032CM - 2025

PROGRAMMING FOR COMPUTATIONAL CHEMISTRY

ASE and NGLview

Gianluca Levi

gianluca.levi@units.it, giale@hi.is

Office: Building C11, 3rd floor, Room 329

Fall 2025

Atomic Simulation Environment (ASE)

https://ase-lib.org/index.html

Python library for **setting up, manipulating, running, visualizing and analyzing atomic-scale simulations**

Atomic Simulation Environment (ASE)

https://ase-lib.org/index.html

Python library for **setting up, manipulating, running, visualizing and analyzing atomic-scale simulations**

Interfaces to many large simulation programs (via calculators)

- GPAW, Quantum ESPRESSO, VASP, CP2K, CRYSTAL, ORCA, LAMMPS, Gromacs, and more…

Atomic Simulation Environment (ASE)

https://ase-lib.org/index.html

Python library for **setting up, manipulating, running, visualizing and analyzing atomic-scale simulations**

Interfaces to many large simulation programs (via calculators)

- GPAW, Quantum ESPRESSO, VASP, CP2K, CRYSTAL, ORCA, LAMMPS, Gromacs, and more...

Core capabilities

- Structure building (molecules, crystals, surfaces, nanoparticles), file I/O
- Geometry optimization
- Molecular dynamics
- Vibrational analysis (normal modes)

Atomic Simulation Environment (ASE)

https://ase-lib.org/index.html

Python library for **setting up, manipulating, running, visualizing and analyzing atomic-scale simulations**

Interfaces to many large simulation programs (via calculators)

- GPAW, Quantum ESPRESSO, VASP, CP2K, CRYSTAL, ORCA, LAMMPS, Gromacs, and more...

Core capabilities

- Structure building (molecules, crystals, surfaces, nanoparticles), file I/O
- Geometry optimization
- Molecular dynamics
- Vibrational analysis (normal modes)

Freely available (https://gitlab.com/ase/ase)

Atomic Simulation Environment (ASE)

```
atoms = Atoms('OCO',
          positions=[[-1.342,  0.,  0.],
                     [-0.28 ,  0.,  0.],
                     [ 1.372,  0.,  0.]])
```

Create **Atoms object**

```
atoms.get_positions()

atoms.set_positions(...)

atoms.get_distances(i, j)

atoms.get_center_of_mass()

    ...
```

```
traj = Trajectory('co2.traj', 'w')

for i in range(N):

   atoms.set_positions(...)

   traj.write(atoms)

traj.close()
```

Create **trajectory of atoms**

NGLview

https://nglviewer.org/nglview/latest/#

https://projects.volkamerlab.org/teachopencadd/talktorials/T017_advanced_nglview_usage.html

Jupyter tool to **interactively view molecular structures and trajectories of atoms**

NGLview

https://nglviewer.org/nglview/latest/#

https://projects.volkamerlab.org/teachopencadd/talktorials/T017_advanced_nglview_usage.html

Jupyter tool to **interactively view molecular structures and trajectories of atoms**

- **Java library** for visualizing 3D structures
- Works with Python libraries, (**ASE,** MDAnalysis, MDTraj, pytraj)
- Multiple **3D representations** (ball-and-stick, licorice, spacefill, etc.)
- **Animate trajectories of atoms**
- Great for **macromolecular structures**

NGLview

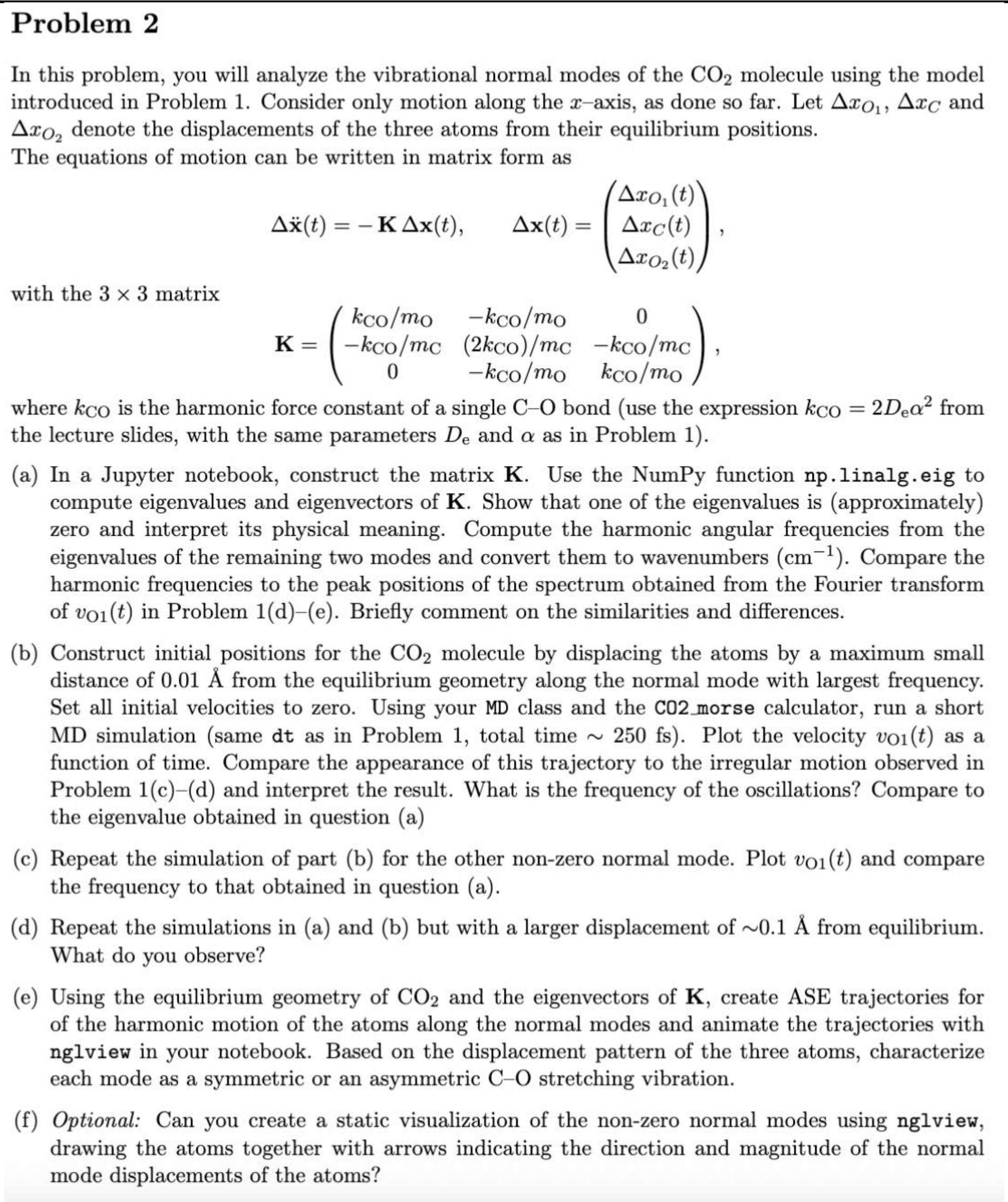
Display **interactive 3D view of the molecule**

```
view = nv.show_ase(atoms, default_representation=False)
view.add_representation('ball+stick’, ...)
view.center()
view
```

