Computational Materials Mini-App1 (CM_mini1) Simulation Code

V. Yamakov
National Institute of Aerospace

The Computational Materials miniapp1 (CM_mini1) code is designed to demonstrate the internal energy calculation of a given simulated atomic structure (e.g., Ni3Al metal alloy crystal). The implemented software algorithm is an essential (but not the only) part of Molecular dynamics and Monte Carlo methods for simulation of the thermodynamic evolution of metal alloy systems at atomic level, and predicting their thermodynamic state, phase diagram, chemical composition and mechanical properties. The CM_mini1 code is being released to serve as a test and training case for students and academia for optimization on multicore cpu computers or computers utilizing graphic processing units (GPUs). The effort is related to the High-Performance-Computing Incubator project at NASA Langley Research Center with PI: Dana Hammond. The CM_mini1 code is extracted from a part of the Parallel Grand Canonical Monte Carlo simulation code (ParaGrandMC) [1]. A successful optimization of the CM_mini1 code will allow to increase the computational efficiency and the capability of the ParaGrandMC software to better utilize the new generation of hardware (manycore and GPU systems) after incorporating the tested features through this miniapp.

Command Line Options (all are optional)

-n [#]– specifies the number [# = 1,2, ..] of iterations for a repeated energy calculation.
Default value: -n 1.
Example: MC_mini1 –n 10 ! executes 10 energy calculations in a loop.

-r – Introduces random displacements for each atom after every iteration of energy calculation. The first iteration is always performed on the input structure without random displacements. The range of displacements is between ±0.01 Å.
Default value: no randomization, i.e., each iteration is performed on the same structure.
Example: MC_mini1 –n 10 -r ! executes 10 energy calculations in a loop while randomly displacing the atoms after each iteration.

-v # – selects the version number [# = 0, 1, ..] of the code, when several versions of one or more subroutines are implemented for comparison. What each version does has to be specified by the programmer in the code.
Default value: -v 0.
Example: MC_mini1 –v 1 ! executes version 1 of the code, if defined by the programmer.

Input Files

ParaGrandMC needs the following input files: an input model structure file, **structure.plt**, and a file defining the interatomic interactions, **pot.dat**, together with a set of files describing the implemented interatomic potential.

Structure file: structure.plt

The input atomic structure is given in a text file, named "**structure.plt**", which lists all of the atoms in the structure with their chemical type and positions in Cartesian coordinates. To preserve compatibility with other codes, the file format follows a simplified version of the format, called "plt", where some of the header information is preserved, but not used. The first several lines in the file, which start with the "#" symbol, form the file header describing the dimensions of the system and the number of atoms it contains. The atoms are listed after the header. Each atom is described by its identification (ID) number, atomic position given in Cartesian (x, y, z) coordinates in Å, a number identifying the associated chemical element, and a code number for the constraint degrees of freedom for this atom, if any. If the structure file contains atomic velocities, they are listed as a follow up list after the atomic coordinates.

Example:

--- structure.plt ---

```
# -0.1792446164E+02 -0.1788684653E+02 -0.1782515785E+02  ! -h11/2 -h22/2 -h33/2 initial
#  0.1792446164E+02  0.1788684653E+02  0.1782515785E+02  !  h11/2  h22/2  h33/2 initial
# -0.1792446164E+02 -0.1788684653E+02 -0.1782515785E+02  ! -h11/2 -h22/2 -h33/2 current
#  0.1792446164E+02  0.1788684653E+02  0.1782515785E+02  !  h11/2  h22/2  h33/2 current
#       2    4000    4000    4000                       ! n/a  N_atoms, N_buf, N_free
#    0.67248840E+01       1       1       1             ! n/a
#      -1      -1      -1                               ! n/a
#       0       0                                       ! n/a
# -0.4430826192E+01     922.6                           ! Pot.energy/atom, T of the system
224 -0.1789335455E+02  0.1597144817E+01  0.1494689416E+01  2  2  ! id X Y Z type constraint
226 -0.1619578319E+02  0.1830532545E+01  0.3288215770E+01  1  0  ! .
230 -0.1617572658E+02  0.1722213549E+01  0.7033719156E+01  1  0  ! .
     .
3682  0.1612777159E+02 -0.8918761584E+01 -0.1782246887E+02  1  0
0                                                         ! end of file
```

In the above example, h_{11}, h_{22}, and h_{33} are the system dimensions in the *x*-, *y*-, and *z*-directions, given in (Å). The first two lines give the initial system dimensions (not used in ParaGrandMC), while the thrid and forth lines give the current dimensions, which are used at the start of the simulation. N_atoms gives the number of all the atoms in the system. *N_buf*, and *N_free* are not used in ParaGrandMC, but are set equal to *N_atoms* to preserve the format compatibility with other codes. The next three lines are not used in ParaGrandMC.

