# 1 Molecular Dynamics

One of the most seminal works in Molecular Dynamics was Dr. Aneesur Rahman's 1964 paper [?] "Correlations in the Motion of Atoms in Lquid Argon", where a system of 864 argon atoms was simulated on a CDC 3600 computer using a predictor-corrector method and the computed quantities were matched against the empirical values. This was followed by a paper in 1967 [?] by Dr. Loup Verlet, which introduced the Verlet Integration method for the same model of 864 argon atoms.

In this section, we look at setting up the molecular dynamics simulation as described in the aforementioned papers. We focus consider the basic foundations of the simulation such as inter-particle potentials and interaction, and then introduce certain techniques that help us implement these foundations in a manner that would not be computationally expensive.

## 1.1 Hamiltonian Equation

For a system of molecules, we have to solve the Hamiltonian [?]

$$H(p,q) = \frac{1}{2}\sum_{i=1}^{N} m_i^{-1} p_i^T p_i + \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} V_{ij}\left(\|q_i - q_j\|\right), \tag{1.1}$$

where $V_{ij}$ is a potential function, and $q_i$, $p_i$ and $m_i$ represent the position, momentum and mass of molecule $i$ respectively. The first term of the equation can be seen as the kinetic energy of the system, while the second term is the potential energy. We re-write Equation 1.1 as

$$\dot{q}_i = \frac{1}{m_i} p_i, \qquad \dot{p}_i = \sum_{j=1}^{N} \nu_{ij}\left(q_i - q_j\right), \tag{1.2}$$

where for $i < j$, $\nu_{ij} = \nu_{ji} = -\frac{V'_{ij}(r_{ij})}{r_{ij}}$ and $\nu_{ii} = 0$, with $r_{ij} = \|q_i - q_j\|_2$.

## 1.2 Lennard-Jones Potential

Before populating the argon atoms, we need to establish how the molecules will interact. We use the Lennard-Jones Potential to approximate the interaction with two atoms

$$V\left(\mathbf{r}_{ij}\right) = 4\varepsilon\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right], \tag{1.3}$$

where $\mathbf{r}_{ij}$ is the displacement between the $i^{th}$ and the $j^{th}$ atoms, $r_{ij} = \|\mathbf{r}_{ij}\|_2$, $\varepsilon$ is the potential well depth (a measure of strength of attraction of two atoms), and $\sigma$ is the distance at which the potential between two atoms is 0. The Lennard-Jones potential is a very simple model for the interactive forces between atoms; nevertheless

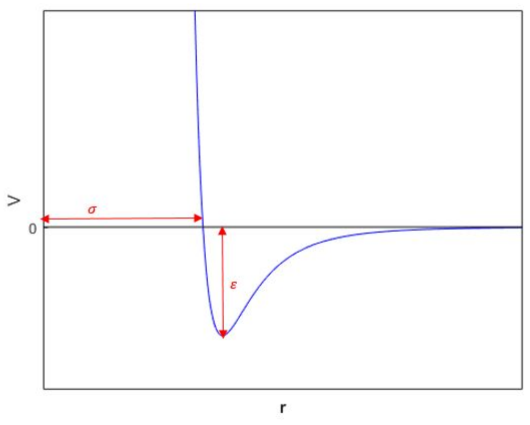

Figure 1.1: Lennard-Jones Potential

it was found to suffice for modelling argon atoms in [**?**]. In fact, the simplicity of the Lennard-Jones potential contributes to speed of the simulation - given a system of $n$ atoms, the potential needs to be evaluated $\frac{n(n+1)}{2}$ times (as $r_{ij} = r_{ji}$).

For an argon atom, $\varepsilon = 1.65 \times 10^{-21}$ J and $\sigma = 3.40 \times 10^{-10}$ m. The computer is prone to making errors when operating on such small numbers, so we look at a technique that handles this problem in Subsection 1.4.

