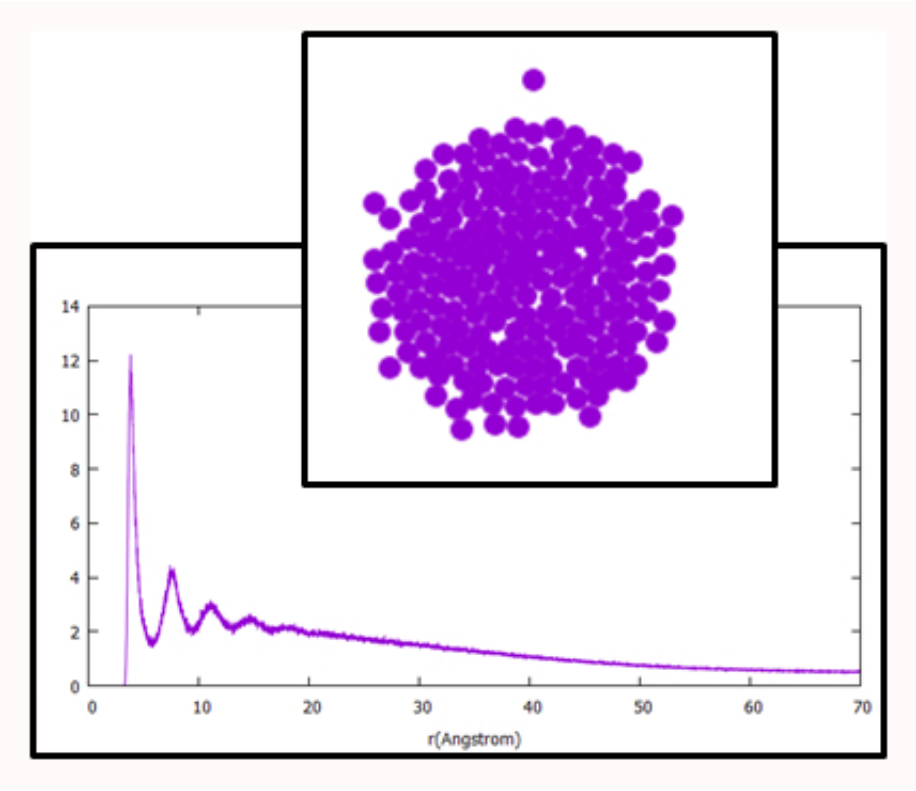
Manual

"RANDOM PHASE"

Atomistic simulations of phase transitions by MMC methods in 2D. v.1.0

O.T, O.G, M.P, and J.E.A

30th June 2019



Aim of this program

In a phase transition, matter commonly undergoes a change in the aggregation state, for example from gas to liquid or solid. During this transformation, some properties of the medium change abruptly (for example the density in the mentioned case) as a result of the change of external factors, typically temperature or pressure. Thermodynamics is the branch of the Physics which deals with this phenomenon, describing the behavior of macroscopic physical properties during these transformations. Statistical Physics solves systems with a very large number of particles allowing to link the microscopic properties of the system to the macroscopic state associated (studied by Thermodynamic). However, both when using Thermodynamics or Statistical Physics, students have to deal with these many-particle phenomena and properties, which usually are hard (or nor really intuitive) to understand. An atomistic computer simulation can help students to grasp these concepts, visualizing how interaction potential among particles, temperature and particle concentration can define our system (and its phase) and how it evolves during the transition. However, this approach usually requires the handling of complicated simulation programs.

We present a new tool [1] by which the change of state of a system can be experienced in an interactive way with no need of previous background. This computational experiment should be suitable for secondary school students, when only the basic features of the simulation are used. In addition, when all the possibilities of the tool are taken, it could even be appropriate for undergraduate students or physics teachers who want to acquire a deeper insight.

In order to simplify the visualization of the results, we have restricted our system to two dimensions (2D), reducing the computational cost of the simulation to be done in a class or laboratory session. The computational experiment is focused on the understanding of the behavior of different states of matter and how phase transitions appear, providing students a first insight into the physical phenomenon and the simulations process.

The so called Monte Carlo computer experiments are based on the evaluation of mean properties related with positions (and not with velocities, as in molecular dynamics approximations (see [2])) in equilibrium systems, obtained at random configurations. In this software, we have implemented the particular Metropolis Monte Carlo method (MMC)[3].

Monte Carlo methods are useful for systems with a probabilistic interpretation, in which the number of samples is large enough. As the ergodic theorem [4] states, the expected value of a random variable can be approximated by taking the mean value of independent random samples with a given stationary probability distribution (i. e., within a statistical ensemble). MMC method simulates particle systems, with a small computational cost, and evaluates mean values of equilibrium properties, as for example: energy, temperature or structural parameters. In our case, the sampling is performed in the Canonical Ensemble, in which particle number, volume and temperature are constants, called the "NVT ensemble". However, as we are constrained to two dimensions, the ensemble will be forced to have a constant area, instead of volume. In NVT ensemble, the system is immersed in an ideal heat bath, keeping the temperature constant and allowing the system to reach states with different energies.

For a first example, we simulate a 2D system composed of Argon atoms that interactuate via a Lennard-Jones (L-J) potential. This system could be described as a pool table (the surface) on which a given number (the concentration) of soft billiard balls (the atoms) are trying to randomly move (MC method) to achieve the required degree of interaction among them (given by the temperature). For selected temperatures and concentrations, the results will allow observing changes in the system structure, as well as other equilibrium properties. Physical properties will be obtained after the so called *thermalization process,* when the system reaches equilibrium.

