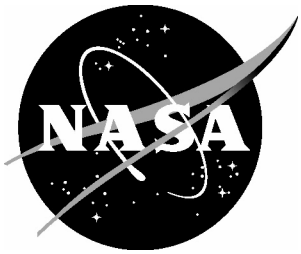
NASA/TM–20260000380



Parallel Grand-Canonical Monte Carlo (ParaGrandMC) User's Manual Version 3.0

Vesselin I. Yamakov
Analytical Mechanics Associates, Hampton, Virginia

Edward H. Glaessgen
Langley Research Center, Hampton, Virginia

January 2026

NASA STI Program Report Series

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NTRS Registered and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

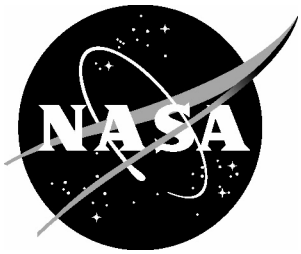
For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at http://www.sti.nasa.gov

- Help desk contact information:

https://www.sti.nasa.gov/sti-contact-form/
and select the "General" help request type.

NASA/TM–20260000380



Parallel Grand-Canonical Monte Carlo (ParaGrandMC) User's Manual Version 3.0

Vesselin I. Yamakov
Analytical Mechanics Associates, Hampton, Virginia

Edward H. Glaessgen
Langley Research Center, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

January 2026

Available from:

NASA STI Program / Mail Stop 050
NASA Langley Research Center
Hampton, VA 23681-2199

Abstract

This manual describes the commands and command line options for the Parallel Grand Canonical Monte Carlo version 3.0 (ParaGrandMC.3.0) simulation code, available upon request at https://software.nasa.gov/software/LAR-20457-1. 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 accuracy 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 Monte Carlo (MC) simulations, 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 shape change (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 parallelization algorithm based on a specific domain decomposition [4] adapted for Metropolis Monte Carlo 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 appropriate thermal environment in solvents and viscous systems. Basic explanations of all three methods, MC, MD, and LD, can be found in ref. [2].

Version 3.0 of ParaGrandMC provides enhanced capabilities for exploring phase coexistence and finding equilibrium lines in the phase diagram of pure metals and alloys. For pure metal systems, an upgrade to the MD method includes constant enthalpy ensemble regime [5], which finds the temperature of equilibrium in a two-phase system, such as the melting temperature of a solid/liquid interface, or the phase transition temperature in a solid state transformation. For alloy systems, the multi-cell Monte Carlo (MC2) algorithm [6] is implemented to search for phase coexistence in a multi-component system under isobaric-isothermal conditions. ParaGrandMC.3.0 also includes the nudged elastic band (NEB) method [7] to explore transition states and to find transition activation energies.

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, compiling and installation instructions are presented. Next, the command line options available at startup are listed. These options help to define the overall behavior of the simulation. 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 text format in file **pgmc.com**. The code does not support a user interface, 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 a novel Physics-Informed Neural Network (PINN) interatomic potential [8,9] available in ParaGrandMC 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) [8,9]. The potential format of BOP, when used separately from the neural network as a standalone potential with fixed parameters, is presented in Appendix B. Appendix C describes in detail the Multi-Cell Monte Calro (MC2) algorithm as implemented in ParaGrandMC following [6]. Appendix D describes the implementation of the Nudged Elastic Band (NEB) method following [7]. A complete list of all commands is given in tabulated form in Appendix E.

Compiling and Installation

ParaGrandMC comes with source code files with extensions .f or .inc written in FORTRAN 2003. The source files are placed in the SOURCE directory. The code is compiled using the command "make" executed from the command line in the SOURCE directory. The make command reads a default **makefile** which contains all of the compilation instructions and produces an executable named "pgmc". The executable is a standalone file and can be moved or copied anywhere in the appropriate place chosen by the user. The makefile can be modified for a specific FORTRAN compiler, such as INTEL, PGI, or gfortran compilers. The makefile also includes optimization options for best performance on a given platform. Depending on the specific compiler and hardware used, these options may need to be modified by the user. Makefiles for several platforms, which have been used successfully are given in the distribution, such as makefile.intel (for INTEL

compiler), makefile.intel.skylake (for INTEL compiler on skylake cpus), makefile.gfort (for gfortran compiler), or makefile.V100 for running on a NVIDIA Tesla* V100 GPU.

Compiling options with examples:

```
make                              ! compilation using the default "makefile" and -O3
                                  ! optimization, if available.
make -f makefile.intel.skylake    ! compiles using makefile.intel.skylake file as a makefile
make OMP=TRUE                     ! Compiles with active OpenMP instructions for multithread
                                  ! parallelization.
make ACC=TRUE                     ! Compiles with active ACC instructions for GPU nodes.
make DEBUG=TRUE                   ! Compiles with a set of compiler debugging options.
make REPORT=TRUE                  ! For INTEL compiler to generate optimization reports in
                                  ! files with extension *.optrpt.
make MPI=FALSE                    ! Compiles without MPI instructions. Some regimes
                                  ! requiring multimode execution may not work.
make clean                        ! Removes *.o and *.mod files left during compilation.
```

where the symbols "!" or "#" are used for inserting comments. Anything after "!" or "#" is ignored.

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.

-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.

* Specific vendor and manufacturer names are explicitly mentioned only to accurately describe the hardware used in this study. The use of vendor and manufacturer names does not imply an endorsement by the authors or the U.S. Government nor does it imply that the specified equipment is the best available.

-pY – Specifies the number of PEs placed along the y-axis of the simulated system box for the spatial decomposition of the system. Must be an even number.

-pZ – Specifies the number of PEs placed along the z-axis of the simulated system box for the spatial decomposition of the system. Must be an even number.

When -pY and -pZ are not set or cannot be satisfied due to a specific system size, the code automatically chooses the most optimal decomposition configuration, which minimizes the area-to-volume ratio of the decomposed regions to minimize inter-PEs communications.

-tX (-tY, -tZ) – Fixes a primary axis along which no spatial decomposition is applied. The 2D spatial decomposition takes place on the other two axes. Default is the x-axis, unless the shortest of the y- and z-axis is more than 20% shorter than the x-axis, in which case the shortest 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. When **–t** option is used, the primary axis is fixed throughout the entire simulation, otherwise it may change if the system volume is deformed during the simulation.

Examples:

```
mpirun –np 8 pgmc                          ! a simple run of the code on 8 PEs
mpirun –np 8 pgmc -sg -pX 2 -pZ 4 -tY      ! Sets Y as a primary axis, using (2x4) spatial
                                           ! decomposition for the PEs on the (x, z) plane.
```

The second example sets the code to calculate atomic stress using global stress calculation (instead of virial stress) and defines a (2x4) PE grid structure with a primary axis set to Y, so that the spatial decomposition is performed on the (x, z) plane.

Input Files

ParaGrandMC needs the following input files: an input atomic structure file, **structure.plt**; a file defining the interatomic potential type, **pot.dat**; a set of files describing the implemented interatomic potential and listed in **pot.dat**; 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 [10] and Vienna Ab initio Simulation Package (VASP) [11].

Structure file types: "structure" with extentions .plt, .lam, .dump, .poscar, .imd,

