Modifying & Extending LAMMPS

Steve Plimpton
Sandia National Labs
sjplimp@sandia.gov

LAMMPS Users and Developers Workshop
International Centre for Theoretical Physics (ICTP)
March 2014 - Trieste, Italy

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Presentation: SAND2014-2243C

Resources for modifying LAMMPS

- Before you start writing code:
  - be familiar with what is already in LAMMPS
    - http://lammps.sandia.gov/doc/Section_commands.html

Resources for modifying LAMMPS

- Before you start writing code:
  - be familiar with what is already in LAMMPS
    - http://lammps.sandia.gov/doc/Section_commands.html
  - search the mail list
    - http://lammps.sandia.gov/mail.html
    - http://lammps.sandia.gov/threads/topics.html
    - google: lammps-users thermostat Lowe
      1st hit: lammps.sandia.gov/threads/msg20748.html
      2nd hit: SourceForge.net: LAMMPS: lammps-users
      Ad hit: Thermostats at Lowe's (www.lowes.com)
  - post a "how can I do this" message to the mail list
    - email to lammps-users@lists.sourceforge.net

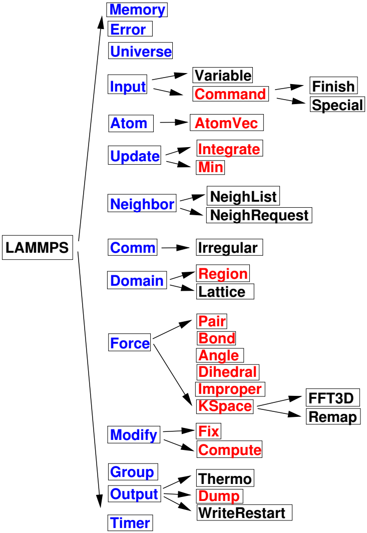
Resources for modifying LAMMPS

- Before you start writing code:
  - be familiar with what is already in LAMMPS
    - http://lammps.sandia.gov/doc/Section_commands.html
  - search the mail list
    - http://lammps.sandia.gov/mail.html
    - http://lammps.sandia.gov/threads/topics.html
    - google: lammps-users thermostat Lowe
      1st hit: lammps.sandia.gov/threads/msg20748.html
      2nd hit: SourceForge.net: LAMMPS: lammps-users
      Ad hit: Thermostats at Lowe's (www.lowes.com)
  - post a "how can I do this" message to the mail list
    - email to lammps-users@lists.sourceforge.net
- Section in manual: Modifying & Extending LAMMPS
  - doc/Section_modify.html

Resources for modifying LAMMPS

- Before you start writing code:
  - be familiar with what is already in LAMMPS
    - http://lammps.sandia.gov/doc/Section_commands.html
  - search the mail list
    - http://lammps.sandia.gov/mail.html
    - http://lammps.sandia.gov/threads/topics.html
    - google: lammps-users thermostat Lowe
      1st hit: lammps.sandia.gov/threads/msg20748.html
      2nd hit: SourceForge.net: LAMMPS: lammps-users
      Ad hit: Thermostats at Lowe's (www.lowes.com)
  - post a "how can I do this" message to the mail list
    - email to lammps-users@lists.sourceforge.net
- Section in manual: Modifying & Extending LAMMPS
  - doc/Section_modify.html
- Developers manual (brief!)
  - doc/Developer.pdf
  - diagram of class hierarchy
  - pseudo-code & explanation of how a timestep works

Class structure of LAMMPS



- LAMMPS itself is a class
  - can be instantiated multiple times
  - has library interface
  - callable via C++, C, Fortran, Python
- Blue are core classes
  - visible anywhere in LAMMPS
- Red are style classes
  - one parent class
  - many child classes

Source files

- Rule of thumb: every input script command has corresponding class and corresponding file name
  - run command ⇒ Run class ⇒ run.cpp + run.h
  - pair_style lj/cut command ⇒ PairLJCut class ⇒ pair_lj_cut.cpp/h
- Src directory
  - core classes are all here
  - many style classes also here
- Package sub-directories (type make package to see)
  - package = group of related style classes
  - src/KSPACE = long-range Coulombic solvers
  - src/USER-OMP = OpenMP versions of many classes (Axel)
  - two flavors: standard (26) and user (13)
- Lib directory
  - some packages require auxiliary libraries
  - those included in LAMMPS are under lib
  - examples: lib/gpu, lib/meam, lib/colvars (Axel)

Core classes

See doc/Developer.pdf for more details

- Memory = memory allocation of 1d, 2d, etc arrays
- Error = error and warning messages
- Universe = partition procs ⇒ multiple "worlds", one per sim
- Input = read input script, variables, added commands
- Atom = per-particle data
- Update = dynamics and minimization
- Neighbor = build neighbor lists
- Comm = inter-processor communication
- Domain = simulation box and geometric regions
- Force = potentials (pair, bond, angle, etc, KSpace)
- Modify = fixes and computes
- Group = collections of particles
- Output = thermodynamics, dump files, restart files
- Timer = timings statistics

Core classes

See doc/Developer.pdf for more details

- Memory = memory allocation of 1d, 2d, etc arrays
- Error = error and warning messages
- Universe = partition procs ⇒ multiple "worlds", one per sim
- Input = read input script, variables, added commands
- Atom = per-particle data
- Update = dynamics and minimization
- Neighbor = build neighbor lists
- Comm = inter-processor communication
- Domain = simulation box and geometric regions
- Force = potentials (pair, bond, angle, etc, KSpace)
- Modify = fixes and computes
- Group = collections of particles
- Output = thermodynamics, dump files, restart files
- Timer = timings statistics

Look at header files (src/domain.h) to understand core classes and LAMMPS generally

