1 Introduction

Geometric Numerical Integration, when considered on the grand timeline of Mathematics, is a field still in its younger days. While work on the numerical treatment of differential equations had started off towards the end of the 19th Century, it wasn't till the 1980s that numerical analysts shifted their focus onto Geometric Numerical Integration. A better part of the century had been involved in improving quantity - producting methods that could produce accurate results without being computationally strenuous. But the interest in running computer simulations over longer time intervals revealed that quality was equally important. Most methods could accurately reflect properties of a system, such as its asymptotic time behaviour or its sensitivity to changes in initial conditions; not all could preserve attributes such as total enery or momentum - properties that are invariants to the flow of the differential equation.

Consider, for example [2], the following set of Lotka-Volterra prey-predator equations

$$\frac{du}{dt} = u - uv, \qquad \frac{dv}{dt} = uv - 2v, \tag{1.1}$$

where $u(t)$ is the population of the prey over time, and $v(t)$ is that of the predator. Dividing the two equations in 1.1, we get that

$$\left(1 - \frac{2}{u}\right) du = \left(\frac{1}{v} - 1\right) dv.$$

Integrating the two sides of the equation, we get that

$$I(u, v) = u - 2\ln u + v - \ln v = const, \tag{1.2}$$

giving us that the solutions to 1.1 lie on the level curves of 1.2.

We solve the differential equation 1.1 with initial points $(u_0, v_0) = (2, 4)$ and $h = 0.02$ using three different numerical methods: Explicit Euler, Implicit Euler, and Symplectic Euler. The results can be found in Figure 1.1.

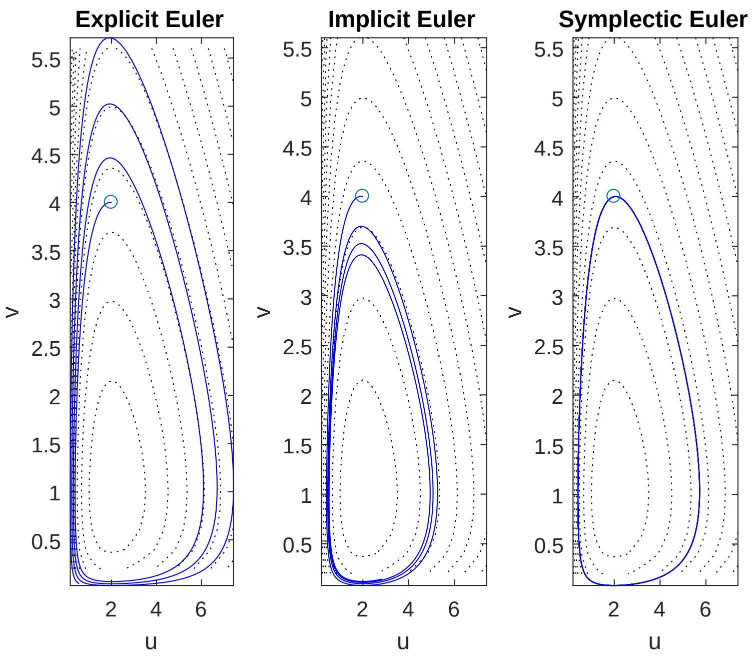


Figure 1.1: Solutions to the Lotka-Volterra Equation

The phase planes in Figure 1.1 (denoted by the dotted lines) convey that the solutions must be a closed loops and periodic - but only the solution using the Symplectic Euler scheme satisifies this criterion. The other two methods are qualitatively incorrect - the solution either spirals inwards or outwards.

Hence, it is important to study methods that preserve qualitative attributes of a system as well. Since these attributes often occur in differential geometry (the study of differential equations in geometry), we call them geometric invariants - thus lending the name ‘Geometric Numerical Integration’ to their study.

In this paper, we will focus on a specific class of geometric invariants - non linear, highly oscillatory Hamiltonian problems. We will devote our attention to the problem of molecular dynamics - specifically a simulation of argon molecules as done by A. Rahman [5], and L. Verlet [7]. We will first set up the molecular dynamics simulation, focusing on various techniques to improve upon its speed. We will then investigate the preservation of geometric invariants of this system in in the long run by the Velocity Verlet algorithm (the current industry standard for molecular dynamics simulation). We will conclude by extending this investigation to the family of Newmark-Beta methods, of which the Velocity Verlet algorithm is a special case.