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 ---

```
# -0.1792446164E+02 -0.1788684653E+02 -0.1782515785E+02  ! -h11/2 -h22/2 -h33/2 initial
#  0.1792446164E+02  0.1788684653E+02  0.1782515785E+02  !  h11/2  h22/2  h33/2 initial
# -0.1792446164E+02 -0.1788684653E+02 -0.1782515785E+02  ! -h11/2 -h22/2 -h33/2 current
#  0.1792446164E+02  0.1788684653E+02  0.1782515785E+02  !  h11/2  h22/2  h33/2 current
#       2    4000    4000    4000                  ! n/a  N_atoms, N_buf, N_free
#    0.67248840E+01      1       1       1            ! n/a
#      -1      -1      -1                      ! n/a
#       0       0                              ! n/a
# -0.4430826192E+01     922.6                      ! Pot.energy/atom, T of the system
224 -0.1789335455E+02  0.1597144817E+01  0.1494689416E+01  2  2  ! id X Y Z type constraint
226 -0.1619578319E+02  0.1830532545E+01  0.3288215770E+01  1  0  ! .
230 -0.1617572658E+02  0.1722213549E+01  0.7033719156E+01  1  0  ! .
    :
3682  0.1612777159E+02 -0.8918761584E+01 -0.1782246887E+02  1  0
1                                                                    ! 0: no velocities; 1: with velocities
224  0.1405660198E+01  0.0000000000E+00  0.1132724982E+01 ! id  v_x   v_y   v_z (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, while the third and the fourth 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 retained for compatibility with the plt format.

The next 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 the calculated energy at the

start of the simulation are reported as a warning for the user to verify the implemented potential or the correctness of 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, $v_{i,\alpha}$,

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

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

The list of coordinates ends with 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 default values of 1.0, 1.0 are used.

When LAMMPS format is used, the input model structure is named "**structure.lam**" or "**structure.dump**" and replaces **structure.plt**. The file format follows the description given in the LAMMPS manual [10], where the atomic coordinates are given in Å, and the atomic velocities, when present, are given in (Å/ps).

VASP structure files are supported through the implementation of the poscar file format [11] in which case the input structure file is "**structure.poscar**".

A file compatible with the open source visualization tool software OVITO [12] can also be used as an input structure file named as "**structure.imd**". (see **Visualization file: filename.#.imd**, and command **ovito:** in the **pgmc.com** file, as described in the **Command File** section.)

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, and a list of files which describe the applied interatomic potential. There are several potential functional types currently implemented. A list of the valid functional type numbers is given in Table E1 in Appendix E.

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 [13], or

as one file in LAMMPS format as described in the LAMMPS manual [10]. The required files describing the interatomic potential are listed in the pot.dat file following the functional type number.

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, the number of output files produced by ParaGrandMC depends on the selected options, as will be described further in this manual. The default output files that are always present are: an output structure file (having an extension **.plt**, **.lam,** or **.poscar**) and an output data file (having an extension **.dat**).

Structure file types: filename.#.plt, filename.#.lam, filename.#.poscar

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 and "**.poscar**" for a file in VASP formats). 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 [10].

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**", and with the MCS (MDS) number (e.g., **filename.00123456.dat**). 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 requested in the **pgmc.com** file. 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. The output data file is continuously appended line by line during the simulation, adding a new line at each measure step as set in the **pgmc.com** file (see **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 $(\alpha = x, y, z)$ 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: [14]

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

The stress file has no header and its format is:

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

236 1 .. ..

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, $f_x, f_y, f_z,\ \sigma_{xx},\ \sigma_{yy},\ \sigma_{zz},\ \sigma_{xy},\ \sigma_{xz},\ \sigma_{yz},\ u_i$.
236 1 .. ..

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 [12]. 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.

Note: Output files with extensions **.plt**, **.lam**, and **.imd** can also be used as input files, after renaming them to **structure.plt**, **structure.lam**, and **structure.imd**, respectively.

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:** (**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:** (**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:

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

where $\sigma_{\alpha\beta}^{(i)}$ are given by Eq. (3). The generation of this file is activated through option **visc** to the command **measure:** (**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: [15]

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

where

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

and

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

As a result,

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

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.

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

The velocity correlations from Eq. (5a) and Eq. (5b) are used to calculate the Green-Kubo diffusivity [16]

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

together with the Einstein diffusivity [15], defined through the mean square displacements of the atoms

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

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:** (**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 [16] for a system of volume V and temperature T, as:

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

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**, one for each element.

The generation of viscosity files is activated through option **visc** to the command **measure:** (**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) are used to calculate the Green-Kubo conductivity [15,16] for a system of volume V and temperature T, as:

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

which, together with its components,

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

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 molecular dynamics (MD) and Langevin dynamics (LD), Monte Carlo (MC), multi-cell Monte Carlo (MC2), energy minimization, and the nudged elastic band (NEB) method. Complete lists of commands for all regimes are given in Tables E2–E9 in Appendix E. 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.

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 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
```

```
# MD 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

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

Parameters indicated in < > are inserted as single numbers of integer or real value, or a string. Their meaning will be explained in the description of the command they are applied to.

Format of Command Description

The description of commands used in the **pgmc.com** file will be given in the following format:

Command name: (optional) – the command syntax (in bold) as it should be used in the **pgmc.com** file, followed by an indication if the parameter is optional and can be omitted.

A brief description of the purpose of the command and how it works may be given after the command name. A detailed description of the operational algorithm for some of the more complex commands is given in the appendices.

Parameters: <parameter 1> <parameter 2> … – list of the parameters that must be provided with the command. Parameters could be numbers or strings of characters.

<parameter 1> – short explanation of parameter 1.

Range: (integer, real, or {interval, e.g., 0 < parameter < 1.0})

or

Values: if a discrete set of values are used, followed by the description of each value.

List of values with explanation.

Default value: – gives the default value if the parameter is optional.

<parameter 2>

…

Example: – gives a working example of the command with valid parameter values as used in the **pgmc.com** file.

OPERATIONAL COMMANDS

Initialization (Table E2)

ini: ! The first line in the pgmc.com file to initialize the simulation

The **pgmc.com** file starts with the initialization command **ini:** with a set of parameters that are general for the entire simulation.

Parameters: <MC_rank> <T> <dr (nm)> <dh/h> ! Monte Carlo related parameters

<MC_rank> – minimum size of the grid-cell partitioned box in which MC-particles are tried in parallel as explained in [1,16,17] (see Fig. 1).

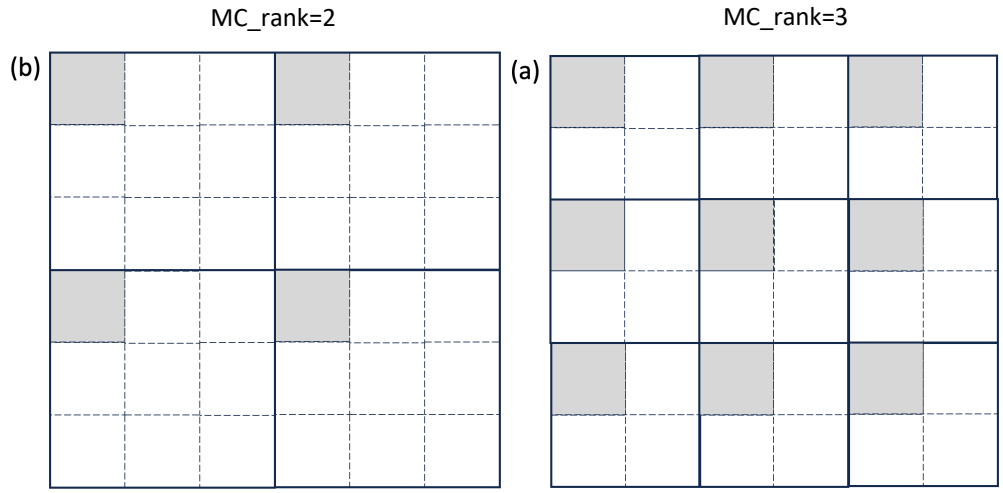


Figure 1. 2d domains partitioned in boxes delineated by bold lines in which individual cells of the size of the cut-off range of the acting interatomic potential are shown by dashed lines. At each MC trial attempt, one particle of each gray cell is tried simultaneously, and the grid of gray (active) cells is moved one position left, right, up, or down at random, and one particle of each of the new gray cells is tried again simultaneously. The size of the grid is defined by MC_rank as (a) MC_rank = 2, and (b) MC_rank = 3.

Values: Integer: ≥ 2

<T> – starting temperature of the simulation in (K)
Range: real, positive
<dr> – upper range of a random trial displacement move of an atom in (Å)
Range: real, positive
<dh/h> – upper range of a trial system dimensions variations during a constant stress/pressure simulation (dimensionless).
Range: real, positive

Example:

```
ini: 2  600.0  0.05  0.005      ! Initialization of the simulation – must be the first command.
                                ! non-commented line in the pgmc.com file.
```

After the **ini:** command follows 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, as listed in the **pot.dat** file. There may be more elements defined in the interatomic potential (or in the pot.dat file), which may not be listed in the **pgmc.com** file if they are not used in the simulation.

Example:

```
3                               ! Number of elements
Al                              ! First element (as defined in the pot.dat file)
Co                              ! Second element
Ni                              ! Third element
```

The line after the list of elements gives the output file name set as a string in single quotation marks **'filename'**. This filename is common to all of the output files.

time: (optional) ! Start time in MCS or MDS

Parameters: <start time> – starting time of the simulation in MD or MC steps.

Range: integer, positive

Default value: 0

Example:

time: 52000

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

avol: (optional) ! Defines atomic volume in [Å^3] for stress calculations.

Parameter: <volume> – atomic volume in [Å^3]

Range: real, positive

Default value: System volume / number of atoms (assumes a fully dense system)

Example:

avol: 16.608

center: (optional) ! Simulation under a fixed mass center of the system

The mass center coordinates of the entire system are subtracted from the current coordinates of each atom during simulation, so that the mass center remains fixed at its initial position.

Parameters: <coordinate 1> <coordinate 2> <coordinate 3>

Values: $\{x, y, z\}$ – sets the direction(s) along which the mass center will be fixed.

no – unfixes the mass center

Default value: no

Example:

center: x y z ! The mass center becomes fixed in all 3 dimensions.
center: no ! Releases the mass center.

Input/Output commands

input: (optional) ! Input structure file format
Parameter: <format> – format of the input structure file
Values:{plt, lam, imd}
plt – File format plt: reads structure.plt
lam – LAMMPS file format: reads structure.lam
imd – OVITO file format: reads structure.imd
Default value: plt
Example:
input: lam ! LAMMPS file format: reads structure.lam

output: (optional) ! Output structure file format
Parameter: <format> – format of the output structure file
Values:{plt, lam}
plt – File format plt
lam – LAMMPS file format
Default value: plt
Example:
output: lam ! LAMMPS file format: filename.#.lam

Operational commands

loop: (optional) ! Performs a repeated loop over the commands placed
… ! between **loop:** and **end: loop** commands.
end: loop

Parameter: <number of loops> – sets the number of loops to be performed.
Range: integer, positive
Default value: single execution (no loops)
Example:

loop: 3 ! Sets a loop of three cycles.
md: 1 20000 100 100.0 1 3 0 ! Performs an MD simulation (see **md:** command).
end: loop ! End of the loop series.

Finalization commands

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

Parameters: (optional) <loop> ! End of a loop, if one has been started.
Example:
end: loop ! End of loop over several commands (see loop:).
end: ! end of the simulation (all lines after that are ignored).

SIMULATION MODES

Molecular Dynamics

md: ! Performs 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> <rigidity> <save_stress>

<Number runs> – number repetitions of this run.
Range: integer, positive

<Run length> – number of MD integration time steps in one run.
Range: integer, positive

<Measure step> – Number of steps after which a measurement is performed, and the results are reported in a single line in the *.dat file.
Range: integer, positive

<T> – system temperature (K)
Range: real, positive
If <T> is set to 0, the current temperature of the system is assumed to continue. The current temperature is either the temperature set from the preceding simulation or the one calculated from the atomic velocities of the input structure, if available. If none of those temperatures are available, <T> from the **ini:** command is used.

<ensemble> – Defines the ensemble type.
Values: 0 - NVE ensemble, constant energy simulation (T is ignored)

1 - NVT (irigid=1, constant volume) or NPT (irigid > 1, constant pressure) ensembles using Nose-Andersen dynamics [19]
2 - NVT (irigid=1) or NPT (irigid > 1) ensembles with Velocity-Verlet integrator as described by Martyna et al. [5]
3 - NPH (irigid > 1) – constant enthalpy ensemble with Velocity-Verlet integrator as described by Martyna et al. [5], same as 2, but without the thermostat (T is ignored)

<rigidity> – Defines system volume adjustments for constant-stress simulations (Table E3).
Values: 1 - Constant strain of fixed-box dimensions (V=const. ensemble)
2 - Constant stress (adjusting all elements of the h_ij matrix, oblique system)
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.

<save_stress> – sets if an atomic stress file, filename.#.stress, will be produced.
Values: 0 - No atomic stress files is generated
1 - Calculates atomic stress for each atom and generates a *.stress file
Examples:

```
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
md:  100   5000   10   600.0   3   3   0 ! Run a Molecular Dynamics NPH  simulation
```

MD initialization commands (Table E4)

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

md_step: (optional) ! MD time step
Parameters: <step> – integration time step given in femtoseconds (1 fs = 10^{-15} sec)
Range: real, positive
Default value: 1 fs
Example:

md_step: 0.5

diss: (optional) ! Heat dissipation coefficient in Nose-Hoover thermostat
Parameters: <diss> – dissipation value in ps^{-1} (1 ps = 10^{-12} sec)
Range: real, positive
Default value: 1.0 ps^{-1}
Example:
diss: 2.0

wmass: (optional) ! Effective wall mass in Parrinello-Rahman dynamics
Parameters: <wmass> – wall mass value
Range: real, positive
Default value: 16.0 atomic units (atu)
Example:
wmass: 24.0

wdamp: ! Effective damping coefficient in Parrinello-Rahman
Parameters: <wdamp> – damping coefficient value
Range: real, positive
Default value: 25.0 ps^{-2}
Example:
wdamp: 25.0

integrator: (optional) ! Defines which integrator algorithm is to be used.
Parameter: <Integrator>
Values: PC – 5-th order Gear predictor-corrector [20]
VV – 2-nd order Velocity-Verlet [21]
Default value: PC
Example:
integrator: VV ! Applies Velocity-Verlet integrator.

Langevin (Viscous) Dynamics

ld: ! Performs Langevin Dynamics simulation

Integrates the equation of motion:

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

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

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

Parameters:

<Number runs> <Run length> <Measure step> <T> <ensemble> <rigidity> <save_stress>

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

<ensemble>

Values: 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):

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

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

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

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

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
```

LD initialization commands (Table E4)

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

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

friction: (optional) ! Defines a friction coefficient, γ.

Parameters: <friction> – friction value in ps^{-1}

Range: real, positive

Default value: 1.0 ps^{-1}

Example:

friction: 1.0

Monte Carlo simulation

mc: ! Performs Monte Carlo simulation

Evolves the system according to the probability equation:

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

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> <rigidity> <save_stress>

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

<ensemble> – Defines the ensemble type as follows [1] (Table E5):

Values: 1 - Displacement moves only; $\Delta\Phi = \Delta E_p - Nk_B T ln\frac{V'}{V}$;

ΔE_p – change of the system potential energy, and V' and V are the new and old volumes of the system, respectively, when irigid > 1.

2 - Displacement moves at $\mu = const.$ (μPT ensemble)

$\Delta\Phi = \Delta E_p - Nk_B T ln\frac{V'}{V} + \Delta\mu_{\alpha\beta} + \frac{3}{2}k_B T ln\frac{m_\alpha}{m_\beta}$;

$\Delta\mu_{\alpha\beta} = \mu_\alpha - \mu_\beta$, where $\mu_\alpha = const$ and $\mu_\beta = const$ are the chemical potentials of atom types α and β, fixed through the command **mu:** as described in the **MC initialization commands** below.
Requires defined **mu:,** and **comp:** is ignored, if defined. The system composition is equilibrated at fixed $\mu_\alpha, \mu_\beta, \ldots$.

3 - Displacement moves + feedback-adjusted μ defined at a MC step (n) as

$\mu_\alpha{}^{(n)} = \mu_\alpha{}^{(n-1)} - A_\alpha\left(\frac{C_\alpha{}^{(n-1)} + C_\alpha{}^{(n-2)}}{2} - C_{\alpha,0}\right)$

where

- the initial value $\mu_\alpha{}^{(0)}$ is defined through the command **mu:**
- A_α is a rate adjustable parameter, defined through command **alpha:**
- $C_\alpha{}^{(n)}$ is the concentration at step (n)
- $C_{\alpha,0}$ is the targeted concentration of atoms of type α, defined through the command **comp:**

The use of **alpha:, mu:**, and **comp:**, commands in the pgmc.com file is explained in the **MC initialization commands** section below.
Updates of μ are made once at every MCS.

4 - Displacement moves + variance constraint [18], where
$\mu_\alpha{}^{(n)} = \mu_\alpha{}^{(n-1)} - 2A_\alpha \frac{C_\alpha{}^{(n-1)} - C_\alpha{}^{(n-2)}}{2}$.

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 chemical type of a pair of atoms. One swap MCS is automatically followed by one displacement MCS. The overall chemical composition of the system remains constant.

Example:

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

MC initialization commands (Table E6)

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

comp: ! Defines targeted chemical composition
! (required for ensembles 3 and 5)

Parameters: $< C_{\alpha,0} >, < C_{\beta,0} >, \ldots$,

$< C_{\alpha,0} >$ – Atomic concentration of the α-element.

Range: $\{0 \le C_{\alpha,0} \le 1.0\}$: $C_{\alpha,0} + C_{\beta,0} + \cdots = 1.0$

Example: ! For a system of three chemical elements, Al, Co, Ni.

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 $C_{Al,0} + C_{Co,0} + C_{Ni,0} = 1$, otherwise the program exits with an error message.

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

Parameters: $< \mu_{\alpha,0} >, < \mu_{\beta,0} >, \ldots$

Range: real, at least one must be 0 used as a reference.

