

# Main Page

From GPUMD

[Jump to navigation](#)[Jump to search](#)

## Contents

- 1 Notice
- 2 What is GPUMD?
- 3 Publications
- 4 Versions of GPUMD
- 5 Install and run GPUMD
  - 5.1 Download GPUMD
  - 5.2 Prerequisites for using GPUMD
  - 5.3 Compile GPUMD
  - 5.4 Run GPUMD
    - 5.4.1 Up to GPUMD-3.3.1
    - 5.4.2 Starting from GPUMD-v3.4
    - 5.4.3 Prepared examples
- 6 Inputs and outputs for GPUMD
  - 6.1 Inputs for the src/gpumd executable
  - 6.2 Output files for the src/gpumd executable
  - 6.3 Inputs for the src/nep executable
  - 6.4 Outputs for the src/nep executable
- 7 Tutorials for using GPUMD
  - 7.1 Tutorials for using the src/gpumd executable
  - 7.2 Tutorials for using the src/nep executable
- 8 Python interfaces of GPUMD
- 9 Theoretical formulations of GPUMD
- 10 Mailing list and discussion group of GPUMD
- 11 Want to develop GPUMD?
  - 11.1 Developers of GPUMD
  - 11.2 Important coding conventions of GPUMD
  - 11.3 Units system adopted within GPUMD
- 12 Acknowledgements

## Notice

**This manual is only for versions up to GPUMD-v3.5. Starting from GPUMD-v3.6, please use the new manual: <https://gpumd.org/>**

## What is GPUMD?

- GPUMD stands for **Graphics Processing Units Molecular Dynamics**. It is a general-purpose molecular dynamics (MD) package fully implemented on graphics processing units (GPU). It is written in CUDA C++ and requires a CUDA-enabled Nvidia GPU of compute capability no less than 3.5.
- It is **highly efficient for doing MD simulations with many-body potentials** such as the Tersoff potential.
- **Particularly good for heat transport applications**. See the theoretical formulations.

- It has **native support for the NEP machine learning potential**. That is, it can be used to **train a NEP potential** and then **run MD simulations with the trained potential**.
- **Installation of GPUMD is easy**. If you have a working CUDA development environment, it just requires a single make to install GPUMD in either Linux or Windows.

## Publications

See the publications (<https://github.com/brucefan1983/GPUMD/tree/master/publications>) that developed or used GPUMD.

## Versions of GPUMD

- This website documents up to GPUMD-v3.5.
- We have been recently focusing on developing the NEP machine-learned potential and the best version for empirical potentials should be GPUMD-v3.3.1.
- For a full description of the various versions of GPUMD, see the Github page <https://github.com/brucefan1983/GPUMD/releases>

## Install and run GPUMD

### Download GPUMD

- Go to GitHub (<https://github.com/brucefan1983/GPUMD>) and download the version(s) you need.

### Prerequisites for using GPUMD

- Hardware: You need to have an **Nvidia GPU card with compute capability no less than 3.5**.
- Software:
  - A CUDA toolkit **9.0 or newer**.
  - In **Linux** system, you also need a g++ compiler supporting at least c++11.
  - In **Windows** system, you also need the cl.exe compiler from Microsoft Visual Studio and a 64-bit version of make.exe (<http://www.equation.com/servlet/equation.cmd?fa=make>).

### Compile GPUMD

- Go to the src directory and type make. When the compilation finishes, two executables, nep and gpumd, will be generated in the src directory.
- **Please check the comments in the beginning of the makefile for some compiling options.**

### Run GPUMD

#### Up to GPUMD-3.3.1

- To run each executable, one has to prepare some input files and a **driver input file**, which is used to do one or more simulations using a single launch of an executable. The driver input file should have the following format:

```
number_of_simulations
path_1
path_2
...
```

Here `number_of_simulations` is the number of individual simulations you want to run within a single launch (by the operating system) of the executable (`src/nep` or `src/gpumd`) and `path_n` is the path of the directory containing the actual input files for the  $n$ -th simulation. Output files will be created in the directories containing the corresponding input files.

## Starting from GPUMD-v3.4

There will be no **driver input file** any more. To run one example, one just need to go to the directory of that example and type one of the following commands:

```
path/to/gpumd
path/to/nep
```

## Prepared examples

- Please check this link: <https://github.com/brucefan1983/GPUMD/tree/master/examples>

# Inputs and outputs for GPUMD

## Inputs for the `src/gpumd` executable

- To run one simulation using the `src/gpumd` executable, one has to prepare **at least** two input files:

Mandatory input files for the `src/gpumd` executable

Input filename	Brief description
<code>xyz.in</code>	Define the simulation model ( <b>before GPUMD-v3.4</b> )
<code>model.xyz</code>	Define the simulation model ( <b>starting from GPUMD-v3.4</b> )
<code>run.in</code>	Define the simulation protocol

- The `run.in` file is used to define the simulation protocol for `gpumd`. The code will execute the commands in this file one by one. If the code encounters an invalid command in this file, it will report an error message and exit. In this input file, blank lines and lines starting with `#` are ignored. One can thus write comments after `#`. All the other lines should be of the following form:

```
keyword parameter_1 parameter_2 ...
```

- The overall structure of a `run.in` file is as follows:
  - First, set up the potential model using the potential keyword.
  - Then, if needed, use the `minimize` keyword to minimize the energy of the whole system.
  - Then one can use the following keywords to do some **static calculations** (that is, non-MD simulations):
    - Use the `compute_cohesive` keyword to compute the cohesive energy curve.
    - Use the `compute_elastic` keyword to compute the elastic constants.
    - Use the `compute_phonon` keyword to compute the phonon dispersions.
  - Then, if you want to do **MD simulations**, set up the initial velocities using the `velocity` keyword and do a number of MD runs in the following way:
    - Specify an integrator using the `ensemble` keyword and optionally add keywords to further control the evolution and measurement processes.
    - Use the keyword `run` to run a number of steps according to the above settings.
    - One can repeat the above two steps.

