membrane wants to maximize the freedom of its fluctuations. Since these inclusion act as as boundary conditions for the fluctuations independently of their distance, these forces can manifest as a very strong and long range effect[86].

All these interaction mechanisms have been found and tested originally for protein inclusions, but are valid also for protein-mimicking objects like NPs. Those mechanisms that involve fine deformations of the membrane are particularly apt to be studied using molecular level models, such as the simulations we are going to present in this thesis.

1.3.2 *Aggregation at the mesoscale: liposomes*

Moving on to mesoscale systems, we have another kind of aggregation which is strictly related to membrane reshaping properties: that of liposomes. As mentioned in a previous section, liposomes are spherical vesicles of lipid bilayer, that incapsulate fluid at the inside, and are naturally formed by patches of lipid bilayer isolated in water. Liposomes are important as means of intra and inter-cellular transport of small molecules[216, 122, 145]; they are used as experimental models of lipid membranes[216, 122, 156] and as drug delivery agents in biomedical applications[172, 159].

The aggregation of liposomes is fundamental in many fields of applications[30, 122], for example due to the possibility for liposome complexes to be used as delivery agents in depot injections therapies[159, 26]. Moreover, liposome aggregation is a basic step in the modeling of membrane fusion[127, 122, 156], a process of paramount biological importance. The adhesion of small vesicles to the membrane of the cell or of another organelle can be mediated by small molecules or ions[165, 122, 127] or by specialized membrane proteins[214, 122]. The formation of vesicles is itself due to collective behavior of membrane proteins and protein complexes[122, 156, 149], an example of the aggregation mechanisms that are linked at different scales.

2

COMPUTATIONAL METHODS

For the research presented in this work, we relied on computational methods, i.e. we used theoretical models of interactions in order to simulate matter using computers: thought experiments extecuted *in silico*. In particular, the main technique that we used is Molecular Dynamics (MD).

At these scales of length and time, the motion of a system composed of many atoms and molecules in time and space is well described by the classical Newton's equations of motion. Indeed, the idea behind MD simulation is that of reproducing the equilibrium thermodynamic properties of a many-body system approximating its real time-evolution. In order to achieve this result, two ingredients are necessary: the knowledge of the equations of motion of the system and a model of their interaction. This method has the peculiarity of yielding both equilibrium averages and non-equilibrium, dynamical description of the system [198]. Due to the large number of degrees of freedom of our systems, typically composed by tens or hundreds of thousand particles, the equations of motion must be integrated numerically. The integrator and several other algorithms for the management of the system's thermodynamic conditions (pressure, temperature...) constitute the core of the Molecular Dynamics technique.

In this chapter we will present a technical description of the MD technique based on the books of Leach [105] and Frenkel [52]. In Section 2.2, we will describe some of the more advanced sampling techniques used in the work presented in the next chapters. These techniques, which are still based on MD, involve external manipulation of the simulated systems in order to work around some of the limitations of classic MD, allowing us to explore larger parts of the configurational space.

2.1 MOLECULAR DYNAMICS SIMULATIONS

Classical Molecular Dynamics (MD) is a set of computational techniques that consist in generating time sequential configurations of the system by numerical integration of Newton's equations of motion. MD simulations are in many aspects similar to real experiments: we set up a system with a set of control variables, and we analyze the averages taken from the time evolution. The advantage is that, unlike in real experiments, the process of setting up the initial configuration of the system is generally easy and the simulations are reproducible and adjustable by only

changing a few control parameters; furthermore, simulations are usually cleaner than experiments, meaning that much of the unwanted interference on the system is eliminated. The price to pay is that the reliability of these simulations heavily depends on both the numerical accuracy and the quality of the interaction model used to define the potential energy function (PEF). Moreover, these simulations are limited in scale and time by the available computational power.

The essential idea is that the integration is broken down into many small steps, each separated in time by a fixed constant δt, the *MD time step*; in this time interval forces are considered to be constant. From the forces we can determine the accelerations of the particles, which are then combined with the positions and the velocities at time t to calculate the positions and velocities at time $t + \delta t$. This is the fundamental step, repeated over the whole time of the simulation. Hence we get a *trajectory*, a set of space phase vectors taken at discrete instants, each a multiple of the time step.

The averages are now computed not as integrations over time, but are a sum over discrete snapshots of the trajectory, say at intervals $\Delta\tau = i\delta t$, $i = 1, 2, 3, \ldots$; when $i > 1$, only a subset of the trajectory is considered for the average, that can be now be expressed as

$$\langle A \rangle = \frac{1}{M} \sum_{n=1}^{M} A(\vec{x}(n\Delta\tau)) \tag{2.1.1}$$

where $M\Delta\tau$ is the total time averaged. The steps of a general MD simulation can be represented as the scheme in figure 2.1.

The forces are calculated from a PEF which parameters are usually constant during the simulation and have to be provided to the MD software before the start. These parameters, along with all the other information that describes the nature of the particles and their interactions, are stored in the so called *force fields*, which we will describe in detail in the next section.

Initial configuration setup

The first step required to perform a MD simulation is to generate an initial configuration, which means a microstate apt to be the start of the simulation. This choice can be non-trivial, since the initial configuration should be coherent with the conditions we want to set up for the entire simulation. In fact, the more the system is complex, the more it will be difficult for it, when not even impossible, to reach a stable configuration.

To generate the initial configuration, it is necessary to assign to each particle its initial position and velocity, which are the $6N$ elements of the phase space vector. The choice of position must take account of the PEF: it can't be too far from that of equilibrium, otherwise problems could arise (for instance force spikes), breaking the system after the first few steps of simulation. A common choice for the velocities

