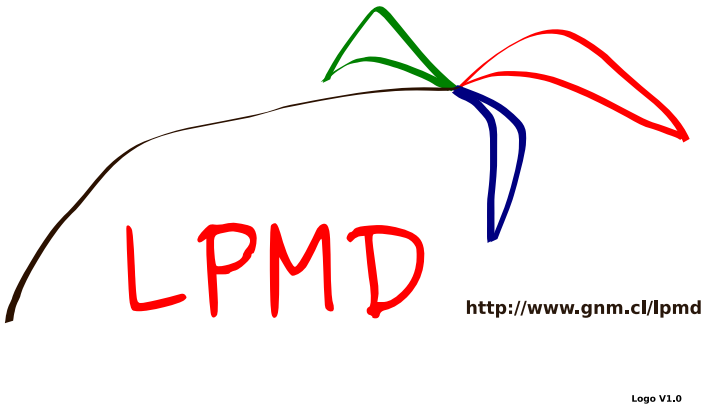
“Las Palmeras” Molecular Dynamics - LPMD.



December 11, 2013

Acknowledgements

We thank the "Proyecto Anillo ACT/24, Chile" for the constant financial support has provided our team, which has made possible the peaceful realization of our projects in the short and long term, because with it we could get top-quality equipment carrying out our tasks. We also thank the doctors in Physics, our mentors and friends, Gonzalo Gutirrez, Eduardo Menendez and Carlos Esparza, for all the advice and suggestions they made and continue to make to our project, and that his desire to carry out this program was a the main motivations for moving forward and constantly improve it.

This work has been realized, thank to the support of
Proyecto Anillo ACT/24

Contenidos

1. The LPMD code

1.1. Founds.

The `lpmd` code, start his developed during June in the 2007. This born like an idea of implement molecular dynamics in a modular way, trying to be user friendly in the plugins and upgrade developing. the princpal developers during the start of the has been: Sergio Davis, Claudia Loyola, Joaquín Peralta and Felipe González, they are all from the *Grupo de NanoMateriales* (`http://www.gnm.cl`) group. At this moment more colaborators are join to us[1].

The stable versions available of the software are:

- Version 0.6.2 : Released November 1st 2011.
- Version 0.6.1 : Released January 3th 2010.
- Version 0.6.0 : Released June 12th 2009.
- Version 0.5.4 : Released January 1st 29th 2009.
- Version 0.5.3 : Released November 1st 2008.
- Version 0.5.2 : Released August 5th 2008.
- Version 0.5.1 : Released April 30th 2008.
- Version 0.5.0 : Released April 4th 2008.
- Version 0.4.0 : Released July 2007.

The upgrade system is controled by developers. From the 0.5 version of `lpmd` the software have three principal packages.

- liblpmd
  Principal programming **API** developed by the team. This is the core of all characteristics that have `lpmd`. For a more detailed description about this take a look in the apendix section B.
- lpmd-plugins
  Is a plugins set that has been developed, and incorporate characteristics from the **API**, usually correspo-nend to Interatomic Potentials, integrators, analyzers, or file managers.
- lpmd
  `lpmd` Is the molecular dynamics (MD) software that use the plugins and the API in order to realize the simulations. This include additional tools like **lpmd-analyzer**, **lpmd-converter**, **lpmd-visualizer** and in the last version **lpmd-plotter** that are used for the generation of structures, analysis, visualization and image generation of atomic configurations.

1.2. Principal Goal

The principal scheme of `lpmd`, is the use of a control file (`file.control`), file that is load directly from the executable `lpmd` or other tools.

Consider by an example, a system of MD in that all options, like initial temperature, simulation setp number, integrator, interatomic potential, atomic positions, etc. are directly specified in a file "`simulation.control`", so `lpmd` could be executed directly using,

```
lpmd simulation.control > simulation.output
```

In this way, `lpmd` load all the neccesary information located in the file `simulation.control` and all output information is saved in the file `simulation.output`. Is important note that `lpmd` even can generate additional output file as modules or plugins had been loaded. From the version 0.6.1 during the `lpmd` execution, you can see what plugins has been loaded.

[1] Ver cap.8

1.3. Characteristics

Other characterisitc of `lpmd` is that this have additional flags for the control of the internal variables of a control file. By example, you can run many different systems with different temperatures using a single control file. Take the next line inside of a control file like as example

```
...
prepare temperature t=$(TEMP)
...
```

with this, we can run `lpmd` in order to assign to `TEMP` variable a real value, given directly by the command line, for example :

```
lpmd -o TEMP=500 simulation.control > simulation.output
```

In this way the program start the simulation with a real temperature sistem of 500K.

The advantage of this characteristic is that we can reduce the difficult of generate scripts for different simulations or even consecutive simulations of molecular dynamics. By example with only one command we can run a set of simulations with different values of a specific variable in a control file. For a set of different temperatures, for example:

```
                                for i in 300 400 500 ;
do lpmd -o TEMP=$i simulacion.control > simulacion-$i.output ; done
```

This realize three simulations with different initial temperatures with just a single control file.

Other interesting flag in `lpmd` and their utilities is `-p` that give us information about the plugins that you have instaled in your system and that are identified directly by `lpmd`. You can check this using, for example:

```
lpmd -p angdist
```

in this case `angdist` is a plugin that calculate the angular distribution function of a simulation cell. For more option of the executable try the `-h` option or `man lpmd`. The next correspond to the standard help information of lpmd using the `-h` flag.

```
username@machine:~$lpmd -h
.....
================================================================================
==================== LPMD VERSION 0.X.Y ========================================
================================================================================
LPMD version 0.X.Y
Using liblpmd version M.N

Usage: lpmd [--verbose | -v ] [--lengths | -L <a,b,c>] [--angles | -A <alpha,beta,gamma>]
      [--vector | -V <ax,ay,az,bx,by,bz,cx,cy,cz>] [--scale | -S <value>] [--option | -O
      <option=value,option=value,...>] [--input | -i plugin:opt1,opt2,...]
      [--output | -o plugin:opt1,opt2,...] [--use | -u plugin:opt1,opt2,...]
      [--cellmanager | -c plugin:opt1,opt2,...] [--replace-cell | -r] [file.control]
       lpmd [--pluginhelp | -p <pluginname>]
       lpmd [--help | -h]
username@machine:~$
```

In the next chapters of this manual we will give you a more detailed description of other characteristics of `lpmd` and their utilities.

2. Install

`lpmd` had been probed in different architechtures and compilers. However in the branch 0.6 and later the developers are only tested the code using two principal architechtures :

- Linux I686/AMD64 (Debian based architectures principally)
- OS/X

On the other hand `lpmd` have the `lpmd-installer` program, focused in the installation of `lpmd` made simple over different machines, wroted in python. We strongly recommend that beginners users use the `lpmd-installer` program to install `lpmd` in the computer.

The basic requirements in order to install lpmd in a single computer are :

- C++ compiler.(recomended 4.2 or later for GNU and 10.1 or later for intel)
- zlib libraries. (recomended the more updated version)
- Python 2.7 or later. (suggested Python 3.0 for branch 0.7)
- If you want install lpvisual plugin and/or use lpmd-plotter, we suggest:
  - OpenGL
  - imagemagick
  - mencoder

2.1. lpmd-installer

Is an program focused in a friendly installation of `lpmd`, this program was wroted in Python and is the manager in install in an automatic way the three principal packages that are part of the lpmd program. These programs are :

- liblpmd : Principal API of the project.
- plugins : Plugins set for MD and Structure analysis.
- lpmd : Principal program with extra utilities.

> If you don't like use the automatic installation process with `lpmd-installer` go to the section 2.2.

2.1.1. Instaling the lpmd-installer program

Because `lpmd-installer` is a python script, the installation of this script is a very simple work, in order to have a correct functionability of `lpmd-installer` you will need :

- Python 2.7 or later.
- Internet connection.

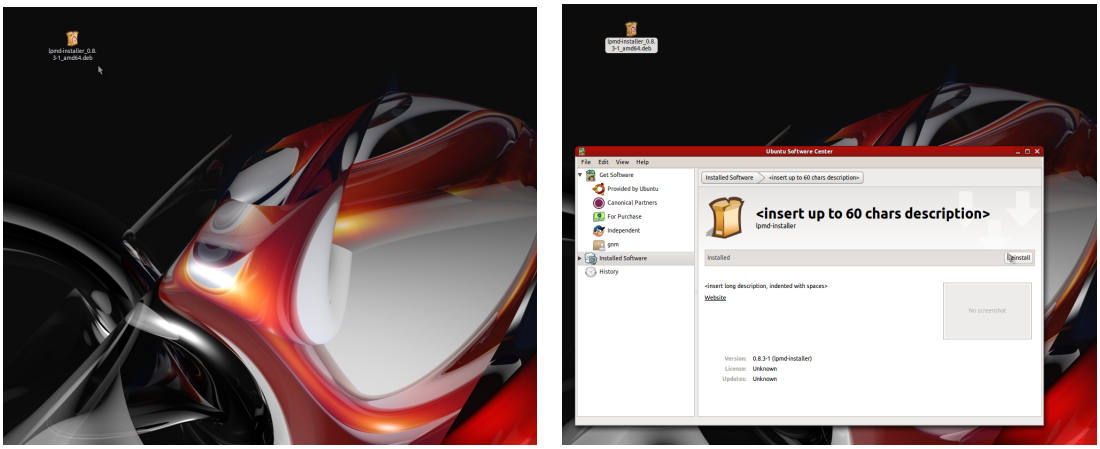
In order to install `lpmd-installer` in the system you can use two different ways. The first one is for distributions based in Debian (like Ubuntu), for this case you can choose download the deb file only or add the GNM repository to your system. The second way is for different kind of distributions, in this case in necessary download the program and install manually.

Debian based distributions - download deb file

You can directly download the deb file in internet and install using your favorite tool. To download the deb file go to the web page :
http://www.lpmd.cl/Download/lpmd-installer/
and download the las version available. To start the installation procedure, you can open a terminal and install the package using dpkg command.

```
user@machine:$ dpkg -i lpmd-installer.deb
```

Or on the other hand you can make double-click on the package in a more typical ubuntu system.



(a) The deb package in the desktop. (b) Double click to the lpmd-installer package.

Figure 2.1.: If you make double click to the package the installation process of the lpmd-installer package will begin.

Debian based distributions - adding repository

At first place, as administrator, edit the file `\etc\apt\sources.list` and add the following lines.

```
#GNM Repositories
deb http://arpa.ciencias.uchile.cl/repo/ gorilla main
```

You can make the same procedure in a Ubuntu distribution using directly the software center and adding the software source adding :

- Type : Binary
- URL : http://arpa.ciencias.uchile.cl/repo/
- Distribution : gorilla
- Components : main
- Comment : A GNM source/binary distribution.

the figure 2.2 show how to setup the repository in a ubuntu distribution in order to install the `lpmd-installer` package.

Finally, as administrator execute :

```
apt-get update
apt-get install lpmd-installer
```

Now check that you have the `lpmd-installer` command :

```
username@machine:~$ lpmd-installer
lpmd-installer [ -i <branch or version> | -u ] [-v] [-p <prefix>] [-P <package>]
               [-s <server>] [-S <suffix>] [ -d <sources dir>] [-t]
username@machine:~$
```

Great! you are ready to install `lpmd` easily.

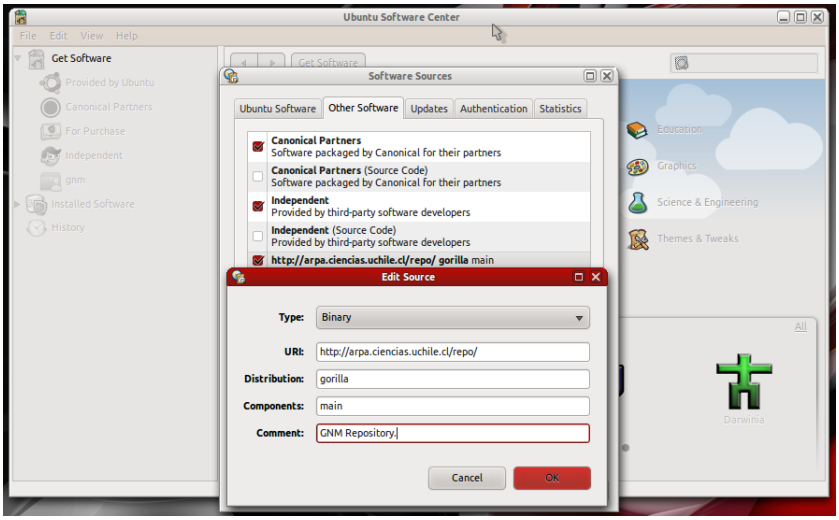


Figure 2.2.: Setting the repositories in a ubuntu distribution.

Others Distributions

In order to install `lpmd-installer` in distributions non based on Debian, is necessary download the program directly from internet, you can found the program here :

http://www.gnm.cl/lpmd/uploads/Main/lpmd-installer

Download the file and change the execution permission of the file :

```
chmod 755 lpmd-installer
```

Afther that, you can copy or move the file to some plece where you have set the `PATH` environment variable. In mostly of the system *nix is enoguh copy to :

```
cp lpmd-installer /usr/local/bin/
```

And this is it, now check that you have the `lpmd-installer` command available in your system.

```
username@machine:~$ lpmd-installer
lpmd-installer [ -i <branch or version> | -u ] [-v] [-p <prefix>] [-P <package>]
               [-s <server>] [-S <suffix>] [ -d <sources dir>] [-t]
username@machine:~$
```

Great! you are ready to install `lpmd` easily.

2.1.2. Testing lpmd-installer

`lpmd-installer` have all necessary in order to install `lpmd` in a system or in personal accounts, like as facilities fo install external *plugins*, like `lpvisual`. Between the principal options of `lpmd-installer` we have :

- `-l` : Show all version available to install.
- `-p` : Espcify the path to install lpmd(prefix).
- `-P` : What package you will install.
- `-d` : Set where the sources will be, if you don't set this values the sources are deleted after the installation process.
- `-u` : Update the `lpmd` version that you have in your system.

Below, we will show you how to install `lpmd` using `lpmd-installer`.

2.1.3. Installing lpmd on the system using lpmd-installer

Before run the installation process of `lpmd` in the system, we recommend that you take a look of the available versions in the principal server, for this just run `lpmd-installer -l`, and you will se a list with the packages, like this :

```
username@machine:~$ lpmd-installer -l
These are the available versions of package lpmd:

From branch 0.5:
    0.5.4-delta2
    0.5.4

From branch 0.6:
    0.6.1

From branch testing:
    svn
username@machine:~$
```

You have to choose wath version do you want install, in this case we will consider the version 0.6.1, if we try to install in the system without administrator permissions, you will have a message like this :

```
username@machine:~$ lpmd-installer -i 0.6.1
Preparing to install lpmd 0.6.1
[Error] You do not have permission to install on /usr/local
username@machine:~$
```

This happen because, `lpmd` by default try to be installed in `/usr/local` and a normal user have not privileges to write in this folder. The right way to proceed is, install as a administrator. For this :

```
username@machine:~$ sudo -s
[sudo] password for username:
root@machine:~# lpmd-installer -i 0.6.1
```

Now, the process will begin in automatic way to download from the principal server the necessary packages to install `lpmd`. Remember that you will request at least one `C++` compiler in order to realize the installation. By the other hand we suggest that you have too installed the `zlib` library, in mostly of the *nix system is called `zlib1g-dev`, for to this in Debian based dsitribution, the preocedure will be :

```
username@machine:~$ sudo -s
[sudo] password for username:
root@machine:~# apt-get install zlib1g-dev
```

2.1.4. Installing lpvisual with lpmd-installer

With `lpmd-installer` you can realize the installation of the `lpvisual` plugin, this is a independient plugin based in OpenGL and is used for the visualization of atomic configurations of different types. For a right compilation of the source code, you will need install the principal libraries of the OpenGL project, for this, in Debian based distribution you can use :

```
username@machine:~$ sudo -s
[sudo] password for username:
root@machine:~# apt-get install libglut3-dev
```

Now, to install `lpvisual` at the system you can use the same options of `lpmd-installer` used previously :

```
username@machine:~$ lpmd-installer -P lpvisual -l
These are the available versions of package lpvisual:

From branch 2.0:
    2.0svn
username@machine:~$
```

So, we will choose the version of `lpvisual` that you want install, and you install this easly with :

```
username@machine:~$ lpmd-installer -P lpvisual -i 6.0
....
```

With this you installed an additional plugin using the lpmd-installer package.

2.1.5. Installing lpmd in a personal account

Sometimes, the user is not an system administrator, under this, you can install `lpmd` in a persinal account, for that `lpmd-installer` have additional flags that you can use for this. Take in acount an example, supouse that you want install `lpmd` in a local folder, for example `~/local`. For this installation process is necessary day where is the directory using the flag `-p` as :

```
username@machine:~$ lpmd-installer -i 0.6.1 -p /home/username/local
....
username@machine:~$
```

With this command we installed `lpmd` en the personal account, we can also choose the possibility to keep the source code in other directory, for example using :

```
username@machine:~$ lpmd-installer -i 0.6.1 -p /home/username/local -d /home/username/sources
....
username@machine:~$
```

For user support and more tips about `lpmd-installer` yo can send an e-mail to the developers or visit the web-page of the project in http://www.lpmd.cl.

2.2. Download without lpmd-installer

If you choose do not install `lpmd` using the `lpmd-installer` program, then you have to download the three principal packages in order to install `lpmd`. You can choose two different version, the last stable version or the developer version, in both you will have to download all the three packaes, the `API`, the plugins set and the executables (with utilities).

2.2.1. Download the last stable version

The last stable version of `lpmd` packages are :

- liblpmd : ver. 0.2.2
- plugins : ver. 0.2.2
- lpmd : ver. 0.6.2

You can download this in the web page http://www.lpmd.cl or request them by e-mail. The webpage **always** is more updated that this document.

2.2.2. Download the last updated development version

For the people that are interested in use tha last version, including last updates, mprovements and bug correction, can download the last updated development (LUD) version in the web-page.
http://www.lpmd.cl/testing
The installation procedure for this packages is similar to the last stable version that is showed in section 2.3.

2.3. Installation without lpmd-installer

In order to install `lpmd` without using `lpmd-installer` the first step is install the `API` of the project before install any other package. The order for the next installation packages will be no important.

2.3.1. Installing the liblpmd API

At first, uncompress the package of **liblpmd**.

```
tar -xvzf liblpmd-X.X.X.tar.gz
```

This will generate a new folder. In order to install this library with all the necessary issues for the good operation of `lpmd`, execute :

```
./setup
make
```

and with administrator privileges,

```
make install
```

By *default* the installation folder of the `API` is `/usr/local`, in the case that you need locate the api in a different folder, set this using the `--prefix` option in the `./setup` process. And in particular if you want install `lpmd` in your personal home check the section **??**.

In case of error messages or any other problems during the installation of the `API` package, please send an e-mail to the developers or to `gnm@gnm.cl`.

2.3.2. Installing plugins

One of the basic requeriments in the installation process of `lpmd` is have a well defined location of the plugins that `lpmd` will use. If the `API` package was installed in a standard way, you can configure and install plugins easily using :

```
./setup
make
```

and with administrator privileges, run :

```
make install
```

With this standard procedure, all the plugins will be installed in the folder `/usr/local/bin/lpmd`. If you install the `API` in a different location using the `--prefix` option, we suggest that you use the same place for the plugin using the same option in the `./setup` command.

2.3.3. Installing lpmd

This is the smaller and faster packages to install. Is similar to the previous packages, if you use the standard procedure and location, execute :

```
./setup
make
```

and with administrator privileges, run :

```
make install
```

If you choose a different location for the `API`, we suggest that use this location too in the `./setup` process for the final package, for this you can use the `--prefix` option too.

Finally you will have a set of different executables availables, `lpmd`, `lpmd-analyzer`, `lpmd-converter`, `lpmd-visualizer` and `lpmd-plotter` availables in your system. In a standard installation process this executables are installed in `/usr/local/bin/`, if you install `lpmd` in a different folder please correct your `PATH` in order to have acces to the executables.

You can run `lpmd` using :

```
username@machine:~$ lpmd
...
LPMD version 0.6.1
Using liblpmd version 2.0.1

Usage: lpmd [--verbose | -v ] [--lengths | -L <a,b,c>] [--angles |
-A <alpha,beta,gamma>] [--vector | -V <ax,ay,az,bx,by,bz,cx,cy,cz>]
[--scale | -S <value>] [--option | -O <option=value,option=value,...>]
[--input | -i plugin:opt1,opt2,...] [--output | -o plugin:opt1,opt2,...]
[--use | -u plugin:opt1,opt2,...] [--replace-cell | -r] [file.control]
       lpmd [--pluginhelp | -p <pluginname>]
       lpmd [--help | -h]
```

2.3.4. Update the lpmd packages

In order to update your actual version of `lpmd` in your system. You have two alternatives. The first one is using the normal installation process explained in the section 2.3 and overwrite your actual version.

The second procedure in order to update your actual version of `lpmd` is using the `lpmd-installer` package. The procedure to update your installation is easy, only type:

```
lpmd-installer -u
```

With this an automatic process will be start using the last update version of `lpmd`.

3. The Control File

One of the fundamental pieces in `lpmd` to run a molecular dynamics simulation is the **control file** of the system. This file specifies all the requirements that the simulation needs to be executed, including the name and type of file where the positions and velocities of the atoms are, if these are needed. In fact usually we are going to work with two kind of files :

- input-file : Atomic positions and cell details. This file have principally the atomic structure of our simulation.
- control-file : Specify the simulation procedure details, some times the atomic structures are included too in this file.

Now, before we continue, we are going to give a short description of the files that have the information about the atomic positions and cell specifications: the input files. And the different types that `lpmd` can handle.

3.1. The file with the atomic positions, the input-file

`lpmd` has many modules (see chapter 4.1), and the input/output files are one of them. The modules are plugins of the program that, in this case, manage the reading and writing of atomic positions and/or velocities and/or accelerations. Some available formats (modules) for doing this are availables in `lpmd` **xyz**, **lpmd**, **dlpoly** (`CONFIG` or `HISTORY` files types from dl_poly), etc.

We expect that the users, depending on their needs, help us to implement (or ask the developers to do it) new input/output modules. Now let's see breafly what they consist of.

