

Topics on Protein Dynamics

David Santo Orcero

irbis@orcero.org

<http://www.orcero.org/irbis>

April 16, 2004

©David Santo Orcero. All of this documentation is Copyright 2004 David Santo Orcero. Verbatim copying and redistribution of this entire documentation is permitted in any medium provided this notice, the link for the source of this document (<http://www.orcero.org/irbis/freesoft.html>) and the authority of the texts are preserved.

Resumen

On this technical report we are going to study some basic concepts about protein dynamics. We are going to overview the basic modeling concepts, that are needed to understand the modeling problem. Then, we will analyse thoughtly the nature of dynamic modeling of proteins. We will review some numerical methods for solving 0, and finnaly we will going to see what software can be found on the market to do molecular dynamics // Keywords: protein dynamics, molecular dynamics, numerical methods, protein modeling

Contents

1	Basic Modeling Concepts	5
1.1	Some basic definitions	5
1.2	System's limits	7
1.3	Taxonomy of systems	7
1.4	The evolved variable	8
1.5	Free variables vs. bounded variables	9
1.6	Dynamic modeling of proteins	10
2	Basic Mathematical techniques	12
2.1	The problem of time integration	12
2.2	Reformulation of the equations	14
2.3	The Verlet algorithm	15
2.4	Velocity Verlet	18
2.5	Leapfrog	19
2.6	Predictor-corrector methods	19
3	Motion modeling	21
3.1	Newton's Law	21
3.2	Lagrangian mechanics	24
3.3	Hamiltonian mechanics	26
3.4	Final remarks	29

4	Molecular dynamics	30
4.1	Introduction to molecular dynamics	30
4.2	Historical notes	32
4.3	Today's role of molecular dynamics	33
4.4	Limitations of molecular dynamics	36
4.4.1	On the limits of Schrödinger's equation	36
4.4.2	How <i>realistic</i> is a molecular dynamics simulation?	37
4.4.3	Time and size limitations	38
4.5	The basic algorithm	39
4.6	Controlling the simulation	41
4.7	Equilibration	42
5	Advanced topics on molecular dynamics	44
5.1	Potential truncation and long-range corrections	44
5.2	Periodic boundary conditions	45
5.3	The minimum image criterion	47
5.4	Geometries with surfaces	48
5.5	Size of the time-step	49
5.6	Molecular dynamics simulations in the aqueous phase	50
5.7	Molecular dynamics as an optimization tool	51
5.8	Brownian Dynamics Simulations	52
6	Analyzing molecular dynamics output	54
6.1	Simple statistical quantities to measure	54
6.2	Potential energy	55
6.3	Kinetic energy	56
6.4	Total energy	56
6.5	Temperature	57
6.6	The caloric curve	58
6.7	Mean square displacement	58

6.8	Pressure	59
6.9	Measuring the melting temperature	62
6.10	Real space correlations	63
6.11	Reciprocal space correlations	65
6.12	Other statistical ensembles	66
7	Final words on molecular dynamics	69

Chapter 1

Basic Modeling Concepts

Molecular mechanics involves some computational topics, that are necessary to understand and to be able to do that kind of simulations. First of all, we need to know some basic concepts about modeling. We are not going to see thoughtfully how to model a protein, but we must to know the basic concepts on modeling. Anyway, how the forces are calculated is out of the scope of this work.

More information about how to model a system can be found in [1], a good book to begin to work with modeling and simulation. Part of the book talks about the control of systems, but we can skip this part.

1.1 Some basic definitions

On modeling, the vocabulary looks somewhat common; but this insight is not really true, because it is used really accurately on the bibliography, and that's why we must know closely their true sense.

- A *system* is a group of related elements, for which we are interested on their global behaviour.
- *Entities* are the elements that form part of a system.

- *Attributes* are entities' properties. On a common system, they can be isomorph to a discrete set - $\aleph = 0$ -, as N ; isomorph to a continue set - $\aleph = 1$ -, as \mathfrak{R} or \mathfrak{S} ; or simply a finite set of possible values. More complex attributes value sets, $\aleph > 1$, as $\wp^n(\mathfrak{R})$, are too much complex to be modeled, and we should project it to a set with simpler order of complexity, with the consequent lost of precision. Anyway, that kind of systems are not common as real system, and only used to be found on metasystems.
- *State* of a particular system is the joint of entities and attributes of a system. The state of a system is able to full-define it. Anyway, we can only determine the future state of a system from the current state of a system on *deterministic* systems. On *schocastic* systems, we can not give exactly the next state of a system based on the information of the current state. Then, we use to give an *statistical approximation*. For that approximation, we define the state as a probabilistic distribution, and the evolution from the current state to the next state as a probability function.
- *Task* is any process that produces a change on the state of the system.
- *Reaction* is the effect of an entity over another entity that becomes a task. If the effect do not does work –on its most physical sense–, it doesn't matter, and it is not called “reaction”.
- *State function*: Defines the evolution of the system between two states. Can be a numeric function, a matrix or a predicate's operator.
- We define *modeling* as, given a system of the real world, the techniques destinated to obtain the equation that rules it.
- *Simulation* is, given a particular modeled system, to analise its behavior.
- *Computational Simulation* is, as it looks evident, using computer to that task.

1.2 System's limits

Limits on real system uses to be as important as the whole system. More than marginal conditions, its effect to the system uses to change dramatically its behaviour. The most important concepts about system limits are:

- *Environment*: All that is not part of our system. That means all that we do not want to study. The environment can be one system or more, and uses to be modeled as a simple black-boxed entity.
- *System Limits*: Separation between the environment and the system. It can be reactions through the system limits. As there is not completely known all the environment, there are reactions really difficult to model. For that modeling, we can get out from the environment some systems that dramatically affect to our system and analyze them as part of the system, or we can apply the limit's rule.

Limit's rule There are times where is not clear what part of the environment can be easily appended to the system to reduce the reactions thought the system limits. There we should add one abstract feedback relationship in all the entities with reactions with the environment, and one direct reaction cause-effect from the environment to the entity. We suppose that the effect of our system over the environment may be disregarded.

1.3 Taxonomy of systems

- *Open systems*: It exists non-disregarding effects with the environment.
- *Close systems*: The effect of the environment over the system might be disregarded for our precision.
- *Static systems*: The information is function only of a set of input variables. Note: a static system may drift till it arrive to its equilibrium point. Some systems may

stay always drifting, and they continue being static. This is common in static chaotic systems, and in overdetermined systems. They used to be modeled by algebraic equations. Usually, we will only be interested in the fixed points of the system.

- *Dynamic systems*: They have feedbacks on its state functions. That means that it has to be modeled by differential equations, because its dynamics uses to be critical. Usually, we will be strongly interested in the evolution of the system. Note that a dynamic system can not be evolving, and it continues of being a dynamic system. It is the case when it arrives to an equilibrium point. A dynamic system can be characterized as lots of static systems along the evolution of a variable.
- *Algebraic system*: Modeled by algebraic equations.
- *Concentrated parameters systems*: Modeled by differential equations. The most of the times they can be linearized over an equilibrium point, and that can be solved using Laplacian transform as algebraic equations.
- *Distributed parameters system*: Modeled by differential equations of partial derivatives.
- *Aheaded systems*: Some equations depend on the time at the future. It means that the effect is before at the time than the cause. There do not exist any aheaded system on physics, as far as I know.
- *Causal systems*: There is not any functional dependence of the future. It means that the cause is before the effect. The most of the common sense problems are causal.

1.4 The evolved variable

On dynamical systems, there exists frequently one variable that we want to study the system meanwhile that variable evolved. That variable used to be time, but in some

problems it might vary. We can find the following taxonomy:

- *Continuum evolved variable*: The evolved variable is isomorph to \mathfrak{R} . They can not be emulated on computers, and must be sampled.
- *Discrete evolved variable*: The evolved variable is isomorph to N .
- *Sampled continuum evolved variable*: The evolved variable is a sample to other continuum evolved variable. The sample period uses to be periodical, but can be non-periodical. The size of the period is strongly responsible of the precision of our model.
- *Event-based system*: The variable takes arbitrary values. Outside the scope of this values, the system is static. Then, the events ride the evolution of the system, and with no new event, there is not evolution on a stable system.

1.5 Free variables vs. bounded variables

Most of times, not all variables are free. When we want to model a system, we must to search a subset of the variables of the systems that are independent between between them. For example, on mechanics and study of motion, we can get mass, momentum, position and acceleration as the free variables of our system, but we can not get also velocity. It is not an efficiency problem; it is also a correction problem. Resolution with the problem with an overset of free variables will cause some really obfusate and weird errors of the simulation, significantly loosing the physical sense of the simulation.

We can define then the following concepts and recursive definitions:

- **Input of the problem**: Is the set of free variables.
- **Output of the problem**: the set of bounded variables.
- **Free variable**: a variable is free if, given a current partial set of free variables, the variable is not non-differential dependent of any combination of the partial

set of current variables. We should add our free variable to the current partial set of free variables.

- **Bounded variable:** a variable is bounded if, given a current partial set of free variable, exist an algebraic, non-differential equation that can be used to compute the variable based on the values of the current partial set of free variables.
- **State variables:** once we can define univocally the inner state of our entity only with the information of the current partial set of free variables, we can define that set as a set of free variables. There will not no more free variables on the system related to that set, and all the rest of variables will be bounded. We can then use that set to define our system. Then, each variable of the set is call a *state variable*.

Note that all the concepts are relative to what set of free variables we begin to choose. Real world problems uses to be very hard to analise in this way. It is not really clear what variables are free and bound. There exists statistical algorithms to detect it, but there are outside the scope of this work.

What set of free variables we choose to define the state variables and the inner state of an entity is an implementation decision, strongly dependent by what we want to do with our model exactly, and some considerations about performance of the resolution of differential equations and the error associated to that magnitudes that we want to be accurately.

1.6 Dynamic modeling of proteins

The dynamic modeling of proteins is a modeling and simulation problem with a system that have the following properties:

1. It is a *deterministic system* –as far as we do not use an “ab-initio” approach–.

2. It is an *open system*. The boundary of the problem, what is determined by the interaction between water molecules and biomolecules and or protein, is somewhat complex.
3. It is a *dynamic system*, as far as we want to model the dynamics of the protein. Static studies of proteins are outside the scope of this work.
4. It is a *causal system*, as far as the most of the problems on physics.
5. *The time as a continue variable* as far as we can not apply on a protein mechanics problem an event-driven approachment.

Computer has no continue variables. That means that computer dynamic modeling of proteins might carry an error by our time approximation –usually, because our time-sampling approach. In our work we are going to study that approximation, and how to minimize the error.

Depending of our Hamiltonian matrix, we will have a distributed or a concentrated parameter systems. Anyway, it is nearly impossible to solve our problem using Laplacian transform due to the complexity of the most common Hamiltonians.

Chapter 2

Basic Mathematical techniques

2.1 The problem of time integration

It is outside the scope of this work a deep theoretical study of the numerical resolution of differential equations that we need to solve a simulation, but we should overview some sparse topics to understand the process of simulation.

The engine of any simulation, including molecular dynamics, is its time integration algorithm, required to integrate the equation of motion of the interacting particles and follow their trajectory.

Time integration algorithms uses to be based on *finite difference methods*, where a variable –time, as it will be on our particular case– is sampled on a finite grid; being the *time step* Δt the distance between two consecutive points on the grid.

Known the positions and some of their time derivates at time t –the exact details depend on the type of algorithm, but uses to be at least first order and second order derivates–, the integration scheme gives the value of that variables at a later time $t + \Delta t$. By iterating the procedure, the time evolution of the system can be followed for long times, usually until the error –that growths with the time of the simulation– becomes so large that our simulation loses any physical sense.

