NASA/TM–20220003134

Parallel Grand-Canonical Monte Carlo (ParaGrandMC) User's Manual Version 2.0

Vesselin I. Yamakov
National Institute of Aerospace, Hampton, Virginia

Edward H. Glaessgen
Langley Research Center, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

March 2022

NASA/TM–20220003134

Parallel Grand-Canonical Monte Carlo (ParaGrandMC) User's Manual Version 2.0

Vesselin I. Yamakov
National Institute of Aerospace, Hampton, Virginia

Edward H. Glaessgen
Langley Research Center, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

March 2022

Acknowledgements

V. Yamakov is sponsored through cooperative agreement NNL09AA00A with the National Institute of Aerospace. The authors are especially grateful to Yuri Mishin from George Mason University for providing the mathematical algorithm implemented in the code, and for in depth discussions throughout this project.

Available from:

NASA STI Program / Mail Stop 148
NASA Langley Research Center
Hampton, VA 23681-2199
Fax: 757-864-6500

Abstract

This manual describes the commands and command line options for the Parallel Grand Canonical Monte Carlo version 2.0 (ParaGrandMC.2.0) simulation code. This is a highly scalable parallel FORTRAN 2003 code for simulating the thermodynamic evolution of materials at the atomic level, and predicting their thermodynamic state, phase diagram, chemical composition and mechanical properties. The code is specifically designed to simulate multi-component alloy systems, predict solid-state phase transformations such as austenite-martensite transformations, precipitate formation, recrystallization, capillary effects at interfaces, surface absorption, etc., which can aid the design of novel metallic alloys. While the software is mainly tailored for modeling metal alloys, it can also be used for other types of solid-state systems, and to some degree for liquid or gaseous systems, including multiphase systems forming solid-liquid-gas interfaces. In addition to performing Monte Carlo (MC) simulations, the code can also perform Molecular Dynamics (MD) and Langevin Dynamics (LD) simulations, which can be combined and interchanged with MC for faster and more efficient system evolution. A detailed description of the MC part of the code is provided in the NASA ParaGrandMC report: NASA/CR–2016-219202; http://www.sti.nasa.gov.

Content

Introduction

The objective of the Parallel Grand Canonical Monte Carlo (ParaGrandMC) simulation code [1] is to provide a flexible computational tool to model solid-state systems, such as metal alloys, from physics-based principles at the atomic level. This modeling can provide insight to the physical processes that govern material properties of an alloy during the thermal and loading conditions representative of the manufacturing and processing stages, as well as in service. Such understanding improves the acuracy and predictability of the computational models of materials beyond what is achievable through constitutive relations obtained from empirical data. As such, ParaGrandMC helps to guide the design process and to predict the functionality of the alloy in specific applications. The code is designed to study the thermodynamic properties, phase diagram, chemical composition, and mechanical properties of condensed matter systems. The approach taken is based on evolving an initially given atomic system (defined through a list of atomic coordinates of all participating atoms) using the Monte Carlo (MC) combined with molecular dynamics (MD) or Langevin Dynamics (LD) methods. [2]

The MC method is based on repeated random sampling, so called Metropolis sampling [3], of the configuration space of the system using the classical Boltzmann probability distribution. In the process, internal variables of the system, such as system energy, internal pressure, system size (strain), and others, are reported continuously during the simulation. Atomic configurations, in terms of coordinates of all atoms, are stored at predefined time intervals for a post-processing analysis, such as phase identification, lattice parameter estimates, free energy integration, etc. A recently suggested parallelization algorithm [4] based on a specific domain decomposition is used.

The MD method evolves the atomic system by solving the classical Newtonian equations of motion, where the interatomic forces are calculated as the spatial gradient of the interatomic potential identical to the one used to calculate the atomic energies in the MC algorithm. The two methods, MC and MD, can be seamlessly interchanged during the simulation in any ratio of MC or MD steps, which allows for optimal system evolution by overcoming slow diffusion processes through MC steps while still following Newtonian mechanics through MD steps.

The LD method is similar to the MD method with the addition of a stochastic force field to simulate the adequate thermal environment in solvents and viscous systems. Basic explanations of all three methods, MC, MD, and LD, can be found in ref. [2].

The simulation code implements several levels of parallelization. It uses Message Passing Interface (MPI) to work on distributed memory systems built of multiple compute nodes, also called processing elements (PEs). At the compute-node level, where each node can contain one or several multi-core Central Processing Units (CPUs), the code is parallelized over the available CPU cores using Open Multi-Processing (OpenMP) programming interface. In addition, the energy and force calculation subroutines have the capability to use Graphic Processing Units (GPUs) when available, utilizing Open Accelerators (OpenACC) programming interface. These several levels of parallelization allow simulations of multimillion atom systems on high-performance computing platforms.

The User's Manual is structured in the following way. First, the command line options available at start up are listed. These options help to define the overall behavior of the simulation from the beginning. The various types of input and output files are listed next, followed by extensive descriptions of all commands governing the simulation. These commands are applied in a script-like format in file **pgmc.com**. The code does not support an application programming interface (API), rather the simulation is controlled automatically through the commands in the **pgmc.com** file. This allows the simulation to be executed by a queue management system available on modern supercomputer clusters. Additional material is given in several appendices. A detailed description of the newly introduced, in this version of the code, Physically Informed Neural Network (PINN) potential [5,6] is given in Appendix A. PINN is a combination of a neural network and an empirical potential, where the parameters of the empirical potential are predicted by a neural network based on the atomic surrounding, specific for each atom. The empirical potential implemented in PINN is a modified type of a Bond-Order-Potential (BOP) [5,6]. The potential format of BOP, when used separately from the neural network as a standalone potential with fixed parameters is presented in Appendix B. A complete list of all commands is given in tabulated form in Appendix C with an example of an operational pgmc.com file illustrating different simulation regimes is given in Appendix D.

Command Line Options (all are optional)

-s – instructs the code to calculate atomic stress and report it in a file with an extension .stress

-g – uses a global, energy-based stress calculation, instead of virial stress calculation. Energy-based stress is determined through the system energy change at applied virtual strain: ($dE = \sum_{i,j=1..3} \sigma_{ij} d\varepsilon_{ij}$). Default is virial stress, except where the potential derivatives are not defined.

-sg – combined –**s** and –**g** options: calculates atomic stress using the global stress, instead of the virial stress.

Options with MPI parallelization

In ParaGrandMC the spatial decomposition of the system for running on multiple PEs is performed in two dimensions only, which by default are the y-, and z- dimensions, while no spatial decomposition is performed along the x-axis, called the primary axis. When the shortest of the other y- or z- axes is more than 20% shorter than x, then that axis is taken as a primary axis. The 20% value was chosen empirically to avoid frequent reconfigurations of the system when the system size evolves during simulations. This automatic arrangement can be overwritten by the user by specifying the node configuration explicitly using the -p option with X, Y, or Z arguments followed by an even number defining the number of nodes on this direction:

-pX (-pY, -pZ) # – specifies the number (#) of PEs placed along the x- (y- or z-) axis of the simulated system box for the spatial decomposition of the system. (#) must be an even number.

When the requested arrangement cannot be satisfied due to a specific system size, the code automatically choses the most optimal decomposition configuration.

Examples:

```
mpirun –np 8 pgmc                       ! a simple run of the code on 8 PEs
mpirun –np 8 pgmc -pX 2                 ! sets 2 PE rows on x-axis and automatically
                                        ! selects a primary axis between y- or z- axes
mpirun –np 8 pgmc –sg -pX 2  -pZ 4      ! does the following:
```

Calculates atomic stress using global stress calculation (instead of virial stress). Defines a (2x4) PE grid structure along the x- and z- axis in the (x, z) – plane, leaving the y-axis as primary.

Input Files

ParaGrandMC needs the following input files: an input model structure file, **structure.plt**; a file defining the interatomic potential type, **pot.dat**, together with a set of files describing the implemented interatomic potential; and a command file, **pgmc.com**, containing a set of commands to control the simulation. ParaGrandMC can also utilize model structure and potential files in the format defined by the broadly used Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) package [7].

Structure file: structure.plt or structure.lam

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 previously developed codes, such as the Simulator Of Lattice Defects (SOLD), 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 ---

```
1:  # -0.1792446164E+02 -0.1788684653E+02 -0.1782515785E+02            ! -h11/2 -h22/2 -h33/2 initial
2:  #  0.1792446164E+02  0.1788684653E+02  0.1782515785E+02            !  h11/2  h22/2  h33/2 initial
3:  # -0.1792446164E+02 -0.1788684653E+02 -0.1782515785E+02            ! -h11/2 -h22/2 -h33/2 current
4:  #  0.1792446164E+02  0.1788684653E+02  0.1782515785E+02            !  h11/2  h22/2  h33/2 current
5:  #      2    4000    4000    4000                           ! n/a[1]  N_atoms, N_buf, N_free
6:  #    0.67248840E+01       1       1       1                ! n/a
7:  #     -1      -1      -1                                   ! n/a
8:  #      0       0                                           ! n/a
9:  # -0.4430826192E+01     922.6                              ! Pot.energy/atom, T of the system
10: 224 -0.1789335455E+02  0.1597144817E+01  0.1494689416E+01  2  2  ! id X Y Z type constraint
11: 226 -0.1619578319E+02  0.1830532545E+01  0.3288215770E+01  1  0
12: 230 -0.1617572658E+02  0.1722213549E+01  0.7033719156E+01  1  0
     :
   3682  0.1612777159E+02 -0.8918761584E+01 -0.1782246887E+02  1  0
   1                                                           ! 0: no velocities; 1: with velocities
  224  0.1405660198E+01  0.0000000000E+00  0.1132724982E+01 ! id  vx  vy  vz (Å/ps)
  226  0.9420957345E+01 -0.2580232213E+01  0.1855807313E+01
  230 -0.1426233611E+01  0.4245608319E+01  0.1019702603E+01
    :
  3682 -0.1731818246E+01  0.2910471448E+01 -0.7523290612E+00
  1.0 1.0                                                      ! default numbers for 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 third 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 but are reatined for compatibility with the plt format.

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, after subtracting the group velocity of the mass center of the system, $\overline{v_\alpha}$ $(\alpha = x, y, z)$, from the individual atomic velocities of each *i*-th atom, $v_{i,\alpha}$,

$$\frac{3}{2}k_B T = \frac{1}{2N}\sum_{i=1}^{N}\left[m_i\left(v_{i,\alpha}(t) - \overline{v_\alpha}\right)^2\right], \tag{1}$$

where k_B is the Boltzmann constant, and m_i is the mass of the *i*-th atom.

[1] Not used in ParaGrandMC.

The lines after line 9 list all atoms in the system. Each line corresponds to one atom and contains: atom's id-numbers, its x-, y-, z-coordinates in (Å), chemical type number, and a constraint index. The chemical type number must reflect the order of the chemical type as listed in the **pot.dat** file discussed next. After the list of coordinates there is a line with a number of 0 or 1, indicating if this is the end of file or if a list of atomic velocities follows, respectively. When present, the atomic velocities, v_x, v_y, v_z, are expressed in (Å/ps), and are listed after the atom's id-number. The velocity list ends with a line of two numbers, which in the original "plt" format are used for a restart of the simulation. In ParaGrandMC, their values are set to 1.0, 1.0.

When LAMMPS format is used, the input model structure is named "**structure.lam**" and replaces **structure.plt**. The file format follows the description given in the LAMMPS manual [7], where the atomic coordinates are given in Å, and the atomic velocities, when present, are given in (Å/ps). The choice between plt of LAMMPS format is defined in a command script file **pgmc.com** discussed on page 18.

Potential file: pot.dat

The representation of the interatomic interactions is defined through a potential description file, **pot.dat**. This file gives the number and type of the chemical elements in the structure, the potential function type, followed by a list of files which describe the relevant interatomic potential used. There are several potential functional types currently implemented. A list of the valid functional type numbers is given in Table 1.

For some functional types, the interatomic potential can be presented in several formats: either as a set of *.dat or *.plt files as published in the NIST repository for interatomic potentials [8], or as one file in LAMMPS format as described in the LAMMPS manual [7]. The required files needed by the potential are listed in the pot.dat file following the functional type number.

Table 1. Supported interatomic potential types

N: in pot.dat	Potential	Description
0	EAM	Embedded Atom Method [9]
1	ADP	Angular Dependent Potential [10]
2	MEAM	Modified Embedded Atom Method (in progress) [11]
3	Tersoff	Conventional Tersoff in functional form (Tersoff_1) [12]
4	Tersoff- modified	Modified Tersoff [13]
5	EAM/fs	Embedded Atom Method / Finnis-Sinclair [14]
6	BOP	Bond Order Potential [5,6], Appendix A, B
7	LJ	Lennard-Jones potential [15]
100	ANN	Artificial Neural Network potential [16], Appendix A
106	PINN	Physically Informed Neural Network potential with BOP as an empirical potential [5,6], Appendix A

Examples:

An example of a pot.dat file for a NiAl system described through an EAM potential in *.dat format is as follows:

--- 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
```

Similarly, an example of a pot.dat file for a NiAl system described through the same potential in LAMMPS format is as follows:

--- pot.dat ---

```
2 lammps - 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/NiAl2004.eam.alloy'  ! path and file name
```

Output Files

As a result of the simulation, ParaGrandMC produces a number of output files, depending on the applied options, as will be described further in this manual. The default output files that are always present are: an output structure file, which can again be used as an input structure file and has an extension **.plt** or **.lam**, and an output data file (with extension **.dat**). Additional optional output files can be generated if the respective options are activated in the program.

