APIE - Molecular Dynamics of Fluids

Bram ter Huurne (s1491784)

February 5, 2020

All exercises of Molecular Dynamics of Fluids, part of the APIE course. Code can be found on GitHub.

Exercise (a)

For $\frac{d\varphi(r_{ij})}{dr_{ij}}$:

$$\begin{aligned}\frac{d\varphi(r_{ij})}{dr_{ij}} &= \begin{cases} \frac{d\varepsilon(\sigma - r_{ij})^2}{dr_{ij}} & r_{ij} \leq \sigma \\ 0 & r_{ij} > \sigma \end{cases} \\ &= \begin{cases} -2\varepsilon(\sigma - r_{ij}) & r_{ij} \leq \sigma \\ 0 & r_{ij} > \sigma \end{cases}\end{aligned}$$

For $\frac{dr_{ij}}{d\mathbf{x}_i}$:

$$\begin{aligned}\frac{dr_{ij}}{d\mathbf{x}_i} &= \frac{d|\mathbf{x}_i - \mathbf{x}_j|}{d\mathbf{x}_i} \\ &= \frac{d}{d\mathbf{x}_i}\sqrt{(\mathbf{x}_i - \mathbf{x}_j)^2} \\ &= \frac{(\mathbf{x}_i - \mathbf{x}_j)}{\sqrt{(\mathbf{x}_i - \mathbf{x}_j)}} \qquad (\mathbf{x}_i \neq \mathbf{x}_j) \\ &= \frac{(\mathbf{x}_i - \mathbf{x}_j)}{|\mathbf{x}_i - \mathbf{x}_j|} \\ &= \frac{\mathbf{r}_{ij}}{|\mathbf{r}_{ij}|} = \frac{\mathbf{r}_{ij}}{r_{ij}}\end{aligned}$$

Exercise (b)

We first initialize the positions, outside the force field of the wall (see (c)). Then the permutation landscape of all particle collisions is created with the `meshgrid` function.

Next, within the loop, the vector-distances are calcaluted from which the euclidean distance is calculated.

Then the forces are calculated using both equations of (a),

1. $f_{ij} \leftarrow -\dfrac{d\varphi(r_{ij})}{dr_{ij}} = 2\varepsilon(\sigma - r_{ij}),$
2. $f_{ij} \leftarrow f_{ij} \cdot (r_{ij} \leq \sigma),$
3. $\mathbf{f}_{ij} \leftarrow f_{ij}\dfrac{\mathbf{r}_{ij}}{r_{ij}},$
4. $\mathbf{f}_i \leftarrow \sum_{j=1}^{N} \mathbf{f}_{ij}.$

```
%% Beginning of script %%

% particle properties
N = 150;
Eps = 100;
mass = 1;
Sigma = 1;

% initialize positions and permutation landscape
x = (L-2*Sigma_wall) * (rand(N, 2) - 0.5);
[i,j] = meshgrid(1:N, 1:N);


%% In loop %%
% particle distances
r_ij(:, :, 1) = reshape(x(i,1) - x(j,1), N, N);
r_ij(:, :, 2) = reshape(x(i,2) - x(j,2), N, N);
r_ij_abs = sqrt(r_ij(:,:,1).^2 + r_ij(:,:,2).^2);

% forces on particles
f_ij = calculate_force(r_ij,r_ij_abs,Sigma,Eps);
f_i = squeeze(sum(f_ij .* ~eye(N),1));
```

```
function f_ij = calculate_force(r_ij,r_ij_abs,Sigma,Eps)
   f_ij = 2.*Eps.*(Sigma - r_ij_abs).*(r_ij_abs <= Sigma) .* r_ij ./ r_ij_abs;
   f_ij(isnan(f_ij)) = 0;
end
```

Exercise (c)

