Las Palmeras Molecular Dynamics: Flexible and modular Molecular Dynamics.

Sergio Davis[1], **Claudia Loyola**[2], **Felipe Gonzalez**[2] **& Joaquín Peralta**[2].

Group of NanoMaterials, http://www.gnm.cl/

[1] Royal Institute of Technology, KTH, Sweden.

[2] Departamento de Física, Facultad de Ciencias, Universidad de Chile.

jperaltac@gmail.com

Abstract

Las Palmeras Molecular Dynamics (LPMD) is a Molecular Dynamics (MD) code written from scratch in C++, as user-friendly, modular and multiplatform as possible. Some of its features are: it is an Open Source code, it works using plugins, it reads simple and intuitive configuration files, and includes utility software to perform analysis, conversion, and visualization of MD simulations.

LPMD *began in july 2007 and, after one month, we released the first version,* **0.4.0**. *Today, the stable branch* **0.5** *in its latest version,* **0.5.3**, *has a lot of new features compared to* **0.4**. *Recently,* **0.5** *was closed to take on the next big branch,* **0.6**, *which will include features such as efficient methods for electrostatic interactions, and others, such as MPI support.*

On the latest tests, LPMD has shown good agreement with other software, but longer simulation times; an issue which we hope to improve in future releases. More information about the current development of LPMD can be found on **http://www.gnm.cl/lpmd**.

1. Introduction

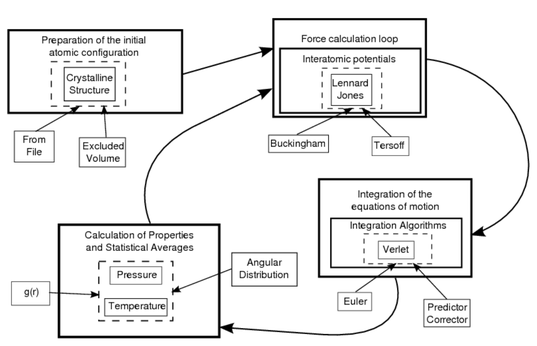
Molecular Dynamics (MD) is a computational simulation technique used mainly in material simulations [1, 2](such as crystalline or amorphous solids, liquids or gases). The MD method consider the atoms or molecules interacting with a potential and obeying the Newton equations of motion, and the physical properties are computed from averages over many atomic configurations at different time steps. It is important to note even when the number of particles is very different from the one in a macroscopic solid, the physical properties (both structural and dynamical) show a very good agreement with experiment.

One of the reasons for starting *LPMD* was the need to create special characteristics in MD, for example, "frozen" atoms, fixed walls, change of parameters during the simulation itself, etc. Basically, to have absolute control over the simulation. Many alternative software have similar approaches, but the inversion in learning-time or money is usually high. *LPMD* has a core of a MD diagram, so the main API [3] developed by us can be used for the different requirements. With this idea in mind, *LPMD* includes software to run MD, analyze output provided for other simulation software or *LPMD* itself, convert between different file formats and visualize results.

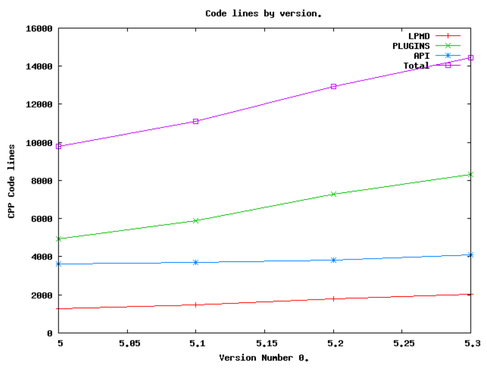
2. LPMD and Associated Software.

2.1 LPMD

The **LPMD** main program includes all the tools needed for performing MD, based, in a general vision, on the diagram shown in the figure 1(a)



(a) Typical diagram of a Molecular Dynamics Cycle.