The ninth line gives the average potential energy per atom of the system, and the system temperature. The potential energy value is used for verification when compared with the calculated energy at the start of the simulation. Deviations, larger than 0.1% from that value are reported as a warning for the user to verify the implemented potential or if there were changes in the file. The temperature value, *T*, given in (K), is the system temperature of the last simulation. If the structure file is a result of an MC simulation, then there are no atomic velocities, and the indicated *T* value is the only way to know the system temperature if needed for further simulations. If the structure file is a result of an MD simulation, then *T* is derived from the average kinetic energy of all the atoms.

Potential file: pot.dat

The representation of the interatomic interactions is defined through a potential file, **pot.dat**. This file gives the number and type of the chemical elements in the structure, the potential functional type number of the interatomic potential implemented, followed by a list of files which define the interatomic potential between these elements.

Example:

An example of a pot.dat file for a NiAl system described through an embedded atom method (EAM) potential as taken from the NIST potential repository [2] is the following:

--- pot.dat ---

```
2 – number of chemical species in the system
'Ni'  58.71     ! chemical symbol and atomic mass
'Al'  26.982
0 – regular EAM alloy potential
'./NiAl-2004/pni.dat'     ! pair Ni-Ni potential
'./NiAl-2004/pnial.dat'   ! pair Ni-Al potential
'./NiAl-2004/pal.dat'     ! pair Al-Al potential
'./NiAl-2004/fni.dat'     ! Ni electron density
'./NiAl-2004/fal.dat'     ! Al electron density
'./NiAl-2004/F_ni.dat'    ! Ni embedded function
'./NiAl-2004/F_al.dat'    ! Al embedded function
```

Output Data

As a result of the simulation, CM_mini1 gives three quantities calculated after each iteration:
"Energy" – total potential energy per atom, given in [eV];
"Force" – total interatomic force per atom, given in [eV/nm];
"Pressure" – total hydrostatic internal pressure of the system, given in [eV/nm^3].

Note that due to the third Newton's law, the total force must always be zero, because there is no external force, applied on the system.

Code Description and Algorithms

Energy calculation

CM_mini1 demonstrates energy calculation of a crystalline atomic structure using the EAM potential functional form expressed as [3]:

$$E = \frac{1}{2}\sum_{i,j(i\neq j)} \Phi_{ij}(r_{ij}) + \sum_i F_i(\bar{\rho}_i) = \sum_i E_i(r_{ij}), \quad (1a)$$

which introduces a potential energy for each atom *i* as:

$$E_i(r_{ij}) = \frac{1}{2}\sum_{j(j\neq i)} \Phi_{ij}(r_{ij}) + F_i(\bar{\rho}_i). \quad (1b)$$

Here, indices *i* and *j* enumerate interacting atoms at a distance r_{ij}, and the superscripts $\alpha, \beta = 1, 2, 3$ refer to the Cartesian coordinates. The first term in Eq. (8) represents pair interactions, $\Phi_{ij}(r_{ij})$ being the pair-interaction potential between atoms *i* and *j*. The term $\mathrm{F}_i(\bar{\rho}_i)$ is the embedding energy of atom *i* as a function of the collective electron density $\bar{\rho}_i$ created at site *i* by all other atoms in the system

$$\bar{\rho}_i = \sum_{j\neq i} \rho_j(r_{ij}), \quad (2)$$

where $\rho_j(r_{ij})$ is the electron density coming from atom *j*, placed at a distance r_{ij} from atom *i*. The three functions, $\Phi(r)$, $F(\rho)$, and $\rho(r)$, are given in a tabulated form.

As seen from Eqs. (1a,b) and Eq. (2), the calculation of the total energy, *E*, involves three steps:

1. Calculating the distances, r_{ij}, between the interacting pair of atoms *i* and *j*.
2. Calculating the collective electron density $\bar{\rho}_i$ created at site *i,* using Eq. (2).

3. Calculating the total energy, E, using Eq. (1a).

Generally, Step 1 requires calculating the distances, r_{ij}, between all of the interacting (i,j) pairs of atoms in the system, or a total of $N(N-1)/2$ pairs, where N is the number of atoms. Because of screening effects between atoms, the interatomic forces in a crystalline solid quickly vanish at a certain distance, r_{cut} , so that in Eq. (1b),

$$E_i(r_{ij} > r_{cut}) = 0. \tag{3}$$

Equation (3) indicates that for each atom i, only neighbors inside a sphere of radius r_{cut} should be considered. To benefit from this cut-off rule, the code in Step 1 must be able to efficiently identify the neighbors, j, inside r_{cut} for each atom i in the system. A fast search for nearest neighbors is performed by using the link-cell algorithm [4]. The link-cell algorithm divides the entire system volume into link-cells (Fig. 1), and each link-cell keeps a list of the atoms that are placed inside it. The link-cells are of close to cubic shape with a side dimension equal to or slightly larger than r_{cut}. The search for a neighbor in the interaction range of an atom does not have to exceed the nearest neighbor link-cells (marked in gray in Fig. 1). In a 3D system, the search range includes a cube of $3^3 = 27$ link-cells, rather than the whole system.