Structure file: filename.#.plt or filename.#.lam

The output structure file has the same format as the input structure file, so that it can be used as an input file for a follow up simulation (after being renamed to **structure.plt** or **structure.lam**). The name of the output structure file is defined in the pgmc.com command file with the added extension “**.plt**” (or “**.lam**” for a file in LAMMPS format). To distinguish between structures produced repeatedly during the simulation, each output structure file has a suffix to its name to indicate the number of the performed Monte Carlo steps (MCS) or molecular dynamics steps (MDS). For example, **filename.00123456.plt** stores the system structure produced after 123456 MCS (or MDS). If the file has extension **.lam** instead of **.plt**, then this describes the same structure, but in LAMMPS format following the LAMMPS manual [7].

Data file: filename.#.dat

The output data file has the name of the structure output file, as defined in the **pgmc.com** file, with the added extension "**.dat**", preceded by the step number corresponding to the initial MCS or MDS in an MC or MD simulation, respectively, set at the beginning of the simulation (e.g., **filename.00123456.dat**). This step number prevents unintentional overwrite of a previous data file during a restart of the simulation from a previous run. The data file stores measurements of various parameters of the system, such as system energy, system dimensions, chemical composition, etc. that are taken periodically during the simulation as instructed in the **pgmc.com** file. The output data file is continuously appended during the simulation. The data are stored in text format in columns. The measured parameter in each column can be specified using the command "measure:" in the pgmc.com file as described in the **Command File** section.

Stress file: filename.#.stress

The output stress file has the name of the structure output file, as given in the **pgmc.com** file, with extension "**.stress**", with the MCS (MDS) number: (#). This file stores the atomic forces, the atomic stress tensors, and the atomic potential energies of each atom. The atomic forces, $f_\alpha^{(i)}$, $(\alpha = x, y, z)$ for atom (i) in the structure are defined as

$$f_\alpha^{(i)} = \sum_{j\neq i}^{N} \frac{\partial U}{\partial \alpha_{ij}}, \tag{2}$$

where $\alpha_{ij} = \alpha_j - \alpha_i$ is the α – component of the relative distance vector between atoms (i) and (j), and $U = \sum_i^N u_i$, is the total potential energy of the system expressed as the sum of the individual atomic potential energies, u_i.

The components of the atomic stress tensor, $\sigma_{\alpha\beta}^{(i)}$, $(\alpha, \beta = x, y, z)$ for atom (i) are defined using the virial stress as: [17]

$$\sigma_{\alpha\beta}^{(i)} = m_i v_\alpha^{(i)} v_\beta^{(i)} - \frac{1}{2}\sum_{j\neq i}^{N} \frac{\partial U}{\partial \alpha_{ij}} \beta_{ij}, \tag{3}$$

The stress file has no header and its format is:

atom id, atom type, $f_x, f_y, f_z,\ \sigma_{xx},\ \sigma_{yy},\ \sigma_{zz},\ \sigma_{xy},\ \sigma_{yx},\ \sigma_{xz},\ \sigma_{zx},\ \sigma_{yz},\ \sigma_{zy}, u_i$,

where the atom id and the atom type correspond to the atom id and type in the **structure.plt** (or **structure.lam**) file. For a centrosymmetric potential, which depends only on the distance from the central atom, but not on the bond angle, the atomic stress tensor is symmetric with $\sigma_{\alpha\beta}^{(i)} = \sigma_{\beta\alpha}^{(i)}$, and only 6 of the components are given, as:

atom id, atom type, $f_x, f_y, f_z,\ \sigma_{xx},\ \sigma_{yy},\ \sigma_{zz},\ \sigma_{xy},\ \sigma_{xz},\ \sigma_{yz},\ u_i$.

The force components are given in (eV/Å), the stress components are given in (GPa), and the potential energy of the atom is given in (eV). The generation of a stress file is optional and is set by an option in the command file **pgmc.com** as described in the **Command File** section.

Visualization file: filename.#.imd

The output visualization file is of the IMD type format used by the open source visualization software OVITO [18]. Its generation and the number and type of the visualized atomistic data is defined through the command **ovito:** in the **pgmc.com** file, as described in the **Command File** section.

Position correlation file: filename.#.rcor

The position correlation file saves the correlation of the *x*-, *y*-, and *z*- coordinates of the atomic positions of the structure, as defined for the *x*- coordinate:

$$\langle x_t x_0 \rangle = \sum_{i=1}^{N} x_i(t) x_i(0), \tag{4a}$$

and likewise for the *y*-, and *z*- coordinates, together with the correlation of the position vector $\vec{\boldsymbol{r}}$:

$$\langle \vec{\boldsymbol{r}}_t \vec{\boldsymbol{r}}_0 \rangle = \langle x_t x_0 \rangle + \langle y_t y_0 \rangle + \langle z_t z_0 \rangle. \tag{4b}$$

The generation of this file is activated through option **rr** to the command **measure:** (i.e., **measure: rr**) in the **pgmc.com** file, as described in the **Command File** section.

Velocity correlation file: filename.#.vcor

The velocity correlation file saves the correlation of the *x*-, *y*-, and *z*- components of the atomic velocities of the structure, as defined for the *x*- component:

$$\langle v_{x,t} v_{x,0} \rangle = \sum_{i=1}^{N} v_{i,x}(t) v_{i,x}(0), \tag{5a}$$

and likewise for the *y*-, and *z*- components, together with the correlation of the entire velocity vector $\vec{\boldsymbol{v}}$:

$$\langle \vec{\boldsymbol{v}}_t \vec{\boldsymbol{v}}_0 \rangle = \left\langle \sum_{i=1}^{N} \vec{\boldsymbol{v}}_i(t) \otimes \vec{\boldsymbol{v}}_i(0) \right\rangle = \langle v_{x,t} v_{x,0} \rangle + \langle v_{y,t} v_{y,0} \rangle + \langle v_{z,t} v_{z,0} \rangle. \tag{5b}$$

The generation of this file is activated through option **vv** to the command **measure:** (i.e., **measure: vv**) in the **pgmc.com** file, as described in the **Command File** section.

Stress (pressure) correlation file: filename.#.pcor

The stress correlation file saves the correlation of the stress tensor components, $\sigma_{\alpha\beta}^{(i)}$, of the of the structure, defined as:

$$\left\langle P_{\alpha\beta}(t)P_{\alpha\beta}(0)\right\rangle = \sum_{i=1}^{N} \sigma_{\alpha\beta}^{(i)}(t)\sigma_{\alpha\beta}^{(i)}(0), \qquad (6)$$

where $\sigma_{\alpha\beta}^{(i)}$ are given by Eq. (3). The generation of this file is activated through option **visc** to the command **measure:** (i.e., **measure: visc**) in the **pgmc.com** file, as described in the **Command File** section.

Heat flux correlation file: filename.#.jcor

The heat flux correlation file saves the correlation $\langle J_\alpha(t)J_\alpha(0)\rangle$ of the heat flux components, J_α, in the system, defined as: [19]

$$\vec{\boldsymbol{J}}(t) = \frac{d\vec{G}(t)}{dt}, \qquad (7)$$

where

$$\vec{G}(t) = \sum_{i=1}^{N}[E_i(t) - \langle E_i\rangle]\vec{\boldsymbol{r}}_i(t), \qquad (8)$$

and

$$\langle E_i\rangle = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{1}{2}m_i v_i^2(t) + u_i(t)\right]. \qquad (9)$$

As a result

$$\vec{\boldsymbol{J}}(t) = \sum_{i=1}^{N}\left[\frac{1}{2}m_i v_i^2(t) + u_i(t)\right]\vec{\boldsymbol{v}}_i(t) - \frac{1}{2}\sum_{i,j\neq i}^{N}\vec{r}_{ij}\left(\vec{\boldsymbol{v}}_i(t)\frac{\partial u_{ij}(t)}{\partial x_{ij}}\right). \qquad (10)$$

The generation of this file is activated through option **cond** to the command **measure:** (i.e., **measure: cond**) in the **pgmc.com** file, as described in the **Command File** section.

Diffusivity file: filename.#.diff, or filename.#.diffE

The velocity correlations from Eq. (5a) and Eq. (5b) are used to calculate the Green-Kubo diffusivity [19]

$$D_{GK} = \frac{1}{3N}\int_0^{\infty}\left\langle \sum_{i=1}^{N} \vec{\boldsymbol{v}}_i(t)\otimes\vec{\boldsymbol{v}}_i(0)\right\rangle dt, \tag{11a}$$

together with the Einstein diffusivity [19], defined through the mean square displacements of the atoms

$$D_E = \lim_{t\to\infty}\frac{1}{2t}\frac{1}{3N}\sum_{i=1}^{N}[\vec{\boldsymbol{r}}_i(t) - \vec{\boldsymbol{r}}_i(0)]^2. \tag{11b}$$

The diffusivity D_{GK} is stored in **filename.#.diff** file, and D_E is stored in **filename.#.diffE** file.

The generation of the *.diff and *.diffE files is activated through options **diff** and **diffE**, respectively to the command **measure:** (i.e., **measure: diff** or **measure: diffE**) in the **pgmc.com** file, as described in the **Command File** section.

Viscosity file: filename.#.visc

The stress correlations from Eq. (6) is used to calculate the Green-Kubo shear viscosity [20] for a system of volume V, and temperature T, as:

$$\eta_{\alpha\beta} = \frac{1}{Vk_BT}\int_0^{\infty}\left\langle P_{\alpha\beta}(t)P_{\alpha\beta}(0)\right\rangle dt; \ (\alpha \neq \beta), \tag{12}$$

which is stored in **filename.#.visc**. When the structure contains multiple elements, like Al and Ni in a Ni-Al alloy, two viscosity files are produced, **filename.#.visc.Al**, and **filename.#.visc.Ni**, for each element, respectively.

The generation of this file is activated through option **visc** to the command **measure:** (i.e., **measure: visc**) in the **pgmc.com** file, as described in the **Command File** section.

Heat conductivity file: filename.#.cond

The heat flux correlations from Eq. (10) is used to calculate the Green-Kubo conductivity [19,20] for a system of volume V, and temperature T, as:

$$\kappa = \frac{1}{Vk_BT^2}\int_0^{\infty}\left\langle \vec{\boldsymbol{J}}(t)\otimes\vec{\boldsymbol{J}}(0)\right\rangle dt\,; \ \vec{\boldsymbol{J}}(t)\otimes\vec{\boldsymbol{J}}(0) = J_x(t)J_x(0) + J_y(t)J_y(0) + J_z(t)J_z(0), \tag{13}$$

which, together with its components,

$$\kappa_\alpha = \frac{1}{Vk_BT^2}\int_0^{\infty}\langle J_\alpha(t)J_\alpha(0)\rangle dt, \tag{14}$$

are stored in **filename.#.cond** file.

The generation of this file is activated through option **cond** to the command **measure:** (**measure: cond)** in the **pgmc.com** file, as described in the **Command File** section.

Command File: pgmc.com

The simulation in ParaGrandMC is controlled by a set of commands with parameters which form a script language to program the simulation execution through the **pgmc.com** file. The commands with their parameters define the input and the output of the simulation, the simulation conditions, such as loading conditions, atomic degrees of freedom, and others in all simulation regimes. The simulation regimes include Monte Carlo (MC), molecular dynamics (MD) and Langevin dynamics (LD). A complete list of commands for all three regimes is given in Tables C1-6 in Appendix C. This section explains their meaning and usage. For each command, an example will be provided first, followed by an explanation and list of the related parameters. Empty lines are allowed for clarity, as well as the insertion of comments through ! or # symbols. Anything after ! or # on the current line is ignored.

File format of the pgmc.com file

An example of the **pgmc.com** file is given below.

Example:

```
--- pgmc.com ---
ini:  2 600.0 0.05 0.0005      ! Initialization with <MC_rank> <temp> <dr> <dh/h>
3                              ! Number of elements
Al                             ! First element (as defined in the pot.dat file)
Co                             ! Second element
Ni                             ! Third element
Filename                       ! Output filename (up to 64 symbols)
input: plt                     ! Input structure file format (plt, or lam for LAMMPS input)
output: lam                    ! Output structure file format (LAMMPS format in this case)
time: 10000                    ! Start time (MCS or MDS)

# MC specific parameters follow:  ! Comment line, not executed
comp: 0.29 0.36 0.35           ! Chemical composition (sum = 1.0)
mu:     0.0  -0.43  0.28        ! Chemical potentials for each element (one must be 0)
alpha: 0.0  0.01  0.01         ! Multiplier coefficients to each mu (ensemble types 3-6)
mc: 1 1000 10 300.0 2 3 0      ! MC run with parameters, which will be described later


# MD specific parameters follow:
md_step: 2.0                   ! MD time step (fs); default value 1.0
diss:  1.0                     ! Heat dissipation coefficient for the Nose-Hoover thermostat
wmass: 16.0                    ! Effective wall mass for Parrinello-Rahmann const. stress
wdamp: 25.0                    ! Effective wall damping for Parrinello-Rahmann const. stress
md: 1 1000 10 300.0 1 3 0      ! MD run with parameters, which will be described later
```

```
end:                          ! end of the simulation (no commands are executed after that)
```

The **pgmc.com** file starts with the initialization command **ini:** with a set of parameters that are general for the entire simulation. Next follow the number and the list of the chemical elements contained in the simulation. Not all of the listed elements may be present in the input structure file, since the chemical composition of the system may change under certain simulation regimes. The listed elements must be a subset of the elements available in the interatomic potential files, as listed in the **pot.dat** file. There may be more elements defined in the **pot.dat** file, which are not listed in the **pgmc.com** file and will not be used in the simulation. The line after the list of elements gives the output file name, common to all of the output files. The **pgmc.com** file must end with the final command **end:** which finalizes the simulation.

