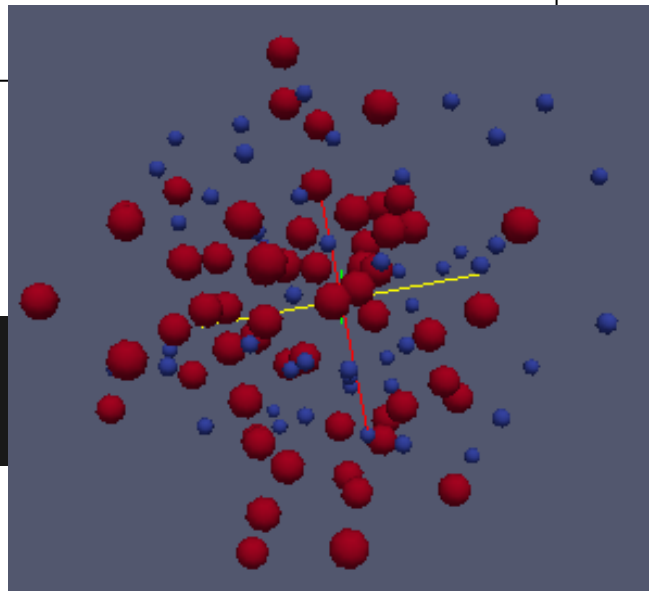
Molecular Dynamics-NBody system

William Fernando Oquendo woquendo@gmail.com
Bruno Guerrero guerrerobruce@gmail.com
Muhammad Umar umar593@hotmail.com
Luis Alfredo Nuñez muhon14@gmail.com



The problem

From an existing c code (serial and parallel):

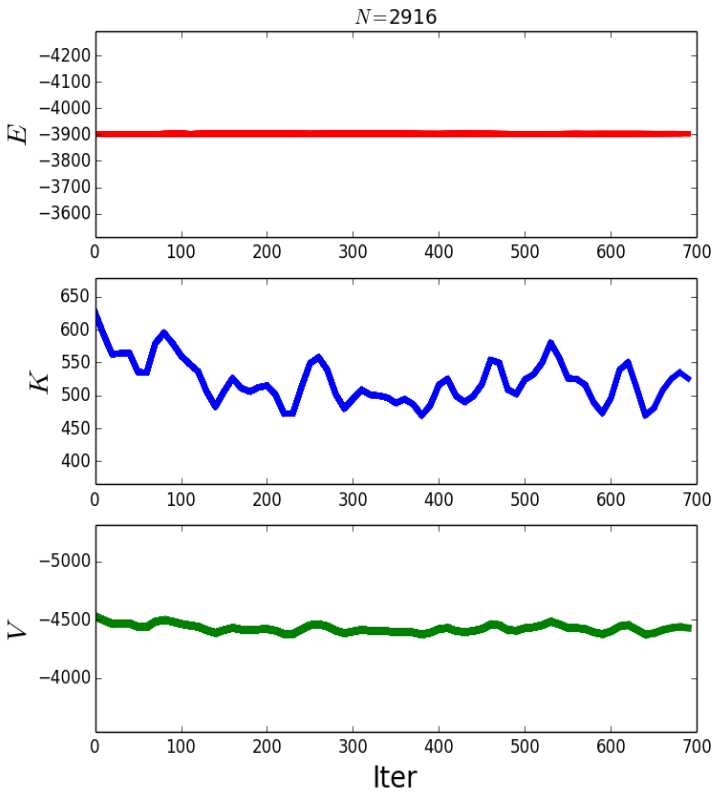
- Write a python interface, keeping in C only the critical parts.
- Generalize the problem for more types of atoms.
- Include a new potential: Morse potential.
- Include documentation.
- Test against old results.
- ...