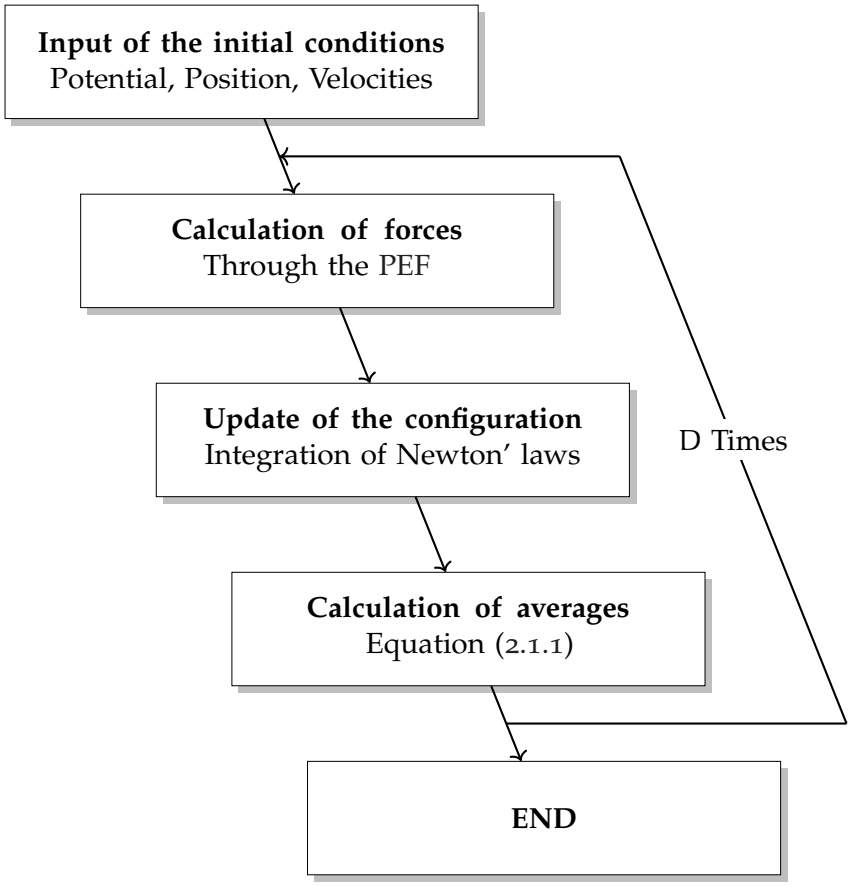

Figure 2.1: Schematic representation of a MD simulation.

is to pick one for each particle extracted randomly from the Maxwell–Boltzmann distribution function at the desired temperature:

$$f(v_i) = \sqrt{\frac{m_i}{2\pi k_B T}} e^{-\left(\frac{m_i v_i^2}{2k_B T}\right)}$$

A random set of velocities will have a non zero center of mass (COM) motion:

$$\vec{v}_{COM} = \frac{\sum_{i=1}^{N} m_i \vec{v}_i}{\sum_{i=1}^{N} m_i} \tag{2.1.2}$$

This is not physically wrong, but nonetheless inconvenient; hence a COM motion removal is usually performed, a fact that needs to be taken in account when counting the degrees of freedom (DOF) of the systems, since after that they are $6N - 3$.

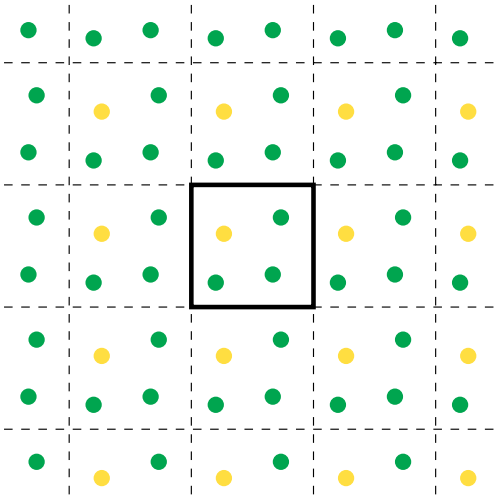

Figure 2.2: Schematic view of a two–dimensional box with PBC imposed. The central, thick contoured, box is the simulation box and it is replicated along each side. One particle is highlighted in gold to show its periodic images

2.1.1 *Periodic boundary conditions*

If the simulated system represents only a small section of a larger bulk body, which is the case for most of the simulations performed in biomolecular physics, the most used method is that of implementing periodic boundary conditions (PBC) to the simulation box. That means that when a particle moves outside the walls of the box, it reappears with the same velocity at the opposite side. This is obtained, for instance, subtracting the size of the box from the x_i coordinate of the particle, where x_i is the direction in which the particle has trespassed the boundary of the box.

This implies that the choice of the box becomes limited to filling-space ones, making it impossible to simulate for instance a spherical system. However, PBC allow to avoid confinement effects and to consider the system as immersed in an infinite bulk. The shape of the space-filling box is chosen according to the characteristic dimension of the simulated system, e.g. a flat membrane system is apt to be simulated in a parallelepiped box.

As shown in figure 2.2, imposing PBC conditions can be seen as placing infinite copies of the box along each other, so that each particle has infinite images. To calculate short-range non bonded interactions, such as Van der Waals interaction, only the first periodic image of a particle is considered.

PBC implementation however has not negligible side effects. Firstly, it makes impossible to see collective fluctuations larger than the size of the box, hence preventing to simulate those modes of vibrations. Moreover, it makes it difficult to treat long-range interactions, like electrostatics.

2.1.2 *Numerical integrators*

As we already explained, the equations of motion are resolved numerically at discrete times; this approach is called *finite differences* method. To obtain a relation

between coordinates and velocities of successive time steps, the simplest way is to start from a Taylor expansion of the trajectory:

$$\vec{r}_i(t+\delta t) = \vec{r}_i(t) + \dot{\vec{r}}_i(t)\ \delta t + \frac{1}{2}\ddot{r}_i(t)\ \delta t^2 + O(\delta t^3)$$

and then using Newton's equation $\vec{f} = m\ddot{\vec{r}}(t)$ (where $\vec{f}$ is the total force exerted on material point and m its mass), which gives

$$\vec{r}_i(t+\delta t) = \vec{r}_i(t) + \vec{v}_i(t)\ \delta t + \frac{1}{2m_i}\vec{f}_i(t)\ \delta t^2\ +\ O(\delta t^3) \tag{2.1.3}$$

where we identified the velocities $\vec{v}_i(t) = \dot{\vec{r}}_i(t)$ and the forces $m_i\ddot{\vec{r}}_i(t) = \vec{f}_i(t)$.

There are many algorithms that are based on equation 2.1.3. We will focus on the Verlet algorithm and on one of its implementations, the Leap–frog algorithm, which is the most relevant for this work of thesis, since it's the default used by GROMACS.

