I modified LAMMPS, and *you can too*!

An introduction to the LAMMPS source code

Tom Gartner
Computational Chemical Science Center
Department of Chemistry
Princeton University
LAMMPS User Group Meeting 09/11/2019

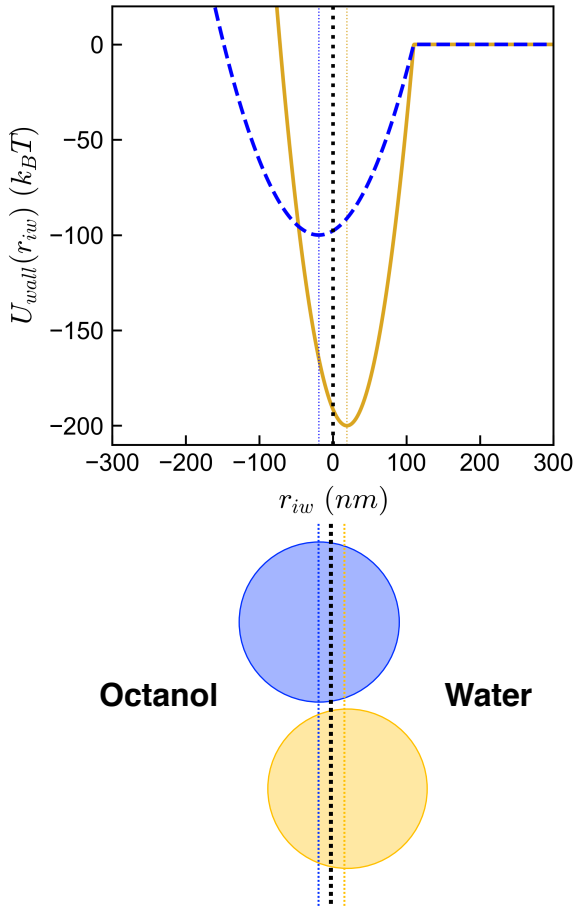
Outline

- Introduction
- LAMMPS source code overview
- LAMMPS modification example: new particle-wall interaction potential
- Resources

Plimpton, S., Fast Parallel Algorithms for Short-Range Molecular Dynamics, *J. Comp. Phys.*, **1995**, *117*, 1-19
https://lammps.sandia.gov/

