MAT 128C - Program # 2 Molecular Dynamics

Douglas Sherman, Aaron Millstein, Douglas Kubota

June 2nd, 2017

---

1 Introduction

We propose a simulation of N particles in a box with periodic boundary conditions (torus) by following Newton's Law of Motion based on the Leonard Jones potential energy function of

$$E(r_{ij}) = 4\epsilon \left(\left(\frac{A}{r_{ij}} \right)^1 2 - \left(\frac{B}{r_{ij}} \right)^6 \right) \tag{1}$$

for some constants ϵ, A, and B due to the particles in question. Newton's Law $\vec{F} = m\vec{a}$ with (1) gives us the differential equation as

$$\begin{aligned} -\frac{dE}{dr} &= m\frac{d^2r}{dt^2} \\ \iff 4\epsilon \left(12 \left(\frac{A}{r_{ij}} \right)^{13} - 6 \left(\frac{B}{r_{ij}} \right)^7 \right) &= m\frac{d^2r}{dt^2} \end{aligned} \tag{2}$$

This study considers Argon atoms with a Van der Waals radius of $\sigma := 3.4\dot{A}$ and simplify the problem as $A = B = \sigma$ and $\epsilon = \sigma = 1$. Thus we obtain dE/dr as

$$f_{ij} := \frac{dE}{dr_{ij}} = 24 \left(\frac{2}{r_{ij}^{13}} - \frac{1}{r_{ij}^7} \right)$$

