Lecture 5:

Solving Nonlinear Equations

Markus Hohle

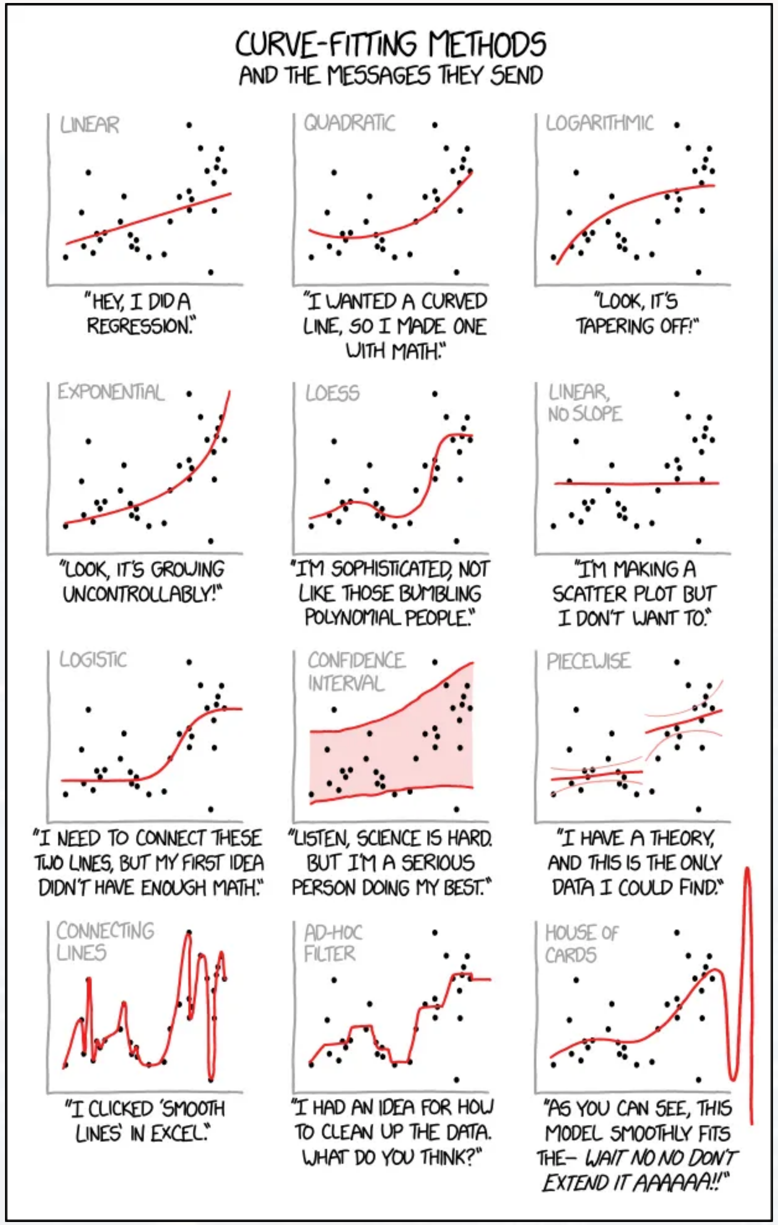
University California, Berkeley

Numerical Methods for Computational Science

MSSE 273, 3 Units

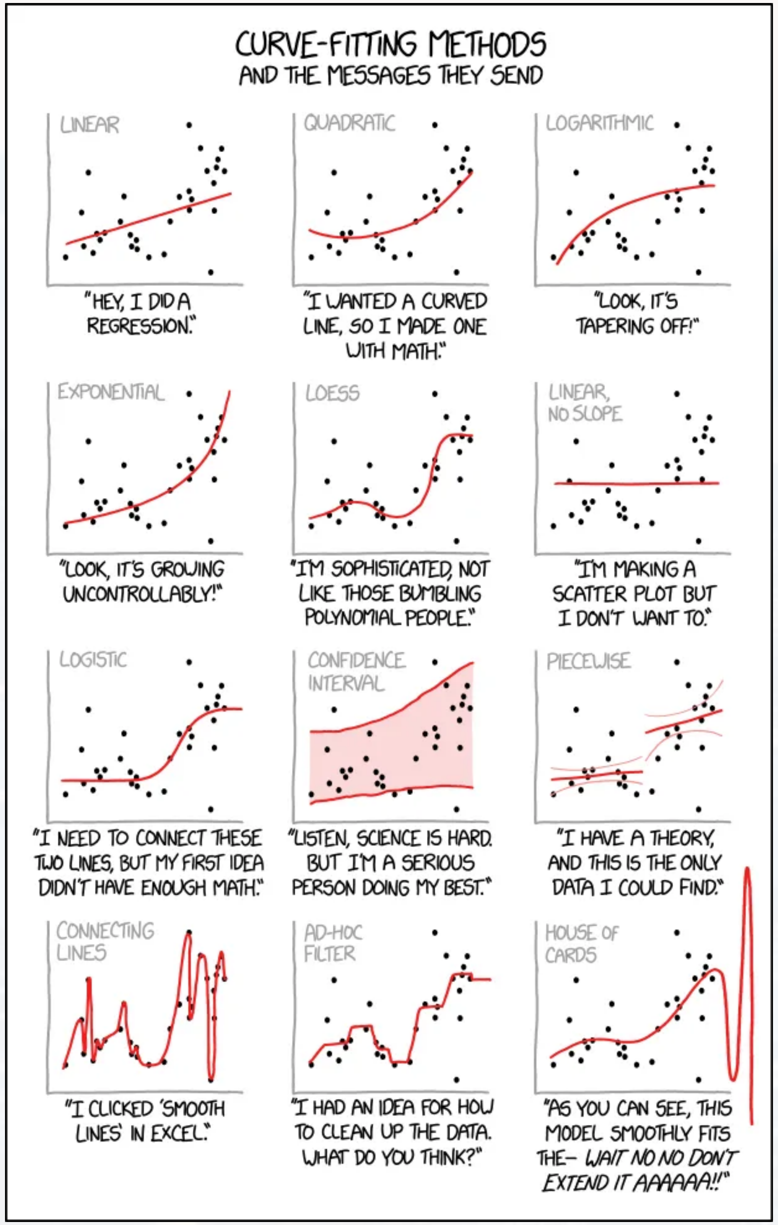
Course Map

Week 1:	Introduction to Scientific Computing and Python Libraries
Week 2:	Linear Algebra Fundamentals
Week 3:	Vector Calculus
Week 4:	Numerical Differentiation and Integration
Week 5:	**Solving Nonlinear Equations**
Week 6:	Probability Theory Basics
Week 7:	Random Variables and Distributions
Week 8:	Statistics for Data Science
Week 9:	Eigenvalues and Eigenvectors
Week 10:	Simulation and Monte Carlo Method
Week 11:	Data Fitting and Regression
Week 12:	Optimization Techniques
Week 13:	Machine Learning Fundamentals