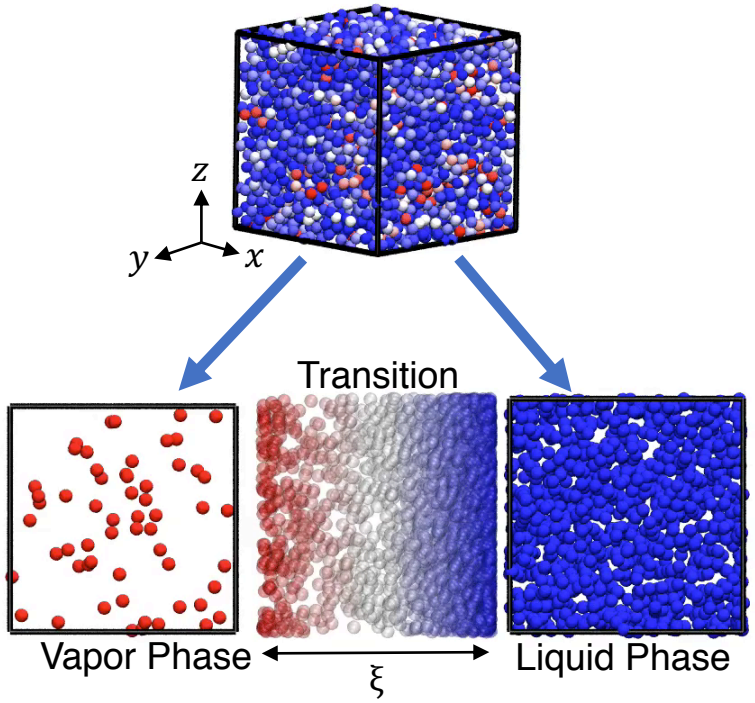
Modifying LAMMPS

Attractive harmonic particle-wall potential



Gartner, III, T.E., Heil, C.M., Jayaraman, A. **2019**, *in preparation*

Gibbs Ensemble Molecular Dynamics



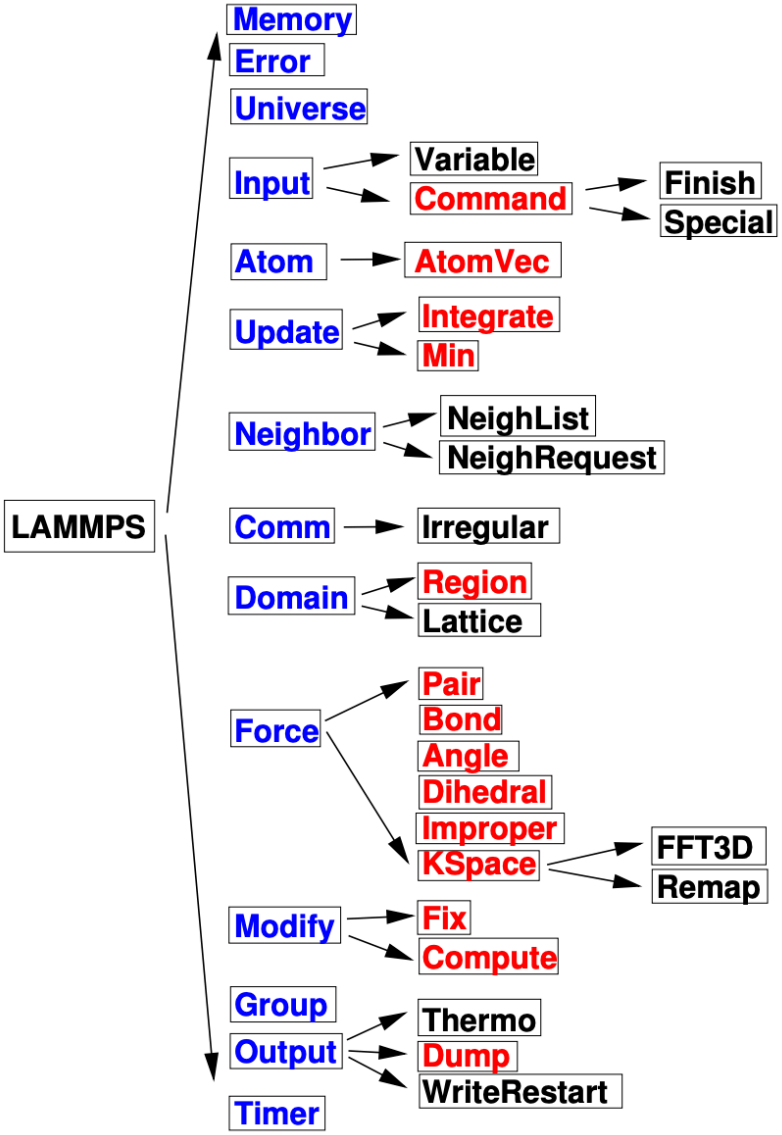
Panagiotopoulos, A.Z. et al. *Mol. Phys.,* **1988,** *63* (4), 527-545.

Kotelyanskii, M.J., Hentschke, R. *Mol. Sim.,* **1996,** *17* (2), 95.

Gartner, III, T.E., Epps, III, T.H., Jayaraman, A. *J. Chem. Theory Comput.,* **2016**, *12* (11), 5501-5510

LAMMPS Code Structure (src directory)

