Homework 2: Numerical Methods in Python

CHEM4050/5050 Fall 2025

Monday, September 15, 2025, at 11:59 PM Central

Contents

1 Optimizing the Geometry of the Argon Dimer and Trimer

In this problem, you will compute the ground-state geometries of the argon dimer (Ar_2) and argon trimer (Ar_3). The interaction between Ar atoms can be modeled using the Lennard-Jones potential, which is given by the formula

$$V(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

where $V(r)$ is the potential energy as a function of the interatomic distance r, ϵ is the depth of the potential well (*i.e.*, the minimum energy), σ is the distance at which the potential is zero, and r is the distance between two atoms. For Ar, typical values are $\epsilon = 0.01$ eV and $\sigma = 3.4$ Å.

1.1 Tasks

1.1.1 Part 1: Argon Dimer

1. **Lennard-Jones Potential**
   - ☐ Write a Python function `lennard_jones(r, epsilon=0.01, sigma=3.4)` that takes the distance r between two Ar atoms and returns the potential energy $V(r)$ using the Lennard-Jones formula.
2. **Optimize the Geometry of Ar_2**
   - ☐ Use `scipy.optimize.minimize` to find the distance between two Ar atoms that minimizes the Lennard-Jones potential.
   - ☐ Start with an initial guess of $r = 4$ Å.
3. **Plot the Potential Energy Curve**
   - ☐ Plot the Lennard-Jones potential $V(r)$ as a function of the distance r between $3\,\text{Å} \le r \le 6\,\text{Å}$.
   - ☐ Mark the equilibrium distance (*i.e.*, the distance at the minimum potential) on the plot.

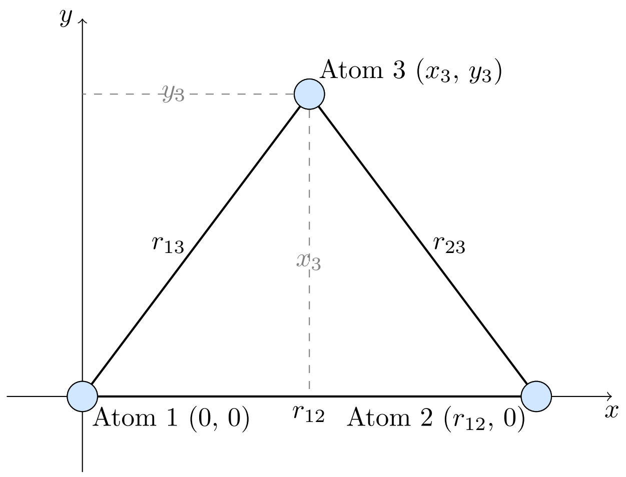

Figure 1: Geometry of Ar_3 in terms of r_{12}, r_{13}, and r_{23}

1.1.2 Part 2: Argon Trimer

For Ar_3, you will need to account for the interactions between all three Ar atoms.

1. **Total Potential Energy for Ar_3**

- ☐ The total potential energy of Ar_3 is the sum of the Lennard-Jones interactions between all three pairs of atoms

$$V_{\text{total}} = V(r_{12}) + V(r_{13}) + V(r_{23})$$

  where r_{12}, r_{13}, r_{23} are the distances between the pairs of atoms as shown in Figure 1.

2. **Optimize the Geometry of Ar_3**
   - ☐ Use `scipy.optimize.minimize` to find the optimal geometry of Ar_3. Represent the positions of the three Ar atoms in two dimensions:
     - ☐ The first atom is fixed at the origin $(0, 0)$,
     - ☐ The second atom is placed on the x-axis at $(r_{12}, 0)$,
     - ☐ The third atom's position is (x_3, y_3), where both x_3 and y_3 are variables to be optimized.
   - ☐ Minimize the total potential energy of the trimer.

Multivariate *vs.* Univariate Minimization

In univariate minimization, the function depends on a single variable, and the optimization algorithm adjusts this one variable to find the minimum. For example, when optimizing the distance between two atoms, r, in a one-dimensional potential energy function $V(r)$, only the variable r is adjusted.

In multivariate minimization, the function depends on multiple variables, and the optimization algorithm adjusts all these variables simultaneously to find the minimum. For example, when optimizing the geometry of Ar_3, you need to consider multiple coordinates representing the positions of atoms in space.