3.1.1. The file type .xyz

This is one of the more standard format used in a lot of different molecular dynamics codes. It is an ASCII file that contains the position (in cartesian coordinates) and atomic symbol of each atom of the simulation. The structure of the file is basically:

`N`	Specifies the number of atoms in the simulation cell.
`comment`	A line for comments, title, etc. This line is usually left in blank.
`Sym X Y Z`	Atomic symbol, X coordinate, Y coordinate and Z coordinate.

Currently, the module `xyz` has three different levels (0,1,2) which indicate the amount of information for each atom. The 0 (zero) level indicates that the file contains (or will contain) the symbol and the 3-dimensional position of each atom only in Åunits. The level 1 indicates that the file contains not only the symbol and position, but the velocity (7 columns file) in Å/fs units. The level 2 is used when the user also needs the aceleration of each atom (3 aditional columns) in Å/fs^2 units.

- 0 : `Sym X Y Z` (default value).
- 1 : `Sym X Y Z VX VY VZ`.
- 2 : `Sym X Y Z VX VY VZ AX AY AZ`.

3.1.2. The file type .lpmd and .zlp

The `.lpmd` and `.zlp` file format are a special type of format in `lpmd`. This files are plane ASCII format (lpmd) and a compress format (zlp). The principal structure for this kind of files is given by :

`LPMD X.X C`	Header line with information about the version of the file (X.X) and a special char (C) that indica
`HDR SYM X Y Z`	Show the information that every atomic line will have.
`cell properties`	A line with cell structure information.
`Sym sx sy sz`	Atomic information, depends of the HDR line.

In a similar way that for the case of `xyz` file types, the `lpmd` file types have different levels (0, 1 and 2), but on the other hand they have available different additional information about the atoms, information like the atom color, or atom tags.

3.1.3. Otros Formatos Soportados

The two principall supported formats are `xyz` and `lpmd` (compress and not compress variants). We have implemented another input plugins in order to read or write atomic configurations from the file. All the availables to the date are listed in the tables A.1 and A.2. If you have questions or suggestions respect to the supported formats, please do not hesitate to contact us.

We do not give a detailed information about this plugins here, because is not deeply necessary.

3.2. The control-file

This is the principall file in order to realize a computational simulation. For this reason we will give you a description of each part of this file and then we will explain each one separately and deeply.

First of all, a general considerations about the control files:

- # Is a comment line. (Avoid this lines in a `use ... enduse` section).
- The `/` symbol indicate that the linea continue in the next line.
- Altought they may be random, we suggest that you have a order with the plugins load.
- You have to load a plugin, before use it.
- Some plugins are mandatory, for example **cellmanager**. However many of them can be applied in command line execution.

With all this point in mind, we will see the principal sections to consider in a control file in order to use `lpmd`. A general scheme is show next.

```
#Cell Properties   →  Properties about the CELL.
cell ...
#Input/Output      →  Input/Output atomic files
input ...
output ...
#General           →  General Simulation settings
prepare ...
steps ...
monitor ...
#Filters           →  Filters in the atomic configuration.
filter ...
#Module Load       →  Load modules.
use ...
enduse ...
#Module Apply      →  Applying modules.
apply ...
potential ...
integrator ...
```

This is a general approximation to the control files used in molecular dynamics by `lpmd`. With this idea in mind we will analyse each of these section separately and deeply.

3.2.1. Simulation Cell.

Es la propiedad que describe la celda de simulación. Es decir el detalle completo de cada uno de los ejes que la conforman, los que pueden ser entregados en forma detallada o en forma general. A continuación se describirá la forma en la cuál se entrega ésta propiedad, que *casi siempre* debe estar presente en el fichero .control y va al comienzo de éste.

cell

El flag cell es utilizado para describir la celda de simulación y asignar las propiedades de ésta. Generalmente una celda de simulación puede venir descrita ya en el formato del fichero de entrada (como es el caso de lo ficheros `lpmd` o `CONFIG`). Sin embargo, hay formatos, como el `xyz`, que no poseen la descripción de la celda, es por eso que es necesario en algunos casos utilizar esta opción. Si se desea dar la opción `cell` aunque la información se encuentre en un archivo, ésta predominará sobre la del archivo.

- Forma 1

  Se utilizan la longitud de los lados y ángulos de la celda, como sigue:

  ```
  cell a=10 b=5 c=5 alpha=45 beta=90 gamma=90
  ```

  donde,

a	=	indica el largo de la celda en **a**.
b	=	indica el largo de la celda en **b**.
c	=	indica el largo de la celda en **c**.
alpha	=	indica el ángulo α.
beta	=	indica el ángulo β.
gamma	=	indica el ángulo γ.

- Forma 2

  Se utilizan los 3 vectores bases, poniéndolos de la siguiente manera en el fichero, note que / indica la continuación de una línea única.

  ```
  cell ax=1.0 ay=0.0 az=0.0 bx=0.0 by=1.0 bz=0.0 /
  cx=0.0 cy=0.0 cz=1.0
  ```

  donde: ai, bi y ci con i=x,y,z son las coordenadas x, y y z de los vectores bases.

  Las posibles formas de ingresar una descripcion de la llamada `cell` pueden ser entregadas como argumentos en la ejecución de `lpmd` y no necesitan estar dentro del fichero de **control**, lo que ayuda a la creación de *scripts*.

  ```
  lpmd -L a,b,c -A alpha,beta,gamma archivo.control
  lpmd -V ax,ay,az,bx,by,bz,cx,cy,cz archivo.control
  ```

- Forma 3

  Existe una forma extra para celdas cubicas que ahorran un poco la escritura completa de cada término de la celda, por ejemplo una celda cubica de largo 5Å, se puede asignar facilmente con :

  ```
  cell cubic a=5
  ```

Omitiendo cell

Como se mencionó previamente, hay ocaciones en que el archivo de posiciones atómicas posee además la información de la celda de simulación, para estos casos hay dos formas de **especificar** a `lpmd` que debe leer la información desde ese archivo.

- Forma 1

  Se utilizan opciones especiales dentro del mismo fichero de control :

  ```
  set replacecell true
  ```

- Forma 2

  Se especifica en la ejecución misma de lpmd con el flag `-r`.

  ```
  lpmd archivo.control -r
  ```

3.2.2. Entrada - Salida

input

Existen actualmente dos formas de ingreso de un sistema de entrada para la configuración atómica que se requiere simular; estas son, el ingreso de las posiciones atómicas de la celda a través de un archivo (por ejemplo `.xyz` o `.lpmd`) y el otro es mediante módulos que generan automáticamente celdas atómicas con ciertas propiedades, por ejemplo celdas **bcc**, **fcc**, etc.

Veamos brevemente a continuación cada uno de ellos,

- Con Fichero

  Para cargar un fichero con configuraciones atómicas es necesario la existencia del módulo que reconozca el tipo de fichero, por ejemplo para cargar un fichero del tipo `.xyz`, es necesario utilizar el módulo `xyz` para poder leer sin problemas el archivo, ya que es el módulo el que *entiende* el archivo de ese tipo.

- Generadores de celda

  A diferencia con el método anterior, este método no requiere de un fichero con posiciones atómicas, en lugar de ello se requiere un módulo que genera automaticamente una celda con átomos, según los requerimientos propios del módulo. Por ejemplo existen módulos actualmente para generar celdas del tipo **sc**, **bcc**, **fcc**, etc. utilizando el plugin `crystal3d`, también hay generadores de redes bidimensionales (`crystal2d`) y finalmente se pueden utilizar métodos más sofisticados como **skewstart**.

La forma general de la orden `input` requiere de argumentos para un funcionamiento adecuado. Para ver más información sobre el módulo revise la sección 4.1. Estos son ejemplos de algunos argumentos:

module	=	indica el módulo con el que cargar la celda.
file	=	indica el fichero con posiciones atómicas.
level	=	indica el nivel del fichero.

Estos son los argumentos más standard ya que cada módulo posee sus propios argumentos, por lo que se hace necesario ver cada uno según el interés.

Veamos algunas formas de uso para la orden `input` :

- Carga posiciones atómicas desde un fichero XYZ.

```
input module=xyz file=fichero.xyz level=0
```

- Carga posiciones y velocidades desde un fichero XYZ (level 1).

```
input module=xyz file=fichero.xyz level=1
```

- Inicializa una celda del tipo fcc con tomos de Au.

```
input module=crystal3d type=fcc nx=3 ny=3 nz=3\
symbol=Au
```

- Inicializa una celda del tipo sc con tomos de Na

```
input module=crystal3d type=sc nx=5 ny=5 nz=5\
symbol=Na
```

- Inicializa con metodo skewstart para 108 átomos de argón.

```
input module=skewstart atoms=108 symbol=Ar
```

La lista de los módulos soportados a la fecha para lectura/generación de configuraciones, se pueden observar en la tabla A.1 y A.2.

output

Con el parámetro `output` se especifican las opciones de salida de las configuraciones atómicas de nuestra simulación, los formatos de salidas son complementamente modulares y pueden ser implementados por los usuarios, sin embargo a partir de la version 0.5.2 del set de plugins `lpmd-plugins` ya se encuentran disponibles muchos módulos, pese a esto es *importante* notar que **cada módulo posee configuraciones independientes**, por ejemplo `level` es utilizado por módulos como **xyz**, **dlpoly** o **lpmd**, sin embargo no es requerido para **mol2**, para más información refierase a la sección 4.1. Los argumentos generales más utilizados del parámetro `output` son:

module	=	indica el módulo (formato) de salida de la simulación.
file	=	indica el fichero en el que graba.
level	=	indica el nivel del modulo de salida.
each	=	indica cada cuantos pasos la celda es gabada en el fichero.

Al igual que antes, existen más parámetros que son independientes de cada módulo.
Algunas formas de uso,

- Grabando la simulación en un fichero XYZ (nivel 0), cada 20 steps.

```
output module=xyz file=fichero.xyz level=0 each=20
```

- Grabando la simulación en fichero LPMD (nivel 1), cada 1 step.

```
output module=lpmd file=fichero.lpmd level=1 each=1
```

- Grabando las posiciones atómicas en formato lpmd nivel 2 y con colores de los átomos.

```
output module=lpd file=saved.lpmd level=2 each=5\
extra=rgb
```

La lista de los módulos soportados a la fecha para escritura de configuraciones, pude verse en la tabla A.1.

restore

Es utilizado para restaurar una simulación a partir de un punto en que se produjo un corte de energía eléctrica o cualquier otro tipo de falla física en un centro de cálculo. El punto de restauración es a partir de el último dumping realizado por la simulacion, dado por la orden "dumping" dentro del fichero de control, indicando el nombre del archivo `dump`. Es recomendable que antes de reiniciar una corrida, respalde los datos en otro directorio, o efectue la *reiniciación* de la corrida en un directorio distinto, para evitar dañar, perder o sobreescribir los datos previos de la simulación.

Actualmente no se han realizado pruebas exhaustivas de este punto, pero debería funcionar sin problemas, por favor si encuentra algún bug, reportelo y **recuerde usar esta opción con precaución**.

Consideremos una corrida estandard de dinámica molecular en donde el fichero de control posee la línea :

```
dumping file=restauracion.dat each=50000
```

de esta forma el sistema guardará una configuracion de restauración cada 50 mil pasos en el fichero `restauracion.dat`. Ahora si ocurrió alguna falla durante el calculo, podemos reiniciar la corrida con `lpmd` para ello recomendamos copie todos los archivos en otro directorio y luego aada las siguientes líneas al fichero de control

```
...
dumping file=restauracion-2.dat each=50000
restore file=restauracion.dat
...
```

De esta forma finalmente el sistema correra a partir del paso (múltiplo de 50 mil) en el cuál se produjo el error.

3.2.3. Propiedades Generales

prepare

Esta opción es utilizada para *setear* valores y características de la simulación, que son brindadas a través de `plugins` o de la misma `API`, tenemos por ejemplo :

- **temperature** Para dar una temperatura inicial al sistema, se prepara la celda con :

```
prepare temperature t=300
```

  de esta forma el sistema asigna velocidades iniciales a las partículas para que la temperatura de nuestro sistema corresponda a 300K en el instante de tiempo inicial.

- **replicate** Para replicar nuestra celda en las distintas direcciones de los vectores bases, la forma de hacerlo para una celda antes de comenzar la simulación es:

```
prepare replicate nx=2 ny=2 nz=2
```

  De esta froma, la celda que se leyó en `input` es replicada 2 veces por cada eje, alcanzando 8 veces el número inicial de partículas. Esta opción es válida sólo cuando se desactiva la optimización previa utilizando `set`.

set

Utilizado para setear valores de la simulación, principalmente para algunas variables globales del sistema. A continuación algunos de los más utilizados :

- Desactivando la optimización de celda previa a la simulación.

```
set optimize-simulation false
```

- Se asigna que la información necesaria para `cell` está en el fichero de entrada.

```
set replacecell true
```

- Seteando la variable `delay` usada en visualización.

```
set delay 0.1
```

charge

Asignación de las cargas en `eV` para las especies atómicas. Estos valores de las cargas, son seteados principalmente para utilización de potenciales interatómicos en los cuales se utilizan las cargas de los atomos involucrados.
Forma de uso

- Seteando las cargas de los atomos de O y Ge.

```
charge O XX
charge Ge XX
```

mass

Asignación de la masa en `a.u.` para las especies atómicas. Estos valores, son seteados principalmente para utilización de potenciales interatómicos en los cuales se desea modificar la masa de los átomos involucrados.
Forma de uso

- Seteando las cargas de los atomos de O y Ge.

```
mass O XX
mass Ge XX
```

periodic

Indica la periodicidad de la celda, en cada eje. Al bloquear la periodicidad en un eje, este se ve "modificado" en ambos lados de la celda, revise con cuidado estas opciones.

```
periodic false false true
```

En éste caso sólo tenemos periodicidad en el eje `z`. El orden de la periodicidad es `x`, `y` y `z`.

steps

Número de pasos de la simulación de dinámica molecular.

```
steps 10000
```

Acá se indica que la simulación se realizará con 10000 pasos.

dumping

Genera una salida global del sistema para poder restaurar a partir de ese punto.

```
dumping file=rescue.dump each=10000
```

Generamos un fichero de volcado cada 10000 pasos de la simulación, en él se graba toda la información necesaria, para reiniciar una corrida. Para más detalle sobre el uso de los ficheros de restauración refiérase a 3.2.2.

monitor

La orden `monitor` indica cada cuantos pasos la simulación muestra las propiedades globales. Estas propiedades, pueden ser asignadas por el mismo usuario, haciendo la salida lo más configurables según los propios requerimientos.

Entre las opciones de monitor, cuentan :

start	=	indica el valor de epsilon.
end	=	indica el valor de sigma.
each	=	indica el cutoff del potencial.
properties	=	indica que valores se desea monitorear.
output	=	archivo de salida para guardar los valores, si no, el *standard output* es utilziado.

Si queremos ir chequeando, los valores de la energía durante la simulación, cada 10 pasos, utilizamos la línea,

```
monitor start=0 end=1000 each=10 properties=step,kinetik-energy,\
        potential-energy,total-energy output=salida.out
```

Algunas de las opciones soportadas por **properties** son:

- step : Muestra el paso actual de la simulación.
- kinetic-energy : Muestra la energía cinética.
- potential-energy : Muestra la energía potencial.
- total-energy : Muestra la energía total.
- temperature : Muestra la temperatura del sistema.
- virial-pressure : Aporte del término del virial a la presión.
- kinetic-pressure : Término de la presión asociado.
- pressure : Presión total del sistema.
- volume : Volumen de la celda de simulación.
- cell-x : x=a,b,c son los largos de la celda en cada eje.
- sij : i,j=x,y,z entrega los valores para el tensor de stress.

3.2.4. Filtros

Aparecen en versiones posteriores a 0.6.0, son módulos cuya característica principal es *filtrar* los átomos de una simulación acorde a ciertos tipos de requerimientos, por ejemplo :

index	:	Filtrado de atomos según sus índices.
box	:	Filtrado de átomos pernetecientes a paralelepípedo.
sphere	:	Filtrado de átomos según esfera.
element	:	Filtrado de átomos según simbolo atómico.

La ventaja de estos filtros es poder generar nuevas configuraciones a partir de resultados previos, así como también análisis mucho más detallados para los *átomos filtrados*.

3.2.5. Carga de Módulos

Acá mostraremos cómo se utilizan en general la carga de módulos dentro de un fichero de control. Los módulos o plugins pueden ser cargados en cualquier sección del fichero de control, sin embargo recomendamos hacerlo de forma ordenada como veremos en los ejemplos posteriores.

Cómo cargar un módulo

Los módulos son de distintos tipos en general, y los de un tipo en común comparten "ciertas" características ya que cada uno posee sus propias ventajas. Una visión general de cómo se han distribuido los módulos es:

Generadores de celda	:	Tales como `xyz`, `pdb`, `crystalfcc`, etc.
Manejadores de celda	:	`minimumimage`, `linkedcell`, etc.
Modificadores de celda	:	`cellscale`, `tempscalling`, etc.
Filtros	:	`sphere`, `box`, etc.
Calculadores de Propiedades	:	`gdr`, `angdist`, `msd`, etc.
Integradores	:	`verlet`, `euler`, etc.
Potenciales	:	`LennardJones`, `SuttonChen`, `Morse`, etc.

En general, siempre a un módulo se le puede asignar un **alias** para su posterior llamado, por ejemplo.

```
use MODULO as ALIAS
...
enduse
```

De esta forma el módulo `MODULO` puede ser llamado en forma posterior con el nombre `ALIAS`, lo que da una ventaja para combinar y simplificar la utilización de un módulo en más ocaciones.

Entonces dentro de un archivo de **control**, debemos cargar los módulos necesarios para un posterior llamado.

3.2.6. Aplicación de módulos

Los módulos son llamados en la parte final de nuestro fichero de **control**, como existen distintas "especies" de módulos, estos deben ser llamados de diferentes maneras, aunque su forma es muy general.

- **Modificadores** Módulos llamados para modificar alguna propiedad de la celda de simulación. Se aplican con:

```
apply module-alias start=0 end=1000 each=20
```

- **Filtros** Módulos que modifican la celda de simulación filtrando átomos según ciertas condiciones, pueden ser utilizados tanto con la instrucción `over` de **apply** como con **filter**.

```
apply mod-alias-apply <apply-options> over mod-alias-filter <filter-options>

filter module-alias <options>
```

- **Calculadores de Propiedades** Estos siempre son llamados a ser evaluados cada cierto tiempo entre un rango de intervalos. A partir de la versión 0.6.0 de `lpmd` se pueden además calcular propiedades sobre un *set* de átomos filtrados con `over`.

```
property module-alias start=0 end=1000 each=10
```

```
property gdr start=0 end=-1 each=5\over <filter-options>
```

- **Integradores** Estos son llamados facilmente con:

```
integrator module-alias
```

- **Potenciales** Estos definen una interaccion entre dos átomos (por el momento no hay de tres cuerpos). Se crea un potencial para cada interacción especificando los valores y llamando luego con

```
potential module-alias Pt Au
```

- **Manejadores de celda** Llaman al manejador de celda que se utilizara en la simulación, en nuestro caso hay dsiponibilidad de dos manejadores, `linkdecell` y `minimumimage`.

```
cellmanager linkedcell
```

3.3. Ficheros de Salida

3.3.1. Ficheros de salida

`lpmd` Tiene variados tipos de ficheros de salida, uno que es generado usualmente por la opción `output` dentro del fichero de control, otro es la salida standard, que por defecto va a pantalla, pero que nosotros recomendamos enviar (o redireccionar) a un fichero además estan los que generan cada uno de los módulos en forma independiente cuando estos requieren de aquello.

Salida standard

Esta es la salida que muestra en pantalla lpmd, la forma de enviar esta salida a un fichero, durante la ejecución de lpmd, es

```
lpmd fichero.control >  salida.out
```

o si desea verla en pantalla y enviar a un fichero:

```
lpmd fichero.control | tee salida.out
```

también puede redireccionar una salida standard y los mensajes de error de forma independiente utilizando:

```
lpmd fichero.control 1&> salida.out 2&> salida.err
```

el fichero `salida.out` tendrá toda la información que debió salir a pantalla utilizando lpmd, entre ella se encuentran,

- Descripción completa de la celda
- Informacion de `startinfo`
- Información de módulos utilizados y variables de cada uno de ellos.
- Energías, Temperatura, Presión y Volúmen, según `monitor`. En caso de que monitor no utilize un valor propio para el valor de la variable `output`.
- *Debugger Information* : Información sobre aplicaciones y otras cosas que se realizan durante la simulación usualmente dirigida a `std::cerr`.

Fichero generado por output

Este fichero se generó con el nombre entregado en el fichero de control a la linea **output**, en él se encuentran las configuraciones atomicas de la **DM** y suelen ser los ficheros a partir de los cuales suelen crearse animaciones y análisis detallados de la simulación.

Todas las propiedades instantaneas pueden calcularse ***durante la simulación misma***, sin embargo `lpmd` también cuenta con herramientas propias de análisis como se puede ver en la sección 5.

Es importante definir un buen formato de salida ya que es la clave para simplificar o dificultar los análisis posteriores. Por ejemplo si se adquiere un archivo `xyz` con `level=0` no es el ideal para calcular por ejemplo la *Velocity autocorrelation Function* (`vacf`) ya que en caso de que el nivel sea menor a uno algunos plugins determinarán la velocidad utilizando configuraciones atómicas sucesivas de forma automática lo que aumentará un poco el tiempo de análisis, que pudo aver sido aprovechado durante la simulación grabando con `level=1` nuestro fichero de salida.

Ficheros generados por módulos

Muchos módulos de lpmd que realizan análisis generan ficheros de información de manera independiente, además los plugins en general manejan dos *estándares* de salidas asignados con el flag `average`. Por ejemplo si deseamos calcular la *Función de distribución de pares* sobre nuestras configuraciones, podemos calcular ésta sobre cada una de las muestras y guardarlas o bien generar un promedio sobre todas las configuraciones.

- `average true` :

  Especifća que luego de realizar los correspondientes análisis sobre las configuraciones especificadas del fichero, estas deben promediarse antes de guardar en el archivo de salida.

