LAMMPS SHORT MANUAL

MODMOL 25-27 Feb 2008, Jouy-en-Josas

olivier.vitrac@agroparistech.fr

LAMMPS

LAMMPS is a molecular dynamics program from Sandia National Laboratories. LAMMPS makes use of MPI for parallel communication and is a free open-source code, distributed under the terms of the GNU General Public License.

LAMMPS was originally developed under a Cooperative Research and Development Agreement (CRADA) between two laboratories from United States Department of Energy and three other laboratories from private sector firms. It is currently maintained and distributed by researchers at the Sandia National Laboratories.

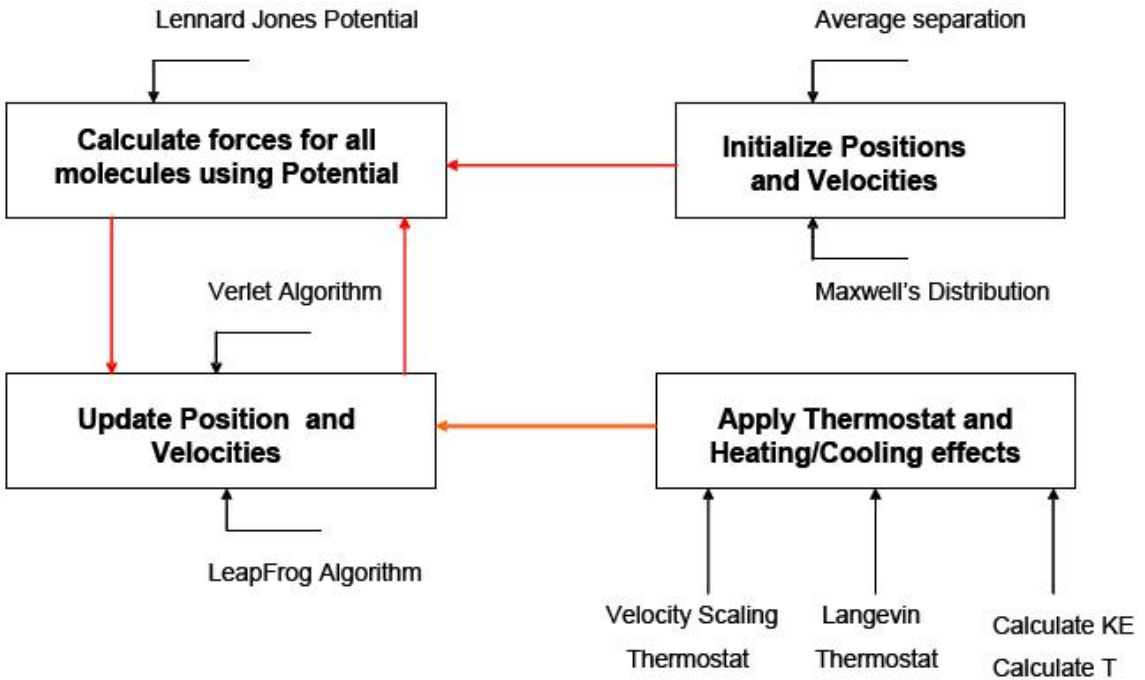
Features

For computational efficiency LAMMPS uses neighbor lists to keep track of nearby particles. The lists are optimized for systems with particles that are repulsive at short distances, so that the local density of particles never becomes too large.
On parallel computers, LAMMPS uses spatial-decomposition techniques to partition the simulation domain into small 3d sub-domains, one of which is assigned to each processor. Processors communicate and store "ghost" atom information for atoms that border their sub-domain. LAMMPS is most efficient (in a parallel computing sense) for systems whose particles fill a 3D rectangular box with approximately uniform density.

http://lammps.sandia.gov/

http://lammps.sandia.gov/doc/Manual.html

PRINCIPLES	
1) Initialization	units, dimension, boundary, atom_style, atom_modify.
2) Atom definition	read_data,read_restart, lattice, region, create_box, create_atoms
3) Settings	pair_coeff, bond_coeff, angle_coeff, dihedral_coeff, improper_coeff, kspace_style, dielectric, special_bonds
	neighbor, neigh_modify, group, timestep, reset_timestep, run_style, min_style, min_modify.
	compute, compute_modify, variable
4) Run	run, minimize



SCRIPT

1) Initialization

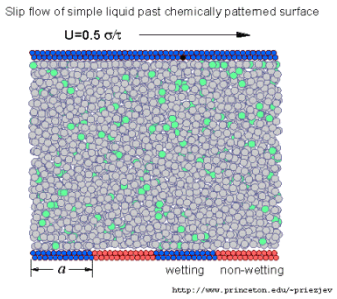
2) Atom definition

3) Settings

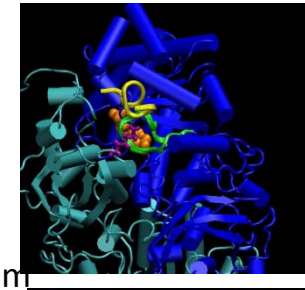
4) Run

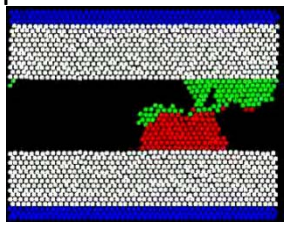
```
# 3d Lennard-Jones melt

units           lj
atom_style      atomic

lattice         fcc 0.8442
region          box block 0 20 0 20 0 20
create_box      1 box
create_atoms    1
mass            1 1.0

velocity    all create 3.0 87287

pair_style      lj/cut 2.5
pair_coeff      1 1 1.0 1.0 2.5

neighbor 0.3 bin
neigh_modify    every 20 delay 0 check no

fix             1 all nve

dump            id all atom 10 dump.melt

thermo          50

run             250
```

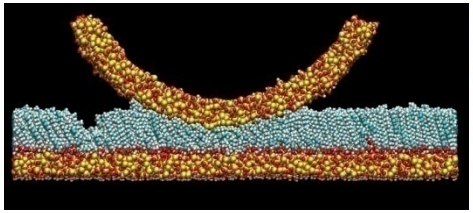
atom_style *style args*

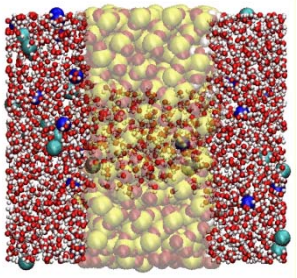


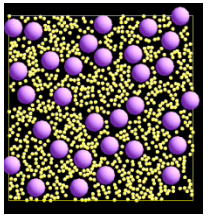
angle = bonds and angles - e.g. bead-spring polymers with stiffness
atomic = only the default values
bond = bonds - e.g. bead-spring polymers
charge = charge
dipole = charge and dipole moment
dpd = default values, also communicates velocities
ellipsoid = quaternion for particle orientation, angular velocity/momentum
full = molecular + charge - e.g. biomolecules, charged polymers
granular = granular atoms with rotational properties
molecular = bonds, angles, dihedrals, impropers - e.g. all-atom polymers

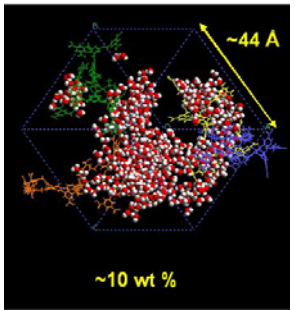


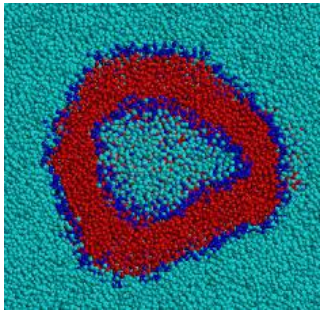


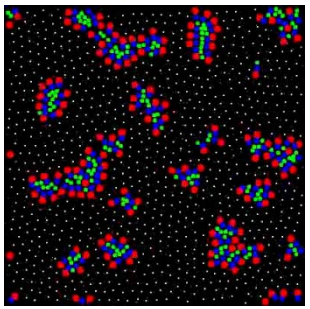


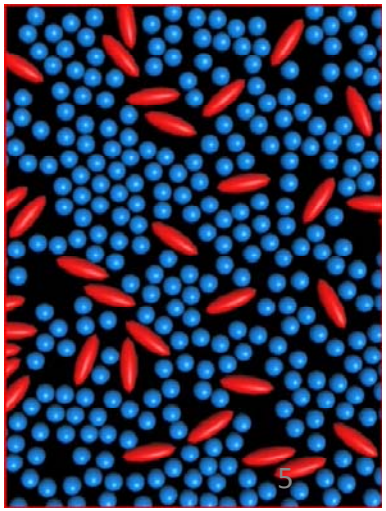








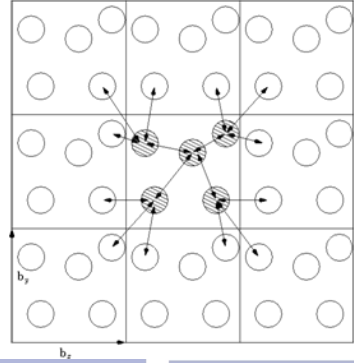


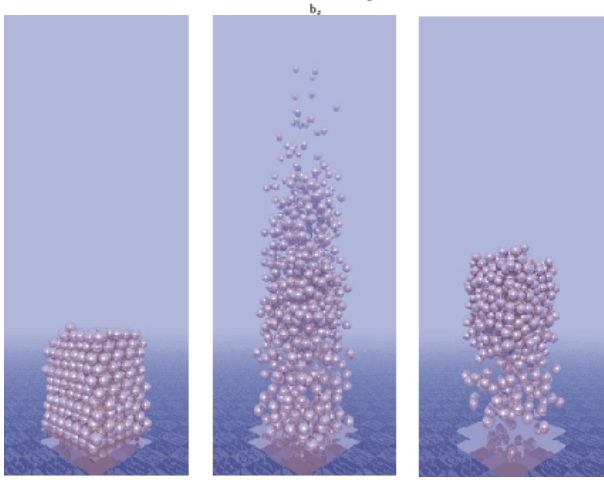


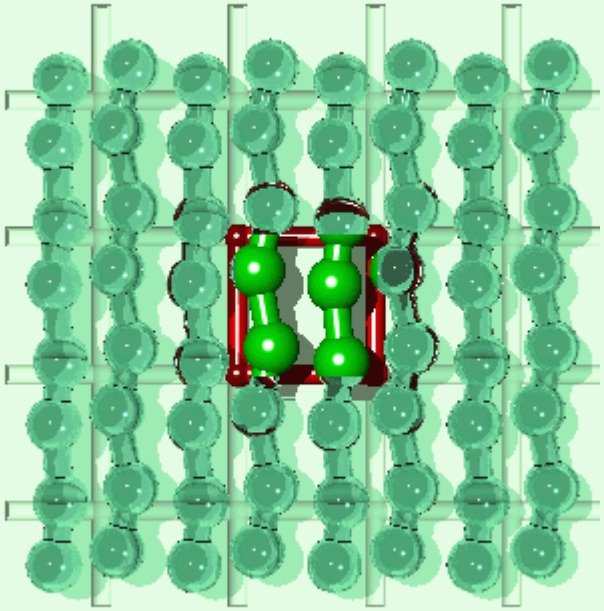
boundary *x y z*

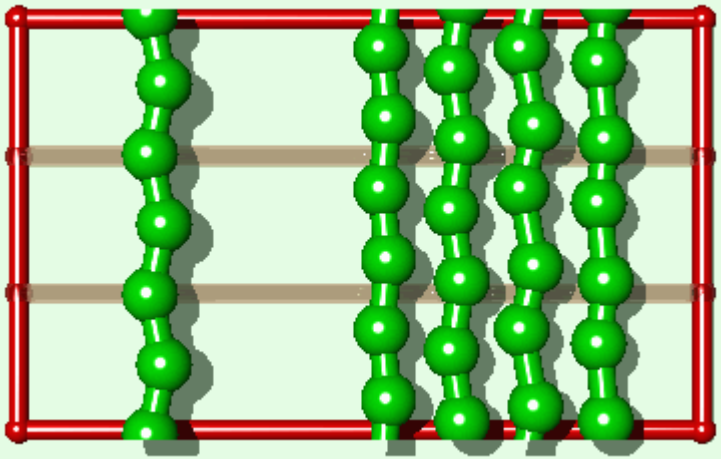
x,y,z = *p* or *s* or *f* or *m*, one or two letters

p is periodic
f is non-periodic and fixed
s is non-periodic and shrink-wrapped
m is non-periodic and shrink-wrapped with a minimum value









INIT
atom_modify
atom_style
boundary
dimension
newton
processors
UNITS

units lj	style real
distance = sigma time = tau mass = one energy = epsilon velocity = sigma/tau force = epsilon/sigma temperature = reduced LJ temperature pressure = reduced LJ pressure charge = reduced LJ charge dipole = reduced LJ dipole moment electric field = force/charge	distance = Angstroms time = femtoseconds mass = grams/mole energy = Kcal/mole velocity = Angstroms/femtosecond force = Kcal/mole-Angstrom temperature = degrees K pressure = atmospheres charge = multiple of electron charge (+1.0 is a proton) dipole = charge*Angstroms electric field = volts/Angstrom

read_data *file*

atoms = # of atoms in system
bonds = # of bonds in system
angles = # of angles in system
dihedrals = # of dihedrals in system
impropers = # of impropers in system
atom types = # of atom types in system
bond types = # of bond types in system
angle types = # of angle types in system
dihedral types = # of dihedral types in system
improper types = # of improper types in system
xlo xhi = simulation box boundaries in x dimension
ylo yhi = simulation box boundaries in y dimension
zlo zhi = simulation box boundaries in z dimension
xy xz yz = simulation box tilt factors for triclinic domain

```
LAMMPS Description            (1st line of file)

100 atoms                  (this must be the 3rd line, 1st 2 lines are ignored)
95 bonds                   (# of bonds to be simulated)
50 angles                  (include these lines even if number = 0)
30 dihedrals
20 impropers

5 atom types               (# of nonbond atom types)
10 bond types              (# of bond types = sets of bond coefficients)
18 angle types
20 dihedral types          (do not include a bond, angle,dihedral,improper type
2 improper types                 line if number of bonds,angles,etc is 0)

-0.5 0.5 xlo xhi           (for periodic systems this is box size,
-0.5 0.5 ylo yhi           for non-periodic it is min/max extent of atoms)
-0.5 0.5 zlo zhi           (do not include this line for 2-d simulations)

Masses

  1 mass
  ...
  N mass                             (N = # of atom types)

Pair Coeffs                           Nonbond Coeffs (in old versions)

  1 coeff1 coeff2 ...
  ...
  N coeff1 coeff2 ...                (N = # of atom types)

Bond Coeffs

  1 coeff1 coeff2 ...
  ...
  N coeff1 coeff2 ...                (N = # of bond types)

Angle Coeffs

  1 coeff1 coeff2 ...
  ...
  N coeff1 coeff2 ...                (N = # of angle types)
```

Dihedral Coeffs

```
  1 coeff1 coeff2 ...
  ...
  N coeff1 coeff2 ...              (N = # of dihedral types)
```

Improper Coeffs

```
  1 coeff1 coeff2 ...
  ...
  N coeff1 coeff2 ...              (N = # of improper types)
```

BondBond Coeffs

```
  1 coeff1 coeff2 ...
  ...
  N coeff1 coeff2 ...              (N = # of angle types)
```

BondAngle Coeffs

```
  1 coeff1 coeff2 ...
  ...
  N coeff1 coeff2 ...              (N = # of angle types)
```

MiddleBondTorsion Coeffs

```
  1 coeff1 coeff2 ...
  ...
  N coeff1 coeff2 ...              (N = # of dihedral types)
```

EndBondTorsion Coeffs

```
  1 coeff1 coeff2 ...
  ...
  N coeff1 coeff2 ...              (N = # of dihedral types)
```

AngleTorsion Coeffs

```
  1 coeff1 coeff2 ...
  ...
  N coeff1 coeff2 ...              (N = # of dihedral types)
```

AngleAngleTorsion Coeffs

```
1 coeff1 coeff2 ...
...
N coeff1 coeff2 ...              (N = # of dihedral types)
```

BondBond13 Coeffs

```
1 coeff1 coeff2 ...
...
N coeff1 coeff2 ...              (N = # of dihedral types)
```

AngleAngle Coeffs

```
1 coeff1 coeff2 ...
...
N coeff1 coeff2 ...              (N = # of improper types)
```

Atoms

```
1 molecule-tag atom-type q x y z nx ny nz  (nx,ny,nz are optional -
...                                        see "true flag" input command)
...
N molecule-tag atom-type q x y z nx ny nz  (N = # of atoms)
```

Velocities

```
1 vx vy vz
...
N vx vy vz                        (N = # of atoms)
```

Bonds

```
1 bond-type atom-1 atom-2
...
N bond-type atom-1 atom-2         (N = # of bonds)
```