The commands between the 'Filename' line and the end: commands are optional for the user to choose how to program the entire simulation. For example, the shortest **pgmc.com** file, which calculates only the potential energy of the initial atomic structure is:

```
ini:  2 600.0 0.05 0.0005     ! Initialization with <MC_rank> <temp> <dr> <dh/h>
1                             ! Number of elements
Al                            ! First element (as defined in the pot.dat file)
'Filename'                    ! Output filename (up to 64 symbols)
md: … Parameters …            ! Simulation mode with parameters described in the next section
mc: … Parameters …            ! Next simulation mode with parameters
…
end:                          ! End of the simulation
```

Depending on the simulation mode, some of the commands may take default values (given in Appendix A) when not explicitly defined.

Another example, showing a variety of modes is given in **Appendix D**.

Initialization commands

```
ini:                          ! initialization line – must be on the first non-commented line
```

Example:

```
ini: 2  600.0  0.05  0.005    ! Initialization of the simulation
```

Parameters: <MC_rank> <temp (K)> <dr (Å)> <dh/h>

MC_rank = 2 or 3 (default 2): size of the partitioned box (in 2x2 cells or 3x3 cells) (see Fig. 2 in ref. [1]). The 3x3 size is recommended for longer range interactions that involve common neighbors between two atoms or 3-body potentials. For EAM and ADP potential the 2x2 size is recommended.

temp – starting temperature of the simulation in (K).
dr – the upper range of a trial displacement move of an atom in (Å).

dh/h - the upper range of a trial system dimensions variations during a constant stress/pressure simulation (dimensionless).

input: ! Input structure file format

Example:
input: plt   ! plt input structure file format
input: lam   ! LAMMPS input structure file format

Parameters:
plt - plt format. Input file name: structure.plt (default, if input: is not defined)
lam – lammps format. Input file name: structure.lam

Default value (when **input:** is not used): plt

output: ! Output structure file format

Example:
output: plt   ! plt output structure file format
output: lam   ! LAMMPS output structure file format

Parameters:
plt - plt format. Output file name: filename.#.plt (default, if input: is not defined)
lam – lammps format. Output file name: filename.#.lam

Default value (when **output:** is not used): plt

time: ! Start time in MCS or MDS (integer number)

Example:
time: 52000
Default value: 0

Note: Useful when the simulation is a continuation from a previous one.

avol: ! Defines atomic volume in [Å^{3}] for stress calculations.

Example:
avol: 16.608
Default value: System volume / number of atoms (suitable for fully dense systems only)

Simulation mode commands

mc: ! Run a Monte Carlo simulation

Evolves the system according to the probability equation:

$$P = \begin{cases} 1 & , \ \Delta\Phi \le 0 \\ exp(-\Delta\Phi/k_B T), & \Delta\Phi > 0 \end{cases}, \tag{15}$$

where P is the acceptance probability of one trial move as defined in ref. [1], and $\Delta\Phi$ is the change in the thermodynamic potential of the system during the trial move.

Parameters:
<Number runs> <Run length> <Measure step> <T> <ensemble> <irigid> <isave_stress>

Number runs - number repetitions of this run.

Run length - number Monte Carlo (MC) steps in one run.

Measure step - Number of steps after which a measurement is performed, and the results are reported as one line in the *.dat file.

T – system temperature (K)

ensemble – Defines the ensemble type as follows (see Table 2):
ensemble = 1 - displacement moves only
2 - displacement moves + $\mu = const.$ (SGMC: requires a defined mu:)
3 - displacement moves + feedback adjusted μ, updated once at every MCS
4 - displacement moves + variance constraint, with μ, updated once at every MCS
5 - same as 3, but μ is updated more frequently inside an MCS
6 - same as 4, but μ is updated more frequently inside an MCS
7 - swap MC move of the chem. type of a pair of atoms. One swap MCS is automatically followed by one MC displacement step

Table 2. Monte Carlo ensemble regimes in constant: Number of particles, Volume, and Temperature (NVT); and in constant: Number of particles, Pressure, and Temperature (NPT) ensembles.

Ensemble	Thermodynamic functional, $\Delta\Phi$	Chemical Composition, c	Chemical Potential, μ	Input Parameters	Output Parameters
1 (NVT)	$\Delta\Phi_1 = \Delta E_p$	Fixed: $c = c_0$	-	$N, V(h_{ij}), T$	-
1 (NPT) Depends on irigid value, see Table 2.	$\Delta\Phi_1 = \Delta E_p - Nk_BTln\frac{V'}{V}$			$N, P(\sigma_{ij}), T$	$V(h_{ij})$
2	$\Delta\Phi_1 + \Delta\mu_{\alpha\beta} + \frac{3}{2}k_BTln\frac{m_\alpha}{m_\beta}$	Varied: $c = c(\mu_0)$	Fixed: $\mu = \mu_0$	μ_0	$c = c(\mu_0)$
3, 5	$\Delta\Phi_1 + \Delta\mu_{\alpha\beta} + \frac{3}{2}k_BTln\frac{m_\alpha}{m_\beta}$	Targeted: $c \to c_0$	Varied through feedback: $d\mu/dt = -\alpha(C - C_0)$ $\mu^{(n)} = \mu^{(n-1)} - \alpha\left(\frac{c^{(n-1)} + c^{(n-2)}}{2} - c_0\right)$	$c_0 = c(t_0)$ $\mu_0 = \mu(t_0)$ α	$c, \mu(\alpha)$
4, 6	$\Delta\Phi_1 + \Delta\mu_{\alpha\beta} + \frac{3}{2}k_BTln\frac{m_\alpha}{m_\beta}$	Varied: $c = c(c_0, \mu_0)$	Variance constraint: $d\mu/dt = -\alpha\, dC/dt$ $\mu^{(n)} = \mu^{(n-1)} - \alpha(c^{(n-1)} - c^{(n-3)})$	$c_0 = c(t_0)$ $\mu_0 = \mu(t_0)$ α	$c, \mu(\alpha)$
7	$\Delta\Phi_1$	Fixed: $c = c_0$	-	As in ensemble 1	As in ensemble 1

irigid – defines system volume adjustments for constant stress simulations (Table 3):

irigid = 1 - constant strain of fixed box dimensions (V=const. ensemble)
2 - constant stress (adjusting all elements of the h_ij matrix)
3 - Andersen constant pressure: (adjusting h11,h22,h33, keeping h12=h13=h23=0)
4 - constant hydrostatic pressure: (ratios h11:h22:h33=const; and keeping h12=h13=h23=0)
5 - constant along x and y (adjusting h_33 only)
6 - constant along x and z (adjusting h_22 only)
7 - constant along y and z (adjusting h_11 only)
8 - constant along x (adjusting h_22 and h_33)
9 - constant along y (adjusting h_11 and h_33)
10 - constant along z (adjusting h_11 and h_22)

Note: Regime irigid=2 automatically uses LAMMPS format for the output structure file, because the plt format does not describe oblique systems.

isave_stress = 0 – no atomic stress files is generated
1 – calculates atomic stress for each atom and generates a filename.#.stress file

Example:

mc: 100 5000 10 600.0 2 3 0 ! Run a semi-grand canonical (μPT)[2] MC simulation

[2] μPT – constant chemical potential, μ, constant Pressure and Temperature ensemble.

md: ! Run a Molecular Dynamics simulation

Integrates the equation of motion:

$$m\ddot{r}(t) = F\big(r(t)\big) - m\gamma\dot{r}(t), \tag{16}$$

where $\dot{r}(t)$ and $\ddot{r}(t)$ are the first and second derivatives of the particle position r with respect to time t. The function F is a deterministic force, defined as the gradient of the interatomic potential, and γ is a friction coefficient imposed on the particle through a thermostat.

Parameters:

<Number runs> <Run length> <Measure step> <T> <ensemble> <irigid> <isave_stress>

All parameters, except <ensemble> have the same meaning as in the **mc:** command.

ensemble = 0 – Constant energy simulation - no thermostat is applied (<T> is ignored).
1 – Constant temperature, <T> with Nose-Hoover thermostat.

Example:

```
md:  100   5000   10   600.0   0   1   0 ! Run a Molecular Dynamics NVE simulation
md:  100   5000   10   600.0   1   3   0 ! Run a Molecular Dynamics NPT simulation
```

ld: ! Run a Langevin Dynamics simulation

Integrates the equation of motion:

$$m\ddot{r}(t) = F\big(r(t)\big) - m\gamma\dot{r}(t) + f_R(t), \tag{17}$$

which is similar to Eq. (16) with an addition of a random force, f_R, defined such that:

$$\langle f_R(t)\rangle = 0; \quad \langle f_R(t)f_R(t')\rangle = 2m\gamma k_B T\delta(t-t'). \tag{18}$$

Parameters:

<Number runs> <Run length> <Measure step> <T> <ensemble> <irigid> <isave_stress>

All parameters, except <ensemble> have the same meaning as in the **mc:** and **md:** commands.

ensemble = 0 – Constant energy simulation - no thermostat is applied (<T> is ignored).
1 – Constant temperature simulation using 2nd order integrator of the Langevin equation (so called, Langevin thermostat):

$$m\ddot{r}(t) = F\big(r(t)\big) - m\gamma\dot{r}(t) + \sqrt{2m\gamma k_B T}\chi(t), \tag{19}$$

where $\chi(t)$ is a Gaussian random numbers of mean = 0, and standard deviation = 1.0.

2 – Viscous dynamics simulation, using overdamped Langevin dynamics defined through the evolution equation:

$$\dot{r}(t) = \frac{F(r(t))}{m\gamma} + \sqrt{\frac{2k_BT}{m\gamma\Delta t}}\chi(t)\,. \tag{20}$$

Example:

```
ld:  100   5000   10   600.0   0   1   0   ! Run a Langevin Dynamics NVE simulation
ld:  100   5000   10   600.0   1   3   0   ! Run an NPT simulation with Langevin thermostat
ld:  100   5000   10   600.0   2   3   0   ! Run a viscous dynamics NPT simulation
```

Table 3. Rigidity constraints

irigid	$\{h_{ij}\}$ - matrix	Description
1	$\{h_{ij}\} = const$	{NVT} ensemble
2	Evolving all $\{h_{ij}\} \neq 0$ elements	Constant stress
3	$\begin{pmatrix} h_{11} & 0 & 0 \\ 0 & h_{22} & 0 \\ 0 & 0 & h_{33} \end{pmatrix}$	Andersen constant pressure
4	$\left.\begin{pmatrix} h_{11} & 0 & 0 \\ 0 & ah_{11} & 0 \\ 0 & 0 & bh_{11} \end{pmatrix}\right\vert_{a,b=const}$	Hydrostatic constant pressure
5	$\begin{pmatrix} h_{11} = const & 0 & 0 \\ 0 & h_{22} = const & 0 \\ 0 & 0 & h_{33} \end{pmatrix}$	Constant pressure along *z*
6	$\begin{pmatrix} h_{11} = const & 0 & 0 \\ 0 & h_{22} & 0 \\ 0 & 0 & h_{33} = const \end{pmatrix}$	Constant pressure along *y*
7	$\begin{pmatrix} h_{11} & 0 & 0 \\ 0 & h_{22} = const & 0 \\ 0 & 0 & h_{33} = const \end{pmatrix}$	Constant pressure along *x*
8	$\begin{pmatrix} h_{11} = const & 0 & 0 \\ 0 & h_{22} & 0 \\ 0 & 0 & h_{33} \end{pmatrix}$	Fixed *x*-dimension
9	$\begin{pmatrix} h_{11} & 0 & 0 \\ 0 & h_{22} = const & 0 \\ 0 & 0 & h_{33} \end{pmatrix}$	Fixed *y*-dimension
10	$\begin{pmatrix} h_{11} & 0 & 0 \\ 0 & h_{22} & 0 \\ 0 & 0 & h_{33} = const \end{pmatrix}$	Fixed *z*-dimension

MC specific initialization commands

(Must be defined before the **mc:** command line)

The use of the MC commands depends on the simulated ensemble regime, as noted to each command (see [1] for the description of the implemented ensemble regimes), otherwise the command is ignored.

comp: ! Defines targeted chemical composition, c_k^0, for each k-th
! element (required for ensembles 3 and 5)

Example:

comp: 0.31 0.35 0.34 ! Defines the targeted chemical compositions in atomic %:
! $c_1^0 = 31\%$ Al, $c_2^0 = 35\%$ Co, and $c_3^0 = 34\%$ Ni.