- `average false` :

  Especifíca que cada vez que realiza un análisis sobre las configuraciones pertenecientes al archivo, estas deben ser guardadas una a una en el archivo de salida.

4. Módulos

En éste capítulo se dará una descripción general de los módulos con los que cuenta lpmd, así como también propiedades de los mismos, no todos tendrán ejemplos, más bien se pretende dar una descripción de lo que es capaz de realizar cada módulo, sin embargo puede encontrar muchos ejemplos en el capítulo 7.

A continuación veremos los módulos y sus detalles dependiendo de su tipo, en cada una de las secciones siguientes. La intención principal de este capítulo es tener un punto de referencia para los módulos que vienen incorporados en `lpmd`.

4.1. Módulos Entrada/Salida

Estos módulos pueden ser cargados por `input` y `output`, por lo que los argumentos necesitan ser entregados en esta misma linea. Es decir la instrucción será siempre de la forma:

```
input module=mod file=..  opc1=..  ...  opcN=...

output module=mod file=..  opc1=..  ...  opcN=...
```

en donde `opcX` son las opciones de cada módulo en paticular, siendo `file` una opción común a la mayoría de los módulos de entrada/salida.

4.1.1. dlpoly

Este plugin, pretende dar una solución eficiente a los archivos de entrada o salida que el usuario ya posee en alguno de los formatos del programa `dl_poly` (ficheros `HISTORY` y `CONTROL`) y ahorrar el tiempo y trabajo que implica migrar estos archivos a otras configuraciones .

Con el módulo `dlpoly`, se pueden cambiar los archivos de configuracion cristalina de `dlpoly` (`HISTORY` o `CONFIG`) a cualquier otro módulo y viceversa. Los argumentos utilizados por el módulo `dlpoly` son:

file	=	Archivo que contiene la configuración de **dlpoly**
level	=	Nivel del archivo (0,1,2).
preiodicity	=	Especifica *key* de periodicidad del fichero CONFIG.
ftype	=	Especifíca el tipo de archivo, `HISTORY` o `CONFIG`, a ser leído.

4.1.2. lpmd

Es un formato propio de `lpmd`, tiene la ventaja de que no sólo guarda la información de las posiciones atómicas de las partículas, sino que además guarda información sobre la celda de simulación. Las posiciones, a diferencia de `xyz`, se encuentran escaladas, por lo que cuenta con opciones muy simples de manejo, encarecidamente recomendamos su uso para trabajos serios. Además a partir de la version 0.6.1 de `lpmd`, este plugin cuenta con la opcion de trabajar directamente formatos comprimidos con zlib. Entre las opciones principales destacan

file	=	archivo de entrada/salida.
level	=	indica el nivel del fichero, estos pueden ser 0(pos),1(pos y vel) y 2(pos,vel y ace).
extra	=	Información extra en el fichero sobre los átomos rgb, type, etc.
type	=	Especifíca el tipo de fichero, `lpmd`(default) o `zlp`.
blocksize	=	tamao del *block* de compresión cuando es del tipo `zlp`.

La forma común de uso en lpmd es,

- **Cargando un fichero lpmd**

```
input module=lpmd file=archivo.lpmd
```

- **Escribiendo la salida en un fichero lpmd con velocidades**

```
output module=lpmd file=salida.lpmd level=1
```

4.1.3. vasp

El plugin, es utilizado para leer ficheros POSCAR de `vasp` como un fichero de configuracion inicial, o bien para otro propósito, entre las opciones del módulo, estan:

file	=	Archivo que contiene la configuracion de **dlpoly**
species	=	lista de las especies (en orden) del fichero POSCAR.
level	=	Especifíca el nivel del archivo.
type	=	Tipo de posiciones, direct/cartesian.

4.1.4. xyz

Éste es el módulo que se utiliza para cargar/escribir archivos de configuración en formato `xyz`, las opciones actuales de este módulo son :

file	=	archivo de entrada/salida.
level	=	indica el nivel del fichero, estos pueden ser 0(pos),1(pos y vel) y 2(pos,vel y ace).
coords	=	utilizado para "posicionar" una celda que ha sido leída, sus valores pueden ser centered/uncentered.
inside	=	Puede ser true/false, indica si los átomos que están fuera de una celda se deben reubicar.
external	=	ignore/consider. Específica si se deben ignorar o considerar los atomos fuera de la celda.

Muchas de estas opciones no son utilizadas en una corrida con `lpmd`, sin embargo pueden ser utiles a la hora de trabajar con utilidades tales como `lpmd-analyzer` u otras. En general la forma de utilizar el módulo es,

- **Cargando un fichero xyz**

```
input module=xyz file=archivo.xyz
```

- **Escribiendo la salida en un fichero xyz con velocidades**

```
output module=xyz file=salida.xyz level=1
```

No todas las opciones son necesarias, muchas de ellas ya tienen valores por defecto, vease `lpmd -p xyz` para más información.

4.1.5. mol2

Utilizado para lectura/escritura de fichefros de tipo `mol2`, el soporte, al igual que el formato `pdb` es básico sin embargo es útil para convertir nuestras configuraciones.

entre las opciones típicas de `mol2` encontramos :

file	=	Archivo que contiene la configuracion atómica

4.1.6. pdb

Utilizado para lectura/escritura de archivos de tipo `pdb`, el soporte actual de este tipos de ficheros es básico, sin embargo puede ser útil para convertir ficheros para la visualización con otros programas.

Entre las opciones típicas de `pdb` están :

file	=	Archivo que contiene la configuracion atómica

4.1.7. rawbinary

Utilizado para lectura/escritura de archivos de tipo binario, son eficientes para grandes set de configuraciones, es similar al antiguo formato `lpmd 1.0` pero escrito de forma binaria, lo que agiliza cualquier análisis que se desee llevar a cabo.

Entre las opciones típicas de `rawbinary` están :

file	=	Archivo que contiene la configuracion atómica
level	=	Especifíca el nivel del archivo.

Generalmente este archivo es utilizado para analisis sobre configuraciones muy grandes, para reducir el tiempo de análisis.

4.2. Módulos Generadores de Celda

Estos módulos pueden ser cargados por `input`, por lo que los argumentos necesitan ser entregados en esta misma linea. Es decir la instrucción será siempre de la forma:

```
input module=mod opc1=..  ...  opcN=...
```

en donde `opcX` son las opciones de cada módulo en paticular. Estos módulos sólo *generan* celdas de simulación, su intención es hacer celdas de distintos tipos de manera fácil y amigable para el usuario. A continuación listamos los principales generadores de celda.

4.2.1. crystal3d

Genera una celda cristalina tridimiensional. Este módulo se utiliza en lugar de un fichero de entrada, para entregar las posiciones atómicas de la red cristalina,

symbol	=	Símbolo atómico de la especie.
nα	=	$\alpha = x, y, z$ El número de replicas de la celda base.
type	=	Tipo de celda, fcc, bcc, hcp, etc.

Consideremos por ejemplo una celda cúbica de **Fe** con un tamaño de 10 Åpor cada lado.

- **Especificamos el detalle del cristal**

```
cell crystal a=10 b=10 c=10 alpha=90\beta=90 gamma=90
```

- **Generamos una celda fcc dentro del cristal**

```
input module=crystal3d type=fcc symbol=Fe\nx=3 ny=3 nz=3
```

Esto generará entonces una celda cristalina `fcc` con parámetro de red $10/3 = 3.33$, para comenzar la simulación.

4.2.2. crystal2d

A diferencia de los generadores de cristalprevios, este módulo genera cristales simples bidimensionales, los argumentos requeridos para generar estos, son:

symbol	=	Símbolo atómico de la especie.
a	=	Longitud del vector base $\vec{a}$.
b	=	Longitud del vector base $\vec{b}$.
gamma	=	Ángulo entre los vectores base.
nα	=	$\alpha = x, y$ El inverso sobre el largo del cristal.

Consideremos por ejemplo una celda triangular con atomos de Ar, donde el tamano del cristal es de 10 Å.

- **Especificamos el detalle del cristal**

```
cell crystal a=10 b=10 c=0 alpha=90 beta=90 gamma=90
```

- **Generamos una estructura bidimensional dentro del cristal**

```
input module=crystal2d a=2 b=2 symbol=Fe gamma=60\nx=2 ny=2
```

4.2.3. voronoi

Utiliza la construcción de **voronoi** para generar nanoestructuras de materiales, entre sus principales características para la generación de estas nanoestructuras, están :

symbol	=	Símbolo atómico de la especie.
type	=	Especifíca el tipo de grano cristalino, fcc, bcc, etc.
a	=	Constante de red del cristal.
grains	=	Cantidad de granos generados dentro de la celda.

Al igual que los métodos generadores de celda previos, el detalle general del cristal es entregado con la orden `cell`, y el resto es especificado por el módulo, consideremos un ejemplo que para hacerlo más vistoso utilizará el plugin lpvisual y el *quick-mode*

```
lpmd-visualizer -i voronoi:symbol=Fe,type=fcc,a=3.62,grains=6 -u lpvisual -L 50,50,50
```

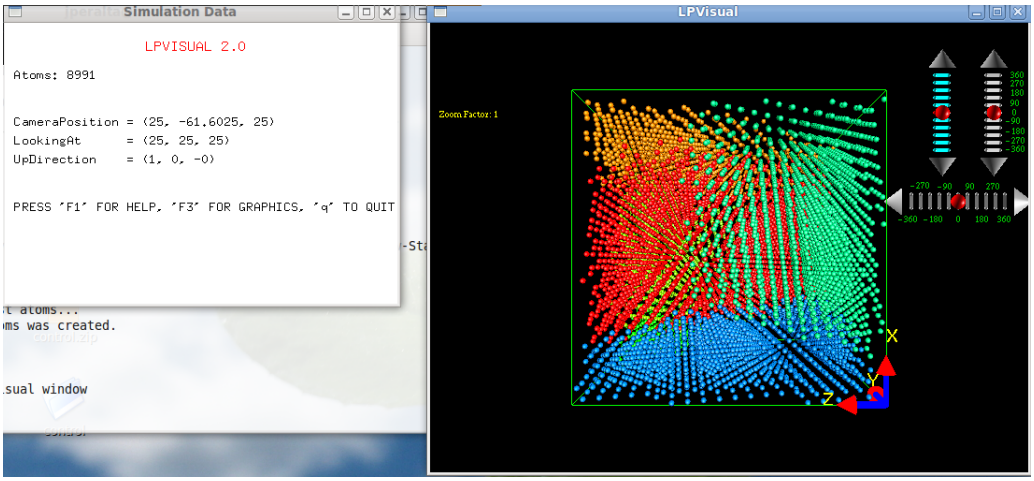


Figure 4.1.: Screenshoot de `lpmd-visusalizer` mostrando una celda generada con voronoi.

Podemos ver el resultado de la figura 4.2.3, si esta celda quisieramos guardarla en un formato `lpmd` incluyendo los colores bastaría ejecutarlo con :

```
lpmd-visualizer -i voronoi:symbol=Fe,type=fcc,a=3.62,grains=6 \
-u lpvisual -L 50,50,50 -o lpmd:file=new.lpmd,extra=rgb
```

4.2.4. skewstart

Método creado por *K. Refson* para el código de dinámica molecular **moldy**. El objetivo del método es generar una configuración no perdiódica suficientemente regular para garantizar una separación mínima entre los átomos. Las opciones junto con la forma de generar una celda con *skewstart*, se describe a continuación:

symbol	=	Símbolo atómico de la especie.
atoms	=	Especifíca el número de átomos dentro de la celda.

Al igual que los métodos generadores de celda previos, el detalle general del cristal es entregado con la orden `cell`, y el resto especificadio por el módulo, consideremos una celda de argón de 108 átomos con un tamaño de 20 Å y que inicializamos las posiciones atómicas con el método *skewstart*

- **Especificamos el detalle del cristal**

```
cell crystal a=20 b=20 c=20 alpha=90 beta=90 gamma=90
```

- **Generamos una celda fcc dentro del cristal**

```
input module=skewstart symbol=Ar
```

4.3. Manejadores de Celda

Estos módulos son los encargados de generar las **lista inteligentes** para poder realizar simulaciones o cálculos de Dinámica Molecular, éstas listas ayudan a reducir el tiempo de cálculo utilizado por las simulaciones.

4.3.1. linkedcell

Utiliza el método de listas linkeadas, éste método es mucho más rapido que el método de mínima imagen, y de por sí es el más utilizado en la mayoría de los códigos de **DM**, se puede subdividir la celda a voluntad, sin embargo es recomendable que el tamaño de subdivisión de la celda no sea menor que la distancia mínima de interacción entre las partículas, con algun potencial. También posee el modo `auto` para simplificar la subdivision de la celda.

- **Cargando el Módulo**

```
use linkedcell
cutoff 2.0
nx 10
ny 10
nz 10
enduse
```

- **Cargando en modo** `auto`.

```
use linkedcell
cutoff 2.0
mode auto
enduse
```

- **Aplicando el Módulo**

```
cellmanager linkedcell
```

4.3.2. minimumimage

Uiliza el método de mínima imagen para realizar los procesos. Es mucho más lento que otros métodos, pero en el sólo se necesita ingresar el radio de corte del sistema. Este método no sirve cuando el radio de corte es del orden del tamaño de la mitad de la celda de simulación.

- **Cargando el Módulo**

```
use minimumimage
cutoff 2.0
enduse
```

- **Aplicando el Módulo**

```
cellmanager minimumimage
```

Con ésto nuestra simulación utilizará el método de mínima imagen para un `cutoff` de 2.0 Å.

4.3.3. lcbinary

Utiliza el método de listas linkeadas, pero a diferencia del anterior, este adapta la subdivisión asignada de manera automática haciendo que no permanezcan más de un átomo por celda, pese a ser un metodo simple y *utilizable*, es recomendable que considere el excesivo consumo de `RAM` que éste acarrea.

- **Cargando el Módulo**

```
use lcbinary
cutoff 2.0
mode auto
enduse
```

- **Aplicando el Módulo**

```
cellmanager linkedcell
```

Si usted no tiene claro que tipo de subdivision debe usar recomenadmos encarecidamente que utilize `linkedcell` en modo `auto`.

4.4. Filtros

Estos módulos son los encargados de modificar de cierta forma la celda de simulación original. Pueden ser utilizados para la generación de celdas complejas. Su intención es seleccionar los átomos que se encuentren dentro de una región específica o que posean una característica dada, eliminando el resto.

Es importante destacar que todos los filtros pueden, además, actuar de forma inversa. Por ejemplo en vez de quedarse con los átomos que se encuentran dentro de una región dada, podemos, utilizando la misma región, seleccionar los átomos que no pertenecen a la región con el flag

```
inverse=true
```

dentro de las opciones del filtro. Además estos filtros pueden ser aplicados con excepciones (`except`) lo que les da una flexibilidad aún mayor, ejemplos sobre filtros pueden contrarse en el capítulo 7.

4.4.1. box

Selecciona los átomos que se encuentran dentro de una región rectangular definida por el plugin. Las opciones del plugin son

x	=	Rango en la dirección `x`.
y	=	Rango en la dirección `y`.
z	=	Rango en la dirección `z`.

Veamos un filtrado típico utilizando `box`.

- **Eliminando todos los átomos excepto los pertenecientes a una región.**

```
filter box x=0-5 y=0-6 z=10-15
```

Ejecutando este comando, desaparecen todos los átomos de la caja, excepto los que se encuentran en el rango $[0, 5]$ del eje X, $[0, 6]$ en el eje Y y $[10, 15]$ en el eje Z. Esto es útil cuando queremos cortar una región de interés dentro de la caja de simulación.

- **Eliminando sólo los átomos pertenecientes a una región.**

```
filter box x=0-5 y=0-6 z=10-15 inverse=true
```

Ejecutando este comando, realizamos la acción anterior, es decir, ślo eliminamos los átomos de la región especificada (hacemos un "agujero" en la caja).

4.4.2. element

Selecciona a los átomos segun el elemento al que pertenecen. Las opciones del plugin `element` están dadas por,

symbol	=	Símbolo átomico de los átomos a eliminar.

Consideremos por ejemplo seleccionar los átomos de `H` de una muestra original, para ello :

- **Eliminando átomos de** `H`.

```
filter species symbol=H
```

4.4.3. index

Selecciona a los átomos según su indice, muy utilizado para igualar configuraciones antiguas en donde el orden de los átomos señalaba alguna característica. Las opciones del plugin `index` están dadas por,

index	=	Lista de índices atómicos.

Consideremos por ejemplo eliminar los atomos del 1 al 3 o bien del 1 al 50:

- **Eliminando átomos del 1 al 3**.

```
filter index index=1,2,3
```

- **Eliminando átomos del 1 al 50**.

```
filter index index=1-50
```

4.4.4. sphere

Selecciona a los átomos ubicados dentro de una región esférica, las opciones del plugin son :

radius	=	Radio de la esfera en Å.
center	=	Centro de la esfera, anotado como vector $\langle a, b, c \rangle$.

Consideremos por ejemplo seleccionar sólo los átomos pertenecientes a una región esférica.

- **Selección esférica** `H`.

```
filter sphere radius=10 center=<50,50,50>
```

4.4.5. tag

Selecciona a los átomos segun algún `tag` que los caracterize, es decir alguna etiqueta propia de los átomos, uno puede asignar tag a los átomos según nuestras propias necesidades, sin embargo también existen `tag` propios de `lpmd`, tales como `fixedpos`, `fixedvel`, etc que son manejados por la `API`. Las opciones del plugin son :

name	=	Nombre del tag.
value	=	Booleano true o flase, sobre estado del tag.

Consideremos por ejemplo seleccionar los átomos que tienen posicion fija

- **Seleccionando átomos de posición fija** `H`.

```
filter tag name=fixedpos value=true
```

4.5. Modificadores

Los plugins modificadores alteran ciertas características de un set de átomos o de la celda de simulación completa, es decir las modificaciones pueden aplicarse de forma filtrada, por ejemplo :

```
apply modify-module start=...  end=...  each=...  over\filter-module <filter-options>
```

Lo que nos da mayor flexibilidad en el manejo de la simulación. Sin embargo si queremos modificar la celda en su totalidad, podemos usar simplemente :

```
apply modify-module start=...  end=...  each=...
```

Casos como el escalamiento de celda, solo pueden aplicar de forma *general* sobre la simulación.

4.5.1. berendsen

Éste módulo controla o escala la temperatura, de igual forma que **tempscaling**, su diferencia es que el método es un escalamiento menos **brusco**, por lo que en ocaciones se consiguen muy buenos resultados y se logra termalizar de manera más eficiente las muestras.

La ecuación que describe el termostato de berendsen es :

$$\frac{dT}{dt} = \frac{T_0 - T}{\tau}$$

Los argumentos requeridos por el módulo son:

from	=	Desde que temperatura comenzar a aplicar.
to	=	Temperatura final a la que se desea llegar.
tau	=	Intervalo del termostato.

4.5.2. cellscaling

Éste módulo es un modificador de la celda, en particular, tiene la propiedad de cambiar el tamaño de la celda durante la simulaci'on, lo que lo hace una herramienta muy util para ver, por ejemplos, comportamiento de materiales bajo distintas densidades durante una sola simulación. El módulo tiene las siguientes opciones:

percent	=	Especifica el porcentaje en el que se variará el largo de la celda.
axis	=	Indica, el eje (x,y o z) en el que se comprimira la celda (all=hidrostático).
constant	=	true/false indica si la modificacion es constante respecto al valor inicial.

Con éste módulo entonces podemos realizar compresión hidrostática o unidireccional, además de mantener constante o no el intervalo de escalamiento.

4.5.3. displace

Módulo que modifica las posiciones atómicas del sistema, a partir de un desplazamiento vectorial de los átomos de la muestra. Las caractrísticas del módulo son :

x	=	Desplazamiento en el eje x.
y	=	Desplazamiento en el eje y.
z	=	Desplazamiento en el eje z.

4.5.4. moleculecm

Crea moléculas diatómicas. Para cada átomo de la simulación busca átomos que estén cerca (en un radio especificado por el usuario) y si los encuentra, los considera enlazados y reemplaza cada par de átomos enlazados por un sólo átomo, ubicado en el centro de masas de ambos.

radius	=	Radio de busqueda en torno a cada átomo.

4.5.5. propertycolor

Módulo que asigna colores a los átomos segun cierta condición, las opciones principales del plugin son :

property	=	Propiedad que se desea colorear.(temperature, velocity, acceleration, neighbours)
min	=	Valor mínimo del rango de valores.
max	=	Valor máximo del rango de valores.
cutoff	=	Algunas propiedades necesitan cutoff.
filterby	=	Filtrado por.

Consideremos por ejemplo que deseamos colorear los átomos según su temperatura, y le asociaremos a 300 la temperatura mínima y 500 a la máxima que usa escala de colores del azul al rojo, entonces :

- **Cargamos el modulo**

```
use propertycolor as temp
min 300
max 300
enduse
```

- **Aplicamos coloreado en la simulación.**

```
apply temp start=1 end=1000 each=2
```

4.5.6. quenchedmd

Módulo que mínimiza la estructura, utilizando el método *Quenched Molecular Dynamics*. Durante el proceso, el integrador no actua como una dinámica molecular simple, ya que verifica la proyeccion entre fuerza y velocidad, la que se ve forzada si la energía se incrementa.

debug	=	Información de debug.

El módulo no necesita argumentos especiales, solo ser llamado y aplicado. Ya que debug es la información de depuración con la que siempre cuentan todos los plugins.

4.5.7. randomatom

Módulo que elimina o modifica de forma aleatoria átomos de una muestra, sus principales opciones son :

type	=	Tipo de acción delete/replace.
value	=	Valor porcentual de atomos a reemplazar.
symbol	=	Símbolo atómico, en el caso de reemplazar.
density	=	fixed/free para fijar o no la densidad de la muestra, útil en modo eliminación.

Consideremos por ejemplo que deseamos remover átomos al azar dentro de un cristal, entonces :

- **Cargamos el módulo**

```
use randomatom
type delete
value 5
density fixed
enduse
```

- **Aplicamos coloreado en la simulación.**

```
apply randomatom
```

la aplicación también puede llevarse acabo en pasos específicos de la simulaciíon, por ejemplo usando `start=10 end=10 each=1` llevara la eliminación de átomos en ese instante específico de la simulación, recuerde que todos los modificadores pueden asignarse así.