What we have learned

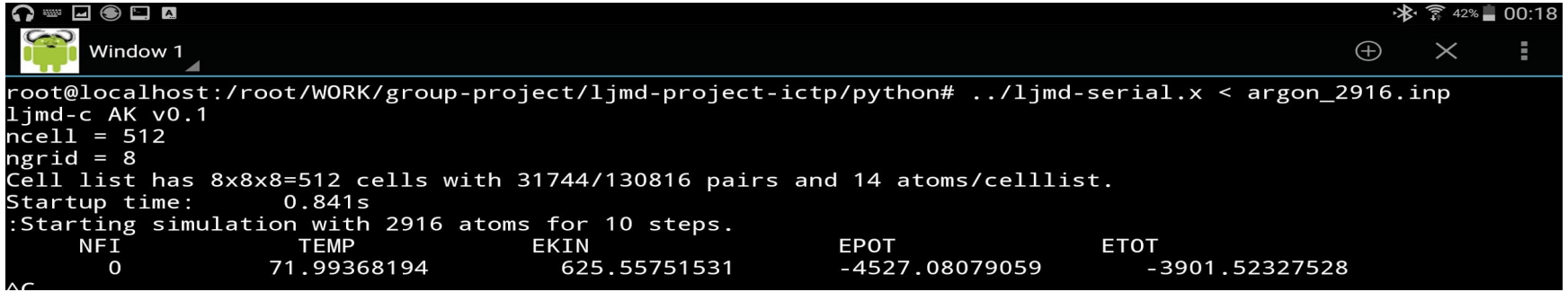
- Communication should be the first thing to be coordinated.
- Dialogue and interaction as soon as possible.
- A master/slave model is sometimes necessary.
- A clear vision of the final product is very helpful .
- Tasks should distributed according to each one strengths.
- Choosing a tool means using it (trello)
- Integration must happens as soon as possible.
- Working on someone' else code can be difficult.
- Learn when to say: done/stop

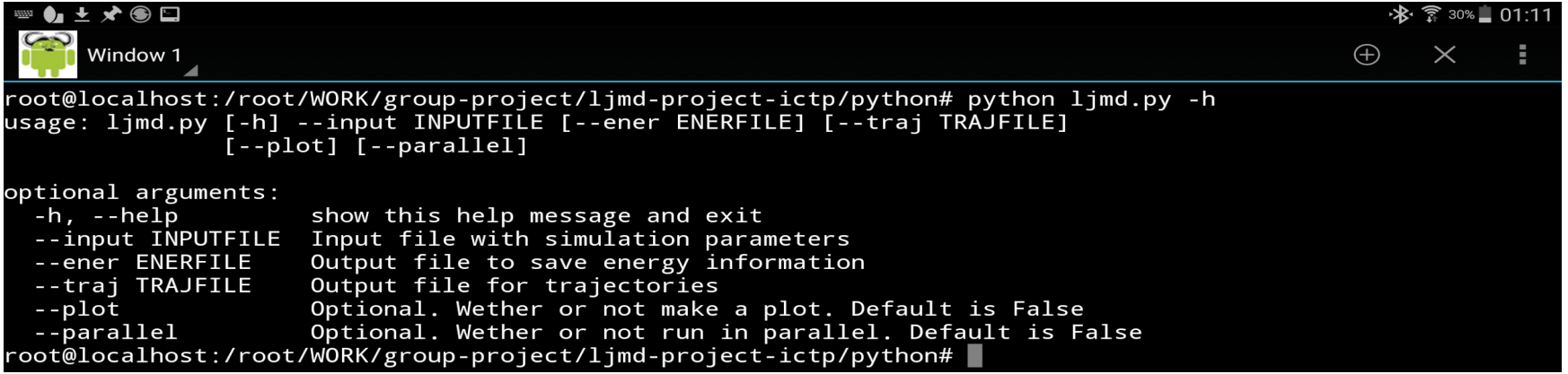
Python interface

- Isolates completely the c interface.
- Communication with c through ctypes.
- Reduces to the minimum the user interaction.
- Plots in real time of some key data.
- Cpu runtime penalty should not be large.
- Allows for using the serial and parallel version.