Verlet Algorithm

Taking the Taylor expansion of the trajectory like in (2.1.3) but expanding in the negative direction of time, we obtain

$$\vec{r}_i(t-\delta t) = \vec{r}_i(t) - \vec{v}_i(t)\ \delta t + \frac{1}{2m_i}\vec{f}_i(t)\ \delta t^2\ +\ O(\delta t^3)$$

Summing this to (2.1.3) gives

$$\vec{r}_i(t+\delta t) + \vec{r}_i(t-\delta t) = 2\vec{r}_i(t) + \frac{1}{m_i}\vec{f}_i(t)\ \delta t^2\ +\ O(\delta t^3)$$

and then, excluding the terms proportional to δt^3, we obtain the Verlet algorithm:

$$\vec{r}_i(t+\delta t)\ =\ 2\vec{r}_i(t)\ - \vec{r}_i(t-\delta t) + \ \frac{1}{m_i}\vec{f}_i(t)\ \delta t^2 + \cancel{O(\delta t^4)} \tag{2.1.4}$$

Equation (2.1.4) shows that the Verlet algorithm only generates positions. To obtain velocities, we can construct them at any instant using

$$\vec{v}_i = \frac{\vec{r}_i(t+\delta t) - \vec{r}_i(t-\delta t)}{2\delta t}$$

Moreover, the first step at which $t = 0$ must be computed by other means, since the term $\vec{r}_i(t-\delta t) = \vec{r}_i(-\delta t)$ is obviously not available. Another issue of this algorithm is that the position are obtained by adding a small number ($\frac{1}{m_i}\vec{f}_i(t)\ \delta t^2$) to the difference of two much larger numbers ($2\vec{r}_i(t)\ - \vec{r}_i(t-\delta t)$), which leads to a loss in precision.

Leap–frog algorithm

The fact that the Verlet algorithm doesn't take care of velocities is somewhat inelegant, since they are coordinates of the phase space as well as positions: indeed the velocities are not available for a certain step until the positions of the following step are computed. The *leap–frog algorithm* solves this, implementing these propagation relations:

$$\begin{aligned} \vec{r}_i(t+\delta t) &= \vec{r}_i(t) + \vec{v}_i(t+\frac{1}{2}\delta t)\ \delta t \\ \vec{v}_i(t+\frac{1}{2}\delta t) &= \vec{v}_i(t-\frac{1}{2}\delta t) + \frac{1}{m_i}\vec{f}_i(t)\ \delta t \end{aligned} \tag{2.1.5}$$

In this algorithm positions and velocities are both computed explicitly, although at alternate times; the precision problem is also solved since the calculation of the difference between two large numbers is no more required.

If we want to calculate averages such as the kinetic energy and the PEF in a way in which they are comparable, however, we must calculate the velocities at the same time of positions. This is done by the equation

$$\vec{v}_i(t) = \frac{\vec{v}_i(t+1/2\delta t) + \vec{v}_i(t-1/2\delta t)}{2}$$

Trajectories of the Verlet algorithm and of the leap–frog are identical, excluding numerical errors.

2.1.3 *Neighbor list*

As the second block in figure 2.1, we need to calculate the forces at each time step, which means calculating the PEF. This is one of the most expensive parts of the simulation in terms of computational time: in principle, it should be done calculating for each particle, at each time step, the interaction with every other particle and their periodic images (and firstly their distance).

Usually this problem is solved using a *cut-off distance*, which act as a maximum distance at which interactions are computed. This is done modifying the PEF, truncating it within the cut-off distance. This can be summarized with the following expressions:

$$U_i(\vec{r}) = \sum_{j\neq i}^{N} U^*_{ij}(\vec{r}), \qquad U^*_{ij}(\vec{r}) = \begin{cases} U_{ij}(\vec{r}) & \|\vec{r}_i - \vec{r}_j\| \leqslant r_c \\ 0 & \text{otherwise} \end{cases}$$

where U_{ij} is the two-particle potential between particles i and j, and r_c is the cut-off distance.

This first approximation is not enough, since, even if we know that at certain distance two particles don't interact we still have to calculate it for every couple. This problem is worked around by the implementation of a *neighbor pair–list*, which gives a significant increase of performance.

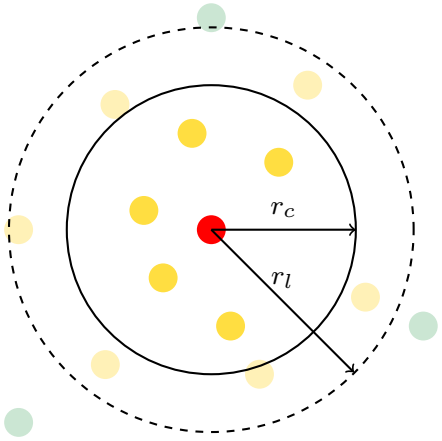

Figure 2.3: Schematic representation of the buffered pair–list construction respect to the central red particle. gold particles are in the pair–list below the cutoff radius r_c, therefore included in the calculation of the interactions. Faded gold particles are in the pair–list for which at every step is checked if their distances became smaller then r_c. Light green particles are not in the pair–list and they are completely neglected until the next list update.

The simplest method that can be implemented to generate a neighbor list consists in storing, for each particle, a list of particles that lies within distance r_c from it. Interactions are then computed only within particles in the same list. This method computationally is effective if the list is updated every M steps, with $M > 1$; the larger is M, the more the performance will be improved, but the energy drift will start to be not negligible, since particles in fluid systems can diffuse in and out of the boundaries of the list. Anyway, depending on the used model, an acceptable compromise can be reached; for instance, for an atomistic model of water simulated at ambient pressure and temperature, the neighbors of don't change significantly for $M < 20$, with a time step of about $\sim 1-2$ fs, before the list is updated.

VERLET CAGE Even with a small value of M, it is clear that an unverifiable, although small, variation in the neighbor list is to be expected, and this leads to an under– or over–estimation of the inter–particle energy contribution. A method that partially overcomes the problems of the group scheme has been proposed by Verlet [206]. This scheme, the *Verlet scheme*, is based on the use of a buffered list, which means the the neighbors in the list are selected to be at a distance $r_V > r_c$, and at every time step, the interaction is computed only for couples which distant is within r_c, thus implementing a group scheme inside the Verlet neighbor list. At the cost of slightly increase the computing time, we know neglect almost no interactions. The Verlet scheme is illustrated in figure 2.3.

There are also two expedients that allows the Verlet scheme to work more efficiently. The first is available when simulating a system with constant temperature coupling, and consists in fixing an upper limit to the energy drift and accordingly determine the buffer radius dynamically during the simulation. The second approach is to consider the maximum distance $\Delta_i = |\vec{r}_i(t+\Delta t) - \vec{r}_i(t)|$ traveled by a particle in the list: when that value is large enough to have permitted to a particle to exit the the cut-off distance, the list is automatically updated; doing so allows one to decrease the refresh rate without deteriorating the accuracy.