Example:

mu: 0. -0.33 -0.20 ! Defines chemical potential per atom for the listed elements.
! in the order of their definition at the beginning of the
! pgmc.com file. Since the code uses only the differences
! between them, at least one of the values must be 0.

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

Parameters: $< A_\alpha >, < A_\beta >, \ldots$

Range: real, positive

Example:

alpha: 0. 0.01 0.01 ! Sets the constant of proportionality, A_α, to the adjustment
! of the chemical potential for element α.

Multi-Cell Monte Carlo (MC2) (Appendix C)

A detailed description of the MC2 algorithm is presented in Appendix C. This Section describes the commands controlling the simulation in the MC2 regime.

mc2: ! Performs Multi-Cell Monte Carlo (MC2) simulation [6].

Parameters: (same as in mc: command)
<Number runs> <Run length> <Measure step> <T> <ensemble> <rigidity> <save_stress>

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

<ensemble> – Defines the ensemble type as follows:
Values: 1 - Simple multi-cell without lever rule [6]
2 - Multi-cell with lever rule [6]
3 - Multi-cell with lever rule and a free energy predictor-corrector [6]
Example:
mc2: 1 40000 100 400.0 2 3 0 ! Execute 1 run of 40,000 cycles, reporting every 100-th
! cycle at T = 400 K in a multi-cell with lever rule regime at
! constant stress adjusting h_{11}, h_{22}, h_{33} system parameters.

MC2 initialization commands

(Must be defined before the first **mc2:** command line)

MC2 regime needs three additional commands in pgmc.com, placed before the **mc2:** command. These commands are **systems:**, **w_lever:**, and **mc2_eqlb:**

systems: ! Number of phases to search for phase coexistence.
Parameters: <number of phases>
Range: integer, positive

The user needs to construct <number of phases> of initial structure files containing the atomic structures for each phase. These are set as multiple input files as: **structure1.plt**, **structure2.plt**, … etc. (or **structure1.lam**, **structure2.lam**, … etc. for the lammps format structure files).

Example:
systems: 2 ! Simulates 2 phases to coexist in equilibrium.

When executed, ParaGrandMC associates one MPI rank to each system (phase), thus the number of MPI ranks must be equal to the number of systems (phases). Even if ParaGrandMC is executed on one processor, multiple MPI ranks need to be used to run in MC2 regime.

MPI-parallelism inside a system (phase) is not supported. The parallelism is provided at the multi-core level or at a GPU level, if available. For example, on a 16-core machine, a two-phase MC2 simulation can be executed as:

```
export OMP_NUM_THREADS=8
mpirun -np 2 ./pgmc                    ! Running 2 mpi ranks of 8 threads each.
```

This will create two MPI-ranks running 8 threads each. Assigning any other number of MPI-ranks will issue an error message.

w_lever: (optional)
Parameters: <w> – Weight factor in the predictor-corrector lever method as described in [6]
Range: real, $0 \leq w \leq 1.0$
Default value: 0
Example:
w_lever: 0.7

mc2_eqlb: ! Defines the equilibration stage at the beginning of each MC2 step
Parameters:
<Equilbr_mode> <Run_length> <T> <ensemble> <rigidity>

<Equilbr_mode> ! Method of equilibration between
Values: {mc, md}
mc - Monte Carlo equilibration
md - Molecular dynamics equilibration

<Run_length> – number of equilibration steps (mc or md steps)
Range: integer, positive

<T> – temperature of equilibration
Range: real, positive

Example:
```
mc2_eqlb: md  1000  400.0  1  3      ! MD run of 1000 MDS at T = 400 K in NPT ensemble.
mc2:  1  40000   100  400.0  2  3  0  ! Continue with MC2 of 1 run at 40,000 MDS at 400 K.
```

Energy Minimization or Quench

q: ! Performs energy minimization of the atomic structure.

The energy minimization algorithm is based on the Projected Velocity-Verlet algorithm as described in [7]. The minimization is performed by selecting a new velocity of a particle according to the rule:

$$\vec{u}^{new}(t) = \begin{cases} \left(\vec{u}(t).\hat{F}\right)\hat{F} & if \quad \vec{u}(t).\hat{F} > 0 \\ 0 & \text{else} \end{cases} \text{ and } \quad \hat{F} = \vec{F}/\left|\vec{F}\right|.$$

Parameters:
<rigidity> – Defines system volume adjustments during energy minimization.
Values: 1 – 10: Have the same meanings as in the **md:** command.

Example:

```
q: 1          ! Performs energy minimization under fixed-box dimensions.
q: 3          ! Performs energy minimization with adjusting all 3 box dimensions.
```

Note: **q:** can be applied anywhere in the pgmc.com file before or after **md:** or **mc:** commands. It will quench the current structure by minimizing its potential energy and effectively setting it at T=0.

Nudged Elastic Band (NEB) method (Appendix D)

A detailed description of the NEB algorithm [7] is presented in Appendix D. This Section describes the execution setup and the commands controlling the NEB simulation.

Execution:

In the NEB regime, the simulation explores a set of transition states between an initial and final state, defined as initial and final structures in structure1.plt and structure2.plt (or structure1.lam and structure2.lam in lammps format) files, respectively. These two structures need to be prepared by the user and must have identical atom numbers and atom ids. Usually, structure2 is a deformed or defected state of structure1.

The number of transition states, including the initial and final state, is defined by the number of the requested MPI-ranks. Since at least one intermediate state must be present, MPI-ranks ≥ 3, such as

mpirun -np 15 ./pgmc

will create 13 intermediate states run in parallel.

Additional parallelization could be achieved through multicore (multithread) parallelism inside each MPI-rank. For example on a 16-core machine, executing:

export OMP_NUM_THREADS=2

mpirun -np 8 ./pgmc

will create NEB simulation with 8 states (6 intermediate) running on 8 MPI-ranks using 2 threads each.

Command

neb: ! Starts the NEB simulation.

Parameters:
<NEB type> <rigidity> <k_{min}><k_{max}>

<NEB type> – defines a NEB algorithm to be used.
Values: 1 - Standard NEB following [7] (see Appendix D)
2 - NEB using a modified tangent vector [22]

<rigidity> – same as in the **md:** command

$< k_{min,max} >$ – minimum and maximum spring constants
Range: real, positive

Example:
neb: 1 3 1.0 2.0 ! start a standard NEB of rigidity=3 and $\Delta k = 2.0 - 1.0 = 1.0$

Commands defining external load (all optional)

stress: ! Apply external stress.
Parameters: $< \sigma_{11}, \sigma_{22}, \sigma_{33}, \sigma_{12}, \sigma_{13}, \sigma_{23} >$ – stress tensor components
Range: real
Example:
stress: 0.5 0.4 0.3 0.12 0.15 -0.2 ! Apply σ(1,1), σ(2,2), σ(3,3), σ(1,2), σ(1,3), σ(2,3) (GPa).

strain: ! Apply external strain.
Parameters: $< \varepsilon_{11}, \varepsilon_{22}, \varepsilon_{33}, \varepsilon_{12}, \varepsilon_{13}, \varepsilon_{23} >$ – strain tensor components
Range: real
Example:
stress: 0.01 0.004 0.01 0.012 0.015 -0.002 ! Apply ε(1,1), ε(2,2), ε(3,3), ε(1,2), ε(1,3), ε(2,3).

Note: stress: and strain: commands have to be consistent with the **rigidity** parameter that defines which components of the h(i,j) matrix can change and in what way (see Table E3).

Incremental Change of System Parameters (all optional)

ParaGrandMC v.3 allows for a stepwise incremental change of certain system parameters, such as temperature, external load in terms of stress or strain, chemical composition, and the chemical potentials applied on the system.

All of the incremental change commands are best implemented in combination with the **loop:** command for repeated increments.

Temperature

dt: ! Defines an increment of the system temperature.
Parameter: <ΔT> temperature change at each consecutive run.
Range: real
Example:
dT: 10.5 ! Increment T by 10.5 K at each follow-up runs: T = T+dT

Example for a gradual temperature increase:
md: 1 10 5 10.0 1 3 0 ! Starts MD at T=10 K.
dT: 10 ! Set dT = 10 K increment.
loop: 3 ! Starts a loop of 3 cycles.
md: 1 10 5 10.0 1 3 0 ! T=100 is ignored; instead: T = T + 10 K is applied.
end: loop ! Loop end
dT: 0 ! Cancels the T-increment.
md: 1 10 5 110.0 1 3 0 ! T is set to the value: T = 110 K

Stress

dstress: ! Defines an increment of the system stress load.
Parameters:
<$\left(\Delta\sigma_{xx}, \Delta\sigma_{yy}, \Delta\sigma_{zz}, \Delta\sigma_{xy}, \Delta\sigma_{xz}, \Delta\sigma_{yz}\right)$> - stress tensor change by components.
Range: real
Example:
dstress: 0.01 0.01 0.01 0. 0. 0. ! Increment $\sigma_{xx}, \sigma_{yy}, \sigma_{zz}$ by 0.01 GPa at each follow
! up run: $\sigma_{\alpha\beta} = \sigma_{\alpha\beta} + \Delta\sigma_{\alpha\beta}$.
dstress: 0. 0. 0. 0. 0. 0. ! Stops the stress increment.

