2D Gas Monte-Carlo Thermodynamic Phase Transitions in a Lennard-Jones Potential with Gravity

Nanometric System Simulation - Nanoscience and Nanotechnology UAB 2022/23

Xabier Oianguren Asua

1 The Model and its Assumptions

Let us consider a set of N particles in 2 dimensions, the k-th one having position coordinates $\vec{r}_k \equiv (x_k, y_k) \in B_x \subset \mathbb{R}^2$ and velocity $\vec{v}_k \equiv (v_{xk}, v_{yk}) \in B_v \subset \mathbb{R}^2$. The particles are confined into a rectangular box B_x of sides L_x, L_y, such that if a particle attempts to cross one of these walls, it will collide elastically with conserved magnitudes of the velocity in each axis, just that the sign of the component attempting the wall-crossing will be flipped (the walls absorb the change in momentum direction without movement). We will assume that the system is set in a thermal bath at temperature T, within a random collision model, by which the particles will have the possibility to suffer a kick by the thermal bath every τ time units. These events may give the k-th particle a random velocity $\vec{u}_k$, distributed[1] in the disk B_v of radious v_{max}, centered in $\vec{v} = (0, 0)$. The rule to know if it will take this velocity or rather it will stop its motion at the location of last collision is the following one. Assuming that the total energy of the system of particles is $E_0 := E(\vec{r}_1, ..., \vec{r}_N, \vec{v}_1, ..., \vec{v}_N)$ and the random kick would yield a system with energy $E_f := E(\vec{r}_1, ..., \vec{r}_k + \tau\vec{u}_k, ..., \vec{r}_N, \vec{v}_1, ..., \vec{u}_k, ..., \vec{v}_N)$, the system will adopt the new configuration with a probability

$$P(accept\ kick) = min\left(e^{-\frac{(E_f - E_0)}{k_B T}}, 1\right) \tag{1}$$

where k_B is the Boltzmann constant. This means that the step to the new configuration will happen for sure if the energy of the final state is smaller than the initial one, and will happen with a Gibbs-Boltzmann probability factor else, as done in Monte Carlo simulations.

After enough time, the system is expected to thermalize around a fixed average particle-energy $\langle E\rangle(T)$ that will depend on the temperature T. That is, the system will arrive to a dynamic equilibrium state, where the macroscopic variables like average energy or density will become stationary up to small fluctuations. The configurations attained by the system from the equilibrium time on will be the ones we expect the system to be found in nature.

The particles will be considered to interact with each other through a Lennard-Jones-like potential

$$U(\vec{r}_k, \vec{r}_j) = 4\varepsilon\left(\left(\frac{\sigma}{||\vec{r}_k - \vec{r}_j||}\right)^{12} - \left(\frac{\sigma}{||\vec{r}_k - \vec{r}_j||}\right)^{6}\right) \tag{2}$$

where $2^{1/6}\sigma \simeq 1.122\sigma$ is the inter-particle distance $r_{kj} := ||\vec{r}_k - \vec{r}_j||$ giving the potential energy minimum, which has an energy of $-\varepsilon$. We plot this potential energy profile in Figure 1.

The particles will be assumed to be inmersed also in an external potential, given by the gravitational potential energy for each particle

$$V(\vec{r}_k) = m_k g y_k \tag{3}$$

where m_k is the mass of the k-th particle (we assume they have the same mass $m = m_k\ \forall k \in \{1, ..., N\}$).

With this, the total energy of the system would be given by these potential energy contributions, plus the kinetic energy of the particles

$$E(\vec{r}_1, ..., \vec{r}_N, \vec{v}_1, ..., \vec{v}_N) = \sum_{k=1}^{N}\left(\frac{1}{2}m_k||\vec{v}_k||^2 + V(\vec{r}_k) + \sum_{1\leq j\leq k} U(\vec{r}_k, \vec{r}_j)\right) \tag{4}$$

[1] We decided to sample uniformly the angle and magnitude of the velocities, which makes the probability of an output kinetic energy homogeneous for each kick. Note that this however does not yield a uniform sampling in the disk.

Assuming the main energetic contribution in the condensed phase is due to the Lennard-Jones potential (as we will see when choosing the parameters), we may measure the energy in units of ε and the distance in units of σ. Since the ratio $\varepsilon/(k_B T)$ will measure when the thermal bath will be able to break the Lennard-Jones bonding, we may define an adimensional temperature $\tilde{T} := Tk_B/\varepsilon$, which when $\tilde{T} \sim 1$ will imply that the thermal characteristic bath energy will be $k_B T \sim \varepsilon$.

Finally, we can define a metric reflecting the density of the ensemble of particles, by taking the inverse of the average distance to the M closest neighbors of each particle. The bigger M is, the better it will reflect when the particles condense in bigger connected components. We use in particular $M = 10$.

A Small Disclaimer

In reality, a less biased way to run the Monte-Carlo approach might be to randomly change the velocities and positions of *all* the particles in the system *at once* and then check, given the energy of the new state of the system whether the step will really take place using equation (1). This might be more reasonable than "locally" changing the state of each individual particle, then checking if the step for the whole takes place and continuing iteratively, because, this last way does not give random configurations for the system in each step (only one-change-at-a-time configurations are tested). Thus, it is not clear that the algorithm will correctly explore the configuration space. Nonetheless, the global state change approach takes very long till thermalization, since by sampling the velocity of the all particles uniformly at once leads typically to unfavourable configurations. Most step attempts end up in false moves. We therefore hold to the postulate that the local exploration might still be enough to find the thermal equilibriums. Interestingly, because single-particle changes produce new states that are very close to the previous one, in a certain sense, such a "biased" walk generates a "smoother" Markov chain, that might be more realistic if we assume deterministic dynamics lay below thermodynamics.

2 Choosing the Parameters

In order to give physical significance to the model, we will choose the parameters that could give us a first approximation to the liquid to gas phase transition of water H_2O.

- Given the molar mass of water is $18.015 g/mol$ [1], a molecule of water has $m = 3 \cdot 10^{-26} kg$.
- Given the predominant intermolecular interaction for water are hydrogen bonds, which have in the case of liquid water an energy of $23.3 kJ/mol$ [2], equivalently $3.87 \cdot 10^{-20} J$ per molecule, we could use this number as the paramter ε of the Lennard-Jones potential. We do not hope to obtain the exact same transition temperatures as the expected ones, since with this ε, $\tilde{T} = 1$ if $T = 2500K$.
- The length of a hydrogen bond in water is of $0.197 nm$ [3], given the diameter of a water molecule is of $0.275 nm$ [4], we could estimate the distance parameter towards the Lennard-Jones potential minimum as $\sigma = 0.472 nm$.
- We will assume a virtual "mean free path" in each iteration towards thermalization of $\tau = 10^{-10} s$, based on the assumption that (in room temperature and atmospheric pressure) the mean speed of the vapor water molecules is of around 600 m/s and they have a mean free path on the order of $10^{-7} m$, both obtained using the kinetic theory of gases (as if only the particles in gas phase moved).
- Fixing a simulation box of $L_x = 45\sigma, L_y = 100\sigma$ and $N = 15^2 = 225$ molecules, we can clearly see the formation of different phases in a reasonable computational time.
- We assume that $M = 10$ neighbors in the equation for density are a good compromise that captures both local proximity and connected component size.
- Finally, we assume a $v_{max} = min(L_x, L_y)/\tau$, such that the particles are capable of crossing the entire scenario in a single step but no more than that.