Display **interactive 3D view of a trajcteory** (movie of the trajectory)

```
view = nv.show_asetraj(traj_fast,
                                   default_representation=False)
view.add_representation('ball+stick’, ...)
view.center()
view
```

Assignment 7



Problem 2

In this problem, you will analyze the vibrational normal modes of the CO_2 molecule using the model introduced in Problem 1. Consider only motion along the x–axis, as done so far. Let Δx_{O_1}, Δx_C and Δx_{O_2} denote the displacements of the three atoms from their equilibrium positions.
The equations of motion can be written in matrix form as

$$\Delta\ddot{\mathbf{x}}(t) = -\mathbf{K}\,\Delta\mathbf{x}(t), \qquad \Delta\mathbf{x}(t) = \begin{pmatrix} \Delta x_{O_1}(t) \\ \Delta x_C(t) \\ \Delta x_{O_2}(t) \end{pmatrix},$$

with the 3 × 3 matrix

$$\mathbf{K} = \begin{pmatrix} k_{\mathrm{CO}}/m_{\mathrm{O}} & -k_{\mathrm{CO}}/m_{\mathrm{O}} & 0 \\ -k_{\mathrm{CO}}/m_{\mathrm{C}} & (2k_{\mathrm{CO}})/m_{\mathrm{C}} & -k_{\mathrm{CO}}/m_{\mathrm{C}} \\ 0 & -k_{\mathrm{CO}}/m_{\mathrm{O}} & k_{\mathrm{CO}}/m_{\mathrm{O}} \end{pmatrix},$$

where k_{CO} is the harmonic force constant of a single C–O bond (use the expression $k_{\mathrm{CO}} = 2D_{\mathrm{e}}\alpha^2$ from the lecture slides, with the same parameters D_{e} and α as in Problem 1).

(a) In a Jupyter notebook, construct the matrix **K**. Use the NumPy function `np.linalg.eig` to compute eigenvalues and eigenvectors of **K**. Show that one of the eigenvalues is (approximately) zero and interpret its physical meaning. Compute the harmonic angular frequencies from the eigenvalues of the remaining two modes and convert them to wavenumbers (cm^{-1}). Compare the harmonic frequencies to the peak positions of the spectrum obtained from the Fourier transform of $v_{\mathrm{O1}}(t)$ in Problem 1(d)–(e). Briefly comment on the similarities and differences.

(b) Construct initial positions for the CO_2 molecule by displacing the atoms by a maximum small distance of 0.01 Å from the equilibrium geometry along the normal mode with largest frequency. Set all initial velocities to zero. Using your `MD` class and the `CO2_morse` calculator, run a short MD simulation (same `dt` as in Problem 1, total time ~ 250 fs). Plot the velocity $v_{\mathrm{O1}}(t)$ as a function of time. Compare the appearance of this trajectory to the irregular motion observed in Problem 1(c)–(d) and interpret the result. What is the frequency of the oscillations? Compare to the eigenvalue obtained in question (a)

(c) Repeat the simulation of part (b) for the other non-zero normal mode. Plot $v_{\mathrm{O1}}(t)$ and compare the frequency to that obtained in question (a).

(d) Repeat the simulations in (a) and (b) but with a larger displacement of ~0.1 Å from equilibrium. What do you observe?

(e) Using the equilibrium geometry of CO_2 and the eigenvectors of **K**, create ASE trajectories for of the harmonic motion of the atoms along the normal modes and animate the trajectories with `nglview` in your notebook. Based on the displacement pattern of the three atoms, characterize each mode as a symmetric or an asymmetric C–O stretching vibration.

(f) *Optional:* Can you create a static visualization of the non-zero normal modes using `nglview`, drawing the atoms together with arrows indicating the direction and magnitude of the normal mode displacements of the atoms?