- Here is the complete list of the keywords:

## Keywords for the run.in input file

Keyword	Brief description	Take action immediately?	Propagating from one run to the next?
velocity	Set up the initial velocities with a given temperature	<b>Yes</b>	N/A
potential ( <b>before GPUMD-v3.4</b> )	Set up a single potential	<b>Yes</b>	N/A
potential ( <b>starting from GPUMD-v3.4</b> )	Set up a single potential	<b>Yes</b>	N/A
minimize	Perform an energy minimization	<b>Yes</b>	N/A
compute_cohesive	Compute the cohesive energy curve	<b>Yes</b>	N/A
compute_elastic	Compute the elastic constants	<b>Yes</b>	N/A
compute_phonon	Compute the phonon dispersion	<b>Yes</b>	N/A
change_box	Change the box	<b>Yes</b>	N/A
ensemble	Specify an integrator for a run	No	No
time_step	Specify the time step for integration	No	<b>Yes</b>
neighbor	Control the neighbor list updating	No	No
fix	Fix (freeze) some atoms	No	No
deform	Deform the simulation box	No	No
dump_thermo	Dump some thermodynamic quantities	No	No
dump_position	Dump the atom positions	No	No
dump_netCDF	Dump the atom positions in the netCDF format	No	No
dump_restart	Dump a restart file	No	No
dump_velocity	Dump the atom velocities	No	No
dump_force	Dump the atom forces	No	No
dump_exyz	Dump an extended XYZ file with some data	No	No
compute	Compute some time- and space-averaged quantities	No	No
compute_shc	Calculate spectral heat current	No	No
compute_dos	Calculate the phonon density of states (PDOS)	No	No
compute_sdc	Calculate the self diffusion coefficient (SDC)	No	No
compute_hac	Calculate thermal conductivity using the EMD method	No	No
compute_hnemd	Calculate thermal conductivity using the HNEMD method	No	No
compute_gkma	Calculate modal heat current using the GKMA method	No	No
compute_hnema	Calculate modal thermal conductivity using the HNEMA method	No	No

run	Run a number of steps	Yes	No
-----	-----------------------	-----	----

## Output files for the src/gpumd executable

Output files for the src/gpumd executable

Output filename	Generated by which keyword?	Brief description	Output mode
thermo.out	dump_thermo	Some global thermodynamic quantities	Append
movie.xyz	dump_position	Trajectory (atom positions)	Append
restart.out ( <b>Before GPUMD-v3.4</b> ) or restart.xyz ( <b>Starting from GPUMD-v3.4</b> )	dump_restart	The restart file	<b>Overwrite</b>
velocity.out	dump_velocity	The velocity file	Append
force.out	dump_force	The force file	Append
compute.out	compute	Time and space (group) averaged quantities	Append
hac.out	compute_hac	Thermal conductivity data from the EMD method	Append
kappa.out	compute_hnemd	Thermal conductivity data from the HNEMD method	Append
shc.out	compute_shc	Spectral heat current data	Append
heatmode.out	compute_gkma	Modal heat current data from GKMA method	Append
kappamode.out	compute_hnema	Modal thermal conductivity data from HNEMA method	Append
dos.out mvac.out	compute_dos	Phonon density of states data	Append
sdc.out	compute_sdc	Self diffusion coefficient data	Append
D.out	compute_phonon	Dynamical matrices $D(\vec{k})$ for the input k points	Overwrite
omega2.out	compute_phonon	Phonon frequency square $\omega^2(\vec{k})$ for the input k points	Overwrite

## Inputs for the src/nep executable

- To run one simulation using the src/nep executable, one has to prepare **at least** three input files:

Input files for the `src/nep` executable

Input filename	Brief description
nep.in	Define the NEP potential
train.in and test.in	Provide the training and testing data ( <b>before GPUMD-v3.4</b> )
train.xyz and test.xyz	Provide the training and testing data ( <b>starting from GPUMD-v3.4</b> )

Outputs for the `src/nep` executable

- The output files for the `src/nep` executable are described here.

## Tutorials for using GPUMD

Tutorials for using the `src/gpumd` executableTutorials for using the `src/gpumd` executable

Tutorial name	Brief description
Tutorial: Thermal expansion ( <a href="https://github.com/brucefan1983/GPUMD/blob/master/examples/empirical_potentials/thermal_expansion/Thermal%20Expansion.ipynb">https://github.com/brucefan1983/GPUMD/blob/master/examples/empirical_potentials/thermal_expansion/Thermal%20Expansion.ipynb</a> )	Study thermal expansion of silicon crystal from 100 K to 1000 K
Tutorial: Density of states ( <a href="https://github.com/brucefan1983/GPUMD/blob/master/examples/empirical_potentials/density_of_states/Density%20of%20States.ipynb">https://github.com/brucefan1983/GPUMD/blob/master/examples/empirical_potentials/density_of_states/Density%20of%20States.ipynb</a> )	Calculate the vibrational density of states of graphene at 300 K
Tutorial: Thermal conductivity from EMD ( <a href="https://github.com/brucefan1983/GPUMD/blob/master/examples/empirical_potentials/thermal_transport_emd/Thermal%20Transport%20-%20EMD.ipynb">https://github.com/brucefan1983/GPUMD/blob/master/examples/empirical_potentials/thermal_transport_emd/Thermal%20Transport%20-%20EMD.ipynb</a> )	Calculate the lattice thermal conductivity of graphene at 300 K using the EMD method
Tutorial: Thermal transport from NEMD ( <a href="https://github.com/brucefan1983/GPUMD/blob/master/examples/empirical_potentials/thermal_transport_nemd_and_hnemd/ballistic/Thermal%20Transport%20-%20Ballistic.ipynb">https://github.com/brucefan1983/GPUMD/blob/master/examples/empirical_potentials/thermal_transport_nemd_and_hnemd/ballistic/Thermal%20Transport%20-%20Ballistic.ipynb</a> )	Calculate the spectral ballistic conductance of graphene using the NEMD and spectral decomposition
Tutorial: Thermal transport from HNEMD ( <a href="https://github.com/brucefan1983/GPUMD/blob/master/examples/empirical_potentials/thermal_transport_nemd_and_hnemd/diffusive/Thermal%20Transport%20-%20Diffusive.ipynb">https://github.com/brucefan1983/GPUMD/blob/master/examples/empirical_potentials/thermal_transport_nemd_and_hnemd/diffusive/Thermal%20Transport%20-%20Diffusive.ipynb</a> )	Calculate the spectral conductivity of graphene using the HNEMD method and spectral decomposition
Tutorial: Phonon dispersion ( <a href="https://github.com/brucefan1983/GPUMD/blob/master/examples/empirical_potentials/phonon_dispersion/Phonon%20Dispersion.ipynb">https://github.com/brucefan1983/GPUMD/blob/master/examples/empirical_potentials/phonon_dispersion/Phonon%20Dispersion.ipynb</a> )	Calculate the phonon dispersion of silicon crystal
Tutorial: Phonon vibration visualization ( <a href="https://github.com/brucefan1983/GPUMD/blob/master/examples/empirical_potentials/phonon_vibration_viewer/Phonon-Vibration-Viewer.ipynb">https://github.com/brucefan1983/GPUMD/blob/master/examples/empirical_potentials/phonon_vibration_viewer/Phonon-Vibration-Viewer.ipynb</a> )	Visualize phonon vibrations
Tutorial: Phonon participation ratio ( <a href="https://github.com/hityingph/Tutorial-on-atomic-simulations/blob/main/1.%20Phonon%20Participation%20Ratios/Phonon%20Participation%20Ratios.ipynb">https://github.com/hityingph/Tutorial-on-atomic-simulations/blob/main/1.%20Phonon%20Participation%20Ratios/Phonon%20Participation%20Ratios.ipynb</a> )	Calculating the phonon participation ratio in the frequency domain for prsitine and defect graphene
Tutorial: Elastic constant ( <a href="https://github.com/hityingph/Tutorial-on-atomic-simulation/blob/main/2.%20Elastic%20Constants%20Based%20on%20Strain%20Fluctuation%20Method/Elastic%20Constants%20Based%20on%20Strain%20Fluctuation%20Method.ipynb">https://github.com/hityingph/Tutorial-on-atomic-simulation/blob/main/2.%20Elastic%20Constants%20Based%20on%20Strain%20Fluctuation%20Method/Elastic%20Constants%20Based%20on%20Strain%20Fluctuation%20Method.ipynb</a> )	Elastic constant calculations from strain fluctuations