Outline

- The Problem
- Newtons Method
- Bisection



Outline

- The Problem

- Newtons Method

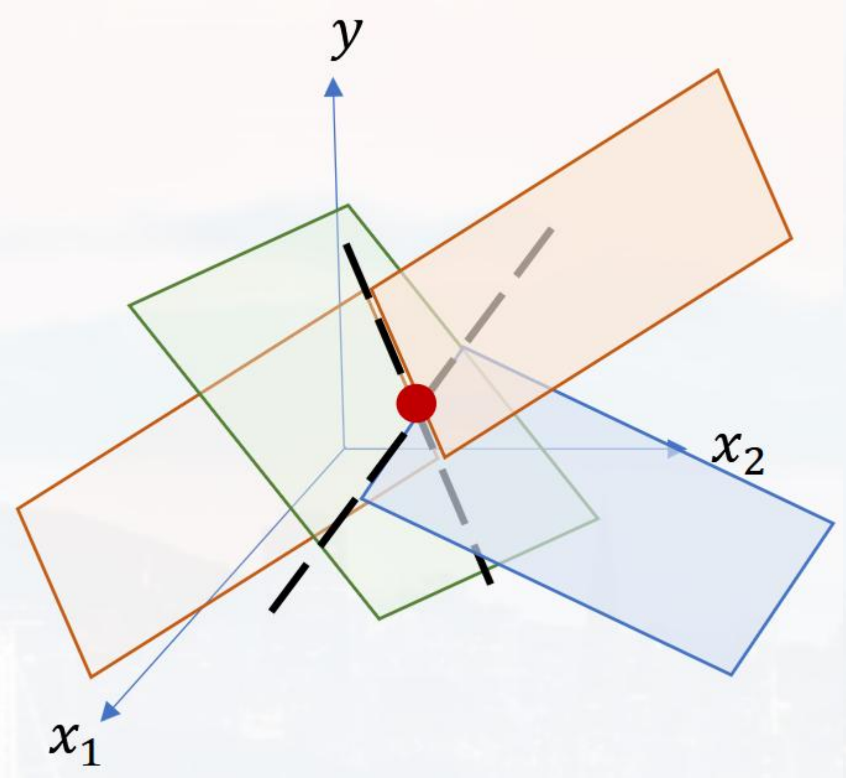
- Bisection

Solving Nonlinear Equations

We know how to solve a set of linear equations:

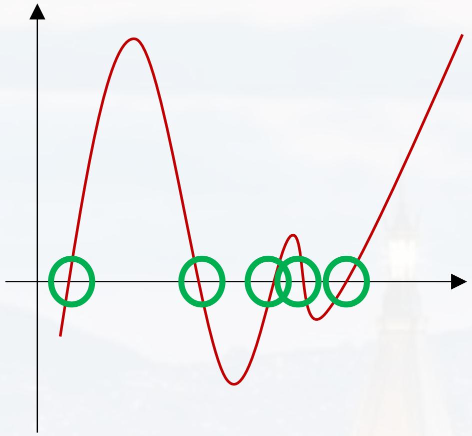
$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}}_{\vec{x}} = \underbrace{\begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_m \end{bmatrix}}_{\vec{c}}$$

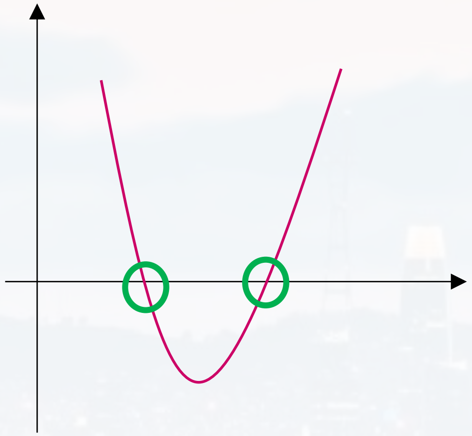
$$\boxed{A\vec{x} = \vec{c}}$$

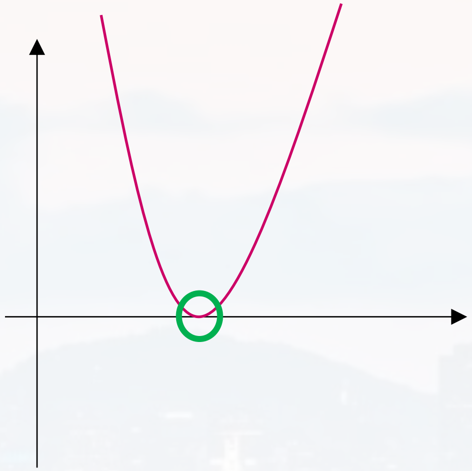


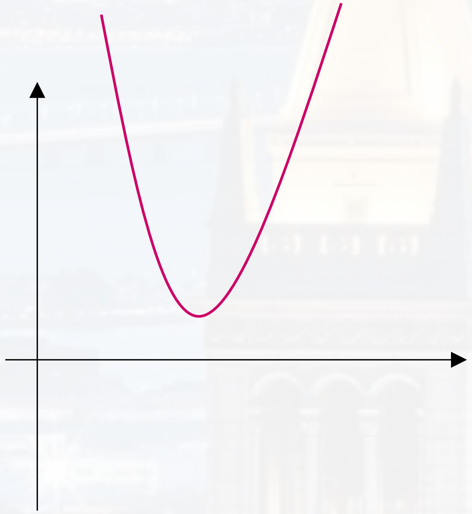
However: what is about non-linear equations?!

root finding: finding the **zeros** of a polynomial









How many roots does a polynomial have?

How many roots does a polynomial have?

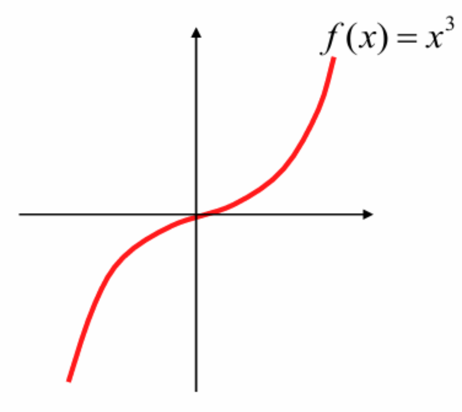
$$f_N(x) = \sum_{i=0}^{N} a_i x^i = \alpha \prod_{i=1}^{N} (x - x_i)$$

factored form

x_i : zeros

- a polynomial of **Nth order** has **N roots** (real & complex)
- for N $\geq$ 5: no analytical solutions
- for N is odd: at least one real zero

$$f(x) = x^3 = (x - x_1)(x - x_2)(x - x_3)$$



zeros: $x_1 = x_2 = x_3 = 0$

one zero with multiplicity m = 3

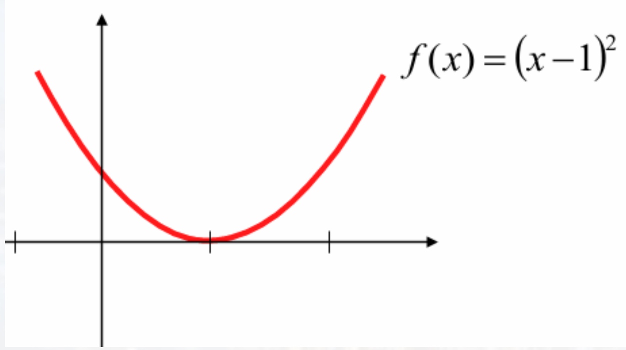
How many roots does a polynomial have?