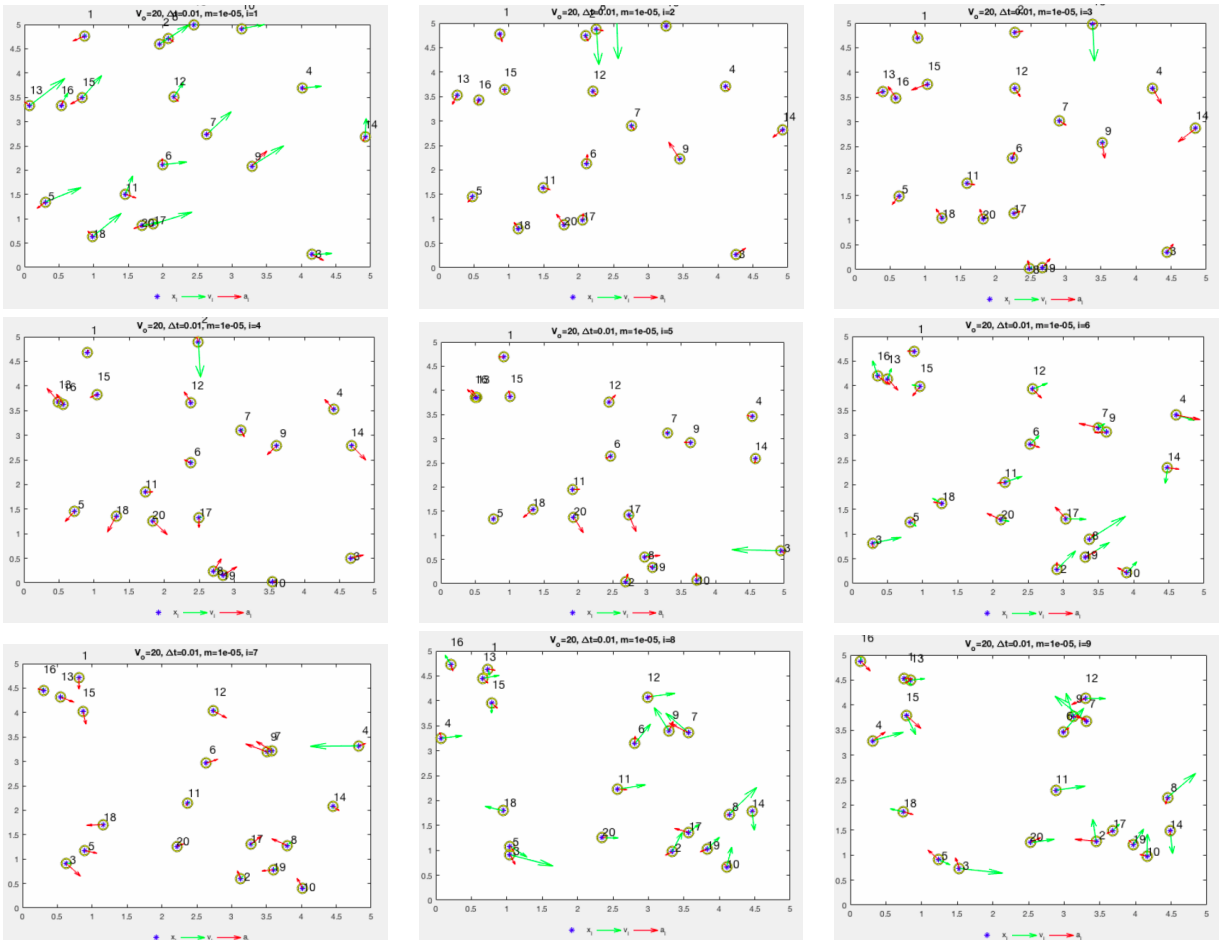
2 Results

2.1 Simulate the Molecular Dynamics of N particles in an L × L torus.

We initialize the system with N particles and assign initial velocities given by $v_0 * \nu$ where ν is random Gaussian scale factor. Then using the potential function above we compute the acceleration at each Δt interval as the sum of all the forces applied to each particle by the remaining $N - 1$ particles. We compute the x and y components seperately using the geometry of the system as

$$f_{ij,x} = f_{ij} \cos\theta = f_{ij} \frac{x_j - x_i}{r_{ij}}$$
$$f_{ij,y} = f_{ij} \sin\theta = f_{ij} \frac{y_j - y_i}{r_{ij}}$$

The following illustrates the evolution of a system using $N = 20, v_0 = 20, \Delta t = 0.01$, and $L = 5$.



The images above following the order of left to right then top to bottom. The green arrows indicate velocity and the red arrows indicate acceleration for each of the particles. Please visit the GitHub repository at https://github.com/dsherma7/Computing/tree/master/MolecularDynamics/gif[1] to see .gif files illustrating the evolution of systems with various v_0, N, and Δt values to properly illustrate the dynamics of the Argon particles.

2.1 Reducing Complexity

In order the reduce the complexity we make one assumption and one observation. Firstly, since f_{ij} is the force on particle i by particle j, it must be true that $|f_{ji}| = |f_{ij}|$. Thus we can store the computed f_{ij} for $i < j$ and replace f_{ij} for $i > j$ with these stored values after accounting for the change in direction of the force. Moreover, since the force decays with distance exceptionally fast we assume that $f_{ij} = 0$ for particles with distance greater than 3σ, or 3 Van der Waal's radii. The MATLAB code at the end of this report illustrates these adjustments for $f_{ij} = f(x)$

[1]This url contains multiple .gif files for various v_0, N, and Δt values. It is a public GitHub repository, and is a far better representation of the system's dynamics than a few still frame iterations.

2.2 Distribution of Particles

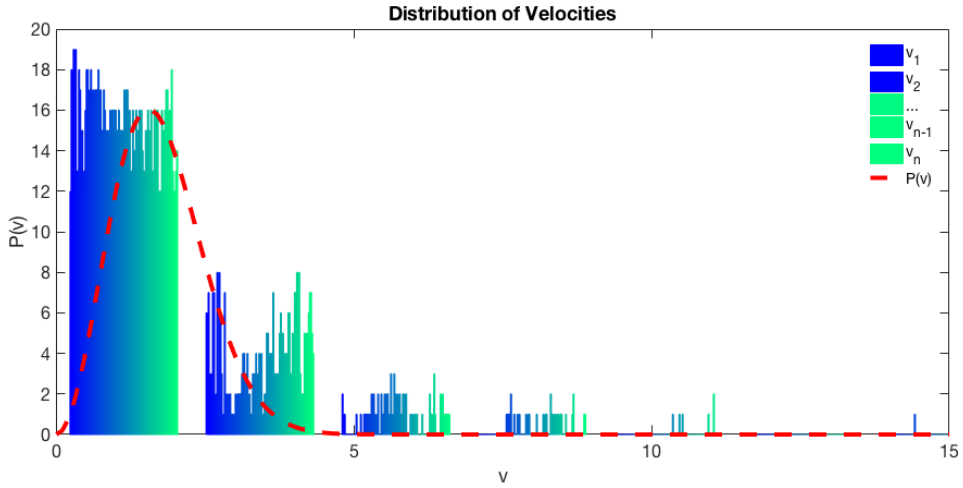
In Molecular Dynamics the distribution of speeds is defined by the following function

$$P(v) = C\frac{v^2}{K_B T}\exp\frac{-mv^2}{K_B T}$$

and by the equipartition theorem we can compute $K_B T$ through the kinetic energy, and without knowing the temperature as

$$K_B T = \frac{m}{2}\left\langle v_x^2 + v_y^2\right\rangle$$

where $\langle x\rangle = E[x]$ is the expected value of x. We compared the distribution of the standardized velocity values obtained by the system shown in part (2) to the value for $P(v)$ as shown below.



