Polymer Initialization Nano Composite Generator (PING)

Table of Contents

1 Introduction

This algorithm was developed in Fortran 77 to allow the random walk algorithm to be used for initializing the polymer in nano composites. In this algorithm there are two RNG algorithms. One is the traditional seed based RNG that uses a table or functions. The other is a novel algorithm which uses mirrors along with the nano particle position and chain position. The final output of the algorithm is the initialized polymer written in the format of a LAMMPS data file. This manual will not go into the details of how to build a LAMMPS data file as this information can be found inside the LAMMPS package itself. But, the manual will explain how to set up the software to build an initialized polymer. Additionally, the requirements and limitations of compiling the software will be discussed.

2 Compiler Requirements and Limitations

To run this software, the g77 compiler needs downloading. This software has not been tested with other Fortran 77 compilers; so, other compilers may not compile the code correctly. If g77 can not be installed on your machine, the code can easily be converted to Fortran 90.

To convert the code to Fortran 90, two steps are required. The first is to replace all of the “c”s at the beginning of comment lines with “!”. The comment lines are not indented and are left justified. The second step is to change the file from “.f” to “.f90”. This conversion process has been tested with the compiler gfortran.

To compile, open up a terminal and type “compiler” “software file” -o “program name” and than hit enter. A file with “program name should be created”. Just type in ./”program name”. If you are worried about optimization flags for additional speed, the software runs very fast and so there was no need for extra flags to be added. This also means if optimization flags are added, the software may not run correctly.

3 Setting up the Algorithm to Initialize the Polymer Chains

3.1 Setting up the Parameters for the Polymer Chains

The parameters needed to set up the polymer chains are below. The chainnum is the number of polymer chains that are produced by the algorithm. The unitnum is the number of units in each polymer chain. The atomnum is the number of atoms or particles in each unit. The kuhncount is the number of units placed before the mirror based RNG algorithm is used. The kuhncount should either be set to match the Kuhn length or to 0 which makes the mirror based RNG algorithm run every time a unit is placed.

```
      integer chainnum, atomnum, unitnum, kuhncount
      parameter (chainnum=30, unitnum=100, atomnum=7,kuhncount=7)
```

3.2 Setting up the Number of Coefficients for Molecular Structures

The different molecular structures are the the atoms, bonds, angles, dihedrals and impropers. For each of these structures, there are coefficients which effect how the simulations are run. The atomtypenum is the number of different atom types in the polymer. This numbers effects the number of atomic masses needed in the data file. The pair potentials which also use atom types are not stored in the data file but are in LAMMPS scripts. The bondtypenum is the number of bond potentials needed to describe the polymer chain. The bond potentials are stored in the matrix bondcoef. The first number next to bondcoef should be the number of parameters needed for the bond potentials. The angletypenum is the number angle potentials needed to describe the polymer chain. Again, the first number next to anglecoef is the number of parameters required by the potential. The diheredraltypenum and dihedralcoef correspond to the dihedrals and follow the same pattern as the angles. The impropertypenum and impropercoef follows the same pattern as the angles as well.

```
    integer atomtypenum, bondtypenum, angletypenum, dihedraltypenum,
impropertypenum
    parameter (atomtypenum=6, bondtypenum=6, angletypenum=6,
dihedraltypenum=6,impropertypenum=1)
    double precision bondcoef(2,bondtypenum),
anglecoef(2,angletypenum)
    double precision
dihedralcoef(5,dihedraltypenum),impropercoef(2,impropertypenum)
```

3.3 Setting up the Size of Different Molecular Structures

The code below shows the size of different matrices which make up the molecular structure. The atoms is the atomic information in one unit of the polymer chain. The bondinit matrix is the bond information for the initial unit of the polymer chain and bonds matrix is the bond information for all other units in the polymer chain. The angleinit matrix and angles matrix are for the angles information and follow the same pattern as the bond matrixes. The dihedral information is stored in dihedralinit and dihedrals and follows the same pattern as the bond matrixes. The impropers matrix does not follow this pattern. In all matrixes, the first number corresponds to the amount of descriptors needed for a molecular structure, and the second number corresponds to the number of the molecular structures in each unit. These values can be changed to fit the requirements for each structure.

```
    double precision atoms(5,atomnum)
    integer bondinit(3,6), bonds(3,7), angleinit(4,7), angles(4,11)
    integer dihedralinit(5,6), dihedrals(5,13), impropers(5)
```

3.4 Setting up the Box Length and Nano Particle Size

The code below sets up the box length and the nano particle size. The lattice size is the box length in angstroms. The hspherer is the radius of the nano particle in angstroms. If there is no nano particle, set the value to 0.0.

```
    double precision hspherer, latticesize
    parameter (hspherer=0.0, latticesize=75.017531)
```

