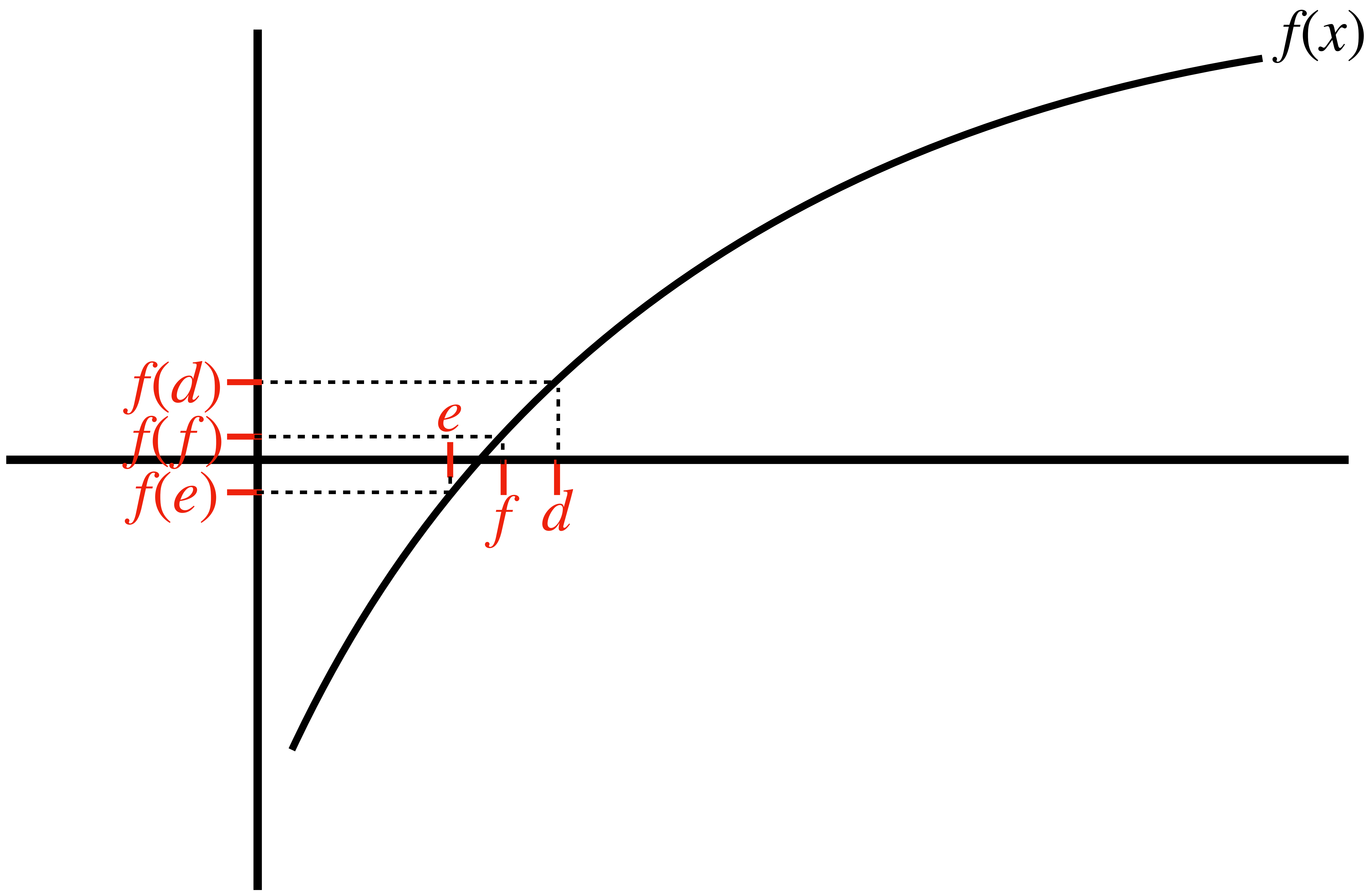
Bisection: an example



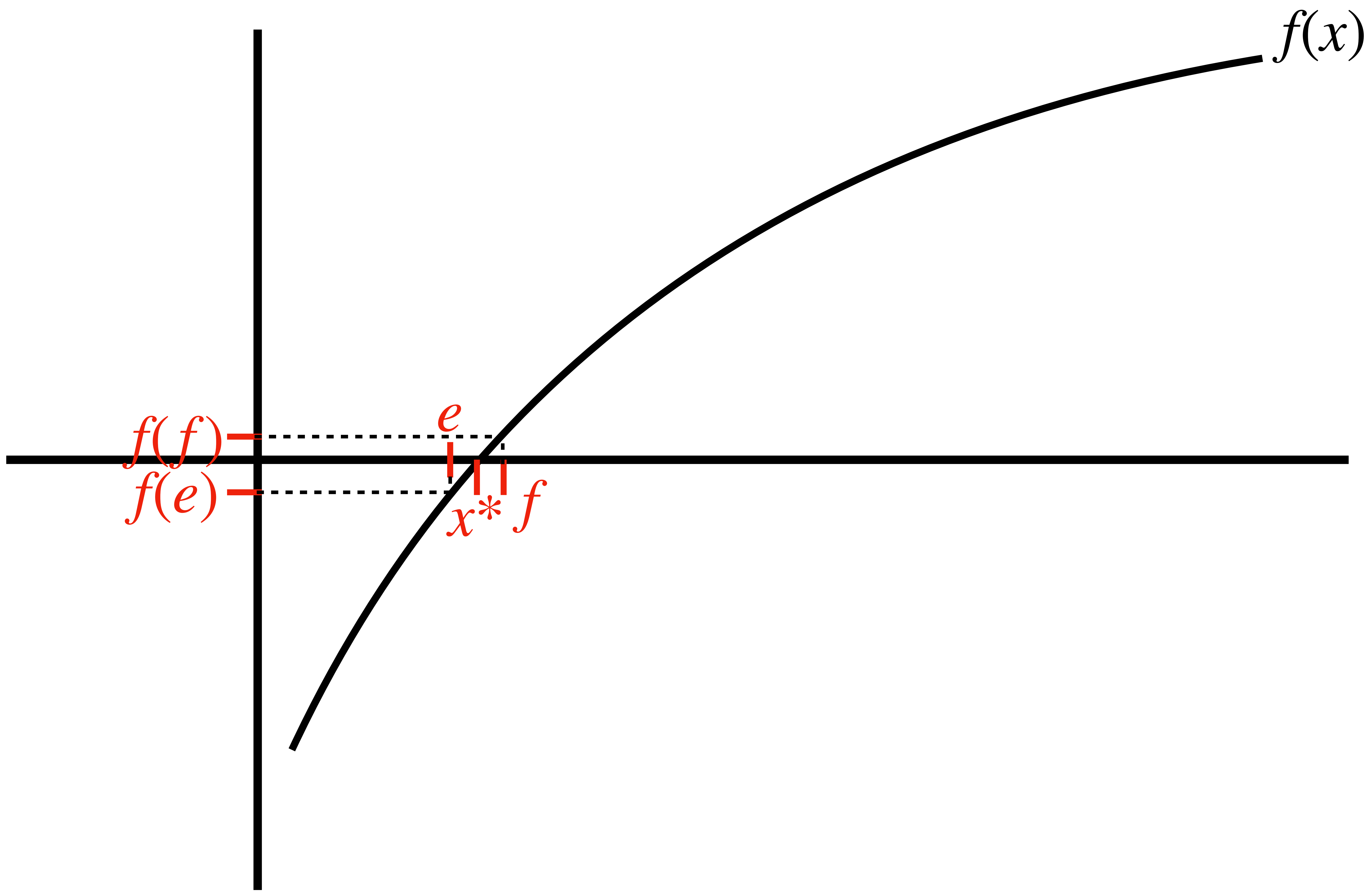
Bisection: an example



Bisection: an example



Bisection: an example



Bisection

Discussion

- Does not require differentiability of $f(x)$
- Always ends after a set of time
- Faster than grid search in general, but can be slower than other method (e.g., Newton-Raphson can be much faster if $f(x)$ is closer to linear)

Newton-Raphson
Idea

- Linear approximation of $f(x)$ with a starting point x_0:

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0)$$

- If this approximation is good, the solution for $f(x) = 0$, x^*, is close to:

$$x^* = x_0 - \frac{f(x_0)}{f'(x_0)}$$

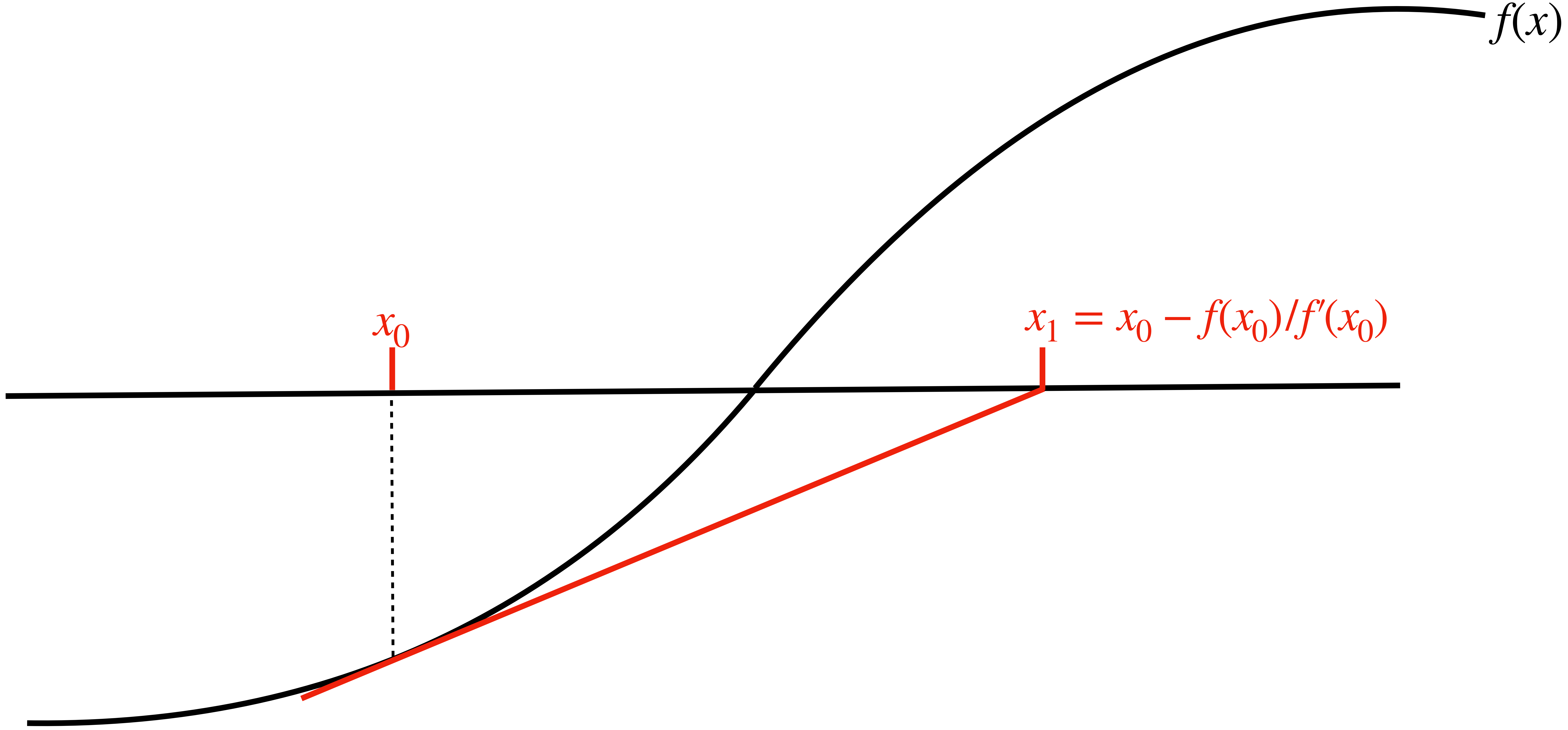
- Use this formula to find the root

Newton-Raphson

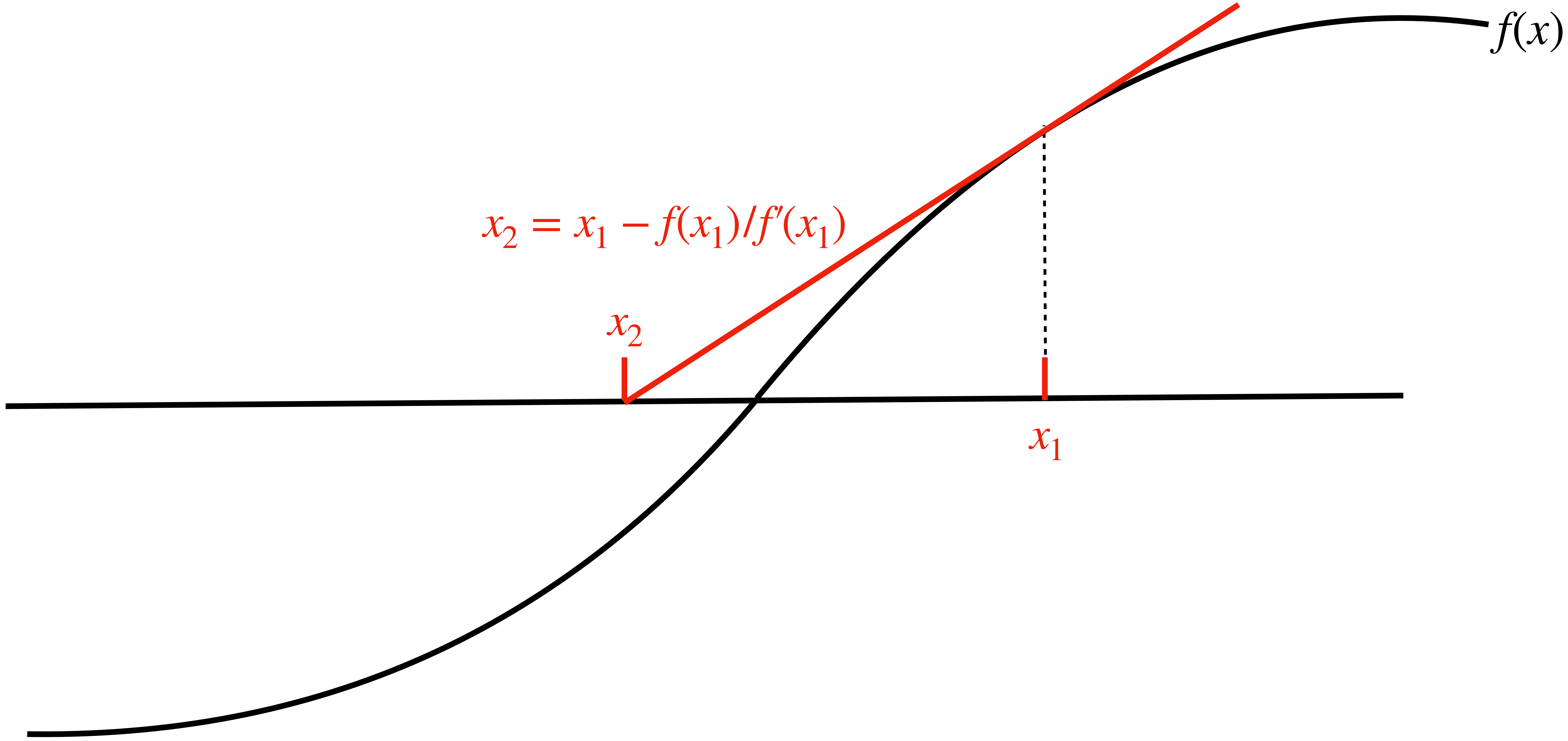
Algorithm

1. Make an initial guess x_0
2. Construct x_1 according to the formula: $x_1 = x_0 - f(x_0)/f'(x_0)$
3. Check whether $f(x_1) \simeq 0$. If not, use x_1 and construct x_2, going back to the previous step. Continue until $f(x_i) \simeq 0$.