4.5.8. replicate

Módulo que replica la celda se simulación en los distintos ejes, sus opciones principales son

nx	=	Número de replicas en la dirección `x`.
ny	=	Número de replicas en la dirección `y`.
nz	=	Número de replicas en la dirección `z`.

Las replicas son números enteros de la celda, usualmente este módulo sólo se utiliza al comienzo de la simulación y puede ser llamado con `prepare` para setear previamente la celda. Es importante destacar que la mayoría de las ocaciones que **utilizan** `prepare replicate` necesitan desactivar la optimización de la celda, por ejemplo :

- **Desactivando optimización**

```
set optimize-simulation false
```

- **Aplicamos replicación de celda.**

```
prepare replicate 2 2 2
```

Con eso entonces replicamos nuestra celda original dos veces en cada dirección.

4.5.9. rotate

Módulo que modifica las posiciones, rotando los átomos cierto grado en torno a un origen. Al igual que el módulo `displace`, está desarrollado para la creación de estructuras más complejas.

x	=	Coordenada x del eje de rotación.
y	=	Coordenada y del eje de rotación.
z	=	Coordenada z del eje de rotación.
angle	=	Ángulo de rotación en **grados**.

4.5.10. setcolor

Módulo que asigna colores a los átomos, es necesario definir colores y con eso aplicarselo a un grupo de átomos o bien a todos los átomos.

color	=	Vector del color.

Entonces, veamos por ejemplo como asignan un color a todos los átomos, recuerde que puede ser aplicado utilizando filtros.

- **Carga el módulo**

```
use setcolor as red
color <1.0,0.0,0.0>
enduse
```

- **Aplicamos el color**

```
apply setcolor
```

4.5.11. settag

Al igual que la forma de asignar colores a un átomo, es posible asignar `tag` o etiquetas a los átomos. Entre las opciones del plugin settag estan :

name	=	Nombre del tag
value	=	Boleano que indica actividad del tag true/false

De esta forma podemos asignar cualquier tag a los átomos. Veamos por ejemplo como asignar `fixedpos` a un set de átomos, sería :

- **Carga el módulo**

```
use settag as pos
name fixedpos
value true
enduse
```

- **Aplicamos el color**

```
apply fixedpos
```

Usualmente las aplicaciones de `settag` o `setcolor` se utilizan fitlrados sobre los átomos para ello recuerdemos que basta con especificar el filtro dentro de `apply`.

4.5.12. setvelocity

Setea las velocidades de un grupo de átomos, los argumentos son :

velocity	=	Velocidad a asignar.

Al igual que `setcolor` este plugin puede asignar la velocidad de toda la muestra o de un grupo de átomos pertenecientes a ella.

4.5.13. shear

Módulo que realiza *shear* (cizalle) sobre la muestra, modificando la forma de la celda, los argumentos principales son :

axis	=	Eje en el que se produce el cizalle.
normal	=	Eje perpendicular al eje del cizalle.
strain	=	Desplazamiento máximo a aplicar es strain*L(normal)

Con esto entonces podemos aplicar cizalle a la simulación, por ejemplo consideremos que en lugar de aplicar el cizalle durante la simulación queremos aplicarlo en la celda de entrada original, entonces usamos.

```
prepare shear axis=X normal=Y strain=0.01
```

Recuerde que en general los módulos varían como son llamados según la forma en que se quiera usar.

4.5.14. temperature

Módulo que aplica una temperatura a un grupo de átomos, su opción principal es

t	=	Temperatura que se le desea asignar al grupo de átomos.

usualmente se utiliza mucho para la temperatura inicial de la celda de simulación es decir :

```
prepare temperature t=300
```

Eso asignará las velocidades iniciales de la muestra para esa temperatura.

4.5.15. tempscaling

Utilizado para controlar y escalar la temperatura de la muestra, utilizando rescalamiento de velocidades en las partículas. Éste es uno de los métodos más utilizados en muchos códigos de dinámica molecular.

El escalamiento consiste en modificar las velocidades atómicas cada cierto intervalo, en un factor:

$$s = \sqrt{\frac{gk_BT}{2\mathcal{K}}}$$

Es importante tener en cuenta, que el proceso de control o escalamiento de la temperatura, no entregan promedios termodinámicos reales, es necesario que estos sean evaluados luego de haber realizado el escalamiento.

los argumentos necesarios para el plugin son :

from	=	Temperatura inicial del escalamiento.
to	=	Temperatura final del escalamiento.

Consideremos ahora un ejemplo de utilización del módulo **tempscaling**, en donde calentamos una muestra de Ar a 300K y luego mantenemos la temperatura fija durante un periódo de tiempo, para finalmente liberar el sistema:

En primer lugar debemos cargar los modulos para cada una de las etapas de la simulación:

- **Cargamos el proceso de calentamiento de la muestra.**

```
use cellscaling as uptemp
from 84.0
to 300.0
enduse
```

- **Cargamos un proceso para mantener la temperatura fija.**

```
use cellscaling as fixtemp
from 300.0
to 300.0
enduse
```

Ahora, que el módulo fue cargado, es necesario, en la parte final del archivo de control, especificar entre que intervalos de tiempo van a ser aplicados estos escalamientos:

- **Aplicamos uptemp desde 1 hasta 10000, cada 50 pasos.**

```
apply uptemp start=1 end=10000 each=50
```

- **Mantenemos la temperatura fija entre 10000 y 15000.**

```
apply fixtemp start=10000 end=15000 each=50
```

De esta forma entonces hemos conseguido una muestra a 300K de temperatura con un proceso de *calentamiento* de la celda inicial.

4.5.16. undopbc

Módulo que deshace las condiciones periódicas de borde de una celda, se utiliza principalmente para deshacer configuraciones de DM.

El módulo no necesita argumentos especiales, solo ser llamado y aplicado. Veamos por ejemplo como quitar la periodicidad en una celda de configuración de forma directa sería utilizando.

- **Carga el módulo**

```
use undopbc
enduse
```

- **Aplicamos el color**

```
apply undopbc start=..  end=..  each=..
```

4.6. Propiedades Instantáneas

Las propiedades instantáneas son aquellas que se pueden evaluar en cualquier instante de tiempo sobre una configuración atómica, las que se listan a continuación son las que se han implementado a la fecha en `lpmd`.

Estas propiedades además de ser analizadas *post-simulación* pueden lelvarse acabo durante la simulación misma.

4.6.1. angdist

Calcula la distribucion angular de la celda de simulación. Para determinar los ángulos característicos de una celda de simulación es necesario entender el esquema o preocedimiento:

1. Se selecciona un átomo i.
2. Se busca un átomo j que se encuentre dentro del radio de corte r_{ij}
3. Se busca un átomo k que se enceuntre dentro del radio de corte r_{ik}
4. Se calcula el ángulo $\angle j-i-k$ y se añade en la cuenta

De esta forma obtenemos una función entre 0 y 180° que nos muestra cuales son los principales angulos de "enlace" o "distancia" de los átomos. Las opciones de uso son:

bins	=	Número de intervalos entre 0 y 180 grados.
atoms	=	Número de especies atómicas y los símbolos asociados a cada una.
rcut	=	Se especifican 2 especies atómicas y su radio de corte.
output	=	Archivo de salida.
average	=	Se promediarán o no los calculos.

4.6.2. atomtrail

Calcula la ... en 2 y 3 dimensiones.

1. Se ..
2. Se ..

De esta forma obtenemos una función entre 0 y 180° que nos muestra cuales son los principales angulos de "enlace" o "distancia" de los átomos. Las opciones de uso son:

nx	=	Número de intervalos en `x`.
ny	=	Número de intervalos en `y`.
nz	=	Número de intervalos en `z`.
plane	=	Plano XY,YZ, etc.
mode	=	2D o 3D.

4.6.3. cna

Calcula el número de cordinación de la celda, informandonos a modo de histograma, como se han encontrado los números de vecinos asociados a la muestra. La manera de calcular este numero de coordinacion es:

1. Se genera un arreglo entre 0 y el máximo número de vecinos posibles.
2. Para cada átomo i, se ve cuantos vecinos j tiene dentro de un radio d corte r_{ij}.
3. Se entrega un valor porcentual del conteo previo.

maxn	=	Número máximo d vecinos para el histograma.
atoms	=	Número de especies atómicas y los símbolos asociados a cada una.
rcut	=	Se especifican 2 especies atómicas y su radio de corte.
output	=	Archivo de salida.
average	=	Se promediarán o no los calculos.

4.6.4. cordnumfunc

Calcula el número de cordinación de la celda, en este caso es la función más usada n publicaciones, pero en ocaciones puede ser más simple de analizar, el método **cordnum**. Corresponde también a la integración de la función de distribución de pares.

1. Se selecciona una distancia máxima y se divide en intervalos.
2. Se selecciona un átomo i.
3. Se analiza cuantos atomos j caen en la distancia asociada al intervalo.
4. Se continua de forma acumulativa, hasta un valor rasonable.

La forma de utilizar el método esta dada por:

bins	=	Número máximo de intervalos entre 0 y **rcut**.
atoms	=	Número de especies atómicas y los símbolos asociados a cada una.
rcut	=	Radio de corte máximo para análisis.
output	=	Archivo de salida.
average	=	Se promediarán o no los calculos.

4.6.5. cordnum

Calcula el número de cordinación de la celda, en este caso es la función más usada n publicaciones, pero en ocaciones puede ser más simple de analizar, el método **cordnum**. Corresponde también a la integración de la función de distribución de pares.

1. Se selecciona una distancia máxima y se divide en intervalos.
2. Se selecciona un átomo i.
3. Se analiza cuantos atomos j caen en la distancia asociada al intervalo.
4. Se continua de forma acumulativa, hasta un valor rasonable.

La forma de utilizar el método esta dada por:

bins	=	Número máximo de intervalos entre 0 y **rcut**.
atoms	=	Número de especies atómicas y los símbolos asociados a cada una.
rcut	=	Radio de corte máximo para análisis.
output	=	Archivo de salida.
average	=	Se promediarán o no los calculos.

4.6.6. densityprofile

Calcúla un perfil de densidades bidimensional. Este método divide la celda de simulación en cajas pequenas, en una direccion privilegiada y calcula las densidades en cada una de ellas, da una idea muy clara de la densidad "por sección" de la celda de simulación.

La forma de utilizarla es:

axis	=	Valor del eje preferencial, o dirección, del cálculo.
bins	=	Número de intervalos para dividir el eje.
range	=	Rango espacial de cada uno de los ejes, con formato: nombre del eje, mínimo y máximo.
output	=	Archivo de salida.
average	=	Se promediarán o no los calculos.

4.6.7. gdr

Caulcula la funcion de distribucion de pares de la celda. Es uno de los métodos utilizados para determinar las distancias principales a primeros y segundos vecinos de una celda de simulación. El procedimiento es el siguiente:

1. Se selecciona un átomo i.
2. Se ve cuantos atomos j estan dentro de un cascarón esférico centrado en i.
3. Se promedia sobre los átomos asociados.

bins	=	Número máximo de intervalos entre 0 y **rcut**.
rcut	=	Radio de corte máximo para análisis.
output	=	Archivo de salida.
average	=	Se promediarán o no los calculos.

4.6.8. localpressure

Calcula una presion local de la celda de simulación, para eso utiliza el stress de los átomo en cada "sub-celda", los valores entregados, recomendamos graficarlos con escala de colores para poder observar fenómenos.

La forma de utilizar el plugin es:

rcut	=	Radio de corte.
nα	=	Divisiones para cada eje ($\alpha = x, y, z$).
output	=	Archivo de salida.
average	=	Se promediarán o no los calculos.

4.6.9. overlap

Calcula una presion local de la celda de simulación, para eso utiliza el stress de los átomo en cada "sub-celda", los valores entregados, recomendamos graficarlos con escala de colores para poder observar fenómenos.

La forma de utilizar el plugin es:

rcut	=	Radio de corte.
nα	=	Divisiones para cada eje ($\alpha = x, y, z$).
output	=	Archivo de salida.
average	=	Se promediarán o no los calculos.

4.6.10. pairdistances

Calcula una presion local de la celda de simulación, para eso utiliza el stress de los átomo en cada "sub-celda", los valores entregados, recomendamos graficarlos con escala de colores para poder observar fenómenos.

La forma de utilizar el plugin es:

rcut	=	Radio de corte.
nα	=	Divisiones para cada eje ($\alpha = x, y, z$).
output	=	Archivo de salida.
average	=	Se promediarán o no los calculos.

4.6.11. rvcorr

Calcula una presion local de la celda de simulación, para eso utiliza el stress de los átomo en cada "sub-celda", los valores entregados, recomendamos graficarlos con escala de colores para poder observar fenómenos.

La forma de utilizar el plugin es:

rcut	=	Radio de corte.
nα	=	Divisiones para cada eje ($\alpha = x, y, z$).
output	=	Archivo de salida.
average	=	Se promediarán o no los calculos.

4.6.12. sitecoord

Calcula una presion local de la celda de simulación, para eso utiliza el stress de los átomo en cada "sub-celda", los valores entregados, recomendamos graficarlos con escala de colores para poder observar fenómenos.

La forma de utilizar el plugin es:

rcut	=	Radio de corte.
nα	=	Divisiones para cada eje ($\alpha = x, y, z$).
output	=	Archivo de salida.
average	=	Se promediarán o no los calculos.

4.6.13. tempprofile

Calcúla un perfil de temperaturas bidimensional. Al igual que el método anterior, se divide la caja en celdas pequenas, en donde calculamos la temperatura de cada una de ellas.

La forma de utilizar esto es:

axis	=	Valor del eje preferencial, o dirección, del cálculo.
bins	=	Número de intervalos para dividir el eje.
range	=	Rango espacial de cada uno de los ejes, con formato: nombre del eje, mínimo y máximo.
output	=	Archivo de salida.
average	=	Se promediarán o no los calculos.

4.6.14. veldist

Muestra como estan distribuidas las velocidades del sistema. La salida es un histograma de velocidades. La forma de uso es:

bins	=	Número de intervalos para dividir el histograma.
output	=	Archivo de salida.
average	=	Se promediarán o no los calculos.

4.7. Propiedades Dinámicas

Las porpiedades dinámicas, **no** pueden ser calculadas durante la simulaciónd e dinámica molecular, ya que poseen correlación temporal en su análisis, es por ello que estos módulos no deben ser cargados en un fichero de control para `lpmd`. Sin embargo estas propiedades pueden calcularase a partir de los ficheros de salida, tales como `xyz` o `lpmd`, utilizando `lpmd-analyzer`.

4.7.1. dispvol

Calcula el desplazamiento cuadratico medio del sistema. Por el momento el plugin esta en etapa de mejora y documentación.

4.7.2. mobility

Calcula el desplazamiento cuadratico medio del sistema. Por el momento el plugin esta en etapa de mejora y documentación.

4.7.3. msd

Calcula el desplazamiento cuadratico medio del sistema. Por el momento el plugin esta en etapa de mejora y documentación.

4.7.4. vacf

Calcula la función de autocorrelación de velocidades de la celda. Por el momento el plugin esta en etapa de mejora y documentación.

4.8. Integradores

4.8.1. beeman

El algoritmo de Beeman es un método para integrar numéricamente ecuaciones difereciales, para el caso de dinámica molecular, la posición y velocidad,es similar al método de verlet, pero las velocidades son con mejor precisión.

Las fórmulas para posición y velocidad estan dadas por:

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

$$v(t+\Delta t) = v(t) + \frac{1}{3}a(t+\Delta t)\Delta t + \frac{5}{6}a(t)\Delta t - \frac{1}{6}a(t-\Delta t)\Delta t + \mathcal{O}(\Delta t^3)$$

4.8.2. euler

El integrador de Euler, es el mátodo númerico de primer orden para resolver ecuaciones diferenciales ordinarias. Este es el método más básico para la integración numérica de las ecuaciones.

Las fórmulas para posición esta dada por:

$$r(t+\Delta t) = r(t) + h \times f(t, r(t))$$

4.8.3. hardspheres

Es un método de integración que calcula las posiciones y velocidades de forma alternada. Las posiciones y velocidades estan dadas por:

4.8.4. leapfrog

Es un método de integración que calcula las posiciones y velocidades de forma alternada. Las posiciones y velocidades estan dadas por:

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

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

4.8.5. metropolis

Es un método de integración que calcula las posiciones y velocidades de forma alternada. Las posiciones y velocidades estan dadas por:

4.8.6. nosehoover

Es utilizado para simulacion de ensambles de tipo NVT, en general el método corresponde a una modificación de las ecuaciones de movimiento, donde aparece una masa ficticia en las ecuaciones a resolver.

4.8.7. nullintegrator

Integrador nulo, solo en caso de que no se requiera mover el sistema, es decir las particulas simplemente *congeladas*.

4.8.8. velocityverlet

Método similar al de verlet, pero con mejoras respecto a la performance en la integración, incorporando directamente la velocidad en el sistema, las ecuaciones estan dadas por:

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

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

4.8.9. verlet

Es un método utilizado para integrar las ecuaciones de Newton de movimiento, las posiciones y velocidades del sistema estan dadas por:

$$r(t+\Delta t) = 2r(t) - r(t-\Delta t) + a(t)\Delta t^2 + \mathcal{O}(\Delta t^4)$$

$$v(t+\Delta t) = \frac{r(t+\Delta t) - r(t)}{\Delta t} + \mathcal{O}(\Delta t)$$

4.9. Potenciales Interatómicos de pares

4.9.1. buckingham

Buckingham, no incluye directamente parte coulombiana, para ello es necesario añadir como un potencial adicional a ewald u otro similar que añada la parte coulombiana.

Esté módulo especifíca la interacción de buckingham entre los átomos, de esta forma, la energía producida por la interacción de dos partículas i y j, queda

$$E(\vec{r}_{ij}) = B1 \exp\left(-\frac{|\vec{r}_{ij}|}{\rho}\right) - \frac{B2}{(|\vec{r}_{ij}|)^6}$$

y la fuerza,

$$\vec{F}(\vec{r}_{ij}) = -\frac{B1 \exp\left(-\frac{|\vec{r}_{ij}|}{\rho}\right)}{|\vec{r}_{ij}|\rho}\vec{r}_{ij} + \frac{6B2}{|\vec{r}_{ij}|^8}\vec{r}_{ij}$$

Las palabras reservadas por el plugin **buckingham**, son :

B1	=	indica el valor de la constante B1.
B2	=	indica el valor de la constante B2.
Ro	=	el valor de rho para el potencial.
cutoff	=	indica el cutoff del potencial.

4.9.2. constantforce

Aplica una fuerza constante sobre cada átomo. Si m es la masa del átomo, la aceleración del átomo provocada por el potencial interatómico tendrá un término adicional $\vec{F}/m$. Todos los átomos seleccionados serán afectados por la misma fuerza, pero no acelerarán igual si poseen distinta masa.

Se utiliza principalmente para aplicarles fuerzas a especies atómicas o bien a atomos seleccionados de alguna forma en especial. Este *potencial*, no retorna una energía (cero) y sólo tiene capacidad de asignar una fuerza constante a un set de atomos.

Cargando el Módulo :

```
use constantforce as gravity
force <0.0,0.0,-4.06E-16>
enduse
```

Llamando al módulo :

```
potential gravity Ar Ar
```

En este ejemplo agregamos la aceleración de gravedad $g = -9.8\left[\frac{m}{s^2}\right]$ a los átomos de argón, de masa $m = 39.948[amu]$, para lo cual se necesita una fuerza de $F = m * a = -391.4904\left[\frac{amu*m}{s^2}\right] = -4.06 \times 10^{-16}\left[\frac{eV}{\text{Å}}\right]$ (el usuario es responsable de discernir si los ordenes de magnitud son razonables).

La fuerza debe ser ingresada en unidades de $\left[\frac{eV}{\text{Å}}\right]$ ($E = F \cdot d \Rightarrow F = E/d$), la cual es convertida internamente por `lpmd` a las unidades naturales del programa, $\left[\frac{amu*\text{Å}}{fs^2}\right]$. Esto permite dividir la fuerza ingresada por la masa del elemento (en este caso, argón), la cual está en unidades de masa atómica (amu), quedando un factor con unidades de aceleración medida en $\left[\frac{\text{Å}}{fs^2}\right]$, la cual es adicionada a la aceleración producida por el potencial interatómico.

Las palabras reservadas por el plugin **constantforce**, son :

forcevector	=	indica de la fuerza constante a aplicarse.

4.9.3. harmonic

Potencial armónico entre especies atómicas. De manera similar a un potencial de morse, tenemos que la energía que sienten las partículas i y j a causa de la interacción a través de éste potencial es,

$$E(\vec{r}_{ij}) = \frac{1}{2}k\left(|\vec{r}_{ij}| - a\right)^2$$

En donde k es la constante de elasticidad y a la separación de equilibrio. Con esto la fuerza para el potencial armónico esta dada por,

$$\vec{F}(\vec{r}_{ij}) = \frac{k}{|\vec{r}_{ij}|}\left(|\vec{r}_{ij}| - a\right)$$

Las palabras reservadas por el plugin **harmonic**, son :

k	=	indica el valor de la constante de elasticidad.
a	=	indica el valor de el largo de equilibrio.
cutoff	=	indica el cutoff del potencial.

4.9.4. lennardjones

El módulo **lennardjones** hace referencia al potencial de Lennard-Jones, que es de la forma,

$$U(r_{ij}) = 4\epsilon\left(\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right)$$

En donde r_{ij} es la distancia interatómica de los átomos i y j.

Para el cálculo de Fuerzas, la forma del potencial que nos interesa, es aquella fuerza que siente el átomo i producida por el átomo j, la que debe ser implementada en el plugin, para potentiales de pares, para el caso del potencial de Lennard Jones, la fuerza esta dada por,

$$F_{ij} = \frac{-48.0\epsilon}{r_{ij}^2}\left(\left(\frac{\sigma}{r_{ij}}\right)^{12} + \frac{1}{2}\left(\frac{\sigma}{r_{ij}}\right)^{6}\right)\vec{r_{ij}}$$

en donde $\vec{r_{ij}}$ es el vector distancia entre los átomos i y j, y r_{ij} es la distancia entre ellos. Las palabras reservadas por el plugin **lennardjones**, son :

epsilon	=	indica el valor de epsilon.
sigma	=	indica el valor de sigma.
cutoff	=	indica el cutoff del potencial.