1 Brief introduction

The *Random Phase* program has been developed to simulate bi-dimensional systems using the Metropolis Monte Carlo (MMC) method. It can be used as an educational tool, or even with basic research purposes. It has been designed to obtain equilibrium properties which characterize the phases and the structures of the simulated systems. The calculated properties are listed below:

- Radial Distribution Function (RDF).
- Bi-dimensional Distribution Function.
- Local Density Distribution.
- Mean Local Density.
- Average Number of Neighbors.
- Spatial Correlation of the orientation of the system.
- Evolution of orientation of the system.
- Evolution of the Specific Heat.
- Evolution of the average Energy of the system.

The evolution of the system during the simulation (as an animation) can be also represented in order to visualize the phases and their transitions. It must be pointed out that these evolutions are not time dependent, as the MMC method only evaluates the probability to obtain a given configuration, not the particles trajectories. If we are interested in the time evolution or in dynamic properties, a Molecular Dynamic (MD) method should be employed.

The system can be defined by the "**system variables**" (mainly NVT and inter-particle potential), which can be fixed by the user. However, the procedure of the simulations depends also on the so-called "**simulation parameters**" (which allow to adjust the MMC method to our particular case) that also can be changed by the user. For a given system,

simulation parameters should be adequately selected to obtain a fair sampling (some input files are included to help with this task). All the values (system variables and simulation parameters) are given in the `input_file`, (explained in detail in the Section 5 "The Input File"), and a brief description is given below:

System Variables

- `Temperature_sys` (real number > 0): Temperature of the system in K.
- `Concentration_sys` (real number > 0): Concentration of particles of the system (in particles/Å^2).
- `Particles_Num` (integer > 0): Number of particles included in the simulation.
- `PotentialEnergy` (integer 0,1 or 2): Inter-particle potential function.
- `Epsilon_lj` (real number > 0): Depth of the potential-well, for Lennard-Jones or Morse potentials (in eV).
- `Sigma_lj` (real number > 0): Atomic radius in Å (see definition for different potentials).
- `WellWidth` (real number > 0): Width of the potential-well for Morse potential.
- `Condensed_ini` (integer 0 or 1): This variable states if the initial configuration is in a condensed (0) or a diluted (1) phase.

Simulation Parameters

- `Steps` (integer > 0): Number of attempts of particle movements to be done in the simulation (in millions of steps).
- `CutofRadius` (real number > 0): Maximum Distance to compute the inter-particle potential function (in Å).
- `VerletRadius` (real number > 0): Radius used to construct the Verlet list of neighbors (in Å).
- `Nsampling` (integer > 0): Number of samplings used to evaluate general system properties.
- `Processors` ($0 <$ integer < 30): Number of CPU-processors used in the simulation.
- `Rneighbor` (real number > 0): Maximum distance for which particles are considered closest-neighbors.

- `Nneighbor` (integer > 0): It determines the minimum number of neighbors surrounding a given particle to be considered as part of a condensed phase.
- `Rdensity` (real number > 0): Radius used to calculate the local density in which each particle is allocated (in Å).
- `SpatialGrid` (real number > 0): Maximum displacement of a particle in each attempt of movement at temperature T=1 K (in Å). Note that the maximum displacement increases along the temperature.
- `MaxSpatialGrid` (real number > 0): Maximum displacement of a movement at any temperature (in Å).
- `TempLoop` (integer 0 or 1): Option to automatically simulate over a given range of temperatures (1) or not (0).
- `TempLoopInit` (real number > 0): Initial temperature of the loop.
- `TempLoopFin` (real number > 0): Final temperature of the loop.
- `TempLoopStep` (real number): Step size in the temperature loop.
- `ConcLoop` (integer 0 or 1): Option to automatically simulate over a given range of particles concentrations (1) or not (0).
- `ConcLoopInit` (real number > 0): Initial concentration of the loop.
- `ConcLoopFin` (real number > 0): Final concentration of the loop.
- `ConcLoopStep` (real number > 0): Step in the concentration loop.
- `Therm_ratio` (0 < real number < 1): Fraction of the total number of simulation steps to be included in the thermalization process.

A more detailed description of all these parameters is included in Section 5, as mentioned above.

2 Simulation basics

To begin with the simulation, we define our system to be in a given configuration, with particles at arbitrary positions. In the first steps (initial and subsequent particle configurations), the system will not comply the statistics of the canonical ensemble, as it is normally far away from equilibrium. As mentioned, these first steps of the simulations, the so-called "thermalization process", must be ignored in the analysis of the equilibrium properties.

The evolution of our system is leaded by the MMC method, according to an (NVT) ensemble, performing recurrent random attempts for the displacements of the system particles with aleatory length and direction.

In order to reach the equilibrium, a Markov process [2] is carried out. In this process, the transition probability, Tr, of any step is independent of the previous configurations, and is given by:

$$\begin{aligned} Tr\left(r_{\text{old}}^{2N} - r_{\text{new}}^{2N}\right) &= \min\left\{\exp\left(-\beta\Delta U\left(r_{\text{old}}^{2N}, r_{\text{new}}^{2N}\right)\right), 1\right\} \\ &= \min\left\{\exp\left[-\beta U\left(r_{\text{old}}^{2N}\right) - \beta U\left(r_{\text{new}}^{2N}\right)\right], 1\right\} \end{aligned} \tag{1}$$

where r_{old}^{2N} and r_{new}^{2N} refer to the configuration coordinates before and after the random movement, respectively.