## 1.3 Forces

We have that force $\mathbf{f}$ along the vector $\mathbf{r}_{ij}$ is given by

$$\mathbf{f}\left(\mathbf{r}_{ij}\right) = -\frac{dV}{d\mathbf{r}_{ij}}.$$

Now,

$$\frac{dV}{dr_{ij}} = 4\varepsilon\left[-12\sigma^{12}\left(\frac{1}{r_{ij}}\right)^{13} + 6\sigma^{6}\left(\frac{1}{r_{ij}}\right)^{7}\right],$$

and

$$\frac{dr_{ij}}{d\mathbf{r}_{ij}} = \frac{\mathbf{r}_{ij}}{r_{ij}}.$$

By the chain rule, we get that

$$\begin{aligned}\mathbf{f}\left(\mathbf{r}_{ij}\right) &= -4\varepsilon\left[-12\sigma^{12}\left(\frac{1}{r_{ij}}\right)^{13} + 6\sigma^{6}\left(\frac{1}{r_{ij}}\right)^{7}\right] \times \frac{\mathbf{r}_{ij}}{r_{ij}} \\ &= 48\varepsilon\left[\sigma^{12}\left(\frac{1}{r_{ij}}\right)^{14} - \frac{\sigma^{6}}{2}\left(\frac{1}{r_{ij}}\right)^{8}\right]\mathbf{r}_{ij}. \qquad (1.4)\end{aligned}$$

Hence, we get that the force exterted on particle $i$, $\mathbf{F}_i$ is given by

$$\mathbf{F}_i = \sum_{i \neq j} \mathbf{f}_{ij}.$$

Note that we can reduce the number of computations of $\mathbf{f}_{ij}$ by half due to Newton's Third Law Motion: every action has an equal and opposite reaction implies that $\mathbf{f}_{ij} = -\mathbf{f}_{ji}$.

## 1.4 Reduced Units

As explained earlier, limits on a computer's precision can introduce floating point errors when expressing miniscule quantities such as potential in terms of the SI units. Hence, we scale - or "reduce" - these SI unis such that the quantities involved in the simulation have an order of magnitude of 1. This makes simplifies the calculations, and has the added benefit of making erroneous results easier to spot (as extreme values are now unlikely). We start off by expressing lengths in terms of $\sigma^*$, where $1\,\sigma^* = 3.40 \times 10^{-10}$ m, and energy in terms of $\varepsilon^*$, where $1\,\varepsilon^* = 1.65 \times 10^{-21}$ J. Then, the Lennard-Jones potential can be calculated as

$$V\left(\mathbf{r}_{ij}\right) = 4\left[\left(\frac{1}{r_{ij}}\right)^{12} - \left(\frac{1}{r_{ij}}\right)^{6}\right], \qquad (1.5)$$

where $\mathbf{r}_{ij}$ and $r_{ij}$ have the same meaning as before, but are expressed as a multiple of $\sigma^*$. Then, we can rewrite Equation 1.4 as

$$\mathbf{f}\left(\mathbf{r}_{ij}\right) = 48 \left(\frac{1}{r_{ij}}\right)^8 \left[\left(\frac{1}{r_{ij}}\right)^6 - \frac{1}{2}\right] \mathbf{r}_{ij}. \tag{1.6}$$

As the SI unit for force is J$\cdot$m$^{-1}$, the reduced unit for force is $\varepsilon \cdot \sigma^{-1}$.

Similarly, we define a reduced unit for mass, m$^*$ for the mass of an argon atom ($6.69 \times 10^{-23}$ g). In fact, we can simplify the calculation $\mathbf{a} = \frac{\mathbf{F}}{m}$ by setting 1 m$^*$ = $48\varepsilon^*{\sigma^*}^{-2}{\mathrm{s}^*}^2$, where s$^*$ is the yet-to-be-calculated reduced unit for time (this definition follows from 1 J = 1 kg$\cdot$m$^{-2}\cdot$s$^2$). Plugging in the values, we get that 1 s$^*$ = $3.125 \times 10^{-13}$ s. Then a step-size of 10 femtoseconds is equivalent to $h = 0.032$. We use such a small step size as we assume during numerical differentiation that the velocity and the acceleration of an argon atom are constant in the interval of one time step. Lastly, we will see in a later subsection that the distribution of velocities in a system of particles depends upon the Boltzmann constant, $k_B = 1.381 \times 10^{-23} \mathrm{J} \cdot \mathrm{K}^{-1}$. We set this to 1 in our reduced units system, by defining 1 K$^*$ = 119.6 K. These results are summarised in Table 1.1.