3 The Implementation

We implement the method in Python 3, using a Jupyter Notebook as can be found in our repository [5]. The calculations heavily relay on the `numpy` library [6], while the plotting routines require the `matplotlib` library [7]. The notebook allows a user to change the paramters and to generate either animations of the "time evolution" towards thermalization or the plots that look for the phase transitions. This code is also provided as standard python scripts in case the user finds them more comfortable to use.

We summarize in Listing 1 the library call, parameter definition and state initialization routine, in Listing 2 the energy and density computation routines, in Listing 3 the plotting routines and in Listing 4 the core code that runs the model.

Listing 1: Part of the designed scripts, where the library call, parameter definition and state initialization routine are given.

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle

m=3e-26 #kg
epsLJ=1
sigmaLJ=1
epsLJ_SI = 23.3e3/(6.022e23) #J hydrogen bond between water molecules
sigmaLJ_SI = 0.197e-9+0.275e-9 #m
g=9.8 #m/s^2
Ls=[ 45,100 ] # units of sigma
t_mfp = 1e-10 # seconds
v_max = min(Ls)/t_mfp
kB=1.38e-23 #J/K
N_particles=15**2
N_iterations=1000
N_neighbors=4
height_density_bins = 10
Temp_adim=0.5

height_bins = np.linspace(-Ls[1]/2, Ls[1]/2, height_density_bins+1)
height_bin_c=np.linspace(-Ls[1]/2+Ls[1]/height_density_bins/2,
                         Ls[1]/2-Ls[1]/height_density_bins/2, height_density_bins)

def get_initialized_state( N_particles, L_x, L_y, v_max,
                        mode_position='a_sixth_of_room', mode_speeds='slow'):
    v_magnitudes =  np.random.uniform(0, 1, size=N_particles)  # [N_particles]
    v_angles = 2*np.pi*np.random.uniform(0,1, size=N_particles) #[N_particles]
    if mode_speeds=='slow':
        v_magnitudes *=(v_max/40)
    elif mode_speeds=='still':
        v_magnitudes *= 0
    else: # randomly
        v_magnitudes *=(v_max)

    if mode_position=='a_sixth_of_room':
        return np.vstack( (np.random.uniform(-Ls[0]/2,-Ls[0]/2+Ls[0]/6, N_particles),
                         np.random.uniform(Ls[1]/2-Ls[1]/6,Ls[1]/2, N_particles),
                         v_magnitudes*np.cos(v_angles),
                         v_magnitudes*np.sin(v_angles) ) ).T  #[N_particles, 4]
    elif mode_position=='lattice':
        n=int(np.sqrt(N_particles)) # aranged in nxn lattice
        if n**2!=N_particles:
            raise ValueError
        xy = np.meshgrid(np.arange(0,n), np.arange(0,n))
        return np.vstack( (2**(1/6)*xy[0].flatten(), 2**(1/6)*xy[1].flatten(),
                         v_magnitudes*np.cos(v_angles),
                         v_magnitudes*np.sin(v_angles) ) ).T  #[N_particles, 4]

    else: # mode_position=='randomly':
        return np.vstack( (np.random.uniform(-Ls[0]/2,Ls[0]/2, N_particles),
                         np.random.uniform(-Ls[1]/2,Ls[1]/2, N_particles),
                         v_magnitudes*np.cos(v_angles),
                         v_magnitudes*np.sin(v_angles) ) ).T  #[N_particles, 4]
```

Listing 2: Part of the designed scripts, where the energy and density computation routines are given.

```python
def E_intrinsic(state, m=m):
    return 0.5*m*np.sum(state[:,2]**2+state[:,3]**2)*(sigmaLJ_SI)**2 # units of joules

def pairwise_distances(xy_positions_listed): # expected to be [N_particles, 2] the input
    # first computed [N_particles, 1, 2]-[1, N_particles, 2]->[N_particles, N_particles, 2]
    # then apply norm along the last coordinate-> a matrix [N_particles, N_particles] of pairwise distances
    return np.linalg.norm( xy_positions_listed[:, np.newaxis, :] -
                          xy_positions_listed[np.newaxis, : , :], axis=-1 )

def E_pair_wise(state, epsLJ=epsLJ, sigmaLJ=sigmaLJ ): # Lennard-Jones
    distance_ij = pairwise_distances(state[:,:2]) # [N_particles, N_particles]
    r = distance_ij[ np.tril_indices(distance_ij.shape[0], k=-1) ] # only select the lower triangular part
    return np.sum(4*epsLJ*((sigmaLJ/r)**12-(sigmaLJ/r)**6))*epsLJ_SI # in J

def E_external(state, m=m, g=g): # gravity y the y direction
    return m*g*np.sum(state[:, 1])*sigmaLJ_SI # in J

def compute_E(state):
    return E_intrinsic(state)+E_pair_wise(state)+E_external(state) # in J

def get_average_density(xy_positions_listed, n_neighbors=N_neighbors):
    # average distance to the n_neighbors closest neighbors
    distance_ij = pairwise_distances(xy_positions_listed)
    return np.mean(np.sort(distance_ij, axis=1)[:,:n_neighbors])**-1

def get_average_and_height_density(xy_positions_listed, n_neighbors=N_neighbors, bins=height_density_bins):
    distance_ij = pairwise_distances(xy_positions_listed)
    closest_dists=np.mean(np.sort(distance_ij, axis=1)[:,:n_neighbors], axis=1) #[N_particles]
    height_density=[np.mean(closest_dists[
               np.where((xy_positions_listed[:,1]>height_bins[k])&(xy_positions_listed[:,1]<height_bins[k+1]))])
                   for k in range(height_density_bins)]
    return np.mean(closest_dists)**-1,np.array(height_density)**-1
```

Listing 3: Part of the designed scripts, where the plotting routines are given.

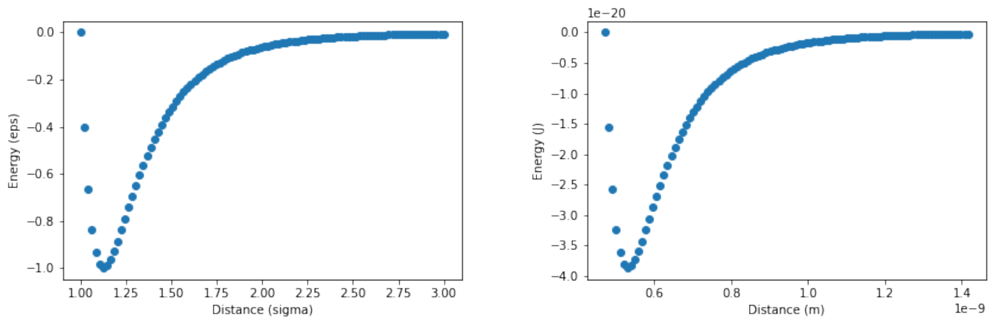
```python
def plot_state_and_metrics(state, old_state, average_Densities, average_Energies, height_Density ):
    fig = plt.figure(figsize=(18,9))
    ax = fig.add_subplot(121)
    ax.add_patch(Rectangle((-Ls[0]/2, -Ls[1]/2), Ls[0], Ls[1], fill=False))
    delta = state[:,:2]-old_state[:,:2]
    ax.quiver(old_state[:,0],old_state[:,1],delta[:,0],delta[:,1],
         scale_units='xy', angles='xy', scale=1, alpha=0.7, color="#348ABD")
    ax.scatter(state[:,0], state[:,1], s=5, c="r")
    ax.set_xlim(-Ls[0]/2-Ls[0]/10, Ls[0]/2+Ls[0]/10)
    ax.set_ylim(-Ls[1]/2-Ls[1]/10, Ls[1]/2+Ls[1]/10)
    ax.set_xlabel(f'sigma ({sigmaLJ_SI:.3} m)')
    ax.set_ylabel(f'sigma ({sigmaLJ_SI:.3}m)')
    ax.set_title(f"Time Iteration {it} at t_mfp={t_mfp}s\nAverage Energy {E_state/epsLJ_SI/state.shape[0]:.3} eps")
    ax = fig.add_subplot(233)
    ax.plot(average_Energies, color="#348ABD")
    ax.set_title("Average Energy vs Iterations")
    ax.set_ylabel(f"Averga Energy in eps ({epsLJ_SI:.3} J)")
    box = ax.get_position()
    ax.set_position([box.x0 - 0.03,box.y0,box.width*1.15, box.height])
    ax2 = plt.axes([0,0,1,1])
    # Manually set the position and relative size of the inset axes within ax1
    ip = InsetPosition(ax, [0.4,0.4,0.6,0.6])
    ax2.set_axes_locator(ip)
    ax2.plot(average_Energies[-200:])
    ax2.set_xlabel("Last 200 iterations"
    ax = fig.add_subplot(2,3,6)
    ax.plot(average_Densities, color="#A60628")
    ax.set_title("Average Density vs Iterations")
    ax.set_ylabel(f"Mean dist. to {N_neighbors} closest neighbs.^-1 (sigma)")
    ax.set_xlabel("Iteration")
    box = ax.get_position()
    ax.set_position([box.x0 - 0.03,box.y0,box.width*1.15, box.height])
    ax2 = plt.axes([0,0,1,1])
    ip = InsetPosition(ax, [0.4,0.0,0.6,0.6])
    ax2.set_axes_locator(ip)
    ax2.plot(average_Densities[-200:], color="#A60628")
    ax2.xaxis.tick_top()
    ax2.set_xlabel("Last 200 its")
    ax2.xaxis.set_label_position('top')
    ax = fig.add_subplot(1,6,4)
    ax.plot(height_Density, height_bin_c, 'o-', color="#A60628")
    ax.set_title("Height-Density")
    ax.set_xlabel("Density")
    ax.set_ylim(-Ls[1]/2-Ls[1]/10, Ls[1]/2+Ls[1]/10)
    ax.set_yticks(height_bins)
    ax.grid(axis='y')
    box = ax.get_position()
    box.x0 = box.x0 - 0.05
    box.x1 = box.x1 - 0.05
    ax.set_position(box)
    ax.yaxis.set_ticklabels([])
    fig.suptitle(f"Adimensionalized Temperature = {Temp_adim}; g={g:.2}m/s^2; N_particles={N_particles}")
    plt.show()
```

Listing 4: Part of the designed scripts, where the core of the model explained in Section 1, is given.

