Harmonic Oscillator

The harmonic oscillator is certainly one of the working horses of physics. In quantum mechanics, it resembles to be a good model for the motion of two bound atoms, i.e. bond vibrations, which are of relevance for al types of molecules and solids.

```
#| edit: false
#| echo: false
#| execute: true

import numpy as np
from scipy.sparse import diags
from scipy.sparse.linalg import eigsh
import matplotlib.pyplot as plt

# Set default plotting parameters
plt.rcParams.update({
    'font.size': 12,
    'lines.linewidth': 1,
    'lines.markersize': 5,
    'axes.labelsize': 11,
    'xtick.labelsize': 10,
    'ytick.labelsize': 10,
    'xtick.top': True,
    'xtick.direction': 'in',
    'ytick.right': True,
    'ytick.direction': 'in',
})

def get_size(w, h):
    return (w/2.54, h/2.54)
```

As compared to the particle in a box, we have to change the potential in the Hamilton operator to solve the harmonic oscillator. The potential energy of the harmonic oscillator is given as

$$V(x) = \frac{k}{2}x^2 \tag{1}$$

where k is the spring constant and x is the deviation from its minimum potential energy position. For an atomic bond between carbon and oxygen, for example, the spring constant corresponds to $k = 396$ N/m.

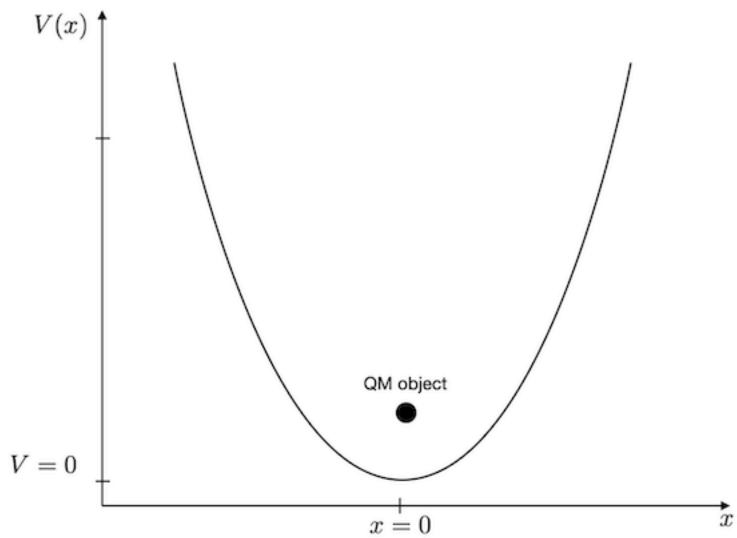

Figure 1: Harmonic Potential

The C=O bond has a bond length of $x_0 = 1.229\mathring{A}$, so we do not have to look at a large domain. A region of $L = 1\mathring{A}$ provides already an energy change by 3 eV.

Definition of the problem

Before we start, we need to define some quantities:

- we will study a domain of L=1 Angström
- we will use N=1001 points for our x_i
- the spring constant shall be $k = 396$ N/m
- we will use the mass of the carbon atom

```
#| autorun: false
# define some useful constants

hbar=1.678e-35 # joule seconds
m_c=1.998467052e-26 # carbon atom mass in kg
m_o=2.657e-26 # oxygen mass in kg
m=m_c*m_o/(m_c+m_o)
N=1001
k=396  # spring constant of the C=O bond N/m

L= 0.5e-10 #m

x = np.linspace(-L/2, L/2, N)
dx = x[1] - x[0]
```

Potential energy

We first define the diagonal potential energy matrix.

```
#| autorun: false

# potential energy for the harmonic oscillator
U_vec = 0.5*k*x**2

# potential energy is only on the diagonal, no deritvative
U = diags([U_vec], [0])
```

Kinetic energy

Next are the derivatives of the kinetic energy matrix.

```
#| autorun: false

# T is the finite difference2 representation of the second derivative in the kinetic energy
T = -hbar**2*diags([-2., 1., 1.], [0,-1, 1], shape=(N, N))/dx**2/2/m
```

An finally the total Hamilton operator matrix again.

```
#| autorun: false

# Sum of kinetic and potential energy
H = T + U
```

Solution

The last step is to solve the system of coupled equations using the `eigsh` method of the `scipy` module again.

```
#| autorun: false

# diagonalize the matrix and take the first n eigenvalues and eigenvectors
n=10

vals, vecs = eigsh(H, k=n, which='SM')
```

Plotting

```
#| autorun: false

# define some scaling to make a nice plot
scale=1e9 # position scale
escale=6.242e18 # energy scale in eV
psiscale=1 # wavefunction scale

plt.figure(figsize=get_size(10,8))

for i in range(n):
    vec = vecs[:, i]
    mag = np.sqrt(np.dot(vecs[:, i],vecs[:, i]))
    vec = vec/mag
    plt.axhline(y=vals[i]*escale)
    plt.plot(x*scale, psiscale*np.abs(vec)**2+vals[i]*escale)

plt.plot(x*scale,U_vec*escale,'--')
plt.xlabel(r"position $x$ [$\mathring{A}$]")
plt.ylabel(r"energy $E$ in eV, Wavefunction $\Psi(x)$")
plt.axhline(y=0.026,color='k',ls='--',lw=2)
```

```
plt.ylim(0,0.2)

plt.tight_layout()
plt.show()
```

Analytical Solution

Theory

The quantum harmonic oscillator is one of the few quantum systems that can be solved exactly analytically. The energy eigenvalues are given by:

$$E_n = \left(n + \frac{1}{2}\right)\hbar\omega \tag{2}$$

where $n = 0, 1, 2, ...$ is the quantum number and $\omega = \sqrt{k/m}$ is the angular frequency of the oscillator.
The corresponding wavefunctions are:

$$\psi_n(x) = \left(\frac{\alpha}{\pi}\right)^{1/4} \frac{1}{\sqrt{2^n n!}} H_n(\sqrt{\alpha}x) e^{-\alpha x^2/2} \tag{3}$$

where $\alpha = \frac{m\omega}{\hbar}$ is a parameter that determines the width of the wavefunction, and H_n are the Hermite polynomials. The ground state ($n = 0$) has a Gaussian shape, while higher states show increasingly complex oscillatory behavior with exactly n nodes.