Angles

```
1 angle-type atom-1 atom-2 atom-3  (atom-2 is the center atom in angle)
...
N angle-type atom-1 atom-2 atom-3  (N = # of angles)
```

```
Dihedrals

  1 dihedral-type atom-1 atom-2 atom-3 atom-4  (atoms 2-3 form central bond)
  ...
  N dihedral-type atom-1 atom-2 atom-3 atom-4  (N = # of dihedrals)

Impropers

  1 improper-type atom-1 atom-2 atom-3 atom-4  (atom-2 is central atom)
  ...
  N improper-type atom-1 atom-2 atom-3 atom-4  (N = # of impropers)
```

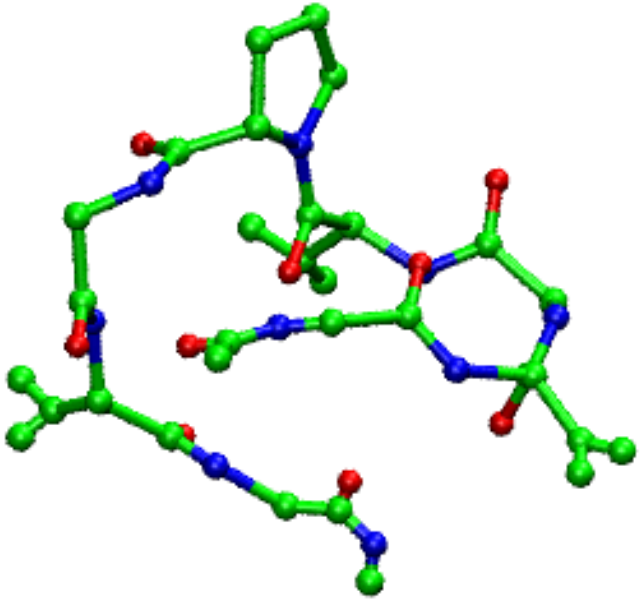
comments

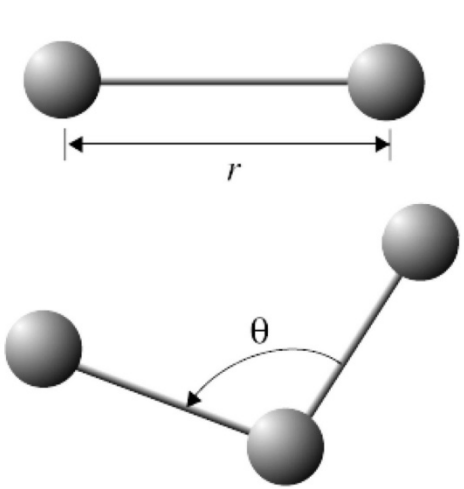
blank lines are ignored

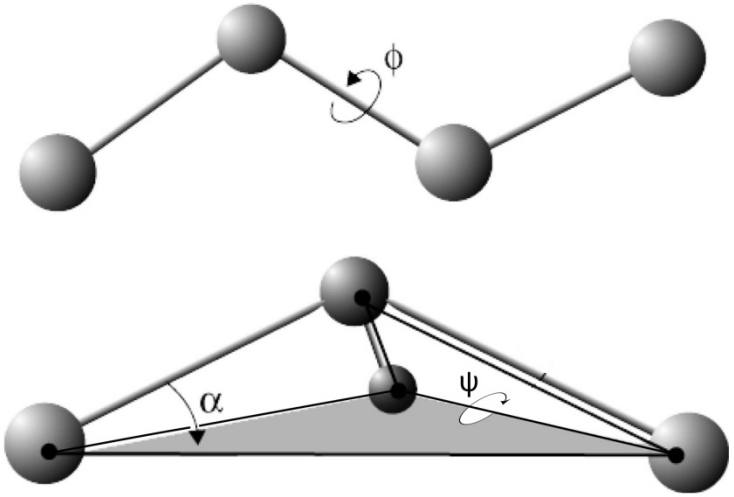
lines starting with a # are echoed into the log file

for commands, everything on a line after the last parameter is ignored

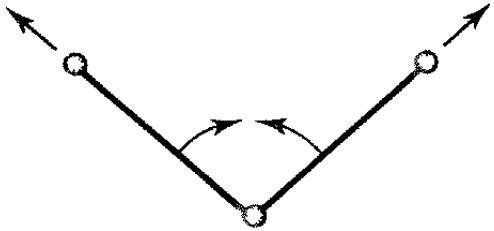
FORCEFIELD
angle_coeff
angle_style
bond_coeff
bond_style
dielectric
dihedral_coeff
dihedral_style
improper_coeff
improper_style
kspace_modify
kspace_style
pair_coeff
pair_modify
pair_style
pair_write
special_bonds







Cross terms: class 1, 2 and 3 force fields



The presence of cross terms in a force field reflects coupling between the internal coordinates. For example, as a bond angle is decreased it is found that the adjacent bonds stretch to reduce the interaction between the 1,3 atoms, as illustrated in Figure. Cross terms were found to be important in force fields designed to predict vibrational spectra that were the forerunners of molecular mechanics force fields, and so it is not surprising that cross terms must often be included in a molecular mechanics force field to achieve optimal performance.

One should in principle include cross terms between all contributions to a force field. However, only a few cross terms are generally found to be necessary in order to reproduce structural properties accurately; more may be needed to reproduce other properties such as vibrational frequencies, which are more sensitive to the presence of such terms,.

FORCEFIELD

- angle_coeff
- angle_style
- bond_coeff
- bond_style
- dielectric
- dihedral_coeff
- dihedral_style
- improper_coeff
- improper_style
- kspace_modify
- kspace_style
- pair_coeff
- pair_modify
- pair_style
- pair_write
- special_bonds

$$E_{pot} = \sum_b D_b[1-e^{-\alpha(b-b_0)}] + \sum_\theta H_\theta(\theta-\theta_0)^2 + \sum_\phi H_\phi[1+s\cos(n\phi)]$$

(1) (2) (3)

$$+\sum_\chi H_\chi\chi^2 + \sum_b\sum_{b'} F_{bb'}(b-b_0)(b'-b'_0) + \sum_\theta\sum_{\theta'} F_{\theta\theta'}(\theta-\theta_0)(\theta'-\theta'_0)$$

(4) (5) (6)

$$+\sum_b\sum_\theta F_{b\theta}(b-b_0)(\theta-\theta_0) + \sum_\phi F_{\phi\theta\theta'}\cos\phi(\theta-\theta_0)(\theta'-\theta'_0) + \sum_\chi\sum_{\chi'} F_{\chi\chi'}\chi\chi'$$

(7) (8) (9)

$$+\sum \varepsilon[(r^*/r)^{12} - 2(r^*/r)^6] + \sum q_i q_j/\varepsilon r_{ij}$$

(10) (11)

TYPE II: CFF (Consistent FF)

TYPE I: CVFF (Covalent FF)

$$E_{pot} = \sum_b [K_2(b-b_0)^2 + K_3(b-b_0)^3 + K_4(b-b_0)^4]$$

(1)

$$+\sum_\theta H_2(\theta-\theta_0)^2 + H_3(\theta-\theta_0)^3 + H_4(\theta-\theta_0)^4$$

(2)

$$+\sum_\phi \left[V_1\left[1-\cos\left(\phi-\phi_1^0\right)\right] + V_2\left[1-\cos\left(2\phi-\phi_2^0\right)\right] + V_3\left[1-\cos\left(3\phi-\phi_3^0\right)\right]\right]$$

(3)

$$+\sum_\chi K_\chi\chi^2 + \sum_b\sum_{b'} F_{bb'}(b-b_0)(b'-b'_0) + \sum_\theta\sum_{\theta'} F_{\theta\theta'}(\theta-\theta_0)(\theta'-\theta'_0)$$

(4) (5) (6)

$$+\sum_b\sum_\theta F_{b\theta}(b-b_0)(\theta-\theta_0) + \sum_b\sum_\phi (b-b_0)[V_1\cos\phi + V_2\cos 2\phi + V_3\cos 3\phi]$$

(7) (8)

$$+\sum_{b'}\sum_\phi (b'-b'_0)[V_1\cos\phi + V_2\cos 2\phi + V_3\cos 3\phi]$$

(9)

$$+\sum_\theta\sum_\phi (\theta-\theta_0)[V_1\cos\phi + V_2\cos 2\phi + V_3\cos 3\phi]$$

(10)

$$+\sum_\phi\sum_\theta\sum_{\theta'} K_{\phi\theta\theta'}\cos\phi(\theta-\theta_0)(\theta'-\theta'_0) + \sum_{i>j}\frac{q_i q_j}{\varepsilon r_{ij}} + \sum_{i>j}\left[\frac{A_{ij}}{r_{ij}^9} - \frac{B_{ij}}{r_{ij}^6}\right]$$

(11) (12) (13)

FORCEFIELD

angle_coeff

ANGLE_STYLE

bond_coeff

bond_style

dielectric

dihedral_coeff

dihedral_style

improper_coeff

improper_style

kspace_modify

kspace_style

pair_coeff

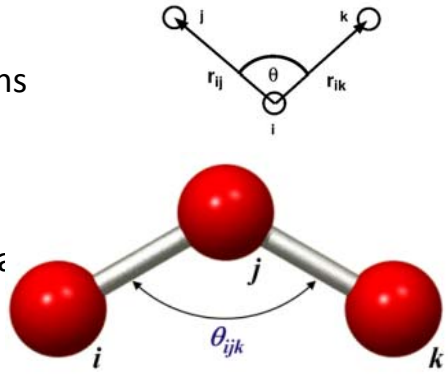
pair_modify

pair_style

pair_write

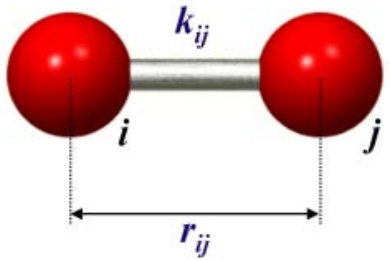
special_bonds

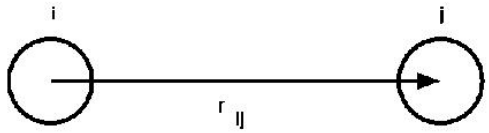
angle_style none - turn off angle interactions
angle_style hybrid - define multiple styles of angle interactions
angle_style charmm - CHARMM angle
angle_style class2 - COMPASS (class 2) angle
angle_style cosine - cosine angle potential
angle_style cosine/delta - difference of cosines angle potentia
angle_style cosine/squared - cosine squared angle potential
angle_style harmonic - harmonic angle



style	Expression	Examples
harmonic	$E = K(\theta - \theta_0)^2$ K (energy/radian^2) theta0 (degrees)	angle_style harmonic angle_coeff 1 300.0 107.0
Charmm	$E = K(\theta - \theta_0)^2 + K_{UB}(r - r_{UB})^2$ K (energy/radian^2) theta0 (degrees) K_ub (energy/distance^2) r_ub (distance)	angle_style charmm angle_coeff 1 300.0 107.0 50.0 3.0
class2	$E = E_a + E_{bb} + E_{ba}$ $E_a = K_2(\theta - \theta_0)^2 + K_3(\theta - \theta_0)^3 + K_4(\theta - \theta_0)^4$ $E_{bb=} M(r_{ij} - r_1)(r_{jk} - r_2)$ $E_{ba} = N_1(r_{ij} - r_1)(\theta - \theta_0) + N_2(r_{jk} - r_2)(\theta - \theta_0)$	angle_style class2 angle_coeff * 75.0

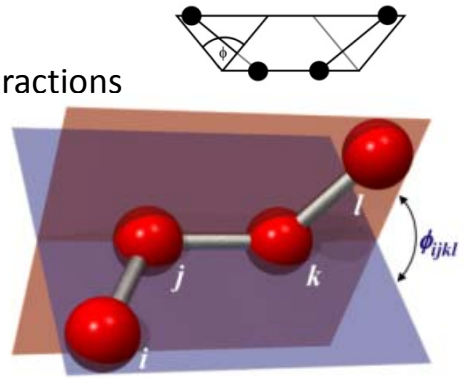
bond_style none - turn off bonded interactions
bond_style hybrid - define multiple styles of bond interactions
bond_style class2 - COMPASS (class 2) bond
bond_style fene - FENE (finite-extensible non-linear elastic) bond
bond_style fene/expand - FENE bonds with variable size particles
bond_style harmonic - harmonic bond
bond_style morse - Morse bond
bond_style nonlinear - nonlinear bond
bond_style quartic - breakable quartic bond





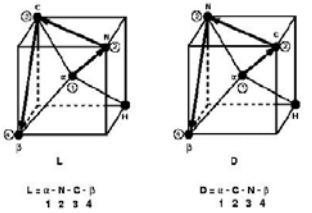
style	Expression	Examples
harmonic	$E = K(r - r_0)^2$ K (energy/distance^2) r0 (distance)	bond_style harmonic bond_coeff 5 80.0 1.2
class2	$K = K_2(r - r_0)^2 + K_3(r - r_0)^3 + K_4(r - r_0)^4$ r0 (distance) K2 (energy/distance^2) K3 (energy/distance^2) K4 (energy/distance^2)	bond_style class2 bond_coeff 1 1.0 100.0 80.0 80.0

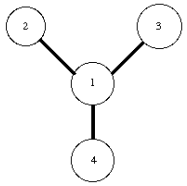
dihedral_style none - turn off dihedral interactions
dihedral_style hybrid - define multiple styles of dihedral interactions
dihedral_style charmm - CHARMM dihedral
dihedral_style class2 - COMPASS (class 2) dihedral
dihedral_style harmonic - harmonic dihedral
dihedral_style helix - helix dihedral
dihedral_style multi/harmonic - multi-harmonic dihedral
dihedral_style opls - OPLS dihedral

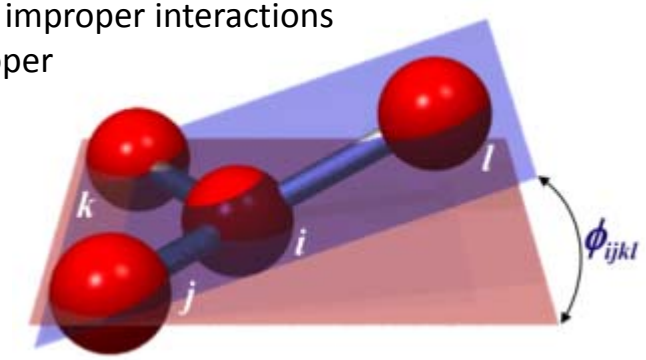


style	Expression	Examples
harmonic	$E = K \cdot \left[1 + d \cdot \cos\left(n \cdot \phi\right)\right]$ K (energy) d (+1 or -1) n (integer >= 0)	dihedral_style harmonic dihedral_coeff 1 80.0 1 2
Charmm	$E = K \cdot \left[1 + \cos\left(n \cdot \phi - d\right)\right]$ K (energy) n (integer >= 0) d (integer value of degrees) weighting factor (0.0 to 1.0)	dihedral_style charmm dihedral_coeff 1 120.0 1 60 0.5
class2	$E = E_d + E_{mbt} + E_{ebt} + E_{at} + E_{aat} + E_{bb13}$ $E_d = \sum_{n=1}^{3} K_n[1 - \cos(n\phi - \phi_n)]$ $E_{mbt} = (r_{jk} - r_2)[A_1 \cos(\phi) + A_2 \cos(2\phi) + A_3 \cos(3\phi)]$ $E_{ebt} = (r_{ij} - r_1)[B_1 \cos(\phi) + B_2 \cos(2\phi) + B_3 \cos(3\phi)] + (r_{kl} - r_3)[C_1 \cos(\phi) + C_2 \cos(2\phi) + C_3 \cos(3\phi)]$ $E_{at} = (\theta_{ijk} - \theta_1)[D_1 \cos(\phi) + D_2 \cos(2\phi) + D_3 \cos(3\phi)] + (\theta_{jkl} - \theta_2)[E_1 \cos(\phi) + E_2 \cos(2\phi) + E_3 \cos(3\phi)]$ $E_{aat} = M(\theta_{ijk} - \theta_1)(\theta_{jkl} - \theta_2)\cos(\phi)$ $E_{bb13} = N(r_{ij} - r_1)(r_{kl} - r_3)$	dihedral_style class2 dihedral_coeff 1 100 75 100 70 80 60

improper_style none - turn off improper interactions
improper_style hybrid - define multiple styles of improper interactions
improper_style class2 - COMPASS (class 2) improper
improper_style cvff - CVFF improper
improper_style harmonic - harmonic improper



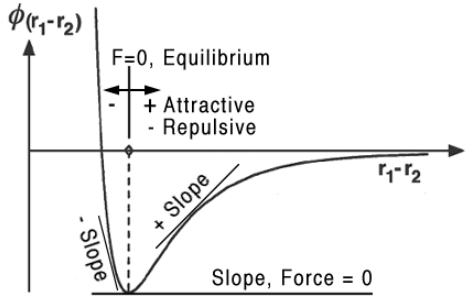




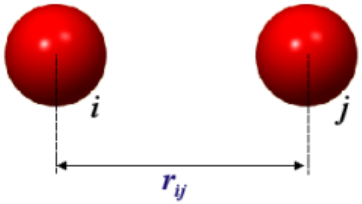
style	Expression	Examples
harmonic	$E = K(\chi - \chi_0)^2$ K (energy/radian^2) X0 (degrees)	improper_style harmonic improper_coeff 1 100.0 0
CVFF	$E = K \cdot [1 + d \cdot \cos(n \cdot \phi)]$ K (energy) d (+1 or -1) n (0,1,2,3,4,6)	improper_style cvff improper_coeff 1 80.0 -1 4
class2	$E = E_i + E_{aa}$ $E_i = K[\frac{\chi_{ijkl} + \chi_{kjli} + \chi_{ljik}}{3} - \chi_0]^2$ $E_{aa} = M_1(\theta_{ijk} - \theta_1)(\theta_{kjl} - \theta_3) + M_2(\theta_{ijk} - \theta_1)(\theta_{ijl} - \theta_2) + M_3(\theta_{ijl} - \theta_2)(\theta_{kjl} - \theta_3)$	improper_style class2 improper_coeff 1 100.0 0