```python
state = get_initialized_state( N_particles, *Ls, v_max,
                        mode_position='random', mode_speeds='random')
state_new = state.copy() #[N_particles, 4]
E_state=compute_E(state)
average_Energies=[]
average_Densities=[]

for it in range(N_iterations):
    # in each time iteration all particles will have the chance to be displaced
    # sample random point non-uniformly in the circle of radious v_max, but uniformly in params
    new_v_magnitudes =  np.random.uniform(0, 1, size=state.shape[0])*v_max # [N_particles]
    new_v_angles = 2*np.pi*np.random.uniform(0,1, size=state.shape[0]) #[N_particles]
    new_vx = new_v_magnitudes*np.cos(new_v_angles)
    new_vy = new_v_magnitudes*np.sin(new_v_angles)

    new_x = state[:, 0]+t_mfp*new_vx
    new_y = state[:, 1]+t_mfp*new_vy
    # bouncing/reflecting boundaries for the particles so whenever a particle
    # escapes one of the boundaries make it go back by the ammount it got out
    # dont need to iterate since v_max is such that tau*v_max=L but else do a while loop
    new_x[ np.where(new_x>Ls[0]/2) ] = Ls[0]-new_x[np.where(new_x>Ls[0]/2) ]
    new_y[ np.where(new_y>Ls[1]/2) ] = Ls[1]-new_y[np.where(new_y>Ls[1]/2) ]
    new_x[ np.where(new_x<-Ls[0]/2) ] = -Ls[0]-new_x[np.where(new_x<-Ls[0]/2) ]
    new_y[ np.where(new_y<-Ls[1]/2) ] = -Ls[1]-new_y[np.where(new_y<-Ls[1]/2) ]

    E_old=E_state
    old_state=state.copy()
    not_changed = np.ones(state.shape[0], dtype=bool)
    for k in range(state.shape[0]):
        state_new[k,:]=np.array([new_x[k], new_y[k], new_vx[k], new_vy[k]])
        E_state_new = compute_E(state_new) # J
        delta_E = E_state_new-E_state
        P=np.exp(-(delta_E)/Temp_adim/epsLJ_SI)
        if (delta_E < 0) or (np.random.rand() < P):
            not_changed[k]=False
            state[k,:] = state_new[k,:]
            E_state = E_state_new
        else:
            state_new[k,:] = state[k,:]
    state[not_changed, 2:] = 0
    average_Energies.append(E_state/N_particles/epsLJ_SI)
    av, height_Density = get_average_and_height_density(state[:,:2])
    average_Densities.append(av)