2 Newmark Beta Methods

In this section, we look at a family of numerical methods used to solve second order ordinary differential equations. We look at the general formula, the algorithm to implement them, and an experimental verification of convergence order for different values of the parameters.

2.1 Introduction to Newmark Beta Methods

The Newmark Beta methods were presented by Nathan M. Newmark in a paper [4] published in 1959. They were intended to be used for find solutions to problems in structural dynamics, *"capable of application to structures of any degree of complication, with any relationship between force and displacement"*, with considerations for *"any time of dynamic loading such as that due to shock or impact, vibration, earthquake motion, or blast from a nuclear weapon"*. However, they can also be used in less apocalyptical scenarios - a special case known as the 'Velocity Verlet' algorithm is the standard for solving the equations of motion in molecular dynamics simulations. Writing $\mathbf{x}_i$, $\mathbf{v}_i$, and $\mathbf{a}_i$ for the displacement, velocity, and acceleration at timestep t_i respectively, the Newmark Beta methods are of the form

$$\begin{aligned}\mathbf{v}_{i+1} &= \mathbf{v}_i + h\left[(1-\gamma)\,\mathbf{a}_i + \gamma\mathbf{a}_{i+1}\right],\\ \mathbf{x}_{i+1} &= \mathbf{x}_i + h\mathbf{v}_i + \frac{h^2}{2}\left[(1-2\beta)\,\mathbf{a}_i + 2\beta\mathbf{a}_{i+1}\right],\end{aligned} \tag{2.1}$$

where $\gamma \in [0,1]$ and $2\beta \in [0,1]$. Note that the Newmark Beta methods are implicit if $\beta > 0$, as $\mathbf{a}_{i+1}$ depends upon the values of $\mathbf{x}_{i+1}$ and $\mathbf{v}_{i+1}$. If $\beta = 0$, the methods can be either explicit or implicit, depending on the situation in question. We consider this in greater detail in the next subsection.

These methods are second order accurate if and only if $\gamma = \frac{1}{2}$, and they are conditionally stable if and only if $2\beta \geq \gamma \geq \frac{1}{2}$. The proof of these attributes is beyond the scope of this paper, and can be found in [3]. However, we provide numerical evidence of the order of convergence in Subsection 2.2.

Some of the common choices of parameters β and γ (and the resulting schemes) are:

- Average Acceleration Method $\left(\gamma = \frac{1}{2},\ \beta = \frac{1}{4}\right)$:

$$\begin{aligned}\mathbf{v}_{i+1} &= \mathbf{v}_i + h\left[\frac{1}{2}\mathbf{a}_i + \frac{1}{2}\mathbf{a}_{i+1}\right],\\ \mathbf{x}_{i+1} &= \mathbf{x}_i + h\mathbf{v}_i + \frac{h^2}{2}\left[\frac{1}{2}\mathbf{a}_i + \frac{1}{2}\mathbf{a}_{i+1}\right].\end{aligned}$$

- Linear Acceleration Method $\left(\gamma = \frac{1}{2},\ \beta = \frac{1}{6}\right)$:

$$\begin{aligned}\mathbf{v}_{i+1} &= \mathbf{v}_i + h\left[\frac{1}{2}\mathbf{a}_i + \frac{1}{2}\mathbf{a}_{i+1}\right],\\ \mathbf{x}_{i+1} &= \mathbf{x}_i + h\mathbf{v}_i + \frac{h^2}{2}\left[\frac{2}{3}\mathbf{a}_i + \frac{1}{3}\mathbf{a}_{i+1}\right].\end{aligned}$$