The sum must be $c_1^0 + c_2^0 + c_3^0 = 1$, otherwise the program exits with an error message.

mu: ! chemical potential, μ (required for ensembles 2-6)

Example:

mu: 0. -0.33 -0.20 ! Defines chemical potential per atom for the listed elements.
! At least one of the values (anyone) must be 0.

alpha: ! Chemical potential factor, α_k (required for ensembles 3-6)

Example:

alpha: 0. 0.01 0.01 ! Sets the constant of proportionality, α_k, to the adjustment
! of the chemical potential for the k-th element.

For ensembles 3 and 5 (feedback adjustment [1]), α_k, is used as:

$$\mu_k^{new} = \mu_k^{old} - \alpha_k\left(c_k - c_k^0\right). \tag{21a}$$

For ensembles 4 and 6 (variance constraint [1]), α_k, is used as:

$$\mu_k^{new} = \mu_k^0 - 2\alpha_k c_k. \tag{21b}$$

In both examples, μ_k^{new}, μ_k^{old}, and μ_k^0 are the new, old, and the initial (at start time) chemical potentials of element k with an adjustment coefficient, α_k. The parameters, c_k is the chemical content of element k averaged between the last two iterations: $c_k = \frac{c_k^{n+1} + c_k^n}{2}$, and c_k^0 is the targeted chemical content of element k.

The input parameters, c_k^0, μ_k^0, and α_k, must be defined prior the start of the ensemble types 2-6, as given in Table 2. If c_k^0 are not defined explicitly at the beginning of the run, they are taken as the current composition of the simulated system. If μ_k^0 are not defined, they are taken from the latest definition, or as the resulted μ from the latest previous run of ensembles 2-6. If those previous runs do not exist and μ_k^0 or α remain undefined, the program exits with an error report.

MD specific initialization commands

(Must be defined before the **md:** command line)

md_step: ! MD time step in femtoseconds (1 fs = 10^{-15} sec).

diss: ! Heat dissipation coefficient in Nose-Hoover thermostat
! (optional: default value: 1.0 ps^{-1}).
wmass: ! Effective wall mass in Parrinello-Rahman dynamics
! (optional: default value: 16.0 atomic units).
wdamp: ! Effective damping coefficient in Parrinello-Rahman
! dynamics (optional: default value: 25.0 ps^{-2}).

integrator: ! Defines which integrator algorithm is to be used.

Example:
integrator: VV

Default value: PC – predictor-corrector.

Parameters:

PC – 5-th order Gear predictor-corrector [21]
VV – 2-nd order Velocity-Verlet [22]

LD specific initialization commands

(Must be defined before the **ld:** command line)

md_step:, diss:, wmass:, and **wdamp:** are the same as for the **MD** regime.

friction: ! Defines a friction coefficient, γ.

External load commands

stress: ! Apply external stress.
Example:
stress: 0.5 0.4 0.3 0.12 0.15 -0.2 ! Apply $\sigma(1,1)$, $\sigma(2,2)$, $\sigma(3,3)$, $\sigma(1,2)$, $\sigma(1,3)$, $\sigma(2,3)$ (GPa).

strain: ! Apply external strain.
Example:
stress: 0.01 0.004 0.01 0.012 0.015 -0.002 ! Apply $\varepsilon(1,1)$, $\varepsilon(2,2)$, $\varepsilon(3,3)$, $\varepsilon(1,2)$, $\varepsilon(1,2)$, $\varepsilon(2,3)$.

Note: stress: and strain: commands have to be consistent with the **irigid** parameter, that defines which components of the h(i,j) matrix can change and in what way (see **Constant stress regimes** and Table 2).

Data and visualization commands

measure: ! Defines simulation quantities to be measured and reported
! in the *.dat file.

Example:
measure: Ti Sij abc comp acc_rate

Default value: The following default quantities are always being reported, even when **measure:** is not used, or used without parameters:

MCS and/or MDS, Total MCS (MDS), E_k, E_p, E_{tot}, T,

where $E_k = 3/2\ k_B T$ is the kinetic energy in (eV/atom), E_p is the potential energy in (eV/atom), E_{tot} is the sum of $E_k +\ E_p$ in (eV/atom), T is temperature in (K).

Additional quantities can be specified as parameters to **measure:** as follows:

Parameters:

hii – prints $h(1,1)$, $h(2,2)$, $h(3,3)$ only (for orthorhombic systems)
hij – prints all $h(i,j)$

Sii – prints diagonal stress components: σ_{xx}, σ_{yy}, σ_{zz} (these are not the principal stresses, only the diagonal components for the given system orientation).

Sij – prints all σ_{ij} components, σ_{xx}, σ_{yy}, σ_{zz}, σ_{xy}, σ_{xz}, σ_{yz} in (GPa).

eii – prints engineering strain diagonal components: ε_{xx}, ε_{yy}, ε_{zz}.
eij – prints all of the engineering strain components, ε_{ij}.
true_eii - prints the true strain diagonal components: $true\_e_{ii} = \int_h^{h'} de_{ii}$.
true_eij - prints all of the true strain components: $true\_e_{ij} = \int_h^{h'} de_{ij}$.

abc – prints current values of the three main axes of the system box, A, B, and C (for an oblique system, they are different from $h(1,1)$, $h(2,2)$, and $h(3,3)$).

angles – prints the angles between the main axes of the system box (useful for an oblique system).

comp – prints the chemical composition in atomic % of all element types in the system.
emax – prints the maximum potential energy of an atom in the system.
det_s – prints the determinant of the stress tensor.

MC specific parameters: (can be used with MD, but they may not have useful meanings)

mu – prints the chemical potential in (eV) of all elements of the system
acc_rate – prints acceptance rate of different MC trial moves in the following order: acceptance rate for the displacement moves, for the element change moves, and for the system volume change moves.

MD specific measure parameters: (can be used with MC, but they may not have useful meanings)

Ti – partial temperature: lists the individual temperatures of each element type in (K).
(Global system temperature, T, is given by default – see default values for measure:)
Tw – Wall temperature, when Parrinello-Rahman dynamics is used.
Q – Heat exchange with the thermostat in eV/atom

ovito: ! Generates IMD type file for OVITO visualization as *.imd file

Example:
ovito: Ep Ek T Sij

Default value: The following quantities are reported when **ovito:** is used without any parameters: atom id, atom type, atom mass (in gmol), and *x*, *y*, and *z* atomic positions in Å.

Additional values for visualization can be specified as follows:

Parameters:

Ep – potential energy (eV/atom)
Ek – kinetic energy (eV/atom) = $\frac{1}{2}mv^2$ (MD) or = $\frac{3}{2}k_B$T (MC)
Et – total mechanical energy = Ep + Ek (eV/atom)
Q – heat dissipation through the thermostat (MD only, otherwise is 0) (eV/atom)
A – total work done on the atom = Ep + Ek + Q (eV/atom)
T – temperature of the atom (K): T = $\frac{2}{3}\mathrm{Ep}/k_B$
V – velocity: (v_x, v_y, v_z) (Å/ps)
Sii - σ(1,1), σ (2,2), σ (3,3) – the diagonal stress components (GPa)
Sij – all stress components (GPa)

histo: ! Calculates a histogram of a certain type[3]

Parameters:
Parameter type – integer number that defines the type of the calculated histogram[3].

Example:

```
histo: 1                    ! Calculates a histogram of type 1 (type definitions are still
                            ! in development)
```

flux: ! Measures[3] the flux of atoms of a given type 1 through a ! membrane of atoms of type 2 of normal [u v w]

Parameters:
atom type 1 – integer number that defines the type of the diffusing atoms
atom type 2 – integer number that defines the type of the membrane atoms
u, v, w – integer numbers that define the membrane normal vector

Example:

```
flux: 2  1   0.0  0.0  1.0  ! Measures the flux of atoms of type 2 through a
                            ! membrane of atoms of type 1 of normal [0 0 1]
```

[3] Still in development – included as a placeholder for future versions.

Constraint commands

high: ! Sets an upper limit of the variation of the chemical
! content of the elements (only applies when MC ensemble
! > 2, ignored otherwise).

Parameters:

< upper limit for element 1> < upper limit for element 2> …

The parameters define the upper limits for the chemical content of each element.

low: ! Sets a lower limit of the variation of the chemical content
! of the elements (MC ensemble > 2, ignored otherwise).

Parameters:

< lower limit for element 1> < lower limit for element 2> …

The parameters define the lower limits for the chemical content of each element.

Example:

```
comp: 0.31   0.35   0.34              ! Defines the targeted chemical compositions:
high:   0.32   0.36   0.35            ! upper limits
low:    0.30   0.34   0.33            ! lower limits
```

Note: The comp: values must be between the respective high: and low: values, otherwise the program exits with an error.

const: ! Defines a constraint type

Parameters:

< constraint type> <chemical element symbol>

constraint type - defines chemical or positional constraint in binary format:

bit = 0 - not constrained; 1 – fixed.

Bit order of the constraints:

bit 4	bit 3	bit 2	bit 1
Chemistry	z-coord.	y-coord.	x-coord.

Example:

```
const:   1011   Ni                ! <constraint type> <element> (all elements if skipped)
                                  ! Fixed chemistry, fixed y- and x- coordinates of all Ni
                                  ! atoms.
const:   0100                     ! Fixed z-coordinate of all atoms.
```

Note: The constraint type is stored in the last 6-th column in the structure.plt file, after being converted to a decimal number, like 1011_2 -> 8 + 2+1 = 11_{10}; $0100_2 = 4_{10}$. There is no equivalent constraint type in LAMMPS format, so the plt format needs to be used when constraints are imposed.

--- structure.plt ---

```
.
 336001   0.62656036E+02   0.62200296E+02   0.64948964E+02   3   11
 336002   0.62345804E+02   0.64233131E+02   0.66714324E+02   2   4
 336003   0.62438297E+02   0.62112418E+02   0.67914654E+02   2   0
.
```

In the above example, atom 336001 will be constrained to preserve its chemical type, and to be fixed in the *x*- and *y*- directions. Atom 336002 will be constrained to be fixed in the *z*-direction, and atom 336003 will have no constraints.

To delete a constraint, one can just overwrite it with another constraint on the same type of atoms, or with a zero (no constraints):

const: 0000 Ni ! Overwriting the previous 1011 Ni constraint.
or
const: 0000 ! All atoms have no constraints.

box: ! Defines a spatial box of active constraints.

Parameters:
<x_min> <x_max> <y_min> <y_max> <z_min> <z_max> <constraint type> <elem>
<x_min>, … <z_max> must be in [0, 1.0] – normalized unit box values.

Examples:
box: 0.0 0.1 0.25 0.5 0.5 1.0 1011 Ni
Meaning: all Ni atoms in a box between (0 < x < 0.1), (0.25 < y < 0.5), and (0.5 < z < 1) have fixed chemistry, and fixed *y*- and *x*- coordinates.

box: 0.0 0.5 0.0 1.0 0.0 1.0 0100
Meaning: all atoms between 0 < x < 0.5 have fixed z-coordinates.

Simulation control commands

loop: ! Creates an execution loop for a set of commands enclosed
! between **loop:** and **end loop:**

Parameters: #   ! Integer number of loops or cycles to be performed

Example:

```
loop: 3                                  ! Repeat 3 times the next lines.
mc:  100 5000 10 600.0   1   3   0       ! Run a MC simulation.
md:   10 5000  10 600.0. 1   1   0.      ! Run a MD simulation.
end: loop                                ! End of loop.
end:                                     ! End of the simulation
```

end: ! End of the simulation (all lines after that are ignored).

Parameters: loop, histo, flux

```
end: loop                                ! End of loop over several commands (see loop:).
end: histo                               ! End of a histogram count (see histo:).
end: flux                                ! End of an atomic flux count through a plane (see flux:).
end:                                     ! end of the simulation (all lines after that are ignored).
```

Example:

```
loop: 3                                  ! Repeat 3 times the next lines.
mc:  100 5000 10 600.0   1   3   0       ! Run a MC simulation.
md:   10 5000  10 600.0. 1   1   0.      ! Run a MD simulation.
end: loop                                ! End of loop.
end:                                     ! End of the simulation
```

Symbols for inserting comments: ! or #. Anything after ! or # is ignored.