Tutorials for using the `src/nep` executableTutorials for using the `src/nep` executable

Tutorial name	Brief description
Tutorial: NEP tutorial ( <a href="https://github.com/brucefan1983/GPUMD/blob/master/example_s/nep_potentials/PbTe/train/nep_tutorial.ipynb">https://github.com/brucefan1983/GPUMD/blob/master/example_s/nep_potentials/PbTe/train/nep_tutorial.ipynb</a> )	Train a NEP machine-learning potential for PbTe crystal

# Python interfaces of GPUMD

Name	Brief description
gpyumd ( <a href="https://github.com/AlexGabourie/gpyumd">https://github.com/AlexGabourie/gpyumd</a> )	Pre- and post-process GPUMD simulations with Python
pyNEP ( <a href="https://github.com/bigd4/PyNEP">https://github.com/bigd4/PyNEP</a> )	Python interface for the NEP machine learning potential
calorine ( <a href="https://gitlab.com/materials-modeling/calorine">https://gitlab.com/materials-modeling/calorine</a> )	Python library to manage GPUMD simulations

## Theoretical formulations of GPUMD

- Here are the Theoretical formulations of GPUMD.

## Mailing list and discussion group of GPUMD

- Mailing list
  - You can use the following link to subscribe and unsubscribe to the mailing list: <https://www.freelists.org/list/gpumd>
  - To post a question, you can send an email to [gpumd@freelists.org](mailto:gpumd@freelists.org)
  - Here is the archive (public): <https://www.freelists.org/archive/gpumd/>
- Discussion group
  - There is a Chinese discussion group based on the following QQ 群:
    - 887975816

## Want to develop GPUMD?

### Developers of GPUMD

- GPUMD was first developed by **Zheyong Fan** (Previously Aalto University; [brucenju@gmail.com](mailto:brucenju@gmail.com)), with help from his colleagues Ville Vierimaa, Mikko Ervasti, and Ari Harju during 2012-2017.
- In 2018, **Alexander J. Gabourie** (PhD candidate at Stanford University; [gabourie@stanford.edu](mailto:gabourie@stanford.edu)) joined in and he is now an active developer.

### Important coding conventions of GPUMD

- We want to keep GPUMD as a **standalone code**. So we only use standard C++ (currently only up to c++11) and CUDA (currently only up to CUDA 9.0) libraries.
- We will make sure that the code is correct for any Nvidia GPU with compute capability no less than 3.5 and in both Linux and Windows systems.
- We use the `.clang-format` file in the package to format any CUDA C++ source code.

### Units system adopted within GPUMD

- We use the following basic units
  - Energy: eV (electron volt)
  - Length: Å (angstrom)
  - Mass: amu (atomic mass unit)
  - Temperature: K (kelvin)
  - Charge: e (elementary charge)
- The units for all the quantities are thus fixed.

- One only needs to make units conversions when dealing with inputs and outputs; all the quantities should be defined in the above units system elsewhere.

## Acknowledgements

- NSFC (<http://www.nsf.gov.cn/>) with project numbers 11404033 and 11974059.
- Aalto Science-IT project (<http://science-it.aalto.fi/>)
- Finland's IT Center for Science (CSC) (<https://www.csc.fi/>)

Retrieved from "[https://gpumd.zheyongfan.org/index.php?title=Main\\_Page&oldid=22495](https://gpumd.zheyongfan.org/index.php?title=Main_Page&oldid=22495)"

- 
- This page was last edited on 3 March 2023, at 18:11.



# The Buckingham-Coulomb potential

From GPUMD

[Jump to navigation](#)[Jump to search](#)

## Contents

- 1 Brief descriptions
- 2 Potential form
- 3 Parameters
- 4 Potential file
- 5 References

## Brief descriptions

- This is the Buckingham-Coulomb potential, which is also usually referred to as the rigid-ion potential.
- It currently only applies to systems with two atom types.
- The Coulomb potential is treated using the damped-shifted-force (DSF) method as proposed in [Fennell 2006].
- This potential will be removed starting from GPUMD-v3.4.

## Potential form

- It consists of the Buckingham potential

$$U_{ij} = A_{ij} \exp(-b_{ij}r_{ij}) - \frac{C_{ij}}{r_{ij}^6}$$

and a Coulomb potential.

- The Coulomb potential is evaluated using the damped-shifted-force (DSF) method [1]. The DSF version of the pairwise Coulomb potential can be written as:

$$U_{ij} = \frac{q_i q_j}{4\pi\epsilon_0} \left[ \frac{\operatorname{erfc}(\alpha r_{ij})}{r_{ij}} - \frac{\operatorname{erfc}(\alpha R_c)}{R_c} + \left( \frac{\operatorname{erfc}(\alpha R_c)}{R_c^2} + \frac{2\alpha}{\sqrt{\pi}} \frac{\exp(-\alpha^2 R_c^2)}{R_c} \right) (r_{ij} - R_c) \right],$$

where **erfc** is the complementary error function.

## Parameters

Parameter	Units
$A_{ij}$	eV
$b_{ij}$	$\text{\AA}^{-1}$
$C_{ij}$	$\text{\AA}^6$
$q_i$	e
$\alpha$	$\text{\AA}^{-1}$
$R_c$	$\text{\AA}$

- $\alpha$  is the electrostatic damping factor and  $R_c$  is the cutoff radius for the Coulomb potential.

- In GPUMD, we have fixed  $\alpha$  to  $0.2 \text{ \AA}^{-1}$ , which is a good choice according to the results in [Fennell 2006].

## Potential file

Currently, this potential only applies to systems with two atom types in GPUMD. The potential file for this potential model reads

```
ri 2
q_0 q_1 cutoff
A_00, b_00 C_00
A_11, b_11 C_11
A_01, b_01 C_01
```

## References

- [Fennell 2006] Christopher J. Fennell and J. Daniel Gezelter, *Is the Ewald summation still necessary? Pairwise alternatives to the accepted standard for long-range electrostatics* (<https://doi.org/10.1063/1.2206581>), J. Chem. Phys. **124**, 234104 (2006).

Retrieved from "[https://gpumd.zheyongfan.org/index.php?title=The\\_Buckingham-Coulomb\\_potential&oldid=22312](https://gpumd.zheyongfan.org/index.php?title=The_Buckingham-Coulomb_potential&oldid=22312)"

- 
- This page was last edited on 24 July 2022, at 17:46.

