Computational Quantum Mechanics

Prof. Gerson J. Ferreira

INFIS/UFU 2020/1

Root finding algorithms

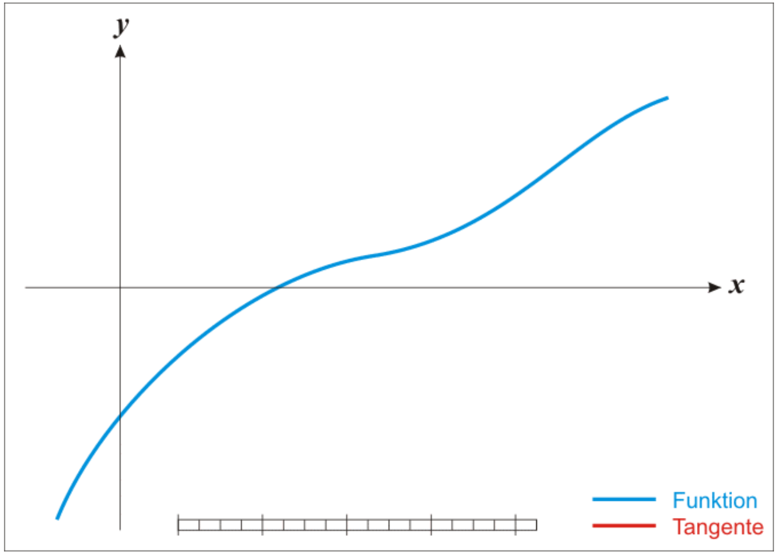
Outline

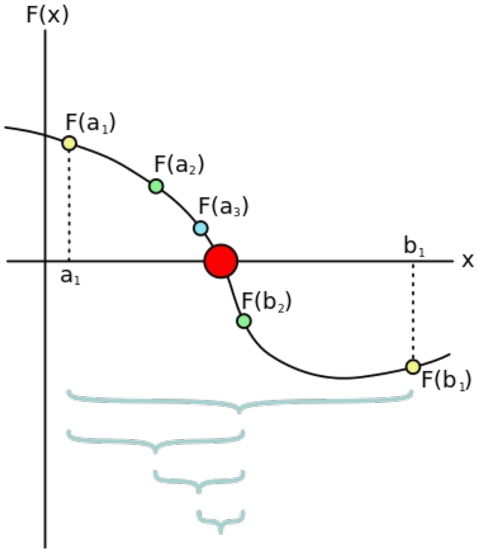
Root finding algorithms → **$\sin(x) = \alpha x$, $x = ?$**

→ Newton's method and the Secant method
→ Bisection method
→ … more at the [Wikipedia] and the books in the class bibliography

Why?

→ We'll need one of these to solve Schrödinger's equation with shooting methods





[images from Wikipedia]

Newton's method

We want to find x such that $f(x) = 0$.
Let's guess that there's a solution at $x=x_0$, and expand $f(x)$ for x near x_0.

$$\underbrace{f(x) \approx f(x_0) + (x - x_0)f'(x_0)}_{} + \cancel{\frac{(x-x_0)^2}{2!}f''(x_0)} + \cdots$$

A better estimate for the root might be...

$$x \rightarrow x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Now repeat the expansion for x near x_1, and iterate to get

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

This method assumes you have an analytical expression for the derivative $f'(x)$

The secant method

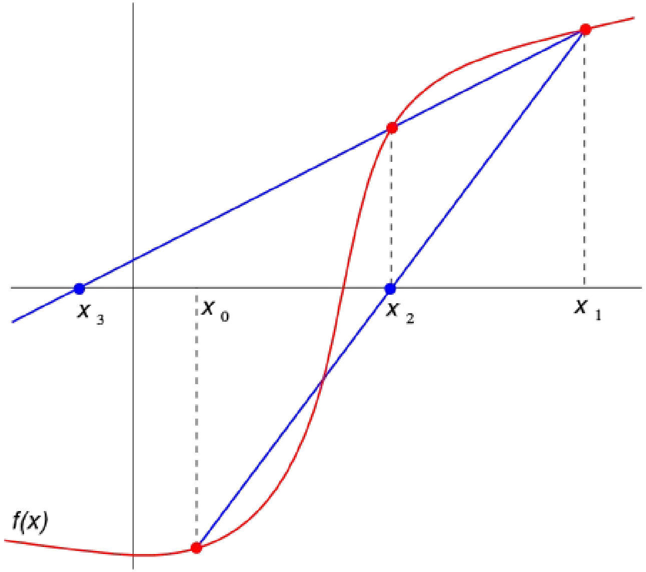
Starting with Newton's method

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

… but not let's assume we don't have *f'(x)*

Replace *f'(x)* with a finite differences derivative

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



Derivation at [Wikipedia]