The minimization function will adjust the following variables to find the optimal geometry of the argon trimer:

- The distance between atoms 1 and 2, r_{12}.
- The position of atom 3, represented by its coordinates (x_3, y_3).

When using `scipy.optimize.minimize` for multivariate optimization, you must provide:

- A function that returns the total potential energy of the system as a function of all the variables.
- A vector of variables that represents all the parameters being optimized (e.g., r_{12}, x_3, y_3).

The optimization algorithm will adjust all variables in this vector simultaneously to minimize the total potential energy.

3. **Report the Optimal Geometry**
   - ☐ Print the optimal distances r_{12}, r_{13}, r_{23} between the atoms in the trimer.
   - ☐ Print the optimal angles between the atoms in the trimer.
   - ☐ Comment on the geometric arrangement of the atoms (*e.g.*, are they in an equilateral triangle?).

1.2 Submission Guidelines

- ☐ Push two Python files named `optimize_argon_dimer.py` and `optimize_argon_trimer.py` to your GitHub repository named `chem-4050-5050` for this course.
- ☐ When run, *e.g.*, using `python optimize_argon_dimer.py`, each file should

- ☐ Use your functions `compute_bond_length` and `compute_bond_angle` from Homework 1
- ☐ Print the optimal distances (and angles) between the atoms
- ☐ Produce an `xyz` file for the geometry in a directory named `homework-2-1`.

Format of an `xyz` File

An `xyz` file format is used to store molecular geometry data. The format consists of three sections:

1. The first line contains a single integer indicating the number of atoms in the molecule.
2. The second line is a comment line, which can be left blank or used to describe the molecule.
3. The subsequent lines (one for each atom) define the atomic positions. Each line contains:
   - The atomic symbol (*e.g.*, H, O, C).
   - The x-coordinate of the atom in Angstroms.
   - The y-coordinate of the atom in Angstroms.
   - The z-coordinate of the atom in Angstroms.

Example of a simple `xyz` file for a water molecule:

```
3
Water molecule
O      0.000000     0.000000     0.000000
H      0.758602     0.000000     0.504284
H     -0.758602     0.000000     0.504284
```

- ☐ Include comments in your code explaining each step.
- ☐ Your plots should be clear, well-labeled, properly formatted, and stored in `homework-2-1`.

2 Numerical Integration of the Second Virial Coefficient

In this problem, you will compute the second virial coefficient at constant volume (B_{2V}) for particles interacting via different pairwise potentials using numerical integration. B_{2V} quantifies non-ideal behavior in gases by accounting for intermolecular interactions at moderate densities. It appears in the virial equation of state, which extends the ideal gas law to describe real gases

$$PV_m = RT\left(1 + \frac{B_{2V}}{V_m} + \dots\right)$$

where V_m is the molar volume, T is the temperature, and P is the pressure. Positive values of B_{2V} indicate repulsive interactions (*e.g.*, at high temperatures). Negative values of B_{2V} reflect attractive interactions (*e.g.*, in low-temperature gases). B_{2V} provides insights into molecular size, shape, and interaction strength, and is essential for understanding gas phase behavior, phase transitions, and properties like compressibility and heat capacities in real systems. B_{2V} is given by the integral

$$B_{2V}(T) = -2\pi N_A \int_0^{\infty} \left[e^{-u(r)/k_{\mathrm{B}}T} - 1\right] r^2\, dr$$

where N_A is Avogadro's number, r is the distance between two particles, $u(r)$ is the pairwise interaction potential, k_{B} is the Boltzmann constant, and T is the temperature. You will compute B_{2V} for three different interaction potentials: hard-sphere, square-well, and Lennard-Jones potentials.

1. **Hard-Sphere Potential**

$$V(r) = \begin{cases} \infty & r < \sigma \\ 0 & r \geq \sigma \end{cases}$$

   where σ is the diameter of the hard sphere.

2. **Square-Well Potential**

$$V(r) = \begin{cases} \infty & r < \sigma \\ -\epsilon & \sigma \leq r < \lambda\sigma \\ 0 & r \geq \lambda\sigma \end{cases}$$

   where σ is the particle diameter, ϵ is the well depth, and λ defines the range of the well.

3. **Lennard-Jones Potential**

$$V(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

   where ϵ is the depth of the potential well and σ is the distance at which the potential is zero.