(b) Number of C++ code lines in lpmd.

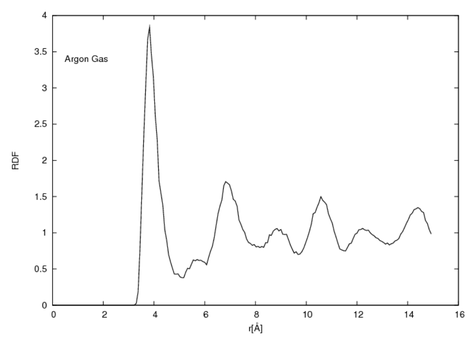
Figure 1: *Working method and code lines number of LPMD.*

2.2 Analyzer, Converter and Visualizer

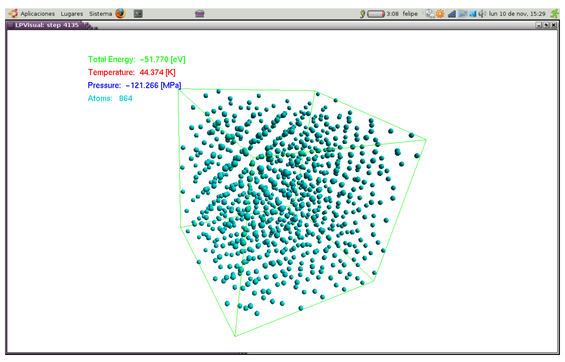
The 0.5 branch includes additional software to compute properties from previous simulations. The idea is to take advantage of the API to develop software for analysis of simulations. The utility software included in branch 0.5 are:

- Analyzer: Analyzes molecular dynamics output files (from other software or LPMD itself).
- Converter: Converts between different MD file formats.
- Visualizer: Visualizes MD simulations. Used to make movies or just for nice presentations.

Most of the plugins developed for *LPMD* can be used in another applications. For example, figure 2(a) shows an analysis out of a simulation performed using *moldy*. Figure 2(b) shows an atomic configuration using the *visualizer* utility and the lpvisual plugin.



(a) Structure Factor, for GeO_2.

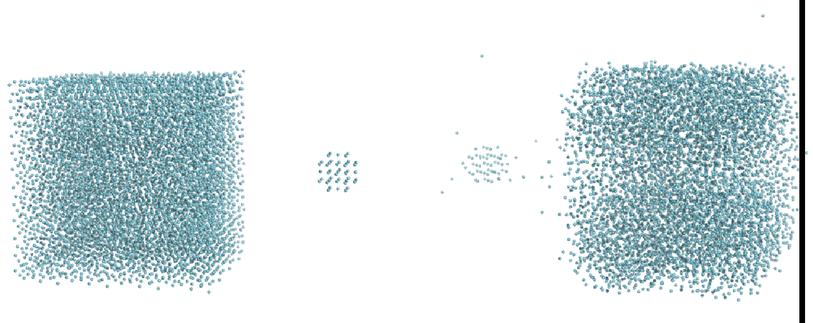


(b) lpmd-visualizer. Argon cell simulation.

Figure 2: *Other characteristics.*

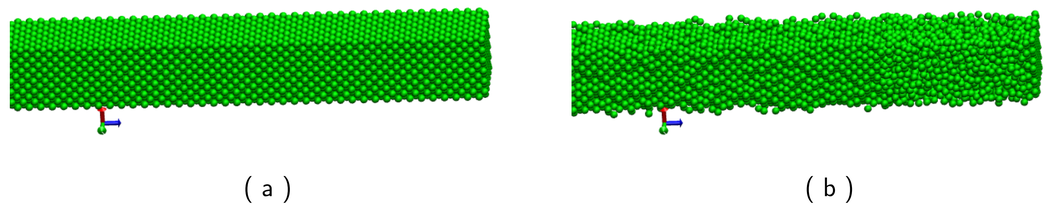
3. Simulations

Figure 3(b) shows a typical MD simulation of a high velocity impact, made with lpmd, by shooting a projectile with fixed velocity. This property is very easy to set in *LPMD*.