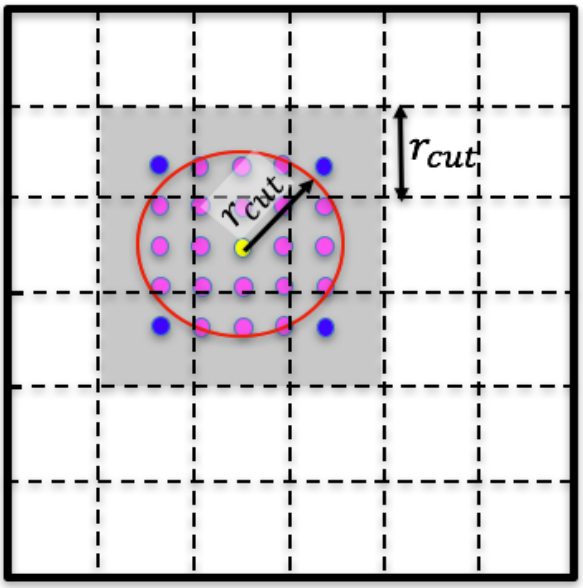

Figure 1. Schematic representation of the link-cell volume decomposition. Bold lines indicate the simulated system box boundaries. Dotted lines indicate the link-cell mesh with the cells in grey indicating the nearest cells, among which a search for neighbors of the central atom (in yellow) is performed.

After the identification of the interacting neighbors, the algorithm proceeds to Steps 2 and 3 to calculate the local electron density, $\bar{\rho}_i$ (Eq. 2), and the total system energy, E (Eq. 1a), respectively. In addition to the energy, the interatomic forces, $f_{i,\alpha}$, are also calculated in Step 3, using the equations:

$$f_{i,\alpha} = \sum_{j \neq i} f_{ij,\alpha}(r_{ij}) , \tag{4a}$$

where

$$f_{ij,\alpha}(r_{ij}) = -\sum_{j\neq i} \frac{\partial E}{\partial r_{ij,\alpha}}, \quad (4b)$$

is the pair force exerted by atom *j* on atom *i*. Here and everywhere below, the Greek symbols, α or β, indicate Cartesian components of vectors and tensors. For a closed system, in the absence of external forces, the third Newton's law postulates that $f_{ij} = -f_{ji}$, so that the total sum of all forces, as reported by CM_mini1 after each iteration, must be zero: $\sum_{i,j\neq i} f_{ij,\alpha} = 0$.

Using the Virial stress equation [5], the calculated pair forces are also used to calculate the components of the internal stress tensor in the system:

$$\sigma_{\alpha\beta} = \frac{1}{N\Omega}\sum_{i,j\neq i} \frac{\partial E}{\partial r_{ij,\alpha}} r_{ij,\beta} = \frac{1}{N\Omega}\sum_{i,j\neq i} f_{ij,\alpha} r_{ij,\beta}, \quad (5)$$

where Ω is the atomic volume.

Block scheme of the algorithm

The block scheme for energy, force and calculations in CM_mini1 is given in Fig. 2. After reading the atomic structure and the tabulated potential in the I/O block, the code builds the link-cell mesh with a list of atoms for each cell. The iteration cycle starts with the subroutine get_neighbors (CM_mini1.f) to identify the neighbors in the interaction range of each atom. The list of the interaction neighbors is used to calculate the electron density at each atom (el_dens_calc_all_EAM in CM_mini1_EAM.f) and after that to calculate the atomic energy, forces, and stresses for each atom in the structure (pot_stress_EAM in CM_mini1_EAM.f). The total energy, force, and pressure (a sum of the stress components) are given as an output after each iteration. The next iteration can be performed either on the same unaltered structure, or after introducing a small random displacement to each atom.

The essential subroutines, which govern the overall efficiency of the computation are get_neighbors, el_dens_calc_all_EAM, and pot_stress_EAM. The goal of CM_mini1 code is to provide a test case for their optimization.