$$f_N(x) = \sum_{i=0}^{N} a_i x^i = \alpha \prod_{i=1}^{N} (x - x_i)$$

factored form

x_i : zeros

- a polynomial of **Nth order** has **N roots** (real & complex)
- for N $\geq$ 5: no analytical solutions
- for N is odd: at least one real zero



zeros: $x_1 = x_2 = 1$

one zero with multiplicity m = 2

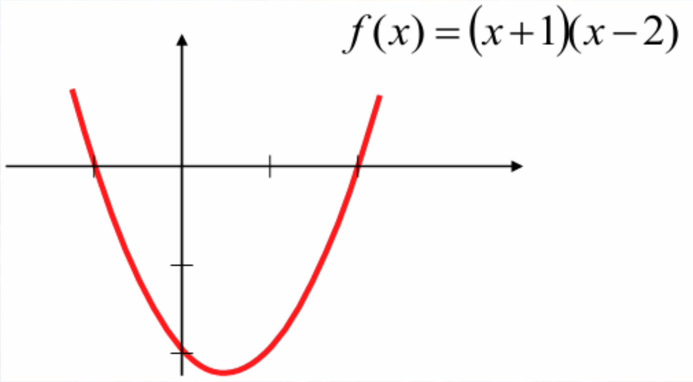
How many roots does a polynomial have?

$$f_N(x) = \sum_{i=0}^{N} a_i x^i = \alpha \prod_{i=1}^{N} (x - x_i)$$

factored form

x_i : zeros

- a polynomial of **Nth order** has **N roots** (real & complex)
- for N $\geq$ 5: no analytical solutions
- for N is odd: at least one real zero



zeros: $x_1 = 2, \; x_2 = -1$

two zeros with multiplicity m = 1 each

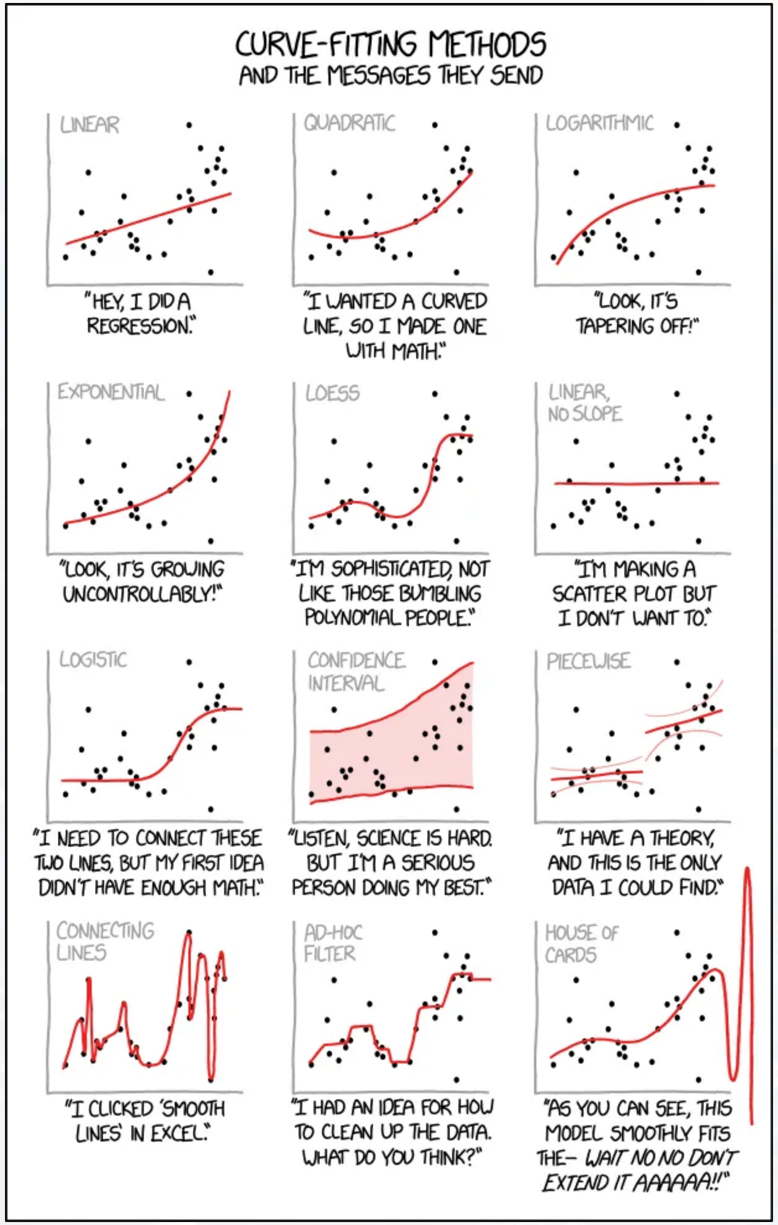
methods:

Root finding [edit]

Main article: Root-finding algorithm

- Bisection method
- False position method: and Illinois method: 2-point, bracketing
- Halley's method: uses first and second derivatives
- ITP method: minmax optimal and superlinear convergence simultaneously
- Muller's method: 3-point, quadratic interpolation
- Newton's method: finds zeros of functions with calculus
- Ridder's method: 3-point, exponential scaling
- Secant method: 2-point, 1-sided

$$\boxed{x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}}$$



Outline

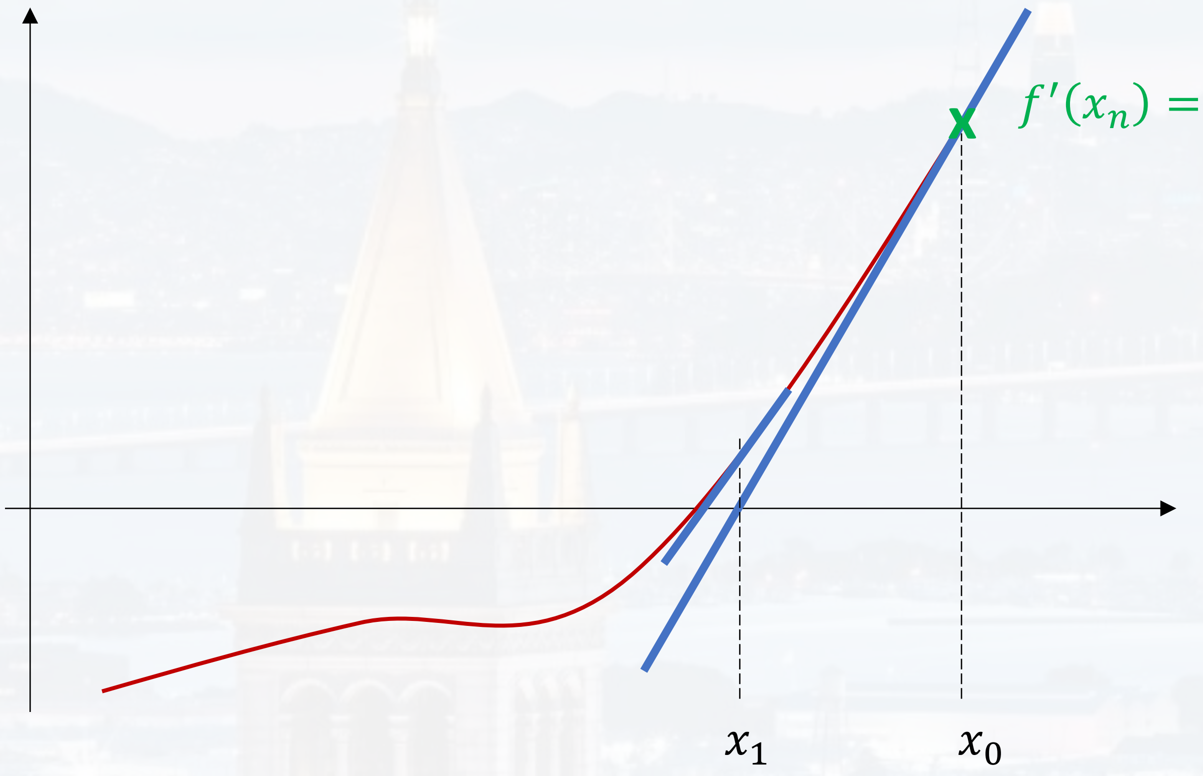
- The Problem

- **Newtons Method**

- Bisection

Newton’s method:

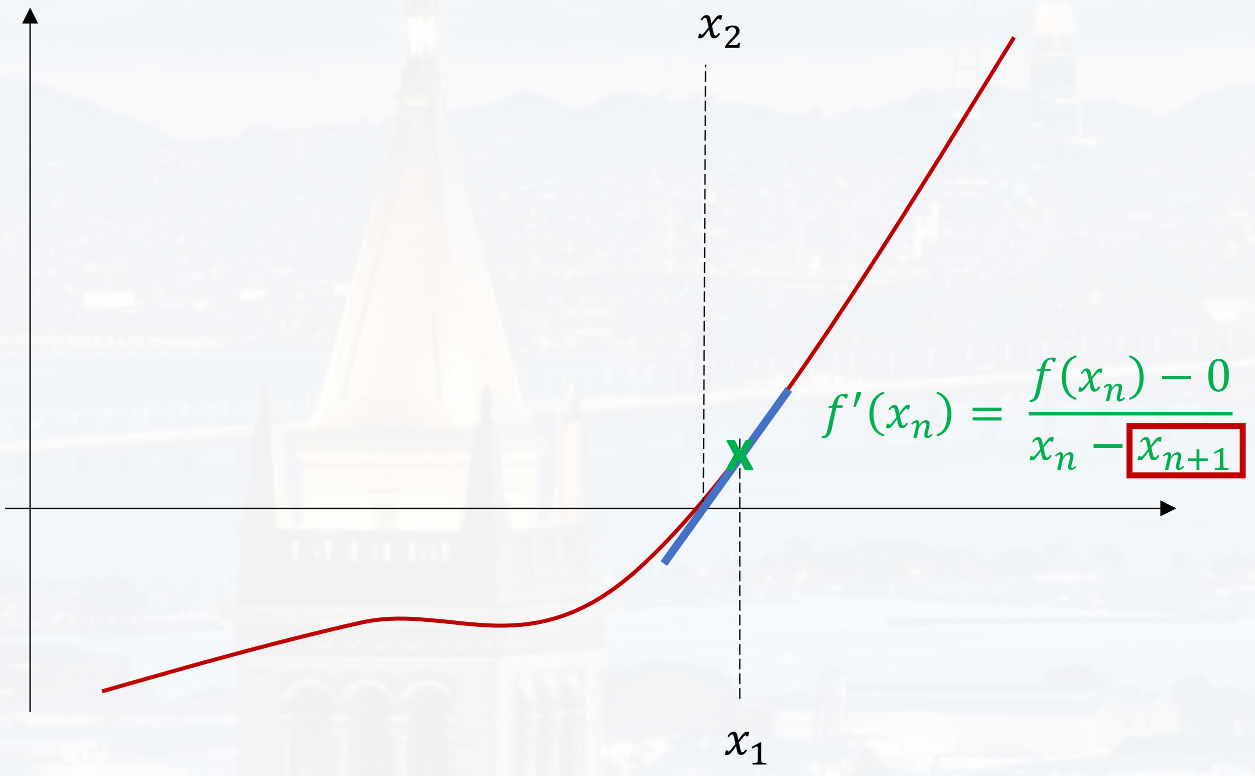
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$



$$f'(x_n) = \frac{f(x_n) - 0}{x_n - \boxed{x_{n+1}}}$$

Newton's method:

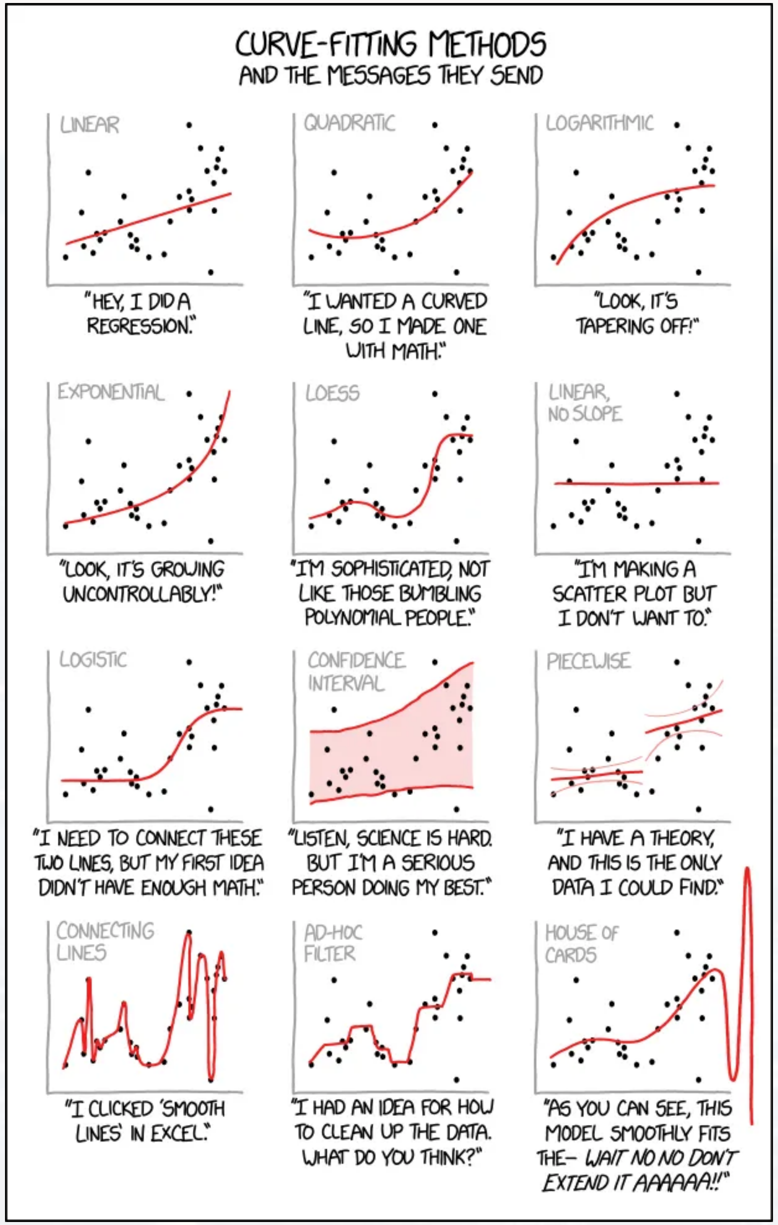
$$\boxed{x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}}$$



Newton's method:

$$\boxed{x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}}$$

- since slope of the function points to next $\mathbf{x_{n+1}}$ → converges quadratically
- needs derivative → evaluation numerically
- convergence depends on initial guess → might not converge!