Las unidades en que deben ser ingresados, las constantes, deben ser basadas en que las distancias estan en [Å] y la energía debe ser adquirida en [eV].

4.9.5. morse

Utiliza Potencial de Morse para la interacción de las especies atómicas. La energía que siente una partícula i a causa de la precensia de otra partícula j, si ambas interactuan con un potencial de este estilo, esta dada por :

$$E(\vec{r}_{ij}) = D_e \left(1 - \exp(-a(\vec{r}_{ij} - \vec{r}_e))\right)^2$$

en donde D_e es la profundidad del pozo, a es el ancho del pozi y r_e es la distancia en equilibrio. Y entonces, la fuerza que siente un átomo i producto de otro átomo j está dada por,

$$\vec{F}_{ij}(\vec{r}_{ij}) = 2aD_e \exp(-a(\vec{r}_{ij} - \vec{r}_e)) \left(1 - \exp(-a(\vec{r}_{ij} - \vec{r}_e))\right) \frac{\vec{r}}{|\vec{r}|}$$

Las palabras reservadas por el plugin **morse**, son :

depth	=	indica el valor de profundidad del pozo.
a	=	indica el valor del ancho del pozo.
re	=	indica el largo de enlaze en equilibrio.
cutoff	=	indica el cutoff del potencial.

4.9.6. nullpairpotential

Utiliza Potencial de LJ tabulado. De forma similar al módulo anterior pero porcentualmente más rápido. Es recomendable utilizarlo para sistemas con grn número de partículas.

4.9.7. tabulatedpair

Utiliza Potencial de LJ tabulado. De forma similar al módulo anterior pero porcentualmente más rápido. Es recomendable utilizarlo para sistemas con grn número de partículas.

4.10. Potenciales Interatómicos Metálicos

4.10.1. finnissinclair

4.10.2. Gupta

Al igual que `suttonchen`, el potencial de `gupta` es utilizado para átomos metálicos, donde los valores para los terminos de pares está dada por:

$$U(r_{ij}) = A \exp\left(-p\frac{r_{ij} - r_0}{r_0}\right)$$

en donde r_ij es la distancia entre un atomo i y otro atomo j del sistema. El término de muchos cuerpos esta dado por

$$F(\rho_i) = -B\sqrt{\rho_i}$$

en donde,

$$\rho_i = \sum_{j \neq i} \exp\left(-2q_{ij}\frac{r_{ij} - r_0}{r_0}\right)$$

lo que corresponde a una densidad local del atomo i, que depende de todos los atomos j cercanos a él. Para el potencial de Gupta, la corrección de la densidad esta dada por,

$$\delta\rho_i = \frac{2\pi\overline{\rho}r_0}{q_{ij}} \left[r_{met}^2 + 2r_{met}\left(\frac{r_0}{q_{ij}}\right) + 2\left(\frac{r_0}{q_{ij}}\right)^2\right] \exp\left(-2q_{ij}\frac{r_{met} - r_0}{r_0}\right)$$

Esta corrección de la densidad debe ser aplicada inmediatamente luego de ser calculada la densidad local. La corrección de la energía para Gupta, se obtiene de esta forma con :

$$\delta U_1 = \frac{2\pi N\overline{\rho}Ar_0}{p} \left[r_{met}^2 + 2r_{met}\left(\frac{r_0}{p}\right) + 2\left(\frac{r_0}{p}\right)^2\right] \exp\left(-p\frac{r_{met} - r_0}{r_0}\right)$$

Hay que notar que δU_2 no es requerido si ρ_i ya fue corregido, con δU_2 de la forma

$$\delta U_2 = \frac{2\pi\overline{\rho}r_0}{q_{ij}}\left[r_{met}^2 + 2r_{met}\left(\frac{r_0}{q_{ij}}\right) + 2\left(\frac{r_0}{q_{ij}}\right)^2\right]\exp\left(-2q_{ij}\frac{r_{met}-r_0}{r_0}\right)\left\langle\frac{NB}{2\sqrt{\rho_i^0}}\right\rangle$$

4.10.3. nullmetalpotential

4.10.4. suttonchen

Este potencial, se utiliza para interacciones de atomos metálicos, es por eso que el plugin **suttonchen** implementa los métodos virtuales de `metalpotential`, que cuentan con una parte de pares y otro término de muchos cuerpos. La parte asociada al término de pares, está dado por,

$$U(r_{ij}) = \epsilon\left(\frac{a}{r_{ij}}\right)^n$$

en donde r_ij es la distancia entre un atomo i y otro atomo j del sistema. El término de muchos cuerpos esta dado por

$$F(\rho_i) = -c\epsilon\sqrt{\rho_i}$$

en donde,

$$\rho_i = \sum_{j\neq i}\left(\frac{a}{r_{ij}}\right)^m$$

lo que corresponde a una densidad local del atomo i, que depende de todos los atomos j cercanos a él, ésta densidad local sin embargo, debe ser corregida para el caso de suttonchen (note que no todos los potenciales asociados a los metales requieren de esta corrección, pero `metalpotential` lo requiere, así que en ocaciones debe ser cero).

Para el potencial de SuttonChen, la corrección de la densidad esta dada por,

$$\delta\rho_i = \frac{4\pi\overline{\rho}a^3}{m-3}\left(\frac{a}{r_{met}}\right)^{(m-3)}$$

Esta corrección de la densidad debe ser aplicada inmediatamente luego de ser calculada la densidad local. La corrección de la energía para Sutton Chen, se obtiene de esta forma con :

$$\delta U_1 = \frac{2\pi N\overline{\rho}\epsilon a^3}{n-3}\left(\frac{a}{r_{met}}\right)^{n-3}$$

Hay que notar que δU_2 no es requerido si ρ_i ya fue corregido, con δU_2 de la forma

$$\delta U_2 = -\frac{4\pi\overline{\rho}a^3}{m-3}\left(\frac{a}{r_{met}}\right)^{n-3}\left\langle\frac{Nc\epsilon}{2\sqrt{\rho_i^0}}\right\rangle$$

y la fuerza asociada al potencial de suttonchen que aplica para un par de atomos i y j está dada por,

$$\vec{F}(\vec{r}_{ij}) = -\epsilon\left[n\left(\frac{a}{\vec{r}_{ij}}\right)^n - \frac{Cm}{2}(\rho_j^{(-1/2)} + \rho_i^{(-1/2)})\left(\frac{a}{\vec{r}_{ij}}\right)^m\right]\left(\frac{1}{\vec{r}_{ij}^2}\right)\vec{r}_{ij}$$

4.11. Visualizadores

Los visualizadores ordinarios antes de pasar a `lpvisual`, el gran visualizador, son:

4.11.1. average

4.11.2. monitor

4.11.3. printatoms

4.11.4. lpvisual

LPVisual es el visualizador de dinámica molecular propio de `lpmd`, diseñado en nuestro *Grupo de Nanomateriales*. Se puede utilizar para ver configuraciones tanto estáticas como dinámicas. Lee archivos tipo `xyz`, `lpmd`, etc (todos los formatos descritos en las tablas A.1 y A.2). Su ejecución es posible a través de la línea de comandos (`lpmd-visualizer`, sec. 5.3) así como de los ficheros de control, vistos en la sección 3). El uso detallado de `lpvisual` se encuentra descrito en el capítulo 6.

5. Utilidades Derivadas de `lpmd`

A continuación veremos algunos utilitarios derivados de la implementación de `lpmd`, estos surgen gracias a la *reutilización* de los códigos, para no sólo realizar dinámica molecular, sino que análisis, conversiones, visualizaciones, etc.

En este capítulo usted encontrará una descripción general de cada uno de los utilitarios, sin embargo podrá encontrar ejemplos mucho más específicos en el capítulo 7. O también en la página web oficial de `lpmd`.

5.1. lpmd-analyzer

Al igual que `lpmd`, `lpmd-analyzer` puede utilizar un fichero de control o bien directamente línea de comandos, basado en los mismos plugins, pero a diferencia de `lpmd`, **lpmd-analyzer** no necesita todos los requerimientos de una dinámica molecular, por lo que su fichero de control es más corto, éste sólo requiere las propiedades que se desean calcular, asi como también la especificación del sistema y como se llevará a cabo la evaluación.

Los análisis son hechos generalmente, sobre ficheros de salida de simulaciones computacionales, tales como `xyz`, `lpmd`, `dlpoly` (ficheros `HISTORY` o `CONFIG`) o cualquiera de los listados en la tabla A.1.

5.1.1. Análisis utilizando un Fichero de Control

Ahora listamos cada una de las áreas principales de un fichero de control para ejcutarlo con **lpmd-analyzer**.

1. Propiedades de la Celda.
2. Carga de módulos de análisis.
3. Forma de análisis, o llamado a módulos.

Propiedades de la Celda

Se especifican las características de la celda de simulación del fichero que posee las configuraciones atómicas. Pueden ser en ocaciones entregados como argumentos de ejecución del comando **lpmd-analyzer**, como veremos en el **quick-mode**.

Los principales campos que deberían estar en esta sección del fichero son:

- cell : Características generales de la celda.
- input : Para cargar el fichero de entrada.
- prepare : Alguna modificación a la celda original, tales como las replicas.

Carga de módulos

En esta sección del fichero especificamos las propiedades que deseamos calcular, pueden ser tanto estáticas como dinámicas, podemos calcular multiples propiedades en una sola ejecución, asi como tambien una misma propiedad con distintos parámetros.

En este caso, lo importante es cargar cada módulo evaluador de propiedades con:

```
use module as alias
...
enduse
```

Llamado a módulos

Al igual que en `lpmd`, el llamado a módulos de propiedades, se realiza con la orden:

```
property alias start=0 end=1000 each=10
```

Por ejemplo, en este caso, un fichero con mas de 1000 configuraciones, es caracterizado en sus primeras 1000 configuraciones y la evaluación de esta propiedad se realiza cada 10 pasos.

Veamos a continuación una idea general de un fichero de control de `lpmd-analyzer`, más ejemplos sobre su uso se pueden encontrar ena la sección 7.

```
#Cell Properties   →  Propiedades de la celda.
cell ...
#Input             →  Principal fichero de entrada
input ...
#General           →  Setting generales sobre el fichero, replicar, etc.
prepare ...
#Filters           →  Filtros a aplicar sobre la muestra.
filter ...
#Module Load       →  Carga de todos los modulos de propiedades a analisar.
use ...
enduse ...
#Module Apply      →  Indica como son aplicados los modulos.
properties ...
```

5.1.2. Análisis utilizando el quick-mode

Usualmente uno espera que las utilidades sean de *fácil acceso* por lo que escribir tediosos ficheros de configuración no es lo esperado, por eso tanto `lpmd` como su set de utilidades tienen el llamado **quick-mode** que és simplementa la ejecución directa de análisis, conversiones o visualizaciones de salidas de dinámica molecular.

Veamos a continuación lo que hay en una ejecución simple de `lpmd-analyzer`.

```
username@machine:~$lpmd-analyzer

LPMD Analyzer, version 0.6.1pre

LPMD Analyzer version 0.6.1pre
Using liblpmd version 2.0.0

Usage: lpmd-analyzer [--verbose | -v ] [--lengths | -L <a,b,c>]
       [--angles | -A <alpha,beta,gamma>] [--vector | -V <ax,ay,az,bx,by,bz,cx,cy,cz>]
       [--scale | -S <value>] [--option | -O <option=value,option=value,...>]
       [--input | -i plugin:opt1,opt2,...] [--output | -o plugin:opt1,opt2,...]
       [--use | -u plugin:opt1,opt2,...] [--cellmanager | -c plugin:opt1,opt2,...]
       [--replace-cell | -r] [file.control]
       lpmd-analyzer [--pluginhelp | -p <pluginname>]
       lpmd-analyzer [--help | -h]

username@machine:~$
```

Podemos ver entonces la gran cantidad de opciones con que dispone `lpmd-analyzer`. Es decir, *no tenemos necesidad* de crear ficheros de control para realizar análisis sobre el sistema.

Consideremos que tenemos un fichero `HISTORY` de una simulación realizada con DL_POLY y deseamos evaluar por ejemplo el numero de coordinación de los átomos de la simulación y deseamos usar para eso el plugin `cordnum`. Podemos ver todo lo que **requiere el plugin** utilizando el comando `lpmd -p cordnum`.

```
username@machine:~$lpmd-analyzer -p gdr
......
General Options   >>
     bins          : Especifica el numero de divisiones entre 0 y rcut.
     rcut          : Especifica el radio maximo para el calculo de gdr.
```

```
     output        : Fichero en el que se graba el RDF.
     average       : Setea si calculo o no el promedio de cada calculo.
.........
```

Entonces por ejemplo podemos calcular directamene el número de coordinación sin la necesidad de un fichero de control utilizando :

```
lpmd-analyzer -i dlpoly:file=HISTORY,ftype=HISTORY -u gdr:output=gdr.dat//
,bins=100,rcut=10,average=true,start=1,end=-1,each=5 -r
```

Vemos entonces un calculo de la *Función de distribución de pares* para un fichero `HISTORY` directamente en línea de comandos, note que también es posible indicar a que configuraciones se les puede calcular.

5.2. lpmd-converter

Utlizado para conversión de celdas entre distintos formatos, pese a que ya existen software que convierten entre distintos formatos de manera fácil y eficiente (ej: `babel`), `lpmd-converter` cuenta con algunas funcionalidades extras que son independientes de cada tipo de módulo. Entre sus ventajas destacan el hecho de hacer filtrados, asignar colores a los átomos seleccionar o invertir selecciones, replicar la celda, generar configuraciones complejas de forma fácil, etc.

Al igual que todos los utilitarios de `lpmd`, `lpmd-converter` trabaja con dos modos principales, utilizando ficheros de control o bien en línea de comandos. Veremos a continuación a grosso modo cada una de ellas.

5.2.1. lpmd-converter utilizando un Fichero de Control

Las áreas principales de un fichero de control para ejecutar con **lpmd-converter**, son:

1. Propiedades de la Celda.
2. Carga de fichero de entrada.
3. Filtros o características especiales
4. Fichero de salida.

Propiedades de la Celda

Se especifican las características de la celda de simulación del fichero que posee las configuraciones atómicas. en ocaciones esta información ya viene en el archivo de entrada principal por lo que no es necesario entregarla.

Los principales campos que deberían estar en esta sección del fichero son:

- cell : Características generales de la celda.

Carga de fichero de entrada

Especificamos claramente el fichero inicial o de entrada, el que corresponde al que queremos *convertir* a un formato distinto, en esta sección la orden más importante es:

```
input module=...  file=...
```

Filtros o Características especiales

En general casi cualquier característica puede ser asignada a la celda, siempre y cuanto esta tengo algún sentido en la aplicación, por ejemplo podemos hacer uso de replicar la celda en distintas direcciones. Además podemos usar los filtros disponibles para seleccionar regiones especiales del espacio que podemos necesitar.

```
prepare replicate 2 2 2

filter sphere radius=5 center=<10,10,10>
```

Con eso entonces replicamos la celda y luego seleccionamos una esfera centrada en `<10,10,10>` con radio de 5, es decir todos los otros átomos del sistema son eliminados.

Ficheros de salida

En esta sección podemos especificar uno o más ficheros de salida para realizar la transformación de nuestra celda inicial. La orden, de modo similar a `lpmd`, es dada por la palabra clave `output`.

```
output module=...  file=...
```

5.2.2. Utilizando lpmd-converter con quick-mode

Por ahora, **lpmd-converter** puede manejar todos los formatos listados en la tabla A.1. Además se pueden aplicar filtros sobre las configuraciones, para modificar, extraer o añadir átomos en el sistema. Veamos una salida típica de `lpmd-converter`,

```
username@machine:~$lpmd-converter
 LPMD Converter, version 0.6.1pre

LPMD Converter version 0.6.1pre
Using liblpmd version 2.0.0

Usage: lpmd-converter [--verbose | -v ] [--lengths | -L <a,b,c>] [--angles | -A <alpha,beta,gamma>] [--ve
       lpmd-converter [--pluginhelp | -p <pluginname>]
       lpmd-converter [--help | -h]
username@machine:~$
```

Vemos entonces que la mayoría de las opciones son similares todas las utilidades de lpmd, consideremos un par de ejemplos sencillos, por ejemplo convertir rápidamente, sin la necesidad de escribir un archivo de control, un fichero lpmd a xyz:

```
lpmd-converter -i lpmd:file=fichero.lpmd -o xyz:file=new.xyz -r
```

listo, ya hemos convertido el fichero `lpmd` a un fichero de tipo `xyz`.

Además podemos replicar una celda por ejemplo con :

```
lpmd-converter -i xyz:file=original.xyz -o xyz:new.xyz -L 15,15,15 -u replicate:nx=2,ny=2,nz=2
```

Note que el tamao de la celda debe ser entregado en el comando, ya que los ficheros de tipo `xyz` no guardan información sobre la estructura de la celda. Más ejemplos de cómo utilizar `lpmd-converter` se pueden encontrar en el capítulo 7.

5.3. lpmd-visualizer

Utilidad capaz de visualizar sistemas atómicos, su función es aplicar cualquier tipo de módulo de visualización del sistema. Por ejemplo visualización de las posiciones atómicas en la terminal (`printatoms`), así como la observación directa de configuraciones atómicas mediante openGL (`lpvisual`).

Es importante **remarcar** la diferencia entre `lpmd-visualizer` y `lpvisual`, en general `lpmd-visualizer` es una utilidad capaz de utilizar cualquier *plugin visualizador* y no necesariamente estos implican imágenes del sistema configuracional, también puede ser información `textual` sobre las configuraciones. Sin embargo en la mayoría de los casos, la intención principal de `lpmd-visualizer` está directamente relacionada al plugin `lpvisual` que tiene un gran soporte y detalle de visualización en OpenGL.

5.3.1. Utilizando lpmd-visualizer con un Fichero de Control

Las áreas principales de un fichero de control para ejcutar con **lpmd-converter**, son:

1. Propiedades de la Celda.
2. Carga de modulos de visualización.
3. Llamado de los módulos.

Propiedades de la Celda

Se especifican las características de la celda de simulación del fichero que posee las configuraciones atómicas.
Los principales campos que deberían estar en esta sección del fichero son:

- cell : Características generales de la celda.
- input : Especificando el fichero de entrada.
- prepare : Alguna modificación a la celda original, tales como las replicas o filtros.

Carga de módulo de visualización

En esta sección podemos cargar l módulo de visualizaión que se desee utilizar, generalmente es lpvisual, pero como ya se mencionó puede ser cualquier módulo visualizador.

```
use lpvisual as module-alias
...
enduse
```

Llamado al módulo

Especificamente llamamos al módulo y vemos como deseamos mostrar la imagen o generar información:

```
visualize module-alias start=0 end=1000 each=20
```

Por ejemplo, en este caso, un fichero con mas de 1000 configuraciones, se muestra cada 20 pasos comenzando en el paso inicial y terminando en el paso 1000.

5.3.2. Utilizando lpmd-visualizer con quick-mode

Poder generar ficheros para visualizaciónes grandes y complejas, asi como tambien una visualización simple y directa. Una de las primeras utilidades que nos brindo `lpmd-visualizer` es poder observar (apoyados por el módulo `lpvisual`) los ficheros de tipo `lpmd` o `zlp`. Veamos ahora como observar un fichero de estos de manera rápida utilziando el `quick-mode`.

El llamado sin argumentos del comando `lpmd-visualizer` nos da ayuda general de las opciones que éste permite :

```
username@machine:~$lpmd-visualizer
 LPMD Visualizer, version 0.6.1pre

LPMD Visualizer version 0.6.1pre
Using liblpmd version 2.0.0

Usage: lpmd-visualizer [--verbose | -v ] [--lengths | -L <a,b,c>]
                       [--angles | -A <alpha,beta,gamma>]
                       [--vector | -V <ax,ay,az,bx,by,bz,cx,cy,cz>] [--scale | -S <value>]
                       [--option | -O <option=value,option=value,...>]
                       [--input | -i plugin:opt1,opt2,...] [--output | -o plugin:opt1,opt2,...]
                       [--use | -u plugin:opt1,opt2,...]
                       [--cellmanager | -c plugin:opt1,opt2,...]
                       [--replace-cell | -r] [file.control]
       lpmd-visualizer [--pluginhelp | -p <pluginname>]
       lpmd-visualizer [--help | -h]
username@machine:~$lpmd-visualizer
```

Entonces podemos visualizar una configuración en formato `lpmd` utilizando el comando

```
lpmd-visualizer -i lpmd:file=fichero.lpmd -u lpvisual -r
```

Para más ejemplos del uso de `lpmd-visualizer` refierase al capítulo 7.

6. LPVisual

En éste capítulo se dará una descripción detallada del visualizador `lpvisual`, el visualizador de `lpmd`, mostrando las opciones disponibles y la forma en que este visualizador trabaja (posicionamiento de la cámara, luces, perspectivas, etc.). Puede encontrar ejemplos en el capítulo 7.

6.1. Descripción General

LPVisual es un proyecto independiente, desarrollado por Felipe González y Sergio Davis. Consiste en un visualizador basado en OpenGL que permite observar las configuraciones atómicas en tiempo real, así como monitorear y graficar propiedades de la configuración a medida que esta evoluciona.

La descarga de **lpvisual**, se puede realizar a través de `subversion`, con:

```
svn co sn://www.gnm.cl/lpmd/lpvisual lpvisual
```

Para la instalación y configuración refierase directamente a la documentación del módulo. Cualquier duda o consulta respecto al módulo, contacte directamente a los autores.

En la figura 6.1 podemos apreciar las principales características del visualizador:

- En la esquina superior derecha, la ventana principal de simulación. Aquí se pueden ver los átomos en movimiento encerrados en la caja de visualización, la cual puede o no tener condiciones de borde periódicas. En la esquina superior derecha de esta pantalla se encuentra el controlador de zoom y de rotaciones, el cual puede ser manejado mediante los click del mouse. Por último, los ejes de coordenadas se muestran en una esquina de la caja, los cuales evitan perder el punto de referencia cuando la cámara rote.
- En la esquina superior izquierda encontramos el graficador, el cual acepta interactivamente, al cargar la simulación, las propiedades que el usuario desee graficar (temperatura, presión, energía potencia, energía cinética, energía total, etc.).
- En la esquina inferior derecha, se encuentra la ventana de datos de simulación. Cuando el programa es cargado, el usuario puede elegir qué propiedades desea que sean monitoreadas numéricamente en esta ventana. Esto es independiente de las propiedades que estén siendo monitoreadas en el archivo de control (ver cap. 3).
- Por último, en la esquina inferior izquierda, se encuentra la terminal de texto desde donde se ejecutó el programa.