Strain

dstrain: ! Defines an increment of the system strain load.
Parameters:

<$(\Delta\varepsilon_{xx}, \Delta\varepsilon_{yy}, \Delta\varepsilon_{zz}, \Delta\varepsilon_{xy}, \Delta\varepsilon_{xz}, \Delta\varepsilon_{yz})$> - strain tensor change by components.
Range: real
Example:
dstrain: 0.01 0.01 0.01 0. 0. 0. ! Increment $\varepsilon_{xx}, \varepsilon_{yy}, \varepsilon_{zz}$ by 0.01 (1% strain) at each
! follow up run: $\varepsilon_{\alpha\beta} = \varepsilon_{\alpha\beta} + \Delta\varepsilon_{\alpha\beta}$.
dstrain: 0. 0. 0. 0. 0. 0. ! Stops the strain increment.

Chemical composition

To be used with **mc:** command, in ensemble = 3 only.

dcomp: ! Changes the chemical composition of the system.
Parameters: <change % of element 1> <change % of element 2> …
Range: real < 1.0

Example:
dcomp: 0.05 -0.05 ! Increases 1-st elem. by 5% decreasing the 2-nd element by 5%
! (to keep the sum of changes = 0) at each follow up runs.

dcomp: 0. 0. ! Stops the change of chemical comp.

Example pgmc.com file:

ini: 2 10. 0.02 0.0002 ! MC_rank, T (K), dr (Ang), dh/h
3 ! Number of elements
Ti ! First element
Al ! Second element
'TiAl_test' ! Output filename
measure: hii Sij
mu: 0. 0.70 ! Set chemical potential values (one must be 0)
dcomp: -0.05 0.05 ! Set dcomp = $\mp$5% increment of the 1-st and 2-nd elements
loop: 3 ! Starts a loop of 3 cycles.
mc: 1 1000 100 100.0 3 3 0 ! MC mode in ensemble=3 regime.
end: loop ! Loop end
dcomp: 0. 0. ! Cancels the dcomp increment
…

Chemical potential

To be used with **mc:** command, in ensemble = 2 only.

dmu: ! Changes the chemical potential of an element in the system.
Parameters: < change $\Delta\mu$ for element 1> <change $\Delta\mu$ for element 2> …

Range: real
Example:
dmu: 0.0 -0.05 ! Decrease the chemical potential of the second element by 0.05 eV.

Example pgmc.com file:

```
ini: 2 10. 0.02 0.0002      ! MC_rank, T (K), dr (Ang), dh/h
3                           ! Number of elements
Ti                          ! First element
Al                          ! Second element
'TiAl_test'                 ! Output filename
measure: hii Sij
mu: 0. 0.70                 ! Set chemical potential values (one must be 0)
dmu: 0.0 -0.05              ! Set dcomp = ∓5% increment of the 1-st and 2-nd elements
loop: 3                     ! Starts a loop of 3 cycles.
mc: 1 1000 100 100.0 2 3 0  ! MC mode in the μPT ensemble=2 regime.
end: loop                   ! Loop end
dmu: 0. 0.                  ! Cancels the dmu increment
…
```

Commands defining data output and visualization (Tables E7, E8)

measure: (optional) ! Defines simulation quantities to be measured and reported
! in the filename.#.dat file.

Parameters: <symbol 1> <symbol 2> <symbol 3> …

Values: List of valid symbol parameters (see Table E7)

General parameters: (can be used with any simulation regime, such as **md:**, **ld:**, **mc:**, **mc2:**, etc.)

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.

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 parameters: (can be used with MD, but they may not have a useful meaning)

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 measure parameters:

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

Values, defined by each of the above-listed parameters, are measured at each measure step and printed as separate columns in the **filename.#.dat** file. The order of the columns follows the order of the measure parameters as listed in the **measure:** command. The first 6 columns are fixed and always present, showing the following values: MCS (MDS), Total MCS (MDS), E_k, E_p, E_{tot}, T.

Default values: hij Sij

Example:
measure: Ti Sij abc comp acc_rate

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

Parameters: (All are optional)

Values: List of valid symbol parameters for visualization (see Table E8)

Ep – potential energy (eV/atom)
Ek – kinetic energy (eV/atom) = $\frac{1}{2}mv^2$ (MD) or = $\frac{3}{2}k_BT$ (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 = $\frac{2}{3}E_k/k_B$
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)

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.

Example:
ovito: Ep Ek T Sij

Commands defining simulation constraints (all optional) **(Table E9)**

const: ! Defines a constraint type

Parameters:
<constraint type> <element>

<constraint type> – defines chemical or positional constraint in binary format:
Values:
bit = 0 – not constrained; 1 – fixed.

Bit order of the constraints:

bit 4	bit 3	bit 2	bit 1
Chem.comp.	z-coord.	y-coord.	x-coord.

<element> – symbol of the chemical element subject to the constraints defined by bits 1-4.
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 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 ($11_{10} = 1011_2$) and be fixed in the *x*- and *y*- directions. Atom 336002 will be constrained to be fixed in the *z*-direction ($4_{10} = 0100_2$), 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 (i.e., no constraints of any type):

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> <element>
<x_min>, … <z_max> – minimum and maximum unit coordinates in x-, y-, and z- directions of the spatial constrained box.

Range: real: (0., 1.0) – normalized unit box values

<constraint type> – defines chemical or positional constraint in binary format, same as in **const:**
<element> – symbol of the chemical element subject to the constraints defined by bits 1-4 in the <constraint type>.

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.

Chemical composition constraints – active in MC ensemble > 2 simulation mode only.

high: ! Sets an upper limit of the variation of the chemical
! content of the elements (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.
Range: real: (0., 1.0)

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.
Range: real: (0., 1.0)

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 above must be between the respective **high:** and **low:** values, otherwise the program exits with an error.

Acknowledgements

V. Yamakov is sponsored through RSES Contract No. 80LARC23DA003 with Analytical Mechanics Associates. 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.

Appendix A. PINN potential file

Description:

The physics and properties of physics-informed neural network (PINN) potentials are published by Pun et al. [8,9]. 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

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

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

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

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

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):

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

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 for all atoms. The values of the base BOP parameters are given in the PINN potential file in the following order:

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

Example of $n_{el} = 1$ (mono-atomic PINN):

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

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

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

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

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

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

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

and

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

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:

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

Note that the cutoff function used in this calculation,

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

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:

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

From 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 $(s, l) : \ s = 1,2, \ldots \ s_{max}, \ l = l_1, l_2, \ldots \ l_{max}$

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

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

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

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

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

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):

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

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.(A8a,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} , $r_0^{(1)}, r_0^{(2)}, \ldots\ r_0^{(s_{max})}$, see Eq.(A12)
4 240 16 16 42 ! number of ANN layers *M*, and node number in each layer: $\eta_{max}^{(1)}, \ldots\ \eta_{max}^{(M)}$,
! Eq. (A16)
1 9.4050390000e+00 1.1232620000e+01 4.5610620000e+0, …
8.5633810000e-01 2.3007250000e-01 -1.9998170000e-01…
…
! flag, a list of base BOP parameters $\left(A_1^1, A_1^2, \ldots\ \lambda_{n_{el}}^{n_{el}n_{el}}\right)$ arranged as in Eq.(A7)
! and spanned over one or more lines. Their number, $\eta_{max}^{(M)} = 42$, is given in the
! preceding line.
! If flag = 0, do not use base BOP parameters,
! i.e., set $\left(A_1^1, A_1^2, \ldots\ \lambda_{n_{el}}^{n_{el}n_{el}}\right) = (0,0, \ldots 0)$ in Eq.(A6).
! If flag = 1, apply the base BOP parameters in Eq.(A6).