Appendix A. Physically informed neural network (PINN) potential file

Description:

The physics and properties of physically informed neural network (PINN) potentials are published by Pun et al. [5,6]. Computationally, PINN uses an artificial neural network (ANN) to predict parameters for a physics-based potential to calculate the energy E_{i_α} of an atom (i_α) of chemical element (α) as a function of its position with respect to other atoms inside a spherical neighborhood of radius R_c. In the current formulation, E_{i_α} is calculated through a modified version of a bond-order potential (BOP) as follows:

$$E_{i_\alpha} = \frac{1}{2} \sum_{j_\beta \neq i_\alpha} \left[e^{\left(A_{i_\alpha}^{\beta} - a_{i_\alpha}^{\beta} r_{i_\alpha}^{j_\beta}\right)} - S_{i_\alpha}^{j_\beta} \Phi_{i_\alpha}^{j_\beta} e^{\left(B_{i_\alpha}^{\beta} - b_{i_\alpha}^{\beta} r_{i_\alpha}^{j_\beta}\right)} \right] f_c\left(r_{i_\alpha}^{j_\beta}\right) + W_{i_\alpha}, \qquad (A1)$$

where

$$S_{i_\alpha}^{j_\beta} = \prod_{k_\gamma \neq i_\alpha, j_\beta} s_{i_\alpha}^{j_\beta k_\gamma}; \quad s_{i_\alpha}^{j_\beta k_\gamma} = 1 - f_c\left(r_{i_\alpha}^{k_\gamma} + r_{j_\beta}^{k_\gamma} - r_{i_\alpha}^{j_\beta}\right) e^{-\lambda_{i_\alpha}^{\beta\gamma}\left(r_{i_\alpha}^{k_\gamma} + r_{j_\beta}^{k_\gamma} - r_{i_\alpha}^{j_\beta}\right)}, \qquad (A2)$$

$$\Phi_{i_\alpha}^{j_\beta} = \left(1 + Z_{i_\alpha}^{j_\beta}\right)^{-1/2}; \quad Z_{i_\alpha}^{j_\beta} = \sum_{k_\gamma \neq i_\alpha, j_\beta} \zeta_{i_\alpha}^{\beta\gamma} S_{i_\alpha}^{k_\gamma} \left(\cos\theta_{i_\alpha}^{j_\beta k_\gamma} - h_{i_\alpha}^{\beta\gamma}\right)^2 f_c\left(r_{i_\alpha}^{k_\gamma}\right), \qquad (A3)$$

$$W_{i_\alpha} = -\sigma_{i_\alpha} {\psi_{i_\alpha}}^{1/2}; \quad \psi_{i_\alpha} = \sum_{j_\beta \neq i_\alpha} S_{i_\alpha}^{j_\beta} \Phi_{i_\alpha}^{j_\beta} f_c\left(r_{i_\alpha}^{j_\beta}\right), \qquad (A4)$$

$$f_c(r) = f_c(r, R_c) = \begin{cases} \dfrac{(R_c - r)^4}{{d_c}^4 + (R_c - r)^4} & : \ R_{min} < r \leq R_c \\ \quad 0 & : \ r > R_c \end{cases}. \qquad (A5)$$

The following nomenclature is used in the above equations: (i) Greek symbols, $\alpha, \beta, \gamma = 1,2, ..n_{el}$, indicate the chemical elements, n_{el} of them in total; (ii) the subscript i_α indicates the atom i_α of element α, whose energy is calculated. This atom is referred to as the *host* atom. The superscripts $j_\beta k_\gamma$ indicate the *neighbor*s of the host atom with their chemical types. The distance between the atoms (i_α) and (j_β) is denoted as $r_{i_\alpha}^{j_\beta}$. $\theta_{i_\alpha}^{j_\beta k_\gamma}$ is the bond angle between the $(i - j)$ and $(i - k)$ bonds of atom (i). When no distinction is made between the *host* and the *neighbor*, and the chemical type is of no consequence, all symbols are used as subscripts, such as r_{ij}.

The BOP parameters $\left(A_{i_\alpha}^{\beta}, a_{i_\alpha}^{\beta}, B_{i_\alpha}^{\beta}, b_{i_\alpha}^{\beta}, h_{i_\alpha}^{\beta\gamma}, \sigma_{i_\alpha}, \zeta_{i_\alpha}^{\beta\gamma}, \lambda_{i_\alpha}^{\beta\gamma}\right)$ in Eqs.(A1-A4) are the sums of the base values and the perturbations (symbol Δ):

$$\left(A_{\alpha}^{\beta} + \Delta A_{i_{\alpha}}^{\beta},\ a_{\alpha}^{\beta} + \Delta a_{i_{\alpha}}^{\beta},\ \dots,\ \lambda_{\alpha}^{\beta\gamma} + \Delta\lambda_{i_{\alpha}}^{\beta\gamma}\right), \tag{A6}$$

where $\left(A_{\alpha}^{\beta}, a_{\alpha}^{\beta}, B_{\alpha}^{\beta}, b_{\alpha}^{\beta}, h_{\alpha}^{\beta\gamma}, \sigma_{\alpha}, \zeta_{\alpha}^{\beta\gamma}, \lambda_{\alpha}^{\beta\gamma}\right)$ is the set of *base* parameters of a globally optimized BOP. These parameters only depend on the chemical type (α, β, γ) but otherwise are the same of all atoms. The values of the base BOP parameters are given in the PINN potential file in the following order:

$$\begin{pmatrix} A_1^1, A_1^2, \dots\ A_1^{n_{el}}, a_1^1, a_1^2, \dots\ a_1^{n_{el}}, B_1^1, B_1^2, \dots\ B_1^{n_{el}}, b_1^1, b_1^2, \dots\ b_1^{n_{el}}, \\ h_1^{11}, h_1^{12}, \dots h_1^{1n_{el}}, h_1^{21}, h_1^{22}, \dots h_1^{2n_{el}}, \dots h_1^{n_{el}1}, h_1^{n_{el}2}, \dots h_1^{n_{el}n_{el}}, \\ \sigma_1, \\ \zeta_1^{11}, \zeta_1^{12}, \dots \zeta_1^{1n_{el}}, \zeta_1^{21}, \zeta_1^{22}, \dots \zeta_1^{2n_{el}}, \dots \zeta_1^{n_{el}1}, \zeta_1^{n_{el}2}, \dots \zeta_1^{n_{el}n_{el}}, \\ \lambda_1^{11}, \lambda_1^{12}, \dots \lambda_1^{1n_{el}}, \lambda_1^{21}, \lambda_1^{22}, \dots \lambda_1^{2n_{el}}, \dots \lambda_1^{n_{el}1}, \lambda_1^{n_{el}2}, \dots \lambda_1^{n_{el}n_{el}} \\ A_2^1, A_2^2, \dots\ A_2^{n_{el}}, a_2^1, a_2^2, \dots\ a_2^{n_{el}}, B_2^1, B_2^2, \dots\ B_2^{n_{el}}, b_2^1, b_2^2, \dots\ b_2^{n_{el}} \\ \vdots \\ \lambda_{n_{el}}^{11}, \lambda_{n_{el}}^{12}, \dots \lambda_{n_{el}}^{1n_{el}}, \lambda_{n_{el}}^{21}, \lambda_{n_{el}}^{22}, \dots \lambda_{n_{el}}^{2n_{el}}, \dots \lambda_{n_{el}}^{n_{el}1}, \lambda_{n_{el}}^{n_{el}2}, \dots \lambda_{n_{el}}^{n_{el}n_{el}} \end{pmatrix}. \qquad (A7)$$

Example of $n_{el} = 1$ (monoatomic PINN):

$$(A_1^1, a_1^1, B_1^1, b_1^1, h_1^{11}, \sigma_1, \zeta_1^{11}, \lambda_1^{11}). \qquad (A8a)$$

Example of $n_{el} = 2$ (binary PINN):

$$\begin{pmatrix} A_1^1, A_1^2, a_1^1, a_1^2, B_1^1, B_1^2, b_1^1, b_1^2, \\ h_1^{11}, h_1^{12} h_1^{21}, h_1^{22}, \sigma_1, \\ \zeta_1^{11}, \zeta_1^{12}, \zeta_1^{21}, \zeta_1^{22}, \\ \lambda_1^{11}, \lambda_1^{12}, \lambda_1^{21}, \lambda_1^{22}, \\ A_2^1, A_2^2, a_2^1, a_2^2, B_2^1, B_2^2, b_2^1, b_2^2, \\ h_2^{11}, h_2^{12} h_2^{21}, h_2^{22}, \sigma_2, \\ \zeta_2^{11}, \zeta_2^{12}, \zeta_2^{21}, \zeta_2^{22}, \\ \lambda_2^{11}, \lambda_2^{12}, \lambda_2^{21}, \lambda_2^{22} \end{pmatrix}. \qquad (A8b)$$

Notice the hierarchical order in Eq.(A8b). First, we list the parameters for the host atom of chemical element $\alpha = 1$, then for $\alpha = 2$, etc. The total number of parameters for the BOP potential becomes

$$N_{BOP} = n_{el}(4n_{el} + 1 + 3{n_{el}}^2). \qquad (A8c)$$

The perturbations $\left(\Delta A_{i_\alpha}^{\beta}, \Delta a_{i_\alpha}^{\beta}, \Delta B_{i_\alpha}^{\beta}, \Delta b_{i_\alpha}^{\beta}, \Delta h_{i_\alpha}^{\beta\gamma}, \Delta\sigma_{i_\alpha}, \Delta\zeta_{i_\alpha}^{\beta\gamma}, \Delta\lambda_{i_\alpha}^{\beta\gamma}\right)$ to the base parameters are predicted by the ANN according to the local atomic environment of the host atom (i_α).

The atomic environment of atom (i) is encoded in a feature vector $\boldsymbol{G_i}$ consisting of a set of local structure parameters (LSPs) $\{G\}_i$. Two kinds of feature vectors are offered in this release of PINN.

- The feature vector of Kind I is defined as $\boldsymbol{G_i^{(1)}} = \left\{G_{sl,\alpha}^{\beta\gamma}\right\}_i$ and depends on the chemical type (α) of the host atom (i).
- The feature vector of Kind II is defined as $\boldsymbol{G_i^{(2)}} = \left\{G_{sl}^{\beta\gamma}\right\}_i$ and does not depend on the chemical type (α) of the host atom (i).

For both kinds, the LSPs are expressed as:

$$G_{sl,\alpha}^{\beta\gamma} = \sinh^{-1}\Gamma_{sl,\alpha}^{\beta\gamma}\,, \qquad (A9a)$$

and

$$G_{sl}^{\beta\gamma} = \sinh^{-1}\Gamma_{sl}^{\beta\gamma}, \qquad (A9b)$$

with

$$\Gamma_{sl,\alpha}^{\beta\gamma} = \Delta + \sum_{j,k\neq i} P_l(\cos\theta_{ijk}) f_s(r_{ij}) f_s(r_{ik}) \delta_{i\alpha}\delta_{j\beta}\delta_{k\gamma}, \qquad (A10a)$$

and

$$\Gamma_{sl}^{\beta\gamma} = \Delta + \sum_{j,k\neq i} P_l(\cos\theta_{ijk}) f_s(r_{ij}) f_s(r_{ik}) \delta_{j\beta}\delta_{k\gamma}. \qquad (A10b)$$

The sum in Eq.(A10a,b) includes $j = k$ terms with $\cos\theta_{ijj} = 1$. Δ is a constant shift parameter, $P_l(x)$ are Legendre polynomials of order l defined by the recursive relations

$$P_{l+1}(x) = [(2l+1)xP_l - lP_{l-1}]/(l+1)\,; \quad P_0(x) = 1; \quad P_1(x) = x. \quad (A11)$$

$f_s(r)$ are Gaussians centered at distances $r_0^{(s)}$ from the host atom:

$$f_s(r) = \frac{1}{r_0^{(s)}} e^{-\left(r-r_0^{(s)}\right)^2/\sigma^2} f_c(r, 1.5R_c). \; (s = 1,2, \dots s_{max}). \qquad (A12)$$

Note that the cutoff function used in this calculation,

$$f_c(r, 1.5R_c) = \begin{cases} \dfrac{(R_c - r)^4}{{d_c}^4 + (R_c - r)^4} : \; r \leq 1.5R_c \\ 0 : \; r > 1.5R_c, \end{cases}$$

has an increased cut-off range compared to Eq.(A5), because the screening atoms in Eq.(A2) extends to $1.5R_c$. Finally, to distinguish between different chemical elements, the symbols $\delta_{i\alpha}$ are introduced:

$$\delta_{i\alpha} = \begin{cases} 1 : if\ atom\ i\ is\ of\ element\ \alpha \\ 0 : \qquad otherwise \end{cases}. \qquad (A13)$$

According to Eqs.(A10a,b), $\Gamma_{sl,\alpha}^{\beta\gamma} = \Gamma_{sl,\alpha}^{\gamma\beta}$ and $\Gamma_{sl}^{\beta\gamma} = \Gamma_{sl}^{\gamma\beta}$. Accordingly, $\mathrm{G}_{sl,\alpha}^{\beta\gamma} = \mathrm{G}_{sl,\alpha}^{\gamma\beta}$ and $\mathrm{G}_{sl}^{\beta\gamma} = \mathrm{G}_{sl}^{\gamma\beta}$.

The arrays $\left\{G_{sl,\alpha}^{\beta\gamma}\right\}_i$ and $\left\{G_{sl}^{\beta\gamma}\right\}_i$ form the feature vectors of Kind I and Kind II, respectively, and are fed as input into the ANN.