Window 1
root@localhost:/root/WORK/group-project/ljmd-project-ictp/python# python ljmd.py --input argon_108.inp
Cell list has 4x4x4=64 cells with 2016/2016 pairs and 6 atoms/celllist.
0 72.64160016 23.16879686 -160.48438434 -137.31558748
100 65.22359737 20.80284954 -158.21120604 -137.40835651


Window 1
root@localhost:/root/WORK/group-project/ljmd-project-ictp/python# ../ljmd-serial.x < argon_2916.inp
ljmd-c AK v0.1
ncell = 512
ngrid = 8
Cell list has 8x8x8=512 cells with 31744/130816 pairs and 14 atoms/celllist.
Startup time: 0.841s
:Starting simulation with 2916 atoms for 10 steps.
NFI TEMP EKIN EPOT ETOT
0 71.99368194 625.55751531 -4527.08079059 -3901.52327528
^C


Window 1
root@localhost:/root/WORK/group-project/ljmd-project-ictp/python# python ljmd.py -h
usage: ljmd.py [-h] --input INPUTFILE [--ener ENERFILE] [--traj TRAJFILE]
[--plot] [--parallel]
optional arguments:
-h, --help show this help message and exit
--input INPUTFILE Input file with simulation parameters
--ener ENERFILE Output file to save energy information
--traj TRAJFILE Output file for trajectories
--plot Optional. Wether or not make a plot. Default is False
--parallel Optional. Wether or not run in parallel. Default is False
root@localhost:/root/WORK/group-project/ljmd-project-ictp/python#

Time and memory

N	C - ser	C - par	py - ser	py - par
108	2.97/764	1.44/960	3.02/6572	1.36/6696
2916	18.88/1236	7.32/1620	**19.54**/9080	**7.23**/9396
78732	16.65/1513.2	9.31/22784	43.2+10.2/**1589626**	68/**1457560**

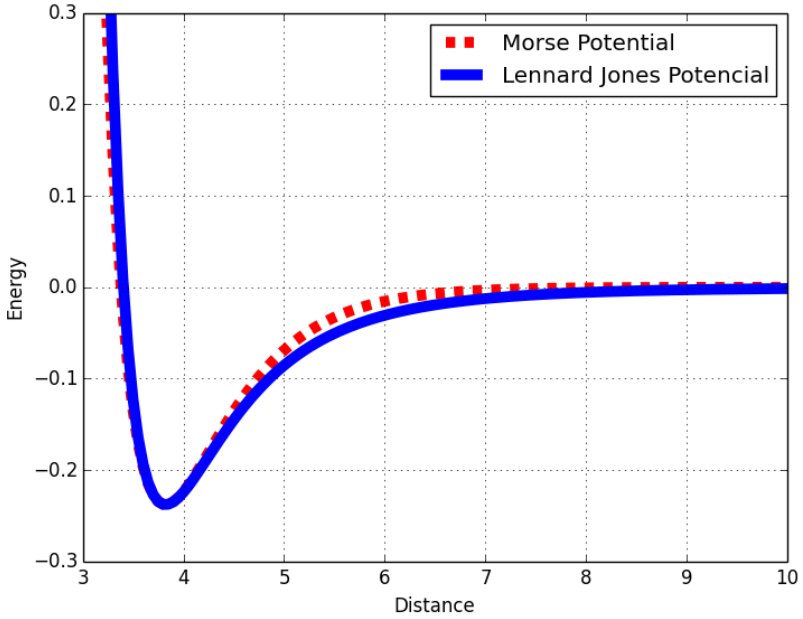
For memory debugging, use memory_profiler and decorate the important function with @profile

Morse Potential

- Development
  - Understand the Code
  - Find the best option to adapt the equations
    - Only we modification a function of the code
    - Input file is the same
    - Choose in the Python interface what potential calculate