The recursive attempts given by the MMC method follow a distribution weighted according the statistics of the canonical ensemble, where the probability to obtain a certain configuration is proportional to the Boltzmann factor, given by:

$$P\left(r^{2N}\right) = \frac{\exp\left[-\beta U\left(r^{2N}\right)\right]}{Z} \tag{2}$$

where Z is the partition function of the system, $\beta = (k_{\text{B}}T)^{-1}$ being k_B the Boltzmann constant, T the selected temperature, and $U(r^{2N})$ is the potential energy of the system, which in 2D depends on the $2N$ spatial coordinates of the N particles. The partition function Z is the sum of the probabilities of all possible configurations of the system, being the total probability normalized to 1.

The probability given by (1) is depicted in Figure 1, showing that every attempt of movement to a new configuration with lower potential energy will be accepted, while configurations with higher energy will have a transition probability weighted by the Boltzmann factor, as given by equation (2). For a large enough number of attempts (accepted or not), and regardless of the initial configuration selected, the equilibrium should be achieved.

Finally, the pairwise potential interaction between particles must be established in order to perform the MMC method. For the first examples, we have selected a 6-12 Lennard-Jones (L-J) potential (eq. 3 and Fig. 2), which is typically used to simulate the dominant dispersion forces in Argon-Argon interactions.

$$U(r_{1,2}) = 4\varepsilon\left[\left(\frac{\sigma}{r_{1,2}}\right)^{12} - \left(\frac{\sigma}{r_{1,2}}\right)^{6}\right] \tag{3}$$

The L-J potential is shown in Figure 2, where ε is the dissociation energy of the pairwise interaction (the depth of the potential well), σ is the distance where the potential energy becomes zero and $r_{1,2}$ is the separation between particles 1 and 2. In the examples included, both parameters are set by default to the experimental values of Argon gas $\sigma = 3.4$ Å and $\varepsilon = 0.01$ eV.

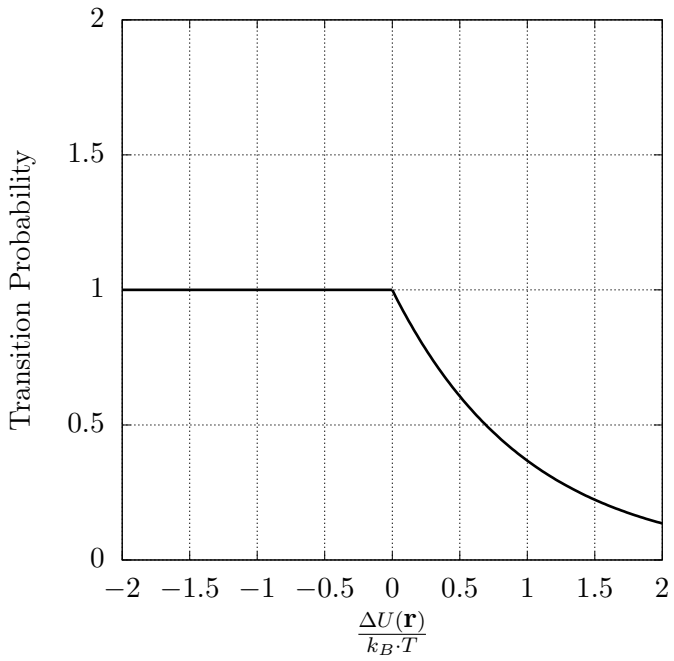


Figure 1: Transition probability (normalized to the thermal energy k_BT) vs the change of the potential energy.

3 Configuration requirements

The Monte Carlo 2D program is developed for Windows and Unix OSs. The representation scripts use `gnuplot` program, which should be installed and also added to the PATH[1] (see README file for more details).

The program can be unzipped and can be directly executed using one of the input files provided or creating your own simulation files.

In addition, if users find problems in the installation (or during the execution) of the program, support can be requested by email oscar.galvez@ccia.uned.es.

4 Running the program

Once gnuplot is installed and its folder added to the PATH (see the former section on gnuplot installation), you are ready to run the first simulation.

To run the simulation, you must open a CMD TERMINAL in the folder **where the mc program is installed** (we will call it the *working directory*) and type:

```
MC-WIN.bat input_file output_KEY
```

[1] `gnuplot` can be downloaded from: `http://www.gnuplot.info`

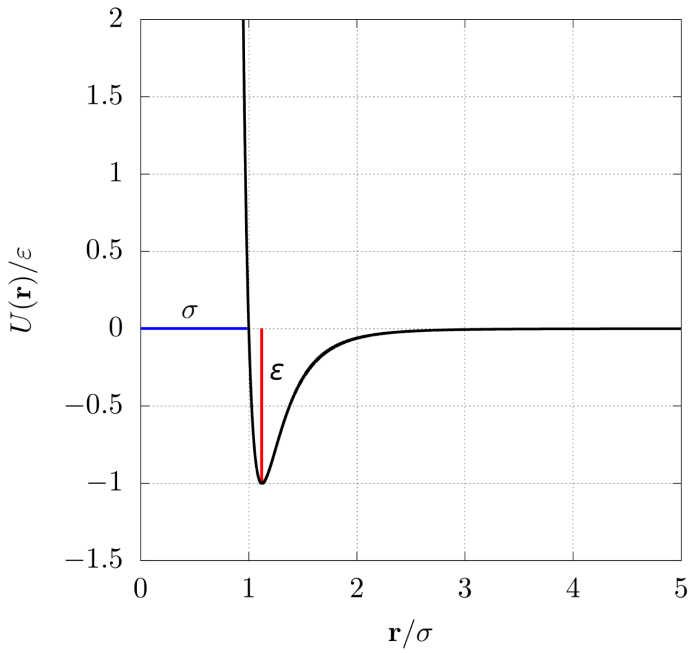


Figure 2: Lennard-Jones potential in reduced units.