# The embedded atom method (EAM) potential

From GPUMD

[Jump to navigation](#)[Jump to search](#)

## Contents

- 1 Brief descriptions
- 2 Potential form
  - 2.1 General form
  - 2.2 The version by [Zhou 2004]
  - 2.3 The version by [Dai 2006]
- 3 Parameters
- 4 Potential file format
- 5 References

## Brief descriptions

- This is the EAM potential in some **analytical** forms as in [Zhou 2004] and [Dai 2006].
- For [Zhou 2004], one can simulate alloys with up to 10 atom types.
- For [Dai 2006], it currently only applies to systems with a single atom type.

## Potential form

### General form

- The site potential energy is

$$U_i = \frac{1}{2} \sum_{j \neq i} \phi(r_{ij}) + F(\rho_i).$$

- Here, the part with  $\phi(r_{ij})$  is a pairwise potential and  $F(\rho_i)$  is the embedding potential, which depends on the electron density  $\rho_i$  at site  $i$ . The many-body part of the EAM potential comes from the embedding potential.
- The density  $F(\rho_i)$  is contributed by the neighbors of  $i$ :

$$\rho_i = \sum_{j \neq i} f(r_{ij}).$$

- Therefore, the form of an EAM potential is completely determined by the three functions:  $\phi$ ,  $f$ , and  $F$ .

### The version by [Zhou 2004]

- The pair potential between two atoms of the same type  $a$  is

$$\phi^{aa}(r) = \frac{A^a \exp[-\alpha(r/r_e^a - 1)]}{1 + (r/r_e^a - \kappa^a)^{20}} - \frac{B^a \exp[-\beta(r/r_e^a - 1)]}{1 + (r/r_e^a - \lambda^a)^{20}}.$$

- The contribution of the electron density from an atom of type  $a$  is

$$f^a(r) = \frac{f_e^a \exp[-\beta(r/r_e^a - 1)]}{1 + (r/r_e^a - \lambda^a)^{20}}.$$

- The pair potential between two atoms of different types  $a$  and  $b$  is then constructed as

$$\phi^{ab}(r) = \frac{1}{2} \left[ \frac{f^b(r)}{f^a(r)} \phi^{aa}(r) + \frac{f^a(r)}{f^b(r)} \phi^{bb}(r) \right].$$

- The embedding energy function is piecewise:

$$F(\rho) = \sum_{i=0}^3 F_{ni} \left( \frac{\rho}{\rho_n} - 1 \right)^i, \quad (\rho < 0.85\rho_e)$$

$$F(\rho) = \sum_{i=0}^3 F_i \left( \frac{\rho}{\rho_e} - 1 \right)^i, \quad (0.85\rho_e \leq \rho < 1.15\rho_e)$$

$$F(\rho) = F_e \left[ 1 - \ln \left( \frac{\rho}{\rho_s} \right)^\eta \right] \left( \frac{\rho}{\rho_s} \right)^\eta, \quad (\rho \geq 1.15\rho_e)$$

### The version by [Dai 2006]

This is a very simple EAM-type potential which is an extension of the Finnis-Sinclair potential. The function for the pair potential is

$$\phi(r) = \begin{cases} (r - c)^2 \sum_{n=0}^4 c_n r^n & r \leq c \\ 0 & r > c \end{cases}$$

The function for the density is

$$f(r) = \begin{cases} (r - d)^2 + B^2(r - d)^4 & r \leq d \\ 0 & r > d \end{cases}$$

The function for the embedding energy is

$$F(\rho) = -A\rho^{1/2}.$$

## Parameters

See [Zhou 2004] and [Dai 2006].

## Potential file format

- The potential file for the version in [Zhou 2004] reads

```
eam_zhou_2004 num_types
r_e f_e rho_e rho_s alpha beta A B kappa lambda F_n0 F_n1 F_n2 F_n3 F_0 F_1 F_2 F_3 eta F_e cutoff
```

- There will be num\_types rows of parameters, but we have only written a single row above. The order of the rows should be consistent with the atoms types you defined in xyz.in and specified by the potential keyword in run.in.
- The last parameter cutoff is the cutoff distance which is not intrinsic to the model. The order of the parameters is the same as in Table III of the paper by Zhou *et al.*. **For multi-component systems, GPUMD will use the largest cutoff for every atom type.**
- The potential file for the version in [Dai 2006] reads

```
eam_dai_2006 1
A d c c_0 c_1 c_2 c_3 c_4 B
```

## References

- [Zhou 2004] X. W. Zhou, R. A. Johnson, and H. N. G. Wadley, *Misfit-energy-increasing dislocations in vapor-deposited CoFe/NiFe multilayers* (<https://doi.org/10.1103/PhysRevB.69.144113>), Phys. Rev. B **69**, 144113 (2004).
- [Dai 2006] X D Dai, Y Kong, J H Li and B X Liu, *Extended Finnis–Sinclair potential for bcc and fcc metals and alloys* (<https://doi.org/10.1088/0953-8984/18/19/008>), J. Phys.: Condens. Matter **18**, 4527 (2006).

Retrieved from "[https://gpumd.zheyongfan.org/index.php?title=The\\_embedded\\_atom\\_method\\_\(EAM\)\\_potential&oldid=22141](https://gpumd.zheyongfan.org/index.php?title=The_embedded_atom_method_(EAM)_potential&oldid=22141)"

- 
- This page was last edited on 20 January 2022, at 09:30.

# The ensemble keyword

From GPUMD

[Jump to navigation](#)[Jump to search](#)

## Contents

- 1 Purpose
- 2 Grammar
  - 2.1 nve
  - 2.2 nvt\_ber
  - 2.3 nvt\_nhc
  - 2.4 nvt\_bdp
  - 2.5 nvt\_lan
  - 2.6 nvt\_bao
  - 2.7 npt\_ber
  - 2.8 npt\_scr
  - 2.9 heat\_nhc
  - 2.10 heat\_bdp
  - 2.11 heat\_lan
- 3 Units and suggested parameters
- 4 Examples
- 5 Caveats

## Purpose

- This keyword is used to set up an integration method (an integrator). The name of this keyword might be somewhat misleading, but is not very bad.

## Grammar

- The number of parameters depends on the first parameter, which can be:

```
nve
nvt_ber
nvt_nhc
nvt_bdp
nvt_lan
nvt_bao
npt_ber
npt_scr
heat_nhc
heat_bdp
heat_lan
```

### nve

- When the first parameter is nve, it means that the ensemble for the current run is NVE (micro-canonical). There is no need to further specify any other parameters. Therefore, the full command is

```
ensemble nve
```

## nvt\_ber

- When the first parameter is `nvt_ber`, it means that the ensemble for the current run is NVT (canonical) generated by using the Berendsen (ber) method. In this case, one needs to specify an initial target temperature  $T_1$ , a final target temperature  $T_2$ , and a parameter  $T_{\text{coup}}$  which reflects the strength of the coupling between the system and the thermostat. The full command is

```
ensemble nvt_ber T_1 T_2 T_coup
```

The target temperature (not the instant system temperature) will vary linearly from  $T_1$  to  $T_2$  during a run. We will discuss about  $T_{\text{coup}}$  later.

