Short Tutorial/Starter For LAMMPS - LJ17 Simulation

Introduction

Hello! This is a short tutorial for getting started using LAMMPS. The files included for this tutorial are: in.run, PARM.FILE, and DATA.FILE which are located in LJ17Needed.zip. A file LJ17_gm.pdb is also provided for use in VMD or some other modeling software that can display pdb style structures.

Quick Bit About LAMMPS

LAMMPS stands for Large-scale Atomic/Molecular Massively Parallel Simulator. It is a classical MD code originally developed by and now distributed by Sandia National Labs. LAMMPS contains a large variety of functionality and is open source, so can be extended by the user. LAMMPS can be run on a single processor or in parallel using some form of message passing, e.g. Message Passing Interface(MPI). The most current source code for LAMMPS is written in C++. More information about LAMMPS including links to the user manual, tutorials, a full list of standard LAMMPS commands, and more, can be found at the LAMMPS web site(http://lammps.sandia.gov/).

Download LAMMPS

First things first. If you don't already have it, you will need to download and install the LAMMPS program (http://lammps.sandia.gov/download.html#download). Most of you are probably Windows users and will want to download the pre-built serial version of LAMMPS for Windows (LAMMPS Windows serial executable at http://lammps.sandia.gov/download.html#download).

Learn About The Input Scripts For LAMMPS

Go to : http://lammps.sandia.gov/doc/Section_commands.html
and read through sections 3.1-3.3

Three files were provided: in.run, PARM.FILE, and DATA.FILE . Let's quickly go through the in.run file with some short descriptions of the commands (However, you should make use of Sections 3.4/3.5 at http://lammps.sandia.gov/doc/Section_commands.html and look through the descriptions given for commands to make sure you understand on a basic level what the command is doing and/or what its purpose is.)

Now let's look at in.run with comments:

 # 17 LJ particles -- this is just an initial comment to denote what type of system you are simulating. In the LAMMPS input script the hashtag (#) is used to comment out lines

units **lj** -- this tells LAMMPS what unit style we want to use. We are using a reduced style of

units called lj, which sets fundamental values to 1, such as the Boltzmann constant. This allows us to simulate LJ systems by using multiples of fundamental values without loss of generality

dimension 3 -- this is the dimensions, so we are running a 3-D simulation

boundary s s s -- this is the boundary condition for the box in x y z. s stands for non-periodic shrinkwrapped.

atom_style atomic -- this sets the atom style. We are running simple non-bonded LJ particles, so we use the atomic style

region sbound sphere 0.00 0.00 0.00 7.6000 units box side in -- this sets a spherical wall shell that surrounds our particles. This is an example of defining a region inside our simulation box.

read_data DATA.FILE -- now we want to read in our data file

include PARM.FILE -- and we want to include the parameters.

timestep 0.005 -- this sets the time step.

run_style verlet -- this sets the integrator, so we want to use the velocity verlet style

velocity all create 0.007 1298371 mom yes rot yes dist gaussian -- this sets the initial velocities of our atoms

fix 1 all nvt temp 0.007 0.007 100.0 -- this defines the ensemble we are using and sets the temperature control

dump 2 all dcd 100 rlx_0.5_LJ17.dcd -- this tells LAMMPS we want to out to file every so many steps of the type dcd. This is the output of our trajectory.

dump_modify 2 unwrap yes -- we tell LAMMPS we want to unwrap the coordinates that are output to the trajectory file.

thermo_style multi -- this tells LAMMPS what thermodynamic data to output to the screen
thermo 1000 -- and then how often (how many timesteps between each output)

fix 5 all wall/region sbound lj126 0.0001 1.0 3.0 -- here we take the spherical region we defined earlier and turn it into a wall.

compute 1 all gyration -- set a compute which calculates the radius of gyration of our atoms

fix Rgave all ave/time 100 5 1000 c_1 file Rg_rlx0.5_LJ17.dat -- now we want to average the radius of gyration over time and output that to a file

restart 1000 run_a.rest run_b.rest -- this sets restart files, which if the simulation crashes for some reason during execution, allow you to restart from the last saved point

run 20000 -- this sets the number of timesteps we want to run.

write_restart rlx_0.5_LJ17.rest -- this tells LAMMPS to output a file that contains restart data after the simulatiion finishes. This allows us to start a new simulation from the end of this one if we choose to.