We have to strongly consider that these schemes are approximate and there are

errors associated with them. In particular, one can distinguish between:

- **Truncation errors**, related to the accuracy of the finite difference method with respect to the true solution. Finite difference methods are usually based on a Taylor expansion truncated at some term, hence the name. These errors do not depend of the implementation: they are intrinsic to the algorithm.
- **Round-off errors**, related to errors associated to a particular implementation of the algorithm. For instance, to the finite number of digits used in computer arithmetics.
- **Errors on floating point semantics**. There are some bizarre errors, derived to errors on the semantics of floating points calculations. Its origin is the difference between common sense arithmetics and floating points arithmetics. As example, add and multiplication are not distributive at all at floating points arithmetics, and on the most of the cases are neither commutative nor distributive with the addition. The most common floating point representations, as IEEE 754, are normalized. This improves accurately, but it does that that representations lacks of zero support. IEEE 754 fix this problem using non-normalized numbers, but we will have signed zeroes –a positive zero and a negative zero–. A more deepest study of that kind of problems were made on a seminary of the biocrystalography group of the IBILCE by me at the beginning of 2000.

Sometimes, the errors can be reduced by decreasing Δt . For large Δt , truncation errors dominate, but they decrease quickly as Δt is decreased. As example, the Verlet algorithm that is going to be discussed in this chapter has a truncation error proportional to Δt^4 for each integration time step.

Round-off errors decrease more slowly with decreasing Δt , and dominate in the small Δt limit. This is specially interesting, because we find that with small Δt the integration algorithm is not the problem anymore, because the round-off error causes by floating point calculations is orders of magnitude greater. Using 64-bit precision

–corresponding to “double precision” when using Fortran on the majority of today’s workstations, or *double C* type–, helps to keep round-off errors at a minimum on medium-sized simulations.

Anyway, solution is not always a shorter Δt ; because below some point –that strongly depends on the order of the adds and products of Δt with other variables–, begin to increase dramatically the effect of the errors on floating point semantics. Products and adds of numbers of very different order of magnitude can originate easily this kind of error.

2.2 Reformulation of the equations

We are going to see some different techniques to integrate our equations. Some, as Verlet algorithm, are more molecular dynamics oriented, that’s why can be applied straightforward. But that specific molecular dynamics methods are not only the only ones that can be used. We have also some other techniques, more accurate, that can be used on modern computers.

The most of this techniques needs a reformulation of the problem. That reformulation goes by changing our set of differential equations as:

$$\frac{\partial y_1}{\partial t} = f(t, y_1, y_2, \dots, y_i, \dots, y_n) \tag{2.1}$$

$$\frac{\partial y_2}{\partial t} = f(t, y_1, y_2, \dots, y_i, \dots, y_n)$$

.....

$$\frac{\partial y_i}{\partial t} = f(t, y_1, y_2, \dots, y_i, \dots, y_n) \tag{*}$$

.....

$$\frac{\partial y_n}{\partial t} = f(t, y_1, y_2, \dots, y_i, \dots, y_n) \tag{2.1'}$$

Where \bar{y} is the set of state variables.

Note that not all the systems are formulated in that way. Some system may need a more carefully choose of the set of state variables. There are times when we must use virtual state variables, to eliminate second-order and third-order derivatives; and, finally, we would need to add some virtual restrictions to our system to redefine the equations. As far as I know, there is not a exact algorithm for convert all systems to that formulation. We use to analyse set of differential equations with polynomial formulation. If a partial element has no a polynomial participation on the equation, we should use Taylor series to linearize. Usually this is not a problem, but we ought to reinitialize regularly the equation, due to bias point variations produced along the time.

There is no matter in how is the dependency of non-differential terms. This affirmation includes variables, while there are not in a position of the equation affected by a differential operator.

Anyway, we do not need to worry about the complexities of this formulation. Taking as set of state variables the three spatial coordinates of acceleration, velocity and position, or acceleration, momentum and position, we will **always** be able to reformulate our problem easily in this way. And there are some algorithms specially adapted to molecular dynamics, as the Verlet algorithm [2, 3], Gear's predictor & corrector [4, 5] and their variants.

We can find more over integration algorithms on [6, 7], for a general survey, or [8] for details.

2.3 The Verlet algorithm

This well-known algorithm is a good compromise between performance and accurately in middle molecular dynamics simulations. That's why it became the most commonly used time integration algorithm. It was developed directly for molecular dynamics, and we will not have to adapt the Verlet algorithm to our system if our sistem is based

on the Newton's equations of motion.¹

The basic idea is to write two third-order Taylor expansions for the positions $r(t)$, one forward and one backward in time. This is the Verlet algorithm, as it was proposed on [2, 3].

Calling v the velocities, a the accelerations, and b the third derivatives of r with respect to t , we have:

$$r(t + \Delta t) = r(t) + v(t)\Delta t + \frac{1}{2}a(t)\Delta t^2 + (1/6)b(t)\Delta t^3 + O(\Delta t^4) \quad (2.2)$$

on the time-step $t + \Delta$ and

$$r(t - \Delta t) = r(t) - v(t)\Delta t + \frac{1}{2}a(t)\Delta t^2 - (1/6)b(t)\Delta t^3 + O(\Delta t^4) \quad (2.3)$$

on the time-step $t - \Delta$. Adding the two expressions gives

$$r(t + \Delta t) = 2r(t) - r(t - \Delta t) + a(t)\Delta t^2 + O(\Delta t^4) \quad (2.4)$$

This is the basic form of the Verlet algorithm.

Since we are integrating Newton's equations, $a(t)$ will be just the force divided by the mass, and the force is in turn a function of the positions $r(t)$:

¹"*ab-initio*" calculations may not obey Newton's equations of motion, but we can use the Car-Parrinello method, where ions are moved under Newton's laws of motion, but solving the electronic structure problem. This approximation is broadly used for little molecules, and can be found on [9]. In this case, we use precise "*ab-initio*" calculations for energy and Newton's laws of motion for the movement, and we can use the Verlet algorithm with a "*ab-initio*" model.

$$a(t) = -(1/m)\nabla V(r(t)) \quad (2.5)$$

As one can immediately see, the truncation error of the algorithm when evolving the system by Δt is of the order of Δt^4 , even if third derivatives do not appear explicitly. This algorithm is at the same time simple to implement, accurate and stable, explaining its large popularity among molecular dynamics simulators.

A problem with this version of the Verlet algorithm is that velocities are not directly generated. While they are not needed for the time evolution, their knowledge is sometimes necessary. Moreover, they are required to compute the kinetic energy \mathcal{K} , whose evaluation is necessary to test the conservation of the total energy $\mathcal{E} = \mathcal{K} + \mathcal{V}$. This is one of the most important tests to verify that a molecular dynamics simulation is proceeding correctly.

One could compute the velocities from the positions by using:

$$v(t) = \frac{r(t + \Delta t) - r(t - \Delta t)}{2\Delta t}. \quad (2.6)$$

However, the error associated to this expression is of order Δt^2 rather than Δt^4 , because we have not the solution on t , but on $t + \Delta t$ and $t - \Delta t$, and we make a linear interpolation between them. To overcome this difficulty, some variants of the Verlet algorithm have been developed. They give rise to exactly the same trajectory, and differ in what variables are stored in memory and at what times.

As we can see, each integration step needs $3n$ adds, $2n$ products and $3n$ memory positions.

2.4 Velocity Verlet

An even better implementation of the same basic algorithm is the so-called *velocity Verlet* scheme proposed by Andersen, Bales and Wilson in 1982; where positions, velocities and accelerations at time $t + \Delta t$ are obtained from the same quantities at time t in the following way:

$$r(t + \Delta t) = r(t) + v(t)\Delta t + \frac{1}{2}a(t)\Delta t^2 \quad (2.7)$$

$$v(t + \frac{\Delta t}{2}) = v(t) + \frac{1}{2}a(t)\Delta t \quad (2.8)$$

$$a(t + \Delta t) = -\frac{1}{m}\nabla V(r(t + \Delta t)) \quad (2.9)$$

$$v(t + \Delta t) = v(t + \frac{\Delta t}{2}) + \frac{1}{2}a(t + \Delta t)\Delta t \quad (2.10)$$

Note how we need $9n$ memory locations to save the $3n$ positions, velocities and accelerations, but we never need to have simultaneously stored the values at two different times for any one of these quantities.

2.5 Leapfrog

The *leap-frog* algorithm [10], is a variant of Verlet algorithm in which velocities are handled somewhat better, and is a little bit more accurate. The equations are:

$$r(t + \Delta t) = r(t) + \Delta t v(t + \frac{1}{2} \Delta t) \quad (2.11)$$

$$v(t + \frac{1}{2} \Delta t) = \Delta t a(t) + v(t - \frac{1}{2} \Delta t) \quad (2.12)$$

It is not needed to compute $v(t)$; but if we need it, we can use the expression:

$$v(t) = \frac{v(t + \frac{1}{2} \Delta t) + v(t - \frac{1}{2} \Delta t)}{2} \quad (2.13)$$

It needs $2n$ adds and $2n$ products, plus $2 + 2n$ positions of memory for each integration time step.

2.6 Predictor-corrector methods

Predictor-corrector algorithms [4] constitute another commonly used class of methods to integrate the equations of motion. Those more often used in molecular dynamics [5] are due to Gear, and consists of three steps:

- *Predictor*. From the positions and their time derivatives up to a certain order q , all known at time t , one “predicts” the same quantities at time $t + \Delta t$ by means of a Taylor expansion. Among these quantities are, of course, accelerations a .

- *Force evaluation.* The force is computed taking the gradient of the potential at the predicted positions. The resulting acceleration will be in general different from the “predicted acceleration”. The difference between the two constitutes an “error signal”.
- *Corrector.* This error signal is used to “correct” positions and their derivatives. All the corrections are proportional to the error signal, the coefficient of proportionality being a “magic number” determined to maximize the stability of the algorithm.

These algorithms are slightly simple; we must only use the empirical tables of magic numbers calculated by Gear in [5] for U.S. army. The tables can also be founded on the most of the molecular dynamics books. It also can be founded more details in [6, 7], and a detailed comparison between Verlet’s algorithm and Gear’s predictor–corrector scheme at [8].

Chapter 3

Motion modeling

3.1 Newton's Law

The Newton's law, as is most commonly used on motion modeling, is:

$$\vec{\mathcal{F}}_i = m_i \vec{a}_i \quad (3.1)$$

Where for each entity i , m_i is a scalar determined by the mass of the particle, \vec{a}_i is a vector that determines entity's acceleration and $\vec{\mathcal{F}}_i$ is a vector that defines the force applied on the entity. Its formulation on differential equations is:

$$\vec{\mathcal{F}}_i = m_i \frac{\partial^2 \vec{r}_i}{\partial t^2} \quad (3.2)$$

where \vec{r}_i is the position of the particle i .

Newton's equations of motion constitute a set of $3n$ coupled second order differential equations. In order to solve these, it is necessary to specify a set of appropriate initial conditions on the coordinates and their first time derivatives, . Then, the solution of

Newton's equations gives the complete set of coordinates and velocities for all time t .