## nvt\_nhc

- When the first parameter is `nvt_nhc`, it is similar to the case of `nvt_ber`, but using the Nose-Hoover chain (nhc) method.

## nvt\_bdp

- When the first parameter is `nvt_bdp`, it is similar to the case of `nvt_ber`, but using the Bussi-Donadio-Parrinello (bdp) method.

## nvt\_lan

- When the first parameter is `nvt_lan`, it is similar to the case of `nvt_ber`, but using the Langevin (lan) method as proposed in [G. Bussi and M. Parrinello, Phys. Rev. E 75, 056707 (2007)].

## nvt\_bao

- When the first parameter is `nvt_bao`, it is similar to the case of `nvt_ber`, but using the Langevin method with BAOAB splitting [Leimkuhler, Benedict, and Charles Matthews. Applied Mathematics Research eXpress 2013.1 (2013): 34-56].

## npt\_ber

- When the first parameter is `npt_ber`, it means that the ensemble for the current run is NPT (isothermal–isobaric) generated by using the Berendsen (ber) method. In this case, apart from the same parameters as in the case of `nvt_ber`, one needs to further specify some target pressure(s), the same number of estimated elastic moduli, and a pressure coupling constant  $p_{\text{coup}}$ . There are three ways of pressure controlling:

```
ensemble npt_ber T_1 T_2 T_coup p_hydro C_hydro p_coup # CONDITION-1
ensemble npt_ber T_1 T_2 T_coup p_xx p_yy p_zz C_xx C_yy C_zz p_coup # CONDITION-2
ensemble npt_ber T_1 T_2 T_coup p_xx p_yy p_zz p_xy p_xz p_yz C_xx C_yy C_zz C_xy C_xz C_yz p_coup # CONDITION-3 and
before GPUMD-v3.3.1
ensemble npt_ber T_1 T_2 T_coup p_xx p_yy p_zz p_yz p_xz p_xy C_xx C_yy C_zz C_yz C_xz C_xy p_coup # CONDITION-3 and
starting from GPUMD-v3.3.1
```

- CONDITION-1:** It means you regard your system as isotropic and want to control the three box lengths uniformly according to the hydrostatic pressure  $p_{\text{hydro}} = (p_{xx} + p_{yy} + p_{zz})/3$ . All the directions should have periodic boundary conditions. Currently, we require the box to be orthogonal, and we may relax this in a future version.

- **CONDITION-2:** The simulation box must be orthogonl in this case. The three box lengths will be controlled independently according to their target pressures. Any direction can be either periodic or nonperiodic and pressure controlling will only be effective in periodic directions.
- **CONDITION-3:** The simulation box must be triclinic and currently all the directions should be periodic. All the box components will be controlled independently according to the 6 target pressure components. **This is introduced in GPUMD-v3.0 and note that we have changed to use the Voigt notation starting from GPUMD-v3.3.1.**
- The elastic constant tensor  $C_{ab}$  is just an estimate and does not need to be accurate. It only needs to be of the correct order of magnitude. **These parameters are introduced in GPUMD-v3.0.**

## npt\_scr

- When the first parameter is npt\_scr, it is similar to the case of npt\_ber, but using the stochastic cell rescaling (scr) method. **This is introduced in GPUMD-v3.0.**

## heat\_nhc

- When the first parameter is heat\_nhc, it means heating a source region and simultaneously cooling a sink region using local Nose-Hoover chain thermostats. The full command is

```
ensemble heat_nhc T T_coup delta_T label_source label_sink
```

The target temperatures in the source region with label label\_source and the sink region with label label\_sink are  $T + \text{delta\_T}$  and  $T - \text{delta\_T}$ , respectively. Therefore, the temperature difference between the two regions is twice of delta\_T. In the command above, the parameter T\_coup has the same meaning as in the case of nvt\_nhc.

## heat\_bdp

- When the first parameter is heat\_bdp, it is similar to the case of heat\_nhc, but using the Bussi-Donadio-Parrinello (bdp) method.

## heat\_lan

- When the first parameter is heat\_lan, it is similar to the case of heat\_nhc, but using the Langevin (lan) method.

## Units and suggested parameters

- The units of temperature and pressure for this keyword are K and GPa, respectively.
- The temperature coupling constant  $\tau_{\text{coup}}$  means  $\tau_T / \Delta t$ , where  $\tau_T$  is the "relaxation time" of the thermostat and  $\Delta t$  is the time step for integration. We require  $\tau_T / \Delta t \geq 1$  and a good choice is  $\tau_T / \Delta t \approx 100$ . **Before GPUMD-v3.0,  $\tau_{\text{coup}}$  actually means the inverse  $\Delta t / \tau_T$  in the Berendsen thermostat.**
- The pressure coupling constant  $p_{\text{coup}}$  means  $\tau_p / \Delta t$ , where  $\tau_p$  is the "relaxation time" of the barostat and  $\Delta t$  is the time step for integration. We require  $\tau_p / \Delta t \geq 1$  and a good choice is  $\tau_p / \Delta t \approx 1000$ . **Before GPUMD-v3.0,  $p_{\text{coup}}$  actually means the inverse  $\Delta t / \tau_p$  in the Berendsen thermostat.**



- The elastic constant is in units of GPa. **This is introduced in GPUMD-v3.0. Before this version, the elastic constant is not treated as an input and is assumed to be about 160/3 GPa.**

## Examples

See the tutorials.

## Caveats

- One should use one and only one instance of this keyword for each run.

Retrieved from "[https://gpumd.zheyongfan.org/index.php?title=The\\_ensemble\\_keyword&oldid=22273](https://gpumd.zheyongfan.org/index.php?title=The_ensemble_keyword&oldid=22273)"

- 
- This page was last edited on 21 May 2022, at 04:22.

# The force constant potential

From GPUMD

[Jump to navigation](#)[Jump to search](#)

## Contents

- 1 Brief descriptions
- 2 Potential form
- 3 Potential files
  - 3.1 The driver potential file
  - 3.2 The force constant files
  - 3.3 The equilibrium position file
- 4 References

## Brief descriptions

- This is the force constant potential (FCP) implemented for Ref. [Brorsson 2021].
- When you use this potential, currently you need to add `-DUSE_FCP` in the makefile. In this case, you cannot use any other potential.
- Neighbor list is not used for this potential, but you still need to specify the `MN` and `r_c` parameters in `xyz.in`. You can choose any reasonable values for them and just remember that they will not be used for calculating the forces and related quantities.