2.1.4 *Temperature and pressure coupling*

Thermostat algorithms

Systems in the canonical ensemble are coupled to a thermal bath, which acts as an external reservoir that keeps the temperature of the system constant. A natural approach to obtain this in a simulation is provided by the fact that temperature is related to kinetic energy as in equation (2.1.6):

$$\langle K \rangle = \left\langle \sum_{i=1}^{N} \sum_{\alpha=1}^{3} \frac{p_{i\alpha}^2}{2m_i} \right\rangle = \left\langle \sum_{i=1}^{N} \frac{1}{2} m_i \vec{v}_i \cdot \vec{v}_i \right\rangle = \frac{3}{2} N k_B T \tag{2.1.6}$$

The idea is to scale velocities by a factor λ altering the instant kinetic energy, an hence the temperature. The simplest implementation of this algorithm consists in performing the scaling of velocities at each step, with a constant factor $\lambda = \sqrt{T_{bath}/T(t)}$. A more elaborate method, proposed by Berendsen et al. [19], is to couple the system with a sort of external thermal bath, which provides thermal energy to the system at a rate proportional to the temperature difference between the two:

$$\frac{dT}{dt} = \frac{1}{\tau}(T_{bath} - T(t)) \tag{2.1.7}$$

This is called *weak coupling algorithm*. Here τ is a measure of how much the system is separated from the thermal bath, meaning that in the limit $\tau = \delta t$ this routine is reduced to the simple scaling velocity method. These algorithms are simple but don't reproduce the exact canonical distribution; in fact they don't cure artifacts like difference in temperature between components of the system, which are artificially prolonged.

A first example of algorithm that reproduce the canonical distribution is that of *stochastic collision*, proposed by Andersen [8]. This consists in selecting a random particle (or a set of particles) periodically and reassigning it a new velocity extracted from the Maxwell-Boltzmann distribution. This is equivalente to coupling the system with a thermal bath that periodically sends impulses modifying the trajectory of particles; between these artificial collisions, the system is simulated as microcanonical. The main drawback is that the trajectories generated with this thermostat are not smooth, and present artificial deviations.

Another kind of algorithm that preserves the canonical distribution is that proposed by Nosé [139] and Hoover [69], which implements in the system a new artificial degree of freedom (DoF) s, that acts as dynamic scaling factor for the velocities and has a time evolution coupled with that of the system. This means that the thermal reservoir is an integral part of the system.

A last choice, and the one used for the work of this thesis, is to use a thermostat that combines the weak coupling and the stochastic collision algorithm; an implementation of this concept is the one suggested by Bussi [28]. This algorithm, the *velocity rescaling* thermostat, is a procedure that follows these steps:

1. Run the simulation for a step with unperturbed NVE dynamics

2. Calculate the kinetic energy

3. Evolve the kinetic energy for a step with an auxiliary stochastic propagation

4. Rescale the velocities with the new kinetic energy

The concept is to use stochastic dynamics as in Andersen thermostat to obtain the canonical distribution, while not changing abruptly the trajectories of particles. The evolution of K is given by:

$$dK = \frac{\langle K \rangle - K}{\tau_T}\, dt + 2\sqrt{\frac{K \langle K \rangle}{N_f \tau_T}}\, dW \tag{2.1.8}$$

Where τ is a parameter that can be identified with that from the weak coupling algorithm in equation (2.1.7), and dW is a noise parameter. This relation is reduced to the simple weak coupling algorithm when $\tau \to 0$, and to the microcanonical dynamics, i.e. without the thermostat, when $\tau \to 1$. While the system is far from equilibrium, the first component dominates bringing it to the required kinetic energy, which is then stabilized by the stochastic dynamics.

Barostat algorithms

To replicate in a MD simulation the isothermic-isobaric ensemble we need, besides a thermostat algorithm, a *barostat algorithm* to keep the system under constant temperature. As we saw in the previous subsection, a feature that all thermostats share is that they act on the velocities of the particles in the system; barostats, similarly, are all based on rescaling the positions. We illustrates two examples, which are both available in GROMACSand that we used for the work of this thesis.

BERENDSEN ALGORITHM A first approach is that suggested by Berendsen [19], which consists in scaling both volume and particle coordinates in a similar manner to that of the weak coupling algorithm for constant temperature. Indeed, the rate of change in pressure is given by

$$\frac{d\mathbf{P}}{dt} = \frac{\mathbf{P}_0 - \mathbf{P}}{\tau_P}$$

which is analog to equation (2.1.7), being $\mathbf{P}_0$ the target pressure, $\mathbf{P}$ the instant pressure and τ_P a coupling constant which represents the response time of the barostat. As in the weak coupling thermostat, the factor $\delta t/\tau_P$ allows to take in account a finite response time of the barostat. The corresponding volume scaling matrix results to be

$$\lambda_{ij} = \delta_{ij} - \beta_{ij}(P_{0ij} - P_{ij})\frac{\delta t}{\tau_P}$$

where $\boldsymbol{\beta}$ is the isothermal compressibility of the system. With this scheme, the positions are scaled by:

$$\vec{r}_i' = \lambda^{1/3}\vec{r}_i$$

This gives a general anisotropical scaling factor, that that be reduced to a isotropic scaling factor taking only one volume of pressure $P = {}^{Tr\{\mathbf{P}\}}/_{3}$.

This algorithm, as in the case of the weak coupling algorithm, has the flaw of not generating the right distributions of the isobaric ensemble; this problem is solved by the extended pressure–coupling systems.

PARRINELLO–RAHMAN ALGORITHM The most used extended pressure–coupling algorithm, and the one used in GROMACS, is the *Parrinello-Rahman barostat* [146, 147], which gives the right thermodynamics of the NPT ensemble, and it is the analog of the Nosé-Hoover thermostat for constant pressure algorithms.

In this algorithm the volume is treated as an independent variable of the system (we can think of it as an external piston of mass M_h), and its time evolution is given by the Euler-Lagrange equations of motion, as any other variable. This new variable is represented as a metric tensor which allows both the volume and the shape of the MD cell to vary with time.

Let $\mathbf{h}$ be the matrix formed by the edges vectors of the box $\{\vec{a}, \vec{b}, \vec{c}\}$ and $V = \vec{a} \cdot \vec{b} \times \vec{c} = |\det \mathbf{h}|$ its volume. The position of a particle i will be given by

$$\vec{r}_i = \xi_i \vec{a} + \nu_i \vec{b} + \zeta_i \vec{c} = \mathbf{h}\vec{s}_i$$

where $\vec{s}_i$ has components (ξ_i, ν_i, ζ_i) each going from 0 to 1. Moreover, let us write $\mathbf{G} = \mathbf{h}^T\mathbf{h}$, where $\mathbf{h}^T$ denotes the transposed matrix. Then the right Lagrangian for a system with N particles will be

$$\mathcal{L} = \frac{1}{2}\sum_{i=1}^{N} m_i \dot{\vec{s}}_i^T \mathbf{G} \dot{\vec{s}}_i - \sum_{i=1}^{N}\sum_{j=i+1}^{N} U_{ij}(\vec{r}) + \frac{1}{2}M_h \mathrm{Tr}(\dot{\mathbf{h}}^T\dot{\mathbf{h}}) - P_0|\det \mathbf{h}|$$

The first and the second terms are that of the normal kinetic energy and couple potential of the NVE Lagrangian, referring only to the particles of the system. The third term can be seen as the kinetic energy of the new DoFs given by the mobility of the dimensions of the box, with M_h acting as a virtual mass. The last term can be seen as the work done by the external force that keeps the pressure at constant level.