The MC folder should contain the `input_file` (e.g. `in.txt`). Note that you should give the complete input file with its extension but without quotation marks.

The output directory `output_KEY` is then created by the program and the results of the simulation will be stored in that folder.

Some examples of input files are provided in the `/examples` folder. To check if the program is correctly running, the results corresponding to these examples are in the folder `/examples_result`.

Results will be placed into the output folder which name will be automatically determined by

`output_KEY-`[*concentration*]`-`[*temperature*]`-`[$NumParticles$]

If a loop over different temperatures or concentrations is performed, a folder for each of the temperatures and concentrations is created and an additional folder, with some of the main properties evaluated in the loop, is also created (named as `output_KEY-Loop-T`$[T_i - T_f]$`-C`$[C_i - C_f]$, where T_i, T_f, C_i and C_f are the initial and final temperatures and concentrations in the loop).

5 The Input File

There are two types of input variables which can be changed by the user. The *system variables* define the properties of the system from a physical point of view. An example of this type of variable is the potential function which describes the interaction between two

particles. The *simulation parameters* define how the simulation is performed (e.g. how to save CPU time and obtain fair results for our purposes).

The input file is a plain text file in which both kinds of variables are selected by the user. Each variable has a *keyword* which is followed by a particular value (separated by spaces and/or tabulations). These keywords must appear only once in the input file. Next, we will define all the system variables and simulation parameters and their associated keywords.

System Variables

- `Temperature_sys` (real number > 0, Kelvin): It defines the temperature of the system. This value affects the probability of evolution of the system during the random movements (see sections before). If a simulation of an automatic range of temperatures is selected, the temperature values are defined by the variables relative to this loop (`TempLoopInit`, `TempLoopFin` and `TempLoopStep`).

- `Concentration_sys` (real number > 0, particles/Å^2): Concentration of particles in 2D. This variable and the number of particles define the area of our system. The simulated box is always a square with periodic boundary conditions. The initial configuration in which particles are located can be defined by the `Condensed_ini` variable.

- `Particles_Num` (integer > 0): Number of particles of the system. The area of the system is defined by multiplying the `Concentration_sys` variable by the value of this variable.

- `Epsilon_lj` (real number > 0, eV): This variable fixes the depth of the potential well between particles for a Lennard-Jones or a Morse function.

- `Sigma_lj` (real number > 0, Å): This variable fixes the width of the potential energy functions. It refers to the distance for which potential energy is 0 in a Lennard-Jones potential, the bond distance for a Morse potential, and the radius of the hard disk for a Hard Disk model.

- `PotentialEnergy` (integer): It determines which potential function is selected in the simulation. (1) for Lennard-Jones, (2) for Morse, and (3) for a Hard Disk potential. If Lennard-Jones potential is chosen, `Epsilon_lj` and `Sigma_lj` variables are required. For Morse potential, the `WellWidth` parameter should also be given and for Hard Disks potential we must only define the `Sigma_lj` parameter.

Simulation Parameters

- `CutofRadius` (real number > 0, Å): Maximum Distance to compute the inter-particle potential function. A fair choice for this value should be about $2 \sim 4$ `Sigma_lj`. This radius must be lower than `VerletRadius` in order to compute all the interactions.

Note that large differences between `VerletRadius` and `CuttofRadius` imply that Verlet list is only occasionally recalculated, reducing the CPU time. But, if the `VerletRadius` value is too large, Verlet list will include too many elements, and every step of the simulation will take a long CPU time.

- `VerletRadius` (real number > 0, Å): The program uses a Verlet list to reduce the computation time. It stores all neighbors of each atom contained within a radius defined by this variable. This radius must be greater than `CutofRadius` in order to compute all the interactions. The Verlet List is updated when a particle is moved more than a certain value, which is set as the half of the difference between `VerletRadius` and `CutofRadius`. In many cases, a good choice for this parameter is to select it about `CutofRadius` + $(2 \sim 4)$ `Sigma_lj`. Note that for systems with many particles, the Verlet List calculation is one of the most CPU demanding parts, as it scales with N^2.

- `Nsampling` (integer > 0): Number of samplings used to evaluate general system properties. This sampling is equidistantly done along the entire simulation. For the equilibrium properties, the thermalization process is excluded of this sampling (it is controlled by the `Therm_ratio` parameter).

- `Rneighbor` (real number > 0, Å): Maximum distance for which particles are considered closest-neighbors. This definition affects to the calculation of the local orientation of each particle, which is determined by the "bonds" among closest neighbors. If a small value is chosen, part of the closest neighbors will be omitted; and for large values, second (or additional) neighbors could be taken into account. In order to choose a correct value, we can run a previous simulation and examine the RDF function. The `Rneighbor` must be set around the value of the first minimum which is found between the first peak (due to the closest neighbors) and the second peak (due to second neighbors). Commonly, `Rneighbor` will take values around $1.25 \sim 1.75$ times the bond distance.

- `Nneighbor` (integer > 0): It determines the minimum number of neighbors surrounding a given particle to be considered as part of a condensed phase. The number of closest neighbors is defined by `Rneighbor`. These two parameters are relevant for the determination of the transition between condensed and gaseous phases.

- `Rdensity` (real number > 0, Å): Radius used to calculate the local density in which each particle is allocated. If a large value is selected, the local density computed will be very similar to the mean density of the system (given by the user), missing the small-scale details. In the other side, if we choose a very small value, we can miss the medium-scale effects, as e.g. the formation of small aggregates. So this value must be chosen according to the scale-size that we want to analyze.