## Potential form

In the force constant potential, the potential energy is calculated as a Taylor expansion in terms of the atomic displacements  $\{u_i^a\}$  relative to a set of reference (equilibrium) positions as

$$U = U_2 + U_3 + U_4 + U_5 + U_6 + \cdots;$$

$$U_2 = \frac{1}{2!} \sum_{ij} \sum_{ab} \Phi_{ij}^{ab} u_i^a u_j^b;$$

$$U_3 = \frac{1}{3!} \sum_{ijk} \sum_{abc} \Phi_{ijk}^{abc} u_i^a u_j^b u_k^c;$$

$$U_4 = \frac{1}{4!} \sum_{ijkl} \sum_{abcd} \Phi_{ijkl}^{abcd} u_i^a u_j^b u_k^c u_l^d;$$

$$U_5 = \frac{1}{5!} \sum_{ijklm} \sum_{abcde} \Phi_{ijklm}^{abcde} u_i^a u_j^b u_k^c u_l^d u_m^e;$$

$$U_6 = \frac{1}{6!} \sum_{ijklmn} \sum_{abcdef} \Phi_{ijklmn}^{abcdef} u_i^a u_j^b u_k^c u_l^d u_m^e u_n^f.$$

Here,  $\Phi_{ij}^{ab}$ ,  $\Phi_{ijk}^{abc}$ ,  $\Phi_{ijkl}^{abcd}$ ,  $\Phi_{ijklm}^{abcde}$ , and  $\Phi_{ijklmn}^{abcdef}$  are the second-order, third-order, fourth-order, fifth-order, and sixth-order force constants. The indices  $i, j, k, l, m$ , and  $n$  refer to the atoms and can take integer values from 0 to  $N - 1$ , where  $N$  is the number of atoms in the system. The indices  $a, b, c, d, e$ , and  $f$  refer to the axes in the Cartesian coordinate system and can take integer values 0, 1, and 2, which correspond to the  $x$ ,  $y$ , and  $z$  axes, respectively. In GPUMD, we only consider force constants up to the sixth order.

## Potential files

One needs to prepared quite a few files related to this potential, but they can be conveniently generated by hiPhive (<https://hiphive.materialsmodeling.org/>) [Eriksson 2019], except for the driver potential file below (which is very easy to prepare).

### The driver potential file

The driver potential file for this potential model reads

```
fcp number_of_atom_types
highest_force_order highest_heat_current_order
path_to_force_constant_files
```

- fcp is the name of this potential and tells the code that we are using the force constant potential.
- number\_of\_atom\_types is the number of atom types defined in the xyz.in file.
- highest\_force\_order is the highest order of the force constants used in the potential. For example, when highest\_order is 4, second-order, third-order, and fourth-order forces constants will be used (and should be prepared).
- highest\_heat\_current\_order is the highest order of the force constants used for heat current calculations. It can only be 2 or 3, which means using the second order force constants only or using both the second and third order force constants for heat current calculations, respectively.
- path\_to\_force\_constant\_files is the path to the force constant files (see below). **Important convention:** Write something like /path/to/your/folder instead of /path/to/your/folder/. That is, there should be no / after the folder name.

### The force constant files

The force constant data should be prepared in some files named as

```
clusters_order2.in
clusters_order3.in
clusters_order4.in
clusters_order5.in
clusters_order6.in
fcs_order2.in
fcs_order3.in
fcs_order4.in
fcs_order5.in
fcs_order6.in
```

These files should be in the folder you specified in the **driver** potential file (see above). If you only consider force constants up to the 4th order, you don't need the files with numbers 5 and 6. **These files can be generated by hiPhive (<https://hiphive.materialsmodeling.org/>)**. We therefore do not discuss the formats of these files.

## The equilibrium position file

Because this potential is defined in terms of the atom displacements, one has to define the equilibrium (reference) positions of the atoms in the system. A file called `r0.in` is used for this purpose. This file should be in the folder you specified in the **driver** potential file (see above). The format of this file is:

```
x_0 y_0 z_0
x_1 y_1 z_1
x_2 y_2 z_2
x_3 y_3 z_3
...
```

That is, each line gives the position of one atom. The order of the atoms should be consistent with that in the `xyz.in` file. The coordinates are in units of angstrom. **This file can be generated by hiPhive (<https://hiphive.materialsmodeling.org/>)**.

## References

- [Brorsson 2021] Joakim Brorsson, Arsalan Hashemi, Zheyong Fan, Erik Fransson, Fredrik Eriksson, Tapio Ala-Nissila, Arkady V. Krasheninnikov, Hannu-Pekka Komsa, Paul Erhart, *Efficient calculation of the lattice thermal conductivity by atomistic simulations with ab-initio accuracy* (<https://doi.org/10.1002/adts.202100217>), *Advanced Theory and Simulations*, **4**, 2100217 (2021).
- [Eriksson 2019] Fredrik Eriksson, Erik Fransson, and Paul Erhart, *The Hiphive Package for the Extraction of High-Order Force Constants by Machine Learning* (<https://doi.org/10.1002/adts.201800184>), *Advanced Theory and Simulations*, **2**, 1800184 (2019).

Retrieved from "[https://gpumd.zheyongfan.org/index.php?title=The\\_force\\_constant\\_potential&oldid=22025](https://gpumd.zheyongfan.org/index.php?title=The_force_constant_potential&oldid=22025)"

- 
- This page was last edited on 9 January 2022, at 09:36.

# The Lennard-Jones potential

From GPUMD

[Jump to navigation](#)[Jump to search](#)

## Contents

- 1 Brief descriptions
- 2 Potential form
- 3 Parameters
- 4 Potential file format

## Brief descriptions

- This is the Lennard-Jones (LJ) potential.
- It applies to systems with up to 10 atom types.
- The potential has not been shifted or damped.

## Potential form

- The LJ potential is one of the most simplest two-body potentials used in MD simulations. The pair potential between particles  $i$  and  $j$  is

$$U_{ij} = 4\epsilon_{ij} \left( \frac{\sigma_{ij}^{12}}{r_{ij}^{12}} - \frac{\sigma_{ij}^6}{r_{ij}^6} \right).$$

## Parameters

Parameter	Units
$\epsilon_{ij}$	eV
$\sigma_{ij}$	Å

## Potential file format

- If there is only one atom type, the potential file for this potential model reads

```
lj 1
epsilon sigma cutoff
```

Here, `cutoff` is the cutoff distance.

- If there are two atom types, the potential file reads

```
lj 2
epsilon_00 sigma_00 cutoff_00
epsilon_01 sigma_01 cutoff_01
```

```
epsilon_10 sigma_10 cutoff_10  
epsilon_11 sigma_11 cutoff_11
```

- If there are three atom types, the potential file reads

```
lj 3  
epsilon_00 sigma_00 cutoff_00  
epsilon_01 sigma_01 cutoff_01  
epsilon_02 sigma_02 cutoff_02  
epsilon_10 sigma_10 cutoff_10  
epsilon_11 sigma_11 cutoff_11  
epsilon_12 sigma_12 cutoff_12  
epsilon_20 sigma_20 cutoff_20  
epsilon_21 sigma_21 cutoff_21  
epsilon_22 sigma_22 cutoff_22
```

- I hope the reader can understand the pattern and figure out how to prepare potential files with more atom types.