I choose to use the soft-wall boundary. This is the cumulative effect of all the walls at the (continuous) locations:

$$
\begin{aligned}
&(x = -L/2,\ y)\\
&(x = L/2,\ \ y)\\
&(x,\ \ y = -L/2)\\
&(x,\ \ y = L/2)
\end{aligned}
$$

We choose a higher ε for the wall to prevent particles from penetrating the boundary.

```

%% Beginning of script

% wall properties
L = 10;
Eps_wall = 500;
Sigma_wall = 1;


%% In loop %%
% wall interaction
r_ij_wall(:, 1) = x(:,1) - L/2;
r_ij_wall(:, 2) = zeros(size(r_ij_wall(:, 1)));
r_ij_wall_abs(:,:,1) = abs(r_ij_wall(:, 1));
f_i_wall = calculate_force(r_ij_wall,r_ij_wall_abs(:,:,1),Sigma_wall,Eps_wall);

r_ij_wall(:, 1) = x(:,1) + L/2;
r_ij_wall(:, 2) = zeros(size(r_ij_wall(:, 1)));
r_ij_wall_abs(:,:,2) = abs(r_ij_wall(:, 1));
f_i_wall = f_i_wall + calculate_force(r_ij_wall,r_ij_wall_abs(:,:,2),Sigma_wall
    ,Eps_wall);

r_ij_wall(:, 2) = x(:,2) - L/2;
r_ij_wall(:, 1) = zeros(size(r_ij_wall(:, 2)));
r_ij_wall_abs(:,:,3) = abs(r_ij_wall(:, 2));
f_i_wall = f_i_wall + calculate_force(r_ij_wall,r_ij_wall_abs(:,:,3),Sigma_wall
    ,Eps_wall);

r_ij_wall(:, 2) = x(:,2) + L/2;
r_ij_wall(:, 1) = zeros(size(r_ij_wall(:, 2)));
r_ij_wall_abs(:,:,4) = abs(r_ij_wall(:, 2));
f_i_wall = f_i_wall + calculate_force(r_ij_wall,r_ij_wall_abs(:,:,4),Sigma_wall
    ,Eps_wall);

% total force
f_i = f_i + f_i_wall;
```

Exercise (d)

The Verlet algorithm is implemented as:

```
tmp = x;
[x,v] = verlet(x,x_old,f_i,dt,mass);
x_old = tmp;
```

```
function [x_out,v_out] = verlet(x_in,x_in_old,f_i,dt,mass)
   x_out = 2 .* x_in - x_in_old + f_i.*dt^2./mass;
   v_out = (x_out - x_in) ./ dt;
end
```

Alternatively, when keeping track of the positions over time, we can use

```
if step == 1
   [x(:,:,step+1),v] = verlet(x(:,:,1),x(:,:,1),f_i,dt,mass);
else
   [x(:,:,step+1),v] = verlet(x(:,:,step),x(:,:,step-1),f_i,dt,mass);
end
```

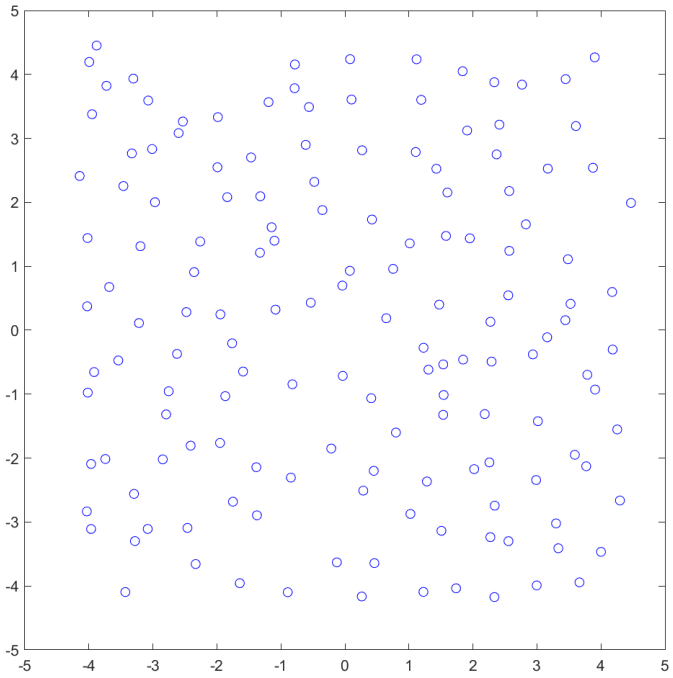


Figure 1: Particles in scatter plot.

