LAMMPS for Dummies

G.A. Frank* and I. Sticco†
Departamento de Física, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Pabellón I, Ciudad Universitaria, 1428 Buenos Aires, Argentina.
(Dated: March 11, 2016)

This article is intended as a starting point in the art of simulation. We will try to make it easy...

PACS numbers: DuMMy, 1.2.me

I. INSTALLATION

We chose LAMMPS as our main simulation tool. Why? Because someone in our team liked it and he encouraged us to use it. But for a dummy like me, the first milestone was to install LAMMPS into a Linux machine. My advise: get a friendly Linux distribution like Ubuntu. Furthermore, use the most widely spread distro that you see in the desks around (in order to have someone to ask in case you get into trouble).

Once you have a working machine running, download the most recent stable version of LAMMP from

`http://lammps.sandia.gov/download.html`

You will get a "tarball", that is, a file named something like `lammps-stable.tar.gz`. You need to unpack the tarball in two steps. The first step will unzip the extension `.gz` and the second one will uncompress all the files into their corresponding directories and sub-directories. How to do that? Easy, open a "terminal" (any icon representing a computer screen with a "prompt" like `$_` or `>_` or similar will open a terminal). Make sure that `lammps-stable.tar.gz` is in the correct directory, say, the directory intended for your own documents (*i.e.* `/home/you/Documents`, where "you" is your username). Then type,

```
cd /home/you/Documents
gunzip lammps-stable.tar.gz
tar xvf lammps-stable.tar
```

It's OK if the terminal becomes verbose. It is showing all the files that are been uncompressed. After it stops printing messages you can type a `dir` command and you will notice that a new directory has been created. It will look something like `lammps-10Aug15` or any other date. All the source files are placed inside.

LAMMPS needs the `C` language and the `MPI` protocol support in order to run. We need to check if both of them are already installed in our machine. You can type `g++` for the first check and `mpirun` for the second one. `g++` is the `C` compiler, while `mpirun` calls the MPI protocol. If any of them are not installed, you will get the corresponding error message. Please, read carefully the message because other error messages are possible, such as "no source file" or whatever. In case you need to install them, type respectively

```
sudo apt-get install g++
sudo apt-get install openmpi-bin libopenmpi-dev
```

Remember that the `sudo` command gives you the power to become a super-heroe inside Linux. You will need to type your personal password to proceed. After that, you are able to finish the LAMMPS installation by typing

```
cd /home/you/Documents/lammps-10Aug15/src
make mpi
```

The process will take some minutes... After that you will notice that the executable file `lmp_mpi` has been created. Do not care if some warnings appear during the process, it happens. But do care for for any error message. The most common messages correspond to missing software in your computer. Read carefully the messages any google how to install them in your computer.

To know if LAMMPS is working fine, just run an example. Try this one

```
./lmp_mpi -i /home/you/Documents/
lammps-10Aug15/examples/melt/in.melt
```

(its only a one-line command; it appears as a broken line for space reasons). The screen will show a small report of somekind of simulation. If you reach this point, you are done!

Be aware that the `./` means that Linux should look for `lmp_mpi` at the current directory, otherwise a "file not found message" will come up. To avoid moving to this directory every time you log into the computer, you need to include the current directory in the searching path. Just type

```
export PATH="$PATH:/home/you/Documents/
lammps-10Aug15/src";
echo export PATH="$PATH:/home/you/Documents/
lammps-10Aug15/src" >> /home/you/.bashrc
```

* guillermo.frank@gmail.com; Also at Universidad Tecnológica Nacional, Facultad Regional Buenos Aires, Av. Medrano 951, 1179 Buenos Aires, Argentina.
† ignaciosticco@gmail.com

and the same for the examples directory. These are one-line commands (broken lines appear for space reasons).

```
export PATH="$PATH:/home/you/Documents/
lammps-10Aug15/examples";
echo export PATH="$PATH:/home/you/Documents/
lammps-10Aug15/examples" >> /home/you/.bashrc
```

II. MY FIRST EXAMPLE

We now encourage ourselves to simulate the movement of molecules confined in a squared box. For simplicity, the molecules will be able to move only in the $x - y$ plane (2D movement).