Class-based code structure



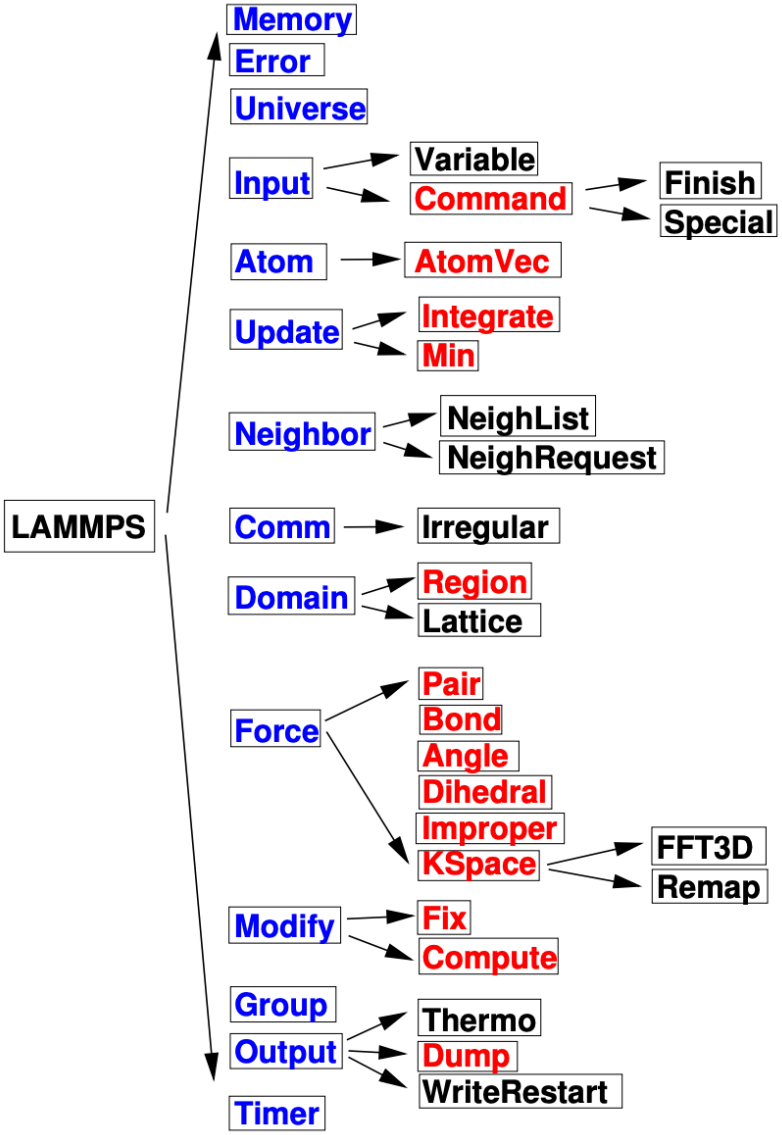
Important classes:

- The Atom class stores all per-atom arrays
- The Update class holds an integrator and a minimizer
- The Neighbor class builds and stores neighbor lists
- The Comm class performs interprocessor communication
- The Domain class stores the simulation box geometry, as well as geometric Regions and any user definition of a Lattice
- The Force class computes various forces between atoms
- The Modify class stores lists of Fix and Compute classes

Text and image from LAMMPS developer's guide:
https://lammps.sandia.gov/doc/Developer.pdf

LAMMPS Code Structure (src directory)

Class-based code structure



Important classes:

- **The Atom class stores all per-atom arrays**
- The Update class holds an integrator and a minimizer
- The Neighbor class builds and stores neighbor lists
- The Comm class performs interprocessor communication
- **The Domain class stores the simulation box geometry, as well as geometric Regions and any user definition of a Lattice**
- **The Force class computes various forces between atoms**
- **The Modify class stores lists of Fix and Compute classes**

Text and image from LAMMPS developer's guide:
https://lammps.sandia.gov/doc/Developer.pdf

LAMMPS Code Structure (src directory)

Almost every LAMMPS input file command has an associated class

```
fix wallhi all wall/lj93 xlo -1.0 1.0 1.0 2.5 units box
```

Modify class → Fix class → Fix_wall class → Fix_wall_lj93 class

```
pair_style lj/cut 2.5
```