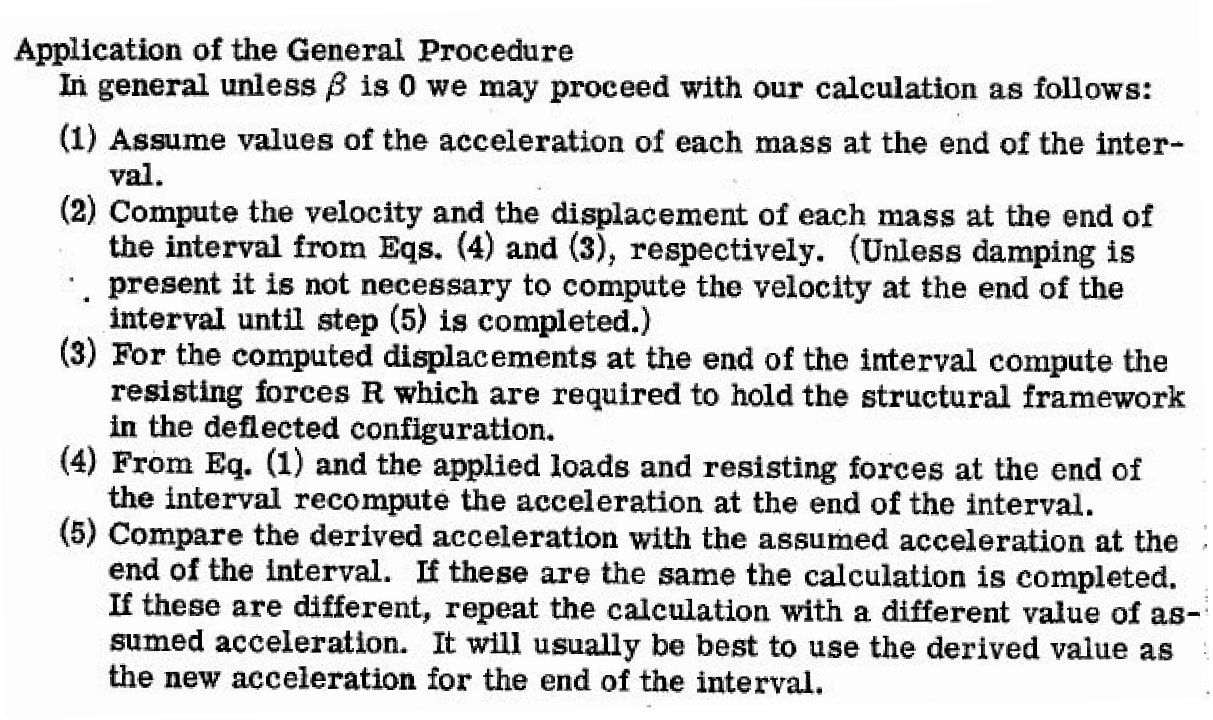
- Velocity Verlet Method $\left(\gamma = \frac{1}{2},\, \beta = 0\right)$:

$$\begin{aligned} \mathbf{v}_{i+1} &= \mathbf{v}_i + h\left[\frac{1}{2}\mathbf{a}_i + \frac{1}{2}\mathbf{a}_{i+1}\right], \\ \mathbf{x}_{i+1} &= \mathbf{x}_i + h\mathbf{v}_i + \frac{h^2}{2}\mathbf{a}_i. \end{aligned} \tag{2.2}$$

2.2 Computational Application of Newmark Beta Methods

Figure 2.1 shows an excerpt from the original manuscript [4] where Newmark outlines an algorithm to calculate the next values. The method suggested belongs to a class of methods known as 'predictor-corrector' methods - an initial 'prediction' of the quantity at t_{i+1} is calculated using some function, and then this value is 'corrected' by using the prediction as the initial value for some other function to calculate the value of the quantity at the same point in time, t_{i+1}. The example below depicts the Heun method for an ODE of the form $y' = f(t,\, y)\,, \quad y(t_0) = y_0$.

$$\begin{aligned} \tilde{y}_{i+1} &= y_i + hf(t_i,\, y_i)\,, && \text{prediciting using Forward Euler Method} \\ y_{i+1} &= y_i + \frac{h}{2}\left(f(t_i, y_i) + f(t_{i+1}, \tilde{y}_{i+1})\right). && \text{correcting using Trapezium Rule} \end{aligned}$$


Application of the General Procedure

In general unless β is 0 we may proceed with our calculation as follows:

(1) Assume values of the acceleration of each mass at the end of the interval.

(2) Compute the velocity and the displacement of each mass at the end of the interval from Eqs. (4) and (3), respectively. (Unless damping is present it is not necessary to compute the velocity at the end of the interval until step (5) is completed.)

(3) For the computed displacements at the end of the interval compute the resisting forces R which are required to hold the structural framework in the deflected configuration.