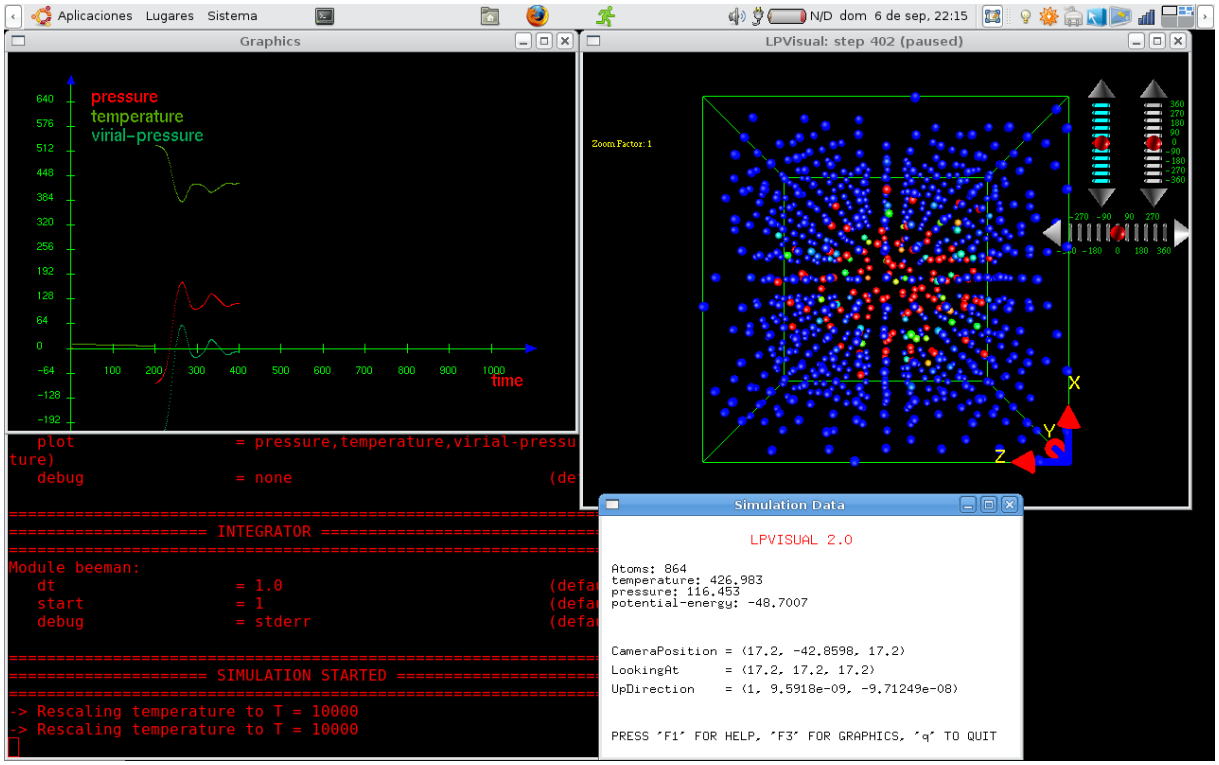


Figure 6.1.: Captura de pantalla mientras se ejecuta una simulación de dinámica molecular con `lpvisual`.

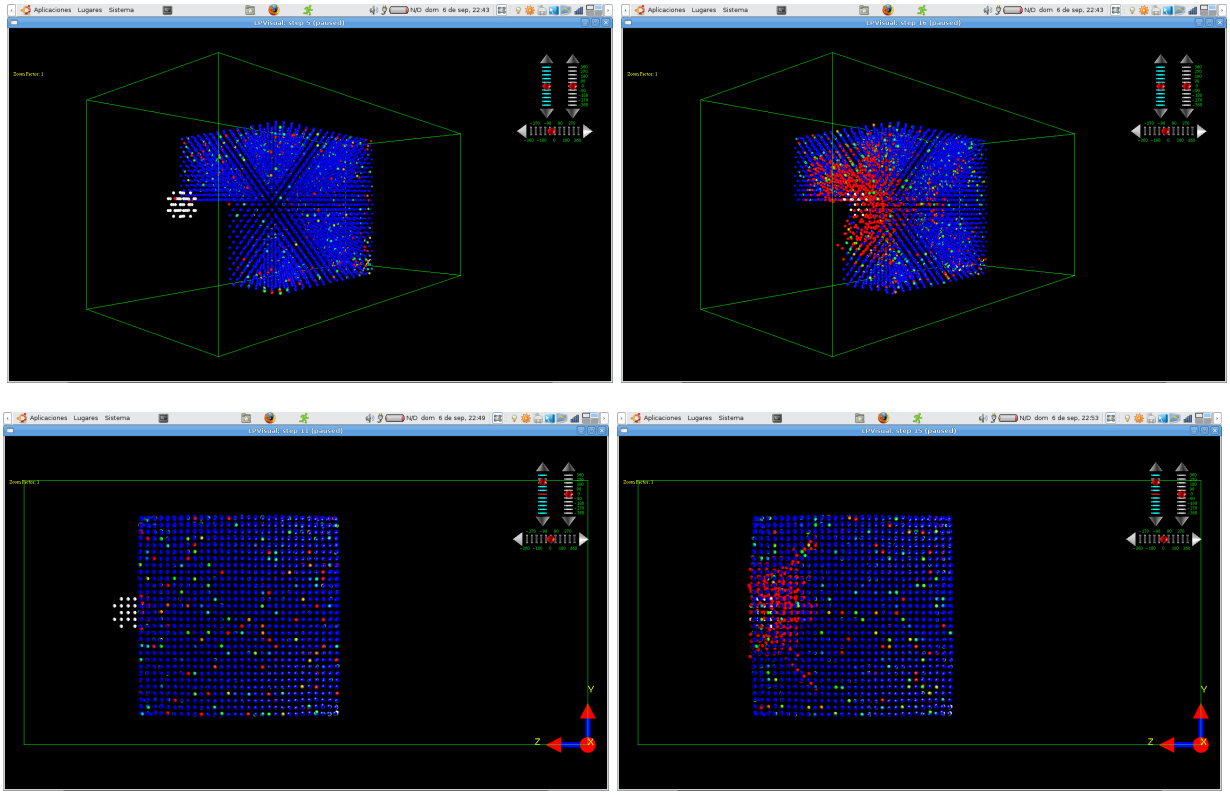


Figure 6.2.: Impacto de un proyectil sobre una celda fcc de 9000 atomos de cobre.

Opciones lpvisual

lpvisual ofrece varias opciones durante la visualización que son ingresandas tanto del mouse como del teclado:

Opciones del Teclado		
a	:	Esconde/muestra los ejes de coordenadas.
c	:	Esconde/muestra los bordes de la celda de simulación.
f	:	Pantalla completa.
F	:	Escala la ventana a su tamaño original.
i	:	Esconde/muestra los controladores zoom y rotaciones.
l	:	Cambia los bordes de la celda de simulación entre placas y espiras.
o	:	Selección de perspectiva (ortográfica / perspectiva).
p	:	Pausa.
q	:	Salir.
r	:	Reset: Deshace rotaciones, movimientos, zoom (restaura la escena).
s	:	Activa/desactiva la selección de átomos.
t	:	Activa/desactiva la rotación automática.
z	:	Comienza/detiene el acercamiento.
Z	:	Comienza/detiene el alejamiento.
+	:	En el modo de selección de átomos, va al átomo siguiente.
-	:	En el modo de selección de átomos, va al átomo anterior.
1->9	:	En el modo de selección de átomos, va al átomo tipeado (p. ej. “38”).
CTRL+	:	Aumenta el factor de zoom (el acercamiento es más rápido).
CTRL+-	:	Disminuye el factor de zoom (el acercamiento es más lento).
↑, ↓	:	Realizan rotaciones verticales.
←, →	:	Realizan rotaciones horizontales.
SHIFT+↑, SHIFT+↓	:	Realizan rotaciones verticales en 5^o.
SHIFT+←, SHIFT+→	:	Realizan rotaciones horizontales en 5^o.
CTRL+↑, CTRL+↓	:	Realizan traslaciones verticales.
CTRL+←, CTRL+→	:	Realizan traslaciones horizontales.
F1	:	Abre esta ventana.
F2	:	Abre la ventana de datos de simulación.
F3	:	Abre la ventana de gráficos.
RePag	:	Cambia la fuente de las letras a la próxima disponible.
AvPag	:	Cambia la fuente de las letras a la anterior disponible.
Opciones del Mouse		
Click en las flechas horizontales	:	Rota lentamente la escena horizontalmente.
Click en las flechas verticales de la esquina	:	Rota lentamente la escena verticalmente.
Click en las flechas verticales a la izquierda	:	La camara se acerca o se aleja de la escena.
Opciones de Movimiento del Mouse		
LEFT_MOUSE_BUTTON	:	Al presionar esta tecla y mover el mouse simultáneamente, permite rotar la celda de simulación.
RIGHT_MOUSE_BUTTON	:	Al presionar esta tecla, se despliega el menú de la ventana activa.
MIDDLE_MOUSE_BUTTON	:	Al presionar esta tecla y mover el mouse simultáneamente, permite acercar (hacer *zoom* a) la celda de simulación.

7. Ejemplos.

Acá encontrará algunos ejemplos de simulaciones realizadas con lpmd, en su última versión estable.
Todos los ejemplos se pueden descargar desde:

http://www.gnm.cl/lpmd/pmwiki.php?n=Manual.Examples

7.1. Ejemplos con LPMD

Existen algunas líneas dentro de cada ejemplo que al final tienen el símbolo \, lo cual significa que la línea no ha finalizado, sino que continúa en la siguiente línea. Sin embargo, no es necesario corregirlo ya que `lpmd` identifica el símbolo \ y comprende que la línea continúa en la siguiente.

7.1.1. Ejemplo1: Celda de Ar de 108 átomos.

A continuación una simulación de Ar con 108 átomos, en este ejemplo no hay escalamientos de temperatura, el sistema es inicializado con una temperatura de 84K para luego dejarlo libre, es decir, sin modificar el sistema.
Veamos el fichero de control.

```
# System file of Ar gas
# using LPMD
#
####################
#CELL and IN/OUT###
####################
cell crystal a=17.1191 b=17.1191 \
     c=17.1191 alpha=90.0 \
     beta=90.0 gamma=90.0

input module=lpmd file=Ar108.lpmd
output module=xyz file=output.xyz \
     each=20 level=0
####################
#GENERAL############
####################
prepare temperature t=84
steps 5000
periodic true true true
monitor start=0 end=5000 each=10 \
 properties=step,kinetic-energy, \
 potential-energy,total-energy,temperature, \
 pressure,volume output=monitor.dat
```

```
####################
#MODULES DEF#######
####################
use lennardjones as lj_Ar
    sigma 3.41
    epsilon 0.0103408
    cutoff 8.5
enduse

use beeman
    dt 10.0
enduse

use minimumimage
    cutoff 8.5
enduse
####################
#MOD APPLICATION###
####################
potential lj_Ar Ar Ar
integrator beeman
cellmanager minimumimage
```

Para correr la simulación, utilizamos solamente como argumento el fichero de control `argon108.control`.

```
lpmd argon108.control > salida.out
```

Podemos entonces ver algunos resultados de la simulación, por ejemplo la conservación de la energía a partir del fichero `monitor.dat` que fue generado. Como se observa en la figura 7.1. En un equipo moderno, la simulación no debería tardar más allá de 40 segundos, se pueden observar los detalles de las cargas de los módulos, asi como también toda la información de la simulación realizada en el archivo `salida.out`. Junto con la finalización de la simulación, se han generado los siguientes ficheros :

monitor.dat	:	Guarda toda la información de monitoreo solicitada por la orden `monitor` del fichero de control.
output.xyz	:	Salida de las posiciones atómicas de la celda de simulación.

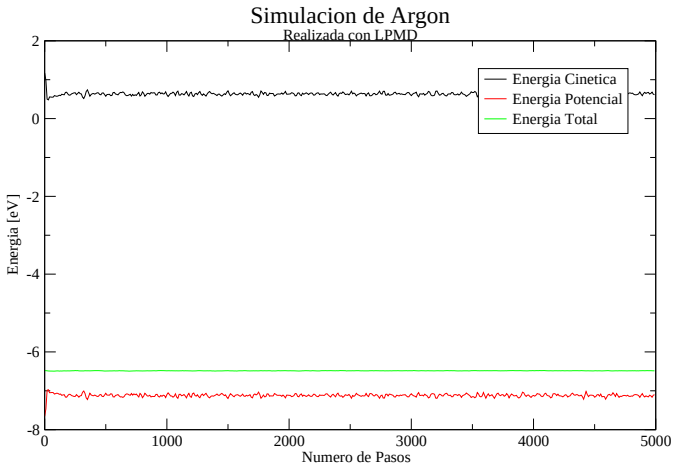


Figure 7.1.: Valores de la energía para la simulación de argón con 108 átomos.

7.1.2. Ejemplo2: Escalamiento de Temperatura.

A continuación veremos un ejemplo de escalamiento de temperatura clásico, éste consistirá en calentar de manera directa una muestra de Ar.

El fichero de control incorpora la carga del módulo tempscaling, el cual será aplicado dos veces, primero para mantener la muestra a una temperatura de 5K y finalmente para escalar la temperatura de la muestra desde 5K hasta los 150K.

```
# System file of Ar gas
# using LPMD
#
####################
#CELL and IN/OUT###
####################
cell crystal a=17.1191 b=17.1191 \
     c=17.1191 alpha=90.0 \
     beta=90.0 gamma=90.0

input module=lpmd file=Ar108.lpmd
output module=xyz file=output.xyz \
     each=20 level=0
####################
#GENERAL############
####################
prepare temperature t=5
steps 15000
periodic true true true

monitor start=0 end=15000 each=10 \
  properties=step,kinetic-energy, \
  potential-energy,total-energy, \
  temperature,pressure \
  output=monitor.dat
####################
#MODULES DEF########
####################
use lennardjones as lj_Ar
    sigma 3.41
    epsilon 0.0103408
    cutoff 8.5
enduse

use beeman
    dt 10.0
enduse

use minimumimage
    cutoff 8.5
enduse

use tempscaling as t1
    from 5
    to 5
enduse

use tempscaling as t2
    from 5
    to 150
enduse
####################
#MOD APPLICATION###
####################
potential lj_Ar Ar Ar
integrator beeman
cellmanager minimumimage
apply t1 start=1 end=5000 each=200
apply t2 start=5001 end=10000 each=200
```

La aplicación del escalamiento de temperatura genera un diseño automático para la mantención y luego el aumento de la temperatura del sistema, en este caso primero se mantiene la temperatura a 5K desde el paso 1 hasta el 5000, luego se realiza el escalamiento de forma lineal y automática, partiendo en el paso 5001 a 5K y finalizando en el paso 10000 con una temperatura de 150K, donde la **inyección** de energía se aplica cada 200 pasos. Como se puede apreciar en la figura 7.2

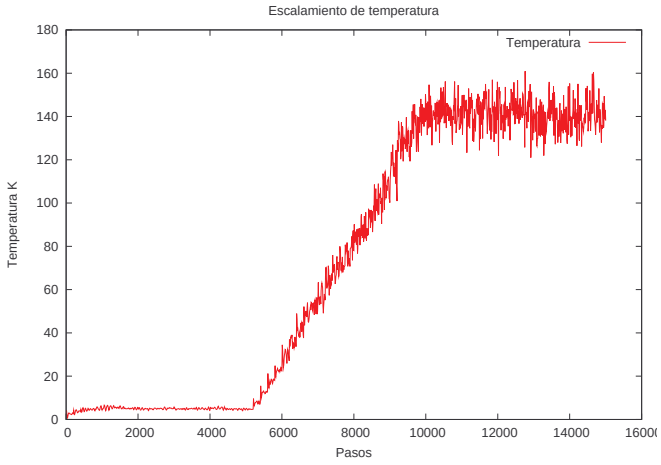
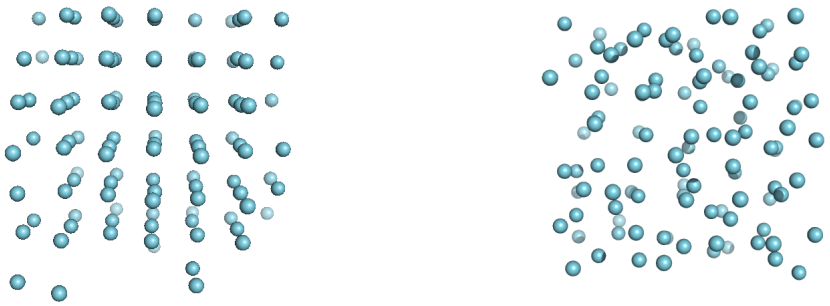


Figure 7.2.: Escalamiento de Temperatura.

Muchas otras características pueden ser observadas por medio de esta simulación, por ejemplo, la muestra durante los primeros 5000 pasos se puede visualizar ordenada como se observa en la figura 7.3(a) y después de la termalización termina desordenada por efecto de la temperatura como se ve en la figura 7.3(b). Propiedades a partir de estas configuraciones `xyz` pueden ser analizadas utilizando **lpmd-analyzer**.



(a) Configuración de Ar a los 4000 pasos de simulación.

(b) Configuración de Ar al finalizar la simulación.

Figure 7.3.: Distintos pasos de una simulación de dinámica molecular de 108 átomos de Argon.

Además de este escalamiento de temperatura clásico, `lpmd` posee dentro de sus plugins otros termostatos, refiérase al capítulo 4.

7.1.3. Ejemplo3: Escalamiento de Celda.

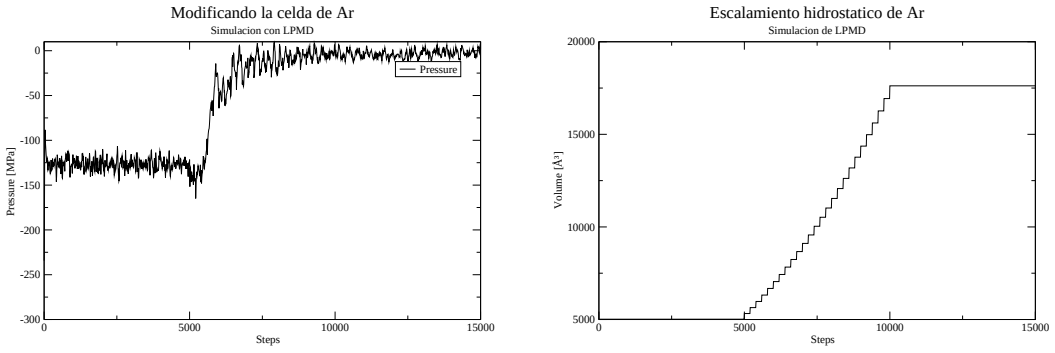
Modificaremos las características de la celda durante la simulación, para ello haremos uso del módulo cellscaling, y luego veremos cómo se modifica la presión del sistema, junto con la temperatura.

De manera similar al ejemplo anterior, se carga un módulo que modifica una propiedad de nuestro sistema, para llevarlo a las condiciones deseadas. En el siguiente fichero de control, se ve cómo se escala una celda.

```
# System file of Ar gas
# using LPMD
#
####################
#CELL and IN/OUT###
####################
cell crystal a=17.1191 b=17.1191 \
     c=17.1191 alpha=90.0 \
     beta=90.0 gamma=90.0

input module=lpmd file=Ar108.lpmd
output module=xyz file=output.xyz \
     each=20 level=0
####################
#GENERAL###########
####################
prepare temperature t=84
steps 15000
periodic true true true
monitor start=0 end=15000 each=10 \
  properties=step,kinetic-energy, \
  potential-energy,total-energy, \
  temperature,pressure,volume \
  output=monitor.dat
####################
#MODULES DEF#######
####################
```

```
use lennardjones as lj_Ar
    sigma 3.41
    epsilon 0.0103408
    cutoff 8.5
enduse

use beeman
    dt 10.0
enduse

use minimumimage
    cutoff 8.5
enduse

use cellscaling as CS
    axis all
    percent 2
    constant true
enduse
####################
#MOD APPLICATION###
####################
potential lj_Ar Ar Ar
integrator beeman
cellmanager minimumimage
apply CS start=5000 end=10000 each=200
```

Podemos ver entonces, cómo cambia la presión y el volumen durante la simulación en las figuras 7.4(a) y 7.4(b), con estos datos se pueden obtener distintos tipos de propiedades mecánicas de los materiales en estudio, por ejemplo el módulo de *Bulk*.



(a) Cambio de Presión en el tiempo.

(b) Cambio de volumen de la celda durante la simulación.

Figure 7.4.: Distintas propiedades del sistema, obtenidas durante la simulación.

7.1.4. Ejemplo4: Calculando Propiedades durante la Simulación.

A continuación, realizaremos un procedimiento simple de dinámica molecular, pero esta vez, con una celda de Oro, sobre la cual calcularemos propiedades durante la simulación. Las propiedades evaluadas serán la *función de distribución radial y angular*, y el *número de coordinación*.

```
cell crystal a=16.320 b=16.320 \
 c=16.320 alpha=90.0 beta=90.0 \
 gamma=90.0

input module=xyz file=au-input.xyz\
      level=1 inside=true
output module=xyz \
      file=au-output.xyz each=30 \
      level=1

# Periodicity in x, y, z
periodic true true true

# Molecular dynamics settings
steps 20000

monitor start=0 end=20000 each=1 \
      properties=step,kinetic-energy,\
      potential-energy,\
      total-energy,temperature \
      output=monitor-cell.out

# Using LSC parameters for gold
use suttonchen as sc
    e 0.013
    n 10
    a 4.08
    m 8
    c 34.408
    cutoff 7.2
enduse

use velocityverlet
    dt 1.0
enduse
```

```
use lcbinary
    mode auto
    cutoff 8
enduse

use gdr
 rcut 10.0
 bins 200
 output gdr.dat
 average true
enduse

use angdist
 bins 200
 atoms 1 Au
 rcut Au Au 3.4
 output angdist.dat
 average true
enduse

use cordnumfunc
 bins 200
 rcut 10
 output cordnumfunc.dat
 average true
enduse

potential sc Au Au
cellmanager lcbinary
integrator velocityverlet
property gdr start=10000 \
         end=15000 each=50
property angdist start=10000\
         end=15000 each=50
property cordnumfunc start=10000\
         end=15000 each=50
```

Las propiedades calculadas se pueden observar en las figuras 7.5(a), 7.5(b) y 7.5(c).

