HOOMD BLUE – Python based package

1. Run on both multiple CPUS and **GPU**S.
2. Interface is **python** based.
3. Well documented (should join the google group: https://groups.google.com/g/hoomd-users)
4. MD trajectories are written in **binary** format (**.gsd**)

INSTALLATION (Linux):

1. Install MiniConda: https://docs.conda.io/en/latest/miniconda.html
2. Type the command: **conda install -c conda-forge hoomd**

Check if installed properly:

1. conda activate
2. import hoomd
3. import hoomd.md

For Official Tutorial:
https://hoomd-blue.readthedocs.io/en/latest/tutorial/00-Introducing-HOOMD-blue/00-index.html

Running LJ Simulation

```
import hoomd
import hoomd.md
import os
import numpy as np

hoomd.context.initialize("");
# ========================= System Initialization =======================================
system = hoomd.data.make_snapshot(N=ParticleN,box=hoomd.data.boxdim(Lx=BOX_L, Ly=BOX_L, Lz=BOX_L),particle_types=['A']);
system.particles.position[:] = np.loadtxt(str(ParticleN));

hoomd.init.read_snapshot(system);

# ========================= Force Fields ========================================
nl = hoomd.md.nlist.cell(); # Change to tree for large box size

lj = hoomd.md.pair.lj(r_cut=2**(1./6.), nlist=nl)
lj.pair_coeff.set('A', 'A', epsilon=1.0, sigma=1.0)
nl.reset_exclusions(exclusions = []);

# ========================= MD Integrator ========================================
hoomd.md.integrate.mode_standard(dt=TIME_STEP);
all = hoomd.group.all();
integrator = hoomd.md.integrate.langevin(group=all, kT=KT, seed=seed, noiseless_t=False, noiseless_r=False)
integrator.set_gamma_r('A', gamma_r=GAMMA)

# ========================= Print Thrmodynamic Quantities ========================================
hoomd.analyze.log(filename="mddata.dat",quantities=['potential_energy','kinetic_energy','pair_lj_energy', 'temperature'],period=nwrite,overwrite=True)

# ========================= Trajectory Print for Movie ========================================
hoomd.dump.dcd(filename="Running_Config.dcd", period=nwrite)

# ========================= Run Steps ========================================
hoomd.run(Run_Steps);
```

Running a Polymer Simulation

1. Initialize and call the HOOMD module

```
import hoomd
import hoomd.md
import os
import numpy as np

hoomd.context.initialize("");
```

2. Read initial configuration from file and make the bond connectivity

```
# ======================== Particles Connection Initialization ======================================
bond_gr=[]
for i in range(ParticleN-1):
        a_bond=[]
        a_bond.append(i)
        a_bond.append(i+1)
        bond_gr.append(a_bond)

angle_gr=[]
for i in range(ParticleN-2):
        a_angle=[]
        a_angle.append(i)
        a_angle.append(i+1)
        a_angle.append(i+2)
        angle_gr.append(a_angle)
```

Running a Polymer Simulation

3. Polymer Force Field (LJ + FENE + BENDING)

Use Tree list for long connected object

```
# ======================== Force Fields ======================================
#nl = hoomd.md.nlist.cell(); # Change to tree for large box size
nl = hoomd.md.nlist.tree();

lj = hoomd.md.pair.lj(r_cut=2**(1./6.), nlist=nl)
lj.pair_coeff.set('A', 'A', epsilon=1.0, sigma=1.0)
nl.reset_exclusions(exclusions = []);

fene = hoomd.md.bond.fene()
fene.bond_coeff.set('polymer', k=30.0, r0=1.5, sigma=1.0, epsilon= 0.0)
```

Need modification of the bending potential term to match the angle defination

```
# Angle Potential: V = k[1-cos(theta - theta0)], where theta0 = np.pi
# Force: T = - dV/d(theta)

def bend_pot(theta, kappa):
        V = kappa * (1.0+np.cos(theta));
        T = kappa*np.sin(theta);
        return (V,T)
```

Use tabular potential for bending

```
btable = hoomd.md.angle.table(width=1000)
btable.angle_coeff.set('polymer', func=bend_pot, coeff=dict(kappa=kappa))
```

Running a Polymer Simulation

4. Langevin Integrator

```
# ======================== MD Integrator ======================================
hoomd.md.integrate.mode_standard(dt=TIME_STEP);
all = hoomd.group.all();
integrator = hoomd.md.integrate.langevin(group=all, kT=KT, seed=seed, noiseless_t=False, noiseless_r=False)
integrator.set_gamma_r('A', gamma_r=GAMMA)
```

5. Run the simulation

```
# ======================== Run Steps ==========================================
hoomd.run(Run_Steps);
```

6. Write the Trajectory file

```
# ======================== Trajectory Print for Movie =========================================
#hoomd.dump.gsd("Running_Config.gsd", period=nwrite, group=all, overwrite=True);
hoomd.dump.dcd(filename="Running_Config.dcd", period=nwrite)
```