https://gitlab.com/gersonjferreira/computational-quantum-mechanics

Computational Quantum Mechanics

Prof. Gerson J. Ferreira

INFIS/UFU 2020/1

The shooting method

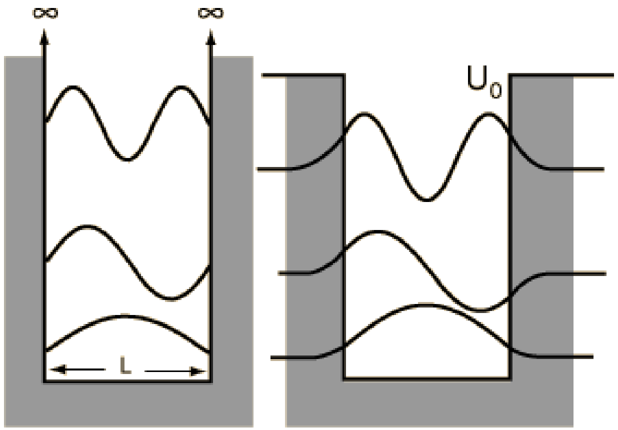
The problem

Goal:

→ Solve the 1D time-independent Schrödinger equation with the **shooting method**

$$H\psi(x) = E\psi(x)$$

$$H = \frac{p^2}{2m} + V(x)$$



$$\left.\begin{aligned} \psi(0) &= 0 \\ \psi(L) &= 0 \end{aligned}\right\}$$ Boundary value problem (BVP)

→ but the shooting method
is for initial value problems (IVP)

The trick

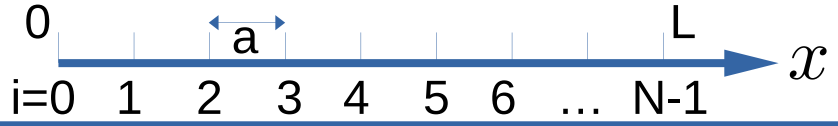
→ write the Schrödinger equation as an IVP and try to find a solution that satisfies the BVP

$$\left.\begin{aligned} H\psi(x) &= E\psi(x) \\ H &= \frac{p^2}{2m} + V(x) \\ \psi(0) &= 0 \\ \psi'(0) &= s \end{aligned}\right\}$$

1. Choose an guess for E
2. Use Euler / RK4 / Numerov to propagate from 0 to L
3. Check if $|\psi(L)| < \varepsilon$, assuming small, e.g. $\varepsilon \sim 10^{-6}$
4. If not, guess E again and loop
   → how to guess E? Bisection, Newton, Numerov-Cooley

How to choose *s*? Any choice of $s \neq 0$ is valid,
since it will be renormalized at the end.

The propagation from 0 to L:



$$x = ia$$
$$i = 0 \ldots N-1$$
$$a = L/(N-1)$$

The 2nd derivative in finite differences with step size *a* is

$$\frac{\partial^2}{\partial x^2}\psi(x) \approx \frac{\psi(x-a) - 2\psi(x) + \psi(x+a)}{a^2} = \frac{\psi_{i-1} - 2\psi_i + \psi_{i+1}}{a^2}$$

Apply to the Schrödinger equation and isolate ψ_{i+1}

$$\psi_{i+1} = -\psi_{i-1} + \left[2 + \frac{2ma^2}{\hbar^2}(V_i - E)\right]\psi_i$$

Use the initial conditions from the IVP:

$$\psi_0 = 0$$

$$\psi_1 = sa$$ $\leftarrow$ got this one using the forward difference

Implementation

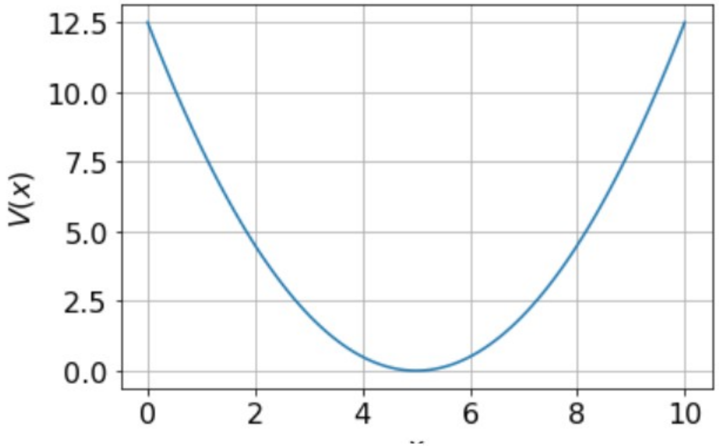
Let's consider $\hbar = m = 1$

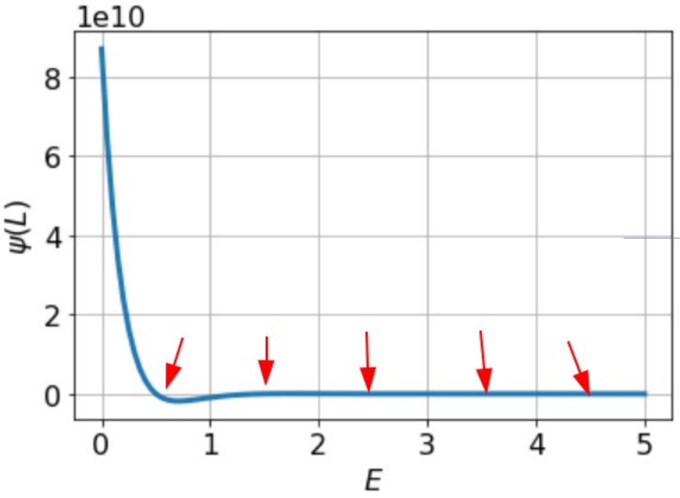
→ $L = 10$, $x = [0, L]$, discretized with $N = 100$ points

→ $V(x) = ½\,(x\text{-}L/2)^2$

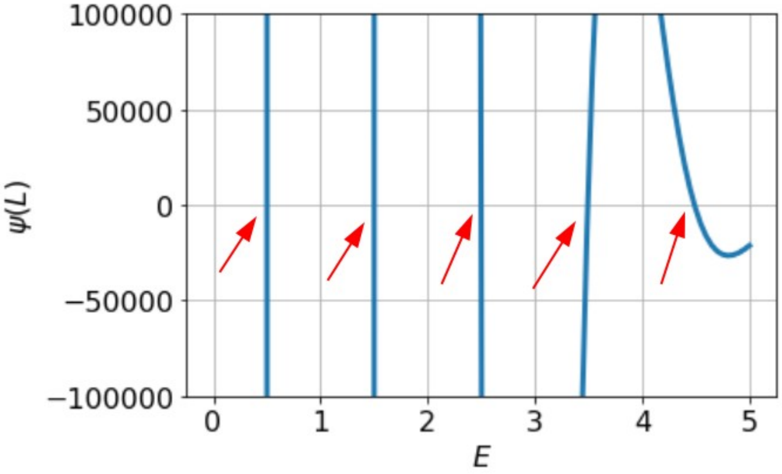
We expect eigenenergies at $E = (n+1/2)\hbar\omega$, with $\hbar\omega = 1$

For each E, propagate the solution and plot $\psi(L)$ vs E





zoom



Using Newton's method

```python
roots = []
for i in range(10):
    e0 = (i+0.5)
    root = optimize.newton(lambda E: shoot(x, v, E)[-1], e0)
    roots.append(root)
roots = np.array(roots)
np.round(roots, 3)
```

```
array([0.5  , 1.498, 2.496, 3.492, 4.487, 5.481, 6.473, 7.467, 8.464,
```