The arrangement of the elements in the feature vector follows a hierarchical ordering. First, by the structural indices $(s, l) : \; s = 1,2, \dots\ s_{max}, \; l = l_1, l_2, \dots\ l_{max}$

$$(\boldsymbol{G_{sl}})_\alpha^{\beta\gamma} = \begin{pmatrix} \left\{G_{01,\alpha}^{\beta\gamma}\right\}, \left\{G_{02,\alpha}^{\beta\gamma}\right\}, \dots \left\{G_{0l_{max},\alpha}^{\beta\gamma}\right\}, \\ \left\{G_{11,\alpha}^{\beta\gamma}\right\}, \left\{G_{12,\alpha}^{\beta\gamma}\right\}, \dots \left\{G_{1l_{max},\alpha}^{\beta\gamma}\right\}, \\ \vdots \\ \left\{G_{s_{max}1,\alpha}^{\beta\gamma}\right\}, \left\{G_{s_{max}2,\alpha}^{\beta\gamma}\right\}, \dots \left\{G_{s_{max}l_{max},\alpha}^{\beta\gamma}\right\} \end{pmatrix}_\alpha^{\beta\gamma}, \qquad (A14)$$

and second, by the chemical indices (α, β, γ) for Kind I:

$$[(\boldsymbol{G_{sl}})_1^{11}, (\boldsymbol{G_{sl}})_1^{12}, (\boldsymbol{G_{sl}})_1^{22}, (\boldsymbol{G_{sl}})_2^{11}, (\boldsymbol{G_{sl}})_2^{12}, (\boldsymbol{G_{sl}})_2^{22}\,] \;\; for\ \ \alpha, \beta, \gamma = 1,2, \quad (A15a)$$

and by (β, γ) for Kind II:

$$[(\boldsymbol{G_{sl}})^{11}, (\boldsymbol{G_{sl}})^{12}, (\boldsymbol{G_{sl}})^{22}\,] \;\; for\ \ \beta, \gamma = 1,2. \qquad (A15b)$$

For Kind I descriptors, if the host atom is of chemical sort $\alpha = 1$, then all $(\boldsymbol{G_{sl}})_2^{\beta\gamma} = \sinh^{-1}(\Delta)$ (see Eq.(A10a)); and if $\alpha = 2$, then all $(\boldsymbol{G_{sl}})_1^{\beta\gamma} = \sinh^{-1}(\Delta)$. Since an atom can only be of one chemical type, most of the descriptors are $const = \sinh^{-1}(\Delta)$, which introduces a redundancy. While this redundancy makes the ANN more sensitive in distinguishing different chemical compositions, it also introduces more computational complexity. The Kind II descriptors avoid the redundancy by ignoring the chemical type of the host atom, thus reducing Eq.(A15a) to Eq.(A15b). The chemical identity of the host atom is taken into account at the output of the ANN as described below.

The dataflow through a feed-forward ANN composed of *M* layers can be described by the iteration scheme computing the signal $t_\eta^{(n)}$ at each node η of layer *n* as

$$t_\eta^{(n)} = f_a^{(n)}\left(\sum_{k=1}^{\eta_{max}^{(n-1)}} w_{\eta k}^{(n-1)} t_k^{(n-1)} + b_\eta^{(n)}\right), \quad n = 2,3,\dots M; \;\; \eta = 1,2,..\eta_{max}^{(n)} \qquad (A16)$$

with the initial condition $\left\{t_\eta^{(1)}\right\} \equiv \{G\}_i$, ordered as in Eq.(A15a,b). The activation functions $f_a^{(n)}(x)$ are defined as

$$f_a^{(n)}(x) = \begin{cases} f_a(x) : \; n < M \\ \quad x \;\; : \; n = M \end{cases}. \qquad (A17)$$

Currently, only one type of activation function is implemented:

Type 1: $f_a(x) = \frac{1}{1+e^{-x}} - 0.5 = \frac{1}{2} tanh \frac{x}{2}$.

The coefficients $w_{k\eta}^{(n)}$ and $b_\eta^{(n)}$ appearing in Eq.(A16) are the weights and biases of the ANN, which were optimized during the training process. The ANN output $t_\eta^{(M)}$ contains the perturbations to the BOP parameters for the host atom (i). Their order follows the order of the base parameters given in Eq.(A7):

$$\left(\Delta A_{i_1}^1, \Delta A_{i_1}^2, ..\; \Delta\lambda_{i_{n_{el}}}^{n_{el}n_{el}}\right) = \left(t_1^{(M)}, t_2^{(M)}, \dots t_{last}^{(M)}\;\right)_i. \qquad (A18)$$

According to Eq.(A7), the ANN output consists of sets of perturbation parameters for each possible chemical type of the host atom (i)

$$\left(\underbrace{\Delta A_{i_1}^1, \Delta A_{i_1}^2, ..\; \Delta\lambda_{i_1}^{n_{el}n_{el}}}_{(i)\; of\; element\; 1}, \underbrace{\Delta A_{i_2}^1, \Delta A_{i_2}^2, ..\; \Delta\lambda_{i_2}^{n_{el}n_{el}}}_{(i)\; of\; element\; 2}, \dots \underbrace{\Delta A_{i_{n_{el}}}^1, \Delta A_{i_{n_{el}}}^2, ..\; \Delta\lambda_{i_{n_{el}}}^{n_{el}n_{el}}}_{(i)\; of\; element\; n_{el}}\right). \quad (A19)$$

Since the host atom can only be of one chemical type at a time, only one subset in the output vector (Eq.A19) is used, making the calculations and storage for the entire vector redundant. This redundancy also exists in Kind I descriptors, where the feature vector uniquely identifies the type of the host atom as in Eq.(A15a), and the parameters in the output vector related to the other chemical types are never used. The Kind II descriptors exploit the redundancy in the potential parameters by using the shorter feature vector in Eq.(A15b), which does not indicate the type of the host atom. In this case, the ANN is trained to produce the correct output parameters for all possible types of the host atom given its environment, and only the actual type is used by the BOP. Kind II descriptors are particularly useful in grand canonical or a semi-grand canonical Monte Carlo simulations [1], where a trial move consists of switching the chemical type from one element to another at random without changing its environment. In such a case, there is no need to recompute the feature vector and the ANN: the ANN already contains the parameters for all possible types of the host atom.

Potential file format: filename.pinn

Example for a binary Cu-Ta system:

Comment 1
Comment 2
Comment 3
2 0.1 1 ! format version (this one is 2), parameter Δ in Eq.(A10a,b), type of activation
! function $f_a(x)$ in Eq.(A17)
2 ! number of chemical species in the system, n_{el}.
Cu 63.546000 ! element symbol, atomic mass
Ta 180.947880 ! element symbol, atomic mass
0 0.100000 6.000000 1.500000 1.000000 ! reserved flag, R_{min}, R_c, d_c, σ, see Eqs.(A5, A12)
5 0 1 2 4 6 ! number of Legendre polynomials $l_{max} = 5$, of orders $l = 0,1,2,4,6$
! Eqs.(A10a,b), and Eq.(A11)
8 2.00 2.50 3.00 3.50 4.00 5.00 6.00 7.00
! s_{max} , $r_0^{(1)}, r_0^{(2)}, \dots\ r_0^{(s_{max})}$, see Eq.(A12)
4 240 16 16 42 ! number of ANN layers M, and node number in each layer: $\eta_{max}^{(1)}, \dots\ \eta_{max}^{(M)}$,
! Eq. (A16)

The next one or several lines list the base BOP parameters $\left(A_1^1, A_1^2, ..\ \lambda_{n_{el}}^{n_{el}n_{el}}\right)$ (Eq. A6) arranged as in Eq.(A7) and spanned over one or more lines. Their number, $\eta_{max}^{(M)} = 42$, is given in the preceding line. The base BOP parameters are preceded by a flag number, which defines if these parameters should be used in Eq. (A6) (flag =1) or not (flag = 0). In the later case, the code sets their values to 0.

1 9.4050390000e+00 1.1232620000e+01 4.5610620000e+0, …
8.5633810000e-01 2.3007250000e-01 -1.9998170000e-01…

The remaining lines until the end of the file list the ANN weights and biases, layer by layer in the order below, together with the respective calculation for each layer:

$$\left.\begin{array}{c} w_{11}^{(1)}, w_{21}^{(1)}, \dots\ w_{\eta_{max}^{(2)}1}^{(1)} \\ w_{12}^{(1)}, w_{22}^{(1)}, \dots w_{\eta_{max}^{(2)}2}^{(1)} \\ \dots \\ w_{1,\eta_{max}^{(1)}}^{(1)}, w_{2,\eta_{max}^{(1)}}^{(1)}, \dots w_{\eta_{max}^{(2)}\eta_{max}^{(1)}}^{(1)} \\ b_1^{(2)}, b_2^{(2)}, \dots b_{\eta_{max}^{(2)}}^{(2)} \end{array}\right\} input\ layer\colon\ t_\eta^{(2)} = f_a\left(\sum_{k=1}^{\eta_{max}^{(1)}} w_{\eta k}^{(1)} t_k^{(1)} + b_\eta^{(2)}\right),$$

$$\left.\begin{array}{c} \vdots \\ w_{11}^{(n-1)}, w_{21}^{(n-1)}, \dots\ w_{\eta_{max}^{(n)}1}^{(n-1)} \\ w_{12}^{(n-1)}, w_{22}^{(n-1)}, \dots w_{\eta_{max}^{(n)}2}^{(n-1)} \\ \dots \\ w_{1,\eta_{max}^{(n-1)}}^{(n-1)}, w_{2,\eta_{max}^{(n-1)}}^{(n-1)}, \dots w_{\eta_{max}^{(n)}\eta_{max}^{(n-1)}}^{(n-1)} \\ b_1^{(n)}, b_2^{(n)}, \dots b_{\eta_{max}^{(n)}}^{(n)} \end{array}\right\} hidden\ layer\colon\ t_\eta^{(n)} = f_a\left(\sum_{k=1}^{\eta_{max}^{(n-1)}} w_{\eta k}^{(n-1)} t_k^{(n-1)} + b_\eta^{(n)}\right),$$

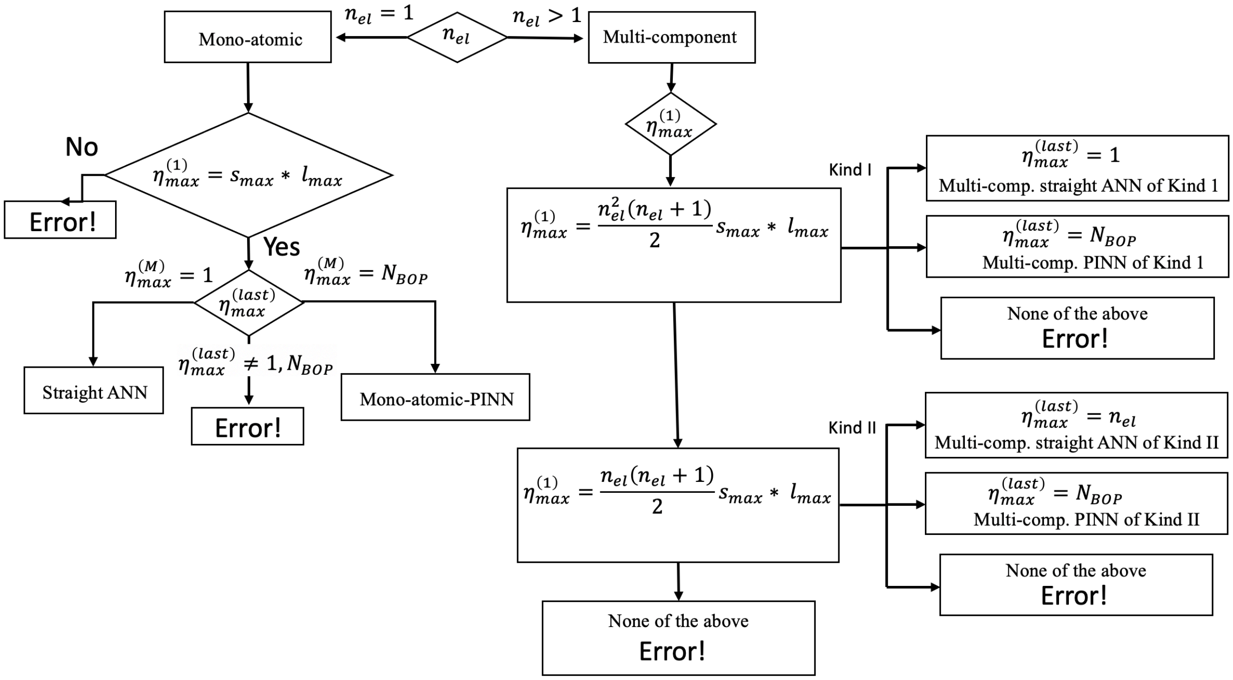
$$\left.\begin{array}{c} \vdots \\ w_{11}^{(M-1)}, w_{21}^{(M-1)}, \dots\ w_{\eta_{max}^{(\ )}1}^{(M-1)} \\ \dots \\ w_{1,\eta_{max}^{(M-1)}}^{(M-1)}, w_{2,\eta_{max}^{(M-1)}}^{(M-1)}, \dots w_{\eta_{max}^{(M)}\eta_{max}^{(M-1)}}^{(M-1)} \\ b_1^{(M)}, b_2^{(M)}, \dots b_{\eta_{max}^{(M)}}^{(M)} \end{array}\right\} output\ layer\colon\ t_\eta^{(M)} = \sum_{k=1}^{\eta_{max}^{(M-1)}} w_{\eta k}^{(M-1)} t_k^{(M-1)} + b_\eta^{(M)}.$$

The PINN file format described above allows for the formulation of several types of PINN potentials.