As we saw on the chapter about mathematical techniques, the second order differential derivatives used to be very complex to solve. That's why it is easier to analyse Newton's law in order of momentum, as Newton originally did. Then, the resulting set of differential equations are:

$$\vec{\mathcal{F}}_i = \frac{\partial \vec{p}_i}{\partial t} \quad (3.3)$$

where \vec{p}_i is the momentum of a particle i . We can relate this differential equation with the position of the particle using the set of differential equations:

$$\vec{p}_i = m_i \frac{\partial \vec{r}_i}{\partial t} \quad (3.4)$$

The acceleration \vec{a}_i of each particle i can be computed from the forces using the Newton's second law algebraic equation:

$$\vec{a}_i = \frac{\vec{\mathcal{F}}_i}{m_i} \quad (3.5)$$

And the velocity \vec{v}_i of each particle i can be computed from each momentum using the algebraic equation:

$$\vec{v}_i = \frac{\vec{p}_i}{m_i} \quad (3.6)$$

In the absence of external fields or dissipative effects, the force $\vec{\mathcal{F}}$ on each degree of freedom j can be assumed to be derivable from a potential of interaction $\mathcal{U}(\vec{r}_1, \vec{r}_2, \dots, \vec{r}_n)$, then the force will be:

$$\vec{\mathcal{F}}_j = -\frac{\mathcal{U}(\vec{r}_1, \vec{r}_2, \dots, \vec{r}_n)}{\partial \vec{r}_j} \quad (3.7)$$

These equations of motion conserve the total energy $\mathcal{E}(\vec{r}_1, \vec{r}_2, \dots, \vec{r}_n)$ (kinetic energy $\mathcal{K}(\vec{r}_1, \vec{r}_2, \dots, \vec{r}_n)$ plus potential energy $\mathcal{U}(\vec{r}_1, \vec{r}_2, \dots, \vec{r}_n)$):

$$\mathcal{E}(\vec{r}_1, \vec{r}_2, \dots, \vec{r}_n) = \mathcal{K}(\vec{r}_1, \vec{r}_2, \dots, \vec{r}_n) + \mathcal{U}(\vec{r}_1, \vec{r}_2, \dots, \vec{r}_n) \quad (3.8)$$

Equation that **must be true every time** on the molecular dynamics. This equation is really important, not because its use on dynamics, but it is broadly used to check that our molecular dynamics simulation has physical sense. To do this, we must check that the principle of the conservation of energy is keep all the time. For knowing the kinetic energy $\mathcal{K}(\vec{r}_1, \vec{r}_2, \dots, \vec{r}_n)$ we can use the expression:

$$\mathcal{K}(\vec{r}_1, \vec{r}_2, \dots, \vec{r}_n) = \frac{1}{2} \sum_{i=1}^n \frac{(\vec{p}_i)^2}{m_i} \quad (3.9)$$

This formulation is the easier one to understand and to implement, and it is broadly used on computers. Anyway, there exists other two that can fit on some applications: Hamiltonian mechanics and Lagrangian mechanics.

3.2 Lagrangian mechanics

There is a later and more sophisticated formulation of classical mechanics due to Lagrange.

We define the kinetic and potential energies \mathcal{K} and \mathcal{U} in terms of the generalized coordinates of a system as

$$\mathcal{K} = \frac{1}{2} \sum_i m_i \frac{\partial q_i}{\partial t}^2 \quad (3.10)$$

$$\mathcal{U} = \mathcal{U}(q_i, \frac{\partial q_i}{\partial t}) \quad (3.11)$$

(we have allowed for the possibility of a velocity-dependent potential). We then introduce a new function called the *Lagrangian* \mathcal{L} , defined as

$$\mathcal{L} = \mathcal{K} - \mathcal{U} \quad (3.12)$$

The equations of motion of the system are then

$$\frac{\partial \mathcal{L}}{\partial q_i} - \frac{\partial}{\partial t} \left(\frac{\partial \mathcal{L}}{\partial (\frac{\partial q_i}{\partial t})} \right) \quad (3.13)$$

These equations are known as the *Euler-Lagrange* equations.

For example, suppose there is no velocity dependence in the potential. Then

$$\frac{\partial \mathcal{L}}{\partial q_i} = - \frac{\partial \mathcal{U}}{\partial q_i} \quad (3.14)$$

$$\frac{\partial \mathcal{L}}{\partial \frac{\partial q_i}{\partial t}} = m_i \frac{\partial q_i}{\partial t} \quad (3.15)$$

$$\frac{\partial}{\partial t} \left(m_i \frac{\partial q_i}{\partial t} \right) = - \frac{\partial \mathcal{U}}{\partial q_i} \quad (3.16)$$

Thus we return to Newton's Second Law. The quantities $\partial \mathcal{L} / \partial \frac{\partial q_i}{\partial t}$ can be identified as a momentum associated with the coordinate q_i . If the q_i are simply Cartesian coordinates, then they are the ordinary mechanical momenta, but in general we refer to them as *conjugate* or *canonical* momenta, which are defined as the derivative of the Lagrangian with respect to the rate of change of the generalized coordinates:

$$p_i = \frac{\partial \mathcal{L}}{\partial \frac{\partial q_i}{\partial t}} \quad (3.17)$$

The Euler-Lagrange equations can be derived from a variational principle, known as *Hamilton's Principle or the Principle of Least Action*—which is somewhat misnomer—. The action for a system between times t_1 and t_2 is defined as:

$$\mathcal{I} = \int_{t_1}^{t_2} dt \mathcal{L} \left(q_i, \frac{\partial q_i}{\partial t}, t \right) \quad (3.18)$$

The Principle of Least Action states that the action is an *extremum* for the path of the motion.

To see this, let us consider a small variation in the path, vanishing at t_1 and t_2 , sends $q_i(t) \rightarrow q'_a(t) = q_i(t) + \delta q_i(t)$. The action also changes, $\mathcal{I} \rightarrow \mathcal{I}' = \mathcal{I} + \delta\mathcal{I}$, where to first order in $\delta q_i(t)$:

$$\delta\mathcal{I} = \int_{t_1}^{t_2} dt \delta\mathcal{L}(q_i, \frac{\partial q_i}{\partial t}, t) = \int_{t_1}^{t_2} dt \left(\frac{\partial\mathcal{L}}{\partial q_i} \delta q_i + \frac{\partial\mathcal{L}}{\partial \frac{\partial q_i}{\partial t}} \delta \frac{\partial q_i}{\partial t} \right) \quad (3.19)$$

Integrating by parts we find:

$$\delta\mathcal{I} = \int_{t_1}^{t_2} dt \left(\frac{\partial\mathcal{L}}{\partial q_i} - \frac{\partial}{\partial t} \left(\frac{\partial\mathcal{L}}{\partial \frac{\partial q_i}{\partial t}} \right) \right) \delta q_i + \left[\frac{\partial\mathcal{L}}{\partial \frac{\partial q_i}{\partial t}} \delta q_i \right]_{t_1}^{t_2} \quad (3.20)$$

The second term vanishes, as we are keeping the end points of the path fixed, and so if the variation of the action $\delta\mathcal{I}$:

$$\frac{\partial\mathcal{L}}{\partial q_i} - \frac{\partial}{\partial t} \left(\frac{\partial\mathcal{L}}{\partial \dot{q}_i} \right) = 0 \quad (3.21)$$

which are precisely the Euler-Lagrange equations.

3.3 Hamiltonian mechanics

There is yet another formulation of classical mechanics due to Hamilton. It may seem excessive to have so many ways of doing the same thing, but each formulation has its own advantages: for example, the Lagrangian and Hamiltonian formulations are very powerful in bringing out conservation laws in dynamical systems, and relating them to symmetries.

Recall that associated with each coordinate q_i there is also a conjugate momentum p_i . We can therefore express the Euler-Lagrange equations as $\partial p_i / \partial t = \partial \mathcal{L} / \partial q_i$. We now define an important function called the *Hamiltonian* from the Lagrangian:

$$\mathcal{H}(p_i, q_i, t) = \sum_i p_i \frac{\partial q_i}{\partial t} - \mathcal{L}(q_i, \frac{\partial q_i}{\partial t}, t) \quad (3.22)$$

The idea is to replace $\frac{\partial q_i}{\partial t}$ by p_i . The first of Hamilton's equations follows from the Euler-Lagrange equations, after noting that $\partial \mathcal{L} / \partial q_i = -\partial \mathcal{H} / \partial q_i$. It is:

$$\frac{\partial p_i}{\partial t} = -\frac{\partial \mathcal{H}}{\partial q_i} \quad (3.23)$$

The second follows straightforwardly from the definition of the Hamiltonian upon partial differentiation with respect to p_i :

$$\frac{\partial q_i}{\partial t} = \frac{\partial \mathcal{H}}{\partial p_i} \quad (3.24)$$

If the Lagrangian does not depend explicitly on time, the Hamiltonian is a constant of the motion, as the following calculation shows. Recall that \mathcal{H} is a function of q_i , p_i , and possibly t only. Hence:

$$\frac{d\mathcal{H}}{dt} = \frac{\partial \mathcal{H}}{\partial t} + \sum_i \left(\frac{\partial \mathcal{H}}{\partial p_i} \frac{\partial p_i}{\partial t} + \frac{\partial \mathcal{H}}{\partial q_i} \frac{\partial q_i}{\partial t} \right) \quad (3.25)$$

Using Hamilton's equations we see that:

$$\frac{d\mathcal{H}}{dt} = \frac{\partial\mathcal{H}}{\partial t} = -\frac{\partial\mathcal{L}}{\partial t} \quad (3.26)$$

Thus if $\partial\mathcal{L}/\partial t = 0$, the Hamiltonian is constant. One can also show that:

$$\mathcal{H} = \mathcal{K} + \mathcal{U} \quad (3.27)$$

where \mathcal{K} and \mathcal{U} are the kinetic and potential energies. Thus the *Hamiltonian can be identified with the total energy of the system.*

As a trivial example, consider a set of n free particles with mass m . The Lagrangian is:

$$\mathcal{L} = \frac{1}{2} \sum_i m \frac{\partial q_i}{\partial t}^2 \quad (3.28)$$

from which we can easily derive the Euler-Lagrange equations:

$$0 - \frac{\partial}{\partial t} \left(m \frac{\partial q_i}{\partial t} \right) = 0 \quad (3.29)$$

or that the acceleration of each particle is zero. The conjugate momentum is:

$$p_i = m \frac{\partial q_i}{\partial t} \quad (3.30)$$

and hence the Hamiltonian becomes:

$$\mathcal{H} = \frac{1}{2} \sum_i \frac{p_i^2}{m} \quad (3.31)$$

This is of course the familiar non-relativistic expression for the kinetic energy. This will be perfect, because it is uncommon using relativistic effects for the study of molecular dynamics. It is unprovable that a protein move itself near the velocity of light c .

3.4 Final remarks

How the Hamiltonian \mathcal{H} is derived is outside the scope of this work. It can be used as bibliography [11], where there is found a more deep study about how the Hamiltonian \mathcal{H} can be calculated by some *ab-initio*, semiempirical and conformational ways.

Really, one can formulate first the Hamiltonian, and after this formulate the Lagrangian operator. That was I did on my M.D. dissertation. Because of this I preferred look for the other way to do the derivation.

On the mosts of the real problems, on biomolecular dynamics, using only Newton's formulation is good enough. For using some advance mathematical techniques, as higher-level Runge-Kutta, Adams-Bashforth or Adams-Moulton, mathematical formulation is perfect.

Lagrangian formulation has other additional problems with non-holonomic constraint, that would need a more complex study, or bounded state variables with holonomic constraints –something as $f(r_1, r_2, r_3, \dots, t) = 0$ -, that would need Lagrange multipliers to solve the problem.

Chapter 4

Molecular dynamics

4.1 Introduction to molecular dynamics

We define *molecular dynamics*, often appeared as *MD* on the literature, as a computer simulation technique where the time evolution of a set of interacting atoms is followed by integrating their equations of motion.