Morse Potential

- Comparative



Morse Potential

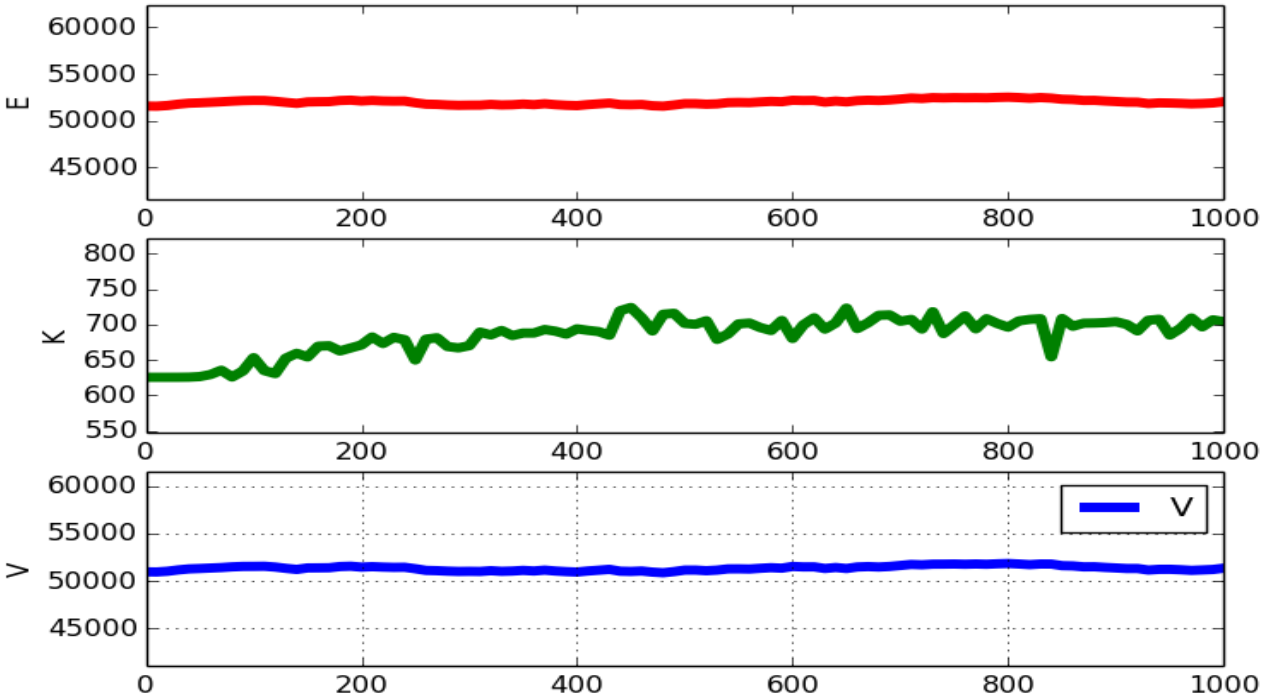
$$V(r) = D_e(1 - e^{-a(r-r_e)})^2$$

Lennard Jones Potencial

$$V_{LJ} = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] = \varepsilon\left[\left(\frac{r_m}{r}\right)^{12} - 2\left(\frac{r_m}{r}\right)^{6}\right]$$

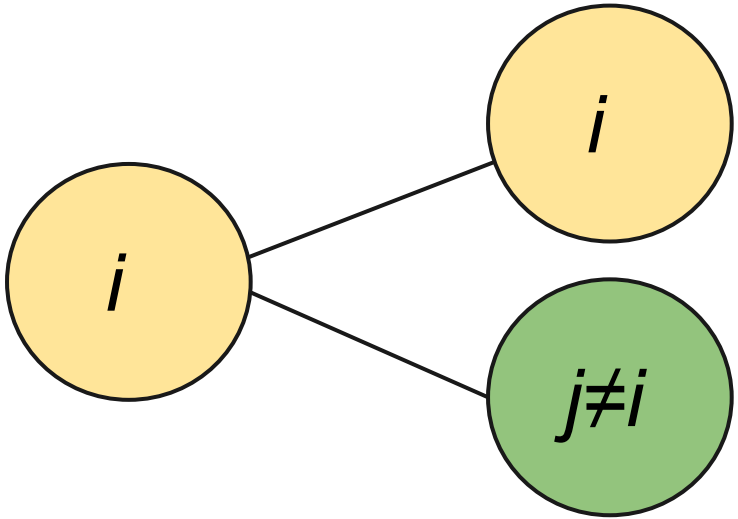
	LJ Pot E	MP Pot E	LJ Total E	LJ Total E
1	5	-154.554	-132.088	137.388	-115.122
29	59	-4378.80	-3621.83	-3902.05	-3113.61
78	1	-120405.5	-98934.3	-105208.8	-83965.2

Morse Potential

- 

Inclusion of different kind of particles

- Was possible assuming that each binary interaction is defined by σ and ε.