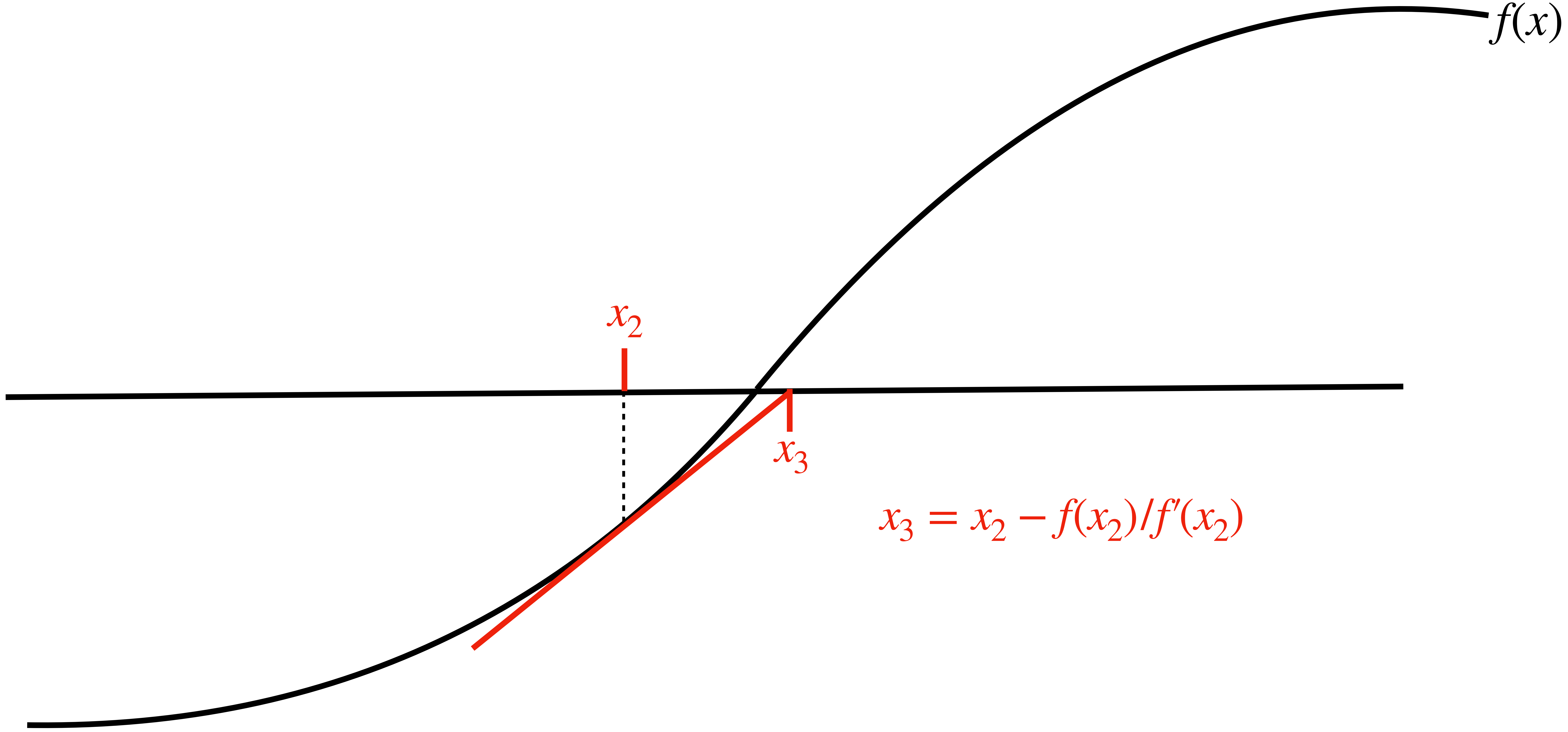
Newton-Raphson: idea



Newton-Raphson: idea

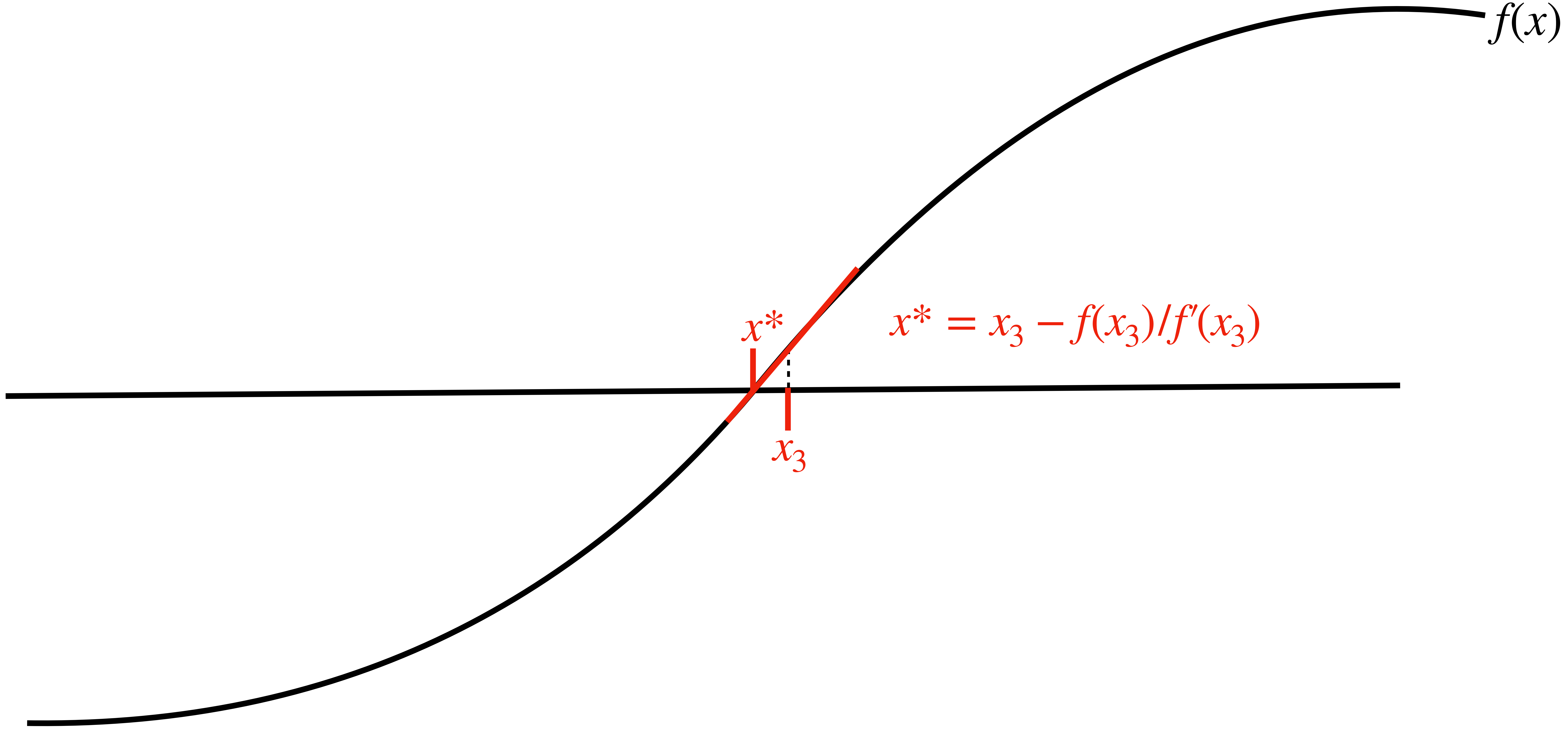


Newton-Raphson: idea



$x_3 = x_2 - f(x_2)/f'(x_2)$

Newton-Raphson: idea



Newton-Raphson
Discussion

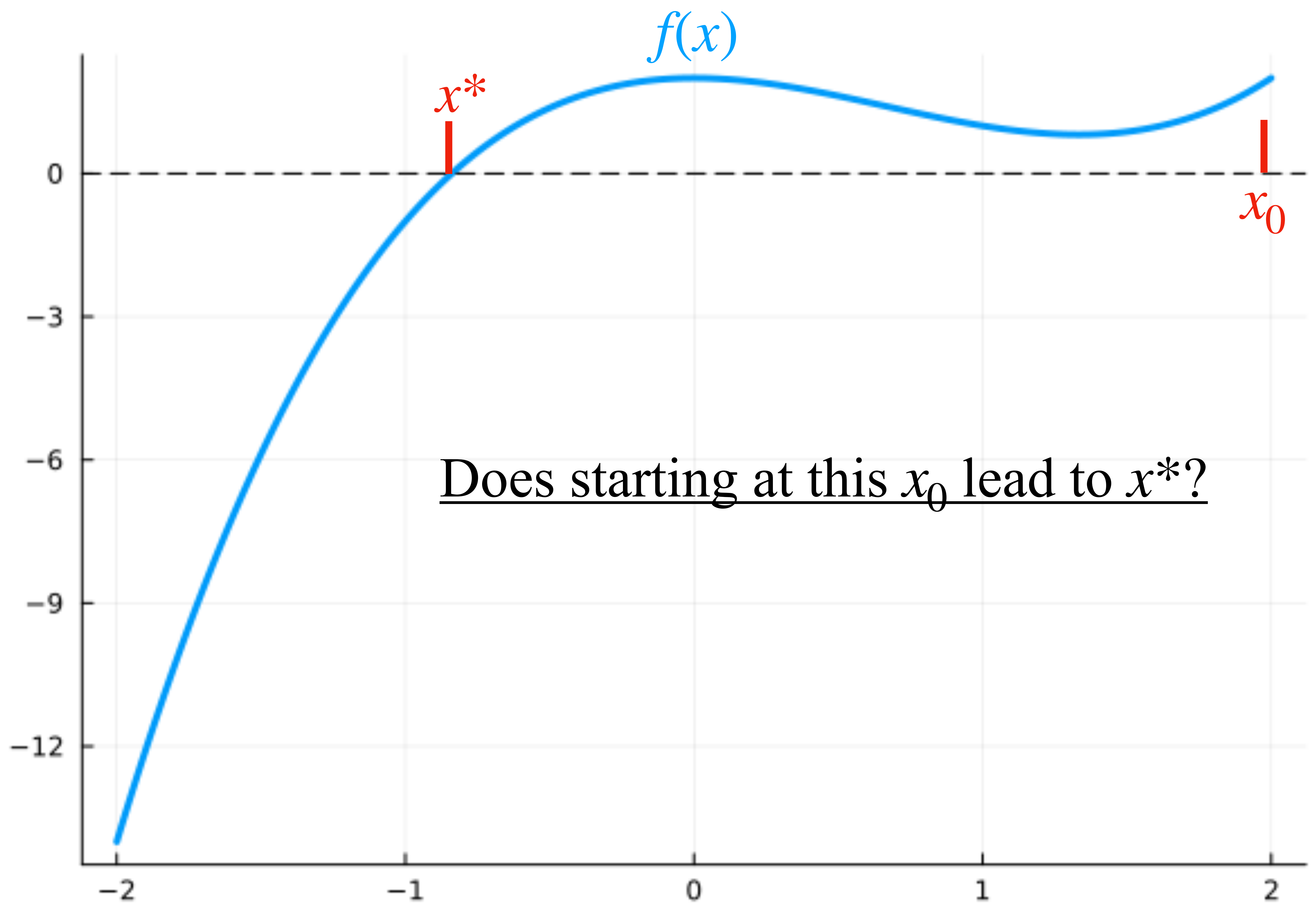
- Can be much faster than bisection if $f(x)$ is close to linear

- Downside:
  - (1) requires differentiability
  - (2) may fail to find a solution
    – initial guess very important esp. when f is highly non-linear (example next page)

- Note: even if you cannot compute $f'(x)$ analytically, you may do it numerically based on a *finite-difference method* by computing $f'(x)$ as:

$$f'(x) \simeq \frac{f(x+h) - f(x-h)}{2h},$$

with some small $h > 0$



<u>Does starting at this x_0 lead to x^*?</u>