Therefore, in contrast with the Monte Carlo simulations, molecular dynamics is a deterministic technique if we use a deterministic dynamics: given an initial set of positions and velocities, the subsequent time evolution is completely determined. Monte Carlo is also a technique without any physical sense, where all the intermediate steps haven't any sense and can be depreciated. If we are interested in how to reach a position with physical sense using a trajectory with physical sense, molecular dynamics is our technique.

This problem has, for a system of n entities, a total of 6 free variables for entity, 3 positional variables and 3 according to the momentum. That means a total of $3 * n$ free variables, plus lots more of bounded variables that are need for the computation - $3 * n$ for acceleration, $3 * n$ for velocity, n for mass and so on-.

If we are interested only on the internal variation of position, as on the case of folding, the three positional coordinates of one of the entities can be depreciated. Some

systems with some constraints allow us to depreciate other state variables; anyway, we can always use the full set without any mistake, and if we depreciate a variable maybe something can get wrong.

Sometimes we are not interested on a particular trajectory; we may want to obtain only a set of configurations distributed according to some statistical distribution function, or an statistical ensemble of the dynamics. This is what prof. Dr. Jose Ruggerio does.

An example is the microcanonical ensemble, corresponding to a probability density in phase space where the total energy is a constant:

$$\delta(\mathcal{H}(\Gamma) - \mathcal{E}) \quad (4.1)$$

Where $H(\Gamma)$ is the Hamiltonian, and Γ represents the set of positions and momenta. δ is the well-known Dirac function, selecting out only those states which have a specific energy \mathcal{E} . Another example is the canonical ensemble, where the temperature T is constant and the probability density is the Boltzmann function:

$$\exp(-\mathcal{H}(\Gamma)/k_B T) \quad (4.2)$$

According to statistical physics, physical quantities are represented by averages over configurations distributed according to a certain statistical ensemble. A trajectory obtained by molecular dynamics provides such a set of configurations. Therefore, a measurements of a physical quantity by simulation is simply obtained as an arithmetic average of the various instantaneous values assumed by that quantity during the molecular dynamics run.

Statistical physics is the link between the microscopic behavior and thermodynam-

ics. In the limit of very long simulation times, one could expect the phase space to be fully sampled, and in that limit this averaging process would yield the thermodynamic properties. In practice, the runs are always of finite length, and one should exert caution to estimate when the sampling may be good, because the system is at equilibrium, or not.

In this way, molecular dynamics simulations can be used to measure thermodynamic properties and therefore evaluate, say, the phase diagram of a specific material.

Beyond this traditional use, molecular dynamics is nowadays also used for other purposes, such as studies of non-equilibrium processes, and as an efficient tool for optimization of structures overcoming local energy minima, in such cases as structure refinement on protein cristalography.

4.2 Historical notes

A full account of early developments of the molecular dynamics technique is certainly beyond the scope of this technical report. I will simply mention below a few key papers appeared in the 50s and in the 60s that can be regarded as milestones in molecular dynamics, and whose reading is certainly recommended. I will not mention developments in the Monte Carlo method, which also occurred during the same period of time. The interested reader can find a first-hand account by Wood in [12].

Reprints of all these papers, and of many other important works in the area of computer simulation of liquids and solids published up to 1986, can be found in [13].

The first paper reporting a molecular dynamics simulation was written by Alder and Wainwright [14] in 1957. The purpose of the paper was to investigate the phase diagram of a hard sphere system, and in particular the solid and liquid regions. In a hard sphere system, particles interact via instantaneous collisions, and travel as free particles between collisions. The calculations were performed on a UNIVAC and on an IBM 704.

The article *Dynamics of radiation damage* by J. B. Gibson, A. N. Goland, M.

Milgram and G. H. Vineyard from Brookhaven National Laboratory, appeared in 1960 [15], is probably the first example of a molecular dynamics calculation with a continuous potential based on a finite difference time integration method. The calculation for a 500-atoms system was performed on an IBM 704, and took about a minute per time step. The paper, dealing with creation of defects induced by radiation damage –a theme appropriate to cold war days–, is done exceedingly well, and is hard to believe that it is almost 40 years old.

Aneesur Rahman at Argonne National Laboratory has been a well known pioneer of molecular dynamics. In his famous 1964 paper *Correlations in the motion of atoms in liquid argon*, [16], he studies a number of properties of liquid Ar, using the Lennard-Jones potential on a system containing 864 atoms and a CDC 3600 computer. The legacy of Rahman’s computer codes is still carried by many molecular dynamics programs in operation around the world, descendant of Rahman’s.

Loup Verlet calculated in 1967 [2, 3] the phase diagram of argon using the Lennard-Jones potential, and computed correlation functions to test theories of the liquid state. The bookkeeping device which became known as Verlet neighbor list was introduced in these papers. Moreover the *Verlet time integration algorithm*, that we have seen on this work, was used for that work. Phase transitions in the same system were investigated by Hansen and Verlet a couple of years later [17].

4.3 Today’s role of molecular dynamics

Reviewing even a tiny subset of the applications of molecular dynamics is far beyond the purpose of this work. I will only briefly mention a few areas of current interest where molecular dynamics has brought and/or could bring important contributions. The list should not be considered as exhaustive, and certainly reflects my background in physics –mostly as part of my Computer Science degree, part of Civil Engineering, a Ms. degree about Si cluster and a little bit of contact with macromolecular biological structures– and the documentation that I could get on Internet for the development of

this work:

- **Liquids.** As shown on the historical review, liquids are where everything started! Nevertheless, they remain an important subject. Availability of new realistic interaction models allows to study new systems, elemental and multicomponent. Through non-equilibrium techniques, transport phenomena such as viscosity and heat flow have been investigated [18]. And we must remember that any protein, membrane or DNA that we simulate is in the water. Nowadays, the main research line of my Ms. Sc. advisor is in that field.
- **Defects.** Another subject whose investigation by molecular dynamics started a long time ago -see [13]-, defects in crystals –crucial for their mechanical properties and therefore of technological interest– remain a favoured topic. And with the new construction process that are used to develop integrated circuits with higher integration scale, it becomes more important day-to-day this application. The focus shifted perhaps from point defects –vacancies, interstitial– to linear –dislocations– and planar –grain boundaries, stacking faults– defects [19]. Again, improved realism thanks to better potentials constitutes a driving force.
- **Fracture.** Under mechanical action, solids break into two or more pieces. The fracture process can occur in different ways and with different speeds depending of several parameters. The technological importance is obvious, and simulation is providing useful insights on the fracture process. Cars industry, construction, and military industry value this application of molecular dynamics a lot. As a *trivia*, this is the main field where the grid of the differential finite method is not over the time. Here it used to be the grid over the force applied, as far as we want to see how the system evolves as the force become stronger. Thus, the system can enter on a inner state in which begin the deformations; if the force is increased more it breaks.

- **Surfaces.** Surface physics had a boom starting in the 80s, thanks to the availability of new wonderful experimental tools with microscopic resolution –scanning tunneling microscopy, high resolution electron microscopy, and several scattering-based techniques–. Simulation is still playing a big role in understanding phenomena such as surface reconstructions, surface melting, faceting, surface diffusion, roughening, etc, often requiring large samples and simulation times. Electronical properties of surfaces also became a key role on science, as newer integration technologies work on the surface of Si dice. Simulating surfaces on MOS transistors and MOS circuits is a key technology to build faster devices.
- **Friction.** Even more recent are investigations of adhesion and friction between two solids, propelled by the development of the atomic force microscope –AFM–. The body of “macroscopic” knowledge is being revised and expanded on microscopic grounds. We must remember that, at this time, nature of friction forces are not fully understood.
- **Clusters.** Clusters–conglomerates of a number of atoms ranging from a few to several thousands–constitute a bridge between molecular systems and solids, and exhibit challenging features. Frequently, an astonishingly large number of different configurations have very similar energies, making it difficult to determine stable structures. Their melting properties can also be significantly different from those of the solid, due to the finite size, the presence of surfaces and the anisotropy. Metal clusters are extremely important from the technological point of view, due to their role as catalysts in important chemical reactions –for instance, in catalytic exhaust pipes of cars–. On my M.D. [11] I worked on some challenging topics about this, because there were found a phase transition on middle sized clusters of Si, with interesting variations of electronic and mechanical properties.
- **Biomolecules.** Molecular dynamics allows to study the dynamics of large macromolecules, including biological systems such as proteins, nucleic acids –DNA,

RNA-, membranes [20, 21]. Dynamical events may play a key role in controlling processes which affect functional properties of the biomolecule. Drug design, with tools as *What if*, is commonly used in the pharmaceutical industry to test properties of a molecule at the computer without the need to synthesize it –which is far more expensive–. This approachment can be broadly used at our department, by all theoretical and experimental researchers that can use molecular dynamics to improve the understood of the biophysical phenomena.

- **Electronic properties and dynamics.** The development of the Car-Parrinello method, where the forces on atoms are obtained by solving the electronic structure problem instead of by an interatomic potential, allows to study electronic properties of materials fully including their dynamics –and, therefore, phase transitions and other temperature-dependent phenomena–. This important work gave rise to a very successful research line during the last decade.
- **Molecular dynamics as a optimization technique.** Molecular dynamics is broadly used as a optimization technique. On the refinement phase of protein cristalography, its contribution is invaluable. Better and faster algorithms might improve in the future the convergence rate of the optimizations.

4.4 Limitations of molecular dynamics

4.4.1 On the limits of Schrödinger's equation

The most of the abstract and the mathematical concepts that we studied on this work are applicable to all the model that we use; *ab-initio*, semiempirical, based on potentials...

Anyway, it looks somewhat strange that we can use Newton's law to move atoms, when everybody knows that systems at the atom level obey quantum laws rather than classical laws. Then, Schrödinger's equation is the one to be followed. But that equation is a complete nightmare.

Fortunately, we have a simple test of the validity of the classical approximation: the *de Broglie thermal wavelength*, defined as:

$$\Lambda = \sqrt{\frac{2\pi\hbar^2}{Mk_B T}} \quad (4.3)$$

where M is the atomic mass and T the temperature. The classical approximation is justified if:

$$\Lambda \ll a \quad (4.4)$$

where a is the mean nearest neighbor separation. If one considers for instance liquids at the triple point, Λ/a is of the order of 0.1 for light elements such as Li and Ar, decreasing further for heavier elements. The classical approximation is poor for very light systems such as H₂, He or Ne.

Moreover, quantum effects become important in any system when T is sufficiently low. The drop in the specific heat of crystals below the Debye temperature, or the anomalous behavior of the thermal expansion coefficient, are well known examples of measurable quantum effects in solids.

Molecular dynamics results should be interpreted with caution all these regions commented here, and we must think about using more accurately –and heavy– models.

4.4.2 How *realistic* is a molecular dynamics simulation?

In molecular dynamics, atoms interact with each other. These interactions originate forces which act upon atoms, and atoms move under the action of these instantaneous forces. As the atoms move, their relative positions change and forces change as well.

The essential ingredient containing the physics is therefore constituted by the forces. A simulation is realistic –that is, it mimics the behavior of the real system– only to the extent that interatomic forces are similar to those that real atoms (or, more exactly, nuclei) would experience when arranged in the same configuration.

As described on this work, forces are usually obtained as the gradient of a *potential energy function*, depending on the positions of the particles. The realism of the simulation therefore depends on the ability of the potential chosen to reproduce the behavior of the material under the conditions at which the simulation is run.

The problem of selecting–or constructing–potentials is addressed in more detail in my M.D. dissertation [11]; but may be resumed with the basic concepts:

- There is not a set of forces that can be used for all situations.
- More accurately forces uses to use boundary conditions that does less extrapolable its results.
- Finally, the propagation of errors and the theory of perturbation can do high-precision forces less accurately than lesser-ones.