Style classes

90% of source code is extensions via 14 styles

See src/style*.h or grep CLASS *.h

Style classes

90% of source code is extensions via 14 styles
See src/style*.h or grep CLASS *.h

Easy for developers and users to add new features:

- particle types = atom style
- force fields = pair, bond, angle, dihedral, improper styles
- long range = kspace style
- fix = fix style = BC, constraint, time integration, ...
- diagnostics = compute style
- geometric region = region style
- integrator = integrate style (Verlet, rRESPA)
- minimizer = min style
- snapshot output =
- dump style
- input command = command style = read_data, velocity, run

Other code details

- Pointers = ultimate base class
  - all classes (except LAMMPS) derive from it
  - holds pointers to all core classes
  - enables easy access anywhere in code
    - domain→xprd for x box-length
- Everything inside LAMMPS_NS namespace
  - no external (global) variables
  - allows multiple instantiations of LAMMPS

Other code details

- Pointers = ultimate base class
  - all classes (except LAMMPS) derive from it
  - holds pointers to all core classes
  - enables easy access anywhere in code
    - domain→xprd for x box-length
- Everything inside LAMMPS_NS namespace
  - no external (global) variables
  - allows multiple instantiations of LAMMPS
- MPI communicators
  - pass in from main() or thru library interface as world
    - enables a LAMMPS instantiation to run on any set of procs
  - universe class partitions allocation into multiple worlds
    - enables multiple simulations to run simultaneously
- C++ vs Fortran
  - pre-2004 LAMMPS was in Fortran
  - re-wrote in C++ for flexibility in adding new features
  - very little fancy C++ (templating, STL, etc)
  - core kernels are C-like, so coding style is really OO C

4 ways to extend LAMMPS

1. Add new styles
    - sky is the limit!
2. Add code to existing files
3. Add new fields to data file as atom properties
4. Add methods to the library interface
    - really “extending” external to LAMMPS

Extending LAMMPS via styles

Again, 90% of source code is extensions via 14 styles

- Enabled by C++
  - virtual parent class defines interface rest of LAMMPS uses
  - style = new child class implementing a few methods

Extending LAMMPS via styles

Again, 90% of source code is extensions via 14 styles

- Enabled by C++
  - virtual parent class defines interface rest of LAMMPS uses
  - style = new child class implementing a few methods
- In theory:
  - just add new *.cpp and *.h file to src and re-compile
  - your new class will work with all LAMMPS functionality
  - your new class won't break anything else
  - in practice, theory and practice are not always the same

Extending LAMMPS via styles

Again, 90% of source code is extensions via 14 styles

- Enabled by C++
  - virtual parent class defines interface rest of LAMMPS uses
  - style = new child class implementing a few methods
- In theory:
  - just add new *.cpp and *.h file to src and re-compile
  - your new class will work with all LAMMPS functionality
  - your new class won't break anything else
  - in practice, theory and practice are not always the same

Now discuss nuts & bolts, then show 5 examples

How to write a new style

See doc/Section_modify.html for overview and key methods

How to write a new style

See doc/Section_modify.html for overview and key methods

- Find an existing style that does something similar
  - ask on mail list or send developers an email
  - especially important if you want to do something complex
    - does functionality you want already exist?
    - is it a good idea to do this in LAMMPS?
    - will it be parallel?
    - can advise you as to possible gotchas

How to write a new style

See doc/Section_modify.html for overview and key methods

- Find an existing style that does something similar
    - ask on mail list or send developers an email
    - especially important if you want to do something complex
        - does functionality you want already exist?
        - is it a good idea to do this in LAMMPS?
        - will it be parallel?
        - can advise you as to possible gotchas
- Decide which style is most appropriate
    - computes calculate at one timestep
    - fixes can alter something during timestep
    - fixes can maintain info from timestep to timestep

How to write a new style

See doc/Section_modify.html for overview and key methods

- Find an existing style that does something similar
  - ask on mail list or send developers an email
  - especially important if you want to do something complex
    - does functionality you want already exist?
    - is it a good idea to do this in LAMMPS?
    - will it be parallel?
    - can advise you as to possible gotchas
- Decide which style is most appropriate
  - computes calculate at one timestep
  - fixes can alter something during timestep
  - fixes can maintain info from timestep to timestep
- Understand how that style works and is structured
  - examine parent class header file (e.g. pair.h)
  - learn what methods it supports (doc/Section_modify.html)
  - look at other *.cpp and *.h files of that style
  - if you get stuck, post to mail list

How to write a new pair style

Find a similar pair style ...

- Flags in constructor: see pair.h
  - manybody_flag, single_enable, respa_enable, comm_forward, etc

How to write a new pair style

Find a similar pair style ...

- Flags in constructor: see pair.h
  - manybody_flag, single_enable, respa_enable, comm_forward, etc
- compute() method
  - loop over atoms and neighbors
  - calculate energy and forces
- settings() method
  - pair_style lj/cut cutoff
- coeff() method
  - pair_coeff I J epsilon sigma

How to write a new pair style

Find a similar pair style ...

- Flags in constructor: see pair.h
  - manybody_flag, single_enable, respa_enable, comm_forward, etc
- compute() method
  - loop over atoms and neighbors
  - calculate energy and forces
- settings() method
  - pair_style lj/cut cutoff
- coeff() method
  - pair_coeff I J epsilon sigma
- init_one() method
  - pre-compute all needed factors, symmetrize I,J = J,I
- write_restart() and read_restart() methods
- single() method
  - energy/force for one I,J pair of particles

How to write a new compute style

Find a similar compute ...

- What will the compute produce?
    - global or per-atom or local values
    - scalar or vector or array
    - see doc/Section_howto 6.15
    - see compute.h for what flags to set

How to write a new compute style

Find a similar compute ...