Optimization

Optimization

- Consider maximizing a function $F(x)$
- We can optimize $F(x)$ by grid search; fail-safe but very slow
- We introduce two popular methods
    - Golden-section search
    - Newton’s method

Golden-section search

Algorithm

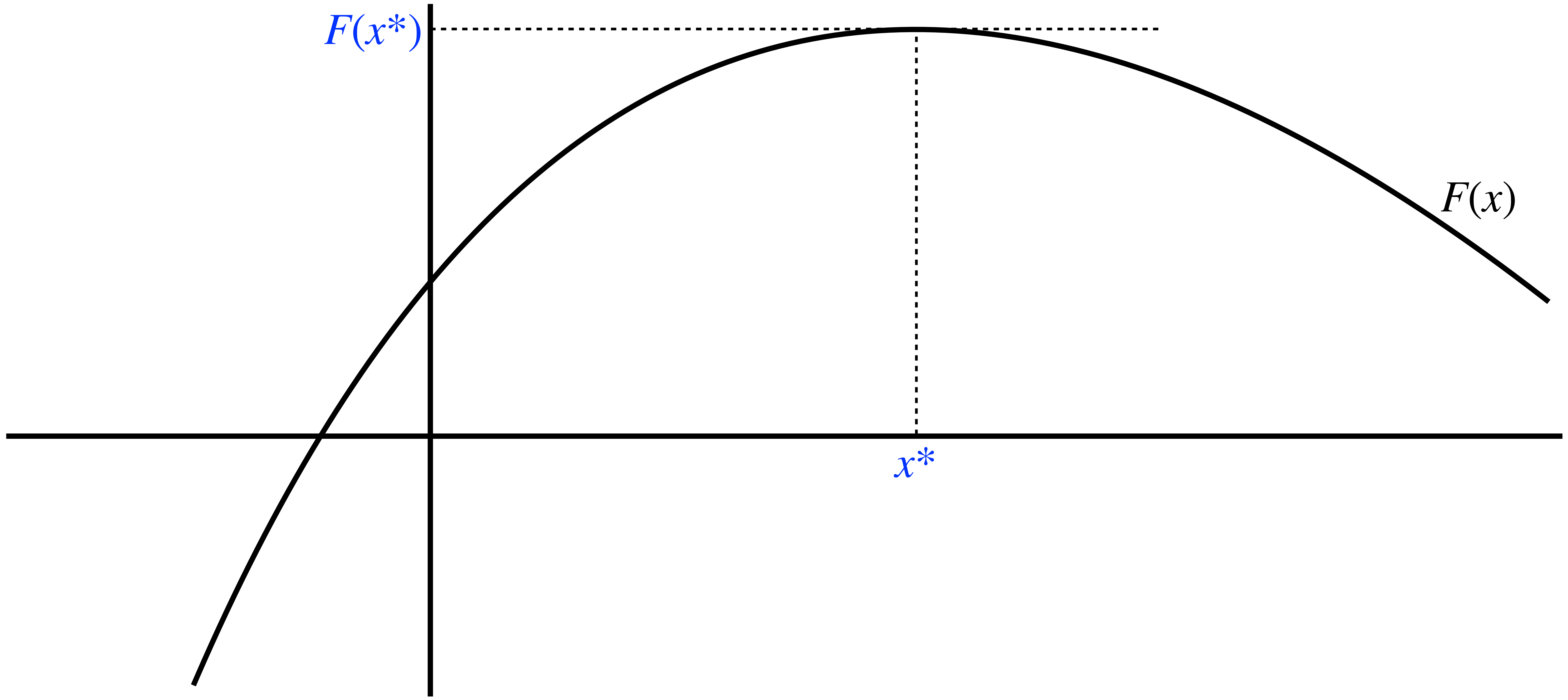
Idea is the same as bisection

1. *Bracketing*: Find three values a, b, and c s.t. $a < b < c$ and $F(a) < F(c) < F(b)$
   $\Rightarrow$ a (local) maximum exists btw a and c (if not, a corner solution is likely)

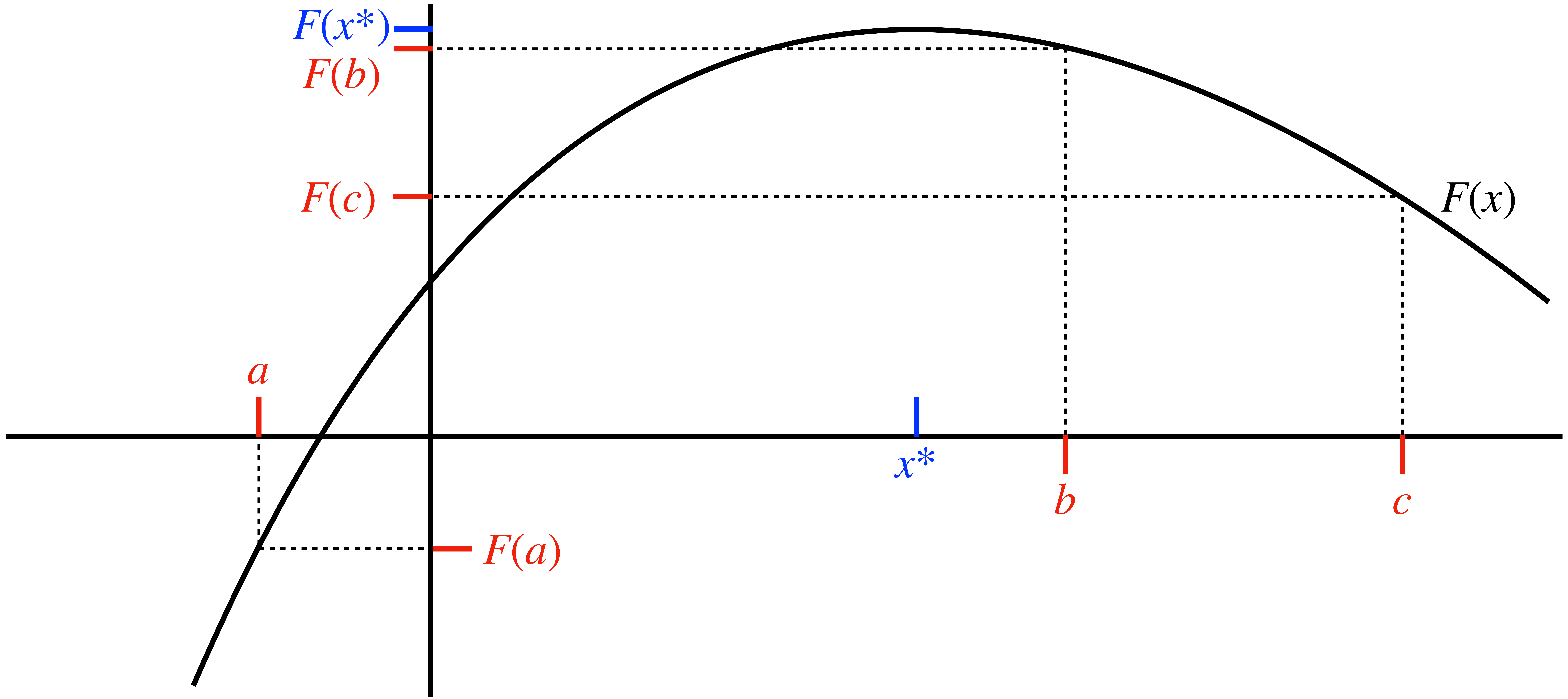
2. *Update*: Take the longer segment btw $[a, b]$ and $[b, c]$. Suppose it's $[a, b]$. Take a point x s.t. $(b-x)/(b-a) = \omega \in (0,1)$. If $F(x) > F(b)$ ($F(b) > F(x)$), eliminate c (a) from the bracket while including x.