- FORCEFIELD
- angle_coeff
- angle_style
- bond_coeff
- bond_style
- dielectric
- dihedral_coeff
- dihedral_style
- improper_coeff
- improper_style
- kspace_modify
- kspace_style
- pair_coeff
- pair_modify
- PAIR_STYLE
- pair_write
- special_bonds

pair_style lj/charmm/coul/charmm - CHARMM potential with cutoff Coulomb
pair_style lj/charmm/coul/charmm/implicit - CHARMM for implicit solvent
pair_style lj/charmm/coul/long - CHARMM with long-range Coulomb
pair_style lj/charmm/coul/long/opt - optimized version of CHARMM with long-range Coulomb
pair_style lj/class2 - COMPASS (class 2) force field with no Coulomb
pair_style lj/class2/coul/cut - COMPASS with cutoff Coulomb
pair_style lj/class2/coul/long - COMPASS with long-range Coulomb
pair_style lj/cut - cutoff Lennard-Jones potential with no Coulomb
pair_style lj/cut/opt - optimized version of cutoff LJ
pair_style lj/cut/coul/cut - LJ with cutoff Coulomb
pair_style lj/cut/coul/debye - LJ with Debye screening added to Coulomb
pair_style lj/cut/coul/long - LJ with long-range Coulomb
pair_style lj/cut/coul/long/tip4p - LJ with long-range Coulomb for TIP4P water
pair_style lj/expand - Lennard-Jones for variable size particles
pair_style lj/smooth - smoothed Lennard-Jones potential



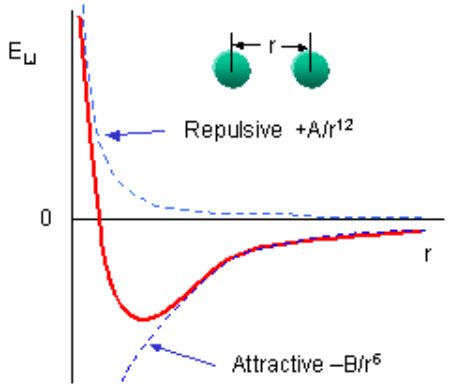
Van-der-Waals with
Lorentz—Berthelot mixing rule



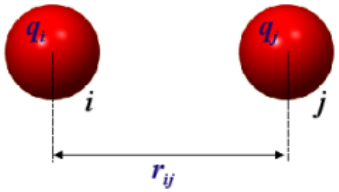
$$E = 4\varepsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right]$$

$$\text{av.diameter} = \sigma_{ij} = \frac{\sigma_{ii} + \sigma_{ij}}{2}$$

$$\text{av.well depth} = \varepsilon_{ij} = \sqrt{\varepsilon_{ii} + \varepsilon_{ij}}$$



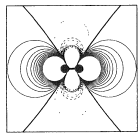
Coulomb

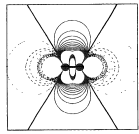


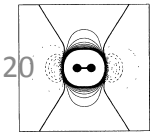
$$E = \frac{q_i q_j}{4\pi\varepsilon_0\varepsilon_R r_{ij}}$$

$$\varepsilon_0 = 8.8542 \cdot 10^{-12}\ \mathrm{C^2 N^{-1} m^{-2}}$$

$$\varepsilon_R = 1 \text{ for a vacuum by definition}$$







FORCEFIELD

- angle_coeff
- angle_style
- bond_coeff
- bond_style
- dielectric
- dihedral_coeff
- dihedral_style
- improper_coeff
- improper_style
- kspace_modify
- kspace_style
- pair_coeff
- pair_modify
- PAIR_STYLE
- pair_write
- special_bonds

pair_style hybrid - multiple styles of pairwise interactions
pair_style hybrid/overlay - multiple styles of superposed pairwise interactions
pair_style airebo - AI-REBO potential
pair_style buck - Buckingham potential
pair_style buck/coul/cut - Buckingham with cutoff Coulomb
pair_style buck/coul/long - Buckingham with long-range Coulomb
pair_style colloid - integrated colloidal potential
pair_style coul/cut - cutoff Coulombic potential
pair_style coul/debye - cutoff Coulombic potential with Debye screening
pair_style coul/long - long-range Coulombic potential
pair_style dipole/cut - point dipoles with cutoff
pair_style dpd - dissipative particle dynamics (DPD)
pair_style eam - embedded atom method (EAM)
pair_style eam/opt - optimized version of EAM
pair_style eam/alloy - alloy EAM
pair_style eam/alloy/opt - optimized version of alloy EAM
pair_style eam/fs - Finnis-Sinclair EAM
pair_style eam/fs/opt - optimized version of Finnis-Sinclair EAM
pair_style gayberne - Gay-Berne ellipsoidal potential
pair_style gran/hertzian - granular potential with Hertizain interactions
pair_style gran/history - granular potential with history effects
pair_style gran/no_history - granular potential without history effects
pair_style lubricate - hydrodynamic lubrication forces
pair_style meam - modified embedded atom method (MEAM)
pair_style morse - Morse potential
pair_style morse/opt - optimized version of Morse potential
pair_style resquared - Everaers RE-Squared ellipsoidal potential
pair_style soft - Soft (cosine) potential
pair_style sw - Stillinger-Weber 3-body potential
pair_style table - tabulated pair potential
pair_style tersoff - Tersoff 3-body potential
pair_style yukawa - Yukawa potential

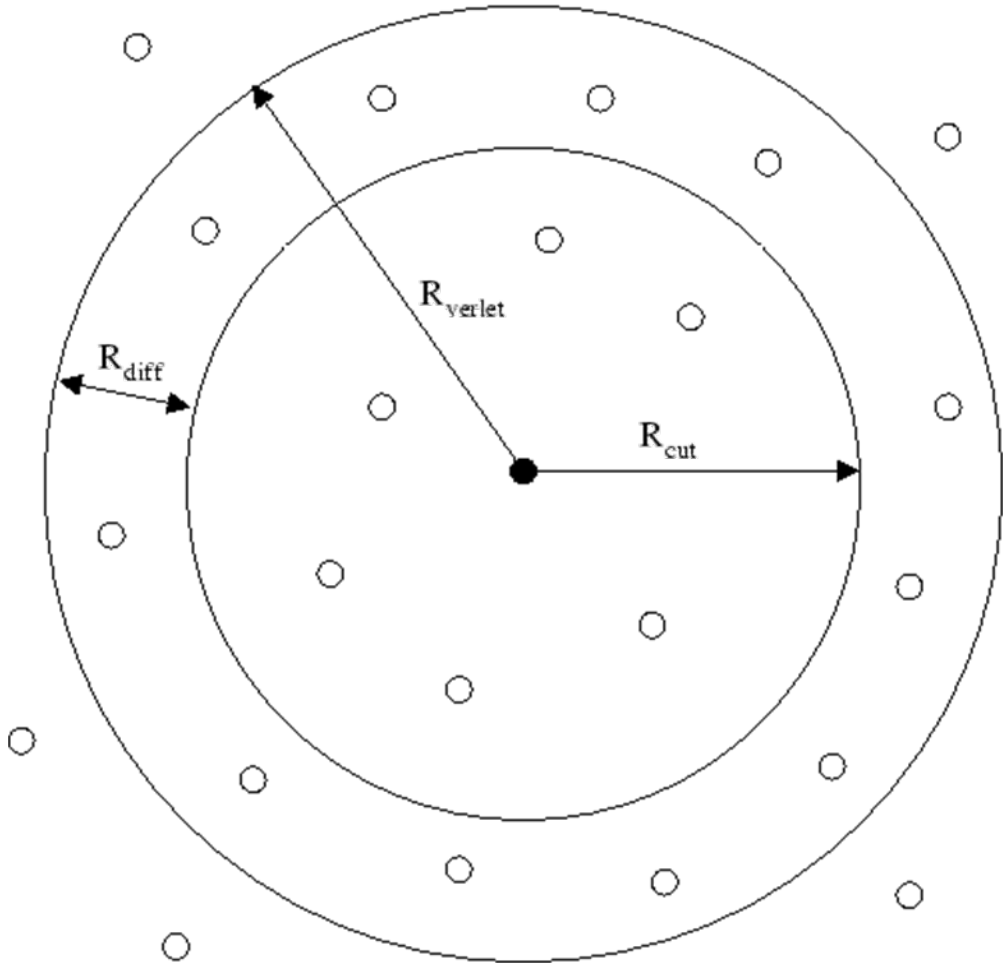
style	Expression	Examples
lj/cut	$E = E_{vdw} = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] \; r < r_C$	pair_style lj/cut 2.5 pair_coeff 1 1 1 1.1 2.8
lj/cut/coul/cut	$E = E_{vdw} + C\frac{q_i q_j}{\varepsilon r} \; r < r_C$ epsilon (energy units) sigma (distance units) cutoff (distance units) cutoff2 (distance units)	pair_style lj/cut/coul/cut 10.0 8.0 pair_coeff 1 1 100.0 3.5 9.0
lj/class2	$E = E_{vdw} = \varepsilon\left[2\left(\frac{\sigma}{r}\right)^{9} - 3\left(\frac{\sigma}{r}\right)^{6}\right]$ for $r < r_C$	pair_style lj/class2 10.0 pair_coeff 1 1 100.0 3.5 9.0
lj/class2/coul/cut	$E = E_{vdw} + C\frac{q_i q_j}{\varepsilon r}$ for $r < r_C$	pair_style lj/class2/coul/cut 10.0 8.0 pair_coeff 1 1 100.0 3.5 9.0

FORCEFIELD
angle_coeff
angle_style
bond_coeff
bond_style
dielectric
dihedral_coeff
dihedral_style
improper_coeff
improper_style
kspace_modify
kspace_style
pair_coeff
pair_modify
pair_style
pair_write
special_bonds

EXAMPLES

(when used, must appear after "read data" or "read restart" command)

```
angle coeff     1 30.0 108.0                (angle style harmonic)
angle coeff     1 30.0 108.0 30.0 2.5       (angle style charmm)
angle coeff     1 30.0 108.0                (angle style cosharmonic)
bond coeff      1 100.0 3.45                (bond style harmonic)
bond coeff      1 30.0 1.5 1.0 1.0          (bond style fene/standard )
bond coeff      1 30.0 1.5 1.0 1.0 0.2      (bond style fene/shift)
bond coeff      1 28.0 0.748308 0.166667    (bond style nonlinear)
dihedral coeff  1 10.0 1 3                  (dihedral style harmonic)
dihedral coeff  1 2.0 2.0 2.0 2.0 2.0       (dihedral style multiharmonic)
dihedral coeff  1 2.0 5 180.0 0.5           (dihedral style charmm)
dihedral coeff  1 2.0 1 3.0                 (dihedral style dreiding)
improper coeff  1 20.0 0.0                  (improper style harmonic)
improper coeff  1 20.0 10.0                 (improper style cvff)

dielectric      1

pair coeff      1 2 1.0 3.45 10.0           (pair style lj/cutoff)
pair coeff      1 2 1.0 3.45 8.0 10.0       (pair style lj/smooth)
pair coeff      1 2 1.0 3.45 2.0 10.0       (pair style lj/shift)
pair coeff      1 2 1.0 30.0 2.5            (pair style soft)
pair coeff      1 2 1.0 3.45 10.0           (pair style class2/cutoff)
pair coeff      1 2 1.0 3.45 1.0 3.45       (pair style lj/charmm)
pair coeff      1 2 1.0 3.45 12.0 10.0      (pair style expo_6/cutoff)
pair coeff      1 2 1.0 3.45 12.0 10.0 12.  0   (pair style expo_6/spline)
pair coeff      1 2 1.0 3.45 12.0 10.0      (pair style expo_6/smooth)

pppm mesh       32 32 64

pppm order      5

special bonds   amber

special bonds   0.0 0.0 0.5
```



neighbor *skin style*

neighbor 0.3 bin
neighbor 2.0 nsq

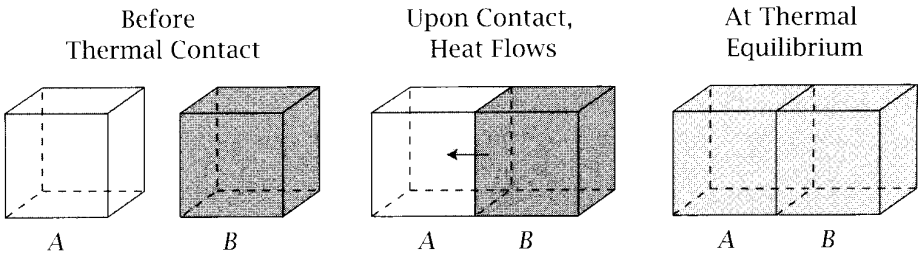
This command sets parameters that affect the building of the pairwise neighbor list. All atom pairs within a cutoff distance equal to the their force cutoff plus the *skin* distance are stored in the list. Typically, the larger the skin distance, the less often neighbor lists need to be built, but more pairs must be checked for possible force interactions every timestep.

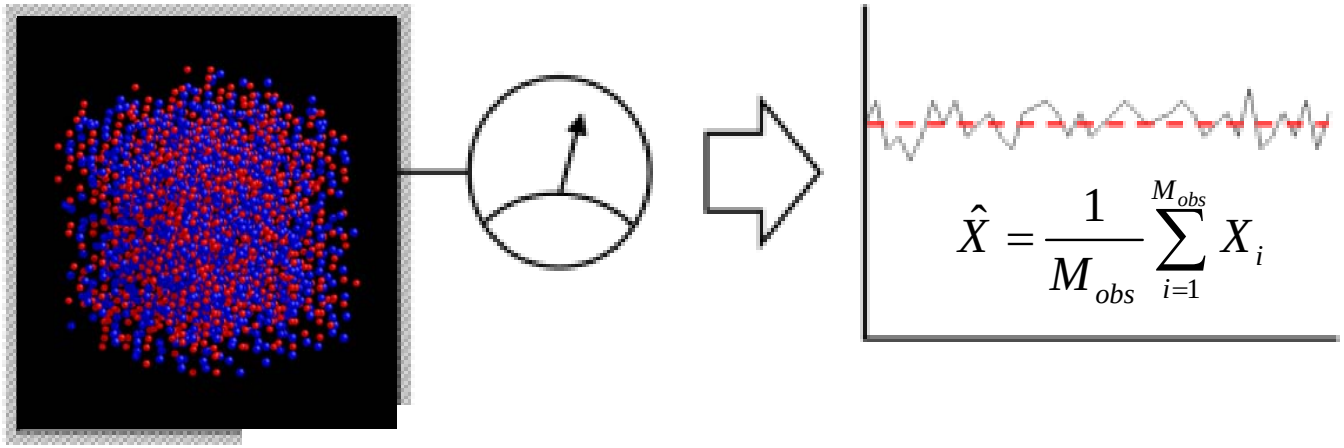
FIX

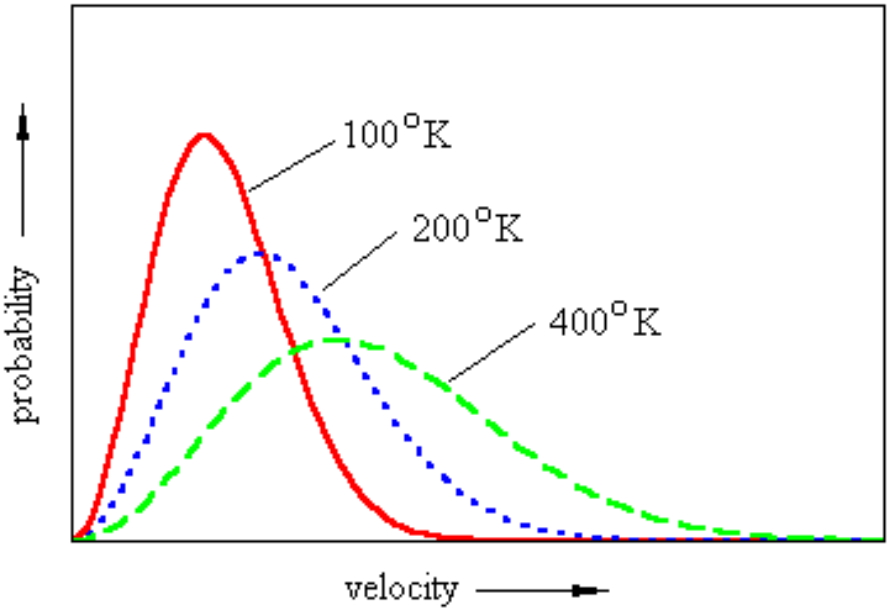
fix

fix_modify

unfix







fix *id all nve*

nve - constant NVE time integration
nve/asphere - NVT for aspherical particles
nve/dipole - NVE for point dipolar particles
nve/gran - NVE for granular particles
nve/limit - NVE with limited step length
nve/noforce - NVE without forces (v only)