- `SpatialGrid` (real number > 0, Å): Maximum displacement of a particle in each attempt of movement at temperature $T = 1$ K. Note that the maximum displacement

grows with $T^{1/2}$, until a maximum value (given by the `MaxSpatialGrid` parameter) is reached. If the displacements of the particles are too small, the configuration will evolve very slowly and many steps to obtain a good sampling over the configurational space will be necessary. In the other side, if the displacements are very large, many of the movement's attempts will be rejected, so the system will evolve slowly again. Commonly, an acceptance ratio of about 0.5 (50 % of the movements) is considered very adequate. This ratio is shown in the CMD TERMINAL when the program is running, so the user can look for a fair choice of this parameter.

- `MaxSpatialGrid` (real number > 0, Å): Maximum displacement of a movement at any temperature. Remember that the maximum displacements will grow with $T^{1/2}$ until it reaches the value of this variable.

- `Therm_ratio` ($0 <$ real number < 1): Fraction of the total number of simulation steps to be included in the thermalization process. If a fast thermalization process is desired, we can set a low value for this parameter, and vice versa. We must take into account that if a long thermalization is carried out, the number of samplings used for the calculation of the equilibrium properties could be too small (see `Nsampling`).

- `TempLoop` (0 or 1): Option to automatically simulate over a given range of temperatures (1) or not (0). If the loop over temperatures is selected (1), this option overrides the `Temperature_sys` value and several simulations will be performed, taking as initial configurations the final configurations obtained in the previous simulation. The temperature will be changed at the beginning of each new simulation, according to the `TempLoopInit`, `TempLoopFin` and `TempLoopStep` parameters.

- `TempLoopInit` (real number > 0, K): Initial temperature of the loop. It can take values smaller or larger than `TempLoopFin` for "heating" or "cooling" the system, respectively.

- `TempLoopFin`(real number > 0, K): Final temperature of the loop.

- `TempLoopStep`(real number > 0, K): Step size in the temperature loop.

- `ConcLoop` (0 or 1): Option to automatically simulate over a given range of particles concentrations (1) or not (0). If the loop over concentration is selected (1), this option overrides the `Concentration_sys` value and several simulations will be performed. The concentration will be changed at the beginning of each new simulation, according to the `ConcLoopInit`, `ConcLoopFin` and `ConcLoopStep` parameters.

- `ConcLoopInit` (real number > 0, part/ Å^2): Initial concentration of the loop. It can take values smaller or larger than `ConcLoopFin` for "compressing" or "uncompressing" the system, respectively

- `ConcLoopFin` (real number > 0, part/ Å^2): Final concentration of the loop.

- `ConcLoopStep` (real number > 0, part/ Å^2): Step size in the concentration loop.

6 The Output Files

When the simulation is done, several output files are generated. For every simulation, a folder, which contains all output files, is created and named as:

$$\texttt{output\_KEY-}[concentration]\texttt{-}[temperature]\texttt{-}[NumParticles]$$

In this folder, a resume file is included with the all data of the simulation (`calc_resume.txt`). If a loop over different temperatures or concentrations is performed, several folders will be created (one for each value of temperature and/or concentration), containing the output files of each simulation. Also an additional folder named as:

$$\texttt{output\_KEY-Loop-T}[T_i]\texttt{-}[T_f]\texttt{-C}[C_i]\texttt{-}[C_f]$$

where T_i, T_f, C_i, C_f are respectively the variables `TempLoopInit`, `TempLoopFin`, `ConcLoopInit` and `ConcLoopFin` defined above. If the same `output-KEY`, temperature, concentration and number of particles are chosen, the output files are overwritten.

In the output folder several text files are included, containing the results of the equilibrium properties calculated, and the changes along the simulations of relevant parameters. The files generated are the following:

- `energias.txt`: In this file, the energy (in eV) of the system is given for each of the sampling steps. The sampling steps are performed both at the thermalization and the equilibrium stages, distributed equidistantly along the simulation. The number of samplings can be set by the `Nsampling` variable (see sections above). The energy evolution can be checked in order to analyze the degree of thermalization of the system.

- `dinamica.txt`: This file stores the positions and local orientations of the particles in the sampling steps. The two first columns are the spatial coordinates, while the third and fourth columns are related to the local orientation of the particles (see details above). In the third column the modulus of the local orientation of each particle is included, which is defined as 1 for a particle with 6 neighbors in an ideal hexagonal pattern and decreasing to 0 as the hexagonal ordering is being lost. The fourth column contains the direction of the local orientation of each particle, varying from -180 to +180. A value of 0 refers to a particle with its closest neighbors oriented in a triangular lattice with a side parallel to the X axis. When this side forms $\pm 30^\circ$ with the X axis, this local parameter becomes ± 180.

- `density_distribution.txt`: This file stores two columns with the normalized local density distribution. The first column includes the values of the normalized local density, whereas the second column contains the averaged probability of each of these densities. It is normalized to density=1 for a non-interacting system (gas phase). This distribution can be analyzed to distinguish different aggregation states in our system. For example, in a pure condensed phase, a single peak with a density

value higher than one will be observed. On the contrary, for a gaseous phase a peak around 1 appears, and for a coexistence of both phases, we will observe two different peaks, one in a density lower than 1 and another with density much greater than 1.

- `rdf.txt`: The RDF values are stored in this file. In the first column, the distance (in Å) from the particle is printed, whereas the normalized probability to find another particle for this distance is given in the second column. This probability is also normalized to 1 for every distance in a non-interacting system. In the third column the RDF value minus 1 is included, for the case that the correlation of this parameter needs to be analyzed. The RDF contains information about the most probable distances among particles. The height of the different peaks is related to the translational ordering of the particles (the degree of crystallization of the system).