(4) From Eq. (1) and the applied loads and resisting forces at the end of the interval recompute the acceleration at the end of the interval.

(5) Compare the derived acceleration with the assumed acceleration at the end of the interval. If these are the same the calculation is completed. If these are different, repeat the calculation with a different value of assumed acceleration. It will usually be best to use the derived value as the new acceleration for the end of the interval.

Figure 2.1: Excerpt on applying the Newmark Beta methods computationally

However, this raises a question: how many times should the cycle of prediction and correction be run to get an accurate result? We can avoid this question by simplifying our assumptions. Since the goal of this paper is to apply the Newmark Beta methods to a molecular dynamics simulation, we can tune the method accordingly. In the absence of external forces, the molecular dynamics simulation is a conservative system. So the force (and by Newton's Second Law, the acceleration) is dependent only on the displacement of the particles. Thus, we can re-write Equation 2.1 as

$$\mathbf{v}_{i+1} = \mathbf{v}_i + h\left[(1-\gamma)\,\mathbf{a}_i + \gamma\mathbf{a}_{i+1}(\mathbf{x}_{i+1})\right], \tag{2.3}$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i + h\mathbf{v}_i + \frac{h^2}{2}\left[(1-2\beta)\,\mathbf{a}_i + 2\beta\mathbf{a}_{i+1}(\mathbf{x}_{i+1})\right]. \tag{2.4}$$

as we will already have computed the values of $\mathbf{x}_i, \mathbf{v}_i,$ and $\mathbf{a}_i$.

Thus, we can see that the equation is implicit only for $\mathbf{x}_{i+1}$. However, it is explicit for $\mathbf{v}_{i+1}$, as we can calculate $\mathbf{a}_{i+1}$ once we have the value of $\mathbf{x}_{i+1}$. We can use Newton-Raphson Iteration to solve for $\mathbf{x}_{i+1}$ in Equation 2.4

$$\begin{aligned}\tilde{\mathbf{F}}(\mathbf{x}_{i+1}) &= \mathbf{x}_{i+1} - \mathbf{x}_i - h\left(\mathbf{v}_i + \frac{h}{2}\left((1-2\beta)\,\mathbf{a}_i + 2\beta\mathbf{a}_{i+1}(\mathbf{x}_{i+1})\right)\right), \\ D_{\mathbf{x}_{i+1}}\tilde{\mathbf{F}}(\mathbf{x}_{i+1}) &= \mathbf{I} - \beta h^2 \mathbf{J}_{\mathbf{a}_{i+1}}, \\ \mathbf{x}_{i+1} &= \mathbf{x}_i - D\tilde{\mathbf{F}}(\mathbf{x}_i)^{-1}\,\tilde{\mathbf{F}}(\mathbf{x}_i).\end{aligned} \tag{2.5}$$

where $\mathbf{J}_{\mathbf{a}_{i+1}}$ is the Jacobian of the acceleration function at time t_{i+1}. This equation requires that the inverse of $D\tilde{\mathbf{F}}(\mathbf{x}_i)$ exists. Then, we use the following algorithm to find the next set of points:

Algorithm 2.1: Newmark-Beta Method Algorithm

```
input  : $\mathbf{x}_i, \mathbf{v}_i, \mathbf{a}_i, h, \gamma, \beta$
output: $\mathbf{x}_{i+1}, \mathbf{v}_{i+1}, \mathbf{a}_{i+1}$
$\mathbf{v}_{i+1} \leftarrow \mathbf{v}_i + h(1-\gamma)\,\mathbf{a}_i$ ;        // Updating the explicit part of the equation
Calculate $\mathbf{x}_{i+1}$ using the Newton-Raphson Method
$\mathbf{a}_{i+1} \leftarrow \frac{1}{m}\mathbf{F}(\mathbf{x}_{i+1})$
$\mathbf{v}_{i+1} \leftarrow \mathbf{v}_{i+1} + \gamma h \mathbf{a}_{i+1}$ ;        // Updating the implicit part of the equation
```

Now that we have the algorithm laid out, we can look at applying it to an example and test the convergence of Newmark Beta methods.