fix *ID group-ID nve*

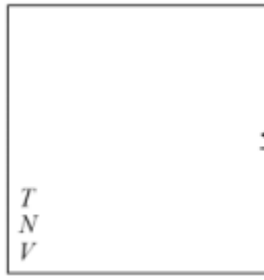
ID, group-ID are documented in fix command
nve = style name of this fix command

fix *id all nvt*

nvt - constant NVT time integration via Nose/Hoover
nvt/asphere - NVT for aspherical particles
nvt/sllod - NVT for NEMD with SLLOD equations

fix *ID group-ID nvt Tstart Tstop Tdamp keyword value ...*

ID, group-ID are documented in fix command
nvt = style name of this fix command
Tstart,Tstop = desired temperature at start/end of run
Tdamp = temperature damping parameter (time units)
zero or more keyword/value pairs may be appended
keyword = *drag*



$$\sum_{i=1}^{N} \frac{p_i^2}{2m_i} = \tfrac{3}{2} N k_B T$$

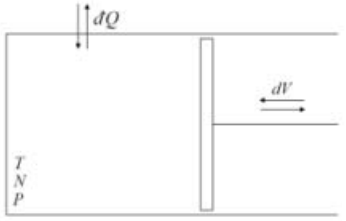
$$\lambda^2 = 1 + \lambda_c \left(\frac{T_0}{T} - 1 \right)$$

fix *id all npt*

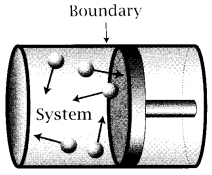
npt - constant NPT time integration via Nose/Hoover
npt/asphere - NPT for aspherical particles

fix *ID group-ID npt Tstart Tstop Tdamp p-style args keyword value ...*

xyz args = Pstart Pstop Pdamp
  Pstart,Pstop = desired pressure at start/end of run (pressure units)
  Pdamp = pressure damping parameter (time units)
xy or yz or xz or aniso args = Px_start Px_stop Py_start Py_stop Pz_start Pz_stop Pdamp
  Px_start,Px_stop,... = desired pressure in x,y,z at start/end of run (pressure units)
  Pdamp = pressure damping parameter (time units)



$$P = \rho k_B T + \frac{1}{3V} \sum_{i<j}^{N} \mathbf{F}_{ij} \cdot \mathbf{r}_{ij}$$



$$\mu^3 = 1 + \mu_c \left(\frac{P}{P_0} - 1 \right)$$

FIX

FIX

fix_modify

unfix

fix *ID group-ID style args*

addforce - add a force to each atom
aveforce - add an averaged force to each atom
ave/atom - compute per-atom time-averaged quantities
ave/spatial - output per-atom quantities by layer
ave/time - output time-averaged compute quantities
com - compute a center-of-mass
coord/original - store original coords of each atom
deform - change the simulation box size/shape
deposit - add new atoms above a surface
drag - drag atoms towards a defined coordinate
dt/reset - reset the timestep based on velocity, forces
efield - impose electric field on system
enforce2d - zero out z-dimension velocity and force
freeze - freeze atoms in a granular simulation
gravity - add gravity to atoms in a granular simulation
gyration - compute radius of gyration
heat - add/subtract momentum-conserving heat
indent - impose force due to an indenter
langevin - Langevin temperature control
lineforce - constrain atoms to move in a line
msd - compute mean-squared displacement (i.e. diffusion coefficient)
momentum - zero the linear and/or angular momentum of a group of atoms
nph - constant NPH time integration via Nose/Hoover
npt - constant NPT time integration via Nose/Hoover
npt/asphere - NPT for aspherical particles
nve - constant NVE time integration
nve/asphere - NVT for aspherical particles
nve/dipole - NVE for point dipolar particles
nve/gran - NVE for granular particles
nve/limit - NVE with limited step length
nve/noforce - NVE without forces (v only)
nvt - constant NVT time integration via Nose/Hoover
nvt/asphere - NVT for aspherical particles
nvt/sllod - NVT for NEMD with SLLOD equations
orient/fcc - add grain boundary migration force
planeforce - constrain atoms to move in a plane
poems - constrain clusters of atoms to move as coupled rigid bodies
pour - pour new atoms into a granular simulation domain
print - print text and variables during a simulation
rdf - compute radial distribution functions
recenter - constrain the center-of-mass position of a group of atoms
rigid - constrain one or more clusters of atoms to move as a rigid body
setforce - set the force on each atom
shake - SHAKE constraints on bonds and/or angles
spring - apply harmonic spring force to group of atoms
spring/rg - spring on radius of gyration of group of atoms
spring/self - spring from each atom to its origin
temp/rescale - temperature control by velocity rescaling
tmd - guide a group of atoms to a new configuration
viscosity - Muller-Plathe momentum exchange for viscosity calculation
viscous - viscous damping for granular simulations
wall/gran - frictional wall(s) for granular simulations
wall/lj126 - Lennard-Jones 12-6 wall
wall/lj93 - Lennard-Jones 9-3 wall
wall/reflect - reflecting wall(s)
wiggle - oscillate walls and frozen atoms

FIX
fix
fix_modify
unfix

EXAMPLES OF CONSTRAINTS

(when used, must appear after "read data" or "read restart" command)

assign fix	1 atom 200
assign fix	1 molecule 50
assign fix	1 type 2
assign fix	1 region 0.0 1.0 INF INF 0.0 1.0
assign fix	1 bondtype 4
assign fix	1 remainder

fix style	none
fix style	1 setforce 0.0 NULL 0.0
fix style	1 addforce 1.0 0.0 0.0
fix style	1 aveforce 1.0 0.0 0.0
fix style	1 rescale 300.0 300.0 100 20.0 0.5
fix style	1 hoover/drag 50.0 50.0 0.001
fix style	1 langevin 50.0 50.0 0.01 12345 1 1 1
fix style	1 springforce 10.0 NULL NULL 1.0
fix style	1 dragforce 10.0 -5.0 NULL 2.0 1.0
fix style	1 shake 3 0.001 100

COMPUTES

COMPUTE

compute_modify

uncompute

compute *ID group-ID style args*

compute 1 all temp
compute newtemp flow temp/partial 1 1 0
compute 3 all ke/atom

centro/atom - centro-symmetry parameter for each atom
coord/atom - coordination number for each atom
displace/atom - displacement of each atom
group/group - energy/force between two groups of atoms
ke/atom - kinetic energy for each atom
pe - potential energy
pe/atom - potential energy for each atom
pressure - total pressure and pressure tensor
reduce - combine per-atom quantities into a single global value
rotate/dipole - rotational energy of dipolar atoms
rotate/gran - rotational energy of granular atoms
stress/atom - stress tensor for each atom
temp - temperature of group of atoms
temp/asphere - temperature of aspherical particles
temp/deform - temperature excluding box deformation velocity
temp/dipole - temperature of point dipolar particles
temp/partial - temperature excluding one or more dimensions of velocity
temp/ramp - temperature excluding ramped velocity component
temp/region - temperature of a region of atoms

ACTIONS

delete_atoms

delete_bonds

displace_atoms

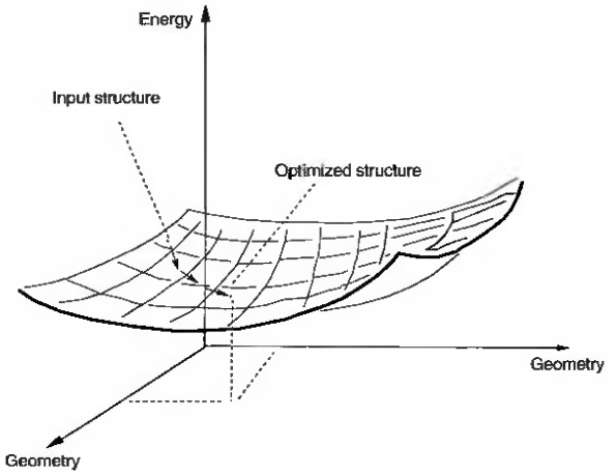
displace_box

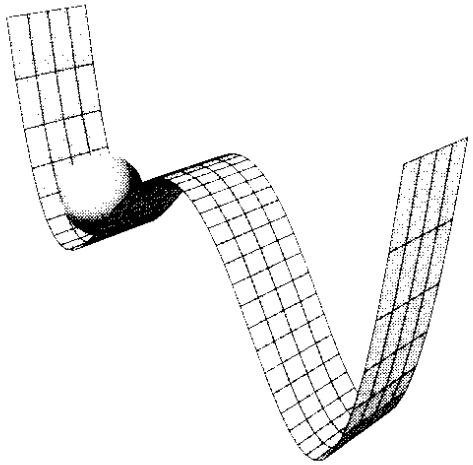
minimize

RUN

temper

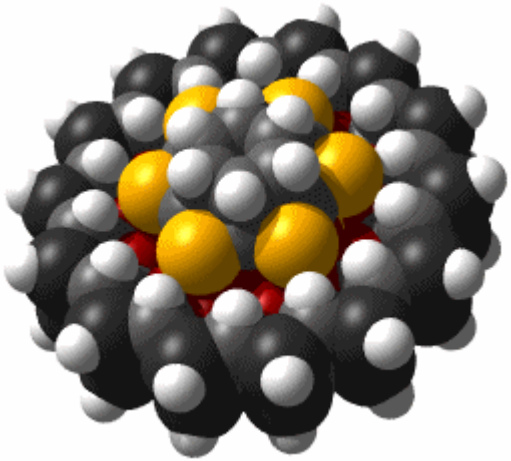
minimize *tolerance maxiter maxeval*





run *N keyword values*

N = # of timesteps
zero or more keyword/value pairs may be appended
keyword = *upto* or *start* or *stop* or *pre* or *post* or *every*



run 10000 run 1000000 upto
run 100 start 0 stop 1000
run 1000 pre no post yes
run 100000 start 0 stop 1000000 every 1000 print "Protein Rg = $r"
run 100000 every 1000 NULL

OUTPUT

- DUMP
- dump_modify
- restart
- thermo
- thermo_modify
- thermo_style
- undump
- write_restart

dump *ID group-ID style N file args*

ID = user-assigned name for the dump
group-ID = ID of the group of atoms to be dumped
style = *atom* or *bond* or *dcd* or *xtc* or *xyz* or *custom*
N = dump every this many timesteps
file = name of file to write dump info to
args = list of arguments for a particular style

dump myDump all atom 100
dump.atom dump 2 subgroup atom 50
dump.run.bin dump 4a all custom 100
dump 1 all xtc 1000 file.xtc

thermo_style *style args*

style = *one* or *multi* or *granular* or *custom*

args = list of arguments for a particular style

```
one args = none
  multi args = none
  granular args = none
  custom args = list of attributes
    possible attributes = step, atoms, cpu, temp, press,
                          pe, ke, etotal, enthalpy,
                          evdwl, ecoul, epair, ebond, eangle, edihed, eimp,
                          emol, elong, etail,
                          vol, lx, ly, lz, xlo, xhi, ylo, yhi, zlo, zhi,
                          pxx, pyy, pzz, pxy, pxz, pyz
                          drot, grot,
                          c_ID, c_ID[n], f_ID, f_ID[n], v_name
      step = timestep
      atoms = # of atoms
      cpu = elapsed CPU time
      temp = temperature
      press = pressure
      pe = total potential energy
      ke = kinetic energy
      etotal = total energy (pe + ke)
      enthalpy = enthalpy (pe + press*vol)
      evdwl = VanderWaal pairwise energy
      ecoul = Coulombic pairwise energy
      epair = pairwise energy (evdwl + ecoul + elong + etail)
      ebond = bond energy
      eangle = angle energy
      edihed = dihedral energy
      eimp = improper energy
      emol = molecular energy (ebond + eangle + edihed + eimp)
      elong = long-range kspace energy
      etail = VanderWaal energy long-range tail correction
      vol = volume
      lx,ly,lz = box lengths in x,y,z
      xlo,xhi,ylo,yhi,zlo,zhi = box boundaries
      pxx,pyy,pzz,pxy,pxz,pyz = 6 components of pressure tensor
      drot = rotational energy of dipolar atoms
      grot = rotational energy of granular atoms
      c_ID = global scalar value calculated by a compute with ID
      c_ID[N] = Nth component of global vector calculated by a compute with ID
      f_ID = global scalar value calculated by a fix with ID
      f_ID[N] = Nth component of global vector calculated by a fix with ID
      v_name = global value calculated by an equal-style variable with name
```

TOOLS	DESCRIPTION
amber2lammps	Python scripts for converting files back-and-forth between the AMBER MD code and LAMMPS
binary2txt	converts one or more binary LAMMPS dump file into ASCII text files
ch2lmp	contains tools for converting files back-and-forth between the CHARMM MD code and LAMMPS
chain	LAMMPS data file containing bead-spring polymer chains and/or monomer solvent atoms
data2xmovie	converts a LAMMPS data file into a snapshot suitable for visualizing with the xmovie tool
eam generate	converts an analytic formula into a tabulated embedded atom method (EAM) setfl potential file
lmp2arc	converting LAMMPS output files to the format for Accelrys's Insight MD code
lmp2cfg	tool for converting LAMMPS output files into a series of *.cfg files which can be read into the AtomEye visualizer
lmp2traj	tool for converting LAMMPS output files into 3 analysis files
matlab / MSLAB	several MATLAB scripts for post-processing LAMMPS output
micelle2d	creates a LAMMPS data file containing short lipid chains in a monomer solution
msi2lmp	tool for creating LAMMPS input data files from Accelrys's Insight MD code (formerly MSI/Biosysm and its Discover MD code)
pymol_asphere	tool for converting a LAMMPS dump file that contains orientation info for ellipsoidal particles into an input file for the PyMol visualization package
restart2data	converts a binary LAMMPS restart file into an ASCII data file
thermo_extract	reads one of more LAMMPS log files and extracts a thermodynamic value (e.g. Temp, Press)
vim , xmovie	X-based visualization package that can read LAMMPS dump files and animate them

MSLAB for LAMMPS on MIGALE

MODMOL 25-27 Feb 2008, Jouy-en-Josas

olivier.vitrac@agroparistech.fr

```
# MSLAB starts within $PROJECT
# example of line to be added/updated in ~/.bashrc
export PROJECT="~/project"
```

Custom LAMMPS installation

FILE: ~/.bashrc

```
Important definitions for LAMMPS in this ./.bashrc (use LAMRC to see the related HELP)

  Installation parameters
  =======================
      topaze: open a ssh connection on topaze
              see also: GENKEY, EXPORTKEY (to make it possible a SSH connection without password on cluster
              nodes)
    $PROJECT: main directory of LAMMPS projects (TO BE CHECKED with "echo $PROJECT")
              LAMMPS scripts must be located in $PROJECT (default) or in its subdirectories (e.g. examples)
              LAMMPS jobs are automatically stored in $PROJECT/XXXXX (where X=[A-Z0-9])
              see also: LSLAM, CDLAM, TREELAM
     Included directories
             bin/         LAMMPS executables (several versions are available)
                          executions with MPICH should be avoided, LAM/MPI must be preferred as it is
                          integrated with SUN GRID ENGINE
             bin/tools/   other executables
             doc/         documentation
             make/        make directory (for advanced users)
             examples/    examples
             pizza/       PYTHON interface to LAMMPS (see PIZZA)
                          installation file has been modified to match $PROJECT
                          NB: PIZZA works on LINUX and WINDOWS XP/Vista
     MSLAB (Matlab with the toolbox MS) requires:
            > a directory $PROJECT/../codes/MS where all MS functions are located
            > a script $PROJECT/startms.m where the GLOBAL variables LAMMPS and PATHPROJECT are defined
            > The variable LAMMPS must be updated to your needs (e.g. define your e-mail).
            NB: MSLAB works on LINUX and WINDOWS XP/Vista
```

CUSTOM LAMMPS JOB

FILE: ~/.bashrc

```
Typical job $PROJECT/XXXXX
==========================
   Each job directory contains initially:
       a LAMMPS executable: e.g. lmp_g++_lam_all_100208 (last version 10/02/08 for LAM/MPI, including all
       modules)
       a LAMMPS input file: e.g. long.in.lj
       additional data files: e.g. data.micelle (see LAMJOB to add user files)
       3 shell scripts:
            run.sh: submit the job on the cluster via SUN GRID ENGINE
                    manual queueue and execution on the cluster via: $PROJECT/XXXXX/run.sh (or ./run.sh)
            lammpsscript.sh: (e.g. long.in.sh) main launcher via MPIRUN (can be used direclty on TOPAZE)
                    manual execution on TOPAZE via: $PROJECT/XXXXX/lammpsscript.sh (or ./lammpsscript.sh)
                    prior a manual execution on TOPAZE, an active lamboot is required and subsequently a
                    lamhalt (lamboot and lamhalt lines in the scrip are inactivated by default in the script
                    since they are managed automatically by the SUN GRID ENGINE)
            mpi.sh: MPI argument (to used only by lammpsscript.sh)
   After/during execution, several files are created:
        jobid: jod id to be used with qstat -j jobid
       project.log: log file
       project.out: standard output (STDOUT)
       project.err: standard error (STDERR)
        lammps.log: additional log (default)
   see also: MSLAB, PIZZA, RMLAMJOB, CLLAMJOB, LAMAN, TREELAM, PSSEARCH
```