m_i

$\{\sigma_{ij}, \varepsilon_{ij}\}$

m_j

They interact between particles of type:

equal $\sigma_{ij}\varepsilon_{ij}$

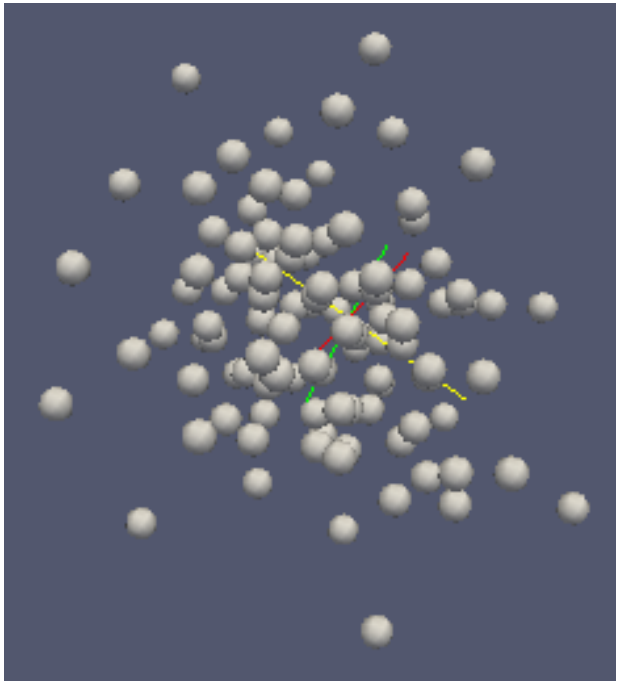
different $\sigma_{ij}\varepsilon_{ij}$

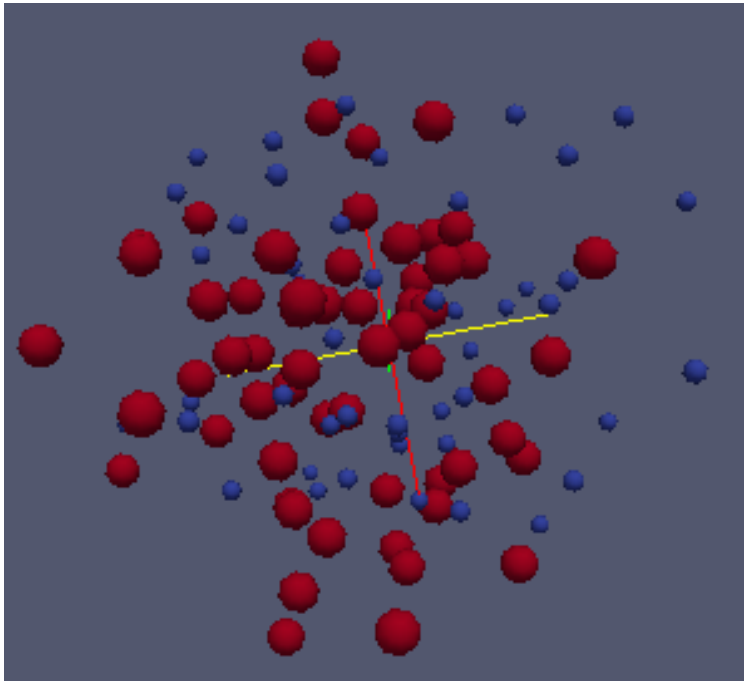
Main modifications to the original code

```
struct _mdsys {
    double dt, *mass, *epsilon, *sigma, box, rcut;
    int *type,kind_potential,nodp;
    /* type[natoms] contain the type of all particles
     * kind_potential =1 (Lennard Jones) or =2 (Moore)
     * nodp is the Number Of Diferent Particles*/
    double ekin, epot, temp, _pad1;
    double *pos, *vel, *frc;
    cell_t *clist;
    int *plist, _pad2;
    int natoms, nfi, nsteps, nthreads;
    int ngrid, ncell, npair, nidx;
    double delta;
};
```

- Changed the way to read input (in order to test)
- Redefined the struct of data, added new variables and modified the functions associated (force,ekin,velverlet)

Creating scripts to convert the original data file and using Paraview to create animations





With the original code **With our generalization**

TODO

- Better documentation (currently using sphinx)
- Unit tests
- Python memory management
- Include the many types of particles generalization in the Python wrapper and integrate with both potentials

Link of Repository

https://bitbucket.org/Bruce_Warrior/ljmd-project-ictp