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:

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

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

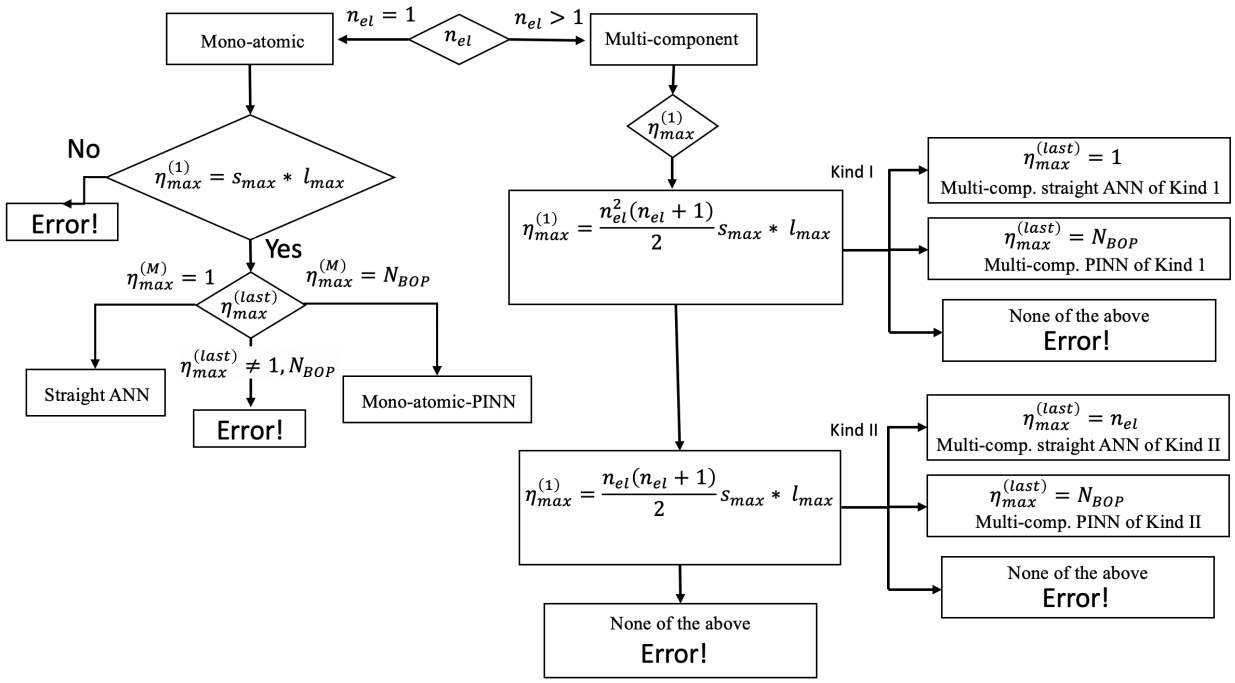
$$\left.\begin{array}{c} \vdots \\ w_{11}^{(M-1)}, w_{21}^{(M-1)}, \ldots\ w_{\eta_{max}^{(\ )}1}^{(M-1)} \\ \ldots \\ w_{1,\eta_{max}^{(M-1)}}^{(M-1)}, w_{2,\eta_{max}^{(M-1)}}^{(M-1)}, \ldots w_{\eta_{max}^{(M)}\eta_{max}^{(M-1)}}^{(M-1)} \\ b_1^{(M)}, b_2^{(M)}, \ldots b_{\eta_{max}^{(M)}}^{(M)} \end{array}\right\}\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

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. Multi-component
    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.



Appendix B. BOP potential 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 [10], 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. Description of the MC2 algorithm

The MC2 algorithm is designed to search for phase coexistence in a multi-component system under isobaric-isothermal conditions at various concentrations of the building components (chemical elements). The algorithm closely follows the technique proposed by E. Antillon and M. Ghazisaeidi [6].

The possible number of phases in coexistence, ϕ, is determined by the Gibbs phase rule:

$$F = C - \phi + 2$$

where $C > 1$ is the number of components (chemical elements), and F is the number of number of degrees of freedom. The imposed isobaric-isothermal conditions in the current implementation sets $F \geq 2$ since $P = const$ and $T = const$ can be satisfied only if P and T could be evolved independently following the phase coexistence line. This gives $C \geq \phi \geq 2$. Currently only two- or three-phase systems ($\phi = 2$ or 3) are possible to simulate, but the number of components C can be larger.

The following symbols are defined:

$\{\alpha, \beta, \dots\}$ – set of phases (systems or cells)
$\{A, B, \dots\}$ – set of chemical elements (species or components)
n^{α}– number of atoms in phase α (regardless of the species)
n_A – number of atoms of species A (regardless of the phase)
n_A^{α} – number of atoms of species A in phase α
N – total number of atoms including all phases and species
X_A^0 – initial concentration of species A
$X_A^{\alpha} = \frac{n_A^{\alpha}}{N}$ – concentration of species A in phase α
$f^{\alpha} = \frac{n^{\alpha}}{N}$ – molar fraction of phase α
G_m^{α} – molar free energy of phase α

The following Monte Carlo moves are implemented in MC2:

1. Displacement move: a random atom is randomly displaced in a random direction.
   Acceptance probability: $\wp_{displ}^{acc} = min\{1, e^{-\Delta U_{\Delta r}^{\alpha}/k_B T}\}$,
   $\Delta U_{\Delta r}^{\alpha}$ is the change of the potential energy in phase α.

2. Flip move: a random atom switches species to another randomly selected species in the same phase.
   Acceptance probability:
   Ensemble 1: simple flip: $\wp_{flip}^{acc} = min\{1, e^{-\Delta U_{A\leftrightarrow B}^{\alpha}/k_B T}\}$, $\Delta U_{A\leftrightarrow B}^{\alpha}$ - change of the potential energy in phase α due to atom type change ($A \leftrightarrow B$)

Ensemble 2: coordinated flip in two phases simultaneously applying mass transfer through lever rule:

$\wp_{flip}^{acc} = min\{1, exp\{-\Delta G\}\} = min\{1, exp\{-(G_{new} - G_{old})\}\} =$

$min\left\{1, exp\left\{-N\left[\left(F_{new}^{\alpha} f_{new}^{\alpha} + F_{new}^{\beta} f_{new}^{\beta}\right) - \left(F_{old}^{\alpha} f_{old}^{\alpha} + F_{old}^{\beta} f_{old}^{\beta}\right)\right]\right\}\right\}$

$F^{\alpha}(P,T) = (u^{\alpha} + Pv^{\alpha})/k_B T + (X_A^{\alpha} \ln X_A^{\alpha} + X_B^{\alpha} \ln X_B^{\alpha} - \ln v^{\alpha})$

$u^{\alpha} = U^{\alpha}/n^{\alpha}; \quad v^{\alpha} = V^{\alpha}/n^{\alpha}$

Mass balance equation: (2 phases)

$$\begin{pmatrix} X_A^0 \\ X_B^0 \end{pmatrix} = \begin{pmatrix} X_{A,new}^{\alpha} & X_{A,new}^{\beta} \\ X_{B,new}^{\alpha} & X_{B,new}^{\beta} \end{pmatrix} \begin{pmatrix} f_{new}^{\alpha} \\ f_{new}^{\beta} \end{pmatrix}$$

Solve for $\begin{pmatrix} f_{new}^{\alpha} & f_{new}^{\beta} \end{pmatrix}$:

$$\begin{pmatrix} f_{new}^{\alpha} \\ f_{new}^{\beta} \end{pmatrix} = \begin{pmatrix} X_{A,new}^{\alpha} & X_{A,new}^{\beta} \\ X_{B,new}^{\alpha} & X_{B,new}^{\beta} \end{pmatrix}^{-1} \begin{pmatrix} X_{Al}^0 \\ X_{Ti}^0 \end{pmatrix}.$$

Ensemble 3: lever rule with predictor-corrector flip:

$\wp_{flip}^{acc} = min\left\{1, exp\left\{-\Delta\tilde{G}\right\}\right\};$

$\Delta\tilde{G} = \Delta G(1-w) + w\frac{n_{ij}^{\alpha}}{2}\left[f_{old}^{\alpha}\left(\Delta\mu_{ij}^{\alpha} - \Delta\mu_{ij}^{\beta}\right)_{old} + f_{new}^{\alpha}\left(\Delta\mu_{ij}^{\alpha} - \Delta\mu_{ij}^{\beta}\right)_{new}\right]$

n_{ij}^{α} - number of replacements (flips) $(i \to j)$ in phase α

$\Delta\mu_{ij}^{\alpha} = \mu_i^{\alpha} - \mu_j^{\alpha}$ - the difference in the chemical potentials between species i and j

$0 \le w \le 1$ – weight factor (w_lever parameter in pgmc.com).