FILE: ~/.bashrc

BASH WRAPPERS	DESCRIPTION	USAGE
cdlam	cd into a LAMMPS dir/path	cdlam localpath
cllamjob or cllammpsjob	clean LAMMPS jobs via MSLAB	ccllamjob jobname1 [jobname2][jobname3] [jobname4] ...
exportkey	export RSA key on the cluster	exportkey firstnode [lastnode]
genkey	RSA genkey (require for SSH)	genkey
killsearch	kill a process on all nodes	killsearch processname [username] [firs
msi2lmp	convert a MSI project into a Lammps data file	msi2lmp msiproject [class] [ff] [print] [ffpath]
laman	help on LAMMPS script command	laman [lammps_command]
lamjob, lammpsjob	Prepare/queue/run a LAMMPS job via MSLAB on the cluster	lamjob mylammpsscript cmd [numproc] [jobname] [jobpath] [jobpath] [datafile1][datafile2]… cmd = run, script or runone
lamrc or lammpsrc	display a general help via .bashrc	lamrc
lslam	ls into a LAMMPS dir/path	lslam localpath
mslab	matlab with MS (text mode)	mslab, see "help MS" for detailed funct
pizza	PYTHON extension for LAMMPS (custom installation)	pizza
pssearch	search a process on all nodes	pssearch processname [username] [firstn
rmlamjob or rmlammpsjob	remove LAMMPS jobs via MSLAB	rmlamjob jobname1 [jobname2][jobname3] [jobname4] ...
treelam	tree all LAMMPS projects	treelam, treelam long, treelam short
u	list all jobs for the current user	u

PATH: $PROJECT/../codes/MS

MSLAB FUNCTIONS	DESCRIPTION
cllammpsjob	clean a or several LAMMPS jobs (remove the LAMMPS executable)
lamdumpread	read LAMMPS dump file (all fields are identified after the keyword 'ITEM:' and the following values are assumed to be numerical)
readlog	LAMMPS log files
lammpsjob	create and launch a LAMMPS job on the cluster (as defined in global variables: PROJECTPATH and LAMMPS)
lmp2cfg	LAMMPS dump file to Extended CFG Format (No velocity) to be used
msi2lmp	converts a MSIproject (MSIproject.car, MSIproject.mdf) into a LAMMPS data file
readdump_all	all timesteps from a LAMMPS dump file
readdump_one	LAMMPS dump file one timestep at a time
readlog	LAMMPS log files
readrdf	to read Radial Distribution Funtion output from LAMMPS
rmlammpsjob	remove a or several LAMMPS job(s) (delete the entire directories created by LAMMPSJOB)
scandump	to scan LAMMPS dump file

MS (Molecular Studio) Toolbox for Matlab include 254 functions

Main contributor: O. Vitrac

LINUX CAPAPLE, WIN32/64 COMPATIBLE

MSLAB custom installation

FILE: $PROJECT/startms.m

```
% STARTMS (PATH: $PROJECT/startms.m) – MATLAB SCRIPT
% MINIMUM setup configuration for MS on UNIX machines
% It works also on WINBOXES (Note: MS was designed on WIN32 machines)
%
% This script defines two global variables PROJECTPATH (string) and LAMMPS (structure)
% A new user must update PROJECTPATH and possibly some fields in LAMMPS
%
% PROJECTPATH: defines the main directory for all LAMMPS projects
% When the environment variable $PROJECT exist, its value is assigned to PROJECTPATH (default behavior)
% If not the script definition is used.
%
% LAMMPS: object used to generate all required BASH scripts to submit a job instance
%       (with MPIRUN or not, with SUN GRID ENGINE or not)
%       In future versions, this variable will be replaced by a single XML file.
%   Customizable fields in LAMMPS include:
%             LAMMPS.bin: bin/ directory in PROJECTPATH (all required binaries are assumed to be located there)
%                         NB: Tools are expected to be located in bin/tools (see MSI2LMP)
%                         Makefiles are located in PROJECTPATH/make
%          LAMMPS.lammps: filename of LAMMPS executable (several versions of LAMMPS can be used according to
                          MPI, compilation options...)
%      LAMMPS.maxnumproc: max number of processors, which can be invoked (theoretically 1000, 16 could be  an
                          acceptable value on MIGALE)
%           LAMMPS.email: contact e-mail (use to send e-mails during main job events)
%
%  The local variable localMS define the relative path from the location of startms, where MS is installed.
%  By default, MS is outside PROJECT (safe behavior). Update its content to your need.
```

custom PIZZA.py on MIGALE

MODMOL 25-27 Feb 2008, Jouy-en-Josas

olivier.vitrac@agroparistech.fr

```
# PIZZA and MSLAB starts within $PROJECT
# example of line to be added/updated in ~/.bashrc
export PROJECT="~/project"
```

PIZZA.py custom installation

FILE: $PROJECT/pizza/src/pizza.py

```
#!/usr/local/bin/python -I
# Pizza.py toolkit, www.cs.sandia.gov/~sjplimp/pizza.html
...
...
# modules needed by pizza.py
import sys, commands, os, string, exceptions, glob, refrom time import clock
# Customization by O. Vitrac
boxname = os.name
if boxname.find("nt") >=0:
    PIZZAROOT = os.path.normpath("C:\Data\Olivier\INRA\Codes\mslab\pizza")
    print "NT system"
else:
    PIZZAROOT = os.path.normpath(os.path.join(os.environ.get('PROJECT'),'pizza'))
    print "assume a LINUX machine"
os.chdir(PIZZAROOT)
print "current path: %s (default directory for data)" % os.getcwd()
print "available functions in src:\n\n %s\n" % os.listdir(os.path.normpath(os.path.join(PIZZAROOT,'src')))
...
...
# ALL SCRIPTS ARE NOW LAUNCHED FROM PIZZAROOT: $PROJECT/pizza
```

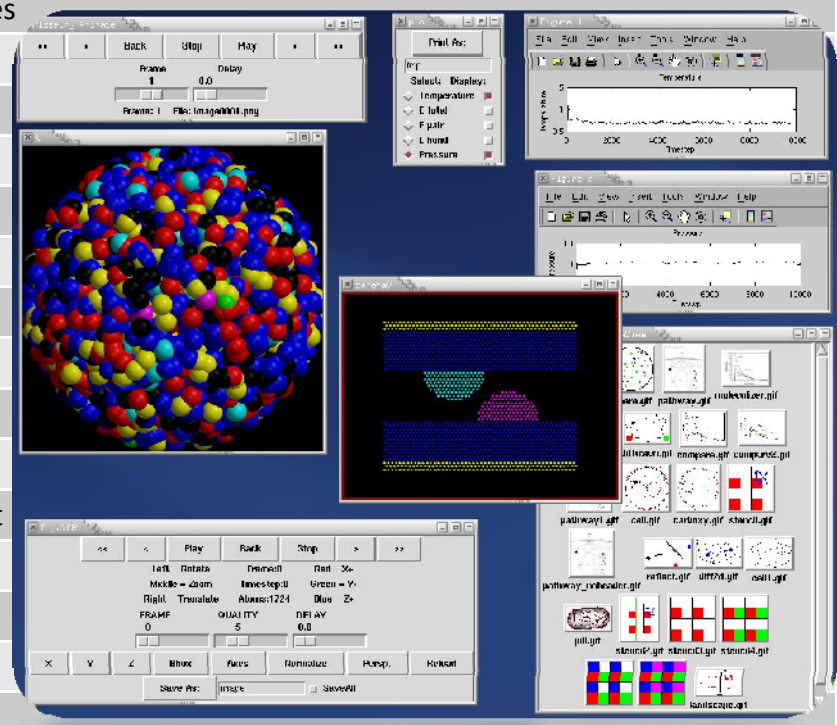
FILE: $PROJECT/pizza/src/DEFAULTS.py

```
# Pizza.py toolkit, www.cs.sandia.gov/~sjplimp/pizza.html
# Steve Plimpton, sjplimp@sandia.gov, Sandia National Laboratories...
...
import osPIZZA_TOOLS =  [os.path.normpath(os.path.join(os.getcwd(),'src'))]
PIZZA_SCRIPTS = [os.path.join(os.getcwd(),'scripts'),os.path.join(os.getcwd(),'examples')]
PIZZA_EXCLUDE = ["pizza", "DEFAULTS", "vizinfo"]
```

CUSTOM PIZZA TEST

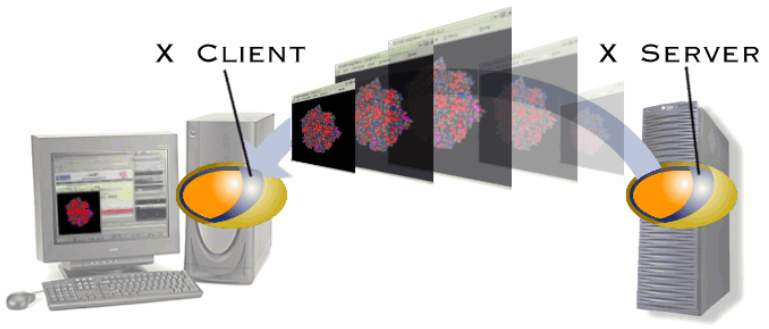
```
# simple test of chain tool
# creates tmp.data.chain file (see test_chain.py)

c = chain(500,0.7,1,1,2)
c.seed = 54321
c.build(25,10)

c.mtype = 2
c.btype = 2
c.blen = 1.5
c.dmin = 1.2
c.id = "end1"
c.build(10,25)

c.write("tmp.data.chain")

print "all done ... type CTRL-D to exit Pizza.py"
```

PIZZA Functions	http://www.cs.sandia.gov/~sjplimp/pizza.html
animate.py	Animate a series of image files
cdata.py	Read, write, manipulate ChemCell data files
chain.py	Create bead-spring chains for LAMMPS input
cfg.py	Convert LAMMPS snapshots to CFG format
clog.py	Read ChemCell log files and extract species data
data.py	Read, write, manipulate LAMMPS data files
dump.py	Read, write, manipulate dump files and particle attributes
ensight.py	Convert LAMMPS snapshots to Ensight format
gl.py	3d interactive visualization via OpenGL
gnu.py	Create plots via GnuPlot plotting program
histo.py	Particle density histogram from a dump
image.py	View and manipulate images
log.py	Read LAMMPS log files and extract thermodynamic data
matlab.py	Create plots via MatLab numerical analysis program
mdump.py	Read, write, manipulate mesh dump files
pair.py	Compute LAMMPS pairwise energies
patch.py	Create patchy Lennard-Jones particles for LAMMPS input
pdbfile.py	Read, write PDB files in combo with LAMMPS snapshots
plotview.py	Plot multiple vectors from a data set
rasmol.py	3d visualization via RasMol program
raster.py	3d visualization via Raster3d program
svg.py	3d visualization via SVG files
vcr.py	VCR-style GUI for 3d interactive OpenGL visualization
vec.py	Create numeric vectors from columns in file or list of vecs
vtk.py	Convert LAMMPS snapshots to VTK format
xyz.py	Convert LAMMPS snapshots to XYZ format



PIZZA with GUI

```
#ADD the export line to your bash or type it:

$ export PYTHONPATH="/usr/local/src/public/PyOpenGL-
      3.0.0a6/:/usr/local/src/public/modules-python/Imaging-
      1.1.6/PIL/:/usr/local/public/python-2.4.3/lib/python2.4/site-
      packages/Numeric/:/usr/local/genome/mgcat1.24-linux32-
      py23/:/usr/local/src/public/ctypes-1.0.2/build/lib.linux-x86_64-
      2.3/:/usr/local/public/python-2.4.3/lib/python2.4/site-packages/setuptools-0.6c7-
      py2.4.egg/:/usr/local/src/public/Togl-1.7/"

$ \pizza                    # \ is used to override the current alias
```



ssh –Y username@topaze.jouy.inra.fr
Check: echo $DISPLAY

LAMMPS EXAMPLES

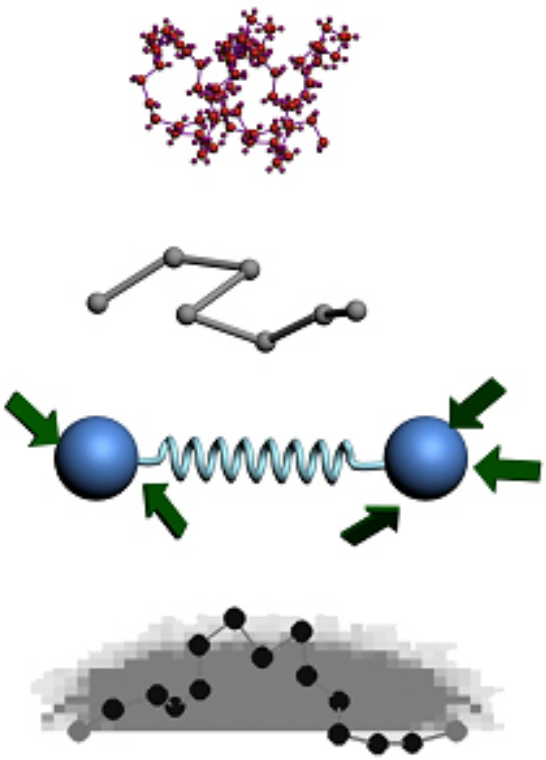
MODMOL 25-27 Feb 2008, Jouy-en-Josas

olivier.vitrac@agroparistech.fr

BEAD-SPRING POLYMER MELT

FENE: Finite Extendible Nonlinear Elastic Model (here: 2880 beads, 2715 bonds)

```
$PROJECT/bin/tools/chain <$PROJECT/examples/example.def.chain
   >$PROJECT/chain/data.chain
```



```
Polymer chain definition

0.8442    rhostar
592984    random # seed (8 digits or less)
2         # of sets of chains (blank line + 6 values for each set)
0         molecule tag rule: 0 = by mol, 1 = from 1 end, 2 = from 2
          ends

160       number of chains
16        monomers/chain
1         type of monomers (for output into LAMMPS file)
1         type of bonds (for output into LAMMPS file)
0.97      distance between monomers (in reduced units)
1.02      no distance less than this from site i-1 to i+1 (reduced
          unit)

5         number of chains
64        monomers/chain
2         type of monomers (for output into LAMMPS file)
2         type of bonds (for output into LAMMPS file)
1.05      distance between monomers (in reduced units)
1.12      no distance less than this from site i-1 to i+1 (reduced
          unit)
```

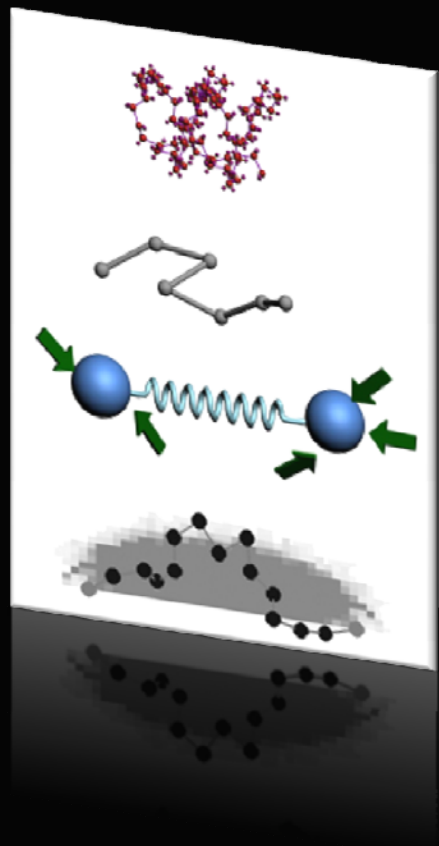
FILE: `$PROJECT/examples/example.def.chain`

BEAD-SPRING POLYMER MELT

FENE: Finite Extendible Nonlinear Elastic Model (here: 2880 beads, 2715 bonds)

pizza

```
# PIZZA-PYTHON SCRIPT using the method CHAIN
# Such script is equivalent to
#  $PROJECT/bin/tools/chain <$PROJECT/examples/example.def.chain
#  >$PROJECT/chain/data.chain
N = 2880                    # total number of monomers
rhostar = 0.8442            # density
#c = chain(N,rhostar)       #setup box with N monomers at reduced density rho
c = chain(N,rhostar,1,1,1) #x,y,z = aspect ratio of box (def = 1,1,1)
c.seed = 592984             #set random # seed (def = 12345)

c.mtype = 1                 #set type of monomers (def = 1)
c.btype = 1                 #set type of bonds (def = 1)
c.blen = 0.97               #set length of bonds (def = 0.97)
c.dmin = 1.02               #set min dist from i-1 to i+1 site (def = 1.02)
c.build(160,16)             #create 160 chains, each of length 16

c.mtype = 2                 #set type of monomers (def = 1)
c.btype = 2                 #set type of bonds (def = 1)
c.blen = 1.05               #set length of bonds (def = 0.97)
c.dmin = 1.12               #set min dist from i-1 to i+1 site (def = 1.02)
c.build(5,64)               #create 5 chains, each of length 64

c.write("data.chain")       #write out all built chains to LAMMPS data file
```