    if it%plot_every==0:
        plot_state_and_metrics(state, old_state, average_Densities,
                 average_Energies, height_Density )
```

4 Results and Discussion

We now present some of the results of interest found by playing around with the implemented simulator. The temperature of the thermal bath (and eventual temperature of the system when thermalized) and the strength of the gravitational potential relative to the Lennard-Jones potential and kinetic energy (reflected by g), were the main modified numbers. In all cases, we used fully random initial configurations for the system.



(a) **(b)**

Figure 1: Employed inter-particle interaction potential of Equation (2), where "distance" is the inter-particle distance, and eps and sigma mean ε and σ respectively. In (a) we can see the plot for a general case, while in (b) we can see the profile in International System units employing the constants and assumptions of Section 2.

4.1 The Thermalization

The first thing to study was the thermalization of the system, that is, whether the system adopted any defined dynamical equilibrium, yielding visually similar configurations an the same macroscopic variables of energy and density up to random fluctuations.

Negligible gravity

By setting $g = 9.8m/s^2$, we have that the effect of gravity on the energy of the system for the considered size of the box is negligible, since the difference in gravitational energy for a particle in each end of the box in the y axis would be of $1.4e - 32$J. In comparison, the characteristic Lennard-Jones energy is 12 orders of magnitude bigger $\varepsilon = 3.87e - 20$J. Thus with $g = 9.8m/s^2$, we can study the effect of the inter-particle interaction independently of gravity.

In particular as can be found in Figure 2, for small temperatures like $\tilde{T} = 0.01$, starting from a fully random state, the system tends towards an equilibrium where particles get bonded with each other and form a lattice, with an average inter-particle distance of about 2σ. The red dots in the left plot represent the particles (in the last iteration), while we can see the density of the particles as a function of height in the middle and the evolution for increasing iterations of the average energy and density in the right. Note how the system effectively achieves a plateau in both the energy and the density[2]. For intermediate temperatures like $\tilde{T} = 0.3$ or $\tilde{T} = 0.342$ (Figures 3 and 4), the system still shows clusters of condensed particles, but an increasing number of particles have the chance to wander around. Note that the blue arrows indicate the particles that were successfully kicked by the thermal bath in the last τ iteration. Finally, a high enough temperature like $\tilde{T} = 1$ (Figure 5) yields an equilibrium distribution where particles no longer form clusters, and where the minimum density of the system, 0.2 $sigma^-1$, is approached[3].

[2]More iterations might have given an even more reliable plateau, as can be seen from the fact that the insets showing the last 200 iterations are not oscillating around a constant.