| Physical Quantity | Reduced Unit | Value |
|---|---|---|
| Length | $\sigma^*$ | $3.40 \times 10^{-10}$ m |
| Energy | $\varepsilon^*$ | $1.65 \times 10^{-21}$ J |
| Mass | m$^*$ | $6.69 \times 10^{-23}$ kg |
| Time | s$^*$ | $3.125 \times 10^{-13}$ s |
| Temperature | K$^*$ | 119.6 K |

Table 1.1: Reduced Units and their Values

## 1.5 Initializing Positions

We initialize the atoms as face-centered cubic lattice, i.e., a cube consisting of periodic images of a unit lattice cell where the atoms are placed at the eight corners, and at the center of each face. It is essential to start off the simulation with the atoms in this structure irrespective of what state of matter we want to study the simulation in. This is because randomly distributing the particles could lead to some of them being placed in close proximity of each other, leading to strong repulsive forces that would introduce errors in the simulation. Hence, if we want to study argon atoms in liquid or gas state, we start off with a face-centered cubic lattice, and increase the temperature of the system while the simulation is running.

We take as input the density of the cube to set the length of the cubic lattice. Thus, for 864 atoms at a density of 1.374 g $\cdot$ cm$^{-3}$, we get the following cube lattice

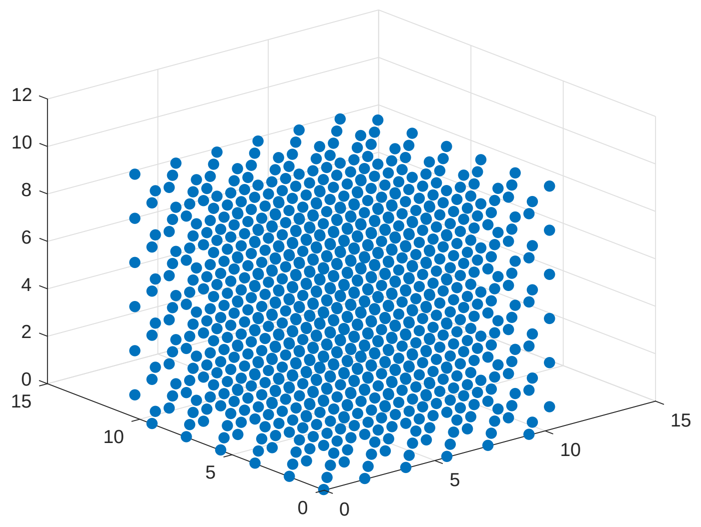

Figure 1.2: Initial cubic lattice

## 1.6 Initializing Velocities

As argon is an ideal gas, we can set the initial temperature to above boiling point (87.8996 K or 0.7356 K$^*$) and use the Maxwell-Boltzmann distibution to generate the initial set of velocities. The probablity density function for velocities under this distribution is given by

$$f_{\mathbf{v}}\left(v_x, v_y, v_z\right) = \left(\frac{m}{2\pi k_B T}\right)^{\frac{3}{2}} \exp\left[-\frac{m\left(v_x^2 + v_y^2 + v_z^2\right)}{2k_B T}\right].$$

Assuming that the velocity in a direction is independent of the velocities in the other directions, we get that the probability density function for velocity $v_i$ in direction i is

$$f_{\mathbf{v}}\left(v_i\right) = \left(\frac{m}{2\pi k_B T}\right)^{\frac{1}{2}} \exp\left[-\frac{m v_i^2}{2k_B T}\right].$$

So, $v_i \sim \mathcal{N}\left(0, \frac{k_B T}{m}\right)$. We use the `normrnd` function of MATLAB to generate the velocities, and then correct them so that there is no overall linear momentum.

Now, we are using the Maxwell-Boltzmann distribution for ideal gases to initialize the velocities, but the particles have been initialized as a solid. Hence, we will 'boil' the solid for a fixed number of steps at the start of the simulation by scaling the velocities

$$\mathbf{v}_{new} = \sqrt{\frac{T_{target}}{T_{old}}} \cdot \mathbf{v}_{old}.$$

This method is known as equilibration.