As a first step, we need to write some code for the simulation. LAMMPS has built-in functions for placing molecules inside the box and moving them according to Newton's laws. Thus, our code only needs to call these specific functions. The text file where we write down the list of instructions (*i.e.* built-in functions) is a "script" file. For some reasons (most probably, tradition) the script file is named `in.name` (replace the word `name` for whatever you like). I will call it `in.myexample`. There are many text editors you can use for writting the script. I will use `gedit`, but choose the one you like most. If the text editor is not installed in your computer, type

```
sudo apt-get install gedit
```

(and your password immediately after that). I will start my script file by entering `gedit in.myexample`. As a first line in the script I will write what the instructions do. It will look like this

```
# Lennard-Jones molecules in a 2D box
# This is my first example
```

Comments should always start with the `#` symbol. Any text behind the `#` will treated as a comment. Notice that it is not mandatory to include comments in your script file, but it's a good practice. The following sections will explain the instruction list in order of appearance.

A. The inital conditions

We must specify the molecules initial position and velocity. This is necesary since the movement equation (Newton law)

$$m\,\frac{d\mathbf{v}}{dt} = \sum_{i=1}^{N} \mathbf{f}_i \tag{1}$$

is a second order differential equation. Thus, we must provide $\mathbf{x}(0)$ and $\mathbf{v}(0)$ to get a unique trayectory for each molecule.

Asume that we want to place 100 molecules in the box. There should be enough room for placing the molecules in such a way that each one does not "feel" the presence of the others. This will ensure that no potential energy is present at the begining of the simulation (that is, only kinetic energy). For example, if the molecules are Lennard-Jones particles, the typical cut-off distance for the potential is $r_c = 2.5$ (in reduced units, called Lennard-Jones units). Consequently, we will place the molecules at a separation distance of 3 (reduced) units. We will arrange them in a "square" symmetric configuration. Actually, the arrangement is not important, since the system will keep no memory of the initial configuration after some time.

The box will be filled with 10×10 molecules separated 3 units. The size of the box should be $1.5 + (10 - 1) \times 3 + 1.5 = 30$ for each dimension. The additional 1.5 units at the beginnig and at the end of the last expression corresponds to the separation distance between de molecules and the walls.

The list of instructions for preparing the molecules for the simulation is as follows

```
dimension    2
units        lj
lattice      sq 0.1111 origin 0.5 0.5 0.0
region       box block 0 30 0 30 -3 3 units box
create_box   1 box
create_atoms 1 box
mass         1 1.0
velocity     all create 2 87287 dist gaussian
```

The list is almost self-explanatory. `dimension` and `units` are obvious instructions.

The `lattice` instruction makes the arrangement of the molecules in a `sq` (square) configuration. The number at the right of the `sq` indication is the (reduced) density ρ^* of molecules. Remember that

$$\rho^* = \frac{N}{V^*} = \frac{N}{(r^*)^d} \tag{2}$$

where N is the number of "basis" molecules of the square configuration and V^* is the (reduced) volume of the square configuration. d is the dimension of the configuration (in this case, $d = 2$).

There is only 1 "basis" molecule at the corner of the square. We do not take into account the rest of the molecules because they belong to the neighbour squares. Or alternatively, we can think that each corner is shared by 4 squares, and thus, only 1/4 of the molecule belongs to each square. Consequently, $4 \times 1/4 = 1$ is the number of molecules per square. The reduced density is

$$\rho^* = \frac{1}{3^2} = 0.1111 \tag{3}$$

(caution! this is only valid for `lj` units).