Retrieved from "[https://gpumd.zheyongfan.org/index.php?title=The\\_Lennard-Jones\\_potential&oldid=21099](https://gpumd.zheyongfan.org/index.php?title=The_Lennard-Jones_potential&oldid=21099)"

- 
- This page was last edited on 19 August 2020, at 14:33.

# The neighbor keyword

From GPUMD

[Jump to navigation](#)[Jump to search](#)

## Contents

- 1 Purpose
- 2 Grammar
  - 2.1 Before GPUMD-v3.3
  - 2.2 Starting from GPUMD-v3.3
  - 2.3 Starting from GPUMD-v3.4
- 3 Examples before GPUMD-v3.3
  - 3.1 Example 1
  - 3.2 Example 2
- 4 Examples starting from GPUMD-v3.3
  - 4.1 Example 1
  - 4.2 Example 2
- 5 Caveats

## Purpose

This keyword is used to control the neighbor list update for a specific run.

## Grammar

### Before GPUMD-v3.3

This keyword only requires a single parameter, which is the skin distance (the difference between the cutoff distance used in neighbor list construction and that used in force evaluation) in units of angstrom. The full command reads

```
neighbor skin_distance
```

### Starting from GPUMD-v3.3

Starting from this version, the neighbor list will be updated by default, using a skin distance (the difference between the cutoff distance used in neighbor list construction and that used in force evaluation) of 1 angstrom for each run. One can use the following command to turn off the neighbor list update if one knows there is no need to update the neighbor list at all:

```
neighbor off
```

This only affects a single run.

## Starting from GPUMD-v3.4

This keyword will be removed starting from GPUMD-v3.4. If you want to use a fixed neighbor list, you can add the following to CFLAGS of the makefile:

```
-DUSE_FIXED_NEIGHBOR
```

This should only be used when you are sure that the system has no diffusion.

## Examples before GPUMD-v3.3

### Example 1

If one wants to update the neighbor list with a skin distance of 1 angstrom, one can add the following command before the run keyword

```
neighbor 1
```

### Example 2

If one does not want to update the neighbor list during one run and wants to update it during the next run, one can write something like this:

```
ensemble nvt_lan 300 300 100 # room temperature
# no neighbor list updating for this run
run 1000000

ensemble nvt_lan 300 3000 100 # increase the temperature
neighbor 1 # will update the neighbor list when needed
run 1000000
```

## Examples starting from GPUMD-v3.3

### Example 1

If one wants to update the neighbor list with a skin distance of 1 angstrom, one does not need to add anything, as this is the default.

### Example 2

If one does not want to update the neighbor list during one run and wants to update it during the next run, one can write something like this:

```
ensemble nvt_lan 300 300 100 # room temperature
neighbor off # no neighbor list updating for this run
run 1000000

ensemble nvt_lan 300 3000 100 # increase the temperature
```



```
# by default, the code will update the neighbor list when needed  
run 1000000
```

## Caveats

- The code will determine when the neighbor list needs to be updated. So the user does not need to specify an updating frequency.
- Using a larger skin distance will reduce the updating frequency, but will increase the computational cost (and memory requirement) for one updating and related calculations. We recommend using a skin distance of 1 angstrom.
- The user should decide wisely. For example, it is a waste of time to update the neighbor list when simulating a very stable solid. On the other hand, it is dangerous to choose to not update the neighbor list when it is needed.
- The atoms are only brought back to the simulation box before updating the neighbor list.
- The total number of neighbor list updates for each run will be printed on the screen.

Retrieved from "[https://gpumd.zheyongfan.org/index.php?title=The\\_neighbor\\_keyword&oldid=22316](https://gpumd.zheyongfan.org/index.php?title=The_neighbor_keyword&oldid=22316)"

- 
- This page was last edited on 24 July 2022, at 17:52.

# The potential keyword

From GPUMD  
Jump to navigationJump to search

## Contents

- 1 Purpose
- 2 Empirical potentials implemented in GPUMD
- 3 Grammar
- 4 Examples
  - 4.1 Silicon crystal
  - 4.2 Silicon Carbide
  - 4.3 MoS<sub>2</sub> + SiO<sub>2</sub>
  - 4.4 Carbon peapod
  - 4.5 Pure LJ systems
- 5 Caveats
- 6 References

## Purpose

- This page only applies to versions before **GPUMD-v3.4**.
- This keyword is used to specify one potential for the system or part of the system. For a simple system, a single potential keyword is sufficient, but for a complicated system, one might need to use multiple potential keywords.

## Empirical potentials implemented in GPUMD

- The individual potentials available in GPUMD are listed in the following table:

Potentials implemented in GPUMD

The Tersoff-1989 potential	The Tersoff-1988 potential
The Tersoff-mini potential	The embedded atom method (EAM) potential
The Stillinger-Weber potential	The Vashishta potential
The REBO-LJ potential for Mo-S systems	The Lennard-Jones potential
The Buckingham-Coulomb potential	The force constant potential

## Grammar

- This keyword needs at least one parameter, `potential_filename`, which is the filename (including relative or absolute path) of the potential file to be used.
- If the potential specified in `potential_filename` is **not** a Lennard-Jones (LJ) potential, the grammar of the potential keyword is

```
potential potential_filename [list_of_types]
```

Here, `list_of_types` gives the list of atom types that will participate in the potential.

- If the potential specified in `potential_filename` is an LJ potential, the grammar of the potential keyword can be one of the following two:

```
potential potential_filename
potential potential_filename grouping_method
```

Here, `grouping_method` is a grouping method used to exclude "intra-material" LJ interactions. The meaning of "intra-material" will be illustrated in the following examples.

- If your system can be completely described by one potential, just use one potential keyword.
- If your system consists of more than one part and each part can be described by one potential and you also want to add an LJ potential between the different parts, you can use one potential keyword for one part of your system and an extra potential keyword for the LJ potential.
- To fully understand the usage of this keyword, it is helpful to check the following examples.

## Examples

### Silicon crystal

For a silicon crystal system described by a Tersoff potential, the potential for this system can be specified by using a single potential keyword:

```
potential Si_Tersoff.txt 0
```

Here, `Si_Tersoff.txt` should be a potential file for the Tersoff potential and `0` means that the potential applies to atoms with type `0`. The Tersoff potential defined in `Si_Tersoff.txt` can be a single-element Tersoff potential, or a multi-element Tersoff potential. In both cases, type `0` defined in `Si_Tersoff.txt` should correspond to type `0` in `xyz.in`.

### Silicon Carbide

For a SiC crystal system described by a two-element Tersoff potential, the potential for this system can be specified by using a single potential keyword:

```
potential SiC_Tersoff.txt 0 1
```

Here, `SiC_Tersoff.txt` should be a potential file for the Tersoff potential and the numbers `0 1` mean that the potential applies to atoms with types `0` and `1`. The Tersoff potential defined in `SiC_Tersoff.txt` can be a two-element Tersoff potential, or a Tersoff potential with more elements. In both cases, types `0` and `1` defined in `SiC_Tersoff.txt` should correspond to types `0` and `1` in `xyz.in`.