2.3 Convergence Tests

We consider the case of a pendulum, governed by the equation:

$$\frac{d^2\theta}{dt^2} = -\frac{g}{l}\sin(\theta) \tag{2.6}$$

We test for convergence in the following way: we fix $T = 5$, β and γ, and consider $N = 16, 32, 64, 128, 256, 512,$ and 1024 $\left(\text{hereby varying } h := \frac{T}{N}\right)$. Then, for each value of h, we calculate the position and velocity of the pendulum at $T = 5$, and compare these values to the reference solution calculated using MATLAB's `ode45` function (with absolute and relative tolerances of 10^{-12}). We define the overall error in the estimation at time t_i as a function of the error in displacement and velocity at t_i:

$$err_i = \sqrt{\left(err_{x_i}\right)^2 + \left(err_{v_i}\right)^2}$$

Looking at the Taylor series expansion of the analytical solution, we get that $err_i \approx \mathcal{O}(h^a)$ for some $a \in \mathbb{Z}^+$. Hence, we get that a - the algebraic order of convergence - is the slope of the straght line when $\log(err_i)$ is plotted against $\log(h)$. We can see that Figure 2.2 corroborates the statement in Subsection 2.1 - the order of a Newmark Beta method is 1 unless $\gamma = 0.50$, when it is 2.