1. The comparison of the known distribution of speeds for a set of particles to the actual distribution of speeds.

We see that the velocities peak at about 1 standard deviation, and then decay rapidly. This behavior mimics the expected distribution of speeds as defined in the equation above.

3 Code

The MATLAB code used in this assignment is given below

```
function [fj,fs] = f(dr,fs,j)
% Computes the force, f_ij, between particles i and j
% using the derivative of the Leonard Jones potential.
% Note: Also takes advantage of the symmetry |f_ij|=|f_ji|.
%
% INPUT:
%   dr: The magnitude of distance between each particle with x_j (dx|dy)
%   fs: The stored force evaluations of previous iterations of f(*)
%   j:  The current iteration of force evaluations for x_j
%
% OUTPUT:
%   fi: The resulting force evaluation on x_j
%   fs: The updated stored force evaluations

    fj = [fs(j,1:j-1)';24*((2./dr(j:end).^(13))-(1./dr(j:end).^7))];
    fs(:,j) = [fj(1:j-1);0;fj(j:end)];
end
```

```
function ai = a(xi,yi,L)
% Computes the acceleration for all particles p_i at locations
% (x_i,y_i) inside a box L with periodic boundary conditions.
%
% INPUT:
%   xi,yi: Coordinates (x,y) of N particles
%   L: Side length of square box
%
% OUTPUT:
%   ai: Acceleration of each particle by Newtons Law a=F/m

    n = size(xi,1);
    ai = zeros(n,1);
    fs = zeros(n,n);
    for j = 1:n
        x = xi([1:j-1,j+1:end]); y = yi([1:j-1,j+1:end]);
        sx = sign(x-xi(j)); sy = sign(y-yi(j));
        dx = [x-xi(j),sx.*(abs(x-xi(j))-L)];
        dy = [y-yi(j),sy.*(abs(y-yi(j))-L)];
        [dx,ix] = max(abs(dx),[],2);
        [dy,iy] = max(abs(dy),[],2);
        dx = sx.*(-1).^(ix).*dx;
        dy = sy.*(-1).^(iy).*dy;
        [fi,fs] = f(abs(dx),fs,j);
        ai(j) = sum(fi.*dx./(dx.^2+dy.^2));
    end
end
```

```
function plot_points(xi,yi,vxi,vyi,axi,ayi,L)
% Illustrates N = size(xi) particles by displaying each
% along with their associated velocity/acceleration vectors.
%
% INPUT:
%   xi,yi: Coordinates (x,y) of each particles' center
%   vxi,vyi: x and y components of each particles' velocity
%   axi,ayi: x and y components of each particles' acceleration
%   L: Side length of square box containing particles

    h(1)=plot(xi,yi,'*b'); hold on; axis([0, L, 0, L]);
    h(2)=plot(xi,yi,'ok','MarkerSize',12,'LineWidth',2);
    h(3)=plot(xi,yi,'oy','MarkerSize',12);
    h(4)=quiver(xi,yi,vxi,vyi,.5,'Color','g');
    h(5)=quiver(xi,yi,axi,ayi,.25,'Color','r');
    text(xi,yi*1.1,cellstr(int2str([1:length(xi)]')),'FontSize',14)
    lgd = legend(h([1,4,5]),'x_i','v_i','a_i');
    lgd.Box = 'off';
    lgd.Location = 'southoutside';
    lgd.Orientation = 'horizontal';
end
```

```
function print_gif(filename,v0,dt,m,i)
% Prints the current frame to a .gif file specified in filename.
%
% INPUT:
%   filename: The path where the gif should be written
%   v0,dt,m: Parameter values for image title
%   i: ith slide of the .gif file

    title(strjoin({'V_o=',num2str(v0),', \Deltat=',num2str(dt),...
                   ', m=',num2str(m),', i=',int2str(i)},''));
    drawnow; frame = getframe(1); im = frame2im(frame);
    [imind,cm] = rgb2ind(im,256);
    if (i==1) imwrite(imind,cm,filename,'gif', 'Loopcount',inf);          % Initialize .gif
    else      imwrite(imind,cm,filename,'gif', 'WriteMode','append');end % Add slides
end
```