[3]We know the finite volume of the box and the number of particles, so we can estimate the minimum average inter-particle distance and its inverse as the measure of density we are employing.

Therefore, we acknowledge at least one clear phase transition from fully ordered to fully free particles (as what a solid to gas or liquid to gas transition could look like).

Powerful gravity

By setting $g = 9.8e12m/s^2$, we can make gravity be as important for our simulation box as the clustering energy in the relevant temperatures for the phase transition we observed above. Then, setting $g = 9.8e13m/s^2$ will make gravity slightly more important than the clustering, and we might thus see a phase transition there as a function of gravitational strength. In particular, we have that the difference of gravitational energy for a particle in each end of the box would be of $1.4e-20$J for the first case, which is exactly on the same order of magnitude as $\varepsilon = 3.87e-20$ J, while it would be of $1.4e-19$J for the second case.

One can make sense of such big gravities on Earth by thinking on a ultra-centrifugated tube containing our particles. Thus, the question we adress here is not as fantastic as it may sound at first.

Figures 6 to 9 show thermalized configurations for $g = 9.8e12m/s^2$, while Figures 10 to 13 show them for $g = 9.8e13m/s^2$. In both cases, we can see that the condensed phases do show a height dependent density (being this bigger at the bottom of the box, where bigger clusters are formed). However, in the first case, gravity is still not big enough as to impede a free particle ensemble at temperatures $\tilde{T} \sim 1$. See this in Figure 9, where there is no clear trend in the density as a function of height. Interestingly, in the second case, where gravity is even bigger, at $\tilde{T} \sim 1$ (Figure 13), we find an ensemble of particles that appear to be free (no cluster is formed), but are constrained to have a linear density profile with height (which is what one expects to happen for the air in the atmosphere for example).

Both cases then suggest that there is a second sort of phase transition, where now it is not about forming clusters and being free, but being bent by gravity or having enough jitter to avoid it.

It would thus be interesting to see the evolution of the macroscopic variables as a function of both $\tilde{T}$ and g, to see if clear phase transitions can be abstracted.

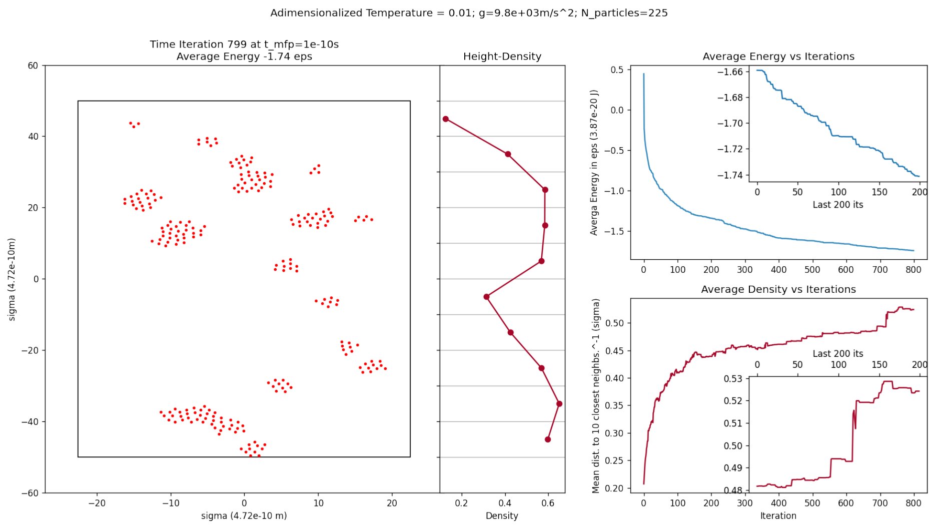


Figure 2: Thermalized system for $\tilde{T} = 0.01$ $g = 9.8m/s^2$, after 800 iterations, following the text in section 4.1.

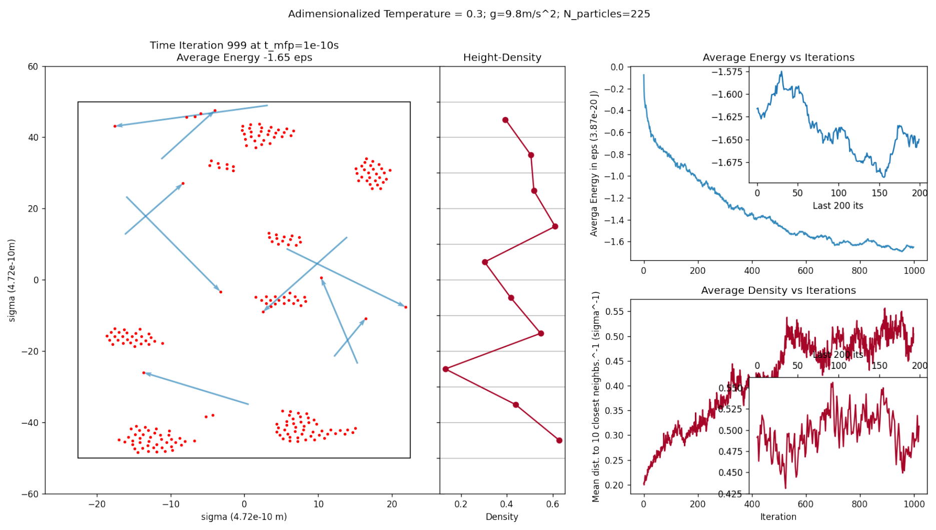


Figure 3: Thermalized system for $\tilde{T} = 0.3$ $g = 9.8m/s^2$, after 1000 iterations, following the text in section 4.1.

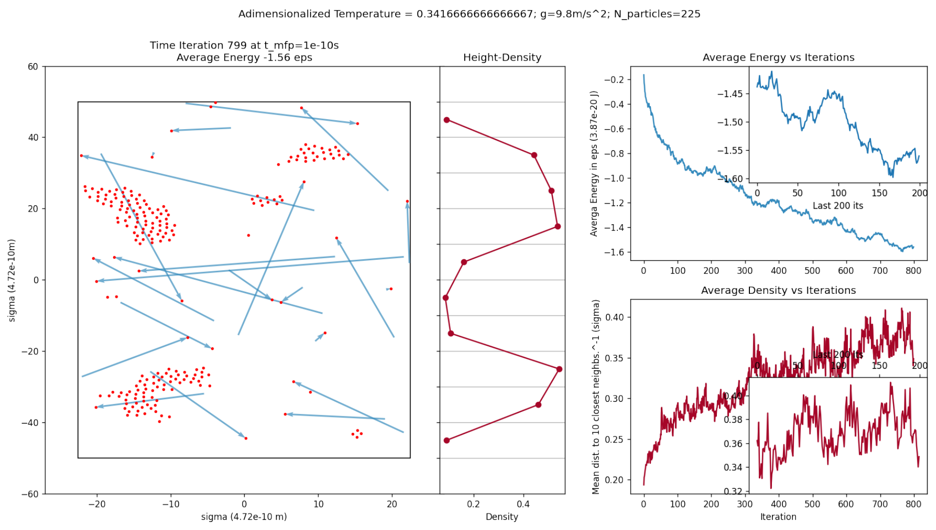


Figure 4: Thermalized system for $\tilde{T} = 0.342$ $g = 9.8m/s^2$, after 800 iterations, following the text in section 4.1.

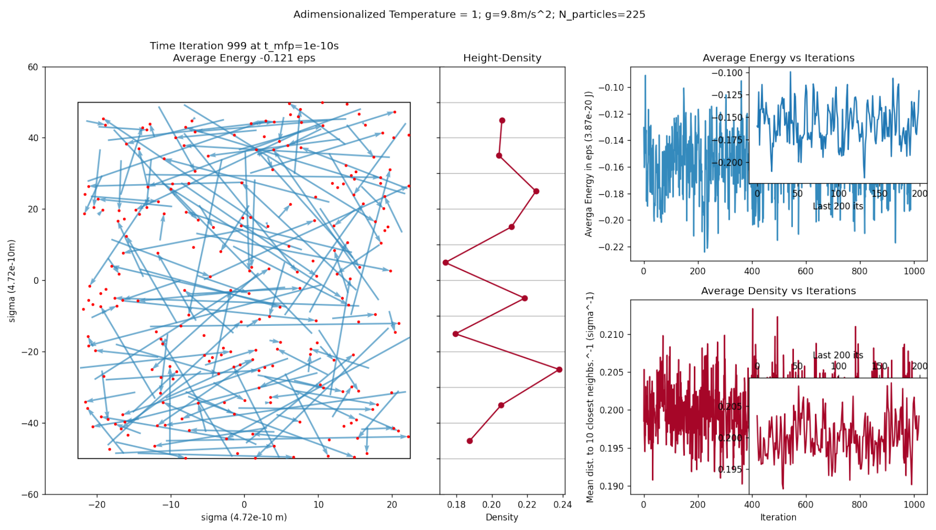