Force class → Pair class → Pair_lj_cut class

Text from LAMMPS documentation:
https://lammps.sandia.gov/doc/

Implementation and header files

Each class contains a *header* (*.h) and *implementation* (*.cpp) file

pair_lj_cut.h

```
#ifdef PAIR_CLASS

PairStyle(lj/cut,PairLJCut)

#else

#ifndef LMP_PAIR_LJ_CUT_H
#define LMP_PAIR_LJ_CUT_H

#include "pair.h"

namespace LAMMPS_NS {

class PairLJCut : public Pair {
 public:
  PairLJCut(class LAMMPS *);
  virtual ~PairLJCut();
  virtual void compute(int, int);
  void settings(int, char **);
  void coeff(int, char **);
  void init_style();
  double init_one(int, int);
  void write_restart(FILE *);
  void read_restart(FILE *);
  void write_restart_settings(FILE *);
  void read_restart_settings(FILE *);
  void write_data(FILE *);
  void write_data_all(FILE *);
  double single(int, int, int, int, double, double, double, double &);
  void *extract(const char *, int &);

  void compute_inner();
  void compute_middle();
  void compute_outer(int, int);

 protected:
  double cut_global;
  double **cut;
  double **epsilon,**sigma;
  double **lj1,**lj2,**lj3,**lj4,**offset;
  double *cut_respa;

  virtual void allocate();
};

}

#endif
#endif
```

pair_lj_cut.cpp

```
#include "pair_lj_cut.h"
#include <mpi.h>
#include <cmath>
#include <cstring>
#include "atom.h"
#include "comm.h"
#include "force.h"
#include "neighbor.h"
#include "neigh_list.h"
#include "neigh_request.h"
#include "update.h"
#include "respa.h"
#include "math_const.h"
#include "memory.h"
#include "error.h"

using namespace LAMMPS_NS;
using namespace MathConst;
```

```
void PairLJCut::compute(int eflag, int vflag)
{
  int i,j,ii,jj,inum,jnum,itype,jtype;
  double xtmp,ytmp,ztmp,delx,dely,delz,evdwl,fpair;
  double rsq,r2inv,r6inv,forcelj,factor_lj;
  int *ilist,*jlist,*numneigh,**firstneigh;

  evdwl = 0.0;
  ev_init(eflag,vflag);

  double **x = atom->x;
  double **f = atom->f;
  int *type = atom->type;
  int nlocal = atom->nlocal;
  double *special_lj = force->special_lj;
  int newton_pair = force->newton_pair;

  inum = list->inum;
  ilist = list->ilist;
  numneigh = list->numneigh;
  firstneigh = list->firstneigh;

  // loop over neighbors of my atoms

  for (ii = 0; ii < inum; ii++) {
    i = ilist[ii];
    xtmp = x[i][0];
    ytmp = x[i][1];
```

Structure of a timestep (src/verlet.cpp)

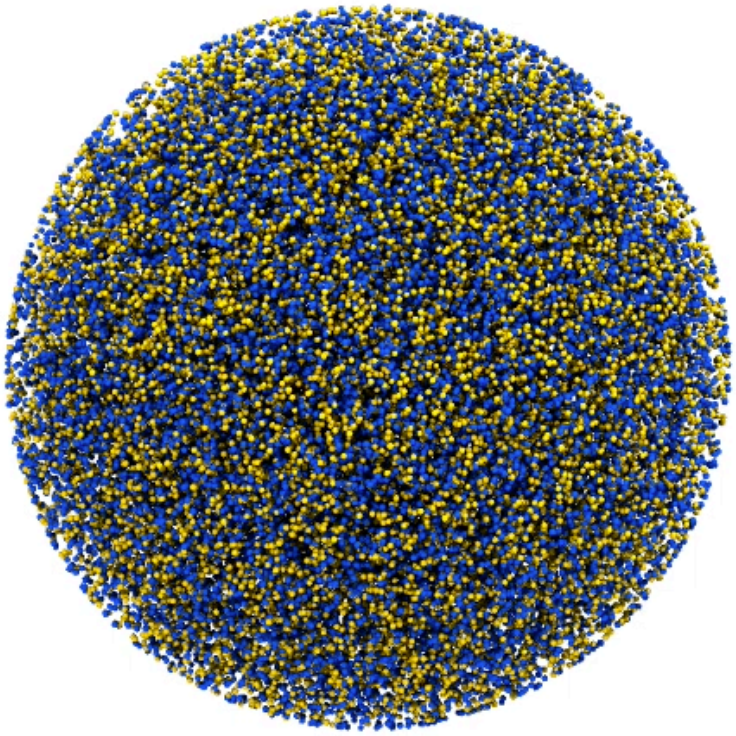
Velocity-Verlet integration algorithm:

```
loop over N timesteps:
  ev_set()

  fix->initial_integrate()
  fix->post_integrate()

  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()
```

```
...
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()
```