The `origin` option specifies the shift of the square configuration from de origin. Thus, `origin 0.5 0.5 0.0` means that the square will be shifted half of its length (*i.e.* half of 3 reduced units) in the x-axis and the y-axis, but no shift at all will be added to the z-axis.

The next instruction is `region`. We label it as `box` but any other name is possible. The `block` option means it will be a squared geometrical space. The `block` is limited by `xmin xmax ymin ymax zmin zmax`, in that order of appearance. The option `units box` means that the limiting distances are expressed in `lj` units (that is, in reduces units). Notice that `block` is a 3D shaped region, so the z-axis limits must be included. However, the z-axis will play no role.

The instruction `region` defines a space region, but it does not create it. Consequently, two additional instruction are needed: `create_box` and `create_atoms`. Both instructions need an "identification number" (in this case `1`). The identification is necessary because multiple regions or molecule types are possible. We link the identificacion number to the `box` region.

Right after we create the molecules, we realize that their mass property has not been specified. If we do not care about this, LAMMPS will complain when executing the `velocity` instruction, as described bellow. So, we set the `mass` value for the type `1` molecules as `1.0`.

The last instruction specifies the molecules temperature. This temperature is associated to the kinetic energy (remember that no potential energy is present at the beginning because of the cut-off distance) and therefore, it depends only on the velocity. Thus, the initial velocities should resemble the temperature of the hole configuration.

The velocities can be asumed to be normally distributed. This corresponds to the ideal gas distribution. The (reduced) temperature of our Lennard-Jones molecules should be high enough to ensure that no droplets can exist, and therefore, behave as an ideal gas. For not very compact configurations, this can be achieved by setting the (reduced) temperature to $T^* = 2$ (for $\rho^* = 0.11$). The velocity distribution (2D movement only) will look like

$$f(v_x, v_y) = \frac{1}{2\pi(T^*/m^*)}\, e^{-(v_x^2+v_y^2)/(2T^*/m^*)} \tag{4}$$

where $T^* = k_B T/\epsilon$ is the reduced temperature and m^* is the reduced mass. Remember that we have already set the mass value to one (in `lj` Lennard-Jones units).

When telling LAMMPS to create a `dist gaussian`, we obtain the (v_x, v_y) pairs distributed as in (4), but for the preset temperature $T^* = 1$ (that is, $T^*/m^* = 1$ by default). The number `87287` appearing on the left of the `dist gaussian` instruction is the initial seed for the random numbers generator. It can be any other number if you like (your birthday may be?).

We need to change the initial (reduced) temperature to $T^* = 2$. The `velocity` instruction does this by specifying `create 2`, as written above. Thus, the whole instruction line reads something like this: `create` an initial `velocity` set for `all` the molecules with (reduced) temperature value `2`.

The procedure followed by LAMMPS to set the (reduced) temperature to an arbitrary value is the velocity re-scaling. That is, if the molecules are at a current (reduced) temperature T^*, and we want them to be at a desired (reduced) temperature T^*_{desired}, then the velocities should be re-scaled to

$$v_x^{\text{new}} = \sqrt{\frac{T^*_{\text{desired}}}{T^*}}\, v_x \quad , \quad v_y^{\text{new}} = \sqrt{\frac{T^*_{\text{desired}}}{T^*}}\, v_y, \tag{5}$$

Notice that we only need to tell LAMMPS the desired temperature, while the re-scaling is done in the background.

B. The boundary conditions

The molecules are confined to the box, that is, they are not able to escape from the $30\times30\times1$ space (in `lj` units). However, we did not give further details on how should the molecules interact with the walls. For example, the box can have "hard" walls or "soft" ones. The former are walls with a very sharp potencial energy profile. For simplicity, we will assume that the walls have a Lennard-Jones profile. The boundary conditions for this kind of walls can be written as follows

```
fix mywalls all wall/region box
lj126 0.1 0.1 2.5
```

(warning: this is a one-line instruction. The line has been broken for space reasons)

The `fix` command does not look very easy to understand at first. But, if we break into parts, it will become more obvious. The `fix` by itself means something like "perform the following from now on, until we explicitly change it". This particular `fix` is labelled as `mywalls` and it applies to `all` the molecules (inside the box). The specific action to be performed is related to the surface

of the box, that is, to the `wall/region` option. The only region that we have already defined is `box`, and we specify it immediately after `wall/region`.