- `distribucion.txt`: This file contains the equivalent to the RDF distribution but with angular resolution (in a 2D grid). In the two first columns, the position (in Å) along the plane is given (x and y coordinates), and in the third column the normalized probability to find a particle at the given position. This probability is normalized in the same way than the RDF, being 1 at every position for the non-interacting system. As for the RDF, in the fourth column, the normalized probability minus 1 is printed.

- `orientation.txt`: The information about the mean orientation of the system at every sampling step is included in this file. If the system presents a defined orientation, this value will be close to 1, while for not oriented systems it will take values near 0. The evolution of this parameter is useful to determine whether the system has reached the equilibrium, but also to distinguish different phases. For solid or hexatic phases it will take values near 1, but for liquid and gaseous phases it will be approximately zero, as they have non-orientational order.

- `orient_correlation.txt`: The spatial correlation of the orientational parameter is stored in this file. The distance to the particles (in Å) is included in the first column, and the second column corresponds to the spatial correlation of the orientational parameter. It will take values near 1 for well-oriented systems, and it will decay to 0 for non-oriented systems.

7 Visualization of the results

In order visualize the results, we provide some representation scripts in the folder `/gnuplot`.

From the CMD TERMINAL in the working directory (i.e. where the Random-Phase program is installed) it must be typed:

`GNU.bat` *keyword* `'output_directory' number_of_particles_simulated`

important: `output_directory` should be written between simple quotes: `'output_directory'`, for example:

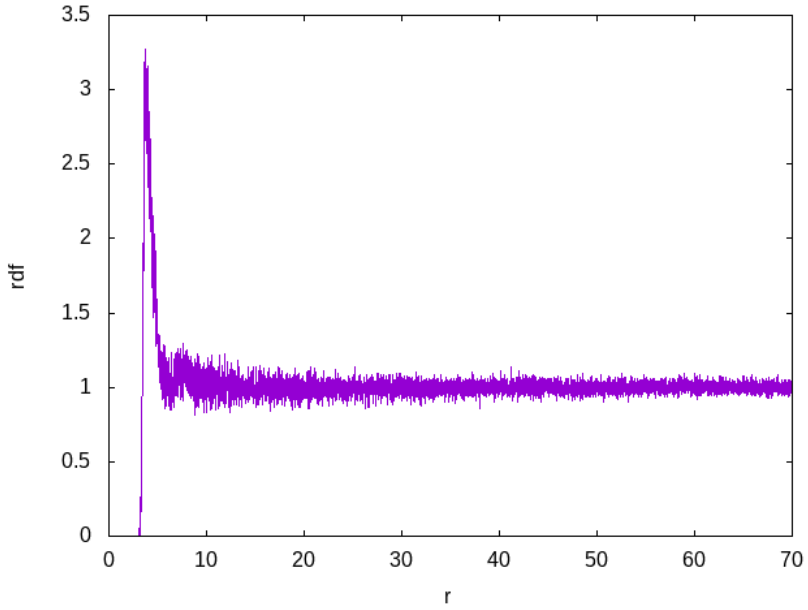


Figure 3: Radial Distribution Function (RDF) of a system with concentration 0.01 part/Å^2 and $T = 100$ K.

For an `output_KEY` = `TEST` and using the file `in.txt` given in the distribution, to plot the RDF function, it should be typed:

```
GNU.bat rdf 'TEST-200-0.01-55-200' 200
```

keyword refers to any of the presentation functions included in the folder `/gnuplot`, as distributed with the program. They are:

- *keyword* = `density` (Local Density Distribution). The mean local density distribution obtained in the simulation, centered at the particles position.
- *keyword* = `energy` (Mean Particle Energy Evolution). The evolution of the mean potential energy per particle is computed at every certain steps.
- *keyword* = `rdf` (Radial Distribution Function, RDF). The RDF measures the probability of finding another particle at a certain distance from a given particle. It is normalized so the unit value is the probability for a non-interacting system.
- *keyword* = `distribution` (2D Distribution Function). It is the equivalent to the RDF, but taking into account not only the distance between particles but also the direction of the relative vector between them.

- *keyword* = `orientation` (Orientational Parameter Evolution). This parameter is computed at every certain number of steps. It takes a value of 1 for a perfect 2D crystal with hexagonal symmetry, and becomes close to 0 for a system with no defined orientation.

- *keyword* = `orientation-corr` (Orientational Correlation Function). This property measures the spatial correlation of the orientational parameter. It takes a value of 1 for strongly correlated orientations and 0 for non-correlated orientations at a certain distance.

- *keyword* = `spec-heat` (Specific Heat Evolution). The specific heat is computed at every certain number of steps.

- *keyword* = `dynamic` (Configuration Evolution). The positions of the particles are printed at every certain number of steps. In order to visualize the evolution of the different configurations, an animation is shown.

- *keyword* = `dynamic-or` (Configuration Evolution 2). The positions and orientations of the particles are printed at every certain number of steps. In order to visualize the evolution of the different configurations, taking into account their orientation, an animation is shown.