It can be shown that with this Lagrangian the correct isobaric ensemble is generated. Indeed, the Hamiltonian of this system can be approximated by

$$\mathcal{H} \simeq \langle K \rangle + U + P_0 V = H$$

where H is the enthalpy and we excluded the term corresponding to the kinetic energy of the box walls DoFs. Since this system is microcanonical, the Hamiltonian is a constant of motion, hence the enthalpy is conserved, and the isobaric system is correctly sampled.

It can be shown that the Parrinello-Rahman barostat evolves like an harmonic oscillator of mass M_h, which is responsible for the characteristic time that the system employs to reach the target pressure from a given initial pressure. The authors suggest that if the simulation is focused on the dynamic properties of the

system rather than the static ones, it is appropriate to choose M_h such that the relaxation time is of the order of L/c, L being the size of the box and c the sound velocity of the system.

Thanks to the harmonic nature of this barostat, equilibration time can result to be long, because of its inherent oscillating behavior; hence the Berendsen barostat is the best suited when equilibrating the system. For the actual simulation instead the Parrinello-Rahman one is to be preferred thanks to its property of correctly reproducing of isobaric distributions.

2.2 ADVANCED SAMPLING TECHNIQUES

2.2.1 *Free energy calculations*

In statistical mechanics free energies are quantities of particular importance. Helmoltz free energy A, Gibbs free energy G and the others in fact are, for the respective ensemble, the link between microscopic knowledge, represented by the partition function, and the thermodynamic quantities.

Usually we are not interested in the absolute value of free energy, rather in its differences between particular thermodynamic states; for instance, the difference in free energy between the initial and final state of a chemical reaction tells us if the reaction is likely to happen and at which rate. The difference in free energy between a state in which a molecule is in a polar phase and a state in which the molecule in in an apolar phase tells us which state the molecule will be likely to solvate at equilibrium.

POTENTIAL OF MEAN FORCE In other cases we are interested in the projection of the free energy along a certain path in phase space, usually a range in which some collective variable $s(\vec{r})$, that might be function of one or more coordinates of the phase space vector, can vary. A typical example is the free energy profile (FEP) of the distance between COMs of two molecules or nanoparticles in a particular solvent. The free energy surface along the chosen *reaction coordinate* s is called the *potential of mean force (PMF)*. For a canonical ensemble[1], the PMF can be expressed as

$$W(s) = W(s^*) - \frac{1}{\beta} \ln \frac{Q(s)}{Q(s^*)}$$

where $W(s^*)$ and $Q(s^*)$ are arbitrary reference constants and $Q(s)$ is

$$Q(s) = \frac{\int_\Theta \delta(s(\vec{r}) - s) \exp[-\beta \mathcal{H}(\vec{x})] \, d\vec{x}}{\int_\Theta \exp[-\beta \mathcal{H}(\vec{x})] \, d\vec{x}} \tag{2.2.1}$$

where Θ is the domain of $s(\vec{r})$. This can be seen as the partition function for the ensemble for a specific value of s.

1 For the isothermal-isobaric ensemble the generalization is straightforward.

Calculation of a PMF can be thought as a calculation of several free energy differences along the chosen path, as shown within the free energy perturbation theory:

$$\Delta A_{\mathcal{AB}} = -k_B T \ln[\left\langle \exp[-\frac{1}{k_B T}(U_{\mathcal{B}} - U_{\mathcal{A}})] \right\rangle_{\mathcal{A}}] \tag{2.2.2}$$

Here, $\Delta A_{\mathcal{AB}}$ is the difference of free energy between states $\mathcal{A}$ and $\mathcal{B}$, and $\langle \ldots \rangle_{\mathcal{A}}$ indicates an average taken with respect with the canonical configurational distribution of the state $\mathcal{A}$ [198].

This technique requires for the two states, between which the difference is computed, to not differ significantly, i.e. the configuration spaces of the two states should have significant overlap. It is apparent that for computing the PMF, one must rely on a method that can sample efficiently the region where $\delta(s(\vec{r}) - s)$ is finite; it is particularly important that each sampled configuration is close enough to the successive to enhance the superposition of states. MD in principle should be such a method, but the fact that the simulation can span only finite amounts of time leads to sampling problems.

Indeed, when for example the system should switch between two important minima separated by a free energy barrier, depending on the height of the barrier the simulation won't be long enough to show such an event occurring; if the barriers are much higher compared to the thermal energy $k_B T$, one can expect that the switching never happens within the simulation time. This is the so called problem of *rare events*, and in most cases it prevents pure MD methods to properly sample a PMF.

This task require the use of advanced sampling techniques, among which we will analyze the method called *umbrella sampling (US)*, the one used in this thesis, and the relative analysis algorithm *weighted histogram analysis method (WHAM)*.

2.2.2 *Umbrella sampling*

US method was developed by Torrie and Valleu [197, 196], and consists in imposing a *bias* to the system; this bias is an additional term in the PEF that ensures that the range of the reaction coordinate s is sampled in an efficient way. This can be done in a single simulation or, if the path of the reaction coordinate is more difficult to sample, in a series of simulations with different biases that we call *windows*.