3.5 Setting up the Different Molecular Structures and Their Coefficients

The code below shows how to set up the different molecular structures and their coefficients. Each row corresponds to the descriptors needed for a single item of the structure. The number of items in each row should correspond to the first number after the matrix declaration. The number of rows in each structure correspond to the second number after the matrix declaration. The only exception is the for the variables adition, and masses, since these are vectors and not matrixes. All of these structures follow the rules in the LAMMPS manual except the id value and molecular id have been removed. These are taken care of in the code.

```
data atoms/3, .11, 0, 0, 0,
     &     2, 0, -1.52429914864554, 0,.2193447182826,
     &     1, 0, .333005967244697,    -1.23820341059349,    -.
8529708903439,
     &     4, .31, .75,     0,    1.29903810567666,
     &     5, -.37, .262586755537238, .528275496308508,
    2.28481360198517,
     &     6, -.31, 2.03631224555881, -.619208163705874,
    1.38199626938253,
     &     1, .26, 2.43869486125905,  -.42813272480992,
    2.74084814035043/,
     &     adition/.333005967244697,  1.23820341059349,     -.
852970890343899/,
     &     masses/12.0107, 12.0107, 12.0107, 12.0107, 15.9994,
15.9994/,
     &     bondinit/1, 1, 3,
     &     2, 1, 2,
     &     3, 1, 4,
     &     4, 4, 5,
     &     5, 4, 6,
     &     6, 6, 7/,
     &     bonds/2, 1, -5,
     &     1, 1, 3,
     &     2, 1, 2,
     &     3, 1, 4,
     &     4, 4, 5,
     &     5, 4, 6,
     &     6, 6, 7/,
     &     bondcoef/368, 1.539,
     &     300, 1.549,
     &     326, 1.517,
     &     968, 1.209,
     &     471, 1.360,
     &     342, 1.446/,
     &     angleinit/2, 2, 1, 3,
     &     2, 3, 1, 4,
     &     2, 2, 1, 4,
     &     4, 1, 4, 5,
     &     3, 1, 4, 6,
     &     5, 6, 4, 5,
     &     6, 7, 6, 4/,
     &     angles/1, -6, -5, 1,
     &     2, -5, 1, 4,
     &     2, -5, 1, 3,
     &     2, 2, 1, 3,
     &     2, 2, 1, 4,
     &     2, 2, 1, -5,
     &     2, 3, 1, 4,
     &     3, 6, 4, 1,
     &     4, 5, 4, 1,
     &     5, 6, 4, 5,
     &     6, 7, 6, 4/,
     &     anglecoef/89.5, 113.3,
     &     87.9, 109.47,
     &     74.5, 111.4,
     &     63.3, 125.6,
     &     126.5, 123.0,
     &     84.8, 116.4/,
     &     dihedralinit/4, 3, 1, 4, 5,
     &     4, 2, 1, 4, 5,
     &     3, 3, 1, 4, 6,
     &     3, 2, 1, 4, 6,
     &     5, 1, 4, 6, 7,
     &     6, 7, 6, 4, 5/,
     &     dihedrals/1, 1, -5, -6, -4,
     &     2, 1, -5, -6, -3,
     &     1, 2, 1, -5, -6,
     &     1, 3, 1, -5, -6,
     &     2, 4, 1, -5, -6,
     &     4, 5, 4, 1, -5,
     &     3, 6, 4, 1, -5,
     &     4, 3, 1, 4, 5,
     &     4, 2, 1, 4, 5,
     &     3, 3, 1, 4, 6,
     &     3, 2, 1, 4, 6,
     &     6, 7, 6, 4, 5,
     &     5, 7, 6, 4, 1/,
     &     dihedralcoef/.00043047, .88728, -.0058908, 3.48522, .
0083951,
     &     -.69938, .88749, 1.39251, 3.48489, .010033,
     &     -.00011517, 1.08001, .00042681, 1.76002, -.00036242,
     &     1.90050, -.78234, -3.80681, 1.04477, .0094108,
     &     2.22989, 4.54992, -4.45939, -.63999, -.00051598,
     &     2.20044, 1.04739, -4.40604, -1.39492, .0085622/,
     &     impropers/1, 4, 1, 5, 6/,
     &     impropercoef/0.735006480784105,  0/
```

3.6 Calculating the Number of Bonds, Angles, Dihedrals, and Impropers

This is the code used to calculate the number of bonds, angles, dihedrals and impropers. These values are written in the LAMMPS data file. If these equations dont match your specific system, than change them.

```
    bondnum=chainnum*(unitnum-1)*atomnum+chainnum*(atomnum-1)
    anglenum=chainnum*(unitnum-1)*(atomnum+4)+chainnum*(atomnum)

dihedralnum=chainnum*(atomnum-1)+chainnum*(unitnum-1)*(2*atomnum-1)
    impropernum=chainnum*unitnum
```

3.7 Setting up the Random Number Generator in Fortran

The code below is used to set up the random number generator that comes with Fortran. The value 14 in the code below is the seed number. This random number generator is used to place the initial unit in the polymer chain.

```
call srand(14)
```

3.8 How to Remove Bonds, Angles, Dihedrals, or Impropers From the Data File

If for some reason you don’t need to place the bonds, angles, dihedrals, or impropers in the LAMMPS data file. The code that takes care of each of these sections can either be removed or commented out. An example of the angles and angle coeficient sections are shown below.

```
C write the angles section
    write (25,*) 'Angles'
    write (25,*)
    count=1
    do 40 i=1,chainnum
           do 41 j=1,atomnum
                write (25,*) count, angleinit(1,j), angleinit(2,j)+
(i-1)*unitnum*atomnum,
     &               angleinit(3,j)+(i-1)*unitnum*atomnum,
angleinit(4,j)+(i-1)*unitnum*atomnum
                count=count+1
 41        continue
           do 42 j=2,unitnum
                do 44 k=1,atomnum+4
```

```
                      write (25,*) count, angles(1,k), angles(2,k)+
(i-1)*unitnum*atomnum+(j-1)*atomnum,
     &                      angles(3,k)+(i-1)*unitnum*atomnum+
(j-1)*atomnum,
     &                      angles(4,k)+(i-1)*unitnum*atomnum+
(j-1)*atomnum
                      count=count+1
 44              continue
 42        continue
 40 continue
    write (25,*)

C write the angle coeficients section
    write (25,*) 'Angle Coeffs'
    write (25,*)
    do 43 i=1,angletypenum
           write (25,*) i, anglecoef(1,i), anglecoef(2,i)
 43 continue
    write (25,*)
```