- What will the compute produce?
  - global or per-atom or local values
  - scalar or vector or array
  - see doc/Section_howto 6.15
  - see compute.h for what flags to set
- Corresponding methods to implement:
  - compute_scalar() = single global value
    - compute temp
  - compute_vector() = few values
    - compute group/group for force components
  - compute_array() = array of few values like
    - compute rdf
  - compute_peratom() = one or more values per atom
    - compute coord/atom, compute displace/atom
  - compute_local() = one or more values per pair, bond, etc
    - compute pair/local, compute bond/local

Fixes allow tailoring of timestep

In hindsight, best feature of LAMMPS for flexibility
Allows control of what happens when within each timestep

Loop over timesteps:

```
    communicate ghost atoms

    build neighbor list (once in a while)
    compute forces
    communicate ghost forces



    output to screen and files
```

Fixes allow tailoring of timestep

In hindsight, best feature of LAMMPS for flexibility
Allows control of what happens when within each timestep

```
Loop over timesteps:
   fix initial                     NVE, NVT, NPT, rigid-body integration
   communicate ghost atoms
   fix neighbor                                          insert particles
   build neighbor list (once in a while)
   compute forces
   communicate ghost forces
   fix force               SHAKE, langevin drag, wall, spring, gravity
   fix final                       NVE, NVT, NPT, rigid-body integration
   fix end                            volume & T rescaling, diagnostics
   output to screen and files
```

How to write a new fix style

Find a similar fix ...

- setmask() method, e.g. for fix nve:

```
int mask = 0;
mask |= INITIAL_INTEGRATE;
mask |= FINAL_INTEGRATE;
return mask;
```

How to write a new fix style

Find a similar fix ...

- setmask() method, e.g. for fix nve:

  ```
  int mask = 0;
  mask |= INITIAL_INTEGRATE;
  mask |= FINAL_INTEGRATE;
  return mask;
  ```

- Corresponding methods to implement:
  - initial_integrate()
    - fix nvt, nvt, npt, rigid = first half of Verlet update
  - pre_exchange()
    - fix deposit, evaporate = insert, remove particles
  - post_force()
    - fix addforce, shake, fix wall = adjust or constrain forces
  - final_integrate()
    - second half of Verlet update
  - end_of_step()
    - fix deform, fix ave/time = change system, diagnostics

How to write a new fix style (continued)

- Fixes can ...
  - request a neighbor list (so can compute)
  - perform ghost-atom communication (so can compute)
  - store values that migrate with atoms
    - grow_arrays(), copy_arrays(), pack_exchange(), unpack_exchange()
  - write/read info to/from restart file
    - fix nvt (global), fix store/state (per-atom)

How to write a new fix style (continued)

- Fixes can ...
  - request a neighbor list (so can compute)
  - perform ghost-atom communication (so can compute)
  - store values that migrate with atoms
    - grow_arrays(), copy_arrays(), pack_exchange(), unpack_exchange()
  - write/read info to/from restart file
    - fix nvt (global), fix store/state (per-atom)
- Will the fix produce any output?
  - global or per-atom or local values
    - fix nvt stores thermostat energy contribution
  - scalar or vector or array
  - see doc/Section_howto 6.15
  - same flags to set in fix.h

How to write a new atom style

Don't do it, if can avoid it ...

- See new fix property/atom command
  - add a molecule ID to style without one
    - example: treat granular clusters as rigid bodies
    - instead of atom_style hybrid sphere bond
  - add arbitrary i_myflag, d_sx d_sy d_sz
  - access the per-atom values in other classes

How to write a new atom style

Don't do it, if can avoid it ...

- See new fix property/atom command
  - add a molecule ID to style without one
    - example: treat granular clusters as rigid bodies
    - instead of atom_style hybrid sphere bond
  - add arbitrary i_myflag, d_sx d_sy d_sz
  - access the per-atom values in other classes
- See new atom_style body command
  - useful for "particles" with internal state
  - example: aspherical particle with sub-particles
  - example: aspherical particle with surface grid
  - end up writing a small body style, not a large atom style
  - see doc/body.html for details

If you really need to write a new atom style (advanced)

Study an existing atom style ...

- Flags in constructor: see atom_vec.h
    - molecular, mass_type, size_forward, size_data_atom, etc

If you really need to write a new atom style (advanced)

Study an existing atom style ...

- Flags in constructor: see atom_vec.h
    - molecular, mass_type, size_forward, size_data_atom, etc
- grow() method - allocates all per-atom arrays
- (un)pack_comm() method - communicate every step
- (un)pack_border() method - communicate every re-neighbor
- (un)pack_exchange() method - migrate info with atom
- create_atom() method - create one atom
- data_atom() method - read atom from data file

If you really need to write a new atom style (advanced)

Study an existing atom style ...

- Flags in constructor: see atom_vec.h
  - molecular, mass_type, size_forward, size_data_atom, etc
- grow() method - allocates all per-atom arrays
- (un)pack_comm() method - communicate every step
- (un)pack_border() method - communicate every re-neighbor
- (un)pack_exchange() method - migrate info with atom
- create_atom() method - create one atom
- data_atom() method - read atom from data file
- And a dozen others ...
  - variants to work in atom_style hybrid mode

Questions?

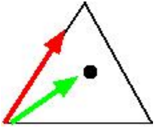
Take a break and stretch ...