Figure 5: Thermalized system for $\tilde{T} = 1$ $g = 9.8m/s^2$, after 1000 iterations, following the text in section 4.1.

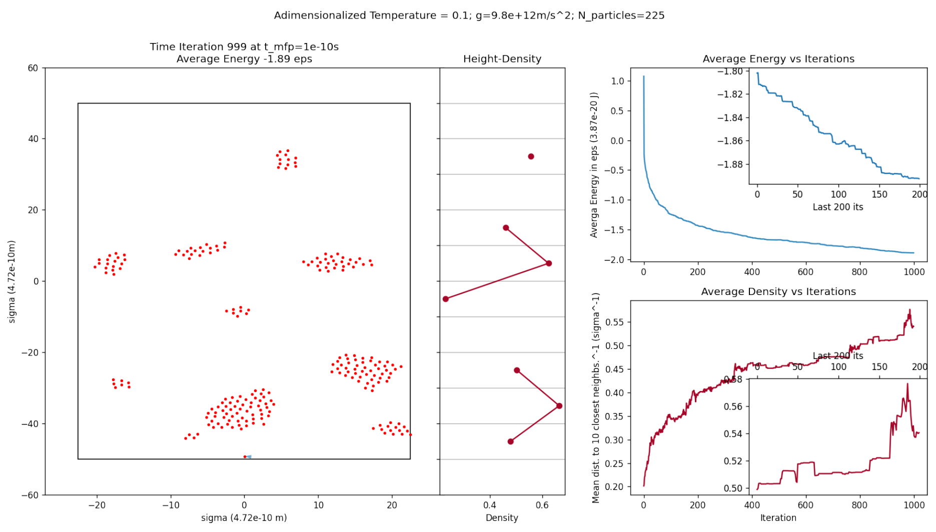


Figure 6: Thermalized system for $\tilde{T} = 0.1$ $g = 9.8e12m/s^2$, after 800 iterations, following the text in section 4.1.

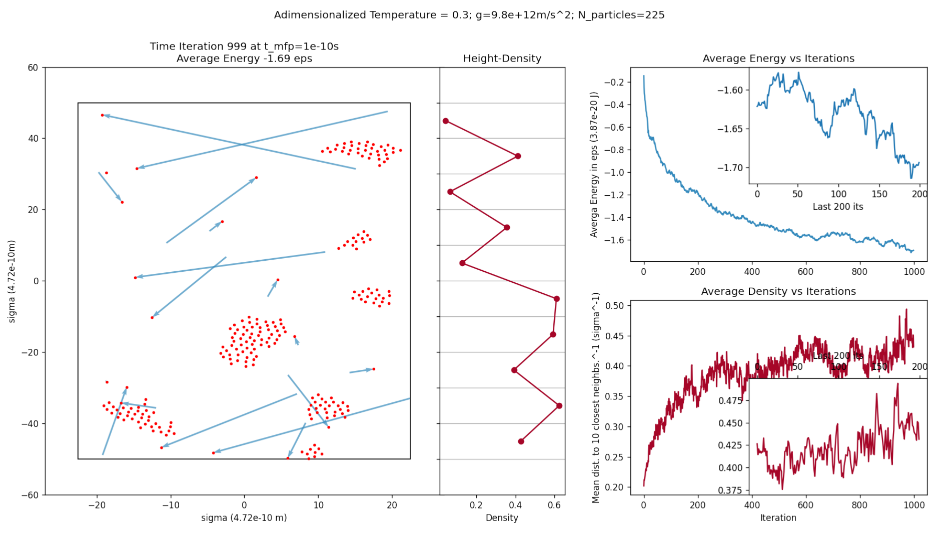


Figure 7: Thermalized system for $\tilde{T} = 0.3$ $g = 9.8e12m/s^2$, after 1000 iterations, following the text in section 4.1.

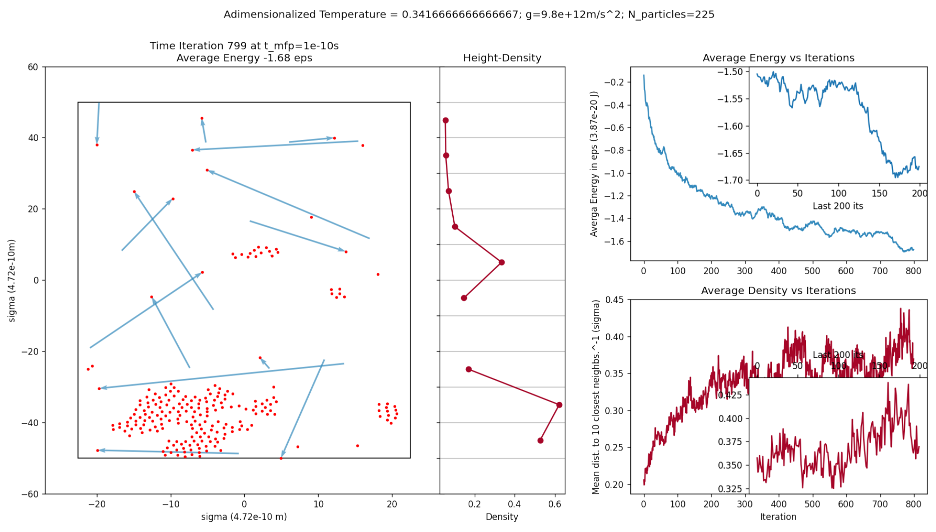


Figure 8: Thermalized system for $\tilde{T} = 0.342$ $g = 9.8e12m/s^2$, after 800 iterations, following the text in section 4.1.

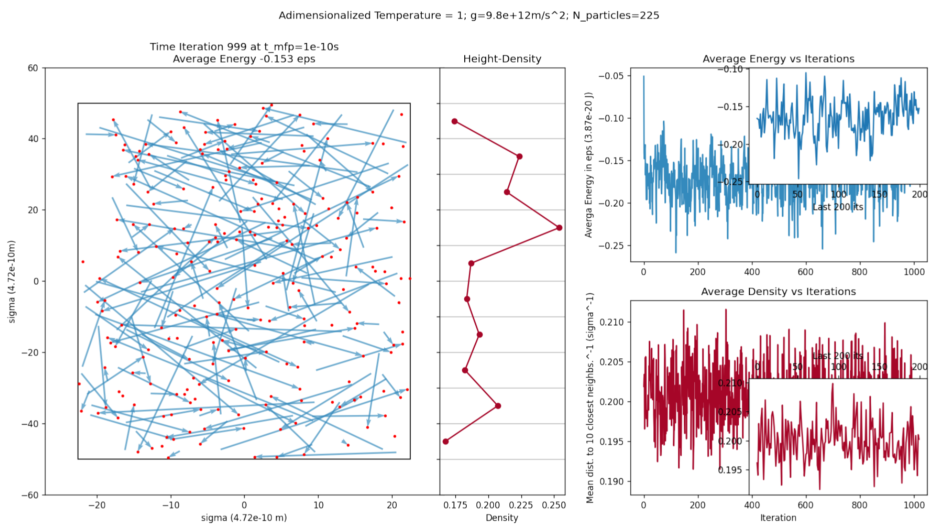


Figure 9: Thermalized system for $\tilde{T} = 1$ $g = 9.8e12m/s^2$, after 1000 iterations, following the text in section 4.1.

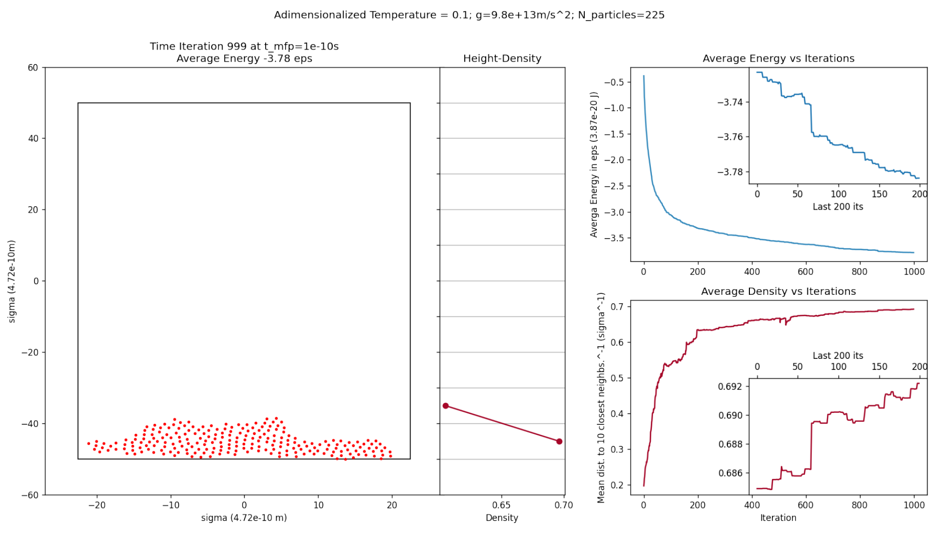


Figure 10: Thermalized system for $\tilde{T} = 0.1$ $g = 9.8e13m/s^2$, after 800 iterations, following the text in section 4.1.

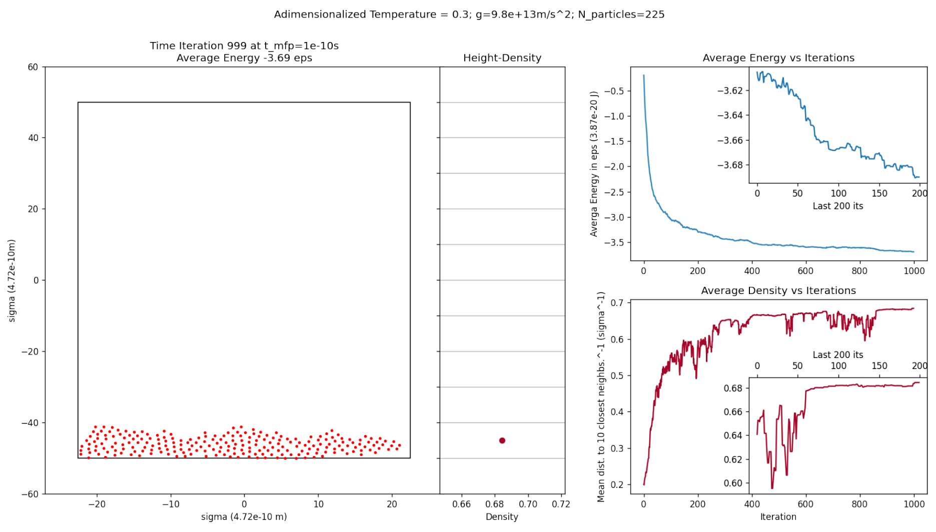


Figure 11: Thermalized system for $\tilde{T} = 0.3$ $g = 9.8e13m/s^2$, after 1000 iterations, following the text in section 4.1.

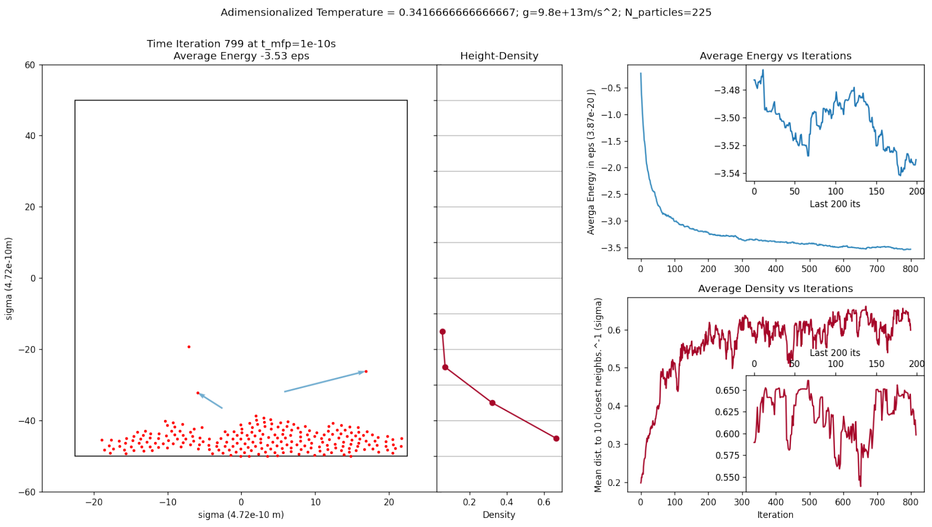


Figure 12: Thermalized system for $\tilde{T} = 0.342$ $g = 9.8e13m/s^2$, after 800 iterations, following the text in section 4.1.

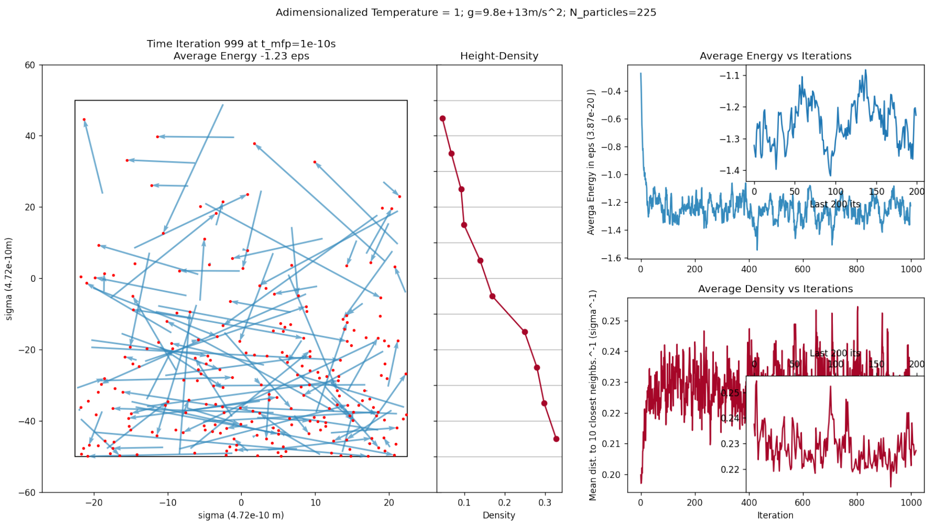


Figure 13: Thermalized system for $\tilde{T} = 1$ $g = 9.8e13m/s^2$, after 1000 iterations, following the text in section 4.1.

4.2 The Phase Transitions

In order to check if we are able to find the different phase transition temperature/gravity boundaries, we sweep both the temperature $\tilde{T}$ and gravity g in a grid of values (9 different points in g and 49 in $\tilde{T}$ for each of them) and compute the equilibrium density and equilibrium energy of each case. We plot the mean and standard deviation (fluctuations) of the average energy and densities over 200 iterations after 600 iterations till thermalization in Figures 14, 15 and 16. The results are very telling.

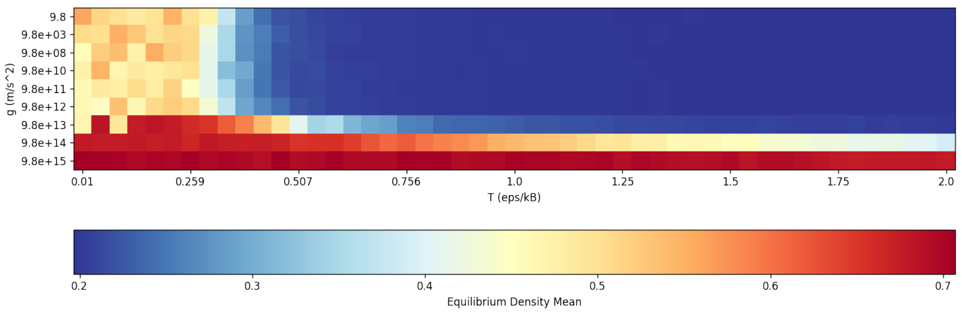
For the range of gravities smaller than $g \sim 9.8e12m/s^2$, we roughly find the same average densities before and after evaporation (Figure 14.a), of around 0.5 and 0.2 σ^{-1} respectively. This transition in all cases happens at a temperature of $\tilde{T} \sim 0.3$ or equivalently $T \sim 750K$ where the density abruptly changes regime. This is a strikingly good result, since empirically the phase transition of water happens at $T \sim 373K$, exactly half the obtained one! Such a finding suggests that the difference might be due to the fact that the phase transition for water is typically measured in 3 dimensions, while our model considers 2 dimensional particles.

It is interesting to note that this phase transition boundary matches a region were density fluctuations are maximal for those g values (Figure 14.b). Meaning that we can equivalently look for phase transition boundaries by studying the fluctuations of the macroscopic variables.

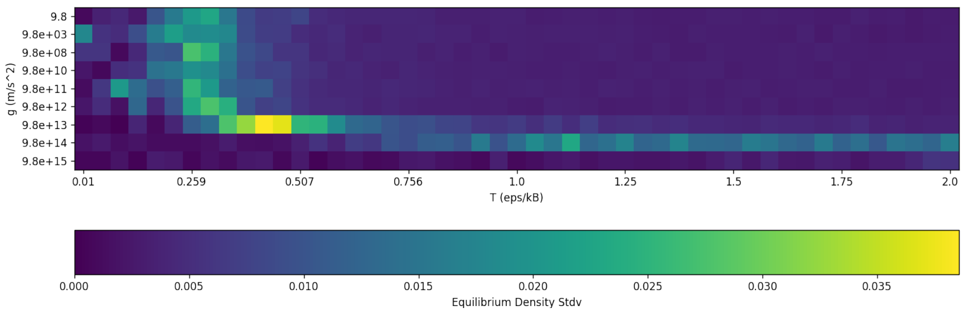