$\Delta\mu_{AB} = \mu_A - \mu_B = -k_B T ln\langle\frac{n_A}{n_B+1} e^{-\frac{\Delta U_{A\leftrightarrow B}}{k_B T}}\rangle$, Widom test particle method [31]

3. Swap move (intracell swap): choose two particles of species A and B both from same phase α and flip their species simultaneously.
   Acceptance probability: $\wp_{swap}^{acc} = min\{1, e^{-\Delta U_{\Delta r}^{\alpha}/k_B T}\}$,
   $\Delta U_{\Delta r}^{\alpha}$ is the change of the potential energy in phase α.

4. Exchange move (intercell swap): choose a particle of species A from phase α and a particle of species B from phase β and flip their species simultaneously.
   Acceptance probability: $\wp_{swap}^{acc} = \frac{n_A^{\alpha} n_B^{\beta}}{\left(n_A^{\beta}+1\right)\left(n_B^{\alpha}+1\right)} e^{-\frac{\Delta U^{\alpha} + \Delta U^{\beta}}{k_B T}}$,
   ΔU^{α} – change of energy in phase α; ΔU^{β} – change of energy in phase β.

5. Volume change move in all phases:
   Acceptance probability:

$$\wp_{volume}^{acc} = min\left\{1, \left(\frac{V^{\alpha} + \Delta V^{\alpha}}{V^{\alpha}}\right)^{n^{\alpha}} \left(\frac{V^{\beta} + \Delta V^{\beta}}{V^{\beta}}\right)^{n^{\beta}} e^{-[\Delta U^{\alpha} + \Delta U^{\beta} + P(\Delta V^{\alpha} + \Delta V^{\beta})]/k_B T}\right\}$$

One MC2 cycle consists of the following:

1. One displacement MC step = N displacement moves
2. One MC move (not a step) randomly chosen between moves 2 to 5 described above with probabilities [6]:
   - Flip move: 30%
   - Swap move: 30%
   - Exchange move: 30%
   - Volume change move (for all systems): 10% - follows <rigidity> number
     If <rigidity>=1, this move is executed, but does not change the system size.

Appendix D. Description of the NEB algorithm

In the NEB regime the simulation explores a set of transition states between an initial and final state. A transition state (i) is defined as a $3N$ dimensional point of coordinates $\vec{R}_i$ in the configurational space of a system of N atoms (see Figure D1). The transition states are connected through virtual "springs" having spring constants k_i, forming a "band". The energy of the band is defined as the sum of the atomic energies of all transition states plus the energy of the elastic springs between them. Minimizing the energy of the band leads to the band forming a minimum energy path between the initial and the final state of the system while the saddle point defines the transition energy between these two states.

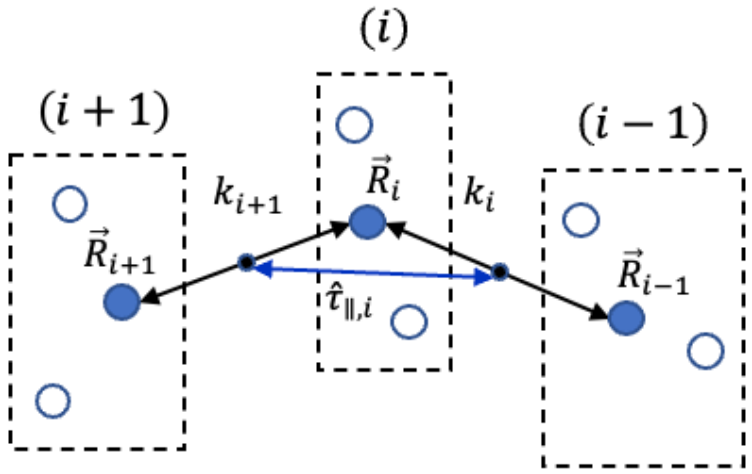


Figure D1. A schematic representation of three transitions states, $\vec{R}_{i+1}, \vec{R}_i, \vec{R}_{i-1}$, connected by virtual "springs" of spring constants k_{i+1} and k_i.

A tangent vector, $\vec{\tau}_{\parallel,i}$ at state (i) is defined as

$$\vec{\tau}_{\parallel,i} = \frac{\vec{R}_{i+1} - \vec{R}_i}{\left|\vec{R}_{i+1} - \vec{R}_i\right|} + \frac{\vec{R}_i - \vec{R}_{i-1}}{\left|\vec{R}_i - \vec{R}_{i-1}\right|}$$

where $\vec{R}_{i=1,\dots P} = \{\vec{r}_{n=1,\dots N,i}\}$ are $3N$ dimensional vectors for P systems of N particles each, and

$$\left|\vec{R}_{i+1} - \vec{R}_i\right| = \left[\sum_{n=1}^{N} \sum_{\alpha=x,y,z} \left(\alpha_{n,i+1} - \alpha_{n,i}\right)^2\right]^{1/2}.$$

A normalized unit tangent vector is introduced as $\hat{\tau}_{\parallel,i} = \vec{\tau}_{\parallel,i} / \left|\vec{\tau}_{\parallel,i}\right|$.

In **NEB type 1**, the force acting on a given transition state (i), $\vec{F}_i^{NEB}$, is defined as

$$\vec{F}_i^{NEB} = \vec{F}_i^0 + f(\phi_i)\left[\vec{F}_i^s - \vec{F}_i^s\big|_{\parallel}\right],$$

where:

$$\vec{F}_i^0 = -\vec{\nabla}V(\vec{R}_i)\big|_\perp + \vec{F}_i^s\big|_\parallel$$

with $V(\vec{R}_i)$ being the atomic potential energy at the state point $\vec{R}_i$, and $-\vec{\nabla}V(\vec{R}_i)\big|_\perp$ is the perpendicular to the elastic band component of the generalized 3*N*-dimensional atomic force, defined as

$$\vec{\nabla}V(\vec{R}_i)\big|_\perp = \vec{\nabla}V(\vec{R}_i) - \left[\vec{\nabla}V(\vec{R}_i).\hat{\tau}_{\parallel,i}\right]\hat{\tau}_{\parallel,i}.$$

Respectively, the force parallel to the elastic band is

$$\vec{F}_i^s\big|_\parallel = (\vec{F}_i^s.\hat{\tau}_{\parallel,i})\hat{\tau}_{\parallel,i}.$$

The switching function, $f(\phi_i)$, is defined as

$$f(\phi_i) = \frac{1}{2}\left[1 + \cos\left(\pi\frac{\cos\phi_i - \cos\phi_2}{\cos\phi_1 - \cos\phi_2}\right)\right]$$

where

$$\phi_1 = \frac{5}{180}\pi = \frac{\pi}{36};\ \ \phi_2 = \frac{30}{180}\pi = \frac{\pi}{6}$$

and

$$\cos\phi_i = \frac{(\vec{R}_{i+1} - \vec{R}_i).(\vec{R}_i - \vec{R}_{i-1})}{|\vec{R}_{i+1} - \vec{R}_i||\vec{R}_i - \vec{R}_{i-1}|}$$

with

$$(\vec{R}_{i+1} - \vec{R}_i).(\vec{R}_i - \vec{R}_{i-1}) = \sum_{n=1}^{N}\sum_{\alpha=x,y,z}(\alpha_{n,i+1} - \alpha_{n,i})(\alpha_{n,i} - \alpha_{n,i-1}).$$

In **NEB type 2**, the switching function $f(\phi_i) \equiv 0$, so that the NEB function is simply defined as:

$$\vec{F}_i^{NEB} = \vec{F}_i^0 = -\vec{\nabla}V(\vec{R}_i)\big|_\perp + \vec{F}_i^s\big|_\parallel.$$

Instead, a modified tangent vector is used [22], defined as:

$$\hat{\tau}_{\parallel,i} = \begin{cases}\tau_i^+ \ if\ V_{i+1} > V_i > V_{i-1}\\ \tau_i^- \ if\ V_{i+1} < V_i < V_{i-1}\end{cases}$$

When $V_{i+1} > V_i < V_{i-1}$ or $V_{i+1} < V_i > V_{i-1}$, then

$$\hat{\tau}_{\parallel,i} = \begin{cases}\tau_i^+\Delta V_i^{max} + \tau_i^-\Delta V_i^{min} \ \ if\ V_{i+1} > V_{i-1}\\ \tau_i^+\Delta V_i^{min} + \tau_i^-\Delta V_i^{max}\ if\ V_{i+1} < V_{i-1}\end{cases}$$

where $\Delta V_i^{max} = max(|V_{i+1} - V_i|, |V_{i-1} - V_i|)$ and $\Delta V_i^{min} = min(|V_{i+1} - V_i|, |V_{i-1} - V_i|)$.