Five examples of LAMMPS style extensions

- Triangular regions: region tri
- Molecule size/shape: compute rg/molecule
- Solvent evaporation: fix evaporate
- Grain boundary migration: fix orient/fcc
- Shock-induced explosive detonation: fix wall/reflect

#1 - Triangular regions

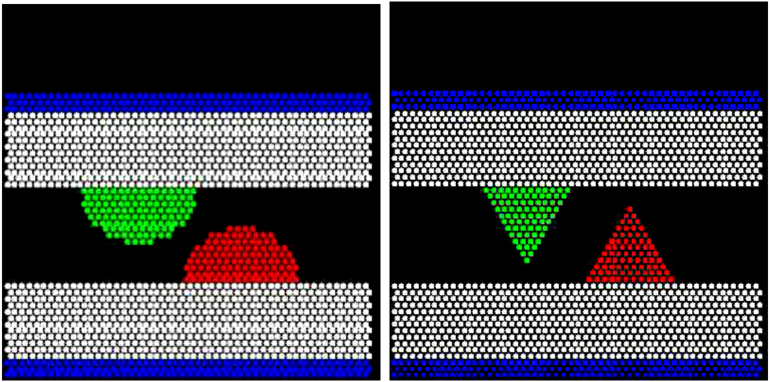
- Derived class: RegionTri in region_tri.cpp/h
- Header file:
  #ifdef REGION_CLASS
  RegionStyle(tri,RegTri)
  #else
- Input script syntax: (just for 2d problems)
  - region bump tri x1 y1 x2 y2 x3 y3
- RegionTri(int narg, char **arg)
  - reads arguments: x1 y1 x2 y2 x3 y3
  - determines bounding box
- inside(double x, double y, double z) method
  - determine if (x,y) is inside triangle
  - 3 positive cross products ⇒ inside
- ∼35 lines of code



Friction example

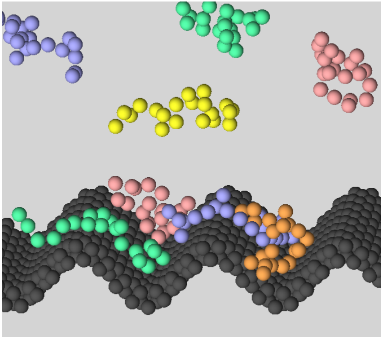
Substitute (twice):

```
region lo-asperity sphere 32 7 0 8
region lo-asperity tri 26 7 32 14 38 7
```



#2 - Molecule size/shape

- Stick-slip flow on corrugated surfaces
- Nikolai Priezjev group at Michigan State U
- *Niavarani and Priezjev, J Chem Phys, 129, 144902 (2008)*



- Flow is function of corrugation wavelength and chain length
- Monitor shape and motion of chains

Compute gyration/molecule for R_g of each polymer chain

- Input script:
  `compute id all gyration/molecule {tensor}`
- compute_vector() method (40 lines, one value/molecule):

```
for (int i = 0; i < nlocal; i++)
  if (mask[i] & groupbit) {
    imol = molecule[i];
    domain->unmap(x[i],image[i],unwrap);
    dx = unwrap[0] - comall[imol][0];
    dy = unwrap[1] - comall[imol][1];
    dz = unwrap[2] - comall[imol][2];
    massone = mass[type[i]];
    rg[imol] += (dx*dx + dy*dy + dz*dz) * massone;
  }
  MPI_Allreduce(rg,vector,nmolecules,...);
```

Compute gyration/molecule for R_g of each polymer chain

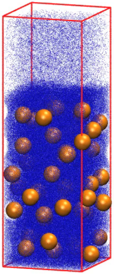
- Input script:
  ```
  compute id all gyration/molecule {tensor}
  ```
- compute_vector() method (40 lines, one value/molecule):

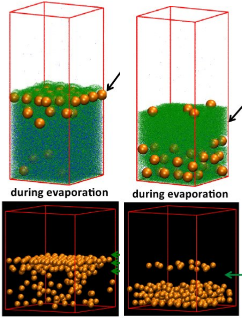
```
for (int i = 0; i < nlocal; i++)
  if (mask[i] & groupbit) {
    imol = molecule[i];
    domain->unmap(x[i],image[i],unwrap);
    dx = unwrap[0] - comall[imol][0];
    dy = unwrap[1] - comall[imol][1];
    dz = unwrap[2] - comall[imol][2];
    massone = mass[type[i]];
    rg[imol] += (dx*dx + dy*dy + dz*dz) * massone;
  }
  MPI_Allreduce(rg,vector,nmolecules,...);
```

- For shape, compute inertia/molecule is similar logic

#3 - Solvent evaporation

- Nanoparticle ordering in polymers w/ solvent evaporation
- *S Cheng & G Grest, J Chem Phys, 138, 064701 (2013)*
- *Spring MRS meeting, 2013*





- Evaporate solvent at controlled rate above L/V interface
- Ordering is function of NP/polymer interaction strength

Fix evaporate removes solvent at specified rate

- Input script:

  ```
  fix id solvent evaporate
      N M topbox 38277 {molecule yes}
  ```

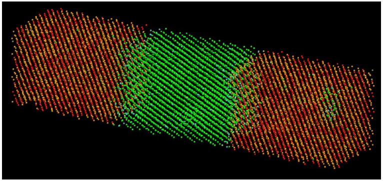
- pre_exchange() method

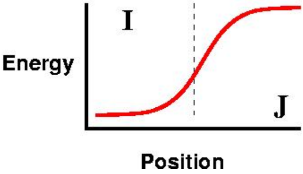
```
identify atoms in region volume
pick random subset (consistent across procs)
delete from system
also remove molecules the deleted particles are in
```

- ∼200 lines of code (molecules add some complexity)

#4 - Grain boundary migration

K Janssens, et al, Nature Materials, 5, 124 (2006)





$$\xi_i = \sqrt{\sum_j (r_j - r_j^I)^2}$$

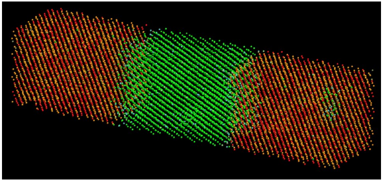
$$\xi_{IJ} = \sqrt{\sum_j (r_j^I - r_j^J)^2}$$