A. Mono-atomic
   a. Straight ANN (no BOP)
   b. PINN (parameterized BOP)
B. Multicomponent
   a. Straight ANN
      i. Kind I
      ii. Kind II
   b. PINN
      i. Kind I
      ii. Kind II

The numbers n_{el}, N_{BOP}, s_{max}, l_{max}, $\eta_{max}^{(1)}$, and $\eta_{max}^{(M)}$, from Eqs (A8c), (A14), and (A16) are used to uniquely identify the type of the potential according to the following below. Note that the type of descriptors (Kind I or Kind II) is determined automatically according to the value of $\eta_{max}^{(1)}$.

Flowchart for identifying the type of PINN used.


n_{el}
$n_{el} = 1$
$n_{el} > 1$
Mono-atomic
Multi-component
No
$\eta_{max}^{(1)} = s_{max} * l_{max}$
Error!
Yes
$\eta_{max}^{(M)} = 1$
$\eta_{max}^{(last)}$
$\eta_{max}^{(M)} = N_{BOP}$
Straight ANN
$\eta_{max}^{(last)} \neq 1, N_{BOP}$
Error!
Mono-atomic-PINN
$\eta_{max}^{(1)}$
$\eta_{max}^{(1)} = \frac{n_{el}^2(n_{el}+1)}{2} s_{max} * l_{max}$
Kind I
$\eta_{max}^{(last)} = 1$
Multi-comp. straight ANN of Kind 1
$\eta_{max}^{(last)} = N_{BOP}$
Multi-comp. PINN of Kind 1
None of the above
Error!
$\eta_{max}^{(1)} = \frac{n_{el}(n_{el}+1)}{2} s_{max} * l_{max}$
Kind II
$\eta_{max}^{(last)} = n_{el}$
Multi-comp. straight ANN of Kind II
$\eta_{max}^{(last)} = N_{BOP}$
Multi-comp. PINN of Kind II
None of the above
Error!
None of the above
Error!

Appendix B. Bond order potential (BOP) file

The version of the bond order potential (BOP) used in ParaGrandMC is described through Eqs.(A1-A5) in Appendix A, where this potential was used as a part of the PINN potential. The BOP can also be used as a standalone potential with fixed parameters, independent of the atomic positions. In this case the potential file format follows the three-body file format used in LAMMPS [2], as follows:

filename.bop

# Format:

# el1 el2 el3 R_c, d_c, A_{el1}^{el2}, $a_{i_{el1}}^{el2}$, B_{el1}^{el2}, b_{el1}^{el2}, $\zeta_{el1}^{el2,el3}$, $h_{el1}^{el2,el3}$, $\lambda_{i_{el1}}^{el2,el3}$, σ_{el1}, $h_c = d_c$

Cu Cu Cu R_c, d_c, $A_{i_{Cu}}^{Cu}$, $a_{i_{Cu}}^{Cu}$, $B_{i_{Cu}}^{Cu}$, b_{Cu}^{Cu}, $\zeta_{i_{Cu}}^{CuCu}$, h_{Cu}^{CuCu}, $\lambda_{i_{Cu}}^{CuCu}$, σ_{Cu}, $h_c = d_c$

Cu Cu Ta R_c, d_c, $A_{i_{Cu}}^{Cu}$, $a_{i_{Cu}}^{Cu}$, $B_{i_{Cu}}^{Cu}$, b_{Cu}^{Cu}, $\zeta_{i_{Cu}}^{CuTa}$, h_{Cu}^{CuTa}, $\lambda_{i_{Cu}}^{CuTa}$, σ_{Cu}, $h_c = d_c$

Cu Ta Cu R_c, d_c, $A_{i_{Cu}}^{Ta}$, $a_{i_{Cu}}^{Ta}$, $B_{i_{Cu}}^{Ta}$, b_{Cu}^{Ta}, $\zeta_{i_{Cu}}^{TaCu}$, h_{Cu}^{TaCu}, $\lambda_{i_{Cu}}^{TaCu}$, σ_{Cu}, $h_c = d_c$

…

Ta Ta Ta R_c, d_c, $A_{i_{Ta}}^{Ta}$, $a_{i_{Ta}}^{Ta}$, $B_{i_{Ta}}^{Ta}$, b_{Ta}^{Ta}, $\zeta_{i_{Ta}}^{TaTa}$, h_{Ta}^{TaTa}, λ_{Ta}^{TaTa}, σ_{Ta}, $h_c = d_c$

Appendix C. Summary of commands in pgmc.com

Table C1. Initialization and regime defining commands

Command	Parameters	Type and range of parameter values	Optional (default value)	Meaning and notes	Placement or an example
ini:	As listed:			Initialization of the simulation	First line
	MC_rank	2 or 3	No (2)	Size of the MC grid being 2x2 or 3x3	Example: Ini: 2 700.0 0.05 0.0005
	temp	Real > 0.	No	System temperature [K]	
	dr	$0 < dr < 0.1$	No	Max. random atomic displacement range	
	dh/h	$0 < dh/h < 0.001$	No	Max. random fluctuation of the system box dimensions	
input:	File format	plt, lam (lammps)	Yes (plt)	Sets the input structure file format	Anywhere before the first mc:, md:, or ld: commands
output:	File format	plt, lam (lammps)	Yes (plt)	Sets the output structure file format	Anywhere before the first mc:, md:, or ld: commands
time:	Start time	Integer > 0 [MCS]	Yes (0)	Sets the initial start time (if a continued simulation)	Anywhere before the first mc:, md:, or ld: commands
avol:	Atomic volume (Å^3)	Real > 0.	Yes (system volume / number atoms)	Used for not fully dense systems to calculate correctly the atomic stress	Anywhere before the first mc:, md:, or ld: commands
center:	axis axis axis	"x", "y", os "z"	Yes (x y z) – all directions	Fix the mass center in x-, y-, or z- direction or any combination of them	Before mc: md: ld: commands center: (fix in all directions) center: x (fix in x only) center: y z (fix in y and z) … center none: (release the fix)
	none	-	Yes	Releases the mass center fix	
stress:	$\sigma(1,1)$, $\sigma(2,2)$, $\sigma(3,3)$, $\sigma(1,2)$, $\sigma(1,3)$, $\sigma(2,3)$	Real numbers for stress in GPa.	Yes (see text)	Apply external stress	Anywhere before the first mc:, md:, or ld: commands
strain:	$\varepsilon(1,1)$, $\varepsilon(2,2)$, $\varepsilon(3,3)$, $\varepsilon(1,2)$, $\varepsilon(1,3)$, $\varepsilon(2,3)$	Real numbers with absolute value << 1.0	Yes (see text)	Apply external strain	Anywhere before the first mc:, md:, or ld: commands

Table C2. Simulation control commands

Command	Parameters	Type and range of parameter values	Optional (default value)	Meaning and notes	Placement or an example
loop:	loops number	Integer > 0	Yes	Creates an execution loop for a set of commands enclosed between **loop:** and **end loop:**	Example: loop: 3 md: … mc: … end: loop
end:	-	-	No	End of the simulation	The last command, with no executions after that
	loop	-	Yes	End of loop over a set of commands (see loop:)	After the command loop: loop: 3 … end: loop
	histo	-	Yes	End of histogram generation (see histo: in Table C6)	After the command histo: histo: 1 … end: histo (see Table 3.5)
	flux	-	Yes	End of particle flux measure (see flux: in Table C6)	After the command flux: flux: … end: flux (see Table 3.5)

Table C3. Monte-Carlo specific commands

Command	Parameters	Type and range of parameter values	Optional (default value)	Meaning and notes	Placement or an example

comp:	chem_comp(1)
	chem_comp(2)
	…
		0 < chem_comp < 1.0
		Total sum must be 1.0
			Yes (current composition)	Sets a targeted chemical composition for each element	Anywhere before the mc: command that will use it

mu:	chem_pot(1)
	chem_pot(2)
	…
		Real numbers	Ensemble dependent, see Table 1	Sets the chemical potential for each element	Anywhere before the mc: command that will use it

alpha:	chem_fact(1)
	chem_fact(2)
	…
		Real > 0	Ensemble dependent, see Table 1	Proportionality coefficient in the feed-back and variance constraint schemes	Anywhere before the mc: command that will use it

mc:	As listed:			Execution of a simulation run.	Anywhere after ‘filename’ line
	Number runs	Integer > 0	No	Number repetitions of this run	Example: mc: 3 10000 100 700. 2 3 1 Executes 3 runs of 10000 MCS each, taking data at each 100 MCS, at T=700K, in ensemble=2 (SGMC) of constant main stress components, and save atomic stress
	Run length	Integer > 0 (MCS)	No	Number MCS in one run	
	Measure step	Integer > 0 (MCS)	No	A step interval for data report	
	temp	Real > 0 [K]	No	System temperature for this run	
	ienesemble	1 – 7 see Table 1	No	Sets the simulation ensemble	
	irigid	1- 10 see Table 2	No	Sets the rigidity control (sys. dimensions constraints)	

Table C4. Molecular Dynamics and Langevin dynamics specific commands

Command	Parameters	Type and range of parameter values	Optional (default value)	Meaning and notes	Placement or an example
md_step:	MD time step	md_step > 0. in fs: 10^{-15} sec.	Yes (1.0 fs)	Sets the MD time step in fs.	Anywhere before md:
diss:	Heat dissipation	Real > 0	Yes (1.0 ps^{-1})	Heat dissipation coefficient in the Nose-Hoover thermostat.	Anywhere before md:
wmass:	Wall mass	Real > 0	Yes (16.0 atom mass)	Effective wall mass in Parrinello-Rahman dynamics	Anywhere before md:
wdamp:	Wall damping	Real > 0	Yes (25 ps^{-2})	Effective damping coefficient in Parrinello-Rahman dynamics	Anywhere before md:
integrator:	PC	string	Yes (PC)	Applies 5-th order Gear predictor-corrector integrator	Anywhere after the 'filename' line
	VV	string		Applies 2-nd order Velocity-Verlet integrator	
md:	Line param. as listed:			Execution of a molecular dynamics simulation run.	Anywhere after the 'filename' line
	Number runs	Integer > 0	No	Number repetitions of this run	Example: md: 3 10000 100 700. 1 3 1 Executes 3 MD runs of 10000 MD steps each, taking data at each 100 MDS, at T=700K, in (NPT) ensemble save atomic stress md: 3 10000 100 700. 0 1 1 same, but in (NVE) ensemble
	Run length	Integer > 0 (MDS)	No	Number MCS in one run	
	Measure step	Integer > 0 (MDS)	No	A step interval for data report	
	temp	Real > 0 [K]	No	System temperature for this run	
	iensemble	0, 1	No	Thermostat (0 - off; 1 - on)	
	irigid	1-10 see Table 3	No	Sets the rigidity control (sys. dimensions constraints)	
	istress	0, 1	No	0 – no stress file; 1 – stress file to record atomic stress.	
ld:	All line parameters are as in md: except			Execution of Langevin dynamics simulation run	Anywhere after the 'filename' line Example: ld: 3 10000 100 700. 1 3 1
	iensemble	0	No	Constant energy simulation	
		1		Constant temperature simulation using Langevin thermostat	
		2		Viscous, overdamped Langevin dynamics simulation	

Table C5. Constraint commands

Command	Parameters	Type and range of parameter values	Optional (default value)	Meaning and notes	Placement or an example
high:	comp_high(el)	comp_high(el) > chem_comp(el)	Yes (no limit)	Upper limits of the chemical concentration of each element	Anywhere before mc:
low:	comp_low(el)	comp_low(n) < chem_comp(el)	Yes (no limit)	Lower limits of the chemical concentration of each element	Anywhere before mc:
constraint:	As listed:		Yes (no constraints)	Sets constraint type on selected atoms	Anywhere before mc: or md: commands
	Constraint type	Bitwise type like: 0010, 1110, etc.	No	Order of constraints: fixed: chemistry z-pos. y-pos. x-pos. bit 4 bit 3 bit 2 bit 1	
	Element symbol	Chemical symbols: H, Al, ..	Yes (apply to all elements)	The constraint is applied only to the given element	Last number in constraint:
box:	As listed:		Yes (no box)	Defines the part of the system box where a certain constraint is applied	Anywhere before mc: or md: commands
	x-min, x-max	0 ≤ x-min < x-max ≤ 1.0		The constraint box is b/n x-min and x-max	Example 1: box: 0.0 0.5 0.0 1.0 0.0 1.0 1011 Ni Meaning: Ni atoms in the range 0 < x < 0.5 have fixed y and z Example 2: box: 0.0 0.5 0.0 1.0 0.0 1.0 1011 Meaning: All atoms in the range 0 < x < 0.5 have fixed y and z
	y-min, y-max	Same for y-	No	The box is b/n y-min and y-max	
	z-min, z-max	Same for z-	No	The box is b/n z-min and z-max	
	Constraint type	Bitwise type: 0010, 1110, etc.	Yes	Constraint type set to this box	
	Element symbol	Chemical symbols: H, Al, ..	Yes (apply to all elements)	The constraint is applied only to the given element	

