Programs description

February 11, 2017

These programs compute the total velocity autocorrelation function (VACF) from LAMMPS [1] dump file (`compute-vacf.c`) and VACF components from the inner sum for a single, chosen atom (`single_atom_vacf.c`).

A dump file that the programs read consists of N frames - snapshots output during LAMMPS simulation. In each snapshot, fixed data is printed for every atom. The amount, order, and type of data printed is hardcoded but modifications should be straightforward. The structure of a snapshot/frame is as follows:

```
ITEM: TIMESTEP
100
ITEM: NUMBER OF ATOMS
1000
ITEM: BOX BOUNDS pp pp pp
0.101913 12.1956
0.178913 21.1211
-0.256216 33.8062
ITEM: ATOMS id x y z vx vy vz c_myPE c_myKE c_myStress[1] c_myStress[2] c_myStress[3]
    c_myStress[4] c_myStress[5] c_myStress[6]
354 3.07598 2.83336 6.5066 -3.50672 -9.69578 0.180922 -7.43928 0.0661875 -136167 -45637.8
    71499.7 85718.2 -99587.7 58634.5
```

A snapshot indicates the time step it was taken, the number of atoms and current system size with it's boundary conditions. Lastly follows a list of properties of each atom, starting with atom ID. Here the properties are positions, velocities, energies, and stress tensor components. The atoms are not in order and it is not guaranteed that their position on the list is not going to change. Atom number in the programs is assumed fixed. `sample.d` is an example of a dump file with 5 frames and 1,000 atoms.

Goal of the programs was to compute VACF and VACF components for a given atom with following requirements:

- The programs have to process potentially large datasets in a relatively short time - no more than several days/a week.
- The datasets sizes were up to 40 GB which meant the dynamic memory limitations of systems on which they were to be processed had to be taken into account.
- The program should preferably use basic libraries that are available on the systems were the processing will take place.

To fulfill the requirements, the programs were developed in C and without libraries such as GNU Scientific Library not available on the systems in question. The programs do not read the entire dataset at any time to save on dynamic memory. The programs were compiled with gcc Linux compilers and successfully tested with an -O3 optimization option against the same codes developed in MATLAB.

The VACF program processed 5-10 GB files, each with during 3-8 h. Future improvement of the programs would be parallelization with shared memory approach, or with `fork()` or a distributed one with MPI. Also, as it is described later, the algorithm could be improved as it is discussed later.

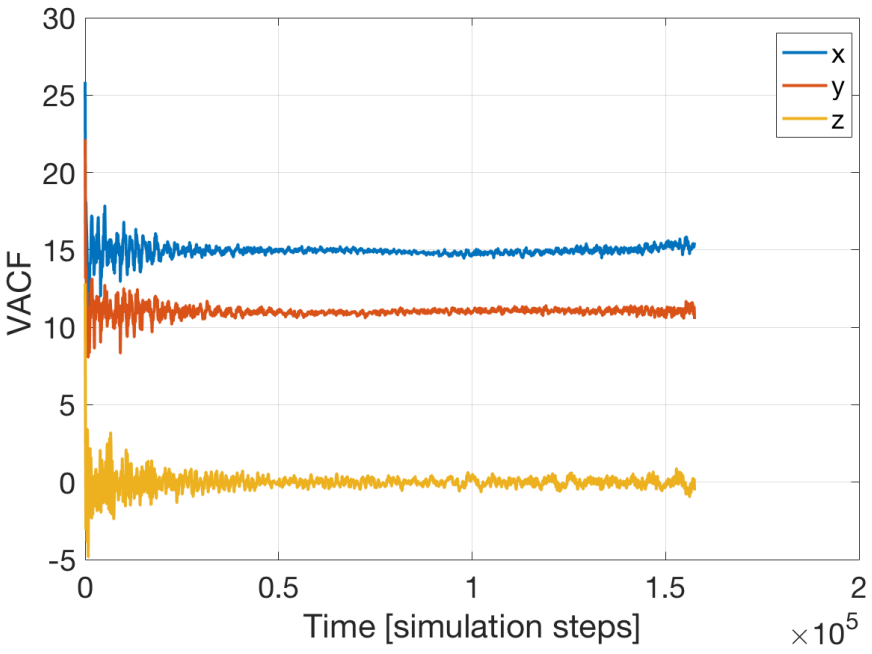

Figure 1:

1 VACF computation - `compute-vacf.c`

VACF at every frame m in direction i (x, y, and z) is computed as

$$VACF_i(t_m) = \frac{1}{A}\sum_{a=1}^{A}\sum_{n=0}^{N-m}\frac{V(t_n)\,V(t_{n+m})}{N-m} \tag{1}$$

Here A is the number of atoms and N is the total number of time steps t, whereas V is the i^{th} velocity component.

Equation 1 implies that to compute VACF at any step we need velocities of all the atoms for as much as all the steps, depending when the computation is performed. Reading the whole file and then working with it was not acceptable due to potential memory limitations, instead the file is read line at a time for a given atom and then rewind. Program loops through atoms in order - though in the file they are not ordered. For a given atom the program collects that atoms velocity from every frame into a $N \times 3$ array. Once the atom is detected at that frame and it's velocity is collected, the search loop is exitted and the program moves on to the next frame. When all the frames are processed, the program computes the inner sum of Eq. 1 for every time step required while adding the result to the outer sum as that atom's contribution.

Although simple, this algorithm fulfilled all three requirements. The improvement to the algorithm would for instance be collecting velocities from the frames for groups of atoms at a time rather than only one atom. The groups could be out of order - i.e. members of a group could be chosen in order they appear in the LAMMPS dump file rather than numerical order.

Figure 1 shows an example result for a file with 1,000 atoms and 1,576 frames. The program ran for 20 minutes, compiled with -O3 on a OS X 10.11.6 with clang-800.0.42.1.

2 Computation of VACF components for a given atom `single_atom_vacf.c`

This program computes components of a single atom VACF from the inner sum for x, y, and z directions. The output for each direction are three $N \times N$ matrices, saved into separate files which names are provided by the user:

$$\begin{bmatrix} V(t_0)V(t_0+t_0) & V(t_1)V(t_1+t_0) & \dots & V(t_N)V(t_N+t_0) \\ V(t_0)V(t_0+t_1) & V(t_1)V(t_1+t_1) & \dots & V(t_N)V(t_N+t_1) \\ \vdots & \vdots & \ddots & \vdots \\ V(t_0)V(t_0+t_M) & V(t_1)V(t_1+t_M) & \dots & V(t_N)V(t_N+t_M) \end{bmatrix} \tag{2}$$

where N is the final time step and M is the step before the final, i.e. $M = N - \Delta t$

The procedure of this computation is the same as in the VACF computation program except it is done only for one atom.

[1] S. Plimpton, Fast Parallel Algorithms for Short-Range Molecular Dynamics, J Comp Phys, 117, 1-19 (1995) (`http://lammps.sandia.gov/`)