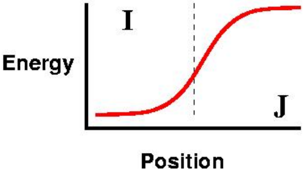
$$u_\xi(i) = \begin{cases} 0 & \xi_i < \xi_l \\ V \sin\omega_i \quad \text{with } \omega_i = \dfrac{\pi}{2}\dfrac{\xi_i - \xi_l}{\xi_h - \xi_l} & \xi_l < \xi_i < \xi_h \\ V & \xi_h < \xi_i \end{cases}$$

- Add synthetic energy/force as function of mis-orientation
- Drives atoms near boundary from orientation I to J

#4 - Grain boundary migration

K Janssens, et al, Nature Materials, 5, 124 (2006)





$$\xi_i = \sqrt{\sum_j (r_j - r_j^I)^2}$$

$$\xi_{IJ} = \sqrt{\sum_j (r_j^I - r_j^J)^2}$$

$$u_\xi(i) = \begin{cases} 0 & \xi_i < \xi_l \\ V \sin\omega_i \quad \text{with } \omega_i = \dfrac{\pi}{2}\dfrac{\xi_i - \xi_l}{\xi_h - \xi_l} & \xi_l < \xi_i < \xi_h \\ V & \xi_h < \xi_i \end{cases}$$

- Add synthetic energy/force as function of mis-orientation
- Drives atoms near boundary from orientation I to J
- Mobility $\propto$ migration velocity / driving force
- Extract accurate mobility from short simulation

Build a bi-crystal

Input script commands:

```
region lower box EDGE EDGE EDGE EDGE EDGE 20.0
region upper box EDGE EDGE EDGE EDGE 20.0 EDGE

lattice fcc 4.04 origin 0 20 0 orient x -3 1 0 ...
create_atoms 1 region lower

lattice fcc 4.04 origin 0 20 0 orient x 3 1 0 ...
create_atoms 1 region upper

delete_atoms overlap 0.5 all all
```

Fix orient/fcc to impose driving force

- 2 files: src/fix_orient_fcc.cpp and fix_orient_fcc.h
- Request full neighbor list, every timestep:

```
int irequest = neighbor->request((void *) this);
neighbor->requests[irequest]->pair = 0;
neighbor->requests[irequest]->fix = 1;
neighbor->requests[irequest]->half = 0;
neighbor->requests[irequest]->full = 1;
```

Post_force() method for fix orient/fcc

```
double loop over atoms and neighbors:
  compute $R_{ij}$ and add to list
  sort list to find 12 nearest neighbors (fcc)

loop over atoms:
  compute contributions from 12 neighbors
  derivative of energy → forces on I and J atoms

communicate partial forces induced on ghost atoms

double loop over atoms and neighbors:
  compute full orientation force on each I atom
```

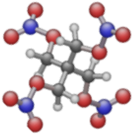
Post_force() method for fix orient/fcc

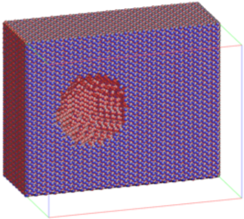
```
double loop over atoms and neighbors:
  compute $R_{ij}$ and add to list
  sort list to find 12 nearest neighbors (fcc)

loop over atoms:
  compute contributions from 12 neighbors
  derivative of energy → forces on I and J atoms

communicate partial forces induced on ghost atoms

double loop over atoms and neighbors:
  compute full orientation force on each I atom
```

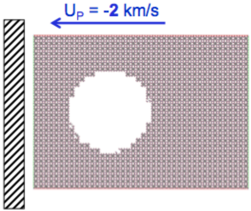
- LAMMPS provides method to perform communication
- ∼250 lines of code

#5 - Shock-induced detonation of explosives

- *R Shan & A Thompson, March APS meeting (2013)*
- PETN is a powerful high explosive
- Simulate “slow” shock wave passing thru PETN crystal

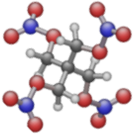


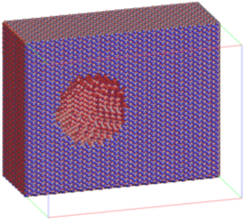


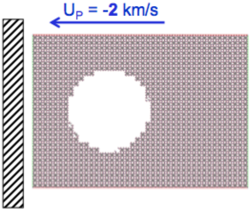


#5 - Shock-induced detonation of explosives

- *R Shan & A Thompson, March APS meeting (2013)*
- PETN is a powerful high explosive
- Simulate "slow" shock wave passing thru PETN crystal







- Use a reactive force field (ReaxFF)
  - detonation is triggered by onset of exothermic reactions
- Quantify detonation sensitivity to orientation, defects, impurities ... a safety issue

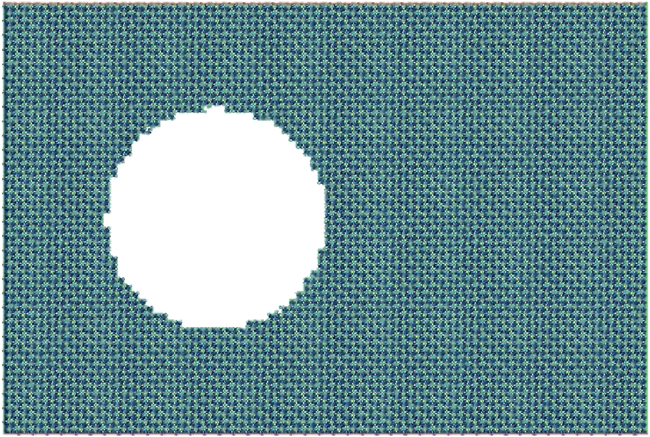
Create a void in PETN crystal

Input script commands:

```
read_data data.petn.molecule
replicate 100 50 50

region void sphere 20.0 30.0 30.0 5.0
delete_atoms all region void
```