Once we told LAMMPS that we want to perform some action between the molecules and the walls (remember, `fix mywalls all wall/region box`) we now detail the kind of action. It corresponds to a specific interaction profile. For simplicity, we consider a "hard" Lennard-Jones(12,6) wall-molecule profile. This is passed to LAMMPS by writting `lj126` (see the `fix` intruction above). The three numbers after `lj126` correspond to ϵ, σ and the cut-off distance r_c of the Lennard-Jones(12,6), respectively. If you do not remember the expression of the LJ(12,6), here it is

$$V(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] \quad , \quad r < r_c \qquad (6)$$

The ϵ parameter corresponds to the minimum value in the profile. We will set it to `0.1`. σ represents the zero crossing distance. We will also set it to `0.1` because we expect very short range interactions between the molecules and the walls. Notice that although we still keep $r_c = 2.5$, we actually narrowed the cut-off distance since we have changed the value of σ.

III. UNDERSTANDING LAMMPS REPORT

Leave our anxiety aside for a while, and check out if your instruction set is working fine. Drop `in.myexample` into the `/src` subdirectory and type in the following command-line (remember to move to that subdirectory)

```
./lmp_mpi -i in.myexample
```

and you will recieve the following report

```
LAMMPS (10 Aug 2015)
Lattice spacing in x,y,z = 3.00015 3.00015 3.00015
Created orthogonal box = (0 0 -3) to (30 30 3)
  1 by 1 by 1 MPI processor grid
Created 100 atoms
```

Great! No `ERROR` messages, so it seems that the configuration has been created successfully. LAMMPS presents itself in the first line and tells the date of the current version. In the second line we can find the lattice edges in reduced units. Although we set the length to 3, there is a small round-off error, moving the length to 3.00015. If this bothers you, enhance the precision of the density to `0.111111` (included in the `lattice` instruction above).

Notice that the lattice spacing in the z-axis has length 3. This means that the box where the molecules are placed should be high enough to include the whole lattice. Thus, make sure to build the box with enough space to completely include at least one lattice cell.

The fourth line informs us that a `1 by 1 by 1 MPI processor grid` has been assigned. It looks somehow like a mysterious message. What is a $1 \times 1 \times 1$ processor grid? Is this good or bad?

LAMMPS is able to run the simulation on many processors to improve the overall time performance. Actually, LAMMPS splits (if possible) each dimension into separate processes. Thats why it reports the number of processors assigned to the x, y and z dimensions. In this example, `1 by 1 by 1` means that only 1 processor has been assigned to the simulation. To increase the number of assigned processors we should use the `processors` instruction. But do not worry for now. We will make the improvements later.

The last line in the report is self-explanatory.

IV. VISUALIZING THE CONFIGURATION

Be patient for a short while! Before we run the simulation, we want to make sure that the molecules are in the right places. LAMMPS has a tool for that, and it is called `dump image`. However this instruction uses the `libjpeg` and the `libpng` libraries to build the `.jpg` or the `.png` file.