Exercise (e)

We have the main conservation of energy in our system containing the kinetic and potential energy, $E_{tot} = E_{kin} + E_{pot}$. Additionally, the conservation of momentum could be included (scalar and vector wise), but due to the soft boundary, this is only valid when averaging over time, such that no momentum is lost on average.

The potential energy is calculated using the equation in the exercise,

$$E_{pot} = \Phi = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} \varepsilon(\sigma - r_{ij})^2.$$

The kinetic energy is given by

$$E_{kin} = \frac{1}{2} m \sum_{i=1}^{N} |\mathbf{v}_i|^2.$$

The total absolute momentum is given by

$$p = m \sum_{i=1}^{N} |\mathbf{v}_i|.$$

```
% calculate energies
phi_ij = Eps .* ( Sigma - r_ij_abs).^2 .*(r_ij_abs <= Sigma) .* ~eye(N);
phi_wall = Eps_wall .* ( Sigma_wall - r_ij_wall_abs).^2 .*(r_ij_wall_abs <=
    Sigma_wall);
E_pot(step) = 0.5 * squeeze(sum(sum(phi_ij))) + squeeze(sum(sum(phi_wall)));
v_abs_sqr(:,step) = v(:,1).^2 + v(:,2).^2;
E_kin(step) = sum(0.5 .* mass .* v_abs_sqr(:,step));
Momentum_tot(step) = sum(v_abs_sqr(:,step) * mass);
```

When choosing the integration time step size, the total energy should not increase or decrease. Choosing $\Delta t = 10^4$ gives stable results. Optionally, this value can be higher, e.g. $\Delta t = 10^3$, but this gives small stochasticity in the output (less smooth).

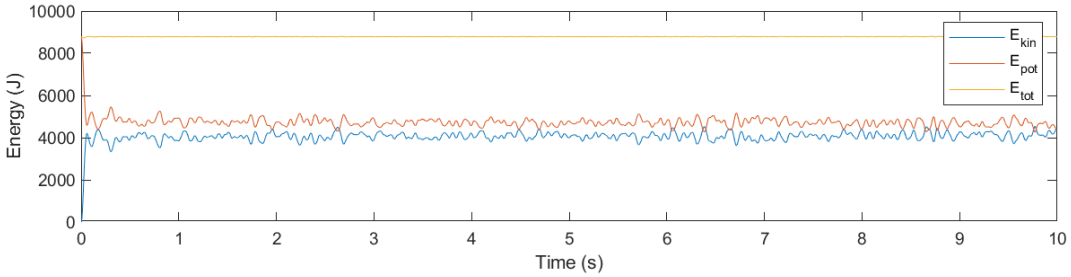


Figure 2: Energy with step size $\Delta t = 10^3$ and 150 particles.

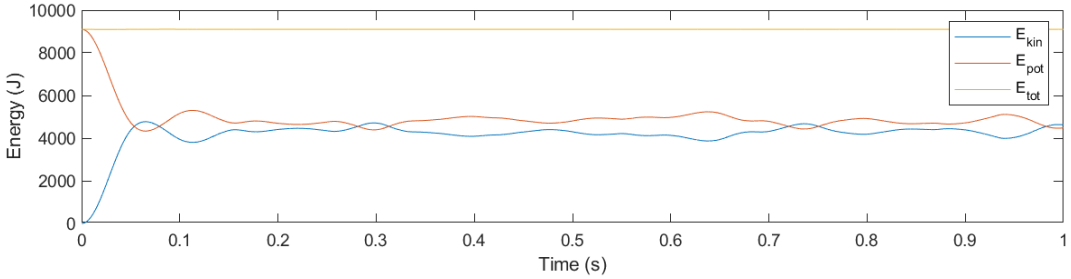


Figure 3: Energy with step size $\Delta t = 10^4$ and 150 particles.