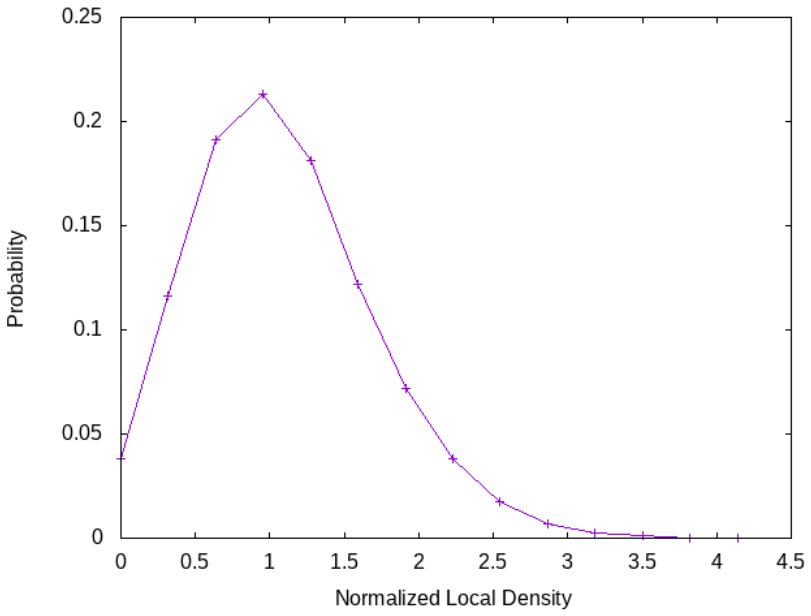


Figure 4: Normalized Local Density of a system with concentration 0.01 part/Å^2 and $T = 100$ K.

8 Examples

In this section, a brief description of some simulations and their results is discussed. These examples and the results are included in the `examples` folder.

For all cases, we have selected the Lennard-Jones potential function with the parametrization of Argon gas, i.e. $\sigma = 3.4$ Å and $\varepsilon = 0.01$ eV.

8.1 Gaseous Phase

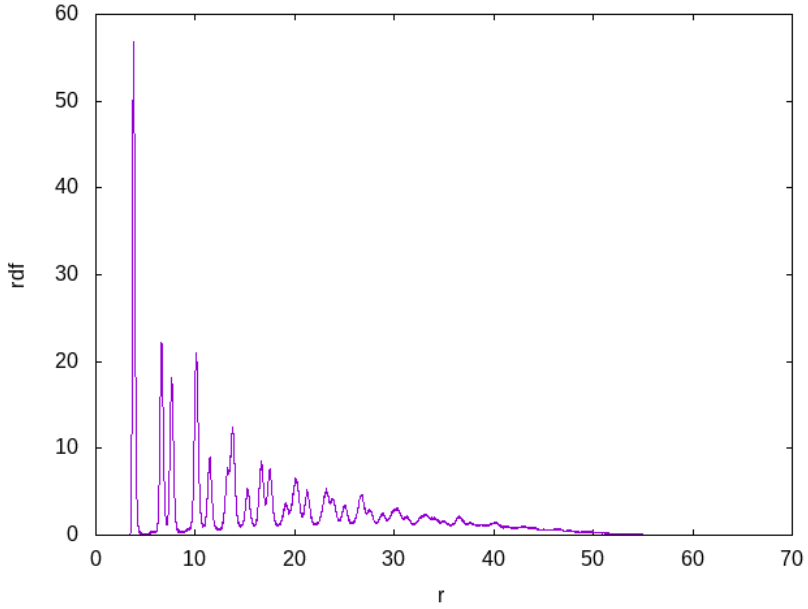


Figure 5: Radial Distribution Function (RDF) of a system with concentration 0.01 part/Å^2 and $T = 10$ K.

In this example, the system is composed of 200 particles, with a low concentration of 0.01 part/Å^2. The temperature of the system is $T = 100$ K, which is high enough to simulate a gaseous phase. As it is shown in Fig. 3, the RDF function becomes 1 for every distance greater than the particle's diameter (aprox. σ), which reveals the lack of translational order. In Fig. 4, we can see that the density distribution presents an only peak centered in the value corresponding to the non-interacting system, i.e. 1. From this two analysis, we can conclude that the system is in a gaseous phase.

8.2 Solid Phase

In this example, the system is composed of 200 particles, with a concentration 0.01 part/Å^2 and a temperature of $T = 10$ K, which is adequate to simulate the solid phase. Fig. 5 shows the RDF function, which presents several peaks and regions with almost zero probability

to find a particle. This result is related to the ordering of the particles along the surface, showing that the particles present some favored relative positions. We can also associate the position of the different peaks with those occupied by the closest neighbors, second neighbors, etc. in a triangular lattice. This lattice should be the expected for an inter-particle potential with radial symmetry in a 2D system, as it is the case.

In Figure 6, we can see that the local density distribution also shows an only peak, but around a value much greater than that obtained for a gaseous phase. This result is in agreement with the simulation of a condensed phase. In addition, due to the translational order is kept for long distances; we can even say wheather our system is in a "solid" or crystalline phase.

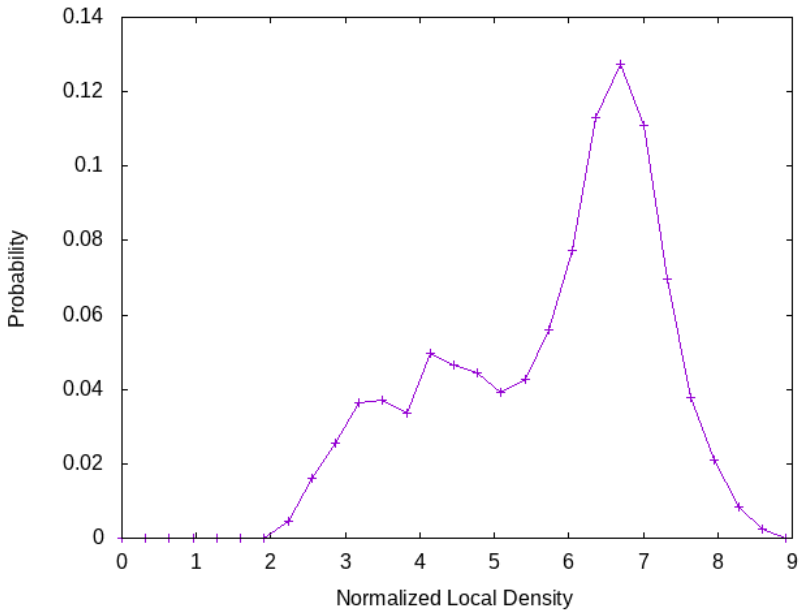


Figure 6: Normalized Local Density of a system with concentration 0.01 part/Å^2 and $T = 10$ K.