First, we will install the necessary libraries (through the "terminal") as follows

```
sudo apt-get install libjpeg-dev
sudo apt-get install libpng-dev
```

Second, we will tell LAMMPS that these libraries are now available in the system. This can be done by making a few changes in the `Makefile.mpi` file. You can find it in

```
/home/you/Documents/lammps-10Aug15
/src/MAKE/Makefile.mpi
```

After you open the file (use, for example, `gedit Makefile.mpi`), find the line `LMP_INC = -DLAMMPS_GZIP` and replace it with

```
LMP_INC = -DLAMMPS_GZIP -DLAMMPS_JPEG -DLAMMPS_PNG
```

Do the same with the line starting with `JPG_LIB` by writing

```
JPG_LIB = -ljpeg -lpng
```

You are almost done! LAMMPS needs to be re-compiled to handle this new capability. Go back to `/src` (inside the LAMMPS directory) and write

```
make clean-all
make mpi
```

(caution! `make clean-all` will reset all)

If no `E:` (error) messages appear during the compilation process, you are now able to print an image of the molecules configuration. Add the following instructions into our `in.myexample` file

```
dump myexample all image 1 myexample*.jpg type type
atom_modify    sort 0 0.0
run            0
```

The first line means "make a dump process, named `myexample`, to image `all` the molecules. The images should be saved into files named `myexample*.jpg`, where the asterisk means a sequence of numbers." The two `type` words means that LAMMPS should use the default color and diameter for imaging the molecules, respectively.

At this stage, the `atom_modify` and `run` instructions are not essential. However, both will be needed for running the simulation. The only reason for including these instructions right now is to avoid error messages.

The `atom_modify sort` instruction handles the way the molecules are sorted during the simulation process. LAMMPS makes a spatial ordering of molecules periodically to speed-up the simulation time. The syntax `atom_modify sort 0 0.0` means that we do not want the spatial ordering to happen at all. Thus, we have disabled the spatial ordering because we want to keep the process as simple as possible (remember that this is our very first example!).

The instruction `run` shoots the process. However, `run 0` does nothing at all since we are telling LAMMPS to run "zero time steps of the process" (say, do not run it at all).

V. ANOTHER LAMMPS REPORT

What will happen if we make a new try by typing again `./lmp_mpi -i in.myexample` ? We will get the same report as in the first try, but now the following information will appear on screen

```
 Setting up Verlet run ...
  Unit style    : lj
  Current step: 0
  Time step     : 0.005
Memory usage per processor = 2.60227 Mbytes
Step Temp E_pair E_mol TotEng Press
 0     2     0     0    1.98   0.22
Loop time of 9.53674e-07 on 1 procs
for 0 steps with 100 atoms

Pair  time (%) = 0 (0)
Neigh time (%) = 0 (0)
Comm  time (%) = 0 (0)
Outpt time (%) = 0 (0)
Other time (%) = 9.53674e-07 (100)

Nlocal:    100 ave 100 max 100 min
Histogram: 1 0 0 0 0 0 0 0 0 0
Nghost:    0 ave 0 max 0 min
Histogram: 1 0 0 0 0 0 0 0 0 0
Neighs:    0 ave 0 max 0 min
Histogram: 1 0 0 0 0 0 0 0 0 0

Total # of neighbors = 0
Ave neighs/atom = 0
Neighbor list builds = 0
Dangerous builds = 0
```

The first four lines tell us that LAMMPS defaults to a standard velocity-Verlet integrator. That is, the Newton eq. (1) is solved numerically by means of the velocity-Verlet scheme. The (reduced) time-step is `0.005` by default.

The `memory usage` and the required `loop time` information are shown in the fifth and eighth line, respectively. Our small simulation needs 2.6 Mbytes of memory for running the process, while it took 9.5e-07 seconds to do nothing (¿?). This is fine because LAMMPS people estimate a time consumption of 7.02e-7 seconds per molecule (through a single time step) for a standard Lennard-Jones simulation (on a single processor). Our process consumed only 9.5e-9 seconds (per molecule) but it did nothing at all. This can be checked by watching the percentages needed for each specific job. See that only the `Other time (%)` line exhibits a time consumption.

We are not going to analyze in detail the rest of the report because it will be very boring at this stage. However, it is important to keep track of the `Nlocal` line. Lost molecules may change this number, and sometimes, it can indicate that the simulation failed.