3. Repeat Step 2 until the size of bracket is less than a tolerance value

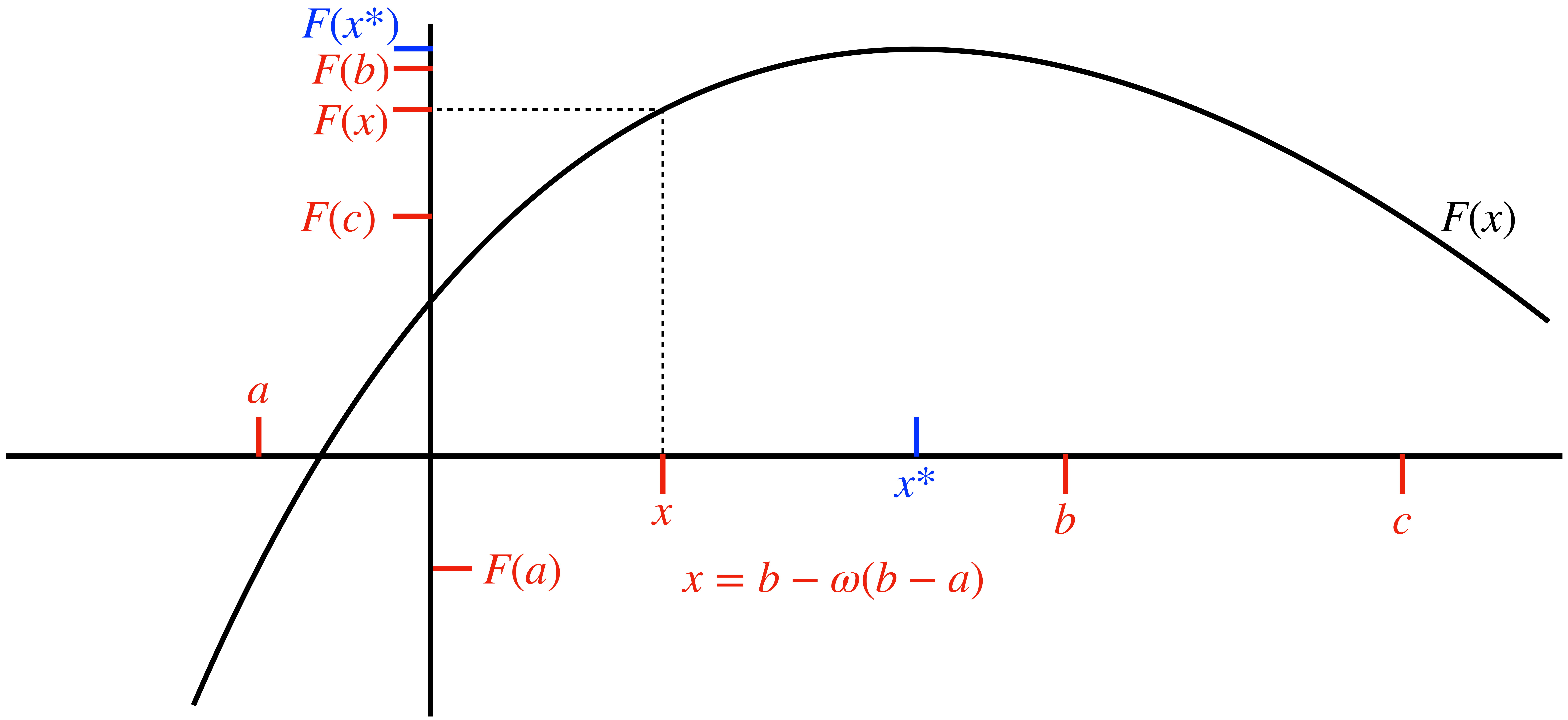
Golden-section search: idea



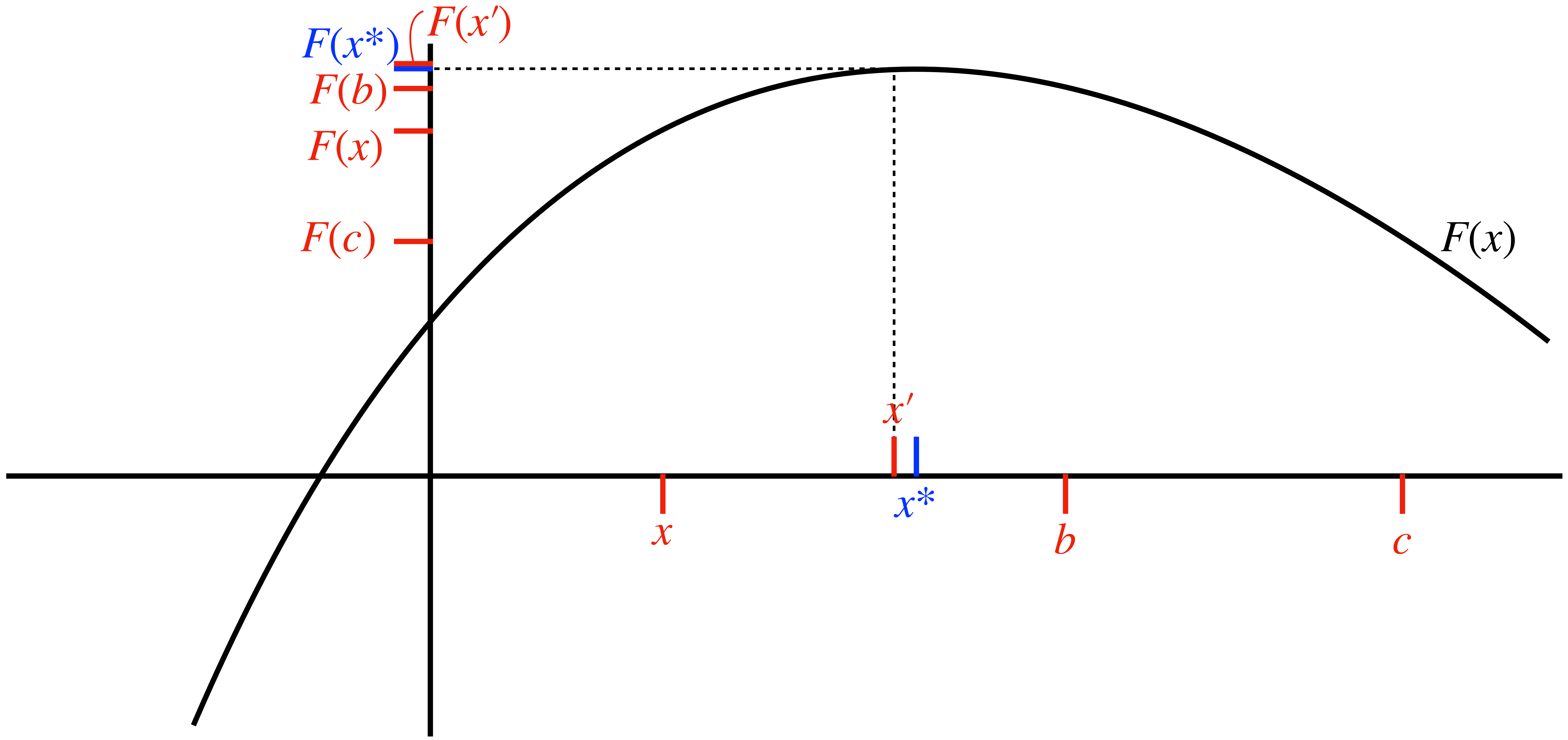
Golden-section search: idea



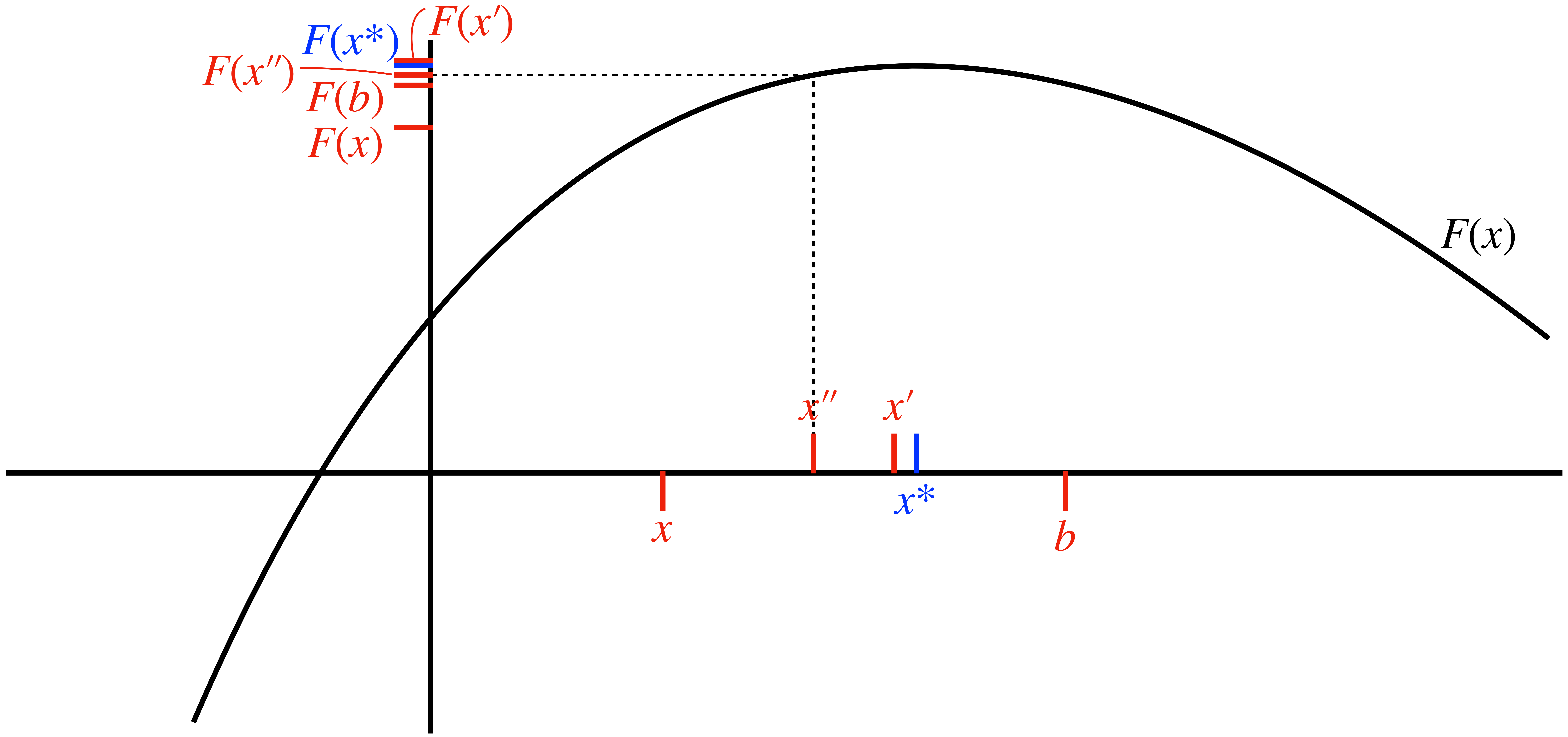
Golden-section search: idea



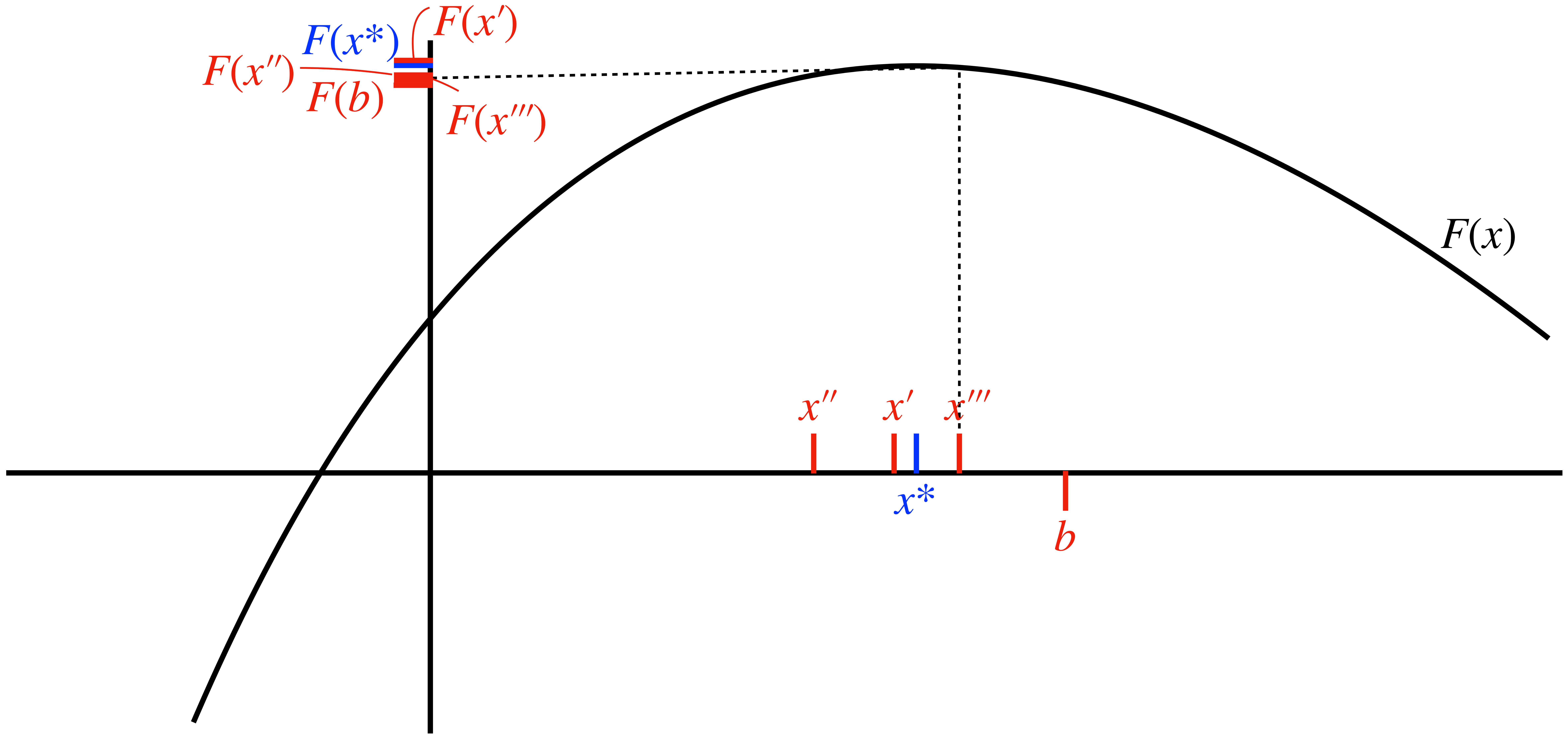
Golden-section search: idea



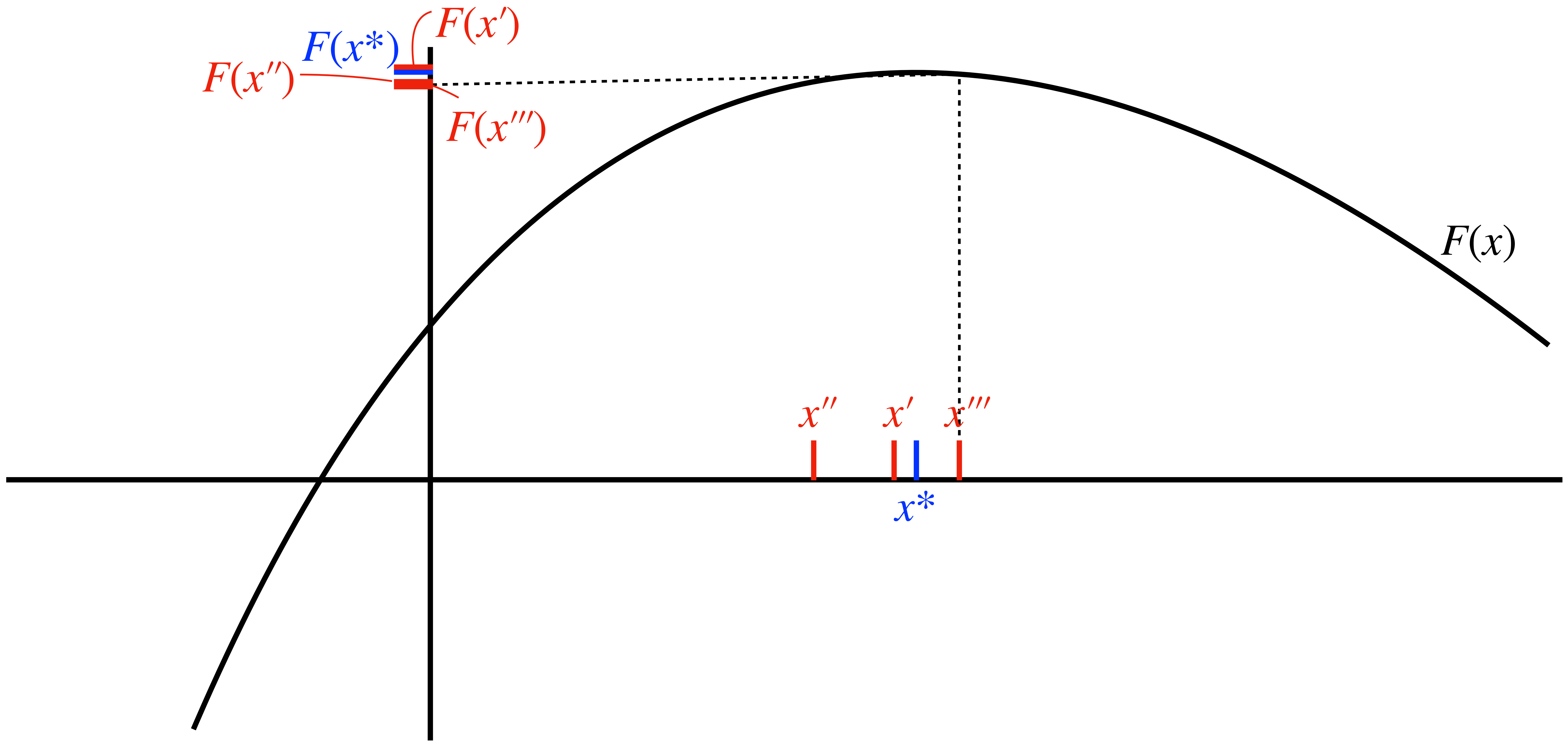
Golden-section search: idea



Golden-section search: idea



Golden-section search: idea



Golden-section search

Some remarks

- Does not require differentiability of $F(x)$ and ends after a set of time
- How to set ω:
  - We often use $\omega = (3 - \sqrt{5})/2 \approx 0.38197$
  - The ratio $(1 - \omega)/\omega \approx 1.61803$ is called *golden ratio* or *golden section*
  - With *s*uch ω, each iteration reduces the interval by a constant factor of $1 - \omega$

Newton’s method

- Second-order approximation of $F(x)$ with a starting point x_0:

$$F(x) \approx F(x_0) + F'(x_0)(x - x_0) + \frac{1}{2}F''(x_0)(x - x_0)^2$$

- Taking the first-order condition of RHS w.r.t. x and equating it with zero, we have