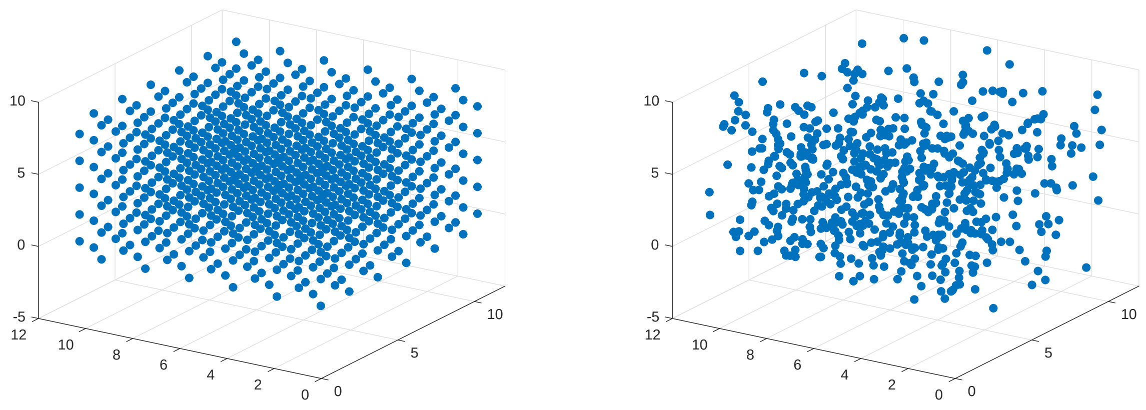

Figure 1.3: Change in the position of particles after 500 steps of equilibration

## 1.7 Periodic Boundary Conditions and Minimum Image Convention

Typically, we'd imagine the argon gas to be in a container, with particles colliding against the walls and moving off in the opposite direction. In the real world, the number of particles colliding against the wall would be miniscule - for a liquid with $10^{21}$ particles, only 1 in $10^7$ would be close to the walls [?]. In comparison, our system of 846 particles has almost half of them at the edges. The simulation would, thus, not reflect the internal state of the system accurately. We get rid off the walls through a technique known as periodic boundary conditions. Given a face-centered cubic lattice with side $L$, we consider it to consist of 27 identical cubes of side $\frac{L}{2}$ (one central box and 26 images of it), and we maintain this idea of images throughout the simulation, as shown in Figure 1.4. This has two effects:

- Say we have two boxes beside each, Box A and Box B (where Box B is an image of Box A). Then, as a particle moves out of Box A (say from the left), its image in Box B also moves out of Box B from the left, and enters Box A. This maintains that all the boxes in the cube are images of each other.
- A particle in a box will interact not only with the particles in the same box, but also the particles in the other boxes - leading to repeated calculation and effect of the same force.

We consider for this 'wrap-around' effect through the Minimum Image Convention - each particle interacts only with the particles in its box. Given x-components $x_i$ for particle $i$, $x_j$ for particle $j$, and the length of the box in the x-component, $L_x$, we implement

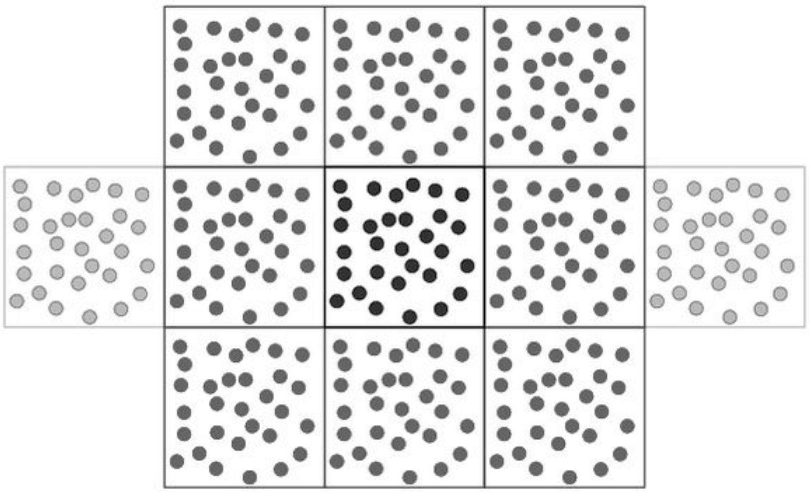

Figure 1.4: Periodic Boundary Conditons in 2-d

the Minimum Image Criterion using the algorithm below.