Regarding the energy (Figures 15.a and 16.a), we find the boundary happens slightly afterwards in $\tilde{T}$, but the change is already significant at $\tilde{T} \sim 0.3$, both regarding the mean and the standard deviation. This might be a consequence of a phenomenon akin to the vaporization entalphy.

For $g = 9.8e13 m/s^2$, we find in Figures 14.a and b, that the phase transition now occurs less sharply, along a range of temperatures with center in $\tilde{T} \simeq 0.507$. Densities as low as those obtained at $\tilde{T} > 0.3$ in lower g (around 0.2 σ^{-1}), are only achieved for $\tilde{T} > 1$. The average energy as well can be seen to need way more temperature to achieve energies as those given by the gas phase of the smaller g (Figure 15.b). This slower transition may be a symptom of the fact that even if particles can be free from bonding, they can still be forced to be close to each other by gravity and the finiteness of the box. What is more, at this g (and greater g), we find that the achieved maximum density (for the low temperature phase) is smaller than the minima for smaller g, of around 0.7 σ^{-1}. This shows the existance of a third phase (possibly linked to the one in which all matter was condensed in the bottom of the box Figure 10), that changing g makes evident for small $\tilde{T}$.

The phenomenon of condensed to free transition is worsen for $g = 9.8e14 m/s^2$, where the transition is stretched even wider with a center at a higher temperature, such that not even at $\tilde{T} = 2$ densities as slow as 0.2 are achieved. Finally, for $g > 9.8e14$, we find that in the studied range of tempreatures there is no change in average density nor energy of the system, and only the maximum density phase (the third phase we found) is possible.



(a)



(b)

Figure 14: Heat-maps showing the (a) mean and (b) the standard deviation of the average **density** in the simulation box, over 200 iterations after the 600-th iteration (after the system thermalized). The y axis shows the strength of gravity, while the x axis shows the adimensional temperature $\tilde{T}$.

All in all, by following the boundary outlined by high density fluctuations in Figure 14.b, and the three predominant colors in Figure 14.a, we find there are three stable phases: a ultra-high density one (the dark red one in Figure 14.a, represented by Figure 10), a relatively high density one -reflecting multiple connected components- (the yellow one, represented by Figure 6) and a low density/free particle one (the dark blue one, represented by Figure 5). Note that we do not see the boundary between the yellow and red domain in the fluctuation map, because it happens at some point in $g \in (9.8e12, 9.8e13) m/s^2$.

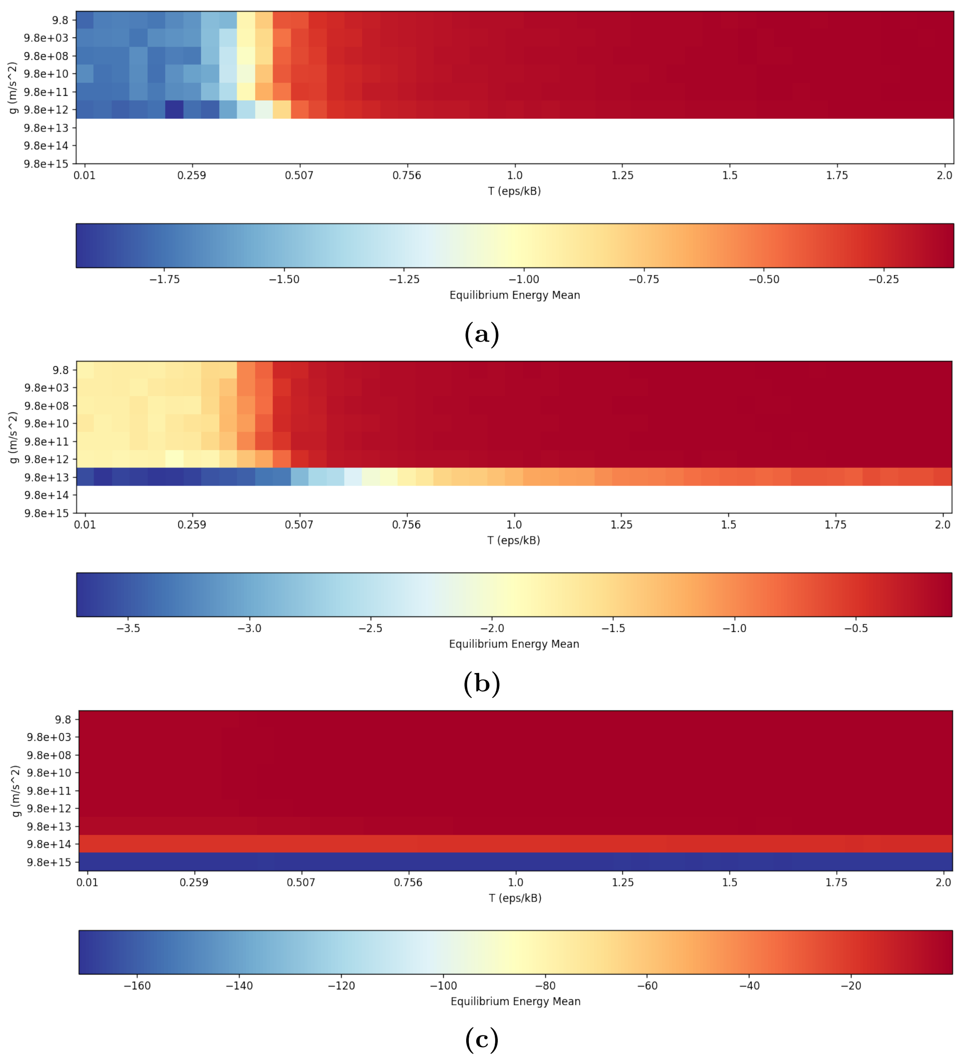


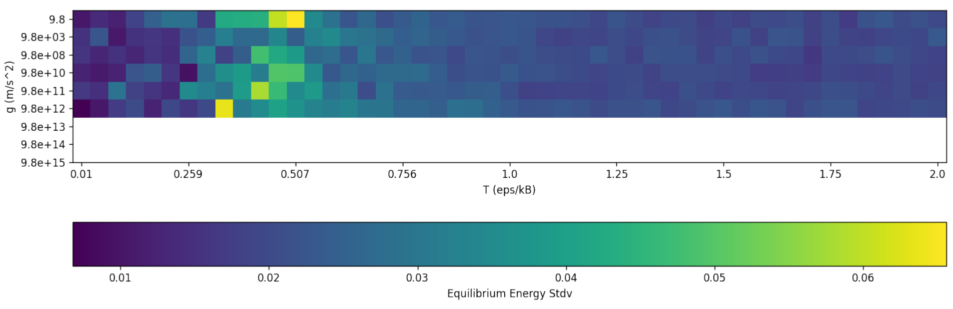
Figure 15: Heat-maps showing the mean of the average **energy** in the simulation box, over 200 iterations after the 600-th iteration (after the system thermalized). The y axis shows the strength of gravity, while the x axis shows the adimensional temperature $\tilde{T}$. In (a),(b),(c) the same data is plotted but progressively adding data of bigger gravity, which re-scales the range and thus makes hard to see the details.