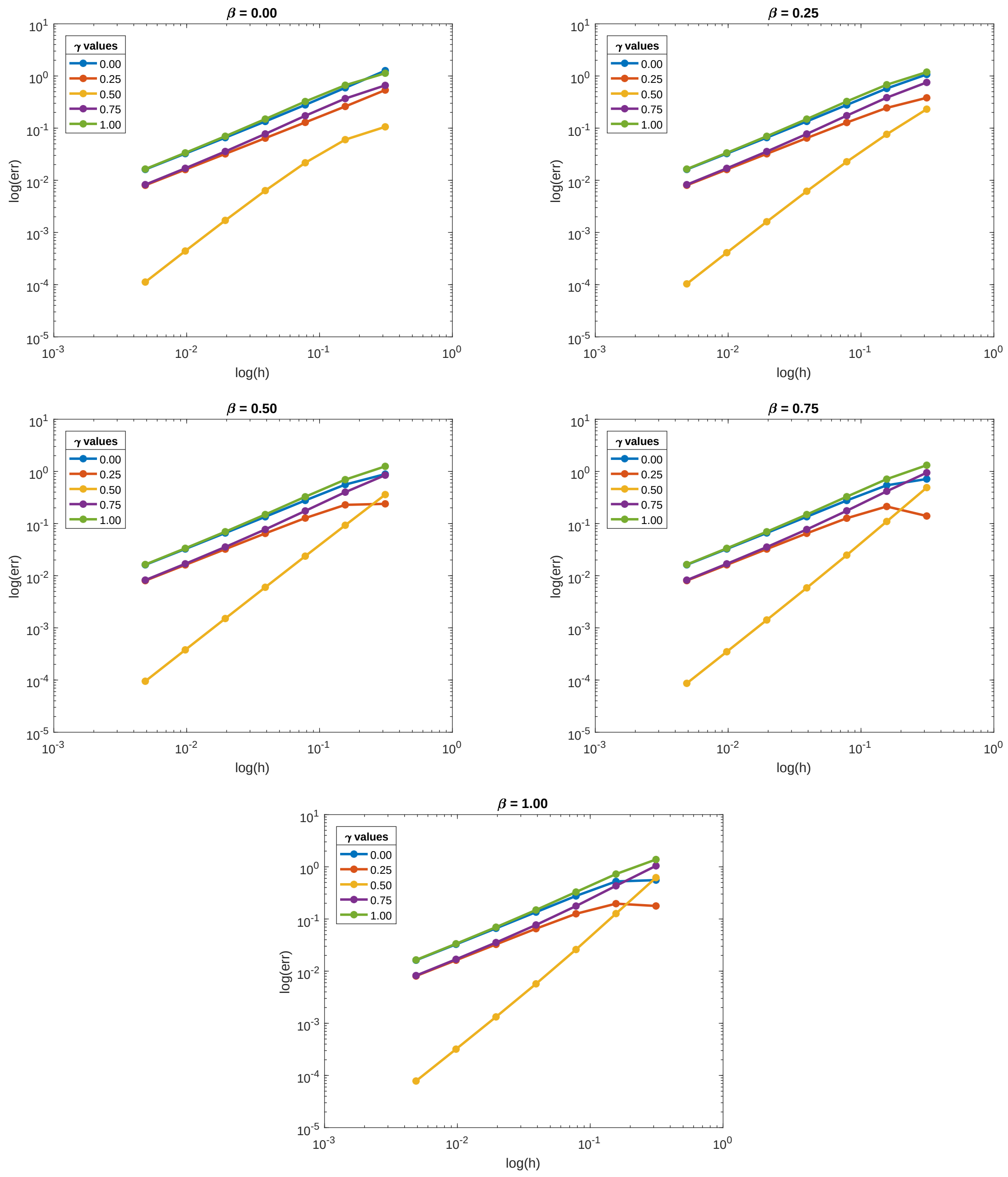


Figure 2.2: Order of convergence of Newmark Beta methods for varying parameter values

3 Molecular Dynamics

One of the most seminal works in Molecular Dynamics was Dr. Aneesur Rahman's 1964 paper [5] "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 [7] by Dr. Loup Verlet, which introduced the Verlet Integration method for the same model of 864 argon atoms.

In this section, we will look at replicating the aforementioned simulation for a system of 864 (a seemingly arbitrary choice) atoms of argon. Once set up, we will use the Velocity Verlet method (a modern variant of Verlet Integration) to calculate the motion of the particles, and observe how quantities such as temperature and local pressure behave over a long time period.

3.1 Hamiltonian Equation

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

$$H(p,q) = \frac{1}{2}\sum_{i=1}^{N} m_i^{-1} p_i^T p_i + \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} V_{ij}\left(\|q_i - q_j\|\right), \tag{3.1}$$

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 3.1 as

$$\dot{q} = \nabla_p H = M^{-1}p, \qquad \dot{p} = -\nabla_q H = -V'(q), \tag{3.2}$$

where $M = \operatorname{diag}(m_1\mathbf{I}, \ldots, m_N\mathbf{I})$, and $\mathbf{I}$ is the 3×3 identity matrix. This is equivalent to the typical second order differential equation of motion $\ddot{q} = f(q)$, with $f(q) = M^{-1}V'(q)$.

3.2 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

$$V(\mathbf{r}_{ij}) = 4\varepsilon\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right], \tag{3.3}$$

where $\mathbf{r}_{ij}$ is the displacement between the i^{th} and the j^{th} atoms, $r_{ij} = \|\mathbf{r}_{ij}\|_2$, ε is the potential well depth (a measure of strength of attraction of two atoms), and σ 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

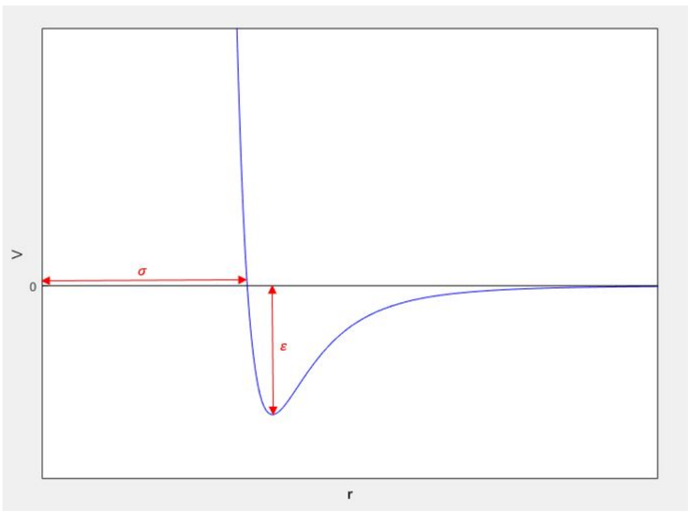


Figure 3.1: Lennard-Jones Potential

it was found to suffice for modelling argon atoms in [5]. 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 the next subsection.

3.3 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

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

. Hence, by the chain rule, we get that

$$\begin{aligned}\mathbf{f}\left(\mathbf{r}_{ij}\right) &= -4\varepsilon\left[-12\sigma^{12}\left(\frac{1}{r_{ij}}\right)^{13} + 6\sigma^{6}\left(\frac{1}{r_{ij}}\right)^{7}\right] \times \frac{\mathbf{r}_{ij}}{r_{ij}} \\ &= 48\varepsilon\left[\sigma^{12}\left(\frac{1}{r_{ij}}\right)^{14} - \frac{\sigma^{6}}{2}\left(\frac{1}{r_{ij}}\right)^{8}\right]\mathbf{r}_{ij}. \qquad (3.4)\end{aligned}$$

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 $\mathbf{f}_{ij} = -\mathbf{f}_{ji}$.