2.1 Tasks

1. **Define the Potentials**
   - ☐ Write Python functions for the hard-sphere, square-well, and Lennard-Jones potentials.
2. **Numerical Integration**
   - ☐ Use numerical integration (*e.g.*, `scipy.integrate.trapezoid`) to compute the second virial coefficient B_{2V} for each potential at 100 K.
   - ☐ Perform the integration over the distance r from 0 (*e.g.*, 1/1000) to a large value (you can choose 5 times the characteristic length σ as an upper bound), using a grid of 1000 points.
   - ☐ Assume values for the potential parameters:
     - ☐ For the hard-sphere potential: $\sigma = 3.4\,\text{Å}$,
     - ☐ For the square-well potential: $\sigma = 3.4\,\text{Å}$, $\epsilon = 0.01\,\text{eV}$, $\lambda = 1.5$,

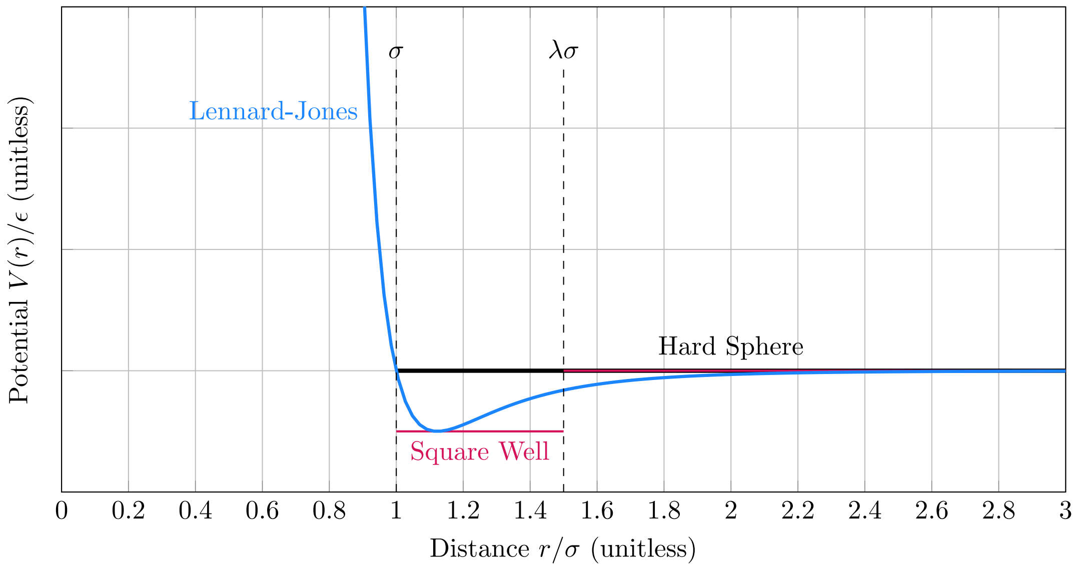

Figure 2: Functional forms of the hard-sphere, square-well, and Lennard-Jones potentials as a function of distance r between two particles.

- ☐ For the Lennard-Jones potential: $\sigma = 3.4$ Å, $\epsilon = 0.01$ eV.
- ☐ Approximate ∞ as 1000.

3. **Compute B_{2V} for Different Temperatures**
   - ☐ Compute B_{2V} for a range of temperatures (*e.g.*, $T = 100$ K to $T = 800$ K) for each potential.
   - ☐ Plot B_{2V} as a function of temperature for all three potentials on the same graph. Use appropriate labels for the axes and a legend to indicate the potentials. Add a line at $B_{2V} = 0$.
4. **Discussion**
   - ☐ Compare the behavior of the second virial coefficient for the different potentials. Discuss how the shape of the potential affects the behavior of B_{2V}.

2.2 Submission Guidelines

- ☐ Push a Python file named `compute_b2v.py` to your GitHub repository named `chem-4050-5050` for this course.
- ☐ When run, `compute_b2v.py` should
  - ☐ Reuse your function `lennard_jones` from Problem 1
  - ☐ Plot B_{2V} as a function of temperature for each potential
  - ☐ Produce an `csv` file for B_{2V} as a function of temperature for each potential in a directory named `homework-2-2`
- ☐ Include a brief discussion comparing the behavior of B_{2V} for the different potentials, and explain how the shape of the potential affects the results in a Markdown `md` file stored in `homework-2-2`. Refer to https://www.markdownguide.org/cheat-sheet/ for a quick overview of Markdown syntax.
- ☐ Include comments in your code explaining each step.
- ☐ Your plots should be clear, well-labeled, properly formatted, and stored in `homework-2-2`.

