<Regression Test Report>

MS-UCG New-'Order of Operation' (MS-UCG-OoOp)

April 22, 2016

Jaehyeok Jin

1. System construction

1.1 Introduction

Various systems with different system size and particles were tested. We fixed the ev_tally routine compared to the solid LJ systems (equally packed 1125 particles) with heterogeneous LJ parameters. Then, the next regression test was held on three particle LJ systems with different LJ parameters.

1.2 New Order of Operation

Compared to the old version of code (composed of four subforces), new order of operation doesn't fully give the value of each subforce- it only returns the two-body forces and the forces from probability derivatives (cv force). Thus, here we only compare the overall force term after accumulating each contribution.

To compare the resultant force value, the Matlab test code was used (also available on GitHub), which was used the last time to check the old version of computation scheme. However, the previous version of Matlab code was constructed as a prototype when $w_{cut} = r_{cut}$. Thus, I've modified the code to make sure that user can type two distinct variables.

<Regression Test Report>

1.3 Regression Test: System

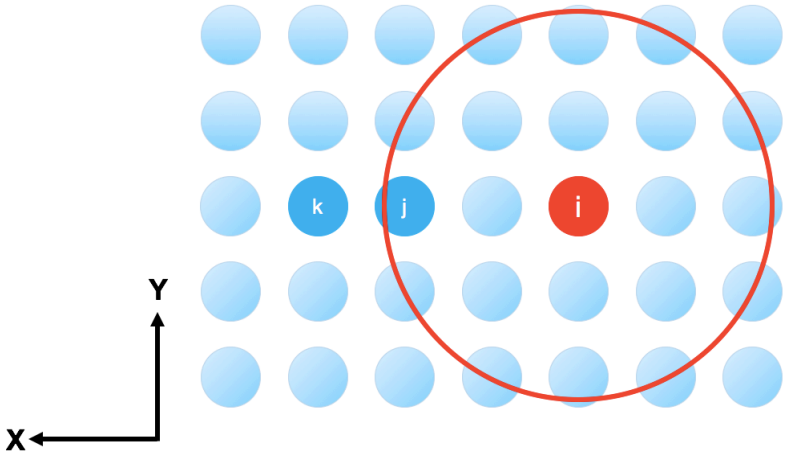

Figure 1: Test System with three particles (i, j, k)

The coordinates of three particles are followed as: particle i (0.0. 0.0, 0.0), particle j (5.0, 0.0, 0.0), and particle k (7.0, 0.0, 0.0).

2. Regression test setting

2.1 Function: compute_proximity_function_der

This function returns the derivative of the proximity function. Detailed value is followed by this equation.

```
double tanh_factor = tanh((distance - distance_threshold) / (0.1 * distance_threshold));
-   return 0.5 * (1.0 - tanh_factor * tanh_factor) / (0.1 * distance_threshold);
+   return -0.5 * (1.0 - tanh_factor * tanh_factor) / (0.1 * distance_threshold);
```

The red row shows the original proposal and derivation from James and the green row shows the modified return value by multiplying -1 to verify the force term.

2.2 Test Parameters

For $r_{cut} = w_{cut} = 3.0$, both (+) and (-) cases show the same force term because w_{cut} value is too large (larger than 1.0 (the local density)) all of the particles have state probabilities 0 and 1 – there is no state-mixing, thus the effect of state derivative is negligible.

Finally, we have tested the following parameters: $r_{cut} = 4.0$, $w_{cut} = 1.0$, $lj_{cut} = 6.0$ and verifying the total force value compared to the modified Matlab code.

3. Results

Table 1 shows the total force for each particle (*i, j, and k)*

Table 1: Total forces from LAMMPS (+ and – signs) simulations and Matlab code

(a) Particle i

Force component	Total force
LAMMPS (+ sign)	(-8.7746 X 10^2, **0**, **0**)
LAMMPS (- sign)	(8.7774 X 10^2, **0**, **0**)
MATLAB	(-8.77461 X 10^2, **0**, **0**)

(b) Particle j

Force component	Total force
LAMMPS (+ sign)	(-7.3911 X 10^4, **0**, **0**)
LAMMPS (- sign)	(-7.5643 X 10^2, **0**, **0**)
MATLAB	(-7.39114 X 10^4, **0**, **0**)

(c) Particle k

Force component	Total force
LAMMPS (+ sign)	(7.4789X 10^4, **0**, **0**)
LAMMPS (- sign)	(7.4765 X 10^2, **0**, **0**)
MATLAB	(7.47889 X 10^3, **0**, **0**)

4. Conclusion

The comparison shows that the (+) sign matches identically to the Matlab data, although (–) sign data shows the opposite signs for the force exerts on the particle i and not correct force values for both particles.

Thus, this regression test shows the verification of the force term and this sign is corresponding to the theoretical derivation of the new order of operation. Now, this new MS-UCG-OoOp reproduces the identical energies and forces compared to various LJ system cases.