FILE: $PROJECT/pizza/examples/example_chain_data.py

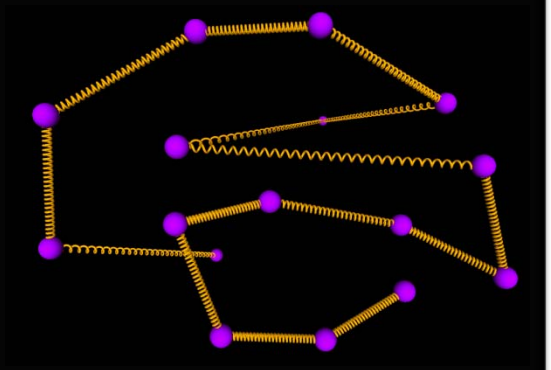
BEAD-SPRING POLYMER MELT

FENE stands for the finitely extensible nonlinear elastic model of a long-chained polymer. It simplifies the chain of monomers by connecting a sequence of beads with nonlinear springs. The spring force law is governed by inverse Langevin function or approximated by the Warner's relationship:

$$F_i = \frac{HR_i}{1-(R/L_{max})^2}$$

- 160+5-mer chains and FENE bonds:
- 2,880 monomers for 10^4 timesteps
- reduced density 0.8442 (liquid)
- force cutoff of 2^(1/6) sigma
- neighbor skin = 0.4 sigma
- neighbors/atom = 5 (within force cutoff)
- NVE time integration

```
# SCRIPT derived from FENE beadspring benchmark

units           lj
atom_style      bond
special_bonds   0.0 1.0 1.0

read_data       data.chain

neighbor        0.4 bin
neigh_modify    every 1 delay 1

bond_style      fene
bond_coeff      1 30.0 1.5 1.0 1.0
bond_coeff      2 30.0 1.5 1.0 1.0

pair_style      lj/cut 1.20
pair_modify     shift yes
pair_coeff      1 1 1.0 1.0 1.12
pair_coeff      2 2 1.5 1.1 1.20

fix             1 all nve
fix             2 all langevin 1.0 1.0 10.0 904297

thermo          100
timestep        0.012

run             10000
```

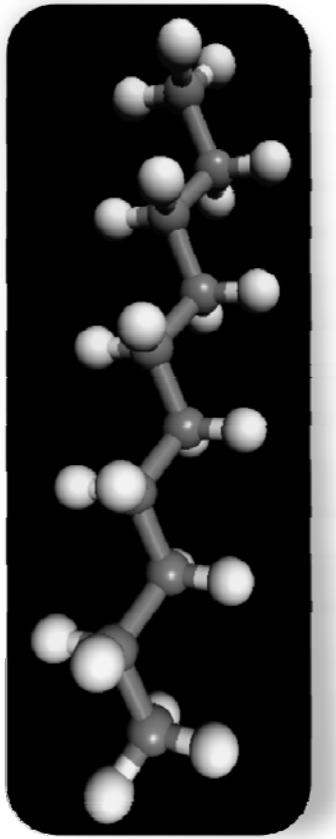


FILE: $PROJECT/examples/bidisperse.demo

```
# DEMO: bidisperse (advanced ex.)
# 26/02/08

echo $PROJECT                  # current queue
lslam  examples                # ls available templates

# PROJECT 1) BOX CREATION AND MINIMIZATION
#=========================================
lamjob examples/in.bidisperse.relax script 1 test.birelax []
def.chain.bidisperse in.bidisperse.nvt # copy templates, generate scripts

cdlam  test.birelax            # cd into the new project

$PROJECT/bin/tools/chain <def.chain.bidisperse >data.chain.bidisperse
                               # generate chains on a lattice
nano in.bidisperse.relax       # edit the relaxation script (no change required)

./run.sh                       # launch/submit the job on a single proc
u                              # check your job

# STEP 2) NVT DYNAMICS
#=====================
echo $QUEUE                    # current queue
export QUEUE="stage.q“         # change queue (empty assign the default “long.q”)

lamjob test.birelax/in.bidisperse.nvt run 4 test.binvt []
bidisperse.relax.restart.10000 # generate/submit the dynamics on 4 procs

u                              # check your job
```

ALL ATOM SIMULATION

EXAMPLE 1: CVFF (no warnings), shrink boundary conditions



decane

```
msi2lmp $PROJECT/examples/decane I CVFF
lamjob examples/in.decane script 1 decane [] decane.lammps05
```

```
units           real
atom_style      full
boundary        s s s

pair_style lj/cut 10.0
bond_style      harmonic
angle_style     harmonic
dihedral_style  harmonic
improper_style  none

read_data       decane.lammps05

neighbor        2.0 bin
neigh_modify    delay 5

timestep        2.0

thermo_style    multi
thermo          50
```

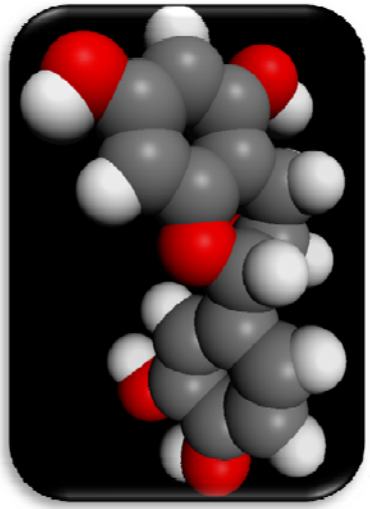
```
fix             1 all nvt
298.0 298.0 100.0

dump            1 all atom
100 dump.decane

minimize        1.0e-4 100
1000
run             10000
```

FILE: $PROJECT/examples/in.decane

ALL ATOM SIMULATION

EXAMPLE 2: CVFF (warnings), shrink boundary conditions

```
msi2lmp $PROJECT/examples/Epicatechin I cvff
lamjob examples/in.Epicatechin script 1 Epicatechin []
   Epicatechin.lammps05
```



Epicatechin

```
units           real
atom_style      full
boundary        s s s

pair_style lj/cut 10.0
bond_style      harmonic
angle_style     harmonic
dihedral_style  harmonic
improper_style  cvff

read_data
Epicatechin.lammps05

neighbor        2.0 bin
neigh_modify    delay 5

timestep        2.0
```

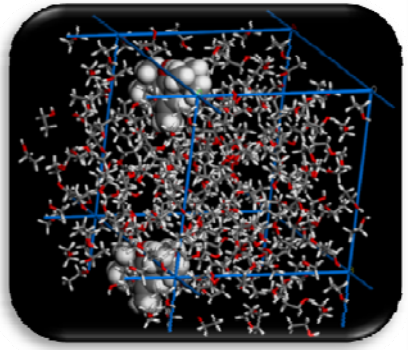
```
thermo_style    multi
thermo          50

fix             1 all nvt
298.0 298.0 100.0

dump            1 all atom
100 dump.decane

minimize        1.0e-4 100
1000
run             10000
```

FILE: $PROJECT/examples/in.Epicatechin

ALL ATOM SIMULATION

EXAMPLE 3: CFF (warnings), periodic boundary conditions, NPT

```
msi2lmp $PROJECT/examples/BHT/ethanol_BHTx1 II cff91
lamjob examples/BHT/in.ethanol_BHTx1 script 1 BHT []
   ethanol_BHTx1.lammps05
```



BHT in 200 molecules of ethanol

```
units           real
atom_style      full
boundary        p p p

pair_style lj/cut/coul/cut 10.0
8.0
bond_style      class2
angle_style     class2
dihedral_style  class2
improper_style  class2

read_data
ethanol_BHTx1.lammps05

neighbor        2.0 bin
neigh_modify    delay 5

timestep        1.0
```

```
thermo_style    multi
thermo          50

fix             1 all npt
298.0 298.0 100.0 xyz 1.0
1.0 1.0

dump            1 all atom
1000 dump.ethanol_BHTx1

run             10000
```

FILE: $PROJECT/examples/BHT/in.ethanol_BHTx1

MISCELLANEOUS

MODMOL 25-27 Feb 2008, Jouy-en-Josas

olivier.vitrac@agroparistech.fr

LEARNING PYTHON

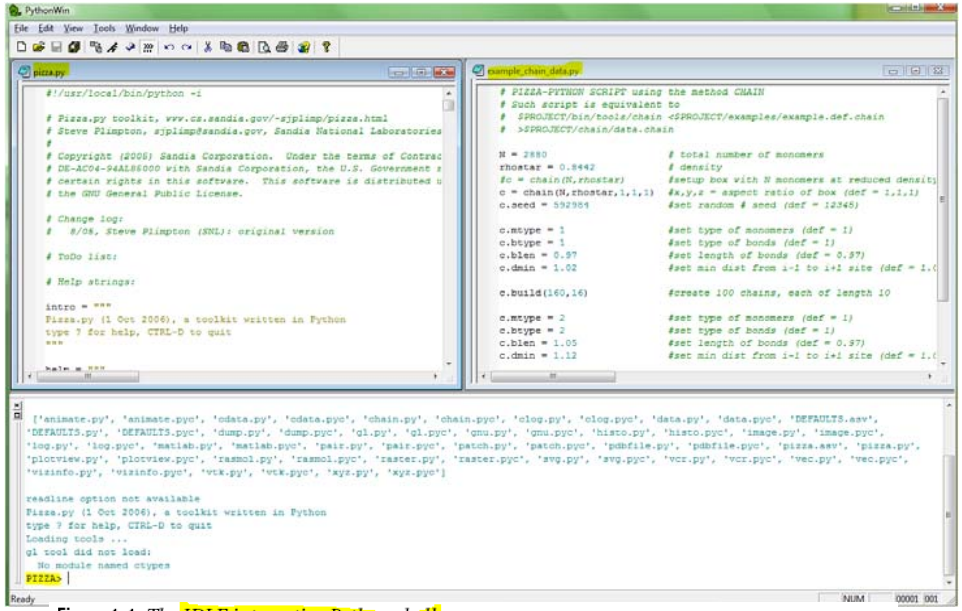


Figure 1-1. *The IDLE interactive Python shell*

Once you've got the IDLE interactive Python shell running, you can continue with the section "The Interactive Interpreter," later in this chapter.

Linux and UNIX

In many, if not most, Linux and UNIX installations, a Python interpreter will already be present. You can check whether this is the case by running the `python` command at the prompt, as follows:

```
$ python
```

Running this command should start the interactive Python interpreter, with output similar to the following:

```
Python 2.4 (#1, Dec  7 2004, 09:18:58)
[GCC 3.4.1] on sunos5
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Table 1-1. *Some Integrated Development Environments (IDEs) for Python*

Environment	Description	Available From . . .
IDLE	The standard Python environment	`http://www.python.org/idle`
Pythonwin	Windows-oriented environment	`http://www.python.org/windows`
ActivePython	Feature-packed; contains Pythonwin IDE	`http://www.activestate.com`
Komodo	Commercial IDE	`http://www.activestate.com`
Wingware	Commercial IDE	`http://www.wingware.com`
BlackAdder	Commercial IDE and (Qt) GUI builder	`http://www.thekompany.com`
Boa Constructor	Free IDE and GUI builder	`http://boa-constructor.sf.net`
Anjuta	Versatile IDE for Linux/UNIX	`http://anjuta.sf.net`
ArachnoPython	Commercial IDE	`http://www.python-ide.com`
Code Crusader	Commercial IDE	`http://www.newplanetsoftware.com`
Code Forge	Commercial IDE	`http://www.codeforge.com`
Eclipse	Popular, flexible, open source IDE	`http://www.eclipse.org`
eric	Free IDE using Qt	`http://eric-ide.sf.net`
KDevelop	Cross-language IDE for KDE	`http://www.kdevelop.org`
VisualWx	Free GUI builder	`http://visualwx.altervista.org`
wxDesigner	Commercial GUI builder	`http://www.roebling.de`
wxGlade	Free GUI builder	`http://wxglade.sf.net`

LEARNING PYTHON

1.1.5 Scripting Yields Shorter Code

Powerful dynamically typed languages, such as Python, support numerous high-level constructs and data structures enabling you to write programs that are significantly shorter than programs with corresponding functionality coded in Fortran, C, C++, C#, or Java. In other words, more work is done (on average) per statement. A simple example is reading an *a priori* unknown number of real numbers from a file, where several numbers may appear at one line and blank lines are permitted. This task is accomplished by two Python statements[2]:

```
F = open(filename, 'r'); n = F.read().split()
```

Trying to do this in Fortran, C, C++, or Java requires at least a loop, and in some of the languages several statements needed for dealing with a variable number of reals per line.

As another example, think about reading a complex number expressed in a text format like `(-3.1,4)`. We can easily extract the real part −3.1 and the imaginary part 4 from the string `(-3.1,4)` using a *regular expression*, also when optional whitespace is included in the text format. Regular expressions are particularly well supported by dynamically typed languages. The relevant Python statements read[3]

```
m = re.search(r'\(\s*([^,]+)\s*,\s*([^,]+)\s*\)', ' (-3.1, 4) ')
re, im = [float(x) for x in m.groups()]
```

We can alternatively strip off the parenthesis and then split the string `'-3.1,4'` with respect to the comma character:

```
m = ' (-3.1, 4) '.strip()[1:-1]
re, im = [float(x) for x in m.split(',')]
```

This solution applies string operations and a convenient indexing syntax instead of regular expressions. Extracting the real and imaginary numbers in Fortran or C code requires many more instructions, doing string searching and manipulations at the character array level.

The special text of comma-separated numbers enclosed in parenthesis, like `(-3.1,4)`, is a valid textual representation of a standard list (tuple) in

[2] Do not try to understand the details of the statements. The size of the code is what matters at this point. The meaning of the statements will be evident from Chapter 2.

[3] The code examples may look cryptic for a novice, but the meaning of the sequence of strange characters (in the regular expressions) should be evident from reading just a few pages in Chapter 8.2.

This Scientific Hello World script will demonstrate

- how to work with variables,
- how to initialize a variable from the command line,
- how to call a math library for computing the sine of a number, and
- how to print a combination of numbers and plain text.

The complete script can take the following form in Python:

```
#!/usr/bin/env python
import sys, math       # load system and math module
r = float(sys.argv[1]) # extract the 1st command-line argument
s = math.sin(r)
print "Hello, World! sin(" + str(r) + ")=" + str(s)
```

```
python hw.py 1.4
```

This command specifies explicitly that a program `python` is to be used to interpret the contents of the `hw.py` file. The number 1.4 is a command-line argument to be fetched by the script.

For the `python hw.py ...` command to work, you need to be in a console window, also known as a terminal window on Unix, and as a command prompt or MS-DOS prompt on Windows. The Windows habit of double-clicking on the file icon does not work for scripts requiring command-line information, unless you have installed PythonWin.

In case the file is given execute permission[1] on a Unix system, you can also run the script by just typing the name of the file:

```
./hw.py 1.4
```

or

```
hw.py 1.4
```

if you have a dot (.) in your path[2].

On Windows you can write just the filename `hw.py` instead of `python hw.py` if the `.py` is associated with a Python interpreter (see Appendix A.2).

When you do not precede the filename by `python` on Unix, the first line of the script is taken as a specification of the program to be used for interpreting the script. In our example the first line reads

[1] This is achieved by the Unix command `chmod a+x hw.py`.

[2] There are serious security issues related to having a dot, i.e., the current working directory, in your path. Check out the site policy with your system administrator.

LEARNING PYTHON

Table 5-2. Python expression operators and precedence

Operators	Description
`yield x`	Generator function send protocol (new in Release 2.5)
`lambda args: expression`	Anonymous function generation
`x if y else z`	Ternary selection expression (new in Release 2.5)
`x or y`	Logical OR (y is evaluated only if x is false)
`x and y`	Logical AND (y is evaluated only if x is true)
`not x`	Logical negation
`x < y, x <= y, x > y, x >= y, x == y, x <> y, x != y, x is y, x is not y, x in y, x not in y`	Comparison operators, value equality operators,[a] object identity tests, sequence membership
`x | y`	Bitwise OR
`x ^ y`	Bitwise eXclusive OR
`x & y`	Bitwise AND
`x << y, x >> y`	Shift x left or right by y bits
`-x + y, x - y`	Addition/concatenation, subtraction
`x * y, x % y, x / y, x // y`	Multiplication/repetition, remainder/format, division[b]
`-x, +x, ~x, x ** y`	Unary negation, identity, bitwise complement, binary power
`x[i], x[i:j], x.attr, x(...)`	Indexing, slicing, qualification, function calls
`(...), [...], {...}, `...` `	Tuple, list,[c] dictionary, conversion to string[d]

[a] In Python 2.5, value inequality can be written as either `X != Y` or `X <> Y`. In Python 3.0, the latter of these options will be removed because it is redundant; use `X != Y` for value inequality tests.

[b] Floor division (`X // Y`), new in Release 2.2, always truncates fractional remainders. This is further described in "Division: Classic, Floor, and True."

[c] Beginning with Python 2.0, the list syntax (`[...]`) can represent either a list literal or a list comprehension expression. The latter of these performs an implied loop and collects expression results in a new list.

[d] Conversion of objects to their print strings can also be accomplished with the more readable `str` and `repr` built-in functions, which are described in the section "Numeric Display Formats" later in this chapter. Due to its obscurity, the backticks expression `` `X` `` is scheduled to be removed in Python 3.0; use `repr(X)` instead.

Table 7-1. Common string literals and operations

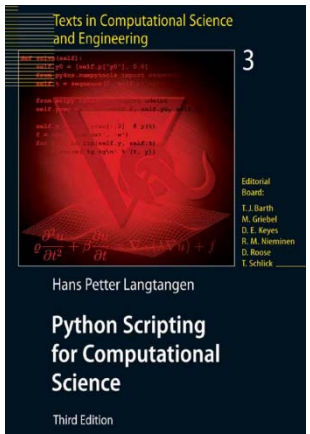
Operation	Interpretation
`s1 = ''`	Empty string
`s2 = "spam's"`	Double quotes
`block = """..."""`	Triple-quoted blocks
`s3 = r'\temp\spam'`	Raw strings
`s4 = u'spam'`	Unicode strings
`s1 + s2` `s2 * 3`	Concatenate, repeat

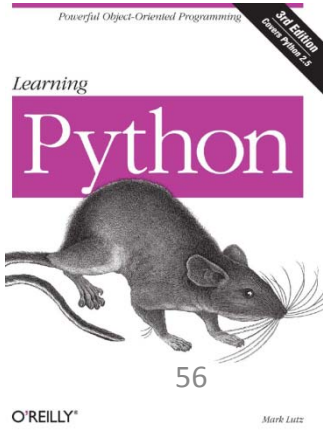
`s2[i]`
`s2[i:j]`
`len(s2)`
	Index, slice, length

`"a %s parrot" % type`	String formatting

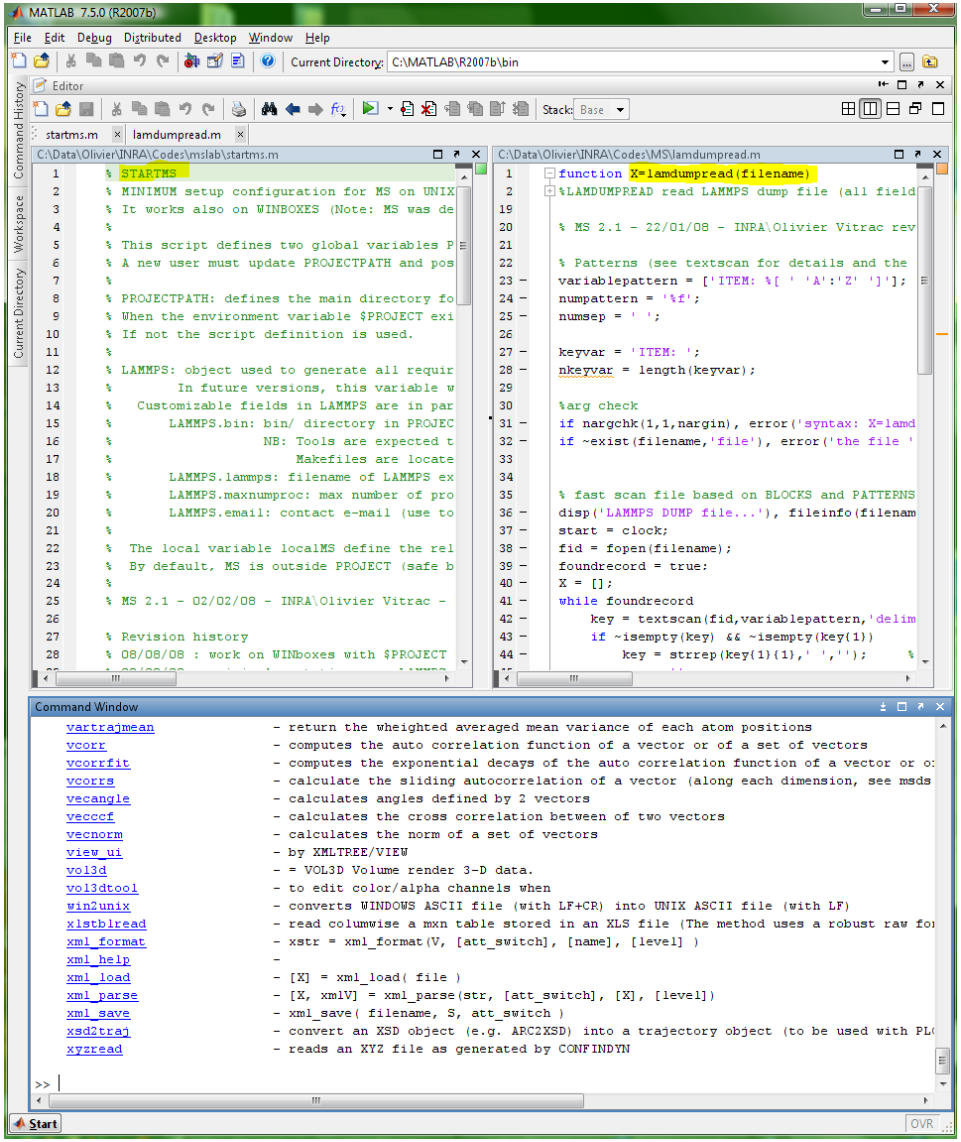
`s2.find('pa')`
`s2.rstrip()`
`s2.replace('pa', 'xx')`
`s1.split(',')`
`s1.isdigit()`
`s1.lower()`
	String method calls: search, remove whitespace, replacement, split on delimiter, content test, case conversion, etc.

`for x in s2` `'spam' in s2`	Iteration, membership





LEARNING MATLAB



3.1 Matrix operations

Matrices may be added, subtracted and multiplied using the conventional symbols +, − and ∗. Matrices may also be easily augmented thus

```
>>M = [1 1; 1 -1];
>>M = [M; M]
M =
   1    1
   1   -1
```

places a copy of the original matrix **M** below the current contents of **M** and hence produces a 4×2 matrix, whereas the command

```
>>M = [M M]
```

would produce a 2×4 matrix. Two matrices can be joined, side by side, provided that they have the same number of rows. They can also be joined one on top of the other, provided they have the same number of columns.

The colon operator is a special feature in MATLAB for constructing row vectors of evenly spaced values. The statement

```
>>x = 1:6
x =
     1     2     3     4     5     6
```

generates a row matrix **x** containing the integers from 1 to 6.

Individual elements of a matrix may be referenced by specifying their indices within parentheses. Thus,

```
>>M = [1 1; 1 -1];
x = M(1,1)
x =
     1
>>y = M(2,:)
y =
1 -1
```

http://www.mathworks.com/access/helpdesk/help/techdoc/index.html?/access/helpdesk/help/techdoc/learnmatlab/bqr_2pl.html

Managing commands and functions

Command	Meanings
help	On_line documentation_
doc	Load hypertext documentation
lookfor	Keyword search through the help entries
which	Locate functions
demo	Run demos_

Managing variables and the workspace

Command	Meanings
who	List current variables
whos	List current variables long form
load	Retrieve variables from disk
save	Save workspace variables to disk
clear	Clear variables and functions
from	memory
size	Size of matrix
length	Length of vector
disp	Display matrix or text

Working with files and the operating system

cd	Change current working directory
dir	Directory listing
delete	Delete file
!	Execute operating system command
unix	Execute operating system command & return result
diary	Save text of MATLAB session_

Controlling the command window

Command	Details
cedit	Set command line edit/recall facility parameters
clc	Clear command window
home	Send cursor home
format	Set output format
echo	Echo commands inside script files
more	Control paged output in command window
quit	Terminate MATLAB

Note that in some cases, the upper limit may not be attainable thing. For example, in case of 1:0.3:2, the upper limit is not reached and the resulting vector in this case is:

```
» 1:0.3:2
ans =
    1.0000    1.3000    1.6000    1.9000
```

If only two of the 'range' specifications are given then a unit step size is automatically assumed. For example 1:4 means:

```
» 1:4
ans =
     1     2     3     4
```

In case, the range is not valid, an error message is issued:

```
» 1:-1:5
ans =
  Empty matrix: 1-by-0
```

Here, the range of numbers given for the generation of row vector was from 1 to 5 in steps of -1. Clearly, one can not reach 5 from 1 using -1 step size. Therefore, the Matlab indicates that this is an empty matrix.

Sections of a Vector

Let us define a vector using the range notation:

```
» W=[1:3, 7:9]
W =
     1     2     3     7     8     9
```

Now, we would like to extract the middle two elements of this vector. This can be done with the range notation again. As you can see, the middle two elements are 3:4 range. Therefore, the required part of vector can be obtained as:

```
» W(3:4)
ans =
7
```

This really is the required part. There are many interesting things that can now be done using the range notation. For example, range 6:-1:1 is the descending range and when used with part-extraction of vector, it gives:

```
» W(6:-1:1)
ans =
     9     8     7     3     2     1
```

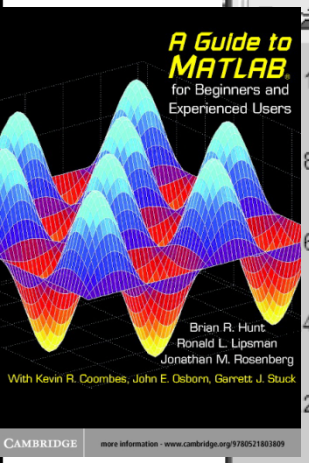
which is the vector W with all entries now in reverse order. So, a vector can be flipped easily. The 'size' function yields the length of a vector. For a given vector V, V(size(V):-1:1) will flip it. Note that flipping of sections of a vector is also possible.

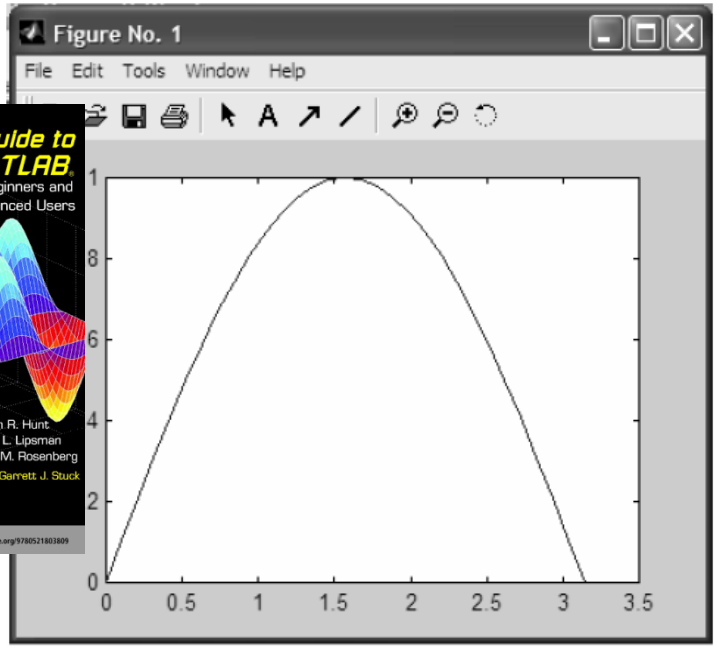
Elementary Plots and Graphs

Matlab offers powerful graphics and visualization tools. Let us start with some of the very basic graphics capabilities of Matlab. The graph of sine function in 0 to π can be obtained in the following way:

```
» N=30; h=pi/N; x=0:h:pi; y=sin(x); plot(x,y)
```

Here, in the first step, the total number of sampling points for the function is defined as N and it is assigned a value 30. Next, the step size 'h' is defined and the x row vector of size N+1 is defined along with the corresponding y row vector composed of the function values. The command 'plot(x,y)' generates the graph of this data and displays it in a separate window labeled Figure No. 1 as shown below:





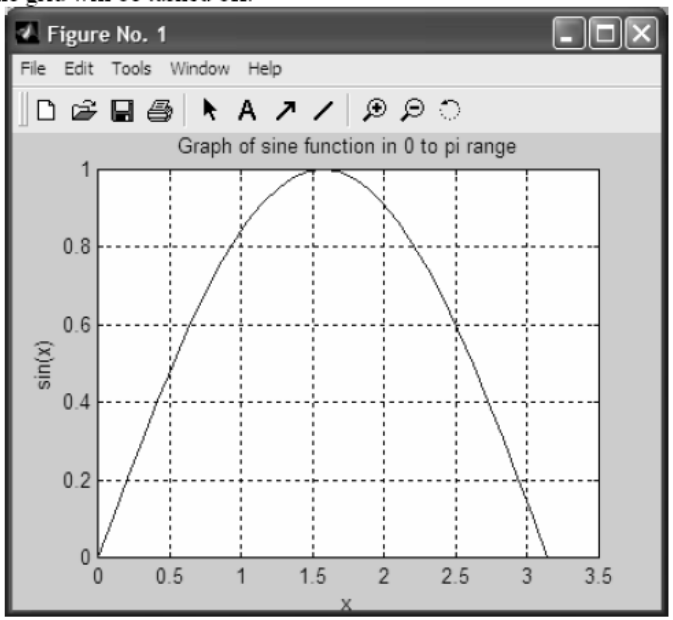
The graph displayed in this window can be zoomed-in and zoomed-out. Both x-any y-axes can also be rescaled with the help of mouse and using appropriate buttons and menu items.

The graph title, x- and y-labels can be assigned using the following commands:

```
>> title('Graph of sine function in 0 to pi range')
>> xlabel('x')
>> ylabel('sin(x)')
```

Note that by using these commands as such, one gets the corresponding response on the graph window immediately.

The grid lines on the graph can be switched on or off using the 'grid' command. By issuing this command once, grid will be turned on. Using it again, the grid will be turned off.



Matlab allows users to change the color as well as the line style of graphs by using a third argument in the plot command. For example, plot(x,y,'w-') will plot x-y data using white (w) color and solid line style (-). Further such options are given in the following table:

Color Symbol	Color	Line Symbol	Line type
y	Yellow	.	Point
m	Magenta	o	Circle
c	Cyan	x	x-mark
r	Red	+	Plus mark
g	Green	-	solid
b	Blue	*	Star
w	White	:	Dotted
b	Black	-.	Dash-dot
		--	dashed

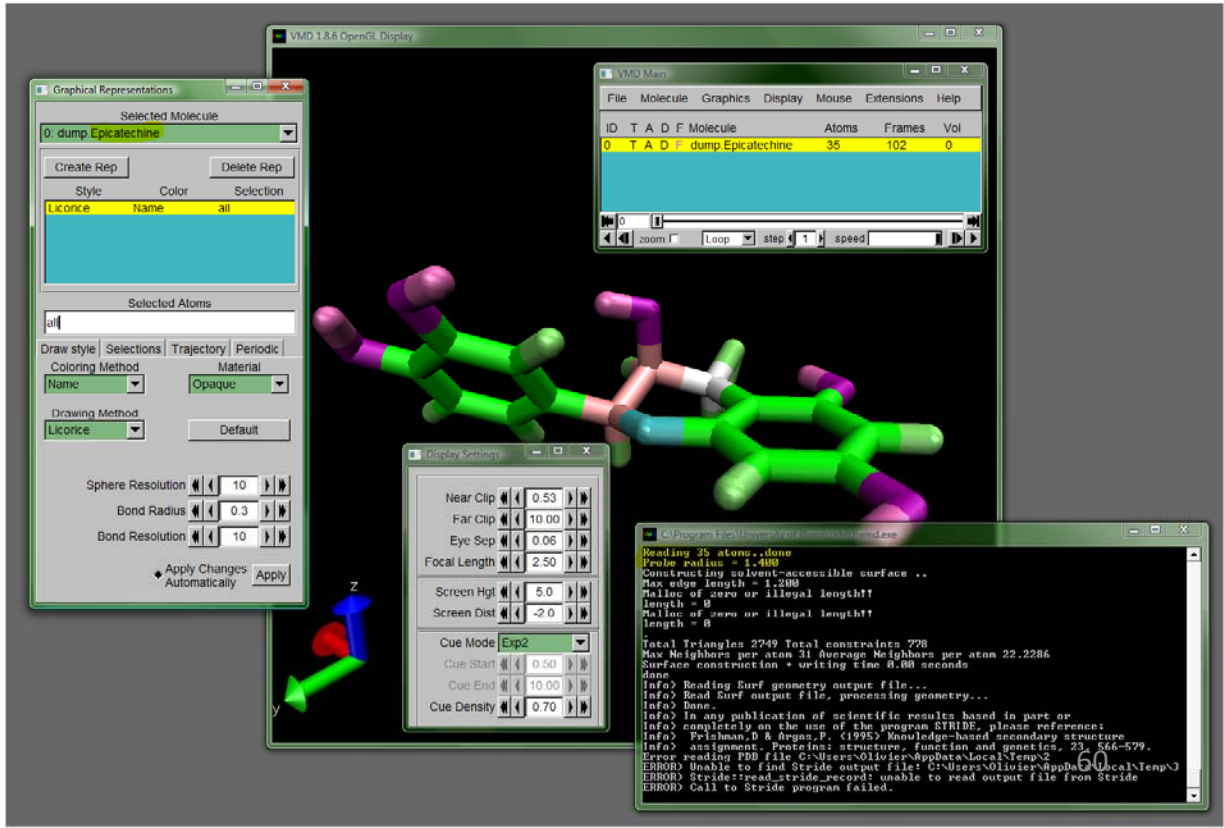
VISUALIZATION OF DUMP FILES

VMD
Visual Molecular Dynamics

THEORETICAL and COMPUTATIONAL BIOPHYSICS GROUP
NIH RESOURCE FOR MACROMOLECULAR MODELING AND BIOINFORMATICS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

http://www.ks.uiuc.edu/Development/Download/download.cgi?PackageName=VMD

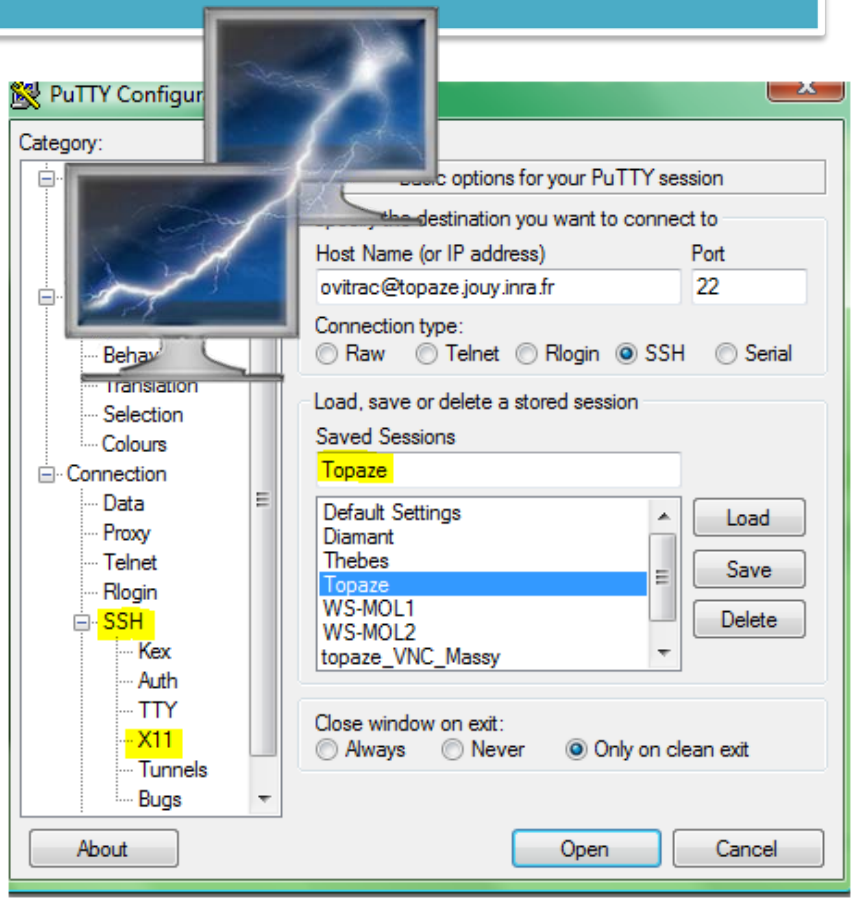
Version 1.8.6 (2007-04-07)



WIN2LINUX Solutions

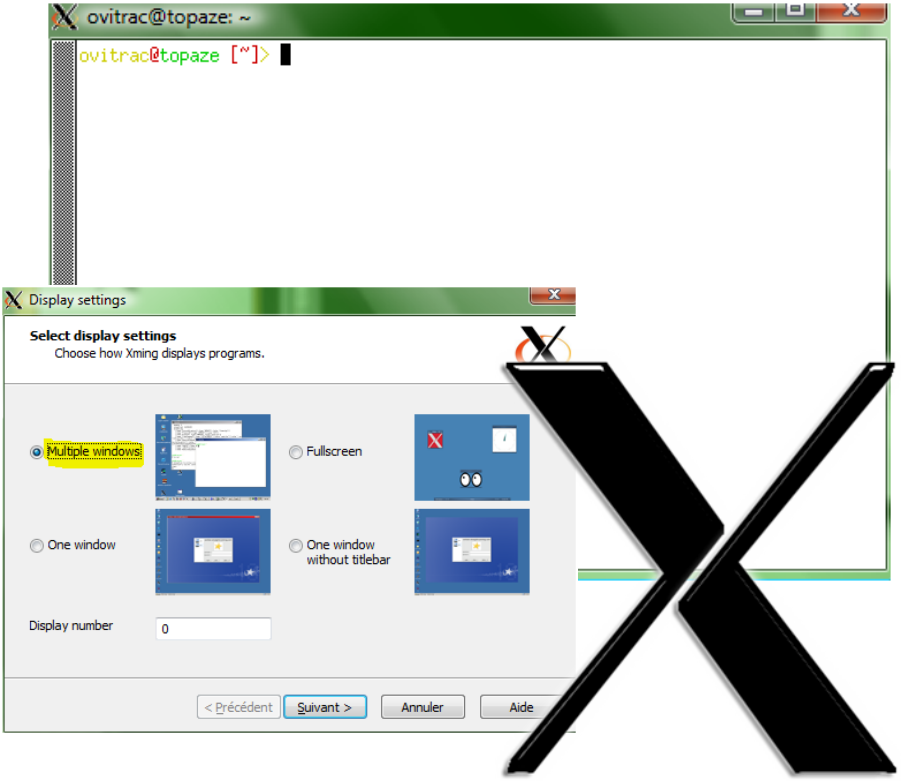
http://cygwin.com/
(free)

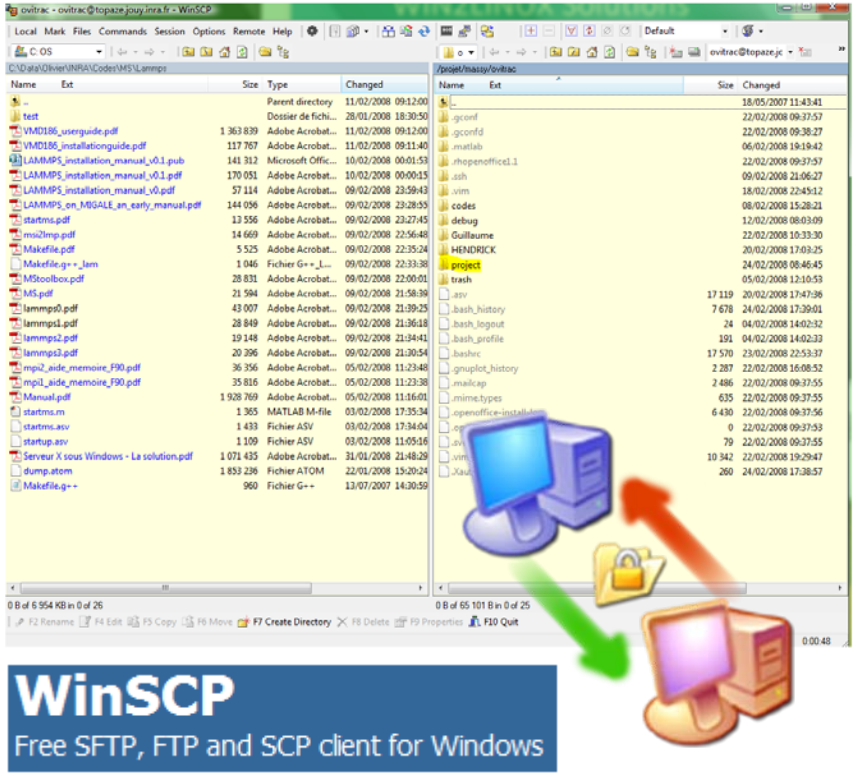
```
# Cygwin with SSHD: requires zlib, tcpwrappers, o
# Install
mkpasswd -l > /etc/passwd
mkgroup -l > /etc/group
exit
# Relogin
which sshd
ssh-host-config -y
cygrunsrv -S sshd
sc description sshd
# create public key
ssh-keygen -t dsa
# send key
cat .ssh/*.pub | ssh ovitrac@topaze.jouy.inra.fr tee -a .ssh/authorized_keys
# autologin
ssh ovitrac@topaze.jouy.inra.fr
```



http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
(free)

WIN2LINUX Solutions

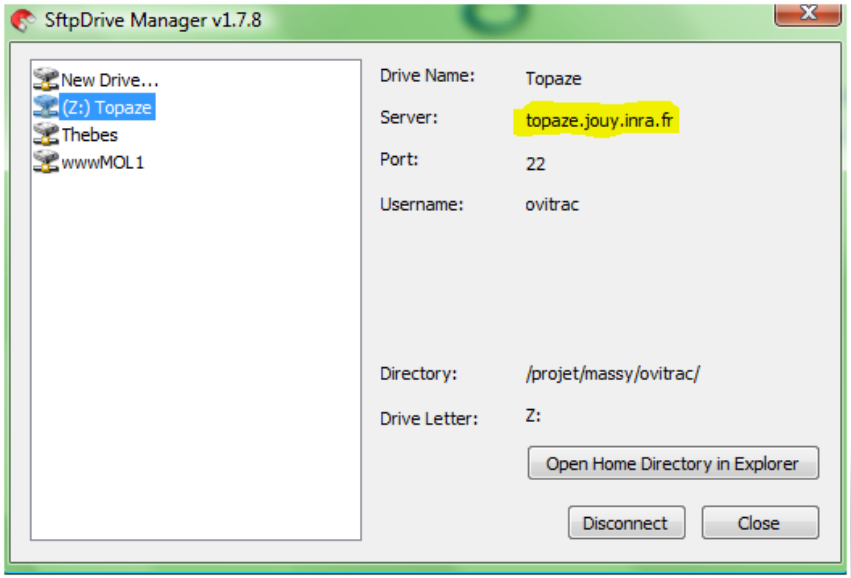




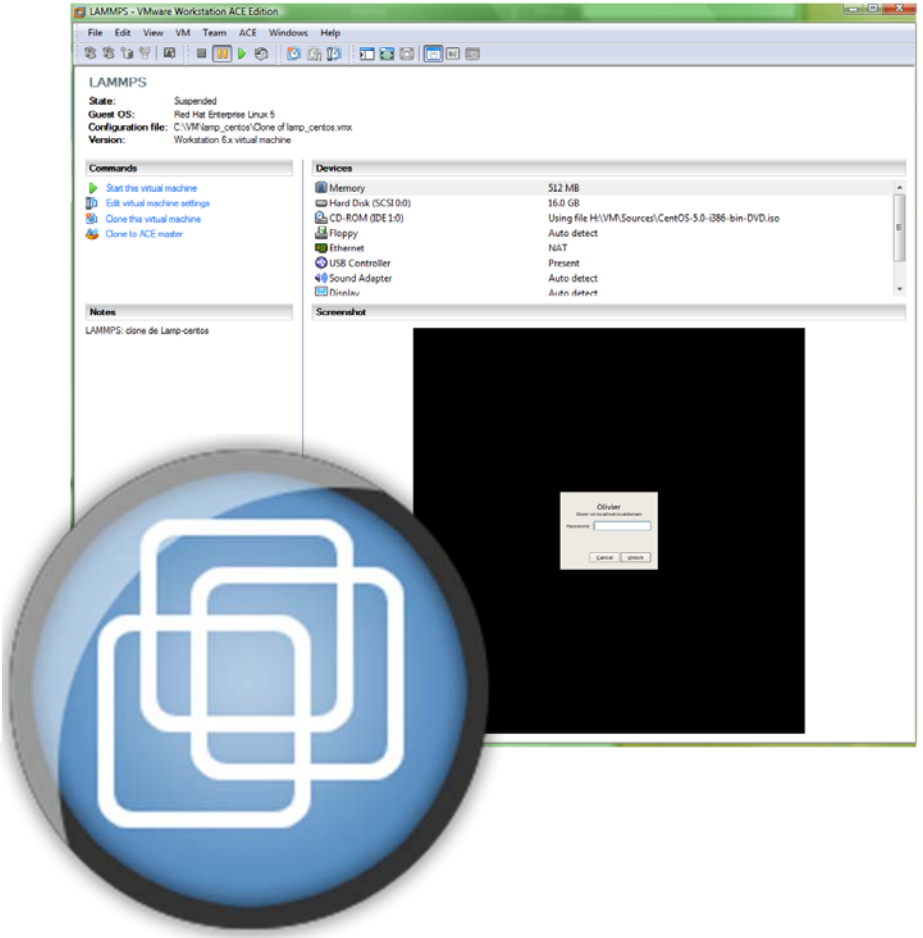
http://sourceforge.net/projects/xming
(free)

http://winscp.net/eng/docs/lang:fr
(free)

WIN2LINUX Solutions



http://www.sftpdrive.com/
SHAREWARE



http://www.vmware.com/products/server/
(VMWARE SERVER: currently Free)

MAKING LAMMPS

MODMOL 25-27 Feb 2008, Jouy-en-Josas

olivier.vitrac@agroparistech.fr

$PROJECT\make\lammps-31Jan08\src\MAKE\Makefile.g++_lam_all_100208

```
# g++ = RedHat Linux box, g++, LAM, FFTW
# INRA\Olivier Vitrac 13/02/08 (for LAMMPS 10/02/08)
#
# Compilation is ok
# >> TEST EXAMPLE:
# >> cd $PROJECT/../testlam; ./lmp_g++_lam <in.lj
#
# >> BUG REPORT
# The MPIRUN hangs.
# cd $PROJECT/../testlam;lamboot; mpirun -np 1 lmp_g++_lam <in.lj


SHELL = /bin/sh
#.IGNORE:

# System-specific settings
CC =       g++
CCFLAGS =   -g -O -I/usr/local/public/lam/include \
          -I/usr/include -DFFT_FFTW -DLAMMPS_GZIP -DMPICH_IGNORE_CXX_SEEK
DEPFLAGS =  -M
LINK =      g++
LINKFLAGS = -g -O -L/usr/local/public/lam/lib \
          -L/usr/lib64
USRLIB =    -lfftw -llammpio -llammpi++ -llamf77mpi -lmpi -llam
SYSLIB =    -lpthread -ldl
ARCHIVE =   ar
ARFLAGS =   -rc
SIZE =      size

# Link target
$(EXE): $(OBJ)
   $(LINK) $(LINKFLAGS) $(OBJ) $(USRLIB) $(SYSLIB) -o $(EXE)
   $(SIZE) $(EXE)

# Library target
lib:    $(OBJ)
   $(ARCHIVE) $(ARFLAGS) $(EXE) $(OBJ)

# Compilation rules
%.o:%.cpp
   $(CC) $(CCFLAGS) -c $<
%.d:%.cpp
   $(CC) $(CCFLAGS) $(DEPFLAGS) $< > $@

# Individual dependencies
DEPENDS = $(OBJ:.o=.d)
include $(DEPENDS)
```

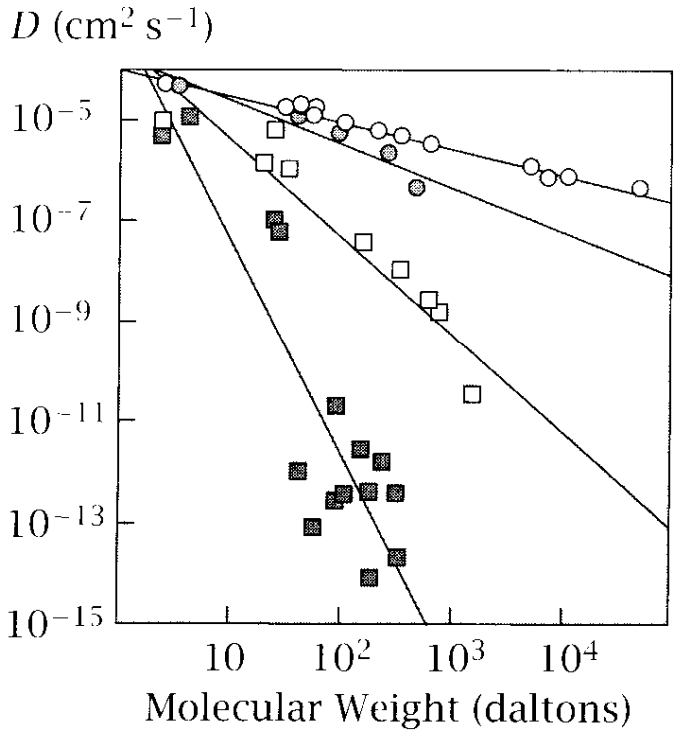


Figure 18.14 Diffusion coefficient as a function of solute molecular weight in water (○) and in three polymers: silicone rubber (◉), natural rubber (□), and polystyrene (■). The regression lines through the measurements have slopes of −0.51 (water), −0.86 (silicone rubber), −1.90 (natural rubber), and −4.20 (polystyrene). Source: adapted from RW Baker, *Controlled Release of Biologically Active Agents*, Wiley, New York, 1987.

Dill, Ken A.

Molecular driving forces: statistical thermodynamics in chemistry and biology / Ken A. Dill, Sarina Bromberg
p. cm.
Includes bibliographical references and index.
ISBN 0-8153-2051-5
1. Statistical thermodynamics. I. Bromberg, Sarina. II. Title.

QC311.5 .D55 2002
536'.7-dc21

2001053202

Published by Garland Science, a member of the Taylor & Francis Group
29 West 35th Street, New York, NY 10001-2299, USA
11 New Fetter Lane, London EC4P 4EE, UK