For each window i, the PEF becomes

$$\mathcal{U}^b(\vec{r}) = \mathcal{U}^u(\vec{r}) + w_i(s)$$

where the superscripts b and u means biased and unbiased. To obtain the PMF $W(s)$ we must compute the unbiased $Q^u(s)$, while MD simulations of the biased system provide the biased one. It can be shown [91] that the first is related to the latter with this equation:

$$Q_i^u(s) = Q_i^b(s) \exp[\beta w_i(s)] \langle \exp[-\beta w_i(s)] \rangle \tag{2.2.3}$$

where

$$\langle \exp[-\beta w_i(s)] \rangle = \frac{\int \exp[-\beta \mathcal{U}^u(\vec{r})] \exp[-\beta w_i[s(\vec{r})]] \, d\vec{r}}{\int \exp[-\beta \mathcal{U}^u(\vec{r})] \, d\vec{r}} \tag{2.2.4}$$

It should be noted that the last term in the 2.2.3, being an integration over its whole range, doesn't depend on s, but only on the properties of the system. From this equation, the PMF is readily expressed as

$$W(s) = -\frac{1}{\beta} \ln Q_i^b(s) - w_i(s) + F_i$$

where $F_i = -1/\beta \ln \langle \exp[-\beta w_i(s)] \rangle$ is a constant. The only assumption made to obtain this result is that the sampling is sufficient, which means that for multiple window simulations the distributions $Q_i^b(s)$ must overlap significantly. Since the terms F_i depend on the bias potential, they are different for every window, thus it is necessary to evaluate them; however, as shown in equation (2.2.4), they depend on the unbiased distribution, so an additional method is required. Here we will describe the WHAM algorithm, which is the one implemented by GROMACSand used in this thesis.

WEIGHTED HISTOGRAM ANALYSIS METHOD The WHAM method is used to solve the problem of calculating the unbiased distribution function $Q^u(s)$ starting from a set of biased distribution functions $Q_i^b(s)$. The WHAM equation have been derived by minimizing the statistical error in the overlapping distribution functions [100]; the concept is to estimate the unbiased distribution as a weighted sum of the unbiased distributions related to each window i. Suppose having N_w simulation windows, the WHAM equations are expressed as:

$$Q^u(s) = \sum_{i=0}^{N_w} p_i Q_i^u(s), \qquad \sum_{i=0}^{N_w} p_i = 1 \tag{2.2.5}$$

Where the weights p_i are

$$p_i = \frac{n_i \exp[-\beta(w_i(s) - F_i)]}{\sum_{j=1}^{N_w} n_j \exp[-\beta(w_j(s) - F_j)]]} \tag{2.2.6}$$

where n_i is the number of snapshots taken in the window i. Using equation (2.2.3), we can write

$$Q^u(s) = \frac{\sum_{i=0}^{N_w} n_i Q_i^b(s)}{\sum_{j=1}^{N_w} n_j \exp[-\beta(w_j(s) - F_j)]]} \tag{2.2.7}$$

What we need more to solve this equation are the constant of free energy F_i, that can be expressed as

$$e^{-\beta F_i} = \int e^{-\beta w_i(s)} Q^u(s) \, ds \tag{2.2.8}$$

We can see that equations (2.2.7) and (2.2.8) compose a system with $Q^u(s)$ and F_i as solutions. In practice, these can be solved with an iteration procedure: an

initial guess for the N_w free energy constants F_i is used to get an estimate of $Q^u(s)$ through (2.2.7), which is then used to generate new estimates of the F_i constants through (2.2.8); this is repeated until convergence. This method can be extended to multidimensional PMFs.

In GROMACS, the WHAM algorithm is implemented in the *g_wham* program [77], which uses bootstrap analysis to estimate the error bars of the PMF profile.

BIAS POTENTIALS The bias potentials w_i must be chosen with the aim of sampling homogeneously all the range of the reaction coordinate, i.e. generating histograms which overlap enough in adjacent windows, in the case of a multi-window US. Thanks to its simplicity, an harmonic bias potential is the most used. This kind of bias potential, for instance when used to restrain the system in a region near to the value s_i of coordinate s, can be expressed as

$$w_i(s) = \frac{1}{2}K(s - s_i)^2$$

On must pay particular attention to the choice of K, the elastic constant of the potential. In fact, a low value of K can lead to large histograms, which allow better overlapping, but could not be strong enough to restrain the system close to the s_i configuration, especially in region across a free energy barrier; conversely, a large value of K leads to a more strict restraining, but makes necessary to fit more windows in the same region to obtain good overlap.

2.2.3 *Bilayer systems with applied tension*

Much part of the works presented in this thesis are focused on exploring the role of membrane curvature in NP-NP aggregation processes and liposome-liposome interactions. There are different approaches for the study of membrane curvature. For instance, liposomes are membrane models in which the bilayer presents a non-zero curvature, and can be used both in experiments and simulations. However, simulating liposomes requires a significant computational effort compared to flat membrane patches, since much of its volume is occupied by water.

In order to work around this issue, we often use the barostat algorithm in order to induce tension on the bilayer patches, which allow us to study states of curvature precluded to unbiased simulations.

BUCKLED MEMBRANE SIMULATIONS Using the anisotropic mode of the barostat algorithm in GROMACSallows us to apply different tensions on the different axis of the bilayer. Typically, a rectangular patch of membrane is solvated in a simulation box, and then a *buckling* simulation is run. In this run, the box is compressed along the axis parallel to the longer edge of the membrane, thus deforming the membrane in a sinusoidal profile. This method is used for instance to compute the bending modulus of the bilayer[46]. After the buckling simulation, the buckled membrane system can be used to simulate a controlled curved membrane environment, with an arbitrary degree of curvature and only one direction affected.

UNDULATIONS SUPPRESSION Another way of using the barostat in order to apply tension to the bilayer is opposite to the previous. In this case, we box is stretched isotropically on the plane of the membrane, which prevent the membrane to exhibit the typical thermal elastic fluctuations. In this way, we impose an artificial state of flatness on the bilayer, which can be used to uncouple fluctuation and curvature effects from the interaction between membrane inclusions (see 1.3.1).

Since this method perturb the lipid density compared to an unbiased systems however, it can create problems in evaluating other effects dependend on the lipids order and disposition. In order to avoid this problem, it is possible to suppress the membrane undulations in an alternative way, using the position restraints available with gromacs. The restraints, which are applied at topology level before the simulations starts, allow bond the selected beads to a fixed position provided in a reference configuration file. In the work presented in Chapter 3, we used this method to obtain a perfectly flat membrane, applying flat bottomed restraints on the polar headgroups of the lipids. This way, the beads are not freezed but have a limited region inside which they can oscillate.

2.3 FORCE FIELDS

Studying the behaviour of a physical system with classical MD simulations requires the knowledge of the interactions between its single components. *Empirical force field (FF)* provide this knowledge, and allow us to parameterize a broad variety of different systems, ranging from small set of atoms to complex solute–solvent systems.

The most important assumption we have to do in order to create a model of inter–particle interaction is the *Born-Oppenheimer approximation*. It consists in separating the motion of the atomic nuclei from that of the electrons: this is possible because the typical frequencies of the electrons motions are considerably higher than that of the nuclei; this approximation allows to include electron presence implicitly in the empirical potentials, without counting them as effective DOF of the systems. Because the physics of electrons is purely of quantum nature, the Born-Oppenheimer approximation is necessary in order to create a PEF that depends only on the position coordinates of the nuclei.