4.4.3 Time and size limitations

Typical molecular dynamics simulations can be performed on systems containing thousands –or, perhaps, millions– of atoms, and for simulation times ranging from a few picoseconds to hundreds of nanoseconds. While these numbers are certainly respectable, it may happen to run into conditions where time and/or size limitations become important.

A simulation is “safe” from the point of view of its duration when the simulation time is much longer than the relaxation time of the quantities we are interested in. However, different properties have different relaxation times. In particular, systems tend to become slow in the proximity of phase transitions with variable sized integration step,

and it is not uncommon to find cases where the relaxation time of a physical property is orders of magnitude larger than times achievable by simulation.

A limited system size can also constitute a problem. In this case one has to compare the size of the molecular dynamics cell with the correlation lengths of the spatial correlation functions of interest. Again, correlation lengths may increase or even diverge in proximity of phase transitions, and the results are no longer reliable when they become comparable with the box length. It is the limits' problem, as it were formulated at the beginning of this work.

This problem can be partially alleviated by a method known as *finite size scaling*. This consist of computing a physical property A using several box with different sizes L , and then fitting the results on a relation

$$A(L) = A_o + \frac{c}{L^n} \quad (4.5)$$

A_o, c, n as fitting parameters.

A_o then corresponds to $\lim_{L \rightarrow \infty} A(L)$, and should therefore be taken as the most reliable estimate for the "true" physical quantity.

4.5 The basic algorithm

The basic algorithm of molecular dynamics is quite simple, once upon we have understood all the concepts that were explained on this work. The structure is:

1. Start with an appropriate initial conformation of the system - i.e. a properly energy-minimized structure. In my M.D. dissertation [11] there were reviewed some algorithms to do this.
2. Give each atom a push : randomly assign to each atom a velocity that is consistent with the Maxwell-Boltzmann distribution for the temperature of interest, usually

300K. The discussion about Maxwell-Boltzmann distribution is outside the scope of this work.

3. Calculate the forces on all atoms
4. Now that we know the position, velocity and force for each atom at time t , we predict the positions of the atoms at some new time
5. Use the new atomic positions to recalculate the forces.
6. Again, use the positions, velocities and forces to step ahead in time.
7. Repeat this process until either
 - your simulation has run long enough to tell you all you need to know –rare, but sometime it happens–.
 - you run out of disk space due to *thor3* abuse by somebody –more common–.
 - something happens and the machine misfunctioned –thanks, polo!–.
 - your 2 –M.D.– or 4 –D.Ph.– years of grant has elapsed.
 - yearly, on FAPESP grants.
 - your advisors wants results **now**.

This process of stepping ahead in time is often termed integrating the equations of motion. And this is what was exposed on this work.

Typically **we ought to analyze only the latter part of an MD simulation. The first part –maybe 10-20% of the total simulation time– is often discarded because the system is settling down**, adjusting to being at 300K.

This *equilibration phase* is then followed by the part of the simulation that we want to analyze: this is generally called the *production phase* of the simulation. This generally involves writing out snapshots of the system to file for later structural and energetic analysis.

4.6 Controlling the simulation

In a molecular dynamics simulation a system could be in a state characterized by a certain density, temperature, pressure –the calculation of these quantities is described later–: the phase diagram of the simulated material can be investigated. However, how can we bring the system in the desired region? In other words, how can we *control* the system?

In a standard calculation, the density is controlled by the choice of the box volume V . Programs usually have provisions for rescaling the volume upon request at the beginning of the calculation. Volume changes should be modest, of the order of at most a few percent. The pressure will be measured during the run, following the method described in this work. In other, more advanced simulation schemes, the user chooses the pressure, and the volume of the box is a variable to be measured.

Temperature changes are usually achieved by enabling a device in the code which brings the system to a desired temperature by rescaling the velocities. In the *velocity Verlet* algorithm discussed in this work, this may be accomplished by replacing the equation

$$v(t + \frac{\Delta t}{2}) = v(t) + \frac{1}{2}a(t)\Delta t \quad (4.6)$$

with

$$v(t + \frac{\Delta t}{2}) = \sqrt{\frac{T_0}{T(t)}}v(t) + \frac{1}{2}a(t)\Delta t \quad (4.7)$$

where T_0 is the desired temperature, and $T(t)$ the “instantaneous temperature”. Such a modification means that we are no longer following Newton’s equations, and the total energy is no longer conserved. Important data should not be collected in

this stage: these “controlled temperature” simulations should be used only to bring the system from one state to another. One should always wait for the system to *reach equilibrium* under clean constant energy conditions before collecting data.

4.7 Equilibration

Every time the state of the system changes, the system will be “out of equilibrium” for a while. We are referring here to thermodynamic equilibrium. By this, it is meant that the indicators of the system state –of which many are described on this work– are not stationary –that is, fluctuating around a fixed value–, but relaxing towards a new value –that is, fluctuating around a value which is slowly drifting with time–.

The state change may be induced by us or spontaneous. It is induced by us when we change a parameter of the simulation–such as the temperature, or the density– thereby perturbing the system, and then wait for a new equilibrium to be reached. It is spontaneous when, for instance, the system undergoes a phase transition, thereby moving from one equilibrium state to another.

In all cases, we usually want equilibrium to be reached before starting performing measurements on the system. A physical quantity A generally approaches its equilibrium value exponentially with time:

$$A(t) = A_o + C \exp(-t/\tau) \quad (4.8)$$

where $A(t)$ indicates here a physical quantities averaged over a short time to get rid of instantaneous fluctuations, but not of its long-term drift.

The relevant variable is here the *relaxation time* τ . We may have cases where τ is of the order of hundreds of time steps, allowing us to see $A(t)$ converge to A_o and make a direct measurement of the equilibrium value. In the opposite case, τ could be much larger than our overall simulation time scale; for instance, of the order of one second.

In this case, we do not see any relaxation occurring during the run, and molecular dynamics hardly seems a valid technique to use in such a situation. In intermediate situations, we may see the drift but we cannot wait long enough to observe convergence of $A(t)$ to A_0 . In these cases, one can often obtain an estimate for A_0 by applying the relaxation equation on the available data, even if the final point is still relatively far from it.

Chapter 5

Advanced topics on molecular dynamics

5.1 Potential truncation and long-range corrections

The most common potentials, as Lennard–Jones, have an infinite range. The problem of this is that we must analyze $n * (n - 1)$ pair–potentials each time step. It is not possible for large applications, and for this reason it is customary to establish a cutoff radius R_c and disregard the interactions between atoms separated by more than R_c . This results in simpler programs and enormous savings of computer resources, because the number of atomic pairs separated by a distance r grows as r^2 and becomes quickly huge.

A simple truncation of the potential creates a new problem though: whenever a particle pair “crosses” the cutoff distance, the energy makes a little jump. A large number of these events is likely to spoil energy conservation in a simulation. To avoid this problem, the potential is often shifted in order to vanish at the cutoff radius.

Physical quantities are of course affected by the potential truncation. The effects of truncating a full-ranged potential can be approximately estimated by treating the system as a uniform –constant density– continuum beyond R_c . For a bulk system

–periodic boundary conditions along each direction–, this usually amounts to a constant additive correction. For example, the potential tail –attractive– brings a small additional contribution to the cohesive energy, and to the total pressure. Truncation effects are not so easy to estimate for geometries with free surfaces, due to the lower symmetry, and can be rather large for quantities like surface energy.

Commonly used truncation radii for the Lennard-Jones potential are in the order of magnitude of 2.5σ and 3.2σ . Particularly, I used 3.5σ on my M.D. dissertation. It should be mentioned that these truncated Lennard–Jones models are so popular that they acquired a value on their own as reference models for generic two-body systems. In many cases, there is no interest in evaluating truncation corrections because the truncated model itself is the subject of the investigation.

Potentials for metals and semiconductors are usually designed from the start with a cutoff radius in mind, and go usually to zero at R_c together with their first two derivatives at least. Such potentials do not therefore contain true van der Waals forces. Since such forces are much weaker than those associated with metallic or covalent bonding, this is usually not a serious problem except for geometries with two or more separate bodies.

5.2 Periodic boundary conditions

What should we do at the boundaries of our simulated system? One possibility is doing nothing special: the system simply terminates, and atoms near the boundary would have less neighbors than atoms inside. In other words, the sample would be surrounded by surfaces.

Unless we really want to simulate a cluster of atoms, this situation is not realistic. No matter how large the simulated system is, its number of atoms n would be negligible compared with the number of atoms contained in a macroscopic piece of matter –of the order of 10^{23} atoms–, and the ratio between the number of surface atoms and the total number of atoms would be much larger than in reality, causing surface effects to

be much more important than what they should.

A solution to this problem is to use *periodic boundary conditions* (PBC). When using PBC, particles are enclosed in a box, and we can imagine that this box is replicated to infinity by rigid translation in all the three Cartesian directions, completely filling the space. In other words, if one of our particles is located at position r in the box, we assume that this particle really represents an infinite set of particles located at

$$r + \ell a + m b + n c, \quad (\ell, m, n = -\infty, \infty) \quad (5.1)$$

where ℓ, m, n are integer numbers, and I have indicated with a, b, c the vectors corresponding to the edges of the box. All these “image” particles move together, and in fact only one of them is represented in the computer program.

The key point is that now each particle i in the box should be thought as interacting not only with other particles j in the box, but also with their images in nearby boxes. That is, interactions can “go through” box boundaries. In fact, one can easily see that we have virtually eliminated surface effects from our system, and at the same time the position of the box boundaries has no effect –that is, a translation of the box with respect to the particles leaves the forces unchanged–.

Apparently, the number of interacting pairs increases enormously as an effect of PBC. In practice, this is not true because potentials usually have a short interaction range. The *minimum image criterion* discussed next simplifies things further, reducing to a minimum the level of additional complexity introduced in a program by the use of PBC.

The PBC is broadly used on Molecular Dynamics programmes for cristalography, because it is an approximation that fits very well with reality, due to the only one differences are the crystal imperfections.

Really, there are other ways for solving the boundary problem. Most of semiempirical potentials have the average of the influence of the boundaries interactions on the

empirical coefficients. And we always can do Brownian Dynamics with the particles that are outside the box; the difference is so little that lots of time there is no way to appreciate it.

5.3 The minimum image criterion

Let us suppose that we are using a potential with a finite range: when separated by a distance equal or larger than a cutoff distance R_c , two particles do not interact with each other. Let us also suppose that we are using a box whose size is larger than $2R_c$ along each Cartesian direction.

When these conditions are satisfied, it is obvious that *at most one* among all the pairs formed by a particle i in the box and the set of all the periodic images of another particle j will interact.

To demonstrate this, let us suppose that i interacts with two images j_1 and j_2 of j . Two images must be separated by the translation vector bringing one box into another, and whose length is at least $2R_c$ by hypothesis. In order to interact with both j_1 and j_2 , i should be within a distance R_c from each of them, which is impossible since they are separated by more than $2R_c$.

When we are in these conditions, we can safely use is the *minimum image criterion*: among all possible images of a particle j , select the closest, and throw away all the others. In fact only the closest is a candidate to interact; all the others certainly do not.

This operating conditions greatly simplify the set up of a molecular dynamics program, and are commonly used. Of course, one must always make sure that the box size is at least $2R_c$ along all the directions where PBCs are in effect.

5.4 Geometries with surfaces

The purpose of PBCs is to eliminate surface effects. However, we may also be interested in situations where we *want to have surfaces*. Obviously, all surface physics problems belong to this class!