Largest void size = 20 nm

8.9M atoms (60x40x40 nm)
10 psec (20K steps, 100 hours on 4096 cores)



Post_integrate() method for fix wall/reflect command

```
for (int m = 0; m < nwall; m++)
  coord = current wall position (fixed or variable)
  dim = wallwhich[m] / 2; side = wallwhich[m] % 2;

for (i = 0; i < nlocal; i++)
  if (side == 0)
    if (x[i][dim] < coord)
      x[i][dim] = coord + (coord - x[i][dim]);
      v[i][dim] = -v[i][dim];
  else
    if (x[i][dim] > coord)
      x[i][dim] = coord - (x[i][dim] - coord);
      v[i][dim] = -v[i][dim];
```

Post_integrate() method for fix wall/reflect command

```
for (int m = 0; m < nwall; m++)
  coord = current wall position (fixed or variable)
  dim = wallwhich[m] / 2; side = wallwhich[m] % 2;

for (i = 0; i < nlocal; i++)
  if (side == 0)
    if (x[i][dim] < coord)
      x[i][dim] = coord + (coord - x[i][dim]);
      v[i][dim] = -v[i][dim];
  else
    if (x[i][dim] > coord)
      x[i][dim] = coord - (x[i][dim] - coord);
      v[i][dim] = -v[i][dim];
```

- Entire fix = ∼200 lines of code

Fix reaxc/species command for molecule statistics

- Written by Ray Shan (Sandia)
- Molecules in ReaxFF and a shock explosion are dynamic
  - not defined by permanent bonds, angles, etc
  - defined by instantaneous bond-order parameters
- Useful to know numbers/locations/atoms of molecules at any timestep, on-the-fly

Fix reaxc/species command for molecule statistics

- Written by Ray Shan (Sandia)
- Molecules in ReaxFF and a shock explosion are dynamic
    - not defined by permanent bonds, angles, etc
    - defined by instantaneous bond-order parameters
- Useful to know numbers/locations/atoms of molecules at any timestep, on-the-fly
- Compute cluster/atom flags clusters based on cutoff
    - each atom starts as own cluster
    - walk outward, merging clusters with lower atom ID
    - parallel communication when clusters overlap proc domains
- Use same logic to merge based on bond-order criterion
- Compile molecule stats and write details to file

Fix reaxc/species command for molecule statistics

- Written by Ray Shan (Sandia)
- Molecules in ReaxFF and a shock explosion are dynamic
  - not defined by permanent bonds, angles, etc
  - defined by instantaneous bond-order parameters
- Useful to know numbers/locations/atoms of molecules at any timestep, on-the-fly
- Compute cluster/atom flags clusters based on cutoff
  - each atom starts as own cluster
  - walk outward, merging clusters with lower atom ID
  - parallel communication when clusters overlap proc domains
- Use same logic to merge based on bond-order criterion
- Compile molecule stats and write details to file
- Entire fix = $\sim$1000 lines of code

Extending LAMMPS by adding to existing files

3 cases where this is straight-forward:

Extending LAMMPS by adding to existing files

3 cases where this is straight-forward:

1. Adding keywords to thermo_style output
    - see thermo.cpp
    - complicated calculation better done as new Compute

Extending LAMMPS by adding to existing files

3 cases where this is straight-forward:

1. Adding keywords to thermo_style output
   - see thermo.cpp
   - complicated calculation better done as new Compute
2. Adding new functions to equal-style and atom-style variables
   - see variable.cpp
   - math functions, special functions, math operators, etc
   - make sure you follow syntax rules for args of similar functions

Extending LAMMPS by adding to existing files

3 cases where this is straight-forward:

1. Adding keywords to thermo_style output
    - see thermo.cpp
    - complicated calculation better done as new Compute
2. Adding new functions to equal-style and atom-style variables
    - see variable.cpp
    - math functions, special functions, math operators, etc
    - make sure you follow syntax rules for args of similar functions
3. Adding keywords for per-atom fields
    - only needed if write new atom style
    - see compute_property_atom.cpp
    - allows use of field in all other commands
        - dump, fix ave/spatial, atom-style variables, etc

Extending LAMMPS by adding to existing files

3 cases where this is straight-forward:

1. Adding keywords to thermo_style output
   - see thermo.cpp
   - complicated calculation better done as new Compute
2. Adding new functions to equal-style and atom-style variables
   - see variable.cpp
   - math functions, special functions, math operators, etc
   - make sure you follow syntax rules for args of similar functions
3. Adding keywords for per-atom fields
   - only needed if write new atom style
   - see compute_property_atom.cpp
   - allows use of field in all other commands
     - dump, fix ave/spatial, atom-style variables, etc

In each case, look for customize comments in appropriate src file

Adding new fields to data file (advanced)

- New header lines and/or new sections
  - 1500 multistates
  - Multistates
  - 1 27 ...
  - ...
  - 1500 13 ...
- Previously required extensions to read_data.cpp

Adding new fields to data file (advanced)

- New header lines and/or new sections
  - 1500 multistates
  - Multistates
  - 1 27 ...
  - ...
  - 1500 13 ...
- Previously required extensions to read_data.cpp
- Can now be done in a fix
  - read_data data.poly fix ID multistates Multistates ...
  - can read from data file and store per-atom info
  - virtual void read_data_header(char *);
  - virtual void read_data_section(char *, int, char *);
  - virtual bigint read_data_skip_lines(char *);