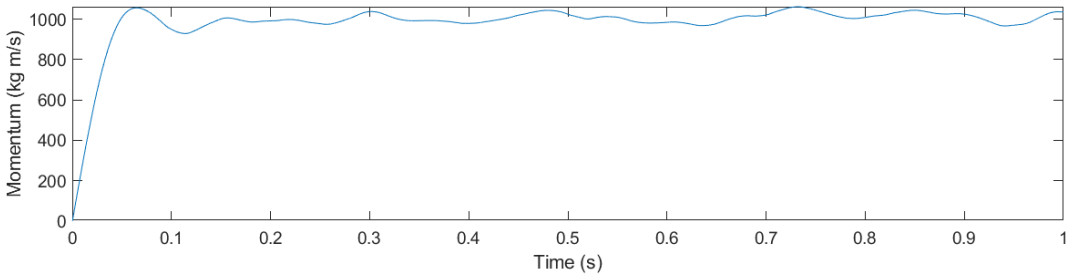


Figure 4: Total momentum with step size $\Delta t = 10^4$ and 150 particles.

Exercise (f)

We incorporate the temperature and pressures according to the handout, where we take $P_{total} = P_{wall} + P_{virial}$. We choose a trailing moving window of 100 stepsizes to smoothen the result. Note that we used $k_B = 1$, such that the pressure and temperature become arbitrary units.

```
% temperature
T(step) = E_kin(step) / (N * kB);

% pressure: wall
f_dot_n = sum(abs(f_i_wall),2);
P_wall = 1/(4*L) * mean(f_dot_n);

% pressure: virial
f_ij_dot_r_ij = sum(r_ij.*f_ij,3);
P_virial = N*kB*T(step)/L^2 + 1/(2*L^3) * mean(f_ij_dot_r_ij(:));

% total pressure
P(step) = P_wall + P_virial;

% total pressure by ideal gas law
P_law(step) = kB* N * T(step) / L^2;

% calculate averages
mvm_size = [100,0];
avg_T = movmean(T,mvm_size);
avg_P = movmean(P,mvm_size);
avg_P_law = movmean(P_law,mvm_size);
```

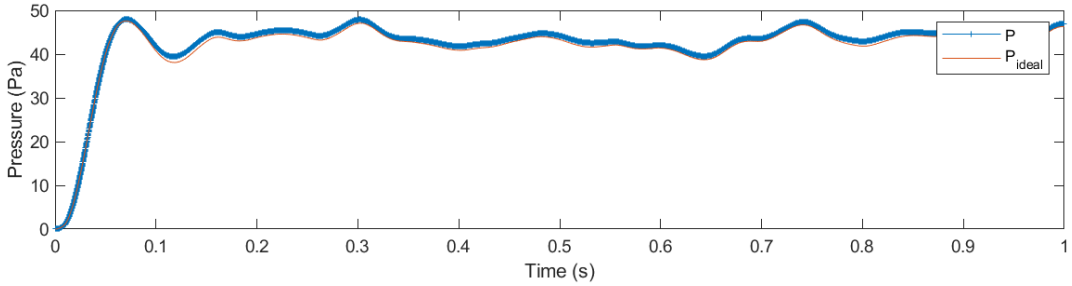


Figure 5: Pressure with step size $\Delta t = 10^4$ and 150 particles.