3 Graduate Supplement

In this problem, you will solve the time-independent Schrödinger equation for a one-dimensional harmonic and anharmonic oscillator using Python and a real-space grid method. The time-independent Schrödinger equation is

$$-\frac{\hbar^2}{2m}\frac{d^2\psi(x)}{dx^2} + V(x)\psi(x) = E\psi(x)$$

where $V(x)$ is the potential, $\hbar$ is the reduced Planck constant, m is the mass of the oscillator, E is the energy of the oscillator, and $\psi(x)$ is the wavefunction of the oscillator. The harmonic and anharmonic (*i.e.*, Morse) potential can be modeled as

$$V_{\text{harmonic}}(x) = \frac{1}{2}m\omega^2x^2$$
$$V_{\text{anharmonic}}(x) = V_{\text{Morse}}(x) = D\left[1 - e^{-\beta(x-x_0)}\right]^2$$

where ω is the frequency of the oscillator, D is the depth of the potential well, β is a parameter controlling the width of the width of the potential well, and x_0 is the equilibrium distance between the oscillating particles. We will solve this equation numerically by discretizing the second derivative using a finite difference method and constructing the Hamiltonian matrix for the system. The eigenvalues of the Hamiltonian matrix will give us the energy levels, and the eigenvectors will give the corresponding wavefunctions.

3.1 Tasks

1. **Define Constants and Discretize the System**
   - ☐ Use atomic units, where $\hbar = 1$ and $m = 1$.
   - ☐ $\omega = 1\,\text{a.u.}$, $D = 10\,\text{a.u.}$, and $\beta = \sqrt{1/(2D)}\,\text{a.u.}$.
   - ☐ Discretize the space using a real-space grid of 2000 points from $-L/2$ to $L/2$, where $L = 40\,\text{a.u.}$
2. **Construct the Potential and Laplacian Matrices**
   - ☐ Write a Python function to construct the potential matrix for the harmonic and anharmonic potentials using the discretized space. For a system with 5 points in space, the potential matrix looks like this:

$$\text{Potential Matrix} = \begin{pmatrix} V(x_1) & 0 & 0 & 0 & 0 \\ 0 & V(x_2) & 0 & 0 & 0 \\ 0 & 0 & V(x_3) & 0 & 0 \\ 0 & 0 & 0 & V(x_4) & 0 \\ 0 & 0 & 0 & 0 & V(x_5) \end{pmatrix}$$

   - ☐ Construct the Laplacian matrix using a finite difference approximation for the second derivative:

$$\text{Laplacian} = \frac{1}{(\Delta x)^2}\left(-2\mathbf{I} + \mathbf{I}_{\text{off-diagonal}}\right)$$

   where $\mathbf{I}$ is the identity matrix and Δx is the spacing between adjacent points on the grid.
3. **Construct the Hamiltonian Matrix**
   - ☐ The total Hamiltonian for the system is given by:

$$H = -\frac{\hbar^2}{2m}\cdot\text{Laplacian} + V(x)$$

   In atomic units, this simplifies to:

$$H = -\frac{1}{2}\cdot\text{Laplacian} + V(x)$$

- ☐ Write a Python function that constructs the Hamiltonian matrix by combining the Laplacian and potential matrices.

4. **Solve for Eigenvalues and Eigenfunctions**
   - ☐ Use NumPy's linear algebra package (`np.linalg.eig`) to compute the eigenvalues (energy levels) and eigenfunctions of the Hamiltonian for the harmonic and anharmonic potentials.
   - ☐ Store the eigenfunctions for both potentials.
   - ☐ Sort the eigenvalues in increasing order and extract the first ten energy levels.
5. **Plot the Results**
   - ☐ Plot the first ten wavefunctions along with their corresponding energy levels for both potentials.
   - ☐ Label the axes appropriately.

3.2 Submission Guidelines

- ☐ Push a Python file named `oscillator.py` that implements all the above tasks to your GitHub repository named `chem-4050-5050` for this course.
- ☐ Include comments in your code explaining each step.
- ☐ Your GitHub repository should include:
  - ☐ A plot of the first ten energy levels and wavefunctions in a directory named `homework-2-grad`.