Text from LAMMPS developer's guide:
https://lammps.sandia.gov/doc/Developer.pdf

Example: attractive particle-wall potential

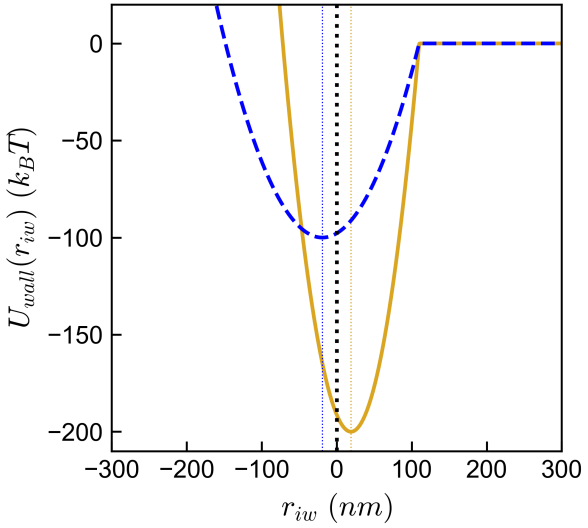


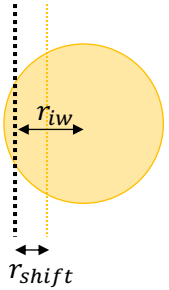
Attractive particle-wall interactions at interior of spherical region of radius R

Assembly of binary nanoparticle mixtures in shrinking spherical confinement

Particles are strongly attracted to the liquid-liquid interface of the emulsion droplet by a harmonic potential

$$U_{wall,i} = \epsilon_i \left(r_{iw} - r_{shift}\right)^2 + U_{cut}$$





B.P. Binks, *Curr. Op. Coll. Int. Sci.* **2002**, *7*, 21-41
Gartner, III, T.E., Heil, C.M., Jayaraman, A. **2019**, *in preparation*

But LAMMPS only has a repulsive harmonic wall! Modified fix_wall_region.cpp and fix_wall_region.h

Example: attractive particle-wall potential

Original code for fix_wall_region.cpp and fix_wall_region.h

fix_wall_region.cpp

```
FixWallRegion::FixWallRegion(LAMMPS *lmp, int narg, char **arg) :
  Fix(lmp, narg, arg),
  idregion(NULL)
{
  if (narg != 8) error->all(FLERR,"Illegal fix wall/region command");

  scalar_flag = 1;
  vector_flag = 1;
  size_vector = 3;
  global_freq = 1;
  extscalar = 1;
  extvector = 1;
  respa_level_support = 1;
  ilevel_respa = 0;
  virial_flag = 1;

  // parse args

  iregion = domain->find_region(arg[3]);
  if (iregion == -1)
    error->all(FLERR,"Region ID for fix wall/region does not exist");
  int n = strlen(arg[3]) + 1;
  idregion = new char[n];
  strcpy(idregion,arg[3]);

  if (strcmp(arg[4],"lj93") == 0) style = LJ93;
  else if (strcmp(arg[4],"lj126") == 0) style = LJ126;
  else if (strcmp(arg[4],"lj1043") == 0) style = LJ1043;
  else if (strcmp(arg[4],"colloid") == 0) style = COLLOID;
  else if (strcmp(arg[4],"harmonic") == 0) style = HARMONIC;
  else error->all(FLERR,"Illegal fix wall/region command");

  if (style != COLLOID) dynamic_group_allow = 1;

  epsilon = force->numeric(FLERR,arg[5]);
  sigma = force->numeric(FLERR,arg[6]);
  cutoff = force->numeric(FLERR,arg[7]);

  if (cutoff <= 0.0) error->all(FLERR,"Fix wall/region cutoff <= 0.0");

  eflag = 0;
  ewall[0] = ewall[1] = ewall[2] = ewall[3] = 0.0;
}
```