This assumption gives a first hint on the reason why we call these force field *empirical*: they are based on functional forms derived not directly from the fundamental laws of physics, but rather on a compromise between accurate reproduction of experimental behaviour and computational efficiency. For applications to biomolecular systems, there are two main classes of empirical FFs: *atomistic*, where the basic interacting particles represent single atoms, and *coarse-grained (CG)*, where particles represent groups of atoms or small chemical moieties.

PARAMETERIZATION Generally, in a FF all the particles interact with each other through a PEF of the same functional form. Hence, the FF is defined by a set of parameters that characterize the interaction between the particles, whether of the

same or different type. The typical form of a FF is in fact that of a set of libraries containing these parameters, beyond functional forms and other informations such as the masses or the charges of the various chemical species of the system of interest. The *parameterization* is the derivation of these parameters, and can be done in three ways:

- Fitting the parameters to let the system reproduce a set of experimental measures;
- Fitting the parameters on the basis of finer–level simulations;
- Calculating the parameters from basic interaction models.

TRANSFERABILITY From these considerations, it is clear that it is important for a FF to be apt to be applied to a larger variety of problems than that on which it was parameterized. This feature is referred as *transferability*. For instance, a FF parameterized to reproduce the diffusion of a compound in a solvent should also able to correctly predict other properties such as its solubility. This is the most challenging task in the development of an empirical model, and the outcome is determined by the choice of the right target properties used during the parameterization.

In this chapter we will describe the basic concepts that come into play during the parameterization of FFs, then we will describe atomistic and CG FFs focusing on the MARTINI CG FF, the one used for the work of this thesis. We will then describe the MARTINI membrane model, which is at the base of the results in chapters 4 and 5.

2.3.1 *Bonded interactions*

Most of the FFs used today are based on the distinction between inter and intra–molecular forces. The *bonded* interactions take place between atoms that are connected by one or more covalent bonds in a molecule; the *non bonded* interactions include dispersion and electrostatic forces between non-bonded atoms. Hence the PEF can be split in two contributes:

$$U(\vec{r}_1, \ldots, \vec{r}_N) = U_b(\vec{r}) + U_{nb}(\vec{r}) \tag{2.3.1}$$

where the b and nb notation means *bonded* and *non–bonded*.

The simplest form of this potential include only three kinds of terms, and can be expressed as

$$\begin{aligned} U_b(\vec{r}) = & \frac{1}{2} \sum_{\text{bonds}} k_i^b (l_i - l_{i0})^2 + \frac{1}{2} \sum_{\text{angles}} k_i^a (\theta_i - \theta_{i0})^2 + \\ & + \frac{1}{2} \sum_{\text{torsions}} V_n (1 + \cos(n\omega - \gamma)) \end{aligned} \tag{2.3.2}$$

Here the bond length and angle are modeled as harmonic oscillators, and the torsional terms as a series of cosines. Generally other terms can be included in the potential, but these basic terms are always included, in this or in a more complex form. These three terms are schematically represented in Figure (2.4), and will be discussed in more detail.

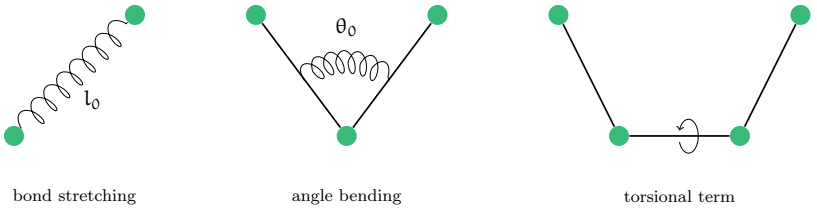

Figure 2.4: Schematic representation of the bonded interactions: bond stretching, angle bending and torsional terms

Bond stretching and angle bending

The typical potential energy curve for bond stretching has the shape as in figure 2.5, and can be modeled by the Morse potential:

$$v(l) = D_e\{1 - \exp[-a(l - l_0)]\}^2$$

Here, D_e is the depth of the potential energy minimum and $a = \omega\sqrt{\mu/2D_e}$, where μ is the reduced mass and ω is the frequency of the bond vibration. The frequency is given by the stretching constant of the bond, k, by $\omega = \sqrt{k/\mu}$. The Morse potential is not usually used because it requires three parameters to be defined; moreover it is usually not needed to have the bond interaction represented over the whole range that the Morse function covers. In fact, the Morse potential describes the bond even at the dissociation state, which is something that doesn't occur in most biological simulation, where the breaking of chemical bonds is not expected.

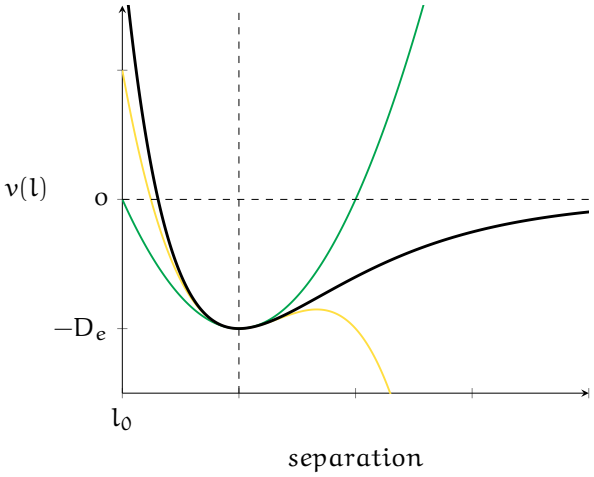

Figure 2.5: The thick black line is the plot of the Morse potential; the green line is the harmonic approximation; the gold line includes the cubic term. We see that when the cubic term is included, the range is enlarged, but the potential pass through a maximum that can lead to catastrophic lengthening of bonds, hence it' used only close to equilibrium.

Hence the typical solution is to use the Hooke's law formula with the square of the displacement from the reference bond length l_0:

$$v(l) = \frac{k}{2}(l - l_0)^2$$

This is the second order expansion of the Morse potential. The reference length is not necessarily that of equilibrium, since all the other terms in the force field contribute. When the accuracy of the quadratic term is not enough, higher terms can be included.

Similarly, the deviation of angles from their reference value is usually described by an harmonic potential, which in turn can be improved including higher order terms.

$$v(\theta) = \frac{k_a}{2}(\theta - \theta_0)^2$$

For each angle we have a reference value θ_0 and a force constant k_a. These constants are smaller in respect to that of bond stretching, since less energy is required to distort an angle than to stretch or compress a bond.