(a) before (b) after

Figure 3: *High Velocity Impact.*



(a) (b)

Figure 4: *Shock Wave Simulation.*

4. Parallelization

Attempts of parallelizing LPMD started with the OpenMP model, so for now it only works for shared memory machines. Despite this, we can already see a good decrease in calculation times for Ar simulations (from 108 atoms to 1296 atoms), as shown in figure 6 from 1 to 4 processors.

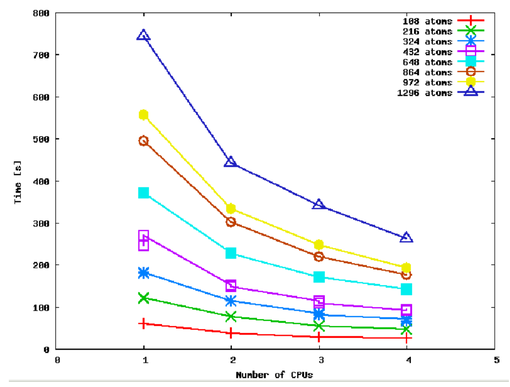


Figure 5: *Simulation time versus number of processors for different cell sizes.*

5. Plugins

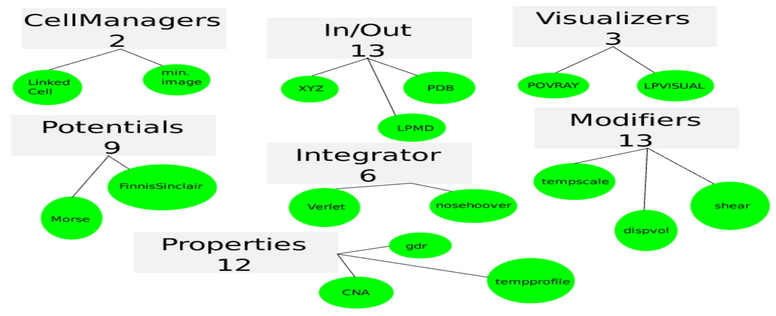


Figure 6: *Some examples of the plugins included with LPMD, and their place in the overall framework*

6. Final Remarks and Future Work

† **Remarks**

- LPMD works on most computer hardware, adding new features does not imply a reinstallation of the full code.
- LPMD has a simple input format (minimal learning curve) based on loading and applying modules.
- LPMD is far from being just a simple MD code: it includes utilities such as *analyzer*, *converter*, *visualizer*, all connected within the common MD framework. Using the LPMD API it is easy to make your own custom C++ and Python MD codes.
- It supports many formats for input and output files, unlike most molecular dynamics codes. Some plugins were developed as interfaces between MD codes, for easy comparison and migration.
- As free software, we aim for frequent releases and bug fixes. During last year, a lot of new features has been implemented, such as structural analysis, potentials and file formats.

† **Future Work**

- MPI support. We hope to achieve combined support for MPI and OpenMP.
- Ewald summation method, or some other faster ones for electrostatics interaction forces.
- Mixer. A new program for building complex material structures.
- Full integration between LPMD and the Python programming language.
- Attend requests from other users or colleagues.

References

[1] M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids* **Oxford Science publications**, (1987)

[2] D.C. Rapaport, *The art of molecular dynamics simulation* **Cambridge**, (2004).

[3] Application Programing Interface, is a set of functions, procedures, methods, classes or protocols that an operating system, library or service provides to support requests made by computer programs.

[4] B. L. Holian and P. S. Lombdahl, *Science* **280**, 2085 (1998).

7. Acknowledgements

This work is supported by grant Anillo Bicentenario-Chile ACT/24 "Computer Simulation Lab for nano-bio systems". JP acknowledges MECESUP UCH008, CL acknowledges to CONICYT Doctoral fellowship grants.