fix_wall_region.cpp

```
/* ----------------------------------------------------------------------
   harmonic interaction for particle with wall
   compute eng and fwall = magnitude of wall force
------------------------------------------------------------------------- */

void FixWallRegion::harmonic(double r)
{
  double dr = cutoff - r;
  fwall = 2.0*epsilon*dr;
  eng = epsilon*dr*dr;
}
```

$$F_{wall,i} = -\frac{dU_{wall,i}}{dr_{iw}}$$

fix_wall_region.h

```
 private:
  int style,iregion;
  double epsilon,sigma,cutoff;
  int eflag;
  double ewall[4],ewall_all[4];
  int ilevel_respa;
  char *idregion;

  double coeff1,coeff2,coeff3,coeff4,offset;
  double coeff5,coeff6,coeff7;
  double eng,fwall;

  void lj93(double);
  void lj126(double);
  void lj1043(double);
  void colloid(double, double);
  void harmonic(double);
};
}
```

Example: attractive particle-wall potential

Modified code for fix_wall_region.cpp

fix_wall_region.cpp

```
FixWallRegion::FixWallRegion(LAMMPS *lmp, int narg, char **arg) :
  Fix(lmp, narg, arg),
  idregion(NULL)
{
  if (narg != 8) error->all(FLERR,"Illegal fix wall/region command");

  scalar_flag = 1;
  vector_flag = 1;
  size_vector = 3;
  global_freq = 1;
  extscalar = 1;
  extvector = 1;
  respa_level_support = 1;
  ilevel_respa = 0;
  virial_flag = 1;

  // parse args

  iregion = domain->find_region(arg[3]);
  if (iregion == -1)
    error->all(FLERR,"Region ID for fix wall/region does not exist");
  int n = strlen(arg[3]) + 1;
  idregion = new char[n];
  strcpy(idregion,arg[3]);

  if (strcmp(arg[4],"lj93") == 0) style = LJ93;
  else if (strcmp(arg[4],"lj126") == 0) style = LJ126;
  else if (strcmp(arg[4],"lj1043") == 0) style = LJ1043;
  else if (strcmp(arg[4],"colloid") == 0) style = COLLOID;
  else if (strcmp(arg[4],"harmonic") == 0) style = HARMONIC;
  else if (strcmp(arg[4],"harmonicMURI") == 0) style = HARMONICMURI;
  else error->all(FLERR,"Illegal fix wall/region command");

  if (style != COLLOID) dynamic_group_allow = 1;

  epsilon = force->numeric(FLERR,arg[5]);
  sigma = force->numeric(FLERR,arg[6]);
  cutoff = force->numeric(FLERR,arg[7]);

  if (cutoff <= 0.0) error->all(FLERR,"Fix wall/region cutoff <= 0.0");

  eflag = 0;
  ewall[0] = ewall[1] = ewall[2] = ewall[3] = 0.0;
}
```

fix_wall_region.cpp

```
void FixWallRegion::init()
{
  // set index and check validity of region

  iregion = domain->find_region(idregion);
  if (iregion == -1)
    error->all(FLERR,"Region ID for fix wall/region does not exist");
```