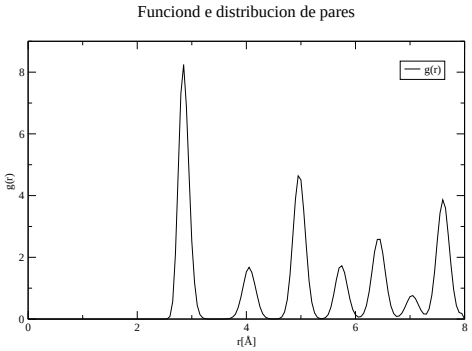
7.1.5. Ejemplo5: Múltiples corridas con bash.

Haremos a continuación un pequeño estudio de un gas de Ar a distintas temperaturas, para ello nos respaldaremos de los flags del comando `lpmd` para poder modificar variables a partir de un fichero de control, realizaremos un estudio de la *función de distribución de pares* para argón bajo distintas temperaturas.

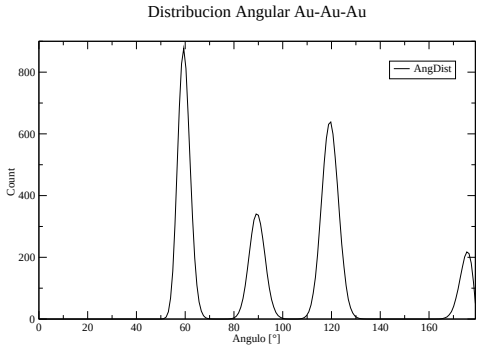
Las temperaturas iniciales que se utilizarán son de 5, 100 y 200 Kelvin, estas tres simulaciones podemos realizarlas a partir de un solo fichero de control, como el que se ve a continuación:

```
# System file of Ar gas
# using LPMD
####################
#CELL and IN/OUT###
####################
cell crystal a=17.1191 b=17.1191 \
     c=17.1191 alpha=90.0 \
     beta=90.0 gamma=90.0
input module=lpmd file=Ar108.lpmd
output module=xyz \
     file=output-$(INITIALTEMP).xyz \
     each=20 level=0
####################
```

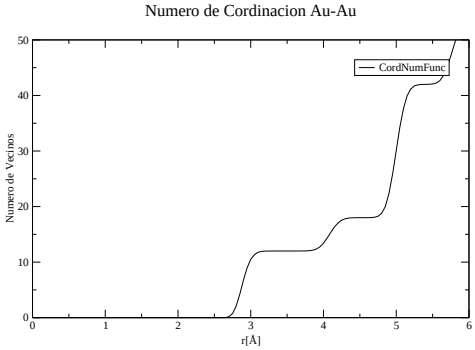
```
#GENERAL############
####################
prepare temperature t=$(INITIALTEMP)
steps 15000
periodic true true true

monitor start=0 end=15000 each=10 \
  properties=step,kinetic-energy, \
  potential-energy,total-energy, \
  temperature,pressure,volume \
  output=monitor-$(INITIALTEMP).dat
####################
#MODULES DEF#######
```



(a) Función $g(r)$ para celda de Au.



(b) Función de distribución angular.



(c) Número de coordinación.

Figure 7.5.: Propiedades calculadas durante una simulación de una celda de Au.

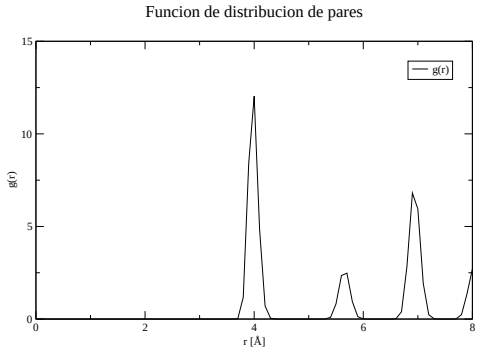
```
####################
use lennardjones as lj_Ar
    sigma 3.41
    epsilon 0.0103408
    cutoff 8.5
enduse

use beeman
    dt 10.0
enduse

use minimumimage
    cutoff 8.5
enduse
```

```
use gdr
 bins 200
 rcut 20
 output gdr-$(INITIALTEMP).dat
 average true
enduse
####################
#MOD APPLICATION###
####################
potential lj_Ar Ar Ar
integrator beeman
cellmanager minimumimage
property gdr start=10000 \\
   end=15000 each=30
```

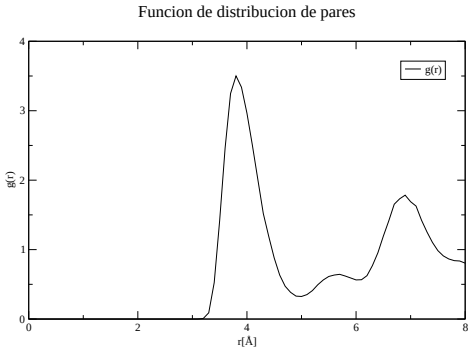
Esta configuración generará archivos `monitor`, `output` y `gdr` de forma independiente para cada una de las temperaturas, para correrlo basta con :

```
for i in 5 100 200 ;
do lpmd -O INITIALTEMP=$i opcion-o.control > simulacion-$i.info ; done
```

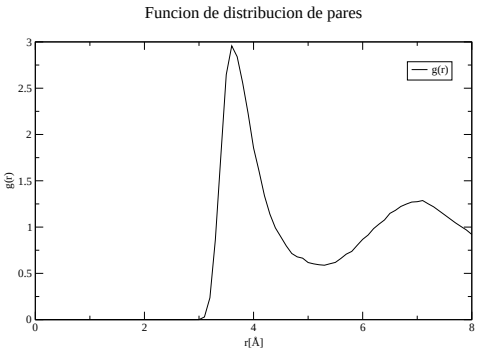
Los resultados de la función radial de distribución de pares, pueden observarse en las figuras 7.6(a), 7.6(b) y 7.6(c), donde se ve claramente que la condición de temperatura inicial es un factor importante en el desarrollo de la dinámica molecular ya que son las condiciones iniciales con las que comienza el sistema su evolución.



(a) Función $g(r)$ para 5K.



(b) Función $g(r)$ para 100K.



(c) Función $g(r)$ para 200K.

Figure 7.6.: Función $g(r)$ para distintas temperaturas iniciales del sistema.

7.2. Ejemplos LPMD-ANALYZER

La función principal de **lpmd-analyzer** es poder cargar configuraciones, provenientes de archivos con distintas extensiones, tales como `xyz` o `lpmd`, para poder obtener análisis de estas configuraciones.

Para los ejemplos de **lpmd-analyzer** utilizaremos un fichero de tipo `xyz` proveniente de una simulación de dinámica molecular hecha con **moldy** de dióxido de germanio, o germania, como se ve en la figura 7.7.

La manera de utilizar **lpmd-analyzer** es ejecutando `lpmd-analyzer archivo.control`.

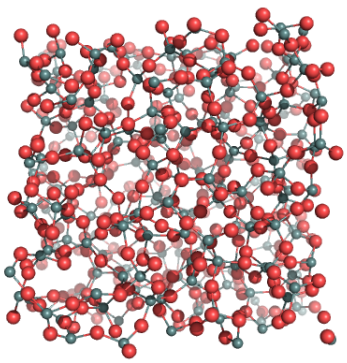


Figure 7.7.: Una de las configuraciones de GeO_2 bajo condiciones ambientales, del total de configuraciones obtendremos distintas propiedades utilizando **lpmd-analyzer**.

7.2.1. Ejemplo1: Calculando la Función Radial de Distribución.

Para calcular la función de distribución radial $g(r)$, creamos un fichero de `control` tomando como input un archivo `xyz` que contiene 101 configuraciones de 576 átomos, pero a diferencia de los ficheros de control de `lpmd`, este fichero cuenta sólo con lo necesario para evaluar una propiedad, como se ve a continuación:

```
#Fichero que calcula g(r)
#para un fichero xyz con 101
#configuraciones

cell cubic a=20.7
input module=xyz file=final20p8.xyz \
      inside=true

use lcbinary
   mode auto
   cutoff 10
enduse

use gdr as pgdr
   bins 200
   rcut 10
   output gdr.dat
   average true
enduse

cellmanager lcbinary
property pgdr start=0 end=100 each=1
```

La manera en que se debe correr este archivo de control es ejecutando `lpmd-analyzer gdr.control`. La información del $g(r)$ será guardada en el fichero `gdr.dat` el cual entregará sólo un $g(r)$ obtenido a partir del promedio de los $g(r)$ de cada configuración. El resultado final del cálculo de $g(r)$ se puede ver en la figura 7.8.

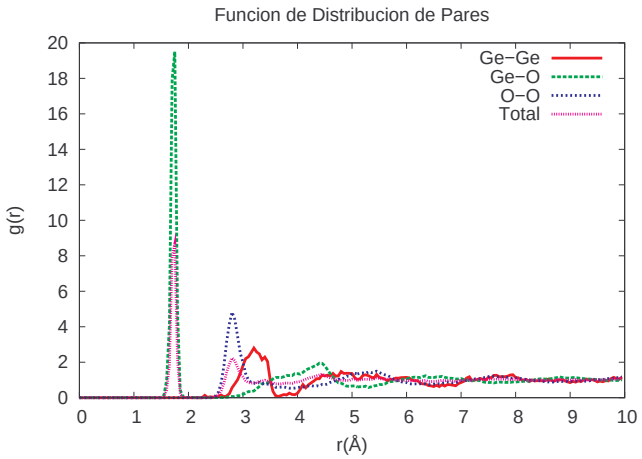


Figure 7.8.: Función radial de distribución de pares, note que `lpmd-analyzer`, automáticamente calcula para cada una de las especies atómicas de la muestra y para el total.

7.2.2. Ejemplo2: Desplazamiento Cuadrático Medio.

Calcularemos ahora el desplazamiento cuadrático medio (`msd`) de nuestra muestra, utilizando el módulo `msd`. El archivo de control, se muestra a continuación:

```
#Fichero que calcula el msd
#para un fichero xyz con 101
#configuraciones

cell cubic a=20.7
input module=xyz file=final20p8.xyz \
      inside=true

use lcbinary
    mode auto
    cutoff 10
enduse

use msd as pmsd
   output msd.dat
enduse

cellmanager lcbinary
property pmsd start=0 end=100 each=1
```

La información obtenida del fichero de salida `msd.dat` se puede ver en la figura 7.9.

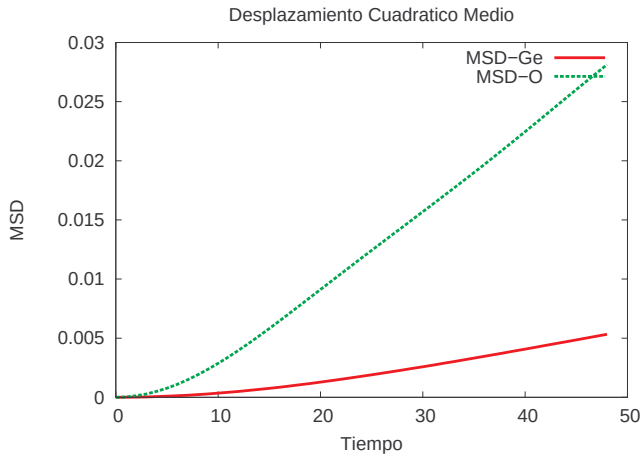


Figure 7.9.: Desplazamiento cuadrático medio (`msd`) calculado con `lpmd-analyzer`.

7.2.3. Ejemplo3: Calculando la Distribución Angular.

Para calcular la distribución angular crearemos el siguiente fichero de control:

```
#Calcula distribucion angular
#para un fichero xyz con 101
#configuraciones

cell cubic a=20.7

input module=xyz file=final20p8.xyz /
      inside=true

use lcbinary
    mode auto
    cutoff 10
enduse

use angdist as ang
   bins 200
   atoms 2 Ge O
   rcut Ge Ge 3.6
   rcut Ge O  1.9
   rcut O  O  3.2
   output ang.dat
   average true
enduse

cellmanager lcbinary
property ang start=0 end=100 each=1
```

El resultado de las distribuciones angulares de toda la muestra se ven en la figura 7.10, los radios de corte utilizados corresponden, como en muchos de los análisis de este tipo, al primer mínimo después del primer *peak* de la función radial de distribución de pares.

7.2.4. Ejemplo4: Número de Coordinación.

El número de coordinación puede ser calculado de distintas formas, en esta ocasión, haremos uso de la función número de coordinación para el cálculo.

```
#Calcula numero de coordinacion
#para un fichero xyz con 101
#configuraciones

cell cubic a=20.7

input module=xyz \
      file=final20p8.xyz \
      level=0 inside=true

use lcbinary
    mode auto
    rcut 10
enduse

use cordnumfunc as cnf
  bins 200
  atoms 2 Ge O
  rcut 10
  output cnf.dat
  average true
enduse

cellmanager lcbinary
property cnf start=0 end=100 each=1
```

Podemos ver el resultado del número de coordinación en la figura 7.11.

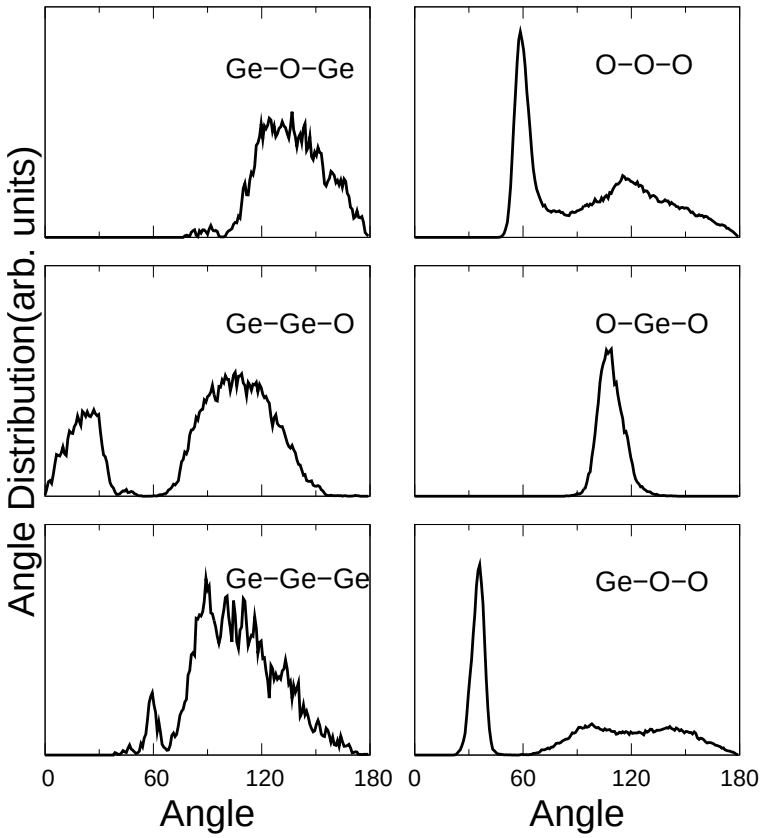


Figure 7.10.: Distribución angular para toda la muestra, calculada con `lpmd-analyzer`.

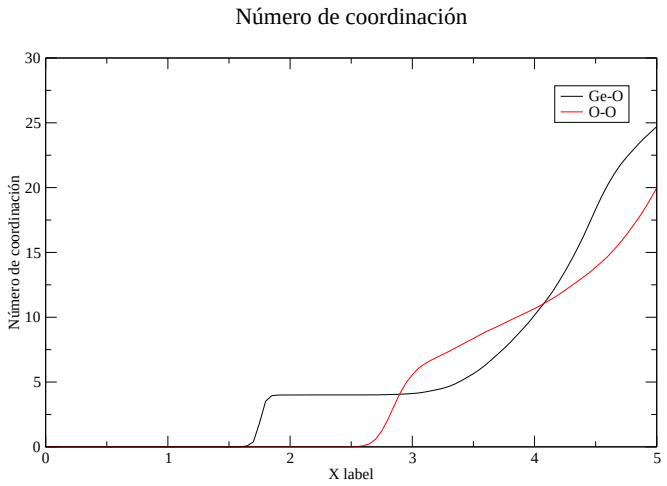


Figure 7.11.: Algunos resultados del número de coordinación calculados con `lpmd-analyzer`.

7.3. Ejemplos LPMD-CONVERTER

Pese a que existe una muy amplia variedad de códigos para convertir entre formatos de ficheros, `lpmd` cuenta con uno propio que, además de poder convertir entre los formatos según el módulo, tiene la característica de poder modificar la celda, agregando, eliminando o bien seleccionando átomos. A continuación veremos un par de ejemplos simples que se pueden realizar con **lpmd-converter**.

7.3.1. Ejemplo1: De un formato a otro

El siguiente ejemplo, muestra cómo convertir un fichero `POSCAR`, que es un fichero de entrada de posiciones atómicas para **vasp**, en un formato de tipo `xyz`.

```
# input para lpmd-converter
cell vector ax=2.650038 ay=0.0 az=0.0 \
     bx=-1.53 by=3.06 bz=0 cx=0.0 \
     cy=0.0 cz=13.45
```

```
input module=vasp file=POSCAR \
      species=Ti,Ga,N
# escribe la salida en formato xyz
output xyz file=TiGaN.xyz level=0
```

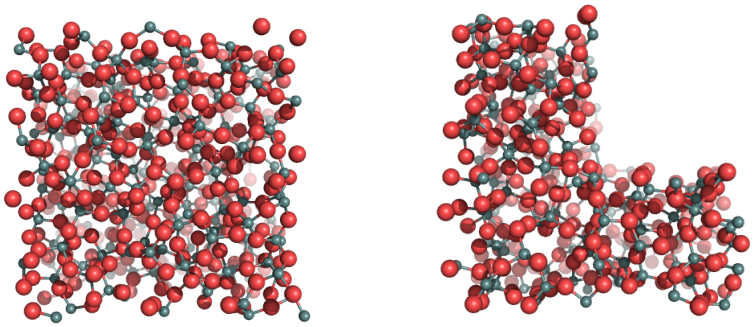
Obteniendo el resultado, ejecutando `lpmd-converter vasp2xyz.control`.

7.3.2. Ejemplo2: Eliminando átomos

Con un archivo de entrada y el plugin `selectatoms`, se pueden seleccionar o modificar átomos dentro de una celda de simulación, es decir, se puede convertir la celda original en una nueva celda modificada. A continuación modificamos una celda original en formato `xyz` en una nueva celda en la cual hemos eliminado una región del espacio.

```
# input para lpmd-converter
cell cubic a=20.7
input module=xyz file=final20p8.xyz \
      level=0 inside=true
output module=xyz file=newcell.xyz \
       level=0
filter box x=10-20.7 y=10-20.7 \
z=0-20.7 inverse=true
```

Con el plugin `filter` se ha seleccionado una parte de la caja, pero, al aplicarle la opción `inverse=true`, se ha seleccionado finalmente lo que quedó fuera de la zona de la caja antes seleccionada, lo cual se muestra en la figura 7.12.



(a) Configuración atómica original.

(b) Nueva configuración atómica en la cual se ha eliminado una región del espacio.

Figure 7.12.: Configuraciones atómicas antes y después de aplicar filter, muy útil para diseñar películas con defectos o elementos distintos dentro de una celda, un posterior análisis también se puede llevar a cabo por regiones.

7.4. Ejemplos LPMD-VISUALIZER

7.4.1. Ejemplo1

Supongamos que tenemos un archivo `configuracion.lpmd` en donde se encuentran las posiciones atómicas, velocidades y colores de 9703 átomos de cobre en formato `lpmd` (ver sec. 4.1.2). La cabecera del archivo tendría el siguiente aspecto:

```
LPMD 2.0
HDR SYM X Y Z VX VY VZ rgb
9703
```

```
70.395 0 0 0 70.395 0 0 0 150
Cu 0.474872 0.448718 0.950733 0 0 -0.04 <1,1,1>
Cu 0.526154 0.448718 0.950733 0 0 -0.04 <1,1,1>
Cu 0.449231 0.474359 0.950733 0 0 -0.04 <1,1,1>
Cu 0.474872 0.474359 0.9387 0 0 -0.04 <1,1,1>
Cu 0.474872 0.5 0.950733 0 0 -0.04 <1,1,1>
Cu 0.500513 0.5 0.9387 0 0 -0.04 <1,1,1>
```

Notamos que los átomos de cobre, en lugar de tener su color natural, aparecen con un color modificado por la última columna: `<1,1,1>`, es decir, blancos. El tamaño de esta celda es de $70.395 \times 70.395 \times 150$. Si quisieramos visualizar la configuración guardada en este archivo desde la terminal de texto, debemos llamar al manejador de visualizadores `lpmd-visualizer` (ver sec. 5.3), el cual llama al visualizador `lpvisual` mediante

```
lpmd-visualizer -i lpmd:file=configuracion.lpmd -u lpvisual -r
```

El conjunto de comandos que sucede a `-i` es propio del formato `lpmd`, visto en la sección 4.1.2. A continuación, se le pide al visualizador que utilice `lpvisual` con las opciones por defecto (nada adicional fue especificado). Por último, el comando `-r` lee (*reads*) los vectores de celda desde dentro del archivo (el formato `lpmd` tiene la ventaja de contener los vectores de celda, por lo que el usuario no tiene necesidad de especificarlos).