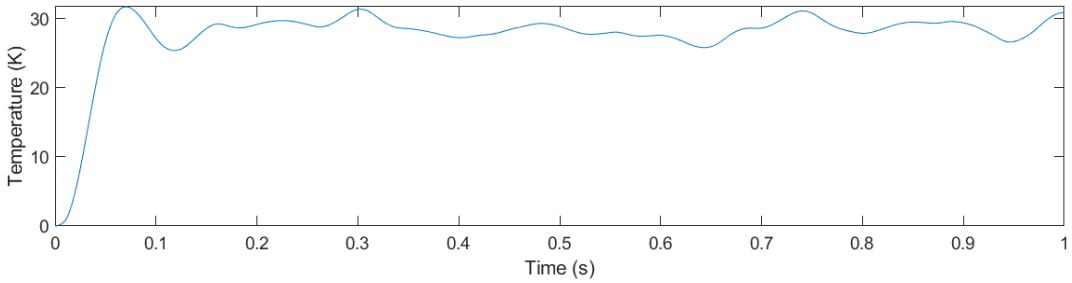


Figure 6: Temperature with step size $\Delta t = 10^4$ and 150 particles.

We loop over different densities $N = [10, 20, 50, 100, 200, 500]$, with $2E3$ steps and stepsize $\Delta t = 10^{-4}$. We calculate the total pressure containing the virial and wall pressures, and we calculate the pressure from the temperature by the ideal gas law. The result can be seen in Table 1.

N particles	Temperature [a.u.]	Pressure [a.u.]	Pressure (ideal gas law) [a.u.]	ΔPressure [a.u.]
10	3.28	0.4	0.3	0.1
20	3.32	0.8	0.7	0.1
50	16.62	8.6	8.3	0.3
100	21.14	21.5	21.1	0.4
200	27.65	56.6	55.3	1.3
500	55.10	277.5	275.5	2.0
1000	213.6	2138.0	2136.0	2.0

Table 1: Temperature, pressure and calculated pressure (from the ideal gas law), simulated with a certain number of particles.

What is visible in Table 1 is that the agreement between the ideal gas law pressure and the measured pressure is close, but not perfect. There could be a small error in our calculations, since our gas is (close to) an ideal gas: there is no energy absorption (only elastic collisions) and no other attractive forces.

Exercise (g)

We have plotted the Maxwell-Boltzmann distribution[1], $P(v)dv = \frac{m}{k_B T} v \exp\left(\frac{-mv^2}{2k_B T}\right) dv$, versus a histogram of the absolute velocities of the particles after integration time. In Figures 7 and 8 we see that both histograms agree with the distribution, with only 150 particles. Again, these results are obtained averaged over a moving window of size 100.

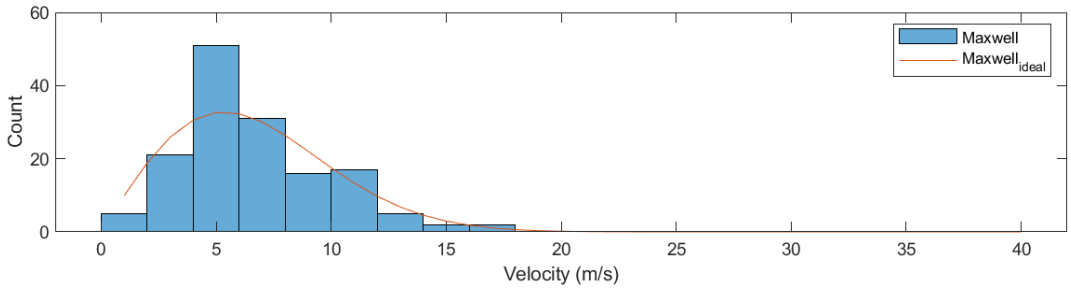


Figure 7: Velocity histogram versus Maxwell-Boltzmann distribution, with step size $\Delta t = 10^3$ and 150 particles.