For a surface simulation, the model usually adopted is that of the *slab*: a thick slice of material, delimited by two free surfaces. This is simply obtained by removing PBCs along one direction –usually taken to be z – while retaining them in the orthogonal plane. Therefore, a slab must be thought as replicated to infinity in the xy plane, but there is no replication along the slab normal z .

If the slab is thick enough, its inner part is expected to be quite similar to the bulk of the material. The two surfaces –at the top and at the bottom of the slab– can then be thought of as two decoupled, independent surfaces. With this assumption, the system behavior would be close to that of a single surface at the top of a semiinfinite system –a condition which would be closer to actual surface experiments but which is unfortunately impossible to realize in simulation–.

There are also circumstances where researchers find it preferable to “freeze” a few layers of material at one side of the slab –with the atoms constrained to sit at perfect bulk-like crystal positions– leaving only the other side free. This is done when it is believed that spurious effects induced by the frozen side into the “good” side of the slab are of smaller entity than the effects induced by another free surface at the same separation distance. This is to be preferred in those cases where massive perturbations occur at the surface, such as atomic rearrangements –surface reconstructions– or local melting. If a slab with a frozen side is chosen, care must be taken to freeze a number of layers corresponding to a thickness of at least R_c , to guarantee that no mobile atom would be able to “see” the surface “through” the fixed atoms.

We can also leave PBCs along only one direction only, thus obtaining a *wire* geometry with the wire axis along the PBC direction, or remove them altogether. This latter condition correspond to a *cluster* of atoms. Clusters are important systems and

simulation of clusters have been performed since the early days of molecular dynamics and Monte Carlo.

5.5 Size of the time-step

Size of the time-step is a very complex topic, but we will try to analyze it briefly and clearly.

We would like to use as large a time-step as possible so that we can explore more of the conformational space of the system. However, the equations we use to project forward in time assume that the force remains constant over the course of the time-step; that means that the error is strongly dependant of the size of the time step. Because of this, long time steps will be a good approximation for slow –low frequency– motions, but could be a bad approximation for high frequency motions where the atomic forces change rapidly. That means that an upper limit is placed on the time-step by the highest frequency motions in the system. In our situations these will be the stretching motions of C-H, O-H and N-H bonds, which means that the time-step is limited to about $1fs$.

Since bond stretches are of little interest, a constraint algorithm called *SHAKE* is almost always employed in molecular dynamics to 'freeze out' these motions. *SHAKE* is applied at the end of every time-step returning all bonds to hydrogens to their equilibrium values, so even if they fly off during the course of a particular time-step they come back to where they should be. Using *SHAKE* allows us to use a larger time-step –i.e. $2fs$ –.

So why not do the same thing with the next highest frequency motions –bond angles–? Because constraining bond angles to their equilibrium values would slow down conformational transitions. Then we can use integration algorithms with a variable size of time-step. This will means a good compromise between accurately and velocity.

5.6 Molecular dynamics simulations in the aqueous phase

Given the biological emphasis on our department, we are more usually interested in macromolecules in the aqueous phase.

Water is usually added to the system by taking a snapshot of a molecular dynamics simulation of water alone and overlaying the macromolecule onto it. Any water molecule that overlaps with an atom of the macromolecule is removed. We end up with a box of water with the macromolecule in the center. This is one of the ways, but there not model hidrofillic and hydrophobic effects, neither models Hydrogen bridge bonds.

Other solution is to model the entire box of water “as is”. Sowler, but more accurately. It depends of our protein which one model is better for us.

Note that adding water dramatically increases the number of atoms in a typical macromolecular system. A large molecular dynamics simulation of an enzyme in water may contain as many as 100,000 atoms. This means that with current computational power, molecular dynamics simulations are restricted to the 10ns range. This is long enough to see interesting motions within a protein –as in the movie–, but is nowhere near long enough to simulate the process of a protein folding. In the reality, water is the main guilty of the computational difficulties of the protein folding, due to the extreme importance of hidrofillic and hidrofobic effects on the folding of a protein.

It is not the only one problem. We also need some way of dealing with water molecules at the boundary of the system. If we do nothing, the water molecules will see only vacuum at the edges of the box and we will be simulating a droplet of water. The problem with doing this is that the effects of the boundary will propagate a long way into the droplet, so that water that is close to the macromolecule may be adversely affected.

The most common way of dealing with the boundary is to use Periodic Boundary Conditions. In this approach, our simulation box is replicated in all dimensions – e.g.

a cubic box has a total of 26 neighbour boxes. Molecules that leave one side of the box during the simulation, return to the box on the other side. This approachment were studied at this work. Other approachment is using Brownian motion to model the motion of the water that is farrest a cut distance.

5.7 Molecular dynamics as an optimization tool

Molecular dynamics may also have a role as on optimization tool. Let us suppose that a set of n particles has many possible equilibrium configurations. The energy of these configurations is in general different, and one of them will be the optimal one; all of them, however, correspond to *local minima* in the energy, and are each other separated by energy barriers. Such a situation occurs very commonly, for instance, in cluster physics.

Finding the optimal structure within an approach based on traditional minimization techniques –steepest descent method, conjugate gradient method, etc.– is tricky, as these methods do not normally overcome energy barriers and tend to fall into the nearest local minimum.

One therefore would have to try out several different starting points, corresponding to different “attraction valleys” in the energy landscape, and relax each of them to the bottom of the basin. The optimal structure would then be the one with the lowest energy, provided that we were sage enough to select it in the list of candidates to try.

As molecular dynamics follows the direction of the gradient of the field, it will find the local minima of the field. The original “kick off” of the Maxwell-Boltzmann distribution gives always a little random component to each atom, that is going always to keep and causes that the system do not get hanged on saddle points.

This method is an excellent exploiting method, but its exploring abilities are really short. Anyway, Temperature in a molecular dynamics calculation provides a way to “fly over the barriers”: states with energy E are visited with a probability $\exp(-E/k_B T)$. If T is sufficiently large, the system can “see” the simultaneous existence of many different

minima, spending more time in the deeper ones. By decreasing slowly T to 0, there is a good chance that the system will be able to pick up the best minimum and land into it. With this formulæ, we keep nearer of *simulated annealing* methods, but we are still far away from the global minimum.

Note that the field of our molecule can be a non-physical field. We can use any objective function to be optimized using this method as a analogy.

Molecular dynamics for optimization is broadly used for protein cristalography.

5.8 Brownian Dynamics Simulations

Some definitions:

- **Brownian dynamics** (BD) simulations are used to simulate the diffusion and association of molecules in solution.
- **Brownian motion** is the random movement of solute molecules in dilute solution that results from repeated collisions with solvent molecules.

The basic principle involved in Brownian dynamics simulations is similar to that involved in molecular dynamics simulations, but introduces a few new approximations that allow us to perform simulations on the microsecond timescale – we must remember that molecular dynamics of proteins is limited to around 10 nanoseconds–.

The technique has been used to calculate the association rates of enzymes with their substrates –i.e. acetylcholinesterase with its substrate acetylcholine–. For diffusion-limited enzymes, this association of the enzyme and substrate is the rate-limiting step of the reaction. The simulations allow us to understand how association rates are affected by mutations in the protein, and by the presence of dissolved ions such as Na^+ and Cl^- in the solution. Brownian dynamics simulations have also been used to simulate protein-protein association events and to simulate the channeling of substrates between the active sites of multi-enzyme complexes.

We could have used from the beginning molecular dynamics for all these work; after all, molecular dynamics is a technique used for simulating dynamic behaviour. In fact, the diffusional behaviour of single ions in solution is something that can be examined in conventional molecular dynamics simulations. Unfortunately, for larger molecules and especially for macromolecules such as proteins, molecular dynamics simulations including explicit solvent molecules are just too computationally demanding for this to be feasible, because it takes around 10ns for a large protein just to randomly rotate in solution. Maybe with the new parallel algorithms and hardware commodities this will gonna be possible; but at this time is a research area in Computers Science.

In Brownian Dynamics we remove the explicit solvent molecules and replace them with an implicit continuum solvent description. We also typically ignore internal motions of the molecules –remember that high frequency bond stretches and angles place an upper limit on the time-step.- by ignoring them, i.e. keeping the molecules rigid, we can use much larger time-steps.

Chapter 6

Analyzing molecular dynamics output

Once our molecular dynamics simulation run, we must analyse its results. Some important physical magnitudes can be obtained from its result. Let's see them.

6.1 Simple statistical quantities to measure

Measuring quantities in molecular dynamics usually means performing *time averages* of physical properties over the system trajectory. Physical properties are usually a function of the particle coordinates and velocities. So, for instance, one can define the instantaneous value of a generic physical property A at time t :

$$A(t) = f(r_1(t), \dots, r_N(t), v_1(t), \dots, v_N(t)) \quad (6.1)$$

and then obtain its average

$$\langle A \rangle = \frac{1}{N_T} \sum_{t=1}^{N_T} A(t) \quad (6.2)$$

where t is an index which runs over the time steps from 1 to the total number of

steps N_T .

There are two equivalent ways to do this in practice:

- $A(t)$ is calculated at each time step by the molecular dynamics program while running. The sum $\sum_t A(t)$ is also updated at each step. At the end of the run the average is immediately obtained by dividing by the number of steps. This is the preferred method when the quantity is simple to compute and/or particularly important. An example is the system temperature.
- Positions (and possibly velocities) are periodically dumped in a “trajectory file” while the program is running. A separate program, running after the simulation program, processes the trajectory and computes the desired quantities. This approach can be very demanding in terms of disk space: dumping positions and velocities using 64-bit precision takes 48 bytes per step and per particle. However, it is often used when the quantities to compute are complex, or combine different times as in dynamical correlations, or when they are dependent on other additional parameters that may be unknown when the molecular dynamics programme is running. In these cases, the simulation is run once, but the resulting trajectory can be processed over and over.

The most commonly measured physical properties are discussed in the following.

6.2 Potential energy

The average potential energy $\bar{\mathcal{V}}$ is obtained by averaging its instantaneous value, which is usually obtained straightforwardly at the same time as the force computation is made. For instance, in the case of two-body interactions

$$\mathcal{V}(t) = \sum_i \sum_{j < i} \phi(|r_i(t) - r_j(t)|) \quad (6.3)$$

Even if it is not strictly necessary to perform in the time integration –forces are all that is needed–, the knowledge of $\mathcal{V}(t)$ is required to verify energy conservation. This is an important check to do in any molecular dynamics simulation.

6.3 Kinetic energy

The instantaneous kinetic energy is of course given by

$$\mathcal{K}(t) = \frac{1}{2} \sum_i m_i [v_i(t)]^2 \quad (6.4)$$

and is therefore extremely easy to compute. We will call $\bar{\mathcal{K}}$ its average on the run.

6.4 Total energy

The total energy is defined as:

$$\mathcal{E}(t) = \mathcal{K}(t) + \mathcal{V}(t) \quad \forall t \quad (6.5)$$

and is a conserved quantity in Newtonian dynamics. However, it is common practice to compute it at each time step in order to check that it is indeed constant with time. In other words, during the run energy flows back and forth between kinetic and potential, causing $\mathcal{K}(t)$ and $\mathcal{V}(t)$ to fluctuate while their sum must remain fixed.

In practice there could be small fluctuations in the total energy, in a typical amount of, say, one part in 10^4 or less. These fluctuations are usually caused by errors in the time integration, and can be reduced in its magnitude by reducing the time step if it is considered excessive. [8] contains an in-depth analysis of total energy fluctuations using various time integration algorithms.