Outline

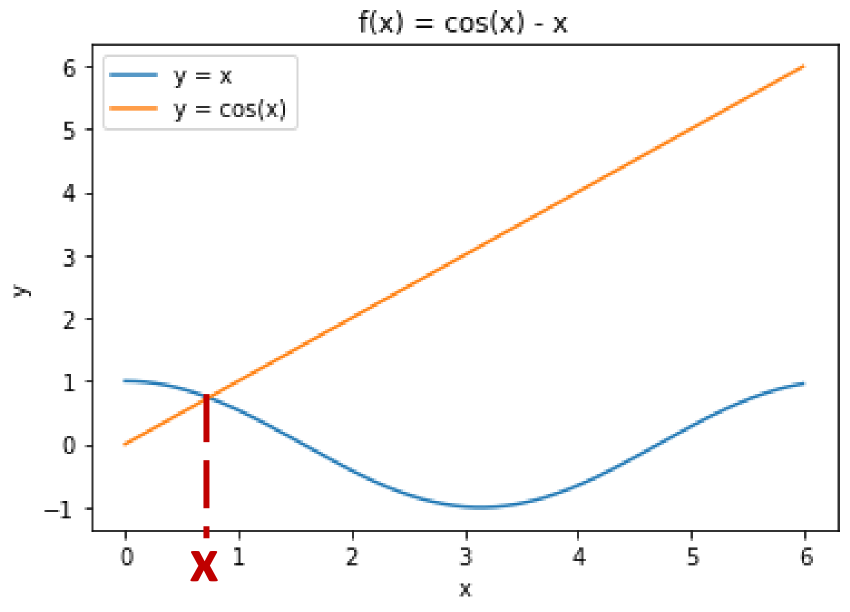
- The Problem
- Newtons Method
- **Bisection**

methods:

Root finding [edit]

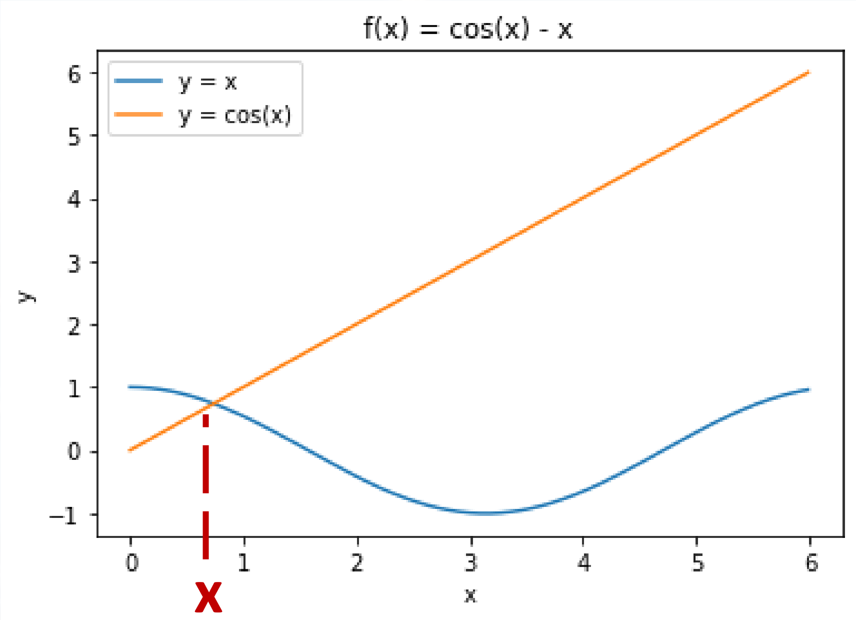
Main article: Root-finding algorithm

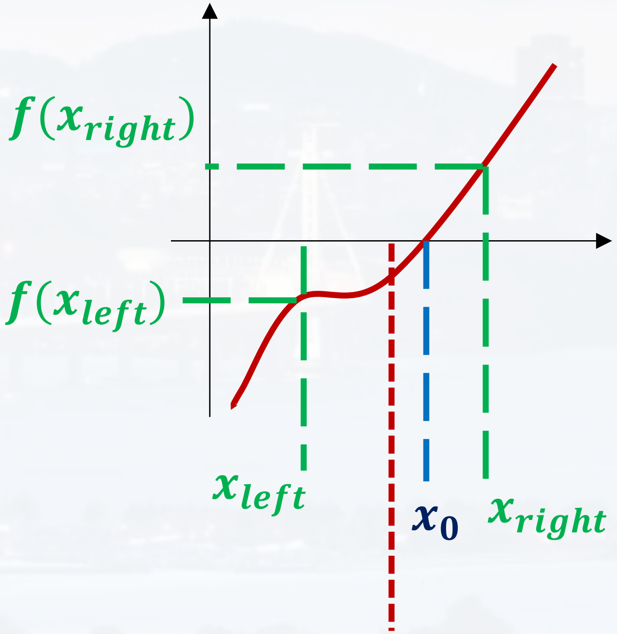
- Bisection method
- False position method: and Illinois method: 2-point, bracketing
- Halley's method: uses first and second derivatives
- ITP method: minmax optimal and superlinear convergence simultaneously
- Muller's method: 3-point, quadratic interpolation
- Newton's method: finds zeros of functions with calculus
- Ridder's method: 3-point, exponential scaling
- Secant method: 2-point, 1-sided



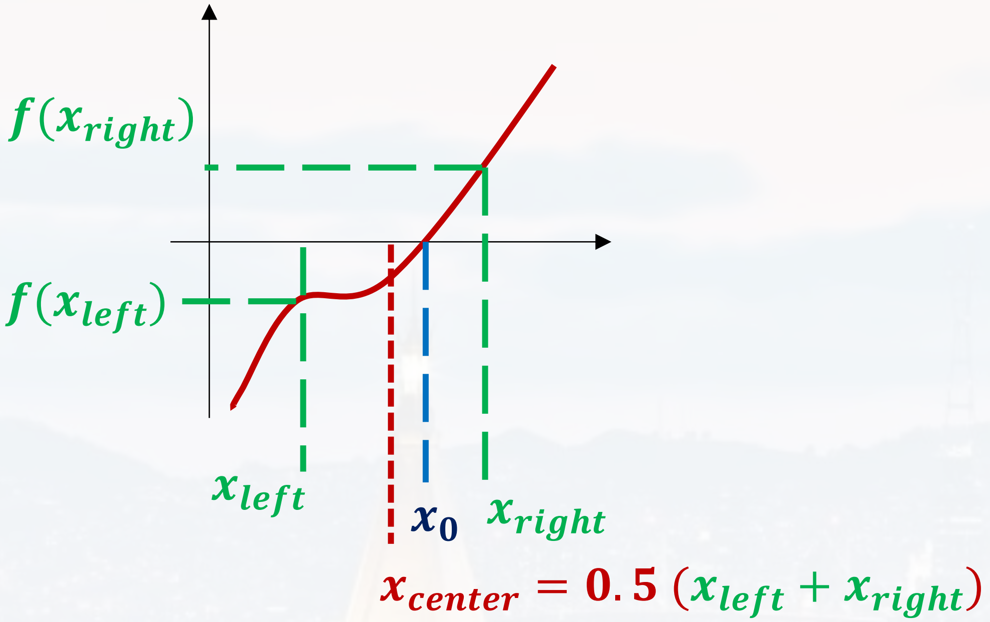
Bisection:

<u>assumption:</u> root is within interval $[x_{left}, x_{right}]$



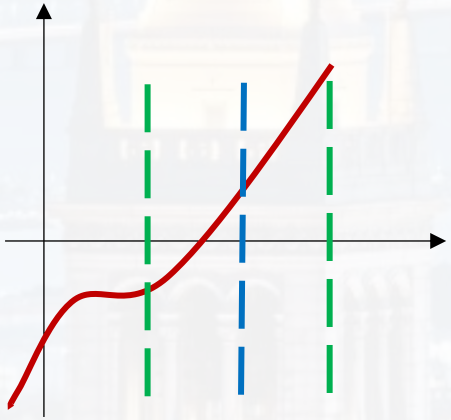


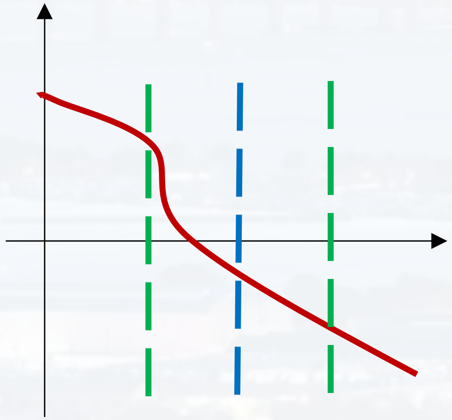
$x_{center} = 0.5\,(x_{left} + x_{right})$



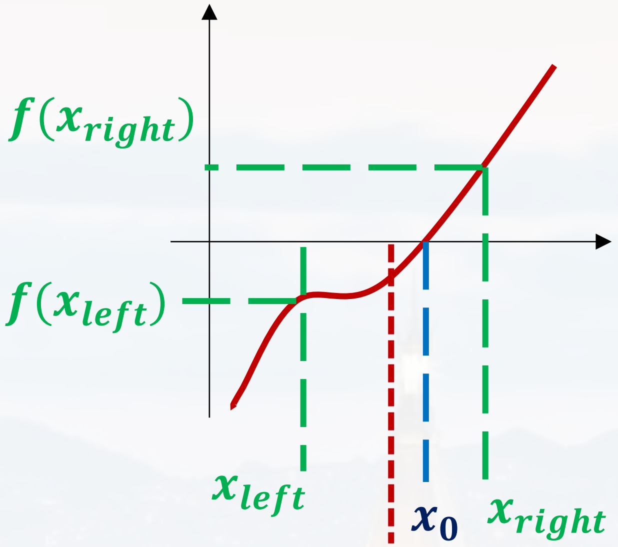
if $f(x_{center}) \cdot f(x_{left}) < 0$

- $x_{left} \rightarrow x_{left}$
- set x_{right} to x_{center}
- reset $x_{center} = 0.5\,(x_{left} + x_{right})$





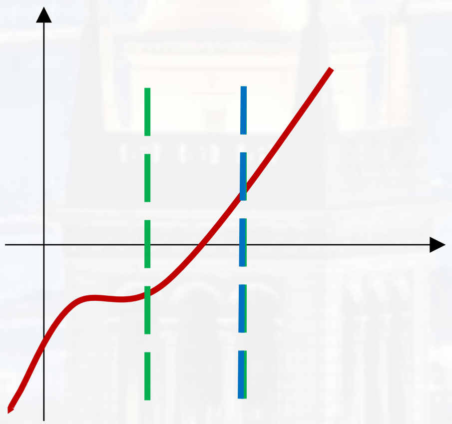
Solving Nonlinear Equations

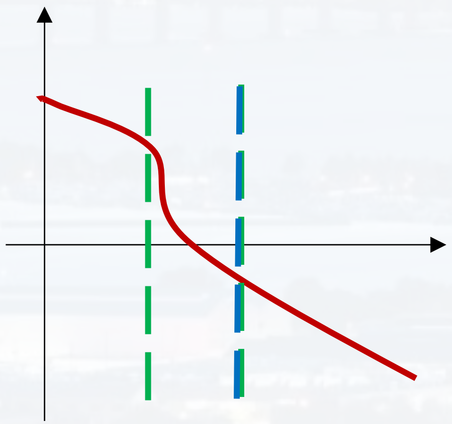


$x_{center} = \mathbf{0.5}\,(x_{left} + x_{right})$

if $f(x_{center}) \cdot f(x_{left}) < 0$

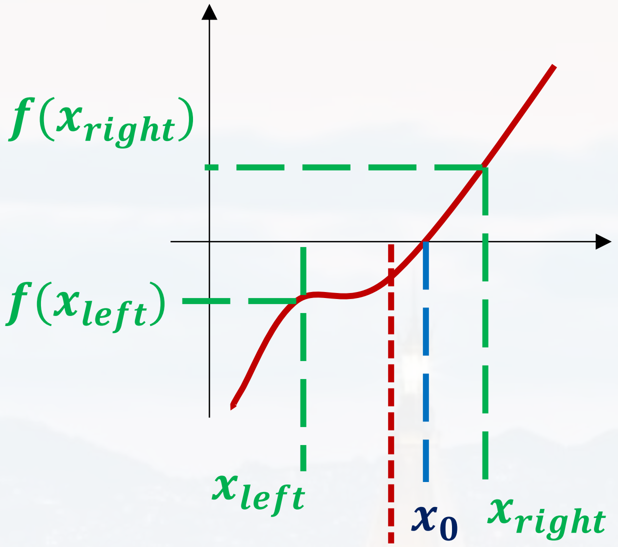
- $x_{left} \rightarrow x_{left}$
- set x_{right} to x_{center}
- reset $x_{center} = \mathbf{0.5}\,(x_{left} + x_{right})$