Another important line in the report is the following

```
Step Temp E_pair E_mol TotEng Press
 0     2     0     0    1.98   0.22
```

We can find relevant information on the state of the system here. It says that at the `Step=0` the system temperature is `Temp=2`, but neither the pairwise energy (`E_pair=0`) nor the molecular energy (`E_mol=0`) are present (all magnitudes are expressed in the corresponding reduced units). This is right since the molecules separation distance was set to 3, while the cut-off distance was set to 2.5. That is, each pair is beyond the

interacting distance at the beginning of the process (see Section II A). `E_mol` and `E_pair` are similar magnitudes in this case because no other kind of interaction has been included in the model (bonds, etc.).

According to eq. (4) the initial velocity distribution at `Temp=2` is

$$f(v_x, v_y) = \frac{1}{4\pi}\, e^{-(v_x^2+v_y^2)/4} \tag{7}$$

We are only interested in the modulus $v = (v_x^2+v_y^2)^{1/2}$. Thus, it is better to sum all the contributions (v_x, v_y) having the same modulus v. The resulting modulus distribution is

$$f(v) = \int_0^{2\pi} \frac{1}{4\pi}\, e^{-v^2/4}\, v\, d\varphi = \frac{v}{2}\, e^{-v^2/4} \tag{8}$$

while the elemental area transforms as $dv_x\, dv_y = v\, dv\, d\varphi$. Notice that the energy distribution can be obtained straight foward from the modulus distribution. The kinetic energy is defined as $e_k = v^2/2$ (assuming that the reduced mass m^* equals unity). Thus, the probability of having kinetic energy distributed according to any function $g(e_k)$ should equal the probability of having a velocity modulus distributed es $f(v)$. This means that

$$g(e_k)\, de_k = f(v)\, dv \quad \Rightarrow \quad g(e_k) = f(v)/v|_{v=\sqrt{2e_k}} \tag{9}$$

The kinetic energy then distribution reads

$$g(e_k) = \frac{1}{2}\, e^{-e_k/2} \tag{10}$$

It is easy to verify that summing (integrating) $g(e_k)$ between $0 \leq e_k < \infty$, gives unity as it should. The kinetic mean energy is $\langle e_k \rangle = 2$. Notice that this is close to the sampled value `TotEng=1.98`. Furthermore, the standrad deviation computed from eq. (10) gives $\sigma = 4$, and therefore, the standard deviation of the sampled mean is $\sigma/n = 4/100 = 0.04$. The error is within the standard deviation as expected!

The report also computes the pressure. We read `Press=0.22`. This must be the same as the pressure for an ideal gas because no interacting forces are present at the beginning. In other words, in the well-known formula

$$P = \frac{N k_B T}{V} + \frac{1}{d\, V} \sum_{i=1}^{N} \mathbf{r}_i \cdot \mathbf{f}_i \tag{11}$$

the second term on the right vanishes. d is actually the dimensionality of the system and equals 2 in our example. The other magnitudes are obvious.

Realize that eq. (11) is not expressed in reduced units. Remember that $T^* = k_B T/\epsilon$ and that $\rho = N/V$ (equivalently, $\rho^* = N/V^*$). Thus, the reduced pressure should be $P^* = \rho^* T^*$. In our example, it gives $P^* = 0.2222$, the same as the reported value. ¡Everything looks correct!

VI. RUNNING THE SIMULATION

Now we are able to make a "real" simulation. Just replace the `run 0` instructionwith the following

```
timestep    0.005
fix         1 all nve
thermo      100
run         5000
```

The first instruction tells LAMMPS that time should be increased by steps of `0.005` (reduced) time units. This is, actually, the time step used for solving eq. (1) numerically. The accuracy of the solution will depend on the chosen time step. Smaller time steps will produce more accurate trayectories for each molecule, at the expense of a greater computational effort. Thus, ¡use time steps that meet your needs, no more no less!.

