Manual for running the BD Codes

January 18, 2026

Instructions for Running the Codes Stored in Folder: viscosity_through_shearing

This section demonstrates how to compute the shear viscosity of a simple Lennard-Jones fluid using non-equilibrium molecular dynamics (NEMD). A steady shear flow is generated by moving solid wall regions, and the viscosity is extracted from the resulting velocity gradient and shear stress.

The corresponding LAMMPS input file for this exercise is `lj_viscosity_shearing.in`, and the analysis is performed using the Python script `viscosity_analysis.py`.

System Configuration

The following steps describe how the sheared system is constructed and simulated. Students are encouraged to consult the *Non-Equilibrium Molecular Dynamics* section of the LAMMPS manual for additional background.

1. **Defining the simulation domain and wall regions:**

   ```
   region box block -3 23 0 20 0 20
   create_box 3 box

   region lowersec block INF 0 INF INF INF INF
   region uppersec block 20 INF INF INF INF INF
   ```

   A rectangular simulation box is defined with extended boundaries in the shear-gradient (x) direction. Two slab-like regions are carved out at the lower and upper ends of the box to serve as solid walls, while the central region is occupied by the fluid.

2. **Creating particles and defining wall and fluid groups:**

```
lattice fcc 0.3
create_atoms 1 box

group upper  region uppersec
group lower  region lowersec
group wall union upper lower
group fluid subtract all wall

set group upper type 2
set group lower type 3
```

Atoms are initially created on an FCC lattice. Particles belonging to the upper and lower regions are assigned different atom types and grouped as rigid walls, while the remaining particles constitute the fluid.

3. **Inter-particle interactions and integration scheme:**

```
pair_style lj/cut 3.0
pair_coeff * * 1.0 1.0 3.5
pair_modify shift yes

fix integrator all nve
```

All particles interact via a truncated Lennard-Jones potential. The equations of motion are integrated using the `nve` integrator, which is appropriate since temperature control is applied separately via a Langevin thermostat.

4. **Applying shear and thermostatting the fluid:**

```
velocity fluid create ${T} 12345 mom yes rot yes
velocity upper set 0.0 ${srate} 0.0 sum no

fix frozenwall wall setforce 0.0 0.0 0.0
fix thermostat fluid langevin 1.1 1.1 0.1 933888
fix_modify thermostat temp bias
```

Shear is imposed by assigning a constant velocity in the y-direction to the upper wall, while the lower wall remains stationary. The wall particles are frozen using `fix setforce`.

The fluid is thermostatted using a Langevin thermostat. A velocity-bias correction is applied so that only the thermal fluctuations (and not the imposed shear flow) are thermostatted. This is essential for obtaining physically meaningful transport properties under shear.

5. **Measuring temperature under shear:**

```
compute bias fluid temp/partial 1 0 1
compute fluid_temp fluid temp/ramp vy 0 ${srate} x 1 19
```

The temperature is computed after subtracting the streaming velocity profile associated with shear flow. This ensures that the reported temperature reflects only thermal motion and not systematic flow.

Production Run and Data Collection

After equilibration, the following quantities are sampled during the production run:

1. **Velocity profile:**

   ```
   compute chunk_1 fluid chunk/atom bin/1d x lower 0.05 units reduced
   compute chunkvel fluid vcm/chunk chunk_1
   fix 1 all ave/time 100 10 1000 c_chunkvel[*] file velocity_profile.txt
   ```

   The fluid domain is divided into bins along the shear-gradient direction (x), and the average center-of-mass velocity in each bin is computed. This yields the steady-state velocity profile $v_y(x)$.

2. **Shear stress measurement:**

   ```
   variable pxy equal pxy
   fix pxy_ave all ave/time 100 10 1000 v_pxy file pxy.dat
   ```

   The off-diagonal pressure tensor component P_{xy}, which represents the shear stress, is averaged over time and written to file.

Post-Processing and Viscosity Calculation

The Python script `viscosity_analysis.py` performs the following steps:

1. Loads the time-averaged velocity profile and computes the mean velocity in each spatial bin.

2. Excludes bins adjacent to the walls to avoid boundary-layer effects.

3. Fits a straight line to the steady-state velocity profile, $v_y(x) = \dot{\gamma}x + c$, to extract the shear rate $\dot{\gamma}$.

4. Computes the time-averaged shear stress $\langle P_{xy} \rangle$.

5. Evaluates the viscosity using the constitutive relation

$$\eta = -\frac{\langle P_{xy} \rangle}{\dot{\gamma}}.$$

6. Produces diagnostic plots showing the velocity profile with linear fit, the velocity gradient, and the shear stress time series.

Learning Outcome

This example demonstrates how non-equilibrium molecular dynamics simulations can be used to compute transport coefficients. By comparing the imposed shear, the resulting velocity gradient, and the measured stress, students directly verify the microscopic origin of Newtonian viscosity and gain insight into the connection between flow fields and momentum transport.