Normal Mode Analysis | Science II

This project analyses the Normal Modes of an Argon System of 108 atoms following Lennard Jones Potential. Firstly, we generate a random configuration of atoms based on some set of constraints and implement the Periodic Boundary conditions. Next, we minimise the LJ Potential by using a gradient descent Algorithm to configure an optimal system. An Hessian Matrix is then generated along with Eigen Values and Vectors. Finally we plot the Frequencies on a Histogram.

Question 1

Question 1 generates a simulated random configuration of Argon System base on the following constraints:

- $N = 108$
- $L_x = L_y = L_z = 18\text{A}^\circ$ $\epsilon = 0.238\ \text{kcal/mol}$
- $\sigma = 3.4\text{A}^\circ$
- $r_{ij} < 3.4\text{A}^\circ$

The `get_config()` in the `main.py` function simply generates random tuples of length 3 which are enclosed within $(0, 0, 0)$ and (L_x, L_y, L_z) and validates if it can be inserted to the existing set of atoms with the constraints provided.

Since the 3D space is quite small, so incase, the code fails for more than $10,000$ trials at putting a new atom, we expect that we have generated a *badly* configured system and restart again.

The code takes significant amount of time(~ $1hr$) to generate an initial configuration. A sample file generated is provided in `init_config.txt` .

Question 2

Question 2 , We calculate the Lennard Jones potential of the above configured system. The Lennard Jones Potential is calculated from teh Lennard Jones Equation:

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

The `lj_potetntial()` function in the `hessian.py` file implements this equation.

The LJ Potential for the sample file$= -121.47326156242956 \text{ kcal/mol}$

Question 3

Here, we minimise our LJ Potential by optimising the initial configuration using a optimisation algorithm. In our case we use the ***Gradient Descent Algorithm.*** We use the `autograd` module to evaluate the partial gradients of LJ Potential with respect to the co-ordiantes of the atoms. We tune our $\alpha = 0.1$ and $num\_itr = 100$. The code for the same is implemented in `gradient_descent()` function of `main.py`

The minimised optimal LJ Potential Energy $= -148.72418559808176$

The new configuration is added to the `optimum_config.txt`

Question 4

In this question, we are required to get the Hessian Matrix, Eigen values and Eigen Vectors. The Hessian Matrix H is defined as:

$$\mathbf{H}_f = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1\,\partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2\,\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_n\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix},$$

or,

$$(\mathbf{H}_f)_{i,j} = \frac{\partial^2 f}{\partial x_i\,\partial x_j}.$$

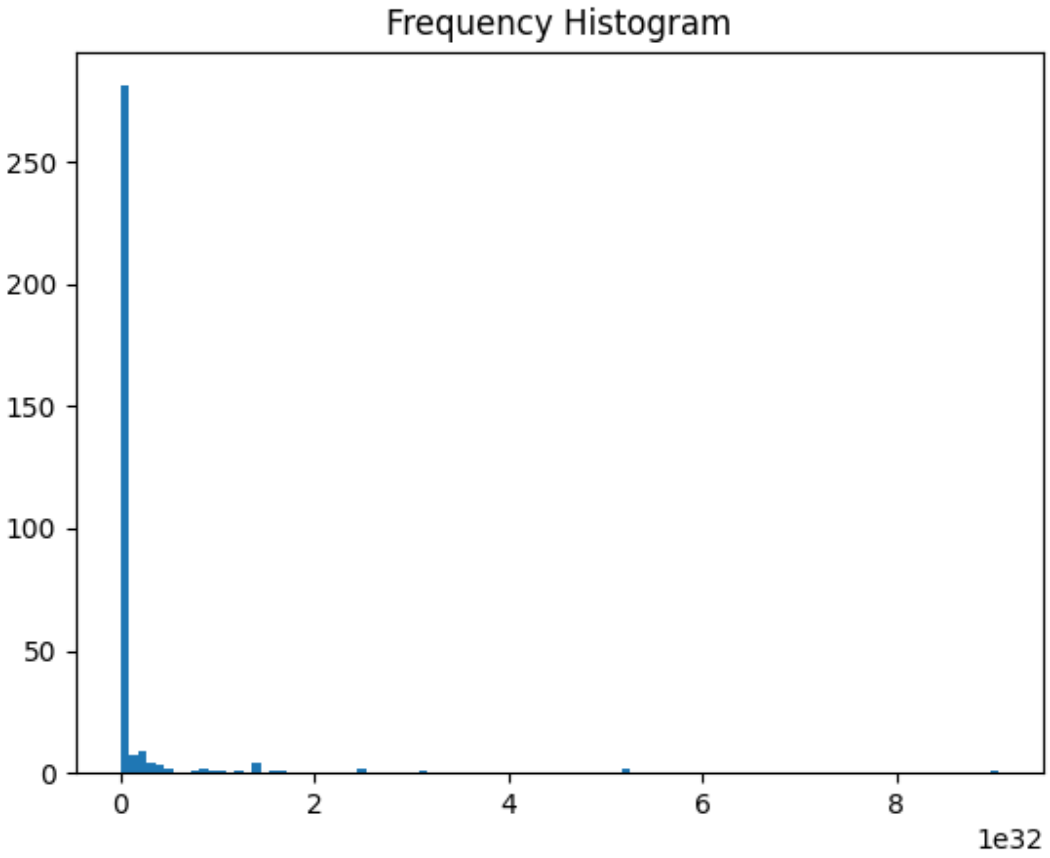
In our case, since LJ Potential is discrete and non-differentiable, we use the ***method of finite differences.***

The `hessian.py` constitutes of the `Hessian()` class which is implemented to calculate the Hessian Matrix, Eigen Vectors and Values. This data is registered into the `hessian.dat`, `eigen_vectors.dat`, `eigen_values.dat` .

Question 5

Given the Hassien Matrix Calculated in previous step, the eigen values and eigen vectors we
can calculate the normal mode frequencies. The corresponding evaluations to calculate the modes are implemented in the `Frequencies()` class of the `frequency.py` file.

The new modes are registered in the `mode.txt` and a corresponding histogram is plotted below



It can be clearly noticed, some Frequencies are high, whearas others are extremely low. It is actually a **Normal Distribution.**