Solving Nonlinear Equations

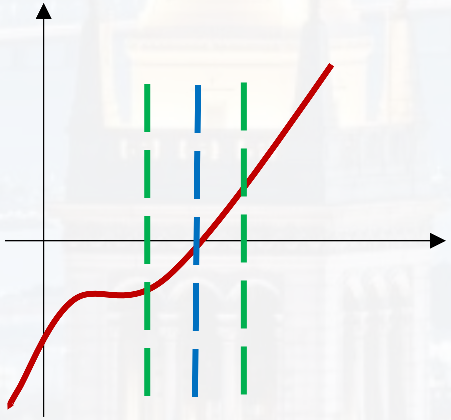
Bisection

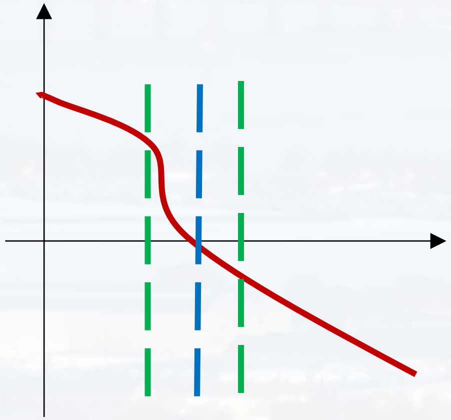


$x_{center} = 0.5\,(x_{left} + x_{right})$

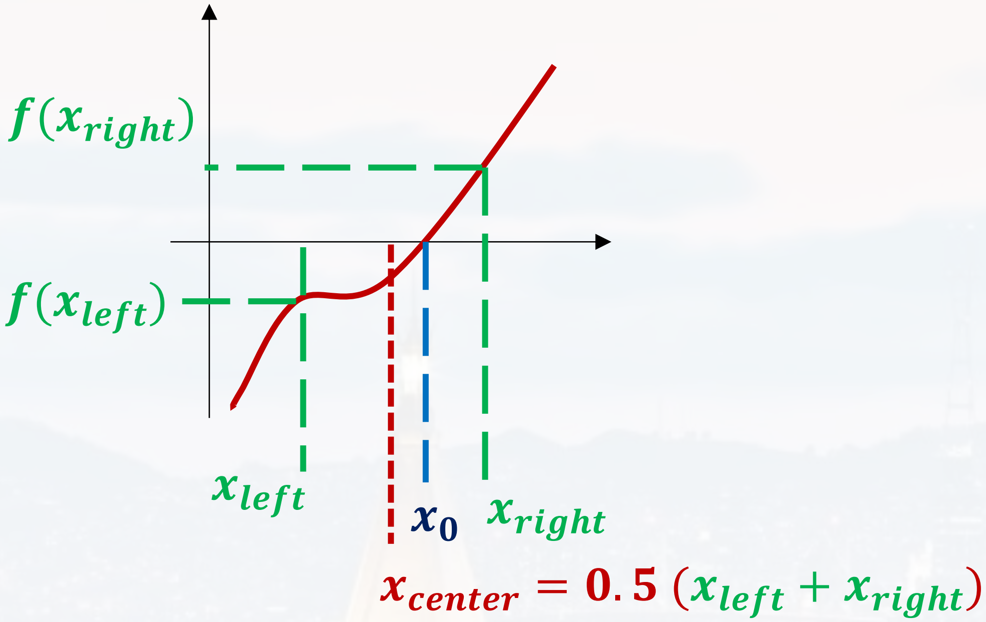
if $f(x_{center}) \cdot f(x_{left}) < 0$

- $x_{left} \rightarrow x_{left}$
- set x_{right} to x_{center}
- reset $x_{center} = 0.5\,(x_{left} + x_{right})$



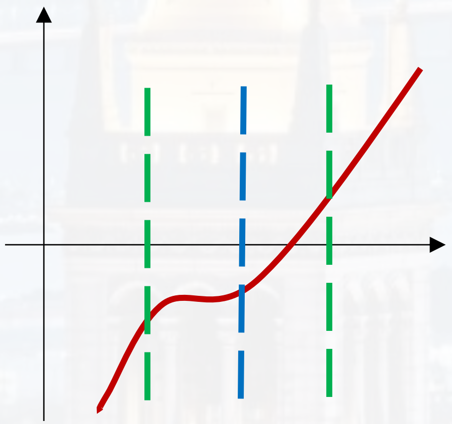


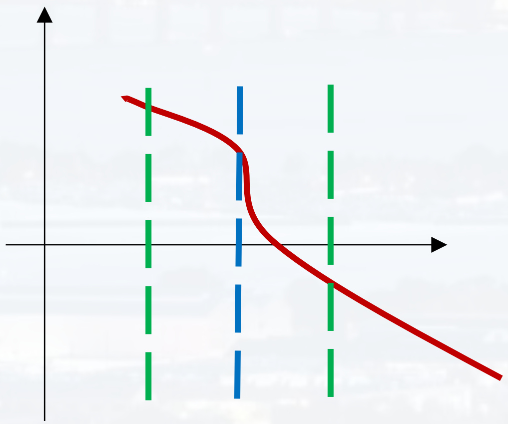
either we end up with the same situation, or…



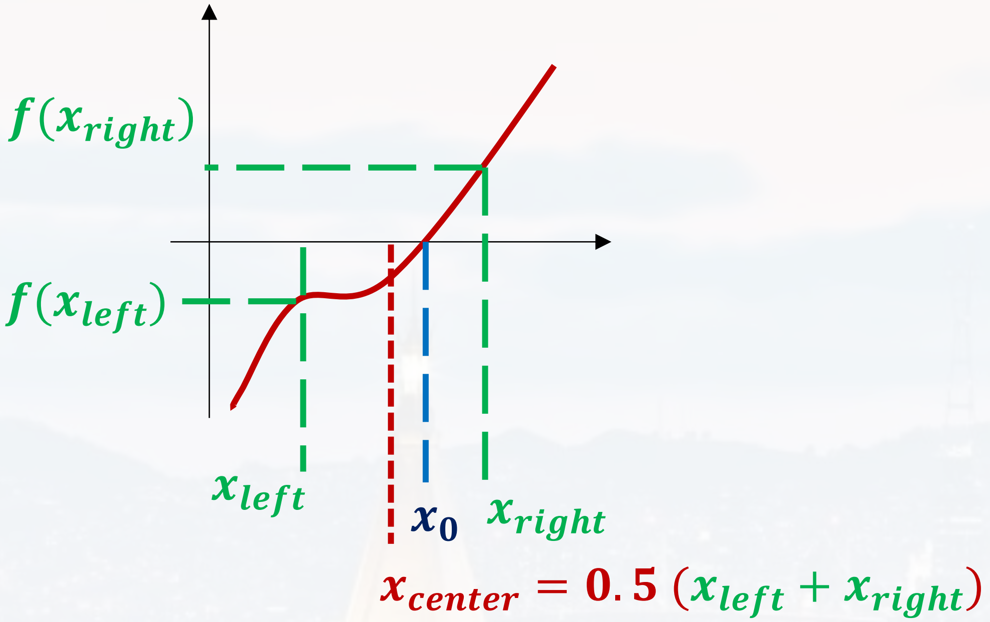
if $f(x_{center}) \cdot f(x_{left}) > 0$

- set x_{left} to x_{center}
- $x_{right} \rightarrow x_{right}$
- reset $x_{center} = 0.5\,(x_{left} + x_{right})$