5 Future Work

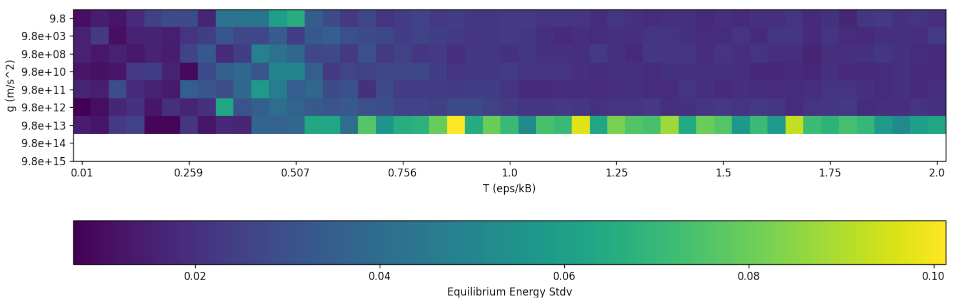
It is left as future work to analyze more deeply the effect of the mass of the particles m on the equilibrium phases, since this would modulate the importance of the kinetic energy relative to gravity and the Lennard-Jones wells.

It would also be of great interest to check the equilibrium distributions of velocities for example, to check that the Maxwell-Boltzmann distribution predicted by statistical mechanics is attained (even if we sample the velocities uniformly). That is, to check that the post-selection of states by the Boltzman distribution, effectively leads to such a velocity distribution. For this, one could wait until the system thermalizes and then record the relative frequencies with which velocities given by the thermal kicks are accepted. In theory, a Gaussian distribution should appear.

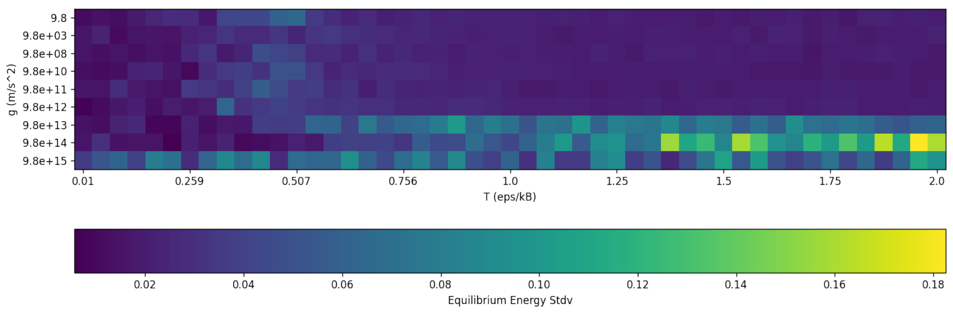
Finally, a study of the entropy would also be of great interest. For this, the relative frequency of the states that lead to the same macrostates (density and energy parameters) would be required for example, or rather the relative probabilities of occupation of each configuration, once the system is thermalized. This however, would require a big amount of calculation, since a big part of configuration space should be explored by the thermalized system. In any case, it would be interesting to check that the equilibrium configuration gives the distribution of configurations that maximizes the entropy for a given average energy, as required in theory by the Boltzmann distribution.



(a)



(b)



(c)

Figure 16: Heat-maps showing the standard deviation of the average **energy** in the simulation box, over 200 iterations after the 600-th iteration (after the system thermalized). The y axis shows the strength of gravity, while the x axis shows the adimensional temperature $\tilde{T}$. In (a),(b),(c) the same data is plotted but progressively adding data of bigger gravity, which re-scales the range and thus makes hard to see the details.

References

[1] "National center for biotechnology information. pubchem compound summary for cid 962, water.." https://pubchem.ncbi.nlm.nih.gov/compound/Water.

[2] S. Suresh and V. Naik, "Hydrogen bond thermodynamic properties of water from dielectric constant data," *The Journal of Chemical Physics*, vol. 113, no. 21, pp. 9727–9732, 2000.

[3] "Wikipedia entry on the hydrogen bond." https://en.wikipedia.org/wiki/Hydrogen_bond.

[4] Y. Zhang and Z. Xu, "Atomic radii of noble gas elements in condensed phases," *American Mineralogist*, vol. 80, no. 7-8, pp. 670–675, 1995.

[5] "Github repository with the python scripts and notebook generated for the report." https://github.com/Oiangu9/_Miscellaneous/tree/main/SSN.

[6] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, Sept. 2020.

[7] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.