**Algorithm 1.1:** Minimum Image Convention in 1-d

```
input  : $x_i, x_j, L_x$
output: $x_{ij}$

$x_{ij} \leftarrow x_i - x_j$ ;
$half\_L_x \leftarrow \frac{L_x}{2}$;
if $x_{ij} > half\_L_x$ then
    $x_{ij} \leftarrow x_{ij} - L_x$;
else
    if $x_{ij} \leq -half\_L_x$ then
        $x_{ij} \leftarrow x_{ij} + L_x$;
    end
end
```

This is repeated for all three dimensions to find the correct image of the particle. We shorten the above algorithm by making use of MATLAB's vector operations: `diff_r = diff_r - L*round(diff_r/L)`. The elimination of the conditional checks gives us a 10x reduction in time spent on implementing the minimum image convention.

## 1.8 Cut-off Potential and Neighbour Lists

A final consideration for our molecular dynamics simulation is the number of times we are evaluating the forces, and how many of these evaluations are effective. Looking at the graph of the Lennard-Jones Potential in Figure 1.1, we see that the potential tends to zero as the inter-particle distance increases. Furthermore, the potential does not change much for over larger distances - implying that it is not efficient to consider interactions over all inter-particle distances. Rather, we can define a cut-off distance, $r_c$ beyond which the potential is zero. We neeed to now shift the Lennard-Jones Potential; otherwise the potential will have a discontinuity at $r_c$ and Newton's Laws of Motion do not hold at discontinuities. So, we get the shifted Lennard-Jones potential

$$V\left(\mathbf{r}_{ij}\right)=\begin{cases}4\left[\left(\left(\frac{1}{r_{ij}}\right)^{12}-\left(\frac{1}{r_{ij}}\right)^{6}\right)-\left(\left(\frac{1}{r_c}\right)^{12}-\left(\frac{1}{r_c}\right)^{6}\right)\right] & ,\quad \text{if } r_{ij}\leq r_c\\ 0 & ,\quad \text{otherwise}\end{cases} \tag{1.7}$$

What is the optimal value for $r_c$? Periodic boundary conditions enforce that $r_c < \frac{L}{2}$ (as particles can only interact with particles in the same box). Generally, $r_c \leq \frac{L}{4}$[?]. In our simulations, we have $L = 10.229\,\sigma^*$; so, we define $r_c = 2.5\,\sigma^*$. We substitute this into Equation 1.5 and confirm that this is a good choice: $V(2.5) = -0.0163$, $V(2.6) = -0.0129$, and $V(3) = -0.005479$. The loss in potential by setting $r_c = 2.5\,\sigma^*$ is minimal. The effect on computing time is phenomenal - only 6.3% of all pairwise distances are smaller than $2.5\,\sigma^*$, thus eliminating 93.7% of the computationally intensive force calculations.

As we saw earlier, repeated conditional checks are not very efficient and consume a lot of time. This is especially expensive if each iteration of the simulation involved calculating the inter-particle distance and checking if that distance is lesser than $2.5\sigma^*$ - around $400,000$ checks per iteration! Assuming that the positions of the particles do not vary much in each successive iteration, we can calculate a list of 'neighbours' for each particle at the start of the simulation. However, it cannot be guaranteed that this list will be accurate for the entire run of the simulation. In fact, tests conducted with the Velocity Verlet algorithm show that this list is only 80% accurate after 10 iterations, and 50% accurate after 35 iterations. Clearly, we can't use the same list for the entire simulation.