Table C6. Measure data commands

Command	Parameters	Meaning and notes	Placement or an example	Result file
measure:	As listed:	Sets which measurements are reported in the *.dat file.	Anywhere before mc: or md: commands	
	hij	Print h(i,j) system matrix	Example: measure: hii tij abc Meaning: Prints h(i,i), σ(i,j), a, b, c	Filename.########.dat
	hii	Print diagonal h(i,i) elements		
	Sij	Print stress σ(i,j) elements		
	Sii	Print diagonal σ(i,i) elements		
	eij	Print engineering strain ε(i,j)		
	eii	Print diagonal ε(i,i) elements		
	true_eij	Print true strain true_ε(i,j)		
	true_eii	Print diagonal true_ε(i,i)		
	abc	Print system box dims. a, b, c		
	angles	Print system box angles.		
	comp	Print chemical comp. in at.%.		
	mu	Print chemical potentials		
	acc_rate	Print MC acceptance rates for displ., chemical change, and volume change trial moves.		
	emax	Print max pot. energy of an atom.		
	det_s	Print det{σ(i,i)}		
	Ti	Print partial T (K) of each element type (MD specific)		
	Tw	Wall temperature (K) in Parrinello-Rahman dynamics		
	Q	Heat exchange (eV/atom) with the thermostat (MD case)		
	Correlation functions and fluxes			
	rr	Calculates position correlations: $\langle x_t x_0 \rangle = \sum_{i=1}^{N} x_i(t) x_i(0)$	measure rr:	Filename.########.rcor
	vv	Calculates velocity correlations: $\langle v_{x,t} v_{x,0} \rangle = \sum_{i=1}^{N} v_{i,x}(t) v_{i,x}(0)$	measure vv:	Filename.########.vcor
	jj	$\langle J_\alpha(t) J_\alpha(0) \rangle$ from Eqs. 7- 10.	measure jj:	Filename.########.jcor
	diff	Green-Kubo diffusion $D_{GK} = \frac{1}{3N} \int_0^\infty \langle \sum_{i=1}^{N} \vec{\boldsymbol{v}}_i(t) \otimes \vec{\boldsymbol{v}}_i(0) \rangle dt$	measure: diff	Filename.########.diff
	diffE	Einstein diffusion $D_E = \lim_{t \to \infty} \frac{1}{2t} \frac{1}{3N} \langle \sum_{i=1}^{N} [\vec{\boldsymbol{r}}_i(t) - \vec{\boldsymbol{r}}_i(0)]$	measure: diffE	Filename.########.diffE
	visc	$\eta_{\alpha\beta} = \frac{1}{V k_B T} \int_0^\infty \langle P_{\alpha\beta}(t) P_{\alpha\beta}(0) \rangle dt;$	measure: visc	Filename.########.visc
	cond	$\kappa_\alpha = \frac{1}{V k_B T^2} \int_0^\infty \langle J_\alpha(t) J_\alpha(0) \rangle dt;$	measure: cond	Filename.########.cond

Table C6. Measure data commands (continue)

Command	Parameters	Meaning and notes	Example	Result file
histo:	histo type	Calculates a histogram of a certain type	Before mc:, md:, ld: histo: 1	
flux:	atom type 1	Element type of the flux atoms	Before mc:, md:, ld: command flux: 2 1 0.0 0.0 1.0 Measure flux of atoms type 2 through a membrane of atoms type 1 of normal [0 0 1]	
	atom type 2	Element type of the membrane atoms		
	x, y, z	Membrane normal vector		

Table C7. OVITO visualization commands

Command	Parameters	Type and units	Optional (default value)	Meaning and notes	Placement or an example
ovito:	As listed:		Yes (see text)	Defines atomic quantities for visualization using OVITO software. Quantities are reported in a file with *.imd extension.	Anywhere before mc:, md: or ld: commands
	Ep	Scalar (eV/atom)	Yes	Potential energy (eV/atom)	Example: ovito: Ep Ek Et T Sij V Meaning: Prints Ep, Ek, Etot = Ep+Ek, T, σ(i,j), and (v_x, v_y, v_z) velocities
	Ek	Scalar (eV/atom)	Yes	Kinetic energy (eV/atom) Ek $= \frac{1}{2}mv^2$ (MD) or Ek $= \frac{3}{2}k_B T$ (MC)	
	Et	Scalar (eV/atom)	Yes	Total mechanical energy Et = Ep + Ek (eV/atom)	
	Q	Scalar (eV/atom)	Yes	Dissipated heat through the thermostat (MD only)	
	A	Scalar (eV/atom)	Yes	Total mechanical work done on the atom $A =$ Ep + Ek + Q	
	T	Scalar (K)	Yes	Temperature T = $\frac{2}{3}$Ek$/k_B$ (K)	
	V	Vector (Å/ps)	Yes	Velocity (v_x, v_y, v_z) (Å /ps)	
	Sii	Tensor (GPa)	Yes	Atomic stress σ(i,i) diagonal elements only	
	Sij	Tensor (GPa)	Yes	Atomic stress σ(i,j) elements	

Appendix D. Some examples of a pgmc.com file

```
ini:  2  800  0.05  0.001               ! MC_rank, T (K), dr (Ang), dh/h
2                                       ! Number of elements
Ni                                      ! First element
Al                                      ! Second element
'TEST'                                  ! Output filename
measure: hii tij comp mu acc_rate       ! Measure hii, stress_ij, composition, mu,
                                        ! acceptance rate.
ovito: Ep Ek Et Q A T V Sij             ! Generates TEST.#.imd file for OVITO visualization
mc:  2  100  20  1000.0  1  3  0        ! 2 MC runs of 100 MCS at T=1000K, report each 20
                                        ! MCS, use (NPT) ensemble, no stress file.
loop: 3                                 ! Start a loop of 3 cycles
mu:  0.0  -0.4300                       ! Set chemical potentials for Ni and Al (one must be zero).
mc:  2  100  20  1000.0  2  3  0        ! 2 MC runs of 100 MCS, report each 20 MCS
                                        ! Use (mu,PT) constant chem. pot. ensemble
                                        ! Composition will change at fixed mu.
end: loop                               ! End of the loop

comp:  0.85  0.15                       ! Set tailored composition at 85% Ni, 15% Al
alpha: 0.0  0.01                        ! Adjustable feedback coefficients for Ni and Al
mc:  2  100  20  1000.0  3  3  0        ! 2 MC runs of 100 MCS, report each 20 MCS,
                                        ! feedback ensemble 3
                                        ! SGMC, no stress file.
mc:  2  100  20  1000.0  7  3  0        ! Use swap MC moves in canonical (NPT).
#
# Start MD simulation from here
#
md_step: 2.0                            ! MD step of 2.0 fs
measure: hii tij Q Ti Tw                ! New measure hii, stress_ij, heat exchange,
                                        ! partial T, T_wall.
                                        ! A new filename.####.dat file will be generated,
                                        ! because the columns have changed, and to help
                                        ! plotting.
md:  2  100  20  1000.0  0  1  1        ! 2 MD runs of 100 MDS, report each 20 MCS, (NVE),
                                        ! generate atomic stress file
md:  2  100  20  1000.0  1  3  1        ! Switch to (NPT) ensemble to maintain T=1000K
diss: 1.5                               ! Set Heat dissipation to 1.5 (from default 1.0)
wmass: 32.                              ! Set the wall mass to 32.0 from the default 16.0
wdamp: 40.                              ! Set the wall damping to 40.0 from default 25.0
md:  2  100  20  1000.0  1  3  1        ! Repeat (NPT) ensemble with the new parameters
# md: 10  10000  100  1000.0  1  3  0   ! Commented a typical long production run
# end:                                  ! Commented end of simulation
# -----------------------------------------------------------------------------------
#  Example using constraints
#
const: 0001 Al                          ! Fix the x-coordinate of all Al atoms
md:  2  100  20  1000.0  1  3  0        ! Run (NPT) ensemble with constraint as follows:
const: 0110                             ! Fixed z- and y-coordinate of all atoms
                                        ! Previous constraints are overwriten (Al are now
                                        ! free on x)
md:  2  100  20  1000.0  1  3  0        ! Run (NPT) ensemble with the new constraint
const: 1000 Ni                          ! Do not allow chemical change of Ni atoms
                                        ! This is valid only for those that are Ni at this
                                        ! time of the simulation.
                                        ! The constraint is not valid for new Ni atoms
mc:  2  100  20  1000.0  2  3  0        ! Runs SGMC with the introduced chemical constraint
const: 0000                             ! Remove all constraints
mc:  2  100  20  1000.0  2  3  0        ! Runs SGMC with no constraints
end:                                    ! End of the simulation
```

```
# ----------------------------------------------------------------------------------
#  Example using Langevin dynamics and histogram
ini:  5  300.0  0.05  0.0002               ! MC_rank, T (K), dr (Ang), dh/h
2                                          ! Number of elements
C                                          ! First element
Li                                         ! Second element
'Li100_h10A'                               ! Output filename
#input: lammps                             ! Reading structure.lam
#output: lammps                            ! Reading structure.lam
measure: Q Ti hii tij Tw
flux: 2 1  0.0 0.0 1.0                     ! flux of type 2 (Li+) through a mem of type 1 (C) of normal (0 0 1)
md_step: 0.25
wmass: 32.0
wdamp: 25.0
diss: 1.0
friction: 220.0 222.0
ovito: Ep Ek Et Q A Ti V fi Sii
#end:
ld:  1   1000 10  10.0  1  10  0           ! irig=10: fixed z-dimension
ld:  1   1000 10  20.0  1  10  0           ! Gradually increasing T
ld:  1   1000 10  40.0  1  10  0           ! using Langevin Thermostat
ld:  1   1000 10  70.0  1  10  0           !
ld:  5   1000 10 100.0  1  10  0           !
ld:  5   1000 10 200.0  1  10  0  !
ld:  5   1000 10 300.0  1  10  0  !
ld:  5   1000 10 330.0  1  10  0  !
histo: 1                                   ! Set histogram type
measure: Q Ti hii tij rr diffE
ld:  100  100000 100 330.0  1  10  0       ! Run to collect histogram
end: histo
end:
```

References

[1] Yamakov, V., "Parallel grand canonical monte carlo (ParaGrandMC) simulation code", *NASA/CR–2016-219202*; http://www.sti.nasa.gov

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

[3] Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E., "Equations of state calculations by fast computing machines," *Journal of Chemical Physics* 21, (1953) 1087.

[4] Plimpton, S., Battaile, C., Chandross, M., Holm, L., Thompson, A., Tikare, V., Wagner, G., Webb, E., Zhou, X., Cardona, CG., Slepoy, A., "Crossing the mesoscale no man's land via parallel kinetic Monte Carlo," SANDIA report: *SAND2009-6226*.

[5] Pun, G.P.P., Batra, R., Ramprasad, R., Mishin, Y., "Physically informed artificial neural networks for atomistic modeling of materials", *Nature Comm.* 10 (2019) 2339.

[6] Pun, G.P.P., Yamakov, V., Hickman, J., Glaessgen, E.H., Mishin, Y., "Development of a general- purpose machine-learning interatomic potential for aluminum by the physically informed neural network method," *Physical Review Materials*, 4 (2020) 113807.

[7] LAMMPS molecular dynamics simulator: http://lammps.sandia.gov/

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

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

[10] Mishin, Y., Mehl, M.J., Papaconstantopoulos, D.A., "Phase stability in the Fe–Ni system: Investigation by first-principles calculations and atomistic simulations", *Acta Materialia* 53 (2005) 4029.

[11] Baskes, I., "Modified embedded-atom potentials for cubic materials and impurities", *Phys. Rev. B.* 46 (1992) 2727.

[12] Tersoff, J., "Modeling solid-state chemistry: Interatomic potentials for mnlticomponent systems", *Phys. Rev. B.* 39 (1989) 5566.

[13] Kumagai, T., Izumi, S., Hara, S., Sakai, S., "Development of bond-order potentials that can reproduce the elastic constants and melting point of silicon for classical molecular dynamics simulation", *Comp. Mayer. Sci.* 39 (2007) 457.

[14] Finnis, M.W., Sinclair, J.E. "A Simple Empirical N-Body Potential for Transition Metals", *Phil Mag A* 50 (1984) 45.

[15] Lennard-Jones, J.E., "On the Determination of Molecular Fields", *Proc. R. Soc. Lond. A*, 106 (1924) 463.

[16] Yamakov, V., Jost, G., Kokron, D., Mishin, Y., Glaessgen, E.H., "High-Performance Computing Optimization for Aladyn – Adaptive Neural Network Molecular Dynamics Mini-Application", *NASA/TM-2019-220409*; http://www.sti.nasa.gov

[17] Cormier, J., Rickman, J. M., Delph, T. J., "Stress Calculation in Atomistic Simulations of Perfect and Imperfect Solids", *J. Appl. Phys.* 89 (2001) 99.

[18] Stukowski, A., "Visualization and analysis of atomistic simulation data with OVITO-the Open Visualization Tool," *Modell. Simul. Mater. Sci. Eng.* 18 (2010) 015012.

[19] Kinaci, A. Haskins, J.B., Çagin, T., "On calculation of thermal conductivity from Einstein relation in equilibrium molecular dynamics", *J. Chem. Phys.* 137 (2012) 014106.

[20] Mondello, M., Grest, G.S., “Viscosity calculations of nalkanes by equilibrium molecular dynamics”, *J. Chem. Phys.* 106 (1997) 9327.
[21] Gear, C. W., “The Numerical Integration of Ordinary Differential Equations of Various Orders”, Technical Report ANL 7126 (1966) Argonne National Laboratory, Argonne, IL.
[22] Verlet, L., "Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules", *Physical Review* 159 (1967) 98.