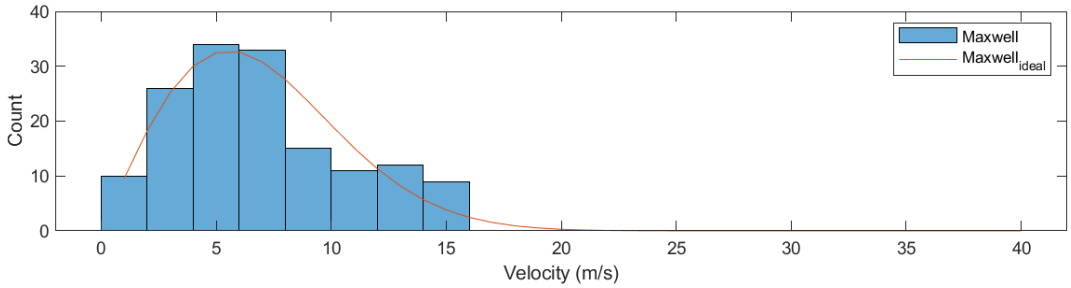


Figure 8: Velocity histogram versus Maxwell-Boltzmann distribution, with step size $\Delta t = 10^4$ and 150 particles.

[1] The distribution, in this case normalized, is scaled to the number of particles.

Exercise (h)

I incorporated both the radial distribution function and the self-diffusion coefficient.

Self-difussion coefficient

The self-difussion coefficient is calculated as

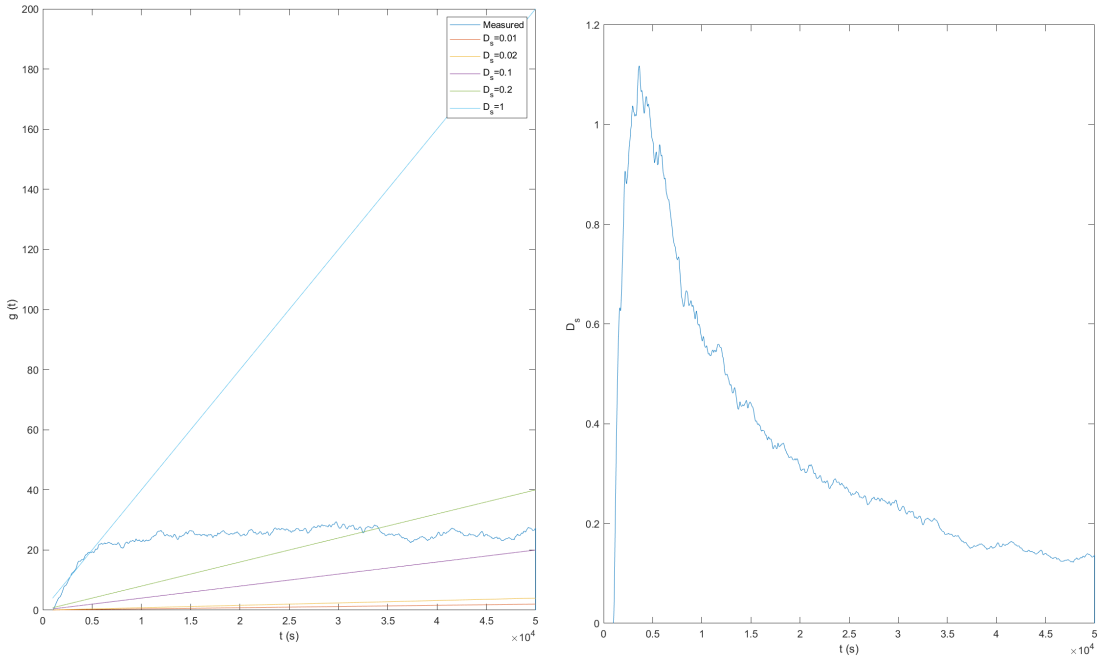
$$\begin{aligned} g(t) &= \langle (\mathbf{r}(\tau + t) - \mathbf{r}(\tau))^2 \rangle \\ &= 2dD_s t \qquad \text{for large } t. \end{aligned}$$