In both NEB types 1 and 2, the spring constant is defined as: [22]

$$k_i = \begin{cases} k_{max} - \Delta k \left(\frac{E_{max} - E_i}{E_{max} - E_{ref}} \right) & if \;\; E_i > E_{ref} \\ k_{max} - \Delta k & if \; E_i < E_{ref} \end{cases}$$

where $E_i = max(V_i, V_{i-1})$ and $E_{ref} = max(V_i, V_P)$ with $\Delta k = k_{max} - k_{min}$.
Note that when $k_{max} = k_{min}$, $\Delta k = 0$ and $k_i = const.$

The minimization of the band energy under NEB forces is performed by the energy minimization using the projected Velocity-Verlet algorithm as described in the **q:** command.

Appendix E. Tables of input and command parameters

Table E1. Supported interatomic potential types

Number in pot.dat file	Potential	Description
0	EAM	Embedded Atom Method [23]
1	ADP	Angular Dependent Potential [24]
2	MEAM	Reserved for the Modified Embedded Atom Method (in progress) [25]
3	Tersoff	Conventional Tersoff in functional form (Tersoff_1) [26]
4	Tersoff- modified	Modified Tersoff [26]
5	EAM/fs	Embedded Atom Method / Finnis-Sinclair [27]
6	BOP	Bond Order Potential [28]
7	LJ	Lennard-Jones potential [29]
100	ANN	Artificial Neural Network potential [30]
106	PINN	Physically Informed Neural Network potential with BOP as an empirical potential [5,6]

Table E2. 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: 2x2 or 3x3 (see Fig.1)	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 run: command
output:	File format	plt, lam (lammps)	Yes (plt)	Sets the output structure file format	Anywhere before the first run: command
time:	Start time	Integer > 0 [MCS]	Yes (0)	Sets the initial start time (if a continued simulation)	Anywhere before the first run: command
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 run: command
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:	σ(1,1), σ(2,2), σ(3,3), σ(1,2), σ(1,3), σ(2,3)	Real numbers for stress in GPa.	Yes (see text)	Apply external stress	Anywhere before mc: or md:
strain:	ε(1,1), ε(2,2), ε(3,3), ε(1,2), ε(1,3), ε(2,3)	Real numbers with absolute value << 1.0	Yes (see text)	Apply external strain	Anywhere before mc: or md:

Table E3. 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	$\begin{pmatrix} h_{11} & 0 & 0 \\ 0 & ah_{11} & 0 \\ 0 & 0 & bh_{11} \end{pmatrix}\Big\vert_{a,b=const}$	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

Table E4. Molecular Dynamics and Langevin dynamics 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	
	ensemble	0, 1	No	Thermostat (0 - off; 1 - on)	
	irigid	1-10 see Table C3.	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
	ensemble	0	No	Constant energy simulation	
		1		Constant temperature simulation using Langevin thermostat	
		2		Viscous, overdamped Langevin dynamics simulation	

Table E5. Monte Carlo ensemble regimes

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 **rigidity** value on page 26	$\Delta\Phi_1 = \Delta E_p - Nk_BTln\frac{V'}{V}$			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: $c \rightarrow c_0$	Varied through feedback: $d\mu/dt = -A(C - C_0)$ $\mu^{(n)} = \mu^{(n-1)} - A\left(\frac{c^{(n-1)} + c^{(n-2)}}{2} - c_0\right)$	$c_0 = c(t_0)$ $\mu_0 = \mu(t_0)$ A	$c, \mu(\alpha)$
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: $d\mu/dt = -A\,dC/dt$ $\mu^{(n)} = \mu^{(n-1)} - A\left(c^{(n-1)} - c^{(n-3)}\right)$	$c_0 = c(t_0)$ $\mu_0 = \mu(t_0)$ A	$c, \mu(\alpha)$
7	$\Delta\Phi_1$	Fixed: $c = c_0$	-	As in ensemble 1	As in ensemble 1

Table E6. Monte Carlo 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 run: 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 run: 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 run: 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: run: 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 C2.	No	Sets the simulation ensemble	
	irigid	1- 10 see Table C3.	No	Sets the rigidity control (sys. dimensions constraints)	

Table E7. 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:	
	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.%		
	mue	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 case)		
	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: $\langle x_t x_0 \rangle = \sum_{i=1}^{N} x_i(t) x_i(0)$	Example: 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 $D_E = \lim_{t \to \infty} \frac{1}{2t} \frac{1}{3N} \langle \sum_{i=1}^{N} [\vec{\boldsymbol{r}}_i(t) - \vec{\boldsymbol{r}}_i(0)]^2 \rangle$	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 E8. 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: or md:
	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) $E_k = \frac{1}{2}mv^2$ (MD) or $E_k = \frac{3}{2}k_BT$ (MC)	
	Et	Scalar (eV/atom)	Yes	Total mechanical energy $E_t = E_p + E_k$ (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: $A = E_p + E_k + Q$	
	T	Scalar (K)	Yes	Temperature: T = $\frac{2}{3}E_k/k_B$ (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	

Table E9. 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:
	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:
	x-min, x-max	$0 \le$ x-min < x-max ≤ 1.0		The constraint box is b/n x-min and x-max	Example: 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 or: box: 0.0 0.5 0.0 1.0 0.0 1.0 1011 Same, but to all atoms
	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	

References

[1] Yamakov, V., "Parallel grand canonical Monte Carlo (ParaGrandMC) simulation code", *NASA/CR–2016-219202*; http://www.sti.nasa.gov, https://software.nasa.gov/software/LAR-20457-1

[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] Martyna G.J., Tuckerma, M.E., Douglas, J.T., Klein, M.L., "Explicit reversible integrators for extended systems dynamics", *Mol. Phys.* 87 (1996) 1117-1157.

[6] Antillon, E., Ghazisaeidi, "Efficient determination of solid-state phase equilibrium with the multicell Monte Carlo method", M., *Phys. Rev. E* 101 (2020) 063306.

[7] Jonsson, H., Mills, G., Jacobsen, K.W., ''Nudged elastic band method for finding minimum energy paths of transitions,'' in *Classical and Quantum Dynamics in Condensed Phase Simulations*, edited by Berne, B.J., Ciccotti, G., Coker, D.F., (World Scientific, Singapore, 1998), p. 385.

[8] 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.

[9] 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.

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

[11] https://www.vasp.at/documentation/

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

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

[14] Cormier, J., Rickman, J. M., Delph, T. J., "Stress calculation in atomistic simulations of perfect and imperfect solids", *J. Appl. Phys.* 89 (2001) 99.

[15] 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.

[16] Mondello, M., Grest, G.S., "Viscosity calculations of alkanes by equilibrium molecular dynamics", *J. Chem. Phys.* 106 (1997) 9327.

[17] Plimpton, S., Battaile, C., Chandross, M, et al., "Crossing the mesoscale no-man's land via parallel kinetic Monte Carlo", *SAND2009–*6226.

[18] Sadigh, B., Erhart, P., Stukowski, A., Caro, A., Martinez, E., Zepeda-Ruiz, L., "Scalable parallel Monte Carlo algorithm for atomistic simulations of precipitation in alloys," *Phys Rev B* 85 (2012) 184203-1-11.

[19] Melchionna S., Giovanni C., Holian, B.L., "Hoover NPT dynamics for systems varying in shape and size." *Mol. Phys.* 78 (1993) 533-544.

[20] Gear, C. W., “The numerical integration of ordinary differential equations of various orders”, *Technical Report ANL 7126* (1966) Argonne National Laboratory, Argonne, IL.
[21] Verlet, L., "Computer "experiments" on classical fluids. I. thermodynamical properties of Lennard-Jones molecules", *Physical Review* 159 (1967) 98.
[22] Henkelman G., Jonsson, H., “Improved tangent estimate in the nudged elastic band method for finding minimum energy paths and saddle points.” *J. Chem. Phys.* 113, 9978 (2000)
[23] 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.
[24] 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.
[25] Baskes, I., “Modified embedded-atom potentials for cubic materials and impurities”, *Phys. Rev. B.* 46 (1992) 2727.
[26] Tersoff, J., “Modeling solid-state chemistry: Interatomic potentials for mnlticomponent systems”, *Phys. Rev. B.* 39 (1989) 5566.
[27] Finnis, M.W., Sinclair, J.E. “A simple empirical n-body potential for transition metals”, *Phil Mag A* 50 (1984) 45.
[28] 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.
[29] Lennard-Jones, J.E., "On the determination of molecular fields", *Proc. R. Soc. Lond. A*, 106 (1924) 463.
[30] 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
[31] Widom, B., “Some topics in the theory of fluids”, *J. Chem. Phys.* 39 (1963) 2808-2812.