To find the optimal number of iterations after which the list of neighbours should be refreshed ($N_n$), we run the simulation for 100 equilibration iterations ($N_e$) followed by 600 iterations ($N_f$), calculating the energy values every 10 iterations ($N_s$) after equilibration. We compute the time taken for each run and the percentage error in the final energy values, using $N_n = 1$ as our reference (neighbours lists are updated on each iteration). The results summarised in Table 1.2 make the case for using the same list for over multiple iterations (5× improvement in speed), but updating them regularly enough to reduce the overall error. There's trade-off between speed and accuracy, and setting $N_n = 15$ seems to be a good compromise.

| $N_n$ | % Error in Final Energy Value | Time taken (s) |
|---|---|---|
| 1 | 0.00 | 945.86 |
| 10 | 3.18 | 164.55 |
| 15 | 4.43 | 146.90 |
| 20 | 5.37 | 127.55 |
| 35 | 7.67 | 99.58 |
| 100 | 8.08 | 85.08 |
| No update | -1.21 | 62.18 |

Table 1.2: Results of tests to find optimal $N_n$

## 1.9 Computing the Jacobian

Recall from Equation **??** that we need to compute the Jacobian of $\mathbf{a}(\mathbf{x}_i)$. For this, we need to store the coordinates of the molecules in a column vector of size $3N$, where N is the number of molecules in the system - rather than a more intuitive $N \times 3$ matrix. We fix the order of coordinates as

$$\begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ \vdots \\ x_{3N} \\ y_{3N} \\ z_{3N} \end{pmatrix}$$

where $x_i, y_i, z_i$ are the x,y, and z coordinates of particle $i$ respectively (we will continue to use an $N \times 3$ matrix for the explicit Newmark Beta methods, as this orientation is proven to be faster during tests). The Jacobian is then a $3N \times 3N$ matrix, which would be very expensive to compute for large $N$.

$$J = \begin{pmatrix} \frac{\partial \mathbf{f}_1}{\partial \mathbf{p}_1} & \frac{\partial \mathbf{f}_1}{\partial \mathbf{p}_2} & \cdots & \frac{\partial \mathbf{f}_1}{\partial \mathbf{p}_n} \\ \frac{\partial \mathbf{f}_2}{\partial \mathbf{p}_1} & \frac{\partial \mathbf{f}_2}{\partial \mathbf{p}_2} & \cdots & \frac{\partial \mathbf{f}_2}{\partial \mathbf{p}_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{f}_n}{\partial \mathbf{p}_1} & \frac{\partial \mathbf{f}_n}{\partial \mathbf{p}_2} & \cdots & \frac{\partial \mathbf{f}_n}{\partial \mathbf{p}_n} \end{pmatrix},$$

where $\mathbf{f}_i = \sum_{i=1}^{N} \mathbf{f}_{ij}$, $\mathbf{p}_i = \left(\begin{smallmatrix} x_i \\ y_i \\ z_i \end{smallmatrix}\right)$, and

$$\frac{\partial \mathbf{f}_i}{\partial \mathbf{p}_j} = \begin{pmatrix} \frac{\partial f_{i_x}}{\partial x_j} & \frac{\partial f_{i_x}}{\partial y_j} & \frac{\partial f_{i_x}}{\partial z_j} \\ \frac{\partial f_{i_y}}{\partial x_j} & \frac{\partial f_{i_y}}{\partial y_j} & \frac{\partial f_{i_y}}{\partial z_j} \\ \frac{\partial f_{i_z}}{\partial x_j} & \frac{\partial f_{i_z}}{\partial y_j} & \frac{\partial f_{i_z}}{\partial z_j} \end{pmatrix}, \tag{1.8}$$

where $f_{i_x}$ is the x-component of the force $\mathbf{f}_i$.
Now, we have from Equation 1.6

$$\begin{aligned} f_{i_x} &= \sum_{j=1}^{N} f_{ij_x} \\ &= \sum_{j \in ngbrs(i)} 48 \left[ \left(\frac{1}{r_{ij}}\right)^{14} - \frac{1}{2}\left(\frac{1}{r_{ij}}\right)^{8} \right] x_{ij}, \end{aligned}$$

where $r_{ij} = \sqrt{x_{ij}^2 + y_{ij}^2 + z_{ij}^2}$, and $ngbrs(i)$ is the list of neighbors of particle $i$.
Focusing on just $f_{ij_x}$ for some $i$ and $j$, we write $f_{ij_x} = 48 \cdot s_{ij} \cdot x_{ij}$. We can see that

$$\begin{aligned} \frac{ds_{ij}}{dx_i} &= -14\,(x_i - x_j)\left(\frac{1}{r_{ij}}\right)^{16} + 4\,(x_i - x_j)\left(\frac{1}{r_{ij}}\right)^{10}, \\ \frac{ds_{ij}}{dx_j} &= 14\,(x_i - x_j)\left(\frac{1}{r_{ij}}\right)^{16} - 4\,(x_i - x_j)\left(\frac{1}{r_{ij}}\right)^{10} \\ &= -\frac{ds_{ij}}{dx_i}. \end{aligned}$$