Since our molecules are Lennard-Jones like particles, energy is expected to remain constant along the simulation. That is, the simultation should resemble an NVE system (remember NVE= unchanged number of particles, unchanged volume and unchanged energy). The second instruction above reads "perform an NVE simulation for al the molecules". The number `1` after the `fix` means that this instruction will be labelled under this number.

The `fix nve` implements a velocity-Verlet algorithm by default. This is actually a standard implementation. The velocity-Verlet algorithm is explained elsewhere.

The `thermo 100` instruction calls for a report every 100 steps (see bellow). The `run 5000` is self-explanatory. After typing again `./lmp_mpi -i in.myexample` we will get the following

```
LAMMPS (10 Aug 2015)
Lattice spacing in x,y,z = 3.00015 3.00015 3.00015
Created orthogonal box = (0 0 -3) to (30 30 3)
  1 by 1 by 1 MPI processor grid
Created 100 atoms
Setting up Verlet run ...
  Unit style    : lj
  Current step: 0
  Time step     : 0.005
Memory usage per processor = 2.60227 Mbytes
```

```
Step  Temp      E_pair E_mol   TotEng       Press Step      Temp E_pair E_mol TotEng         Press
   0         2   0      0       1.98        0.22    0         2   0      0        1.98        0.22
 100 2.0005342   0      0  1.9805289  0.22005876  500 2.0005250   0      0   1.9805197 0.22005775
 200 2.0066896   0      0  1.9866227  0.22073586 1000 2.0004502   0      0   1.9804457 0.22004953
 300 2.4082693   0      0  2.3841866  0.26490963 1500 2.0010575   0      0   1.9810469 0.22011632
 400 2.4131575   0      0  2.3890259  0.26544733 2000 2.0004209   0      0   1.9804167 0.2200463
 500 2.4010132   0      0  2.3770031  0.26411146 2500 2.0011037   0      0   1.9810927 0.22012141
 600 2.4068912   0      0  2.3828223  0.26475804 3000 2.0015309   0      0   1.9815156 0.22016839
 700 2.4023948   0      0  2.3783708  0.26426342 3500 2.0002086   0      0   1.9802065 0.22002295
 800 2.4027414   0      0  2.3787139  0.26430155 4000 2.0002881   0      0   1.9802852 0.22003169
 900 2.4058498   0      0  2.3817913  0.26464347 4500 2.000643    0      0   1.9806366 0.22007074
1000 69.055823   0      0  68.365265   7.5961406 5000 2.0000054   0      0   1.9800054 0.2200006
ERROR: Lost atoms: original 100 current 99
(../thermo.cpp:395)
```

The simulation crashed after the step `1000` (that is, 5 (reduced) time units). Notice that before the crash, at exactly the step 1000, the total energy jumps from `2.38` to `68.36`, while the pressure shoots to `7.59`. There is no "physical" reason for this, but a computational reason.

The `ERROR` message tells that a molecule has been "lost", that is, has left the box. This can only be possible if the leaving molecule has enough energy to surpass the potential barrier at the walls. Indeed, this is what happened since the system energy blew out at step 1000. The leaving molecule trayectory becomes meaningless at this point. The velocity-Verlet algorithm has failed in resembling the "true" (*i.e.* physical) trayectory.

The simulation claims for some kind of adjustment. We will try to reduce the time step to `0.001` and see what happens. We now get

The energy keeps almost constant now. Notice that 5000 steps now correspond to 5 (reduced) time units, the same as before since $1000 \times 0.005 = 5$ (reduced) units.

This first experience shows how should we proceed if things go wrong. The following are some pills of advice

- Follow the reports. Try making some adjustments to the parameters.
- Read the LAMMPS on-line help if you are not sure on what a specific instruction is doing.
- If necessary open the source files where the `ERROR` is supposed to have occured. Look deep inside the code until it becomes clear.
- Ask a friend.... but do not bother Axel or Steve for a small deal. Its dangerous to make Axel get angry.