7.4.2. Ejemplo2

Supongamos que ahora deseamos visualizar una configuración de átomos guardada en el archivo `atomos.xyz`, contenidos en una celda de $50 \times 20 \times 40$; pero esta vez queremos que la visualización empiece con opciones distintas a las fijadas por defecto, como por ejemplo, el lugar desde donde queremos mirar la celda. En este caso, ejecutamos, por ejemplo,

```
lpmd-visualizer -i xyz:file=atomos.xyz -u lpvisual:zenith=90,azimuth=45 -L 50,20,40
```

Nuevamente, lo que sucede a `-i` es propio del formato del archivo (formato `xyz` en este caso, lo que significa que debemos dar el tamaño de celda con la opción `-L`). Al `lpvisual` le pasamos los ángulos de visualización en coordenadas polares, donde, en la notación habitual, `azimuth` es el ángulo azimutal φ y `zenith` es el ángulo cenital θ. En este caso, $\theta = 90^o$ y $\varphi = 45^o$, lo que significa que

$$x = \sin\theta\cos\varphi = \frac{1}{\sqrt{2}} \tag{7.1}$$

$$y = \sin\theta\sin\varphi = \frac{1}{\sqrt{2}} \tag{7.2}$$

$$z = \cos\theta = 0 \tag{7.3}$$

por lo tanto, la cámara se situará en un vector proporcional al vector unitario $(x, y, z) = \frac{1}{\sqrt{2}}(1, 1, 0)$. La constante de proporcionalidad es el módulo de la diagonal de la celda de simulación.

7.4.3. Ejemplo3

A continuación mostramos un fichero de control en donde la celda de entrada `input` (equivalente a `-i`) no es un archivo, sino una celda de oro generada por el *plugin* `crystal3d`. En este fichero se carga `lpvisual` con todas las disponibles que éste posee, para visualizar los primeros 10000 pasos de simulación (`end=10000`):

```
########################################
# System file of Au crystal            #
# using LPMD                           #
########################################

cell cubic 32.08
input crystal3d \
       type=fcc symbol=Au nx=8 ny=8 nz=8
output module=lpmd \
       file=au.lpmd each=15 level=1
periodic true true true
steps 750000


monitor properties=step,kinetic-energy,\
        potential-energy,virial-pressure,\
total-energy,temperature \
output=au.out start=0 end=750000 each=1
```

```
#######################################
########## Integrator ################
#######################################

use velocityverlet
 dt 1.0
enduse

#######################################
############ CellManager #############
#######################################

use linkedcell
 mode auto
 cutoff 7.5
enduse

#######################################
########## Potential #################
#######################################

# Parameters for gold
use suttonchen as sc
 e 0.013
 n 10
 a 4.08
 m 8
 c 34.408
 cutoff 7.1
enduse



#######################################
######### Tempscaling ################
#######################################

#Aplicar un escalamiento de t de 300K
prepare temperature t=300.0

use tempscaling as tmpsc
 from 300
 to 300
enduse


#######################################
######## CellScaling #################
#######################################

#Estirar la celda
use cellscaling as cs
 percent 1
enduse

#######################################
######### Visualization ##############
#######################################

#Visualizar la simulacion
use lpvisual
 width  640
 height 480
 radius 0.5
 quality 3
 azimuth 0.0
 zenith -90.0
 background white
 autorotate true
 perspective false
 properties potential-energy,kinetic-energy
 plot total-energy,temperature
enduse

#--------------------------------------#
#           use plugins                #
#--------------------------------------#
cellmanager linkedcell
integrator velocityverlet
potential sc Au Au
apply cs start=70000 end=700000 each=70000
apply tmpsc start=0 end=20000 each=50
visualize lpvisual start=0 end=10000 each=1
```

- Esto visualiza la simulación cada 50 pasos (each) durante los primeros 10000 pasos (`end`) de simulacion, comenzando desde el paso inicial (`start`).

- Los parámetros `width` y `heigh` dan el ancho de la ventana deseados, mientras que `radius` y `quality` dan el radio de los atomos y su calidad (que tan parecidos a una esfera serán), respectivamente.

- Con `azimuth` (φ) y `zenith` (θ) podemos dar las coordenadas esféricas angulares que determinan el vector posicion inicial de la camara.

- Con `backgroun`, elegimos el fondo de la pantalla blanco (`white`). Fondos disponibles: white, gray, orange, black (default).

- Con `autorotate`, iniciamos la simulacion en rotación automatica.

- `perspective` inicia una visualización ortorombica (`perspective=false`).

- Finalmente, `properties` son las propiedades cuyo valor numerico sera monitoreado en una ventana secundaria del visualizador, mientras que `plot` serán las propiedades que se graficarán en otra ventana independiente mientras transcurre la simulación.

8. People

The lpmd code had been developed by **Joaquín Peralta**, **Claudia Loyola**, **Felipe González** and **Sergio Davis**, the last one is the principal programmer. The work team aims that lpmd attract the attention of future programmers and colaborators in the project, could be improving or developing plugins as well as suggestions and recommendations about the program.

- PRINCIPAL DEVELOPERS: People that give the initial kick in the project to generate the first versions of the lpmd code. At this moment everyone still be an active member of the lpmd community and are working in different areas providing a variety of aspects at the code, prioritizing upgrades systems, efficient developing, friendly to the end–user, support to the final user, documentation and web administration.
  - Sergio Davis, Royal Institute of Technology, sergdavis@gmail.com.
  - Claudia Loyola, Universidad de Chile, claudial.81@gmail.com.
  - Joaquín Peralta, Universidad de Chile, jperaltac@gmail.com.
  - Felipe González, Universidad de Chile, fullofmetal@gmail.com.
- EXTERNAL DEVELOPERS: They gave to the project, and some of them still doing, support to the code, in areas like utilization, control and studies. Even modifications and direct new developing tool. We hope that soon more new interested developers want bring to us and with their help improve the code and the utilities of lpmd.
  - Yasmín Navarrete, Universidad de Chile, yasmin.navarrete@gmail.com.
  - Pablo Ravelo, Universidad de Chile, pablo.ravelo@gmail.com.
- COLLABORATORS: They are the principal motivators of the lpmd project in order to still working with this project. And they give us suggestions, corrections, contributions to the ideas and overview of lpmd. From technical details to physics are under every improvement.
  - PhD Gonzálo Gutiérrez (gonzalo@fisica.ciencias.uchile.cl).
  - PhD Eduardo Menéndez (emenendez@fisica.ciencias.uchile.cl).
  - PhD Carlos Esparza.

Any questions, contribution or suggestions, is always welcome in our little community. Please do not hesitate to contact us, could be using e-mail to any specific author or directly by the web page http://www.lpmd.cl or e-mail address admin@lpmd.cl.

A. Plugins

A continuación se muestran las listas de los tipos de módulos implementados a la fecha en `lpmd`. Además se indica la calidad del módulo, segun la tabla A

Calidad	Descripción
S	*Stable* El plugin funciona de manera estable.
T	*Testing* El plugin funciona y se encuentra usable, pero debe ser precavido.
U	*Unstable* El plugin esta en fase de desarrollo, no utilizar para publicaciones.

Table A.1.: Tabla de calidad de implementación de módulos, sólo se presta soporte para los módulos incluidos en el paquete **lpmd-plugins**

A.1. Entrada Salida

Módulos para manejar los ficheros de entrada/salida para las configuraciónes atómicas que se simulan.

Módulo	Versión	Calidad	Descripción
`dlpoly`	2.0	S	Lectura/Escritura de ficheros `HISTORY` y `CONFIG` de dl_poly.
`lpmd`	2.0	S	Formato propio de `lpmd`, soporte lectura/escritura, 3 niveles distintos y tags.
`vasp`	5.0	F	Lee ficheros **POSCAR** de vasp, el cuál posee las posiciones atómcias de la configuración.
`xyz`	1.0	S	Formato de ficheros `xyz`, sporta 3 niveles distintos.
`zlp`	2.0	S	Formato propio de `lpmd`, utiliza las zlib, y 3 niveles distintos de manejo, la utilización es similar a lpmd pero con ficheros de menor tamaño.
`mol2`	1.0	T	Lectura/Escritura de ficheros `mol2`, soporte básico.
`pdb`	1.0	T	Lectura/Escritura de ficheros `pdb`, soporte básico.
`rawbinary`	1.0	S	Lectura/Escritura en modo binario. Ideal para espacio y velocidad.

Table A.2.: Tabla con los módulos de entrada y salida utilizados por `lpmd` y sus utilitarios.

A.2. Generadores de Celda

Módulos de `lpmd` que generan celdas atómicas de forma automática.

Módulo	Versión	Calidad	Descripción
`crystal3d`	1.0	S	Generador de celdas tridimensionales.
`crystal2d`	1.0	F	Generador de celdas bidimensionales.
`voronoi`	1.0	F	Generador de Nanoestructuras utilizando método de voronoi.
`skewstart`	1.0	F	Generador de celdas con método skewstart, desarrollado por *K. Refson*, para el programa de dinámica molecular **moldy**.

Table A.3.: Módulos generadores de celda usados por `lpmd` y sus utilitarios.

A.3. Manejadores de Celda

Determinan como es la forma de interactuar entre los átomos pertenecientes a la celda de simulación.

Módulo	Versión	Calidad	Descripción
`linkedcell`	2.0	S	Módulo que utiliza lpmd, para manejar las listas de interacción, utilizando el método *Linked Cell*.
`minimumimage`	2.0	S	Módulo que utiliza lpmd, para manejar las listas de interacción, utilizando el método de *mínima imágen*.
`lcbinary`	1.0	S	Módulo que utiliza lpmd, para manejar las listas de interacción, utilizando el método de *Linked Cell* con ún átomo por celda.
`verletlist`	1.0	U	Módulo que utiliza lpmd, para manejar las listas de interacción, utilizando el método de *Verlet List*.

Table A.4.: Tabla con los módulos que manejan las interacciones atómicas en la dinámica molecular.

A.4. Filtros

Actúan sobre la celda de simulación y son capaces de seleccionar átomos de la celda en distinta forma.

Módulo	Versión	Calidad	Descripción
`box`	1.0	S	Selecciona átomos fuera o dentro de una caja de largos específicos.
`element`	1.0	S	Módulo que utiliza lpmd, para seleccionar átomos según su símbolo atómico.
`index`	1.0	S	Módulo que utiliza lpmd, para seleccionar átomos según su índice.
`sphere`	1.0	S	Selecciona átomos fuera y dentro de una esfera.
`tag`	1.0	S	Módulo que utiliza lpmd, para seleccionar átomos según su tag.

Table A.5.: Tabla con los módulos que manejan las interacciones atómicas en la dinámica molecular.

A.5. Modificadores

Son los módulos que alteran propiedades de la celda, tales como tamaño, forma, o bien modifican los átomos que se encuentran dentro de ella.

Módulo	Versión	Calidad	Descripción
`berendsen`	2.0	S	Escalamiento de temperatura usando algoritmo de berendsen.
`cellscaling`	2.0	S	Escalamiento dinámico de Celda.
`displace`	2.0	S	Desplazamiento de átomos.
`moleculecm`	2.0	S	Crea moléculas diatómicas a partir de átomos enlazados.
`propertycolor`	1.0	S	Colorea átomos según propiedad.
`quenchedmd`	2.0	S	Modificación estructural usando *Quenched MD*.
`randomatom`	2.0	S	Eliminación/Selección de átomos al azar.
`replicate`	2.0	S	Replica celda original.
`rotate`	2.0	S	Rotación de átomos.
`setcolor`	2.0	S	Setea color de los átomos.
`settag`	2.0	S	Setea tag de los átomos.
`setvelocity`	2.0	S	Setea velocidad de los átomos.
`shear`	2.0	S	Modifica la celda realizando cizalle.
`temperature`	2.0	S	Asignación de temperatura a grupos de átomos.
`tempscaling`	2.0	S	Escalamiento de temperatura.
`thermalneedle`	2.0	S	Aguja térmica - obsoleto.
`undopbc`	2.0	S	Deshacer periodicidad en los ejes.

Table A.6.: Tabla con los módulos modificadores del sistema utilizado por `lpmd`.

A.6. Propiedades Instantáneas

Calculan caracteristicas propias del sistema atómico en estudio, son porpiedades que pueden ser calculadas en cada instante de tiempo, adems de la posibilidad de ser promediadas sobre intervalos. Estas propiedades pueden ser calculadas durante la simulación o bien ser calculadas en forma independiente a posteriori.

Módulo	Versión	Calidad	Descripción
`angdist`	2.0	S	Calcula la distribución angular de la muestra.
`atomtrail`	1.0	S	.
`cna`	2.0	S	Realiza un *Common Nieghbor Analysis* de la muestra.
`cordnumfunc`	2.0	S	Calcula la *función número de coordinación* de la muestra.
`cordnum`	2.0	S	Calcula el número de coordinación en forma de histograma.
`densityprofile`	2.0	S	Genera un perfil de la densidad de la muestra.
`gdr`	2.0	S	Calcula la función de distribución de pares de la muestra.
`localpressure`	2.0	T	Genera un perfil de presiones locales.
`overlap`	2.0	S	.
`pairdistances`	2.0	S	Busca la distancias entre los pares de una muestra.
`rvcorr`	2.0	S	.
`sitecoord`	2.0	S	.
`tempprofile`	2.0	S	Perfil de temperaturas de la muestra.
`veldist`	2.0	S	Dsitribución de velocidades de la muestra.

Table A.7.: Tabla con los módulos generales utilizados por lpmd.

A.7. Propiedades Temporales

Calculan caracteristicas temporales del sistema, estas propiedades no pueden ser calculadas *durante* la simulación, pero si pueden calcularse utilizando `lpmd-analyzer` para los archivos de salida de las simulaciones previas.

Módulo	Versión	Calidad	Descripción
`dispvol`	2.0	S	Calcula el volumen desplazado de los átomos.
`mobility`	2.0	S	Calcula la mobilidad atómica.
`msd`	2.0	S	Calcula el desplazamiento cuadrático medio.
`vacf`	2.0	S	Calcula la función de autocorrelación de velocidades.

Table A.8.: Tabla con los módulos generales utilizados por lpmd.

A.8. Integradores

Resuelven las ecuaciones de movimiento del sistema.

Módulo	Versión	Calidad	Descripción
`beeman`	2.0	S	Integrador de Beeman.
`euler`	2.0	S	Integrador de euler.
`hardspheres`	1.0	T	Método de esferas duras para *mover* los átomos.
`leapfrog`	2.0	S	Integrador de leapfrog.
`metropolis`	2.0	T	Método de metropolis, usado en minimización de estructuras.
`nosehoover`	2.0	T	Integrador de nosehoover para ensambles NPT.
`nullintegrator`	2.0	S	No mueve los átomos.
`velocityverlet`	2.0	S	Integrador de VelocityVerlet.
`verlet`	2.0	S	Integrador de Verlet.

Table A.9.: Tabla con los módulos generales utilizados por lpmd.

A.9. Potenciales de Pares

Plugins especializados en la interacción de pares que llevan a cabo los átomos involucrados.

Módulo	Versión	Calidad	Descripción
`buckingham`	2.0	S	Interacción atómica con potencial de Buckingham.
`harmonic`	2.0	S	Interacción atómica con potencial Armónico.
`lennardjones`	2.0	S	Interacción atómica con potencial de Lennard-Jones.
`morse`	2.0	S	Interacción atómica con potencial de Morse.
`nullpairpotential`	2.0	S	Interacción atómica nula.
`tabulatedpair`	1.0	U	Interacción atómica leida desde una tabla de datos.

Table A.10.: Tabla con los Potenciales interatómicos con los que cuenta `lpmd`.

A.10. Potenciales Metalicos

Son los que determinan como interactuan los átomos durante la simulación.

Módulo	Versión	Calidad	Descripción
`finnissinclair`	2.0	T	Interacción atómica con potencial de Finnis-Sinclair.
`gupta`	2.0	T	Interacción atómica con potencial de Gupta.
`nullmetalpotential`	2.0	S	Interacción atómica nula.
`suttonchen`	2.0	S	Interacción atómica con potencial de Sutton-Chen.

Table A.11.: Tabla con los Potenciales interatómicos con los que cuenta `lpmd`.

A.11. Visualizadores

Utilizados para obtener imagenes de la simulación.

Módulo	Versión	Calidad	Descripción
`average`	2.0	S	Visualizador de promedios de datos de simulación.
`lpvisual`	2.0	S	Visualizador de archivos de dinámica molecular basado en openGL.
`monitor`	2.0	S	Visualizador de datos instantáneos de la simulación.
`printatoms`	2.0	S	.

Table A.12.: Tabla con los módulos visualizadores de lpmd.

B. API - liblpmd

La **API** (Ap. Programming Interface) es una herramienta de programación que puede ser utilizada por cualquier usuario/programador que se vea beneficiado por sus características.

Consideramos que la mejor forma de comprender el funcionamiento de esta **API**, es directamente con códigos de ejemplo que pueden escribir los desarrolladores. A continuación se muestran 3 ejemplos de utilización de la **API**, el primero se enmarca en un "nano-programa" de **DM**, el segundo es la evaluación de una propiedad estática de una celda del tipo `.xyz` y la última una propiedad dinámica de una celda.

B.1. Dinámica Molecular Básica

A continuación un código que utilza todas las características de la **API**, para realizar dinámica molecular.

```
 /*
 * Ejemplo simple de dinamica molecular usando el API de liblpmd
 */

#include <lpmd/api.h>
#include <iostream>

using namespace lpmd;

int main()
{
 MD md;            // define md como un objeto de dinamica molecular
 PluginManager pm; // define pm como un manejador de plugins

 SimulationCell cell(1, 1, 1, true, true, true); // cell es la celda de simulacion
 cell.SetVector(0, Vector(17.1191, 0.0, 0.0));   // define los vectores de la celda
 cell.SetVector(1, Vector(0.0, 17.1191, 0.0));
 cell.SetVector(2, Vector(0.0, 0.0, 17.1191));
 md.SetCell(cell);                    // asigna la celda de simulacion al objeto MD

 // Carga de plugins con sus parametros
 pm.LoadPlugin("minimumimage", "");
 pm.LoadPlugin("crystalfcc", "symbol Ar nx 3 ny 3 nz 3");
 pm.LoadPlugin("lennardjones", "sigma 3.41 epsilon 0.0138");
 pm.LoadPlugin("velocityverlet", "dt 1.0");
 pm.LoadPlugin("temperature", "t 600.0");
 pm.LoadPlugin("energy", "");

 CellManager & cm = CastModule<CellManager>(pm["minimumimage"]);
 cell.SetCellManager(cm);            // asigna el manejador de celda

 CellGenerator & cg = CastModule<CellGenerator>(pm["crystalfcc"]);
 cg.Generate(cell);

 Potential & pot = CastModule<Potential>(pm["lennardjones"]);
 PotentialArray & potarray = md.GetPotentialArray();
 potarray.Set("Ar", "Ar", pot); // asigna lennardjones al arreglo de potenciales de MD

 Integrator & integ = CastModule<Integrator>(pm["velocityverlet"]);
 md.SetIntegrator(integ);
```

```
 InstantProperty & energ = CastModule<InstantProperty>(pm["energy"]);

 SystemModifier & therm = CastModule<SystemModifier>(pm["temperature"]);
 therm.Apply(cell);  // aplica el termalizador "temperature" a la celda de simulacion

 // Loop principal de la simulacion, hace 500 pasos
 md.Initialize();
 std::cout << "# Pasos   Temperatura" << '\n';
 for (long i=0;i<500;++i)
 {
  md.DoStep();                        // avanza el sistema un paso de simulacion
  energ.Evaluate(cell, pot);          // evalua las propiedades en el plugin energy
  double T;
  T = pm["energy"].GetProperty("temperature"); // pide valor de temp al plugin energy
  std::cout << i << "         " << T << '\n';
 }
 return 0;
}
```

Para generar el ejecutable,

```
g++ -o nanodm main.cc -llpmd -ldl -lm
```

y listo, tendremos entonces un ejecutable llamado `nanodm` que realizará una simple corrida de dinámica molecular.

B.2. Calculo de Propiedad estática

Consideremos que tenemos una celda de simulación y queremos utiliar las ventajas de la **API** para calcular una propiedad, que sabemos existe en un módulo, por ejemplo **gdr**. El código para el cálculo de **gdr** de la celda nos quea así,

```
 /*
 *
 *
 *
 */

#include <lpmd/api.h>

using namespace lpmd;

int main(int argc, char *argv[])
{
 if (argc < 2)
 {
  std::cerr << "testgdr <file.xyz>" << '\n';
  exit(1);
 }
 PluginManager pm;
 pm.LoadPlugin("xyz", "file="+std::string(argv[1]));
 pm.LoadPlugin("gdr", "rcut 8.0 bins 300 average true");
 pm.LoadPlugin("nullpotential", "");
 pm.LoadPlugin("linkedcell", "nx 7 ny 7 nz 7 cutoff 8.0");

 CellReader & cread = dynamic_cast<CellReader &>(pm["xyz"]);
 InstantProperty & gdr = dynamic_cast<InstantProperty &>(pm["gdr"]);
 ScalarTable & gdrvalue = dynamic_cast<ScalarTable &>(pm["gdr"]);
 CellManager & cm = dynamic_cast<CellManager &>(pm["linkedcell"]);
 Potential & dummy = dynamic_cast<Potential &>(pm["nullpotential"]);
```

```
 pm["gdr"].Show();

 std::vector<SimulationCell> configs;
 cread.ReadMany(std::string(argv[1]), configs);

 Cell cell(13.16, 13.16, 21.39, M_PI/2, M_PI/2, M_PI*120.0/180.0);
 Vector v1 = cell.GetVector(0);
 v1 = Vector(v1.Get(1), v1.Get(0), v1.Get(2));
 Vector v2 = cell.GetVector(1);
 v2 = Vector(v2.Get(1), v2.Get(0), v2.Get(2));
 cell.SetVector(0, v2);
 cell.SetVector(1, v1);
 for (int i=0;i<3;++i) std::cerr << cell.GetVector(i) << std::endl;

 std::cerr << "Read " << configs.size() << " configurations." << '\n';
 std::cerr << "Configuration 0 has " << configs[0].Size() << " atoms\n";

 for (unsigned long i=0;i<configs.size();++i)
 {
  configs[i].SetCell(cell);
  configs[i].SetCellManager(cm);
  gdr.Evaluate(configs[i], dummy);
  gdrvalue.AddToAverage();
 }

 std::cout << gdrvalue << '\n';

 return 0;
}
```

Esto, lo compilamos de manera similar al caso anterior, obteniendo un ejecutable para calcular una propiedad estática, en este caso `gdr` para la celda de simulación.

Índice Alfabético