A Classical Coulomb Solver

PRACE Summer of HPC 2016 – Training Week

July 4, 2016

In modern molecular dynamics (MD) simulations a fair amount of computation time is spent to repeatedly evaluate pairwise interaction between particles. In this exercise we want to implement a simple Coulomb solver in a language of your choice.

Problem Description

MD simulations need to compute interactions of a set of N particles. For long-range interactions, namely Coulomb forces and Coulomb potentials this is especially adverse, since the number of interactions is not limited to particles in the proximity. To obtain physically-consistent results in a simulation all pairwise interactions have to be taken into account. The aim of this exercise is to implement such a Coulomb solver.

The input data for a Coulomb solver consists of only two entities; the charge q_i of a particle i and the position $\mathbf{x}_i$ in $\mathscr{R}^3$ with the components x_i, y_i, and z_i. The position of the particle i at $\mathbf{x}_i$ is defined with its components as

$$\mathbf{x}_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}. \tag{1}$$

The distance between two particles $|\mathbf{x}_i - \mathbf{x}_j|$ can be computed as

$$|\mathbf{x}_i - \mathbf{x}_j| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}\,. \tag{2}$$

Physical properties relevant for the simulation are the Coulomb potential Φ and the Coulomb force $\mathbf{F}$ as well as the Coulomb energy E_c of the system. A Coulomb solver should at least calculate the forces acting on all particles due to the presence of the other particles. Therefore, one has to compute the Coulomb forces $\mathbf{F}_i$ with $i = 1 \ldots N$ acting on each particle via

$$\mathbf{F}_i = q_i \sum_{j=1}^{N} q_j \frac{\mathbf{x}_i - \mathbf{x}_j}{|\mathbf{x}_i - \mathbf{x}_j|^3} \qquad (i \neq j)\,. \tag{3}$$

Optional part I: Some MD codes also need the potential Φ at each particle position. A compile-time switch (e.g. template parameter) should be used to enable or disable this feature. Compute the Coulomb potentials Φ_i with $i = 1 \ldots N$ at each particle position $\mathbf{x}_i$ due to all other charges via

$$\Phi_i = \sum_{j=1}^{N} \frac{q_j}{|\mathbf{x}_i - \mathbf{x}_j|} \qquad (i \neq j)\,. \tag{4}$$

Optional part II: Another physical property – the Coulomb energy – might be useful for some applications. Compute the Coulomb energy E_c of the system represented by

$$E_c = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{q_i q_j}{|\mathbf{x}_i - \mathbf{x}_j|} \qquad (i \neq j)\,. \tag{5}$$

The Coulomb energy can be used as a quick check for the correctness of the computed results.

1 Requirements

1.1 General Requirements

Correct results for all entities up to datatype precision are mandatory. Additional numerical errors (round-off) arising through a different summation order can be neglected in this exercise.

1.2 Further Optional Features

Additional goodies could be:

- Documentation via `doxygen`. `www.doxygen.org`
- Build environment via `cmake`. `www.cmake.org`

1.3 Format of Input/Output

The input and output files are defined as simple text files. The input file consists of four columns for each particle. The first column contains the charge q_i, the second, third and fourth columns hold the Cartesian coordinates x_i, y_i, and z_i of the particle i. The output file must contain the force components F_{x_i}, F_{y_i}, and F_{z_i} of $\mathbf{F}_i$ in the first, second and third column. The potential Φ_i should be stored in the fourth column if computed. The order of the lines in the output must correspond to the order of the particles in the input file. For the Coulomb energy E_c it is sufficient to print it together with other outputs (timings, additional information) on the console.

2 Guidelines

The implementation should allow different data types for the input and output data, e.g. single precision (32 bit) or double precision (64 bit). The type should be a compile-time parameter. Do not use `boost` or other third-party libraries containing the Coulomb solver for this exercise.

2.1 Workflow

First, read the data from the input file. Then, start the clock and measure the computation time for the forces, potentials and Coulomb energy. Stop the time measurement. Finally write the results to the output file. Print the measured computation time on the console.

2.2 Hardware and Parallelization

The code should run on any modern hardware, but can be optimized for a single compute node of the JURECA cluster. One node of JURECA consists of two Intel Xeon E5-2680 v3 (Haswell) 12-core processors, which means you can benefit from 2×12 cores and SMT features, if you decide to apply parallelization. First start with a simple sequential program. Parallelization can be added as we go.

2.3 Language & Compiler Version

The source code should also contain appropriate comments in English. The program should compile with all modern compilers.

3 Discussion

Some question that may arise are:

- What is the computational complexity of the solved problem?
- How long does the computation of one time step take on your machine?
- How long would the computation of 3 trillion particles take?
- How many digits do you expect to be correct for such an computation?
- What is the limiting performance factor (bandwidth, clock speed, etc.)?
- What language features are important for the performance?
- Why did you chose one language feature/design pattern over another?