Molecular Dynamics - NBody system

William Fernando Oquendo
Bruno Guerrero
Muhammad Umar
Luis Alfredo Nuñez

woquendo@gmail.com
guerrerobruce@gmail.com
umar593@hotmail.com
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 be distributed according to each one's strengths.
- Choosing a tool means using it (Trello).
- Integration must happen as soon as possible.
- Working on someone else's 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.
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

\[ V(r) = D_e \left(1 - e^{-a(r-r_e)}\right)^2 \]

<table>
<thead>
<tr>
<th></th>
<th>MP Pot E</th>
<th>LJ Pot E</th>
<th>MP Total E</th>
<th>LJ Total E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-154.554</td>
<td>-137.388</td>
<td>-115.122</td>
<td>-115.122</td>
</tr>
<tr>
<td>29</td>
<td>-132.088</td>
<td>-362.183</td>
<td>-3113.61</td>
<td>-3113.61</td>
</tr>
<tr>
<td>78</td>
<td>-4375.80</td>
<td>-3902.05</td>
<td>-83965.2</td>
<td>-83965.2</td>
</tr>
<tr>
<td>959</td>
<td>-120405.5</td>
<td>-105208.8</td>
<td>-105208.8</td>
<td>-105208.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Morse Potential

Lennard-Jones Potential
Morse Potential
Inclusion of different kind of particles

- Was possible assuming that each binary interaction is defined by $\sigma$ and $\varepsilon$.

They interact between particles of type:

- equal $\sigma_{ij} \varepsilon_{ij}$
- different $\sigma_{ij} \varepsilon_{ij}$
Main modifications to the original code

- 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)

```c
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 Different 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;
};
```
Creating scripts to convert the original data file and using Paraview to create animations.

With the original code

With our generalization
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
https://bitbucket.org/Bruce_Warrior/ljmd-project-ictp