3.4 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 σ^*, where $1\,\sigma^* = 3.40 \times 10^{-10}$ m, and energy in terms of ε^*, where $1\,\varepsilon^* = 1.65 \times 10^{-21}$ J. Then, the Lennard-Jones potential can be calculated as

$$V\left(\mathbf{r}_{ij}\right) = 4\left[\left(\frac{1}{r_{ij}}\right)^{12} - \left(\frac{1}{r_{ij}}\right)^{6}\right], \qquad (3.5)$$

where $\mathbf{r}_{ij}$ and r_{ij} have the same meaning as before, but are expressed as a multiple of σ. Then, we can rewrite Equation 3.4 as

$$\mathbf{f}(\mathbf{r}_{ij}) = 48\left(\frac{1}{r_{ij}}\right)^8\left[\left(\frac{1}{r_{ij}}\right)^6 - \frac{1}{2}\right]\mathbf{r}_{ij}. \tag{3.6}$$

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 (6.69×10^{-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* = 3.125×10^{-13} s. Then a step-size of 10 femtoseconds is equivalent to $h = 0.032$. 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 3.1.

Physical Quantity	Reduced Unit	Value
Length	σ^*	3.40×10^{-10} m
Energy	ε^*	1.65×10^{-21} J
Mass	m*	6.69×10^{-23} kg
Time	s*	3.125×10^{-13} s
Temperature	K*	119.6 K

Table 3.1: Reduced Units and their Values

3.5 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 lead to 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 $1.374 \ \mathrm{g}\cdot\mathrm{cm}^{-3}$, we get the following cube lattice

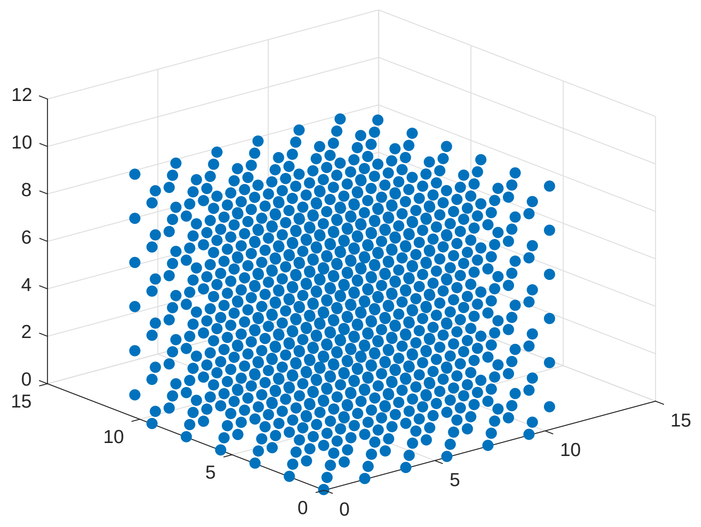


Figure 3.2: Initial cubic lattice

3.6 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 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.

