25/01/17

Goal: To verify the `fix group/group/atom` by comparing it to another lammps function, `fix addforce`, that is known to work.

I have configured a system of size $15 \times 15 \times 10$, frozen the bottom by excluding it from time integration, and made the top a rigid body, meaning the atoms of this group cannot move relative to each other. I have used the

```
fix addforceID groupID addforce 0.0 0.0 v_Fz
```

with `v_Fz=-1`. This adds the given force to all atoms in the group. Since the input parameter is a variable and given as `v_Fz` as opposed to `${Fz}`, the variable is evaluated at every time step. After equilibrating the system for 20000 time steps, the state was saved to a restart-file. When running the simulation, the variable was set as

```
variable Fz equal (20000-step)/${N}
```

where `step` is a native lammps variable retrieving the current time step, and `${N}` is the number of time steps we wish to simulate. This will give a linear increase in the appliance of this *external* force, that starts at 0 and ends at -1, at the end of the simulation. The reason for this gradual increase, is to counteract oscillations. If the full force were to be applied immediately, it would resemble an elastic collision. Not that I know why that would cause oscillations, though...

I then checked if the top was oscillating by studying the position of the center of mass of the group using

```
compute comID groupID com
thermo_style custom step temp v_Fz c_comID[3]
```

The second line defines the format of the output of the `thermo` command. In this case I got the time step with corresponding temperature, and z-position of the top block (third component of the compute). Once we have monitored that the rigid top has come to rest at a specific hight, we save the state to a restart-file, from which we continue the next fase of the test.

Now, we change the format f the `thermo` command to

```
thermo_style custom step temp v_Fz  f_addforceID[3]
```

Where now the last element represents the total force acting upon the group (whom the addforce works on), excluded the added force. In the same run we use the `group/group/atom`, and compare it to the mean of this value.

31/01/17

Until now I have been struggling with results I did not understand. I have compared the output value from the `fix addforce` for the top group, to the output from `compute group/group` for the top group and middle group. I thought these should match, since the top is at rest, and thus the force acting on the top group from the middle group should be equal to the force we are adding. However, the results did not match. Then I tried to see if the force acting on the middle from the bottom would give a match. It turns out it did. I tested both with and without the external force, and it gives a force of the same magnitude. Well, almost. It's a bit less, though i suspect that is because it only account for pairwise contributions. I believe the reason for weird values when using the top group is due to the `fix rigid` command.

```
fix rigidTop topG rigid single force 1 off off on torque 1
    on on on
```

Where treat the top group as one body, where the total force on it will move its center of mass. We only allow the top group to be affected by force in the z-direction.

I have tried `group/group` on bulk regions, using

```
region R1 block INF INF INF INF 24 35
region R2 block INF INF INF INF 14 24
group G1 region R1
group G2 region R2
compute groupGroup G1 group/group G2 pair yes
```

However, the result seems to vary strongly with the position of this plane.