We therefore take $\Delta t = 10^{-3}$ and $N_{steps} = 5 \cdot 10^4$ to obtain a converged result.

```
function [g,Ds] = mean_square_displacement(x,N_steps,dt)
   tau = 1000;
   tt = tau:N_steps;
   g = zeros(size(tt));
   for idx = 1:length(tt)-1
      g(idx) = mean((x(:,1,tau+idx)-x(:,1,tau)).^2 + ...
         (x(:,2,tau+idx)-x(:,2,tau)).^2);
   end

   D = g./(2.*2.*tt.*dt);
   Ds = D(end);
end
```

However, it seems our boundary condition does not work with this approach, as our $g(t)$ converges to a number by that all particles are being bounded inside the box and thus cannot diffuse outside. From Figure 9 we can either approximate $D_s \approx 1.1$, or as $D_s = 0.13$



(a) $g(t)$ for measured data and calculated from different D_s values.

(b) Deduced D_s from measured data.

Figure 9

Radial distribution function

We have the definition of the radial distribution function as:

$$g(r) = \frac{\text{Actual number of particles in the volume between shells with radius } r \text{ and } r+dr \text{ around a particle}}{\text{Expectation value of the number of particles in this shell for a homogeneous fluid}}$$

For the expectation value, we have:

$$\begin{aligned}
\mathbb{E}N &= 2\pi \int_r^{r+dr} \frac{N}{L^2} r' dr' \\
&= 2\pi \left[\frac{1}{2} \frac{N}{L^2} r'^2 \right]_r^{r+dr} \\
&= \frac{\pi N}{L^2} \left((r+dr)^2 - r^2 \right) \\
&= \frac{\pi N}{L^2} \left(2rdr + dr^2 \right) \\
&\approx \frac{\pi N}{L^2} \left(2rdr \right) \qquad\qquad (dr << r) \\
&= \frac{2\pi N r dr}{L^2}.
\end{aligned}$$

We calculate the actual number of particles as:

```
sum(sum(r_ij_abs(r_ij_abs >= r & r_ij_abs < r+dr)))
```

This gives in total the following algorithm:

```
function g = radial_dist(r_ij_abs,dr,L,N)
   r_max = sqrt(2)*L;
   rr = 0:dr:r_max;
   g = zeros(size(rr));
   for idx = 1:length(rr)
      r = rr(idx);
      expected = 2*pi*N*r*dr/L^2;
      g(idx) = sum(sum(r_ij_abs(r_ij_abs >= r & r_ij_abs < r+dr)))/expected;
   end
end
```

Again, we are bound by the dimensions of our box, such that the location of the particle has an enormous influence on the eventual result. You can imagine that a particle close to two borders counts much fewer particles than a particle located close to the origin. We have not taken this into account in our simulations, but should be done once for example periodic boundary conditions are applied.

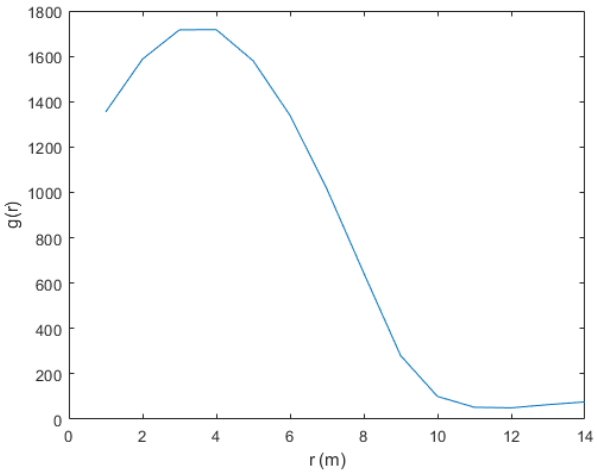


Figure 10: Radial distribution function of our system with $N = 1000$ and $L = 10$.