Torsional terms

Bond stretching and angle bending are regarded as 'hard' DOF, since relatively large energy is required to modify them significantly; on the other hand torsional motions allow most of the variability in molecular conformation (along with non-bonded interactions). Hence it is important that the force field properly represents the free energy profile of such rotations of bonds.

Torsional terms are implemented explicitly with contribution from each quartet of bonded particles A-B-C-D in the system, and are almost always expressed as a cosine expansion, for instance

$$v(\omega) = \frac{1}{2}\sum_{n=0}^{N} V_n(1 + \cos(n\omega - \gamma))$$

where ω is the torsional angle, γ indicates the position of the energy barriers and V_n their height. The factor n is the multiplicity, which value gives the number of the minimum points in the function as the bond is rotated from 0 to 2π. Most of the times, a single term in the series is enough to represent the bond rotation well. An example with different values of V and n is represented in figure 2.6.

The use of more terms in the torsional energy allows to obtain better accuracy, but has the drawback of requiring a large number of parameters even for a small range of molecule. Some FFs don't include in the bonded interaction the torsional energy, which is implemented as non-bonded interaction between the two atoms at the end of each torsion angle.

OUT OF PLANE BENDING MOTIONS For certain molecules, the angle bending potential is not enough to predict the accurate equilibrium conformation. For example, in cyclobutanone (C_4H_6O): the oxygen would remain out of the plane of

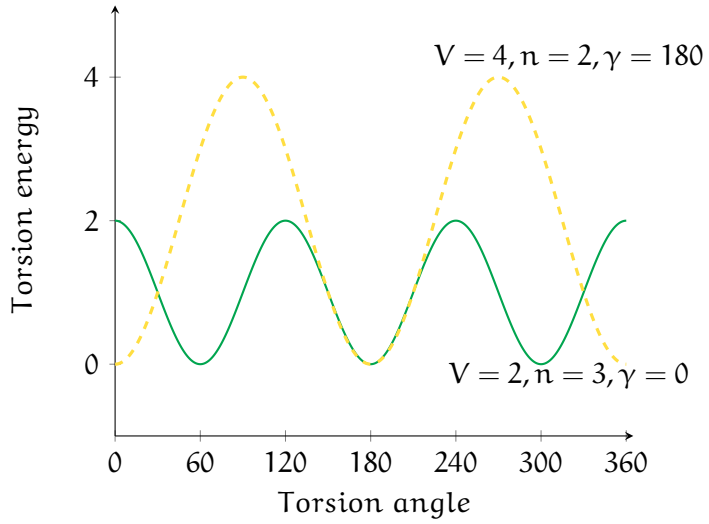

Figure 2.6: Two examples of single torsional terms.

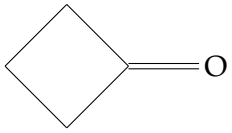

Figure 2.7: Structure of cyclobutanone.

the carbon ring, while experimentally it is shown that at equilibrium it lies on the plane, due to the nature of the π-bonding energy. To work out such situations it is necessary to incorporate one or more additional terms in the force field, involving the four atoms that form an *out of plane* bending term. This can be done with an *improper* torsion angle, i.e. a torsional potential that acts on the angle formed by the double bond and the plane.

2.3.2 *Non-bonded interactions*

Non-bonded interactions are important between pairs of independent particles, that may be atoms (or group of atoms) in different molecules, or in the same molecule but not directly bonded. These interactions don't depend upon a specific bonding relationship like a chemical bond, but act between all pairs of particles, depending only on distances and on the chemical nature of the particle. Indeed, usually they are modeled as a function of some inverse power of the distance, and they are distinguished in two groups: electrostatic interactions and Van der Waals (VdW) interactions.

The typical functional form used to model the non-bonded potential in force fields is

$$U_{nb}(\vec{r}) = \sum_{i=1}^{N} \sum_{j>i} \left(4\epsilon_{ij} \left(\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{6} \right) + \frac{q_i q_j}{4\pi\varepsilon_0 r_{ij}} \right) \tag{2.3.3}$$

The first term represents the energy contribution given by VdW interactions; it is modeled as a Lennard-Jones potential, and fully characterized by the constants σ_{ij}

and ϵ_{ij}, assigned to each pair of particle species. The last term is the electrostatic potential, given by Coulomb's law: here the parameters that characterize the interaction of two particles are their charges (see also section 2.3.2).

While all pairs of particles are subject to the non-bonded interactions, when considering intra-molecular interactions the energy contribution is computed only for pairs separated by enough bonds to be considered independent. In figure 2.8 a scheme of the non-bonded interactions.

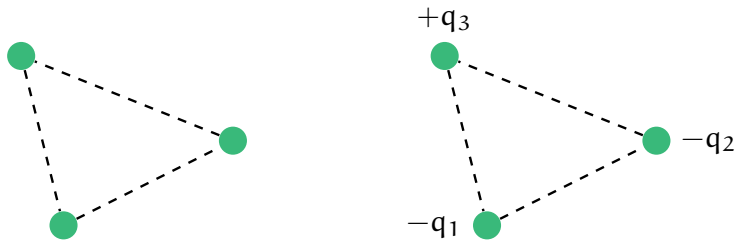

Figure 2.8: Schematic representation of the non-bonded interactions: dispersion forces (VdW) and electrostatic.

Treatment of non-bonded interactions

In MD simulations where PBC are applied, the effective number of molecules is infinite, because not only the particles in the simulation box have to be counted, but also their periodic images. This makes impossible to exactly calculate the forces that arise from non-bonded interactions, whose range, in principle, extends to infinite distances.

CUT-OFF METHOD Already mentioned in section 2.1.3, the cut-off method is the simplest approximation used to treat non-bonded interactions. The idea is to consider that each particle interacts only with particles within a distance r_c from itself, otherwise set to 0. The new functional form for the potential will be:

$$U^*(r) = \begin{cases} U(r) & r \leqslant r_c \\ 0 & r > r_c \end{cases}$$

SHIFT METHOD The cut-off method generates a discontinuity in the potential and in its first derivatives, the forces: this leads to bad energy conservation. To avoid this problem, it is possible to apply a *shift* to the potential, in order to make it touch 0 exactly at r_c; this can be made adding a constant $-U(r_c)$ for $r \leqslant r_c$, which doesn't affect the forces, or a term $-U(r_c) - \frac{dU(r)}{dr}\Big|_{r_c} (r - r_c)$, which cures also the discontinuity in the forces.