The bisection method

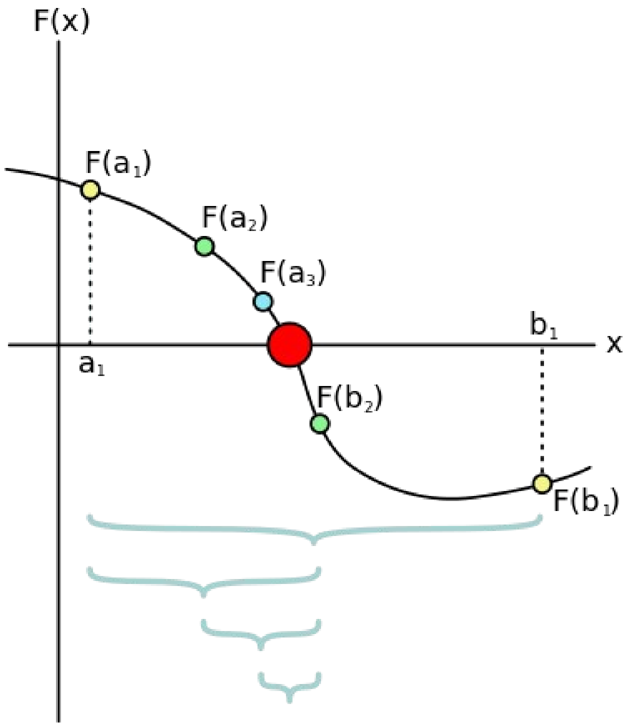
If there's a (single) root $F(x) = 0$ in the interval $a < x < b$,

$\rightarrow F(a)\,F(b) < 0$

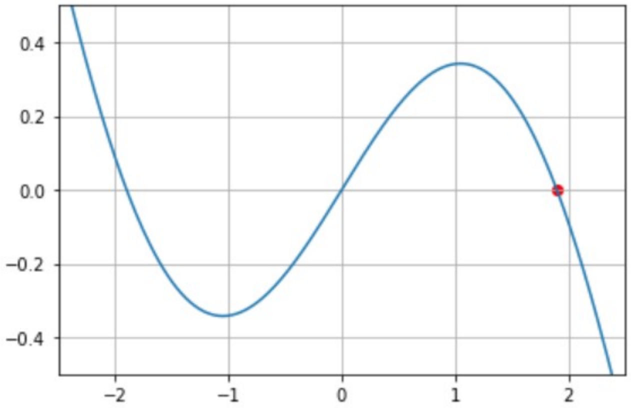
1. Take the midpoint $c = (a+b)/2$.

2. If $F(a)F(c) < 0$, switch to the interval $a < x < c$.
   If $F(c)F(b) < 0$, switch to the interval $c < x < b$.

3. Repeat until convergence (interval small enough)



Examples and implementation at the course webpage



Suggestion, use Scipy:

from scipy import optimize
→ newton
→ bisect

The bisection method

```
def bisect(a, b, f, eps=1e-6, maxsteps=1000):
    '''
    Solves f(x)=x for x using the bisection method.

    INPUT:
        a,b: interval a<x<b to search the root
        f: callable function f(x)
        eps: required precision for the interval [default 1e-6]
        maxsteps: maximum number of bisections [default 1000]

    OUTPUT:
        tuple: root, estimate of error, number of steps

    '''
```

The Secant method

```
def secant(a, b, f, eps=1e-6, maxsteps=1000):
    '''
    Solves f(x)=0 for x using the secant method.

    INPUT:
        a,b: initial guesses for x near the desired root
        f: callable function f(x)
        eps: required precision for |f(x)|<eps
        maxsteps: maximum number of steps

    OUTPUT:
        tuple: root and number of steps

    '''
```