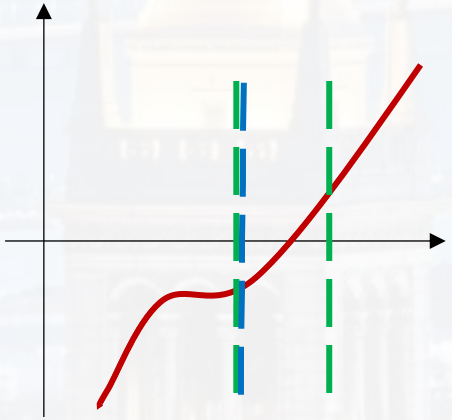


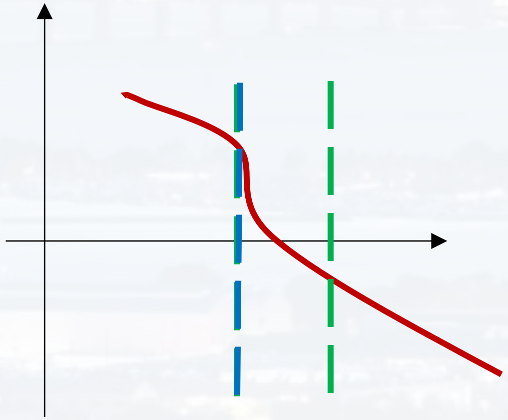
Solving Nonlinear Equations

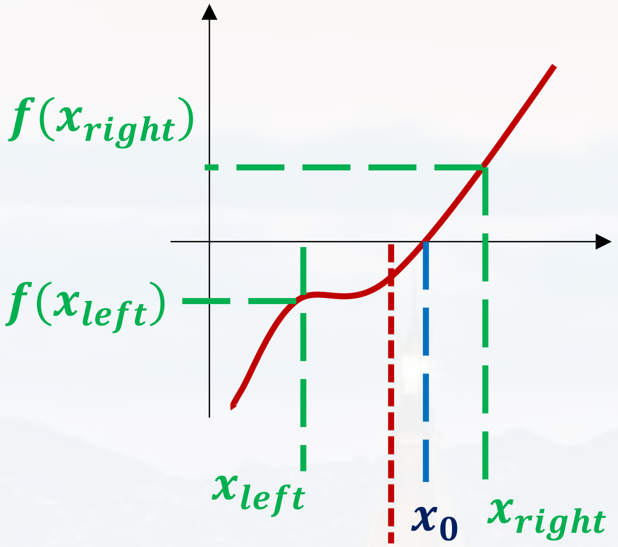


if $f(x_{center}) \cdot f(x_{left}) > 0$

- set x_{left} to x_{center}
- $x_{right} \rightarrow x_{right}$
- reset $x_{center} = \mathbf{0.5}\,(x_{left} + x_{right})$



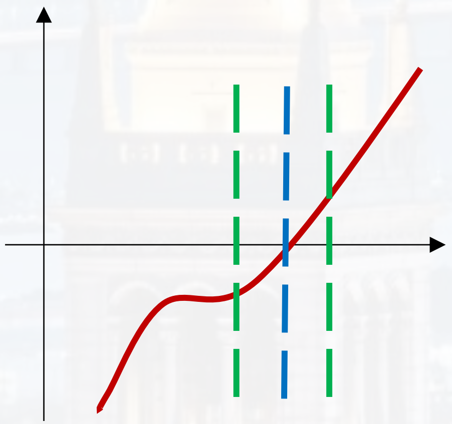


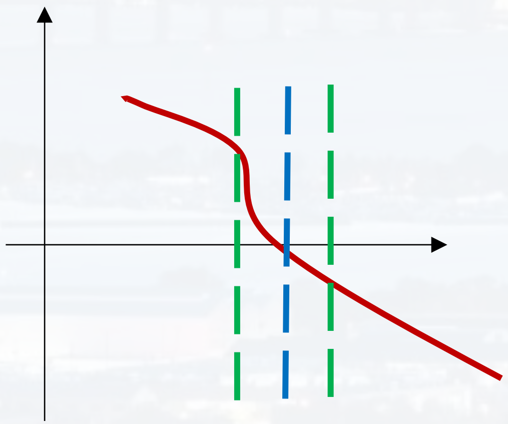


$x_{center} = 0.5\,(x_{left} + x_{right})$

if $f(x_{center}) \cdot f(x_{left}) > 0$

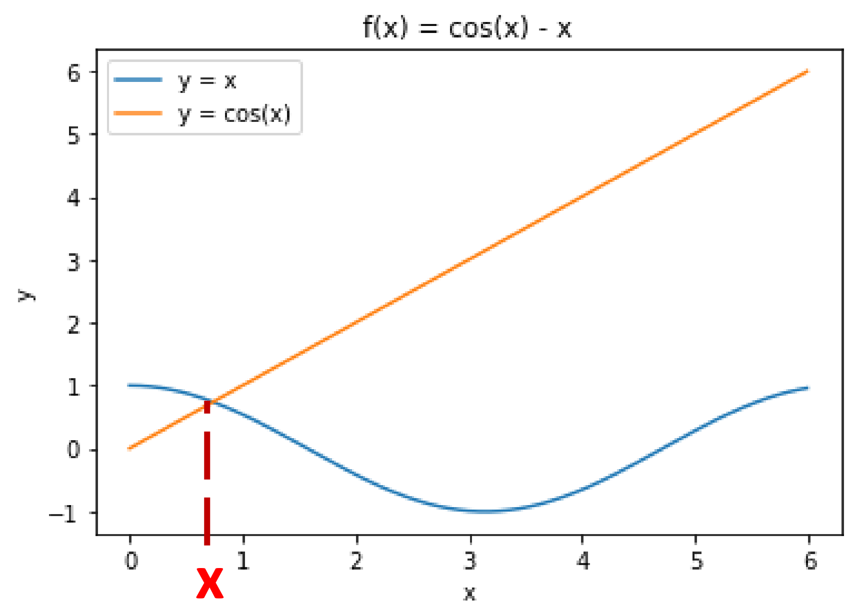
- set x_{left} to x_{center}
- $x_{right} \rightarrow x_{right}$
- reset $x_{center} = 0.5\,(x_{left} + x_{right})$

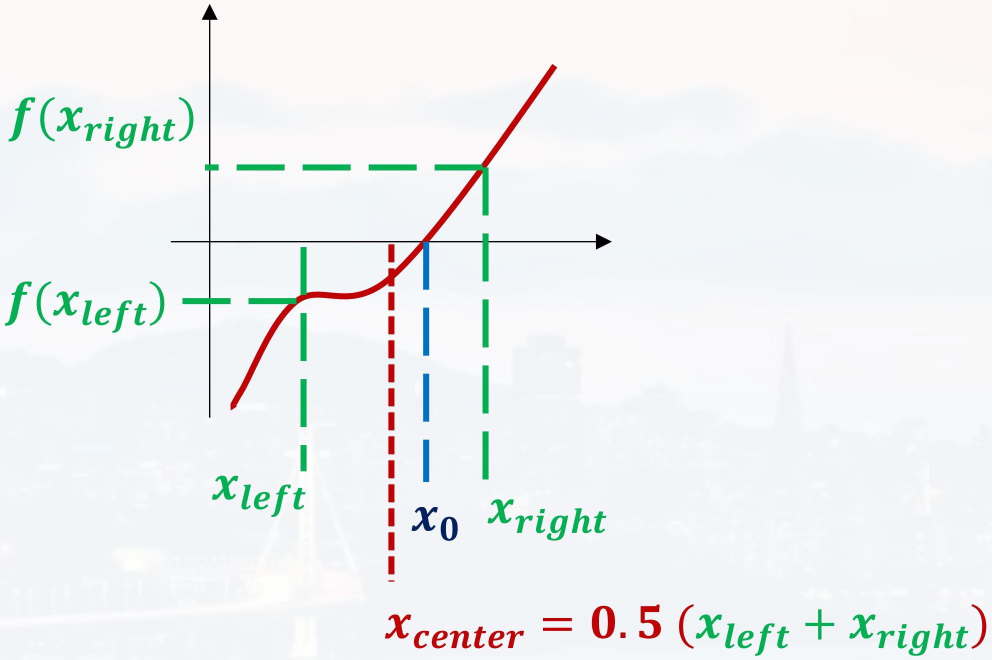




…and so on…

Bisection:





- robust: always finds a root
- easy to implement (recursion), → Lecture Exercise
- slow: converges linearly **(accuracy increases by factor of 2 for each step *n*)** with *n* required for a certain accuracy

Thank you for your attention!