8.3 Loop over Temperatures

In this example, we will simulate a system composed of 200 particles with a concentration of 0.01 part/Å^2. The temperature of the system will be increased from 10 to 100 K, in order to evaluate the transition between condensed and diluted phase. Fig. 7 shows the normalized local density, which decreases from high values obtained at low temperatures (around 5.7), to values close to 1 (for the non-interacting system) when the temperature is increased.

In Figure 8, we can see the specific heat of the system during the transition, showing a maximum value at the transition temperature between the condensed and diluted

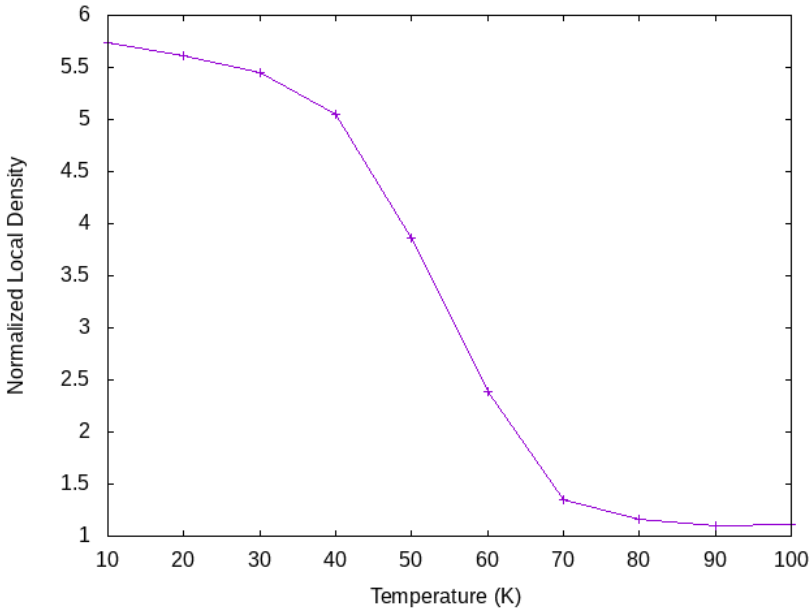


Figure 7: Normalized Local Density obtained for different temperatures for a system with concentration 0.01 part/Å^2.

phase, which agree with the temperature where the local density values start to decrease significantly (Fig. 7).

Acknowledgments

This project has been developed within the framework of the *Grupo de Innovación Docente en Física* (GIDF), GIF2016-25, Facultad de Ciencias UNED.

References

1. *Understanding Phase Transitions with the Metropolis Monte Carlo method*, The Physics Teacher, to be published, 2019.

2. D. Frenkel, B. Smit. *Understanding Molecular Simulation*, Academic Press, USA, 1996.

3. N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. N. Teller and E. Teller. *Equation of state calculations by fast computing machines*, J. Chem Phys., 21:1087-1092, 1953.

4. C.D. Birkhoff. *Proof or ergodic theorem*, Proceedings of the National Academy of Sciences of the United States of America, 1931, 17, 656-650.

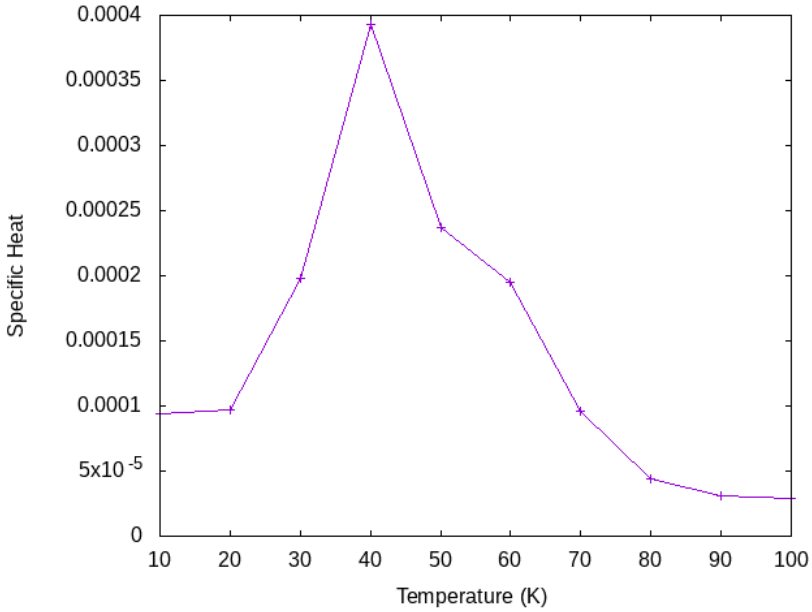


Figure 8: Specific heat obtained for different temperatures for a system with concentration 0.01 part/Å^2.

5. M. E. J. Newman, G. T. Barkema. *Monte Carlo Methods in Statistical Physics*, Oxford University Press, USA (1999).