# Modifying LAMMPS

Steve Plimpton, sjplimp@sandia.gov

3rd LAMMPS Workshop
August 2013 - Albuquerque, NM

# 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

# Extending LAMMPS by adding to existing files

Look for customize comment in appropriate src file

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

# Extending LAMMPS by adding to existing files

Look for customize comment in appropriate src file

- Adding keywords to thermo_style output
  - see thermo.cpp
  - complicated calculation better done as new Compute
- Adding keywords for per-atom fields
  - see compute_property_atom.cpp
  - allows its use in all other commands
    - dump, fix ave/spatial, atom-style variables, etc

# Extending LAMMPS by adding to existing files

Look for customize comment in appropriate src file

- Adding keywords to thermo_style output
  - see thermo.cpp
  - complicated calculation better done as new Compute
- Adding keywords for per-atom fields
  - see compute_property_atom.cpp
  - allows its use in all other commands
    - dump, fix ave/spatial, atom-style variables, etc
- 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 library interface

See library.cpp

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

# Extending LAMMPS library interface

See library.cpp

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

# Extending LAMMPS library interface

See library.cpp

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

# Adding new fields to data file

- 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

- 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

- 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

# Extending LAMMPS via styles

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

# Extending LAMMPS via styles

90% of source code is extensions via 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 style, bond, angle, dihedral, improper
  - long range = kspace style
  - fix = fix style = BC, constraint, time integration, ...
  - diagnostics = compute style
  - geometric region = region style
  - output = dump style
  - minimizer = min style
  - integrator = integrate style
  - input command = command style = read_data, velocity, run

# Extending LAMMPS via styles (2)

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

# Extending LAMMPS via styles (2)

- 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

# 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
  - redcompute_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
Need 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
Need 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 (2)

- 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 (2)

- 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 this if can avoid it ...

- See new fix property/atom command
  - add a molecule ID to style without one
  - instead of atom_style hybrid sphere bond
  - add arbitrary i_myflag, d_sx d_sy d_sz
  - use the per-atom values in other classes

# How to write a new atom style

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

- See new fix property/atom command
  - add a molecule ID to style without one
  - instead of atom_style hybrid sphere bond
  - add arbitrary i_myflag, d_sx d_sy d_sz
  - use 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

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

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

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

# How to get your code added to the LAMMPS distro

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

# How to get your code added to the LAMMPS distro

- Mail it us, but first ...
  - see doc/Section_modify.html#package
  - 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 us, but first ...
  - see doc/Section_modify.html#package
  - 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 (2)

- 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 (2)

- 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

# That's all

Questions?