Then,

$$\begin{aligned}
\frac{df_{ij_x}}{dx_i} &= 48\left(\frac{ds_{ij}}{dx_i}\cdot x_{ij} + s_{ij}\cdot\frac{dx_{ij}}{dx_i}\right)\\
&= 48\left(-14\left(x_i - x_j\right)^2\left(\frac{1}{r_{ij}}\right)^{16} + 4\left(x_i - x_j\right)^2\left(\frac{1}{r_{ij}}\right)^{10} + s_{ij}\right), \qquad (1.9)\\
\frac{df_{ij_x}}{dx_j} &= 48\left(\frac{ds_{ij}}{dx_j}\cdot x_{ij} + s_{ij}\cdot\frac{dx_{ij}}{dx_j}\right)\\
&= 48\left(-\frac{ds_{ij}}{dx_i}\cdot x_{ij} - s_{ij}\right)\\
&= -\frac{df_{ij_x}}{dx_i}. \qquad (1.10)
\end{aligned}$$

Furthermore,

$$\begin{aligned}
\frac{df_{ij_x}}{dy_i} &= 48\left(\frac{ds_{ij}}{dy_i}\cdot x_{ij} + s_{ij}\cdot\frac{dx_{ij}}{dy_i}\right)\\
&= 48\left(-14\left(x_i - x_j\right)\left(y_i - y_j\right)\left(\frac{1}{r_{ij}}\right)^{16} + 4\left(x_i - x_j\right)\left(y_i - y_j\right)\left(\frac{1}{r_{ij}}\right)^{10}\right), \qquad (1.11)\\
\frac{df_{ij_x}}{dy_j} &= 48\left(\frac{ds_{ij}}{dy_j}\cdot x_{ij} + s_{ij}\cdot\frac{dx_{ij}}{dy_j}\right)\\
&= 48\left(-\frac{ds_{ij}}{dy_i}\cdot x_{ij}\right)\\
&= -\frac{df_{ij_x}}{dy_i}. \qquad (1.12)
\end{aligned}$$

We extend these equations to all three components, and get from Equations 1.9 and 1.11 that

$$\frac{\partial \mathbf{f}_{ij}}{\partial \mathbf{p}_i} = \begin{pmatrix} s_{ij} & 0 & 0\\ 0 & s_{ij} & 0\\ 0 & 0 & s_{ij}\end{pmatrix} + 48\left(-14\left(\frac{1}{r_{ij}}\right)^{16} + 4\left(\frac{1}{r_{ij}}\right)^{10}\right)\mathbf{r}_{ij}\mathbf{r}_{ij}^T,$$

where $\mathbf{r}_{ij} = \left(\begin{smallmatrix} x_i - x_j\\ y_i - y_j\\ z_i - z_j\end{smallmatrix}\right)$. Lastly, we need the following relations

$$\begin{aligned}
\frac{\partial \mathbf{f}_{ij}}{\partial \mathbf{p}_j} &= -\frac{\partial \mathbf{f}_{ij}}{\partial \mathbf{p}_i} && \cdots \text{ from Equations 1.10 and 1.12,}\\
\frac{\partial \mathbf{f}_{ji}}{\partial \mathbf{p}_i} &= -\frac{\partial \mathbf{f}_{ij}}{\partial \mathbf{p}_i} && \cdots \text{ as } \mathbf{f}_{ji} = -\mathbf{f}_{ji},\\
\frac{\partial \mathbf{f}_{ji}}{\partial \mathbf{p}_j} &= -\frac{\partial \mathbf{f}_{ij}}{\partial \mathbf{p}_j} = \frac{\partial \mathbf{f}_{ij}}{\partial \mathbf{p}_i}
\end{aligned}$$

Now, we can use the last four relations to populate Equation 1.8, and we can calculate the Jacobian.