Slow *drifts* of the total energy are also sometimes observed in very long runs. Such drifts could also be originated by an excessive Δt . Drifts are more disturbing than fluctuations because the thermodynamic state of the system is also changing together with the energy, and therefore time averages over the run do not refer to a single thermodynamic state. If drifts over long runs tend to occur, they can be prevented, for instance by breaking the long run into smaller pieces and restoring the energy to the nominal value between one piece and the next. A common mechanism to adjust the energy is to modify the kinetic energy via rescaling of velocities.

A final word of caution: while we may be tempted to achieve “perfect” energy conservation by reducing the time step as much as desired, working with an excessively small time step may result in waste of computer time, or, what is worse, some strange calculation errors due to errors on floating point computations. A practical compromise would probably allow for small energy fluctuations and perhaps slow energy drifts, as a price to pay to work with a reasonably large Δt . We can find in [6, 7] a deeper study of this topics.

6.5 Temperature

The temperature T is directly related to the kinetic energy by the well-known equipartition formula, assigning an average kinetic energy $k_B T / 2$ per degree of freedom:

$$\bar{\mathcal{K}} = \frac{3}{2} n k_B \bar{T} \quad (6.6)$$

An estimate of the temperature is therefore directly obtained from the average kinetic energy $\bar{\mathcal{K}}$. For practical purposes, it is also common practice to define an “instantaneous temperature” $T(t)$, proportional to the instantaneous kinetic energy $\mathcal{K}(t)$ by the relation:

$$\mathcal{K}(t) = \frac{3}{2}nk_B T(t) \quad (6.7)$$

6.6 The caloric curve

Once the total energy \mathcal{E} –also called internal energy– and the temperature T are measured in different runs corresponding to different thermodynamic states, one can construct the *caloric curve* $\mathcal{E}(T)$. This is a useful tool to monitor the occurrence of phase transitions, which imply a jump of $\mathcal{E}(T)$ if first-order, or in a derivative of $\mathcal{E}(T)$ if second –or higher– order. This strategy, however, is not suitable for a reliable estimate of the melting temperature T_m .

The most common first-order transition is melting, easily observable by simulation. When the system abandons the crystalline structure and becomes a disordered liquid, we observe a jump of $\mathcal{E}(T)$, corresponding to the latent heat of fusion. This usually happens at a temperature somewhat higher than the true melting temperature T_m of the model, due to hysteresis effects associated with the necessity to wait for a seed of the liquid phase to appear by a spontaneous fluctuation. Only when a liquid seed containing a few atoms is formed, the liquid phase can start to grow at the expense of the solid one. On the typical molecular dynamics time scale, one has to “overshoot” and set the temperature 20-30% above T_m to see melting happening.

6.7 Mean square displacement

The mean square displacement of atoms in a simulation can be easily computed by its definition

$$\text{MSD} = \langle |r(t) - r(0)|^2 \rangle \quad (6.8)$$

where $\langle |r(t) - r(0)|^2 \rangle$ denotes here averaging over all the atoms –or all the atoms in a given subclass–. Care must be taken to *avoid* considering the “jumps” of particles to refold them into the box when using periodic boundary conditions as contributing to diffusion.

The mean square displacement contains information on the atomic diffusivity. If the system is solid, the mean square displacement saturates to a finite value, while if the system is liquid, mean square displacement grows linearly with time. In this case it is useful to characterize the system behavior in terms of the slope, which is the *diffusion coefficient* \mathcal{D} :

$$\mathcal{D} = \lim_{t \rightarrow \infty} \frac{1}{6t} \langle |r(t) - r(0)|^2 \rangle \quad (6.9)$$

The “6” in the above formula must be replaced with “4” in two-dimensional systems.

6.8 Pressure

The measurement of the pressure in a molecular dynamics simulation is based on the Clausius virial function

$$\mathcal{W}(r_1, \dots, r_n) = \sum_{i=1}^n r_i \cdot \mathcal{F}_i^{\text{TOT}} \quad (6.10)$$

where $\mathcal{F}_i^{\text{TOT}}$ is the total force acting on atom i . Its statistical average $\langle \mathcal{W} \rangle$ will be obtained, as usual, as an average over the molecular dynamics trajectory:

$$\langle \mathcal{W} \rangle = \lim_{t \rightarrow \infty} \frac{1}{t} \int_0^t d\tau \sum_{i=1}^n r_i(\tau) \cdot m_i \frac{\partial^2 r_i(\tau)}{\partial t^2} \quad (6.11)$$

where use has been made of Newton's law.

by integrating by parts:

$$\langle \mathcal{W} \rangle = - \lim_{t \rightarrow \infty} \frac{1}{t} \int_0^t d\tau \sum_{i=1}^n m_i \left| \frac{\partial r_i(\tau)}{\partial t} \right|^2 \quad (6.12)$$

This is twice the average kinetic energy, therefore by the equipartition law of statistical mechanics

$$\langle \mathcal{W} \rangle = -\mathfrak{D} \cdot n \cdot k_B \cdot T \quad (6.13)$$

where \mathfrak{D} is the dimensionality of the system (2 or 3), n the number of particles, k_B the Boltzmann constant.

Now, we may see the total force acting on a particle as composed of two contributions:

$$\mathcal{F}_i^{\text{TOT}} = \mathcal{F}_i + \mathcal{F}_i^{\text{EXT}} \quad (6.14)$$

where \mathcal{F}_i is the internal force arising from the interatomic interactions, and $\mathcal{F}_i^{\text{EXT}}$ is the external force exerted by the container's walls. If the particles are enclosed in a parallelepipedic container of sides L_x , L_y , L_z , volume $V = L_x \cdot L_y \cdot L_z$, and with the coordinates origin on one of its corners, the part $\langle \mathcal{W}^{\text{EXT}} \rangle$ due to the container can be evaluated as:

$$\langle \mathcal{W}^{\text{EXT}} \rangle = L_x \cdot (-\mathcal{P} \cdot L_y \cdot L_z) + L_y \cdot (-\mathcal{P} \cdot L_x \cdot L_z) + L_z \cdot (-\mathcal{P} \cdot L_x \cdot L_y) = -\mathcal{D} \cdot \mathcal{P} \cdot V \quad (6.15)$$

where $-\mathcal{P} \cdot L_y \cdot L_z$ is, for instance, the external force $\mathcal{F}_x^{\text{EXT}}$ applied by the yz wall along the x directions to particles located at $x = L_x$, and so on. This equation can then be written

$$\left\langle \sum_{i=1}^n r_i \cdot \mathcal{F}_i \right\rangle - \mathcal{D} \cdot \mathcal{P} \cdot V = -\mathcal{D} \cdot n \cdot k_B \cdot T \quad (6.16)$$

or

$$\mathcal{P} \cdot V = n \cdot k_B \cdot T + \frac{1}{\mathcal{D}} \left\langle \sum_{i=1}^n r_i \cdot \mathcal{F}_i \right\rangle \quad (6.17)$$

This important result is known as the *virial equation*. All the quantities except the pressure \mathcal{P} are easily accessible in a simulation, and therefore constitutes a way to measure \mathcal{P} . Note how it reduces to the well-known equation of state of the perfect gas if the particles are non-interacting.

In the case of pairwise interactions via a potential $\phi(r)$ that equation becomes

$$\mathcal{P} \cdot V = n \cdot k_B \cdot T - \frac{1}{\mathcal{D}} \left\langle \sum_i \sum_{j>i} r_{ij} \frac{d\phi}{dr} \Big|_{r_{ij}} \right\rangle \quad (6.18)$$

This expression has the additional advantage over the general one, to be naturally suited to be used when periodic boundary conditions are present: it is sufficient to take them into account in the definition of r_{ij} .

6.9 Measuring the melting temperature

The behavior of the mean square displacement as a function of time easily allows to discriminate between solid and liquid. One might be then tempted to locate the melting temperature T_m of the simulated substance by increasing the temperature of a crystalline system until diffusion appears, and the caloric curve exhibits a jump indicating absorption of latent heat.

While these are indeed indicators of a transition from solid to liquid, the temperature at which this happens in a molecular dynamics simulation is invariably higher than the melting temperature. In fact, the melting point is by definition the temperature at which the solid and the liquid phase coexist –they have the same free energy–. However, lacking a liquid seed from where the liquid could nucleate and grow, overheating above melting commonly occurs. In this region the system is in a thermodynamically metastable state, nevertheless it appears stable within the simulation time. An overheated bulk crystal breaks down when its *mechanical instability point* is reached. This point may correspond to the vanishing of one of the shear moduli of the material or to similar instabilities, and is typically larger than T_m by an amount of the order of 20-30%.

While mechanical instability is certainly useful to estimate roughly the location of T_m , more precise methods exist. One of them consists of setting up a large sample consisting approximately of 50% solid and 50% liquid. Such a sample could be initially constructed artificially and it will contain interface regions between solid and liquid. One then tries to establish an equilibrium state where solid and liquid can coexist. When this goal is achieved, the temperature of the state has to be T_m by definition.

To this end, the system is evolved microcanonically at an energy \mathcal{E} believed to be in proximity of the melting jump in the caloric curve. Let us call T_o the initial temperature assumed by the system. This temperature is an increasing but unknown function of the energy \mathcal{E} , and it will in general differ from T_m . Suppose that $T_o > T_m$. Then, the liquid is favored over the solid, and the solid-liquid interfaces present in the system

will start to move in order to increase the fraction of liquid at the expense of the solid. In other words, some material melts. As this happens, the corresponding latent heat of melting is absorbed by the system. Since the total energy is conserved, absorption of latent heat automatically implies a decrease of the kinetic energy. Therefore, the temperature goes down. The force driving the motion of the solid-liquid interface will also decrease, and as time goes on, the position of the interface and the temperature will exponentially reach an equilibrium state. The temperature will then be T_m .

If $T_0 < T_m$, the inverse reasoning applies: latent heat is released and converted into kinetic energy, and temperature again converges exponentially to T_m , this time from below.

The melting temperature depends on the pressure of the system. A pressure measurement on the final state is also required to characterize it thermodynamically. More often, the above procedure is carried out using constant pressure techniques. In this case, the volume of the box automatically changes as material melts or crystallizes, to accommodate the difference in density between the two phases at constant pressure.

6.10 Real space correlations

Real space correlation functions are typically of the form

$$\langle A(r)A(0) \rangle \quad (6.19)$$

and are straightforward to obtain by molecular dynamics: one has to compute the quantity of interest $A(r)$ starting from the atomic positions and velocities for several configurations, construct the correlation function for each configuration, and average over the available configurations.

The simplest example is the pair correlation function $g(r)$, which is essentially a density-density correlation. $g(r)$ is the probability to find a pair a distance r apart,

relative to what expected for a uniform random distribution of particles at the same density:

$$\rho \cdot g(r) = \frac{1}{n} \left\langle \sum_{i=1}^n \sum_{\substack{j=1 \\ (j \neq i)}}^n \delta(r - r_{ij}) \right\rangle \quad (6.20)$$

This function carries information on the structure of the system. For a crystal, it exhibits a sequence of peaks at positions corresponding to shells around a given atom. The positions and magnitude of the peaks are a “signature” of the crystal structure –fcc, hcp, bcc...– of the system. For a liquid, $g(r)$ exhibits its major peak close to the average atomic separation of neighboring atoms, and oscillates with less pronounced peaks at larger distances. The magnitude of the peaks decays exponentially with distance as $g(r)$ approaches 1. In all cases, $g(r)$ vanishes below a certain distance, where atomic repulsion is strong enough to prevent pairs of atoms from getting so close.

One quantity often computed from $g(r)$ is the average number of atoms located between r_1 and r_2 from a given atom,

$$n_{av} = \rho \int_{r_1}^{r_2} g(r) 4\pi r^2 dr \quad (6.21)$$

This allows to define coordination numbers also in situations where disorder is present.