⋮

```
  // setup coefficients for each style

  if (style == LJ93) {
    coeff1 = 6.0/5.0 * epsilon * pow(sigma,9.0);
    coeff2 = 3.0 * epsilon * pow(sigma,3.0);
    coeff3 = 2.0/15.0 * epsilon * pow(sigma,9.0);
    coeff4 = epsilon * pow(sigma,3.0);
    double rinv = 1.0/cutoff;
    double r2inv = rinv*rinv;
    double r4inv = r2inv*r2inv;
    offset = coeff3*r4inv*r4inv*rinv - coeff4*r2inv*rinv;
  } else if (style == LJ126) {
    coeff1 = 48.0 * epsilon * pow(sigma,12.0);
    coeff2 = 24.0 * epsilon * pow(sigma,6.0);
    coeff3 = 4.0 * epsilon * pow(sigma,12.0);
    coeff4 = 4.0 * epsilon * pow(sigma,6.0);
    double r2inv = 1.0/(cutoff*cutoff);
    double r6inv = r2inv*r2inv*r2inv;
    offset = r6inv*(coeff3*r6inv - coeff4);
```

⋮

```
  } else if (style == HARMONICMURI) {
    coeff1 = 1.0/((cutoff - sigma)*(cutoff - sigma));
    coeff2 = -2.0*sigma/((cutoff - sigma)*(cutoff - sigma));
    coeff3 = sigma*sigma/((cutoff - sigma)*(cutoff - sigma)) - 1.0;
  } else if (style == COLLOID) {
    coeff1 = -4.0/315.0 * epsilon * pow(sigma,6.0);
    coeff2 = -2.0/3.0 * epsilon;
    coeff3 = epsilon * pow(sigma,6.0)/7560.0;
    coeff4 = epsilon/6.0;
    double rinv = 1.0/cutoff;
    double r2inv = rinv*rinv;
    double r4inv = r2inv*r2inv;
    offset = coeff3*r4inv*r4inv*rinv - coeff4*r2inv*rinv;
  }
```

Example: attractive particle-wall potential

Modified code for fix_wall_region.cpp and fix_wall_region.h

fix_wall_region.cpp

```
/* ----------------------------------------------------------------------
   attractive harmonic interaction for particle with wall
   compute eng and fwall = magnitude of wall force
------------------------------------------------------------------------- */

void FixWallRegion::harmonicMURI(double r)
{
  fwall = -2.0*epsilon*coeff1*r-epsilon*coeff2;
  eng = epsilon*(coeff1*r*r+coeff2*r+coeff3);
}
```

$$U_{wall,i} = \epsilon_i \left(r_{iw} - r_{shift}\right)^2 + U_{cut}$$

$$F_{wall,i} = -\frac{dU_{wall,i}}{dr_{iw}}$$

Make sure to always validate your simulation output to verify you get the behavior you expect!

fix_wall_region.h

```
#ifdef FIX_CLASS

FixStyle(wall/region,FixWallRegion)

#else

#ifndef LMP_FIX_WALL_REGION_H
#define LMP_FIX_WALL_REGION_H

#include "fix.h"

namespace LAMMPS_NS {

class FixWallRegion : public Fix {
 public:
  FixWallRegion(class LAMMPS *, int, char **);
  ~FixWallRegion();
  int setmask();
  void init();
  void setup(int);
  void min_setup(int);
  void post_force(int);
  void post_force_respa(int, int, int);
  void min_post_force(int);
  double compute_scalar();
  double compute_vector(int);

 private:
  int style,iregion;
  double epsilon,sigma,cutoff;
  int eflag;
  double ewall[4],ewall_all[4];
  int ilevel_respa;
  char *idregion;

  double coeff1,coeff2,coeff3,coeff4,offset;
  double coeff5,coeff6,coeff7;
  double eng,fwall;

  void lj93(double);
  void lj126(double);
  void lj1043(double);
  void colloid(double, double);
  void harmonic(double);
  void harmonicMURI(double);
};
```

Questions and resources

- LAMMPS documentation:
  - https://lammps.sandia.gov/doc/
- LAMMPS developer's guide:
  - https://lammps.sandia.gov/doc/Developer.pdf
- Various presentations available:
  - E.g., https://lammps.sandia.gov/tutorials/italy14/italy_modify_Mar14.pdf