Implementation

Below we plot the analytical results:

```python
#| autorun: false

# Plot analytical wavefunctions of the harmonic oscillator
plt.figure(figsize=get_size(10,8))


# Angular frequency
omega = np.sqrt(k/m)
psiscale=5e-14

# Function to calculate analytical wavefunction
def analytical_psi(n, x, m, omega, hbar):
    """Calculate the analytical wavefunction for a harmonic oscillator"""
    # Characteristic length parameter
    alpha = m * omega / hbar

    # Normalization constant
    from scipy.special import factorial
    norm = (alpha/np.pi)**(1/4) / np.sqrt(2**n * factorial(n))

    # Hermite polynomial
    from scipy.special import eval_hermite

    # Calculate wavefunction
    psi = norm * np.exp(-alpha * x**2 / 2) * eval_hermite(n, np.sqrt(alpha) * x)

    return psi

# Plot potential energy
plt.plot(x*scale, U_vec*escale, '--', label='Potential')

# Plot analytical wavefunctions for n=0 to n=9
for n in range(10):
    # Energy level
    E_n = (n + 0.5) * hbar * omega * escale

    # Wavefunction
    psi = analytical_psi(n, x, m, omega, hbar)

    # Plot probability density (scaled and shifted to energy level)
    plt.plot(x*scale, psiscale*np.abs(psi)**2 + E_n)

    # Plot energy level
    plt.axhline(y=E_n)

plt.xlabel(r"position $x$ [$\mathring{A}$]")
plt.ylabel(r"energy $E$ in eV, Wavefunction $|\Psi(x)|^2$")
plt.axhline(y=0.026, color='k', ls='--', lw=2)
plt.ylim(0, 0.2)

plt.tight_layout()
plt.show()
```

Energies of the states

```
#| autorun: false

fig = plt.figure(figsize=(6,7))
plt.ylabel(r"$E$ [eV]")
for i in range(n):
    plt.axhline(y=vals[i]*escale)
    plt.scatter([0],(i+0.5)*hbar*np.sqrt(k/m)*escale)

plt.xticks([])
plt.show()
```

Where to go from here?

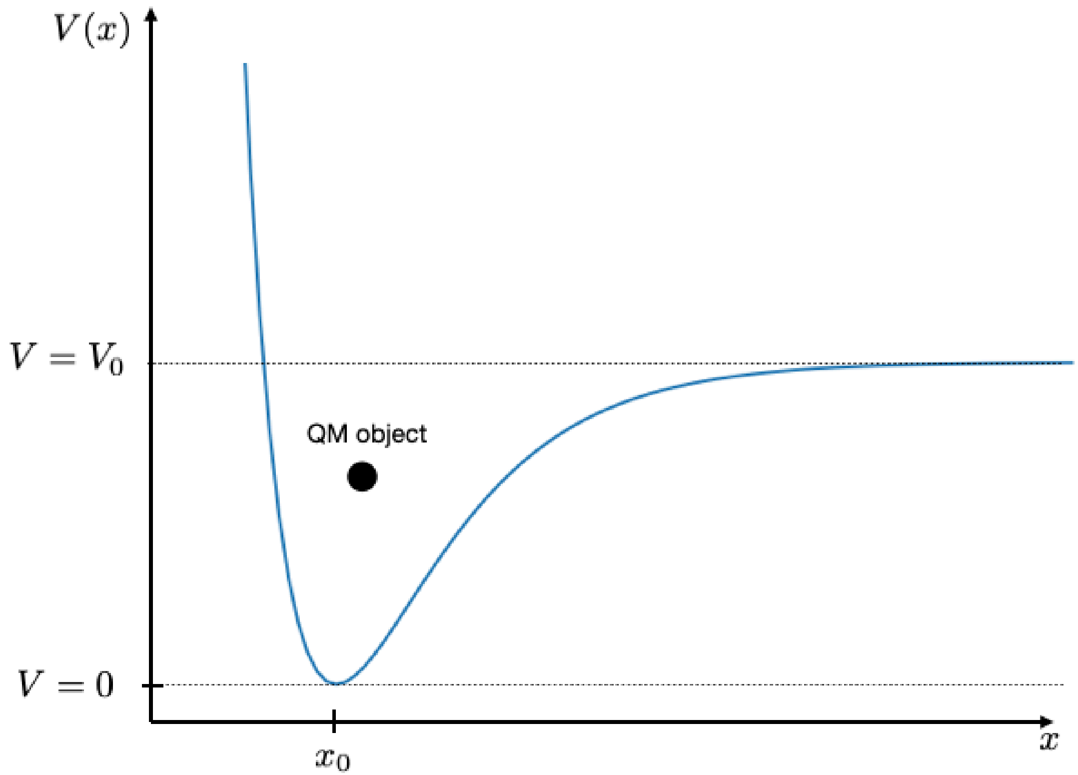

Figure 2: Morse Potential

Where to go from here?

2.
3.

4.