Adding new fields to data file (advanced)

- New header lines and/or new sections
  - 1500 multistates
  - Multistates
  - 1 27 ...
  - ...
  - 1500 13 ...
- Previously required extensions to read_data.cpp
- Can now be done in a fix
  - read_data data.poly fix ID multistates Multistates ...
  - can read from data file and store per-atom info
  - virtual void read_data_header(char *);
  - virtual void read_data_section(char *, int, char *);
  - virtual bigint read_data_skip_lines(char *);
- See fix property/atom for a working example
- CMAP 5-body interactions are being implemented this way

Using LAMMPS thru its library interface

See Section_howto.html 6.19 and Section_python.html in manual
See src/library.cpp and src/library.h

```
void lammps_open(int, char **, MPI_Comm, void **)
void lammps_close(void *)
void lammps_file(void *, char *)
char *lammps_command(void *, char *)

void *lammps_extract_global(void *, char *)
void *lammps_extract_atom(void *, char *)
void *lammps_extract_compute(void *, char *, int, int)
void *lammps_extract_fix(void *, char *,int,int,int,int)
void *lammps_extract_variable(void *, char *, char *)
int lammps_get_natoms(void *)
void lammps_get_coords(void *, double *)
void lammps_put_coords(void *, double *)
```

Example with GnuPlot

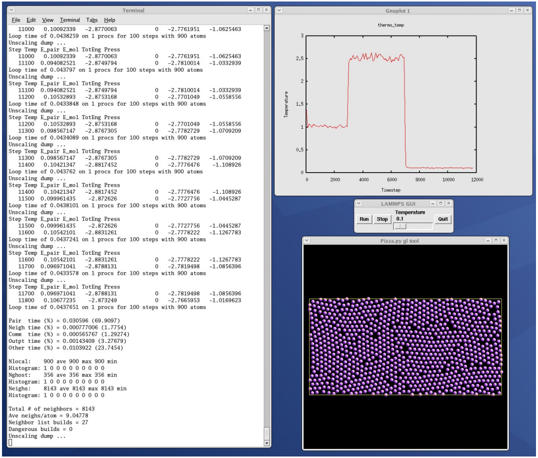
See examples/COUPLE/simple for C, C++, Fortran
See python/examples for Python, Pizza.py for GnuPlot wrapper

```
% python plot.py in.lammps Nfreq Nsteps compute-ID

from gnu import gnu
from lammps import lammps
lmp = lammps()
lmp.file(infile)
lmp.command("thermo %d" % Nfreq)

lmp.command("run 0 pre yes post no")
value = lmp.extract_compute(computeID,0,0)
ntimestep = 0
xaxis = [ntimestep]
yaxis = [value]
```

Example with GnuPlot (continued)

```
if me == 0:
  gn = gnu()
  gn.plot(xaxis,yaxis)
  gn.xrange(0,nsteps)
  gn.title(computeID,"Timestep","Temperature")

while ntimestep < Nsteps:
  lmp.command("run %d pre no post no" % Nfreq)
  ntimestep += nfreq
  value = lmp.extract_compute(computeID,0,0)
  xaxis.append(ntimestep)
  yaxis.append(value)
  if me == 0:  gn.plot(xaxis,yaxis)

lmp.command("run 0 pre no post yes")
```

What it produces, in real time



0.1

```
   11000  0.10092339   -2.8770063            0   -2.7761951   -1.0625463
Loop time of 0.0438259 on 1 procs for 100 steps with 900 atoms
Unscaling dump ...
Step Temp E_pair E_mol TotEng Press
   11000  0.10092339   -2.8770063            0   -2.7761951   -1.0625463
   11100 0.094082521   -2.8749794            0   -2.7810014   -1.0332939
Loop time of 0.043797 on 1 procs for 100 steps with 900 atoms
Unscaling dump ...
Step Temp E_pair E_mol TotEng Press
   11100 0.094082521   -2.8749794            0   -2.7810014   -1.0332939
   11200  0.10532893   -2.8753168            0   -2.7701049   -1.0558556
Loop time of 0.0433848 on 1 procs for 100 steps with 900 atoms
Unscaling dump ...
Step Temp E_pair E_mol TotEng Press
   11200  0.10532893   -2.8753168            0   -2.7701049   -1.0558556
   11300 0.098567147   -2.8767305            0   -2.7782729   -1.0709209
Loop time of 0.0434089 on 1 procs for 100 steps with 900 atoms
Unscaling dump ...
Step Temp E_pair E_mol TotEng Press
   11300 0.098567147   -2.8767305            0   -2.7782729   -1.0709209
   11400  0.10421347   -2.8817452            0   -2.7776476    -1.108926
Loop time of 0.043762 on 1 procs for 100 steps with 900 atoms
Unscaling dump ...
Step Temp E_pair E_mol TotEng Press
   11400  0.10421347   -2.8817452            0   -2.7776476    -1.108926
   11500 0.099961435    -2.872626            0   -2.7727756   -1.0445287
Loop time of 0.0438101 on 1 procs for 100 steps with 900 atoms
Unscaling dump ...
Step Temp E_pair E_mol TotEng Press
   11500 0.099961435    -2.872626            0   -2.7727756   -1.0445287
   11600  0.10542101   -2.8831261            0   -2.7778222   -1.1267783
Loop time of 0.0437241 on 1 procs for 100 steps with 900 atoms
Unscaling dump ...
Step Temp E_pair E_mol TotEng Press
   11600  0.10542101   -2.8831261            0   -2.7778222   -1.1267783
   11700 0.096971041   -2.8788131            0   -2.7819498   -1.0856396
Loop time of 0.0433578 on 1 procs for 100 steps with 900 atoms
Unscaling dump ...
Step Temp E_pair E_mol TotEng Press
   11700 0.096971041   -2.8788131            0   -2.7819498   -1.0856396
   11800  0.10677235    -2.873249            0   -2.7665953   -1.0169623
Loop time of 0.0437651 on 1 procs for 100 steps with 900 atoms

Pair  time (%) = 0.030596 (69.9097)
Neigh time (%) = 0.000777006 (1.7754)
Comm  time (%) = 0.000565767 (1.29274)
Outpt time (%) = 0.00143409 (3.27679)
Other time (%) = 0.0103922 (23.7454)

Nlocal:    900 ave 900 max 900 min
Histogram: 1 0 0 0 0 0 0 0 0 0
Nghost:    356 ave 356 max 356 min
Histogram: 1 0 0 0 0 0 0 0 0 0
Neighs:    8143 ave 8143 max 8143 min
Histogram: 1 0 0 0 0 0 0 0 0 0

Total # of neighbors = 8143
Ave neighs/atom = 9.04778
Neighbor list builds = 27
Dangerous builds = 0
Unscaling dump ...
```