The calculation of $g(r)$ is intrinsically a Θ^2 operation, and therefore it can slow down considerably an optimized molecular dynamics program. If the behavior at large r is not important, it might be convenient to define a cutoff distance, and use techniques borrowed from fast force calculations using short-ranged potentials to decrease the computational burden. It should also be noted that periodic boundary conditions impose a natural cutoff at $L/2$, where L is the minimum between the box sizes L_x , L_y , L_z in the three directions. For larger distance the results are spoiled by size effects.

6.11 Reciprocal space correlations

Structural informations can also be collected by working in reciprocal space. The most basic quantity is the space Fourier transform of the density $\rho(r) = \sum_i \delta(r - r_i)$:

$$\rho(k) = \sum_{i=1}^n e^{ik \cdot r_i} \quad (6.22)$$

This quantity, easily *and quickly* obtained for a given configuration, is the building block to construct the *static structure factor* $S(k)$:

$$S(k) = \frac{1}{n} \langle \rho(k) \rho(-k) \rangle \quad (6.23)$$

or

$$S(k) = \frac{1}{n} \left\langle \sum_{ij} e^{ik(r_i - r_j)} \right\rangle \quad (6.24)$$

or

$$S(k) = \frac{1}{n} \left\langle \sum_{ij} \cos(k \cdot r_{ij}) \right\rangle \quad (6.25)$$

This quantity is extremely useful for comparisons with results from scattering experiments. In liquids, which are isotropic, $S(k)$ depends only on the modulus of the wavevector, and is essentially a Fourier transform of the pair correlation function:

$$S(k) = 1 + \rho \cdot g(k) \quad (6.26)$$

The computation of $S(k)$ via 6.23, 6.24, or 6.25 is however much faster than that of $g(r)$.

By replacing $\rho(k)$ with some other generic observable $A(k)$, any other correlation function in reciprocal space can be computed.

It should be noted that, working with periodic boundary conditions, results could not depend on the choice of a particular particle image. That is, one must impose that—choosing as example the direction x —

$$e^{ik_x(x_i+L_x)} = e^{ik_x x_i} \quad (6.27)$$

where L_x is the length of the repeated box. This implies a restriction in the choice of the allowable wavevectors:

$$k_x = \frac{2\pi}{L_x} n_x \quad (6.28)$$

where n_x is an integer, and similarly along y and z . Therefore, the allowed wavevectors are quantized as a consequence of adopting periodic boundary conditions.

6.12 Other statistical ensembles

We have discussed so far the standard molecular dynamics scheme, based on the time integration of Newton's equations and leading to the conservation of the total energy. In the statistical mechanics parlance, these simulations are performed in the *micro-canonical ensemble*, or NVE ensemble: the number of particles, the volume and the energy are constant quantities.

The microcanonical average of a physical quantity A is obtained as a time average on the trajectory:

$$\langle A \rangle_{\text{NVE}} = \frac{1}{n_T} \sum_{t=1}^{n_T} A(\Gamma(t)) \quad (6.29)$$

where I denote with $\Gamma(t)$ the phase space coordinate of the system $-3n$ positions and $3n$ velocities.

There are other important alternatives to the NVE ensemble, that we will mention here only briefly. The basic idea is that of integrating other equations in place of Newton's equations, in such a way that sampling is performed in another statistical ensemble. Averages of physical quantities in the new ensemble will be again obtained as time averages.

A scheme for simulations in the isoenthalpic-isobaric ensemble (NPH) has been developed by Andersen [22]. Here, an additional degree of freedom representing the volume of the box has been introduced, and all the particle coordinates are given in units relative to the box. The volume V of the box becomes a dynamical variable, with a kinetic energy and a potential energy which just $\mathcal{P} \cdot V$, where \mathcal{P} is the external pressure. The enthalpy $H = \mathcal{E} + \mathcal{P} \cdot V$ is a conserved quantity.

Parrinello and Rahman [23, 24, 25] developed a variant where the shape of the box can vary as well as the volume. This is achieved by introducing 9 new degrees of freedom instead of 1: the components of the three vectors spanning the molecular dynamics box. Each of them is a new dynamical variable, evolving accordingly to equation of motion derived from an appropriate Lagrangian. This scheme allows to study structural phase transitions as a function of pressure, where for example the system abandons a certain crystal structure in favor of a more compact one.

Another very important ensemble is the canonical ensemble (NVT). In a method developed by Nosè and Hoover [26, 27], this is achieved by introducing a time-dependent frictional term, whose time evolution is driven by the imbalance between the instant-

neous kinetic energy and the average kinetic energy $(\frac{3n}{2}) \cdot k_B \cdot T$.

Chapter 7

Final words on molecular dynamics

Molecular dynamics, as the most of computer science applied to physics, has to be using having in consideration its own limitations. One century ago, the physics researcher had to construct a model of the system, usually in the form of a set of mathematical equations. The model was then validated by its ability to describe the system behavior in a few selected cases. Anyway, the model had to be simple enough to allow to compute a solution from its equations. Sometimes, it worked; but in the most of the cases, this implied a considerable amount of simplification in order to eliminate all the complexities invariably associated with real world problems, and make the problem hand-computable. For that reason, theoretical models could be easily tested only in a few simple “special circumstances”. So, for instance, in condensed matter physics a model for intermolecular forces in a specific material could be verified in a diatomic molecule, or in a perfect, infinite crystal. All of we remember that kind of approximations on Si studies; and things like the “bulk model”, without working with some important effects, as surface effects. Even with simple models, approximations were often required to carry out the calculation. Unfortunately, many physical problems of extreme interest –both academic and practical– fall outside the realm of these “special circumstances”. Among them, one could mention the physics and chemistry of defects, surfaces, clusters of atoms, organic molecules, involving a large amount of degrees of freedom; an accurate treatment of temperature effects, including phase transitions;

disordered systems in general, where symmetry is of no help to simplify the treatment; and so on. As a example, on the biological studies the most complete darkness covered all related with how the biomolecules worked on the reality.

The advent of computers—which started to be used in the 50s—altered the picture by inserting a new element right in between experiment and theory: the computer experiment. In a computer experiment, a *model* is still provided by physics theoretical people, but the calculations are carried out by the machine by following a “recipe” —the algorithm, implemented in a suitable programming language—. In this way, more complexity can be introduced and more realistic systems can be investigated, opening a road towards a better understanding of real experiments.

Needless to say, the development of computer experiments altered substantially the traditional relationship between theory and experiment. On one side, computer simulations increased the demand for accuracy of the models. For example, a molecular dynamics simulation allows to evaluate the melting temperature of a material, modeled by means of a certain interaction law, or we can see the effect of the active center of a proteins over its chemistry ligand. This is a difficult test for the theoretical model to pass —and a test which has not been available in the past. Therefore, simulation “brings to life” the models, disclosing critical areas and providing suggestions to improve them.

On the other side, simulation can often come very close to experimental conditions, to the extent that computer results can sometimes be compared directly with experimental results. When this happens, simulation becomes an extremely powerful tool not only to understand and interpret the experiments at the microscopic level, but also to study regions which are not accessible experimentally, or which would imply very expensive experiments, such as under extremely high pressure. We can do thousands and thousands of expensive experiments on a cheap way, and acquire new insights how life works.

The same model evaluation became a science itself. How we could obtain better outputs for the same model, how we can do deepest analysis faster, accurately, cheaper, is a new science of great interest. While physics began to be studied centuries ago,

this new science became on a very few time the cornerstone of the mosts of the areas of science.

It is wonderful as, using things that were discovered centuries ago, as Newton's Law, or the equations of motion, computer scientists can develop tools that discover and emulate the deepest secrets to nature to physics, chemistry and biology researchers, and can be used for them to explain the nature.

Now, the lasts advances of computer science, as parallelism, better numerical methods, better optimization methods, artificial intelligence, and the improvements of the "state-of-art" of algorithms in general, as examples, promise us a broad future of new exciting discoveries.

Bibliography

- [1] K. Ogata, *Modern Control Engineering*. Prentice Hall, 1990.
- [2] L. Verlet *Phys. Rev.*, no. 159, p. 98, 1967.
- [3] L. Verlet *Phys. Rev.*, no. 165, p. 201, 1967.
- [4] A. Rahman *Phys. Rev.*, no. 136, p. A405, 1964.
- [5] C. W. Gear, *Value Problems in Ordinary Differential Equations*. Prentice Hall, 1971. Appendix E.
- [6] M. P. Allen and D. J. Tildesley, *Computer simulation of liquids*. Oxford press, 1987.
- [7] J. M. Haile, *Molecular dynamics simulation*. Wiley, 1992.
- [8] H. J. C. Berendsen, *Molecular Dynamics Simulation of Statistical-Mechanical Systems*, p. 43. North-Holland, 1986. G. Ciccotti and W. G. Hoover (Editors).
- [9] R. Car and M. Parrinello *Phys. Rev. Lett.*, no. 55, p. 2471, 1985.
- [10] R. P. Feynman, R. B. Leighton, and M. Sands, *The Feynman Lectures on Physics*, vol. 1, ch. 9. Addison-Wesley, 1963.
- [11] D. S. Orcero, "Parallel simulation and optimization of si clusters," Master's thesis, Escuela Técnica Superior de Ingenieros en Informática, 1999. Advisor: Prof. Dr. Sc. (H. C.) Carlos Renato Zacharias (UNESP/Brazil); coadvisor: Prof. Dr. Ernesto Pimentel (UMA/Spain).

- [12] W. W. Wood, *Molecular Dynamics Simulation of Statistical-Mechanical Systems*, p. 3. North-Holland, 1986. G. Ciccotti and W. G. Hoover (Editors).
- [13] G. Ciccotti, D. Frenkel, and I. R. McDonald, eds., *Simulation of Liquids and Solids*. North-Holland, 1986.
- [14] B. J. Alder and T. E. Wainwright *J. Chem. Phys.*, no. 27, p. 1208, 1957.
- [15] J. B. Gibson, A. N. Goland, M. Milgram, and G. H. Vineyard *Phys. Rev.*, no. 120, p. 1229, 1960.
- [16] A. Rahman *Phys. Rev.*, no. 136, p. A405, 1964.
- [17] J.-P. Hansen and L. Verlet *Phys. Rev.*, no. 184, p. 151, 1969.
- [18] G. Ciccotti and W. G. Hoover, eds., *Molecular Dynamics Simulation of Statistical-Mechanical Systems*. North-Holland, 1986.
- [19] S. Yip, *Molecular Dynamics Simulation of Statistical-Mechanical Systems*, p. 523. North-Holland, 1986. G. Ciccotti and W. G. Hoover (Editors).
- [20] H. J. C. Berendsen, *Molecular Dynamics Simulation of Statistical-Mechanical Systems*, p. 496. North-Holland, 1986. G. Ciccotti and W. G. Hoover (Editors).
- [21] H. J. C. Berendsen *Comp. Phys. Commun.*, no. 44, p. 233, 1987.
- [22] H. C. Andersen *J. Chem. Phys.*, no. 72, p. 2384, 1980.
- [23] M. Parrinello and A. Rahman *Phys. Rev. Lett.*, no. 45, p. 1196, 1980.
- [24] M. Parrinello and A. Rahman *J. Appl. Phys.*, no. 52, p. 7158, 1981.
- [25] M. Parrinello, *Molecular Dynamics Simulation of Statistical-Mechanical Systems*, p. 204. North-Holland, 1986. G. Ciccotti and W. G. Hoover (Editors).
- [26] S. Nose *Molec. Phys.*, no. 52, p. 255, 1984.
- [27] W. G. Hoover *Phys. Rev. A*, no. 31, p. 1695, 1985.