3.7 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

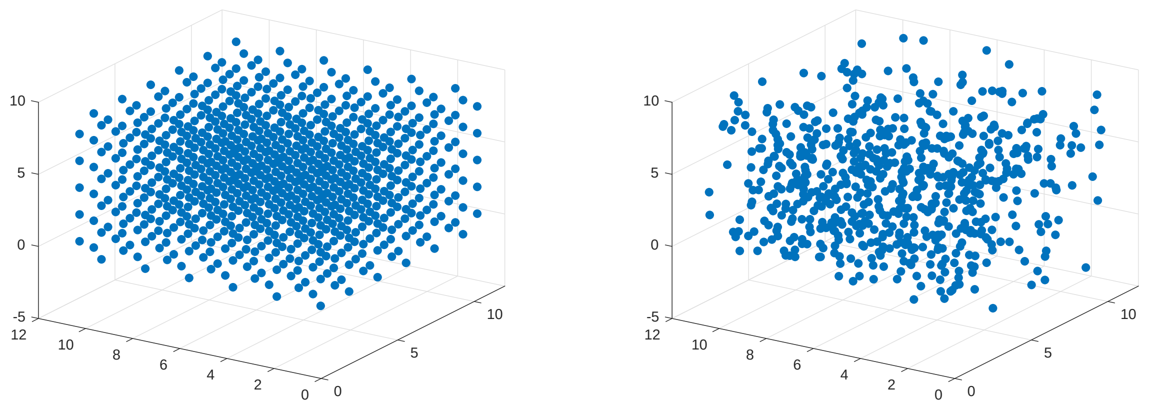


Figure 3.3: Change in the position of particles after 500 steps of equilibration

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 [6]. 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 3.4. This has two effects:

- Say we have to boxes beside each, Box A and Box B (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 the Minimum Image Criterion using the algorithm below.

Algorithm 3.1: Minimum Image Convention in 1-d

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

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

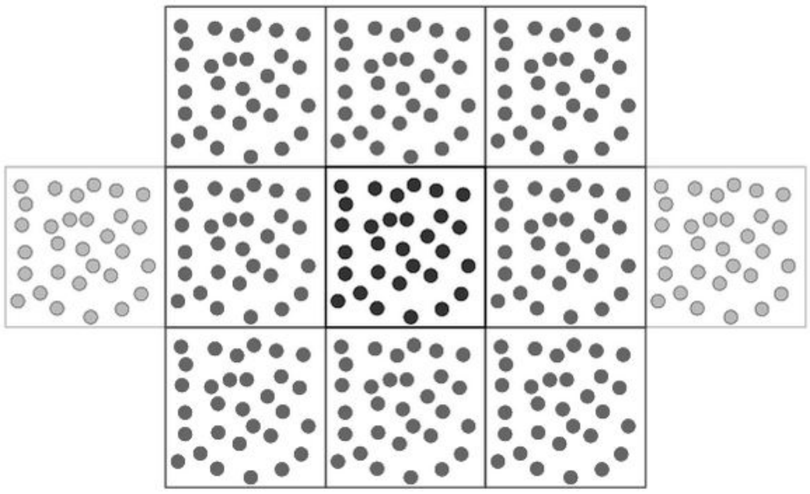


Figure 3.4: Periodic Boundary Conditons in 2-d

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.

3.8 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 3.1, 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. This is equivalent to shifting the Lennard-Jones Potential

$$V\left(\mathbf{r}_{ij}\right) = \begin{cases} 4\left[\left(\left(\frac{1}{r_{ij}}\right)^{12} - \left(\frac{1}{r_{ij}}\right)^{6}\right) - \left(\left(\frac{1}{r_c}\right)^{12} - \left(\frac{1}{r_c}\right)^{6}\right)\right] & , \quad r_{ij} \leq r_c \\ 0 & , \quad otherwise \end{cases} \tag{3.7}$$

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, $r_c \leq \frac{L}{4}$[1]. In our simulations, we have $L = 10.229\,\sigma^*$; so, we define $r_c = 2.5\,\sigma^*$. We substitute this into Equation 3.5 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 computing power 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 3.2 make the case for using the same list for over multiple iterations (5$\times$ 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 3.2: Results of tests to find optimal N_n

References

[1] Daan Frenkel and Berend Smit. *Understanding Molecular Simulation*. Academic Press, 2001.

[2] Ernst Hairer, Gerhard Wanner, and Christian Lubich. *Examples and Numerical Experiments*, pages 1–26. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[3] Nathan M. Newmark. Computation of dynamic structural response in the range approaching failure. In *Proceedings of the Symposium on Earthquake and Blast Effects on Structures*, Los Angeles, 1952. Earthquake Engineering Research Institute.

[4] Nathan M. Newmark. A method of computation for structural dynamics. *American Society of Civil Engineers*, EM 3, 1959.

[5] Aneesur Rahman. Correlations in the motion of atoms in liquid argon. *Physical Review*, 136(2A):405–411, 1964.

[6] D. C. Rapaport and Dennis Rapaport. *The Art of Molecular Dynamics Simulation*. Cambridge University Press, 2004.

[7] Loup Verlet. Computer experiments on classical fluids. i. thermodynamical properties of lennard -jones molecules. *Physical Review*, 159(1):98–103, 1967.