Running LAMMPS

To run LAMMPS you will need to use the Command Prompt in Windows or the Terminal in Linux/Mac OS. If you are unfamiliar with using these tools, here are some great resources to get you started:
Windows: http://www.computerhope.com/issues/chusedos.htm
Linux/Mac OS : http://blog.codingdojo.com/introduction-terminal-navigation/

Once you are comfortable navigating and using Command Prompt/Terminal you can continue. You will need to have the LAMMPS executable and the files (in.run, etc.) located in the same directory/folder. To run LAMMPS use:

Windows: lmp_win_no-mpi -in in.run

Linux/Mac: lmp_linux/mac < in.run

Running LAMMPS is also covered at http://lammps.sandia.gov/doc/Section_start.html#start_6

When you run LAMMPS with the provided files you should get something like:

LAMMPS (2 Feb 2011)
Reading data file ...
orthogonal box = (-100 -100 -100) to (100 100 100)
1 by 1 by 1 processor grid
17 atoms
Setting up run ...
Memory usage per processor = 0.401821 Mbytes
---------- Step 0 ----- CPU = 0.0000 (sec) ----------
TotEng = -0.9355 KinEng = 0.0099 Temp = 0.0070
PotEng = -0.9454 E_bond = 0.0000 E_angle = 0.0000
E_dihed = 0.0000 E_impro = 0.0000 E_vdwl = -0.9454
E_coul = 0.0000 E_long = 0.0000 Press = 14.2901
Volume = 14.6665
---------- Step 1000 ----- CPU = 0.0225 (sec) ----------
TotEng = -1.6500 KinEng = 1.4418 Temp = 1.0213
PotEng = -3.0918 E_bond = 0.0000 E_angle = 0.0000

E_dihed = 0.0000 E_impro = 0.0000 E_vdwl = -3.0918
E_coul = 0.0000 E_long = 0.0000 Press = 0.2522
Volume = 18.4623
.....
.....
---------- Step 19000 ----- CPU = 0.3941 (sec) ----------
TotEng = -2.9864 KinEng = 0.0744 Temp = 0.0527
PotEng = -3.0608 E_bond = 0.0000 E_angle = 0.0000
E_dihed = 0.0000 E_impro = 0.0000 E_vdwl = -3.0608
E_coul = 0.0000 E_long = 0.0000 Press = 0.2090
Volume = 29.8852
---------- Step 20000 ----- CPU = 0.4125 (sec) ----------
TotEng = -3.0006 KinEng = 0.0537 Temp = 0.0381
PotEng = -3.0544 E_bond = 0.0000 E_angle = 0.0000
E_dihed = 0.0000 E_impro = 0.0000 E_vdwl = -3.0544
E_coul = 0.0000 E_long = 0.0000 Press = 0.0285
Volume = 30.0537
Loop time of 0.4126 on 1 procs for 20000 steps with 17 atoms

Pair time (%) = 0.105628 (25.6006)
Neigh time (%) = 0.000360489 (0.0873701)
Comm time (%) = 0.00351048 (0.850818)
Outpt time (%) = 0.210592 (51.0402)
Other time (%) = 0.092509 (22.421)

Nlocal: 17 ave 17 max 17 min
Histogram: 1 0 0 0 0 0 0 0 0 0
Nghost: 0 ave 0 max 0 min
Histogram: 1 0 0 0 0 0 0 0 0 0
Neighs: 117 ave 117 max 117 min
Histogram: 1 0 0 0 0 0 0 0 0 0

Total # of neighbors = 117
Ave neighs/atom = 6.88235
Neighbor list builds = 38
Dangerous builds = 0
System init for write_restart ...

If you get this then you have successfully run a LAMMPS simulation.