$$\hat{x} = x_0 - \frac{F'(x_0)}{F''(x_0)}$$

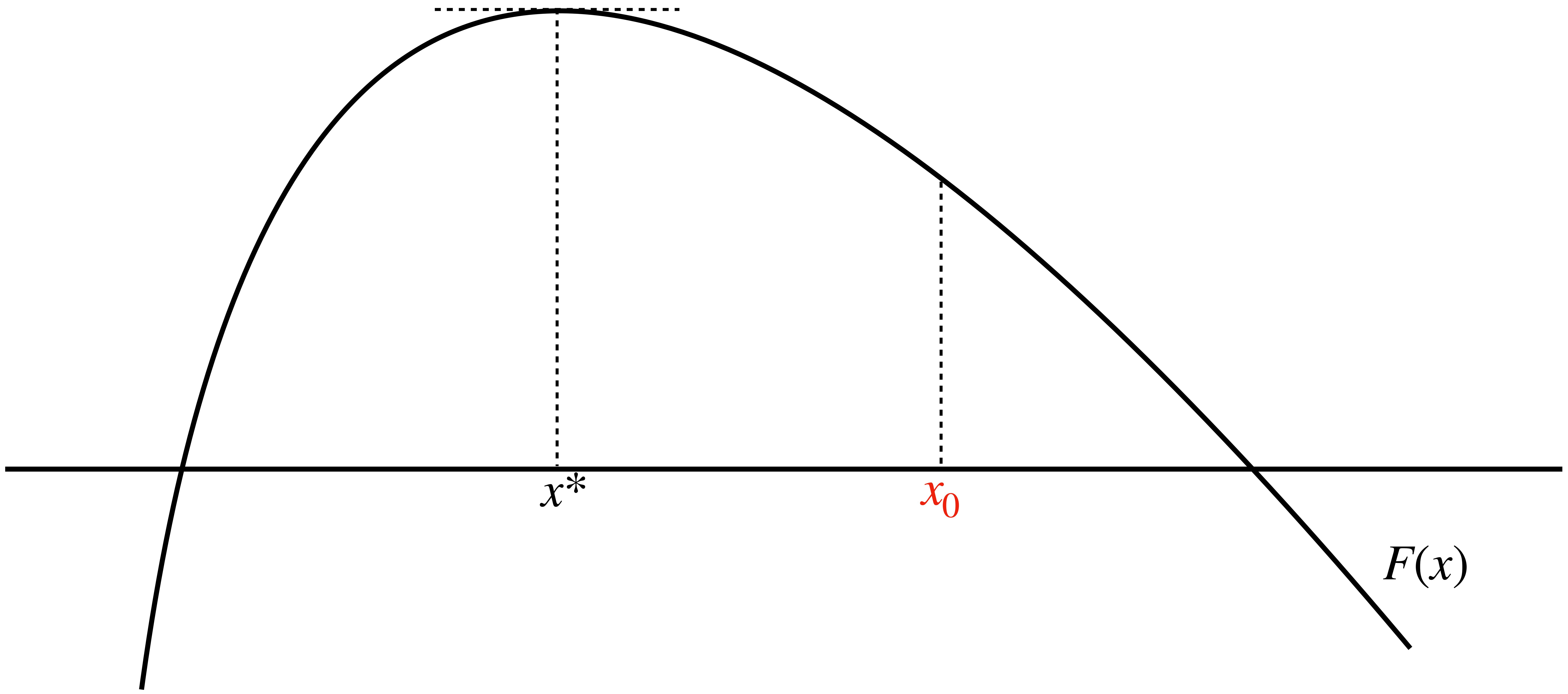
- Use this formula to find the solution

Newton's method

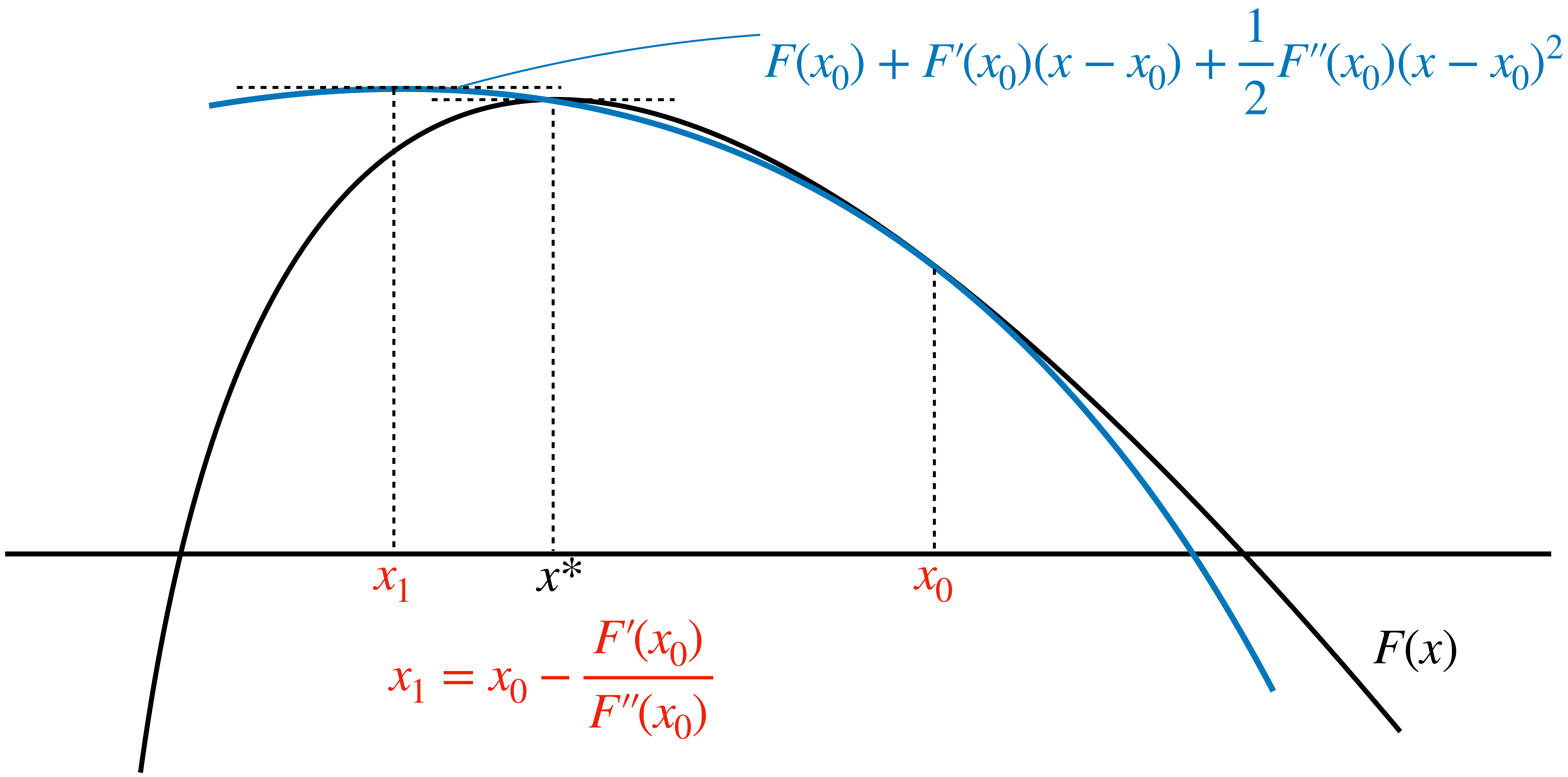
Algorithm

- Make an initial guess x_0
- Construct x_1 s.t. $x_1 = x_0 - F'(x_0)/F''(x_0)$
- Check whether $|x_0 - x_1| < \varepsilon$ where $\varepsilon > 0$ is the tolerance value. Repeat Step 2 until convergence according to the formula $x_i = x_{i-1} - F'(x_{i-1})/F''(x_{i-1})$ where i denotes the count of iteration.

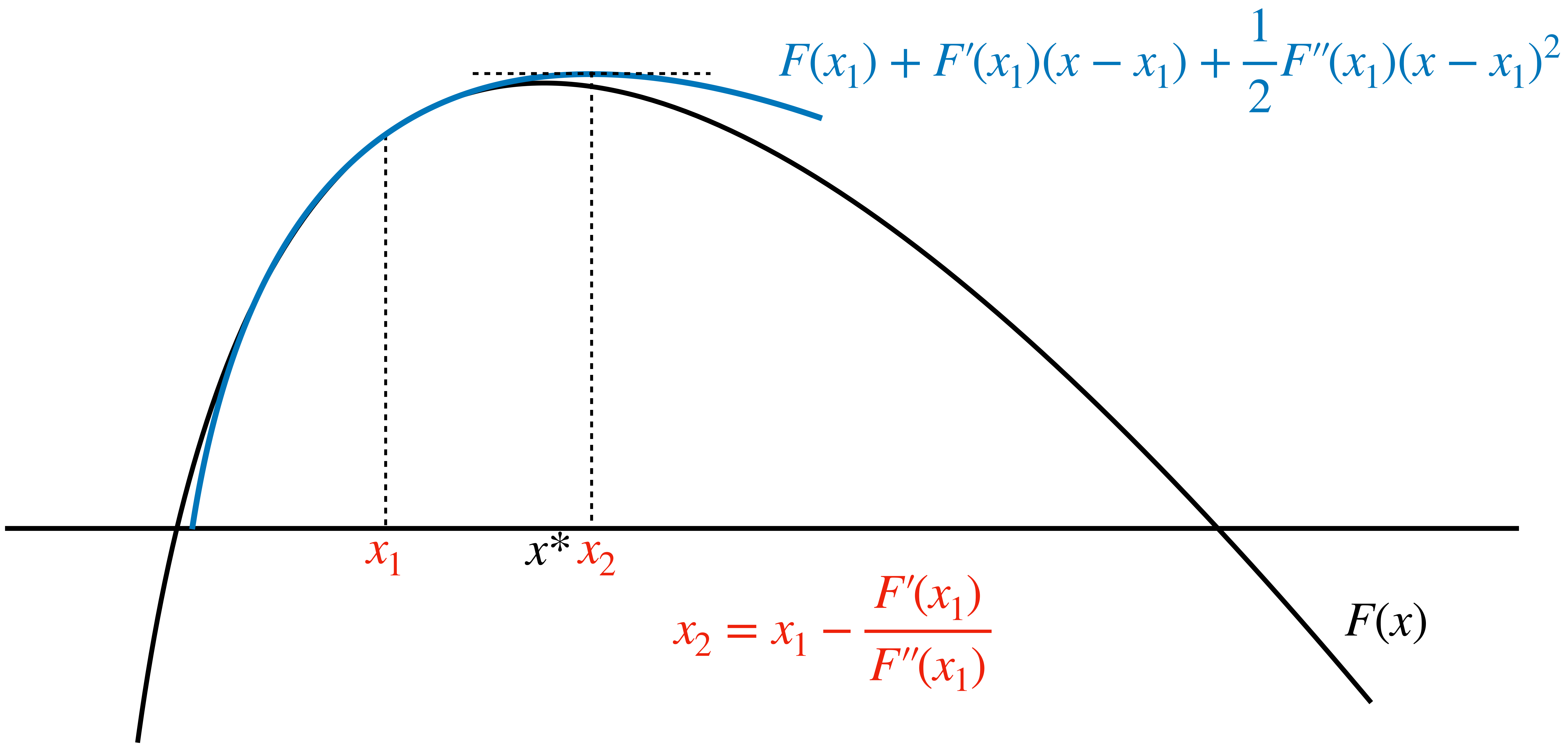
Newton’s method: idea



Newton's method: idea


$F(x_0) + F'(x_0)(x - x_0) + \frac{1}{2}F''(x_0)(x - x_0)^2$
x_1
x^*
x_0
$F(x)$
$x_1 = x_0 - \frac{F'(x_0)}{F''(x_0)}$

Newton's method: idea



$F(x_1) + F'(x_1)(x - x_1) + \frac{1}{2}F''(x_1)(x - x_1)^2$
x_1
x^*
x_2
$x_2 = x_1 - \frac{F'(x_1)}{F''(x_1)}$
$F(x)$

Newton’s method
Discussion

- Can be substantially faster especially if $F(x)$ is close to quadratic
- Caveats: (1) needs 1st and 2nd derivatives; (2) no guarantee of finding optimum
- Starting from a good initial guess is helpful

Final remarks

- Relationship btw root finding and optimization
  - “Optimizing” can be considered as “finding a root of FOC”
  - If little is known about the properties of the function, the golden-section method is advantageous because it does not require differentiability
- Importance to understand the algorithms and their properties, which helps you recognize why a method might fail
- Try coding the algorithm once: it will boost your understanding (see exercise)

Exercise

Consider the following static problem:

$$\max_{c>0, l\in(0,1)} \ln(c) - \frac{\phi}{\gamma} l^{\gamma}$$

Subject to

$$c = l^{\alpha} + x,$$

where c, l, and x denote consumption, labor supply, and non-labor income.

Consider that $\alpha = 0.33$, $\phi = 1$, $\gamma = 2$ and $x = 0.1$. Solve the problem using the four methods we learned.