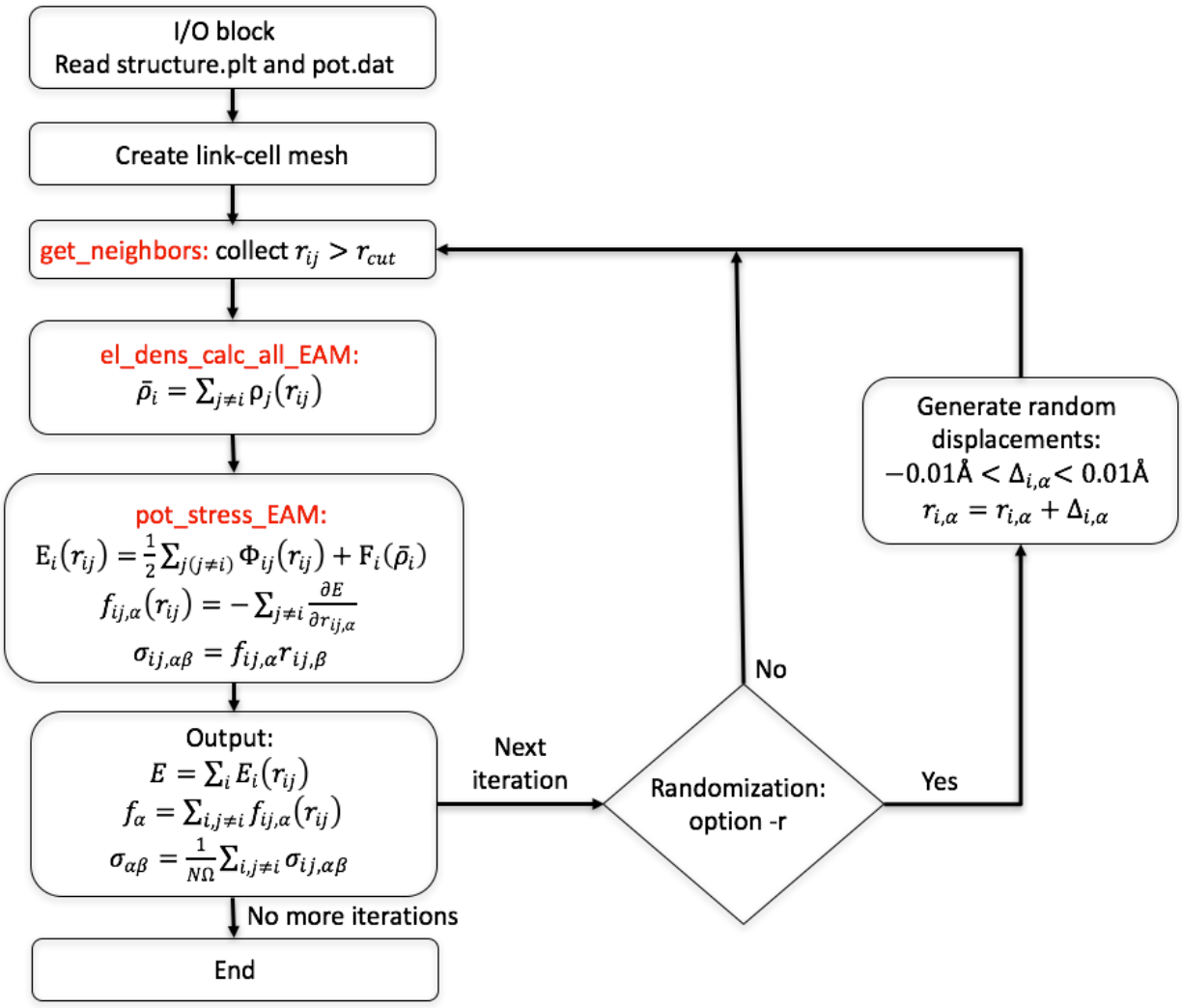

Figure 2. Block scheme for the algorithm of CM_mini1. The essential subroutines that are subject to optimization are given in red.

References

[1] Yamakov, V., "Parallel Grand Canonical Monte Carlo (ParaGrandMC) Simulation Code", NASA/CR–2016-219202; http://www.sti.nasa.gov

[2] NIST Interatomic Potentials Repository: www.ctcms.nist.gov/potentials/

[3] Daw, M.S., Baskes, M.I., "Embedded-atom method: Derivation and application to impurities, surfaces, and other defects in metals," Phys. Rev. B 29, (1984) 6443-6453.

[4] Frenkel, B., Smit, B., Understanding Molecular Simulation (Academic Press, London, 2001).

[5] Cormier J, Rickman JM, Delph TJ (2001) Stress calculation in atomistic simulations of perfect and imperfect solids. J. Appl. Phys. 89:99-104.