This includes GUI slider & dump output to Pizza.py GL tool
(or AtomEye or Pymol or VMD) - see python/examples scripts

Extending the LAMMPS library interface

Again, see library.cpp and library.h

- Accessor functions already exist for ...
  - system variables (box, timestep, etc)
  - per-atom pointers (x, v, etc)
  - compute and fix output
  - variable evaluation

Extending the LAMMPS library interface

Again, see library.cpp and library.h

- Accessor functions already exist for ...
  - system variables (box, timestep, etc)
  - per-atom pointers (x, v, etc)
  - compute and fix output
  - variable evaluation
- Accessor functions in library.cpp or atom.h can be augmented
  - one-line addition
  - access a new system variable
  - access a new per-atom property

Extending the LAMMPS library interface

Again, see library.cpp and library.h

- Accessor functions already exist for ...
  - system variables (box, timestep, etc)
  - per-atom pointers (x, v, etc)
  - compute and fix output
  - variable evaluation
- Accessor functions in library.cpp or atom.h can be augmented
  - one-line addition
  - access a new system variable
  - access a new per-atom property
- New functions in library.cpp can ...
  - access any public data within LAMMPS
  - invoke any public methods of any classes
- New functions are limited only by your imagination!

How a timestep works - part 1

Most important class to understand: Verlet ⇒ src/verlet.cpp

How a timestep works - part 1

Most important class to understand: Verlet ⇒ src/verlet.cpp

Look at the run() method (in 3 parts)
See doc/Developer.pdf for more details

```
loop over N timesteps:
  ev_set()
  fix->initial_integrate()
  fix->post_integrate()
  ...
```

How a timestep works - part 2

```
loop over N timesteps:
  ...
  nflag = neighbor->decide()
  if nflag:
    fix->pre_exchange()
    domain->pbc()
    domain->reset_box()
    comm->setup()
    neighbor->setup_bins()
    comm->exchange()
    comm->borders()
    fix->pre_neighbor()
    neighbor->build()
  else
    comm->forward_comm()
  ...
```

How a timestep works - part 3

```
loop over N timesteps:
  ...
  force_clear()
  fix->pre_force()
  pair->compute()
  bond->compute()
  angle->compute()
  dihedral->compute()
  improper->compute()
  kspace->compute()
  comm->reverse_comm()

  fix->post_force()
  fix->final_integrate()
  fix->end_of_step()

  if any output on this step:  output->write()
```

How to get your code added to the LAMMPS distro

- Mail it to us, but first ...
  - see doc/Section_modify.html
  - sub-section: Submitting new features for inclusion in LAMMPS

How to get your code added to the LAMMPS distro

- Mail it to us, but first ...
  - see doc/Section_modify.html
  - sub-section: Submitting new features for inclusion in LAMMPS
- Why release it as part of main LAMMPS?
  - open source philosophy
  - fame and fortune, name on author page and in source code
  - acquire users of your feature
    - find and fix bugs
    - extend its functionality
    - become collaborators

How to get your code added to the LAMMPS distro

- Mail it to us, but first ...
  - see doc/Section_modify.html
  - sub-section: Submitting new features for inclusion in LAMMPS
- Why release it as part of main LAMMPS?
  - open source philosophy
  - fame and fortune, name on author page and in source code
  - acquire users of your feature
    - find and fix bugs
    - extend its functionality
    - become collaborators
- Must provide a doc page as a *.txt file
  - one for every command that appears in input script
  - see similar doc/*.txt file as starting point
  - if needed, equations for doc/Eqs as LaTeX files
  - we auto-convert to HTML (and JPG if needed)

How to get your code added (continued)

- Rule: don't make changes in core of LAMMPS
  1. if you think you need to, talk to developers
  2. the more I need to think, the longer it will take to release
- Suggestion: write your code in the LAMMPS format
  1. easier for everyone to read, maintain
  2. required if you want it in src dir or standard packages

How to get your code added (continued)

- Rule: don't make changes in core of LAMMPS
  1. if you think you need to, talk to developers
  2. the more I need to think, the longer it will take to release
- Suggestion: write your code in the LAMMPS format
  1. easier for everyone to read, maintain
  2. required if you want it in src dir or standard packages
- USER-MISC package
  1. if it compiles, we'll add it (within limits)
  2. don't really care if written in LAMMPS format
  3. you own it, answer Qs, and update it
  4. set of related commands can be an entire USER package
- Commands that link to an external library
  1. must become a package (standard or user)
  2. type "make package" for list

What features do you need for your models?

Happy to brainstorm & discuss this week