### MoS<sub>2</sub> + SiO<sub>2</sub>

For a hybrid MoS<sub>2</sub> + SiO<sub>2</sub> system with LJ interactions between MoS<sub>2</sub> and SiO<sub>2</sub>, the potentials can be specified as follows:

```
potential SiO_Tersoff.txt 0 1
potential MoS_REBO.txt 2 3
potential Si_O_Mo_S_LJ.txt 1 # The number 1 here is not atom type, but grouping method
```

This means:

- The Si and O atoms have atom types 0 and 1 respectively. The Tersoff potential defined in `SiO_Tersoff.txt` will be applied to this material.
- The Mo and S atoms have atom types 2 and 3 respectively. The REBO potential defined in `MoS_REBO.txt` will be applied to this material. There is a type conversion here: types 2 and 3 will be converted to  $2-2=0$  and  $3-2=1$ , before applying the potential defined in `MoS_REBO.txt` to  $\text{MoS}_2$ .
- An LJ potential is defined in `Si_O_Mo_S_LJ.txt`. **It must be an LJ potential defined for all the 4 atom types.** The number 1 after `Si_O_Mo_S_LJ.txt` means that grouping method 1 (should be defined in `xyz.in`) will be used to exclude "intra-material" LJ interactions. In this grouping method, one can put Si and O atoms into group 0 and put Mo and S atoms into group 1. Then, LJ interactions will be excluded for Si-Si, Si-O, and O-O pairs in  $\text{SiO}_2$  and similarly for Mo-Mo, Mo-S, and S-S pairs in  $\text{MoS}_2$ .
- Of course, in this example, one can also simply set the LJ cutoff distance for these pairs to 0 in `Si_O_Mo_S_LJ.txt` without specifying a grouping method, but using a grouping method results in better performance (this is specific to the implementation of GPUMD and the user does not need to figure out why). While the use of a grouping method in this example is optional, it is mandatory in the next example.

## Carbon peapod

Carbon peapod is formed by encapsulating fullerene molecules (such as  $\text{C}_{60}$ ) into a carbon nanotube (CNT). We have recently studied thermal transport properties of this material in [Dong 2020], where we have considered 40  $\text{C}_{60}$  molecules. If we want to apply a Tersoff potential to the covalent C-C bonds and an LJ potential to the C-C pairs in which the two atoms are not both from the CNT and not from the same  $\text{C}_{60}$  molecule, we can use the following two commands:

```
potential C_Tersoff.txt 0
potential C_LJ.txt 1 # The number 1 here is not atom type, but grouping method
```

This means:

- All the carbon atoms are of type 0 in `xyz.in` and the Tersoff potential defined in `C_Tersoff.txt` will be applied to all the atoms. Note that due to the short cutoff distance (2.1 Å) of the Tersoff potential, there will be no Tersoff interactions between the CNT and any  $\text{C}_{60}$  molecule or between two  $\text{C}_{60}$  molecules.
- An LJ potential is defined in `C_LJ.txt`. The name of this file suggests that there is a single entry, which is for the C-C pairs.
- The number 1 after `C_LJ.txt` means that grouping method 1 will be used to exclude "intra-material" LJ interactions based on grouping method 1 (should be defined in `xyz.in`). In this grouping method, atoms in the CNT should be in one group and atoms in each  $\text{C}_{60}$  molecule should also be in its own group. That is, there should be 41 groups in this grouping method. Based on this grouping method, two carbon atoms from the same "material" (the CNT or any  $\text{C}_{60}$  molecule) will have no LJ interactions.

## Pure LJ systems

The user must have realized that the LJ potential is kind of special in GPUMD. This is true and we find it beneficial to treat it as a special potential. Our special rules for the LJ potential are:

- One does not need (and cannot) specify atom type(s) for this potential. **It is always assumed that the LJ potential applies to all the atom types defined in xyz.in.**
- Related to the above rule, **the LJ potential cannot be specified more than once in run.in.**

Base on the above rules, the potential for an argon system could be specified as

```
potential Ar_LJ.txt
```

If one writes

```
potential Ar_LJ.txt 0
```

GPUMD will misunderstand you by assuming that you want to exclude "intra-material" LJ interactions based on grouping method 0.

Similarly, the potential for an argon-krypton system could be specified as

```
potential Ar_Kr_LJ.txt
```

If one writes

```
potential Ar_Kr_LJ.txt 0 1
```

GPUMD will response more cleverly by reporting an error message complaining that there are too many parameters for the potential keyword.

## Caveats

- It is very important to make sure that the atom types in the xyz.in file are consistent with the potential files.

## References

- [Dong 2020] Haikuan Dong, Zheyong Fan, Ping Qian, Tapio Ala-Nissila, and Yanjing Su, *Thermal conductivity reduction in carbon nanotube by fullerene encapsulation: A molecular dynamics study* (<https://doi.org/10.1016/j.carbon.2020.01.114>), Carbon **161**, 800-808 (2020).

Retrieved from "[https://gpumd.zheyongfan.org/index.php?title=The\\_potential\\_keyword&oldid=22435](https://gpumd.zheyongfan.org/index.php?title=The_potential_keyword&oldid=22435)"

- 
- This page was last edited on 2 September 2022, at 14:36.

# The REBO-LJ potential for Mo-S systems

From GPUMD

[Jump to navigation](#)[Jump to search](#)

## Contents

- 1 Brief descriptions
- 2 Potential form
- 3 Parameters
- 4 Potential file format
- 5 References

## Brief descriptions

- This is the REBO-LJ potential corresponding to [Liang 2009] and [Stewart 2013].
- It only applies to Mo-S systems.
- This potential will be removed starting from GPUMD-v3.4.

## Potential form

See [Liang 2009] and [Stewart 2013].

## Parameters

See [Liang 2009] and [Stewart 2013].

## Potential file format

The potential file for this potential model reads

```
rebo_mos2 2
# Nothing here, because this is a special potential model and
# the parameters are thus hard coded
```

- When preparing the xyz.in file, one has to define Mo atoms as type 0 and S atoms as type 1.

## References

- [Liang 2009] T. Liang, S. R. Phillpot, and S. B. Sinnott, *Parametrization of a reactive many-body potential for Mo–S systems* (<https://doi.org/10.1103/PhysRevB.79.245110>), Phys. Rev. B **79**, 245110 (2009).
- [Stewart 2013] J. A. Stewart and D. E. Spearot, *Atomistic simulations of nanoindentation on the basal plane of crystalline molybdenum disulfide (MoS<sub>2</sub>)* (<https://doi.org/10.1088/0965-0393/21/4/045003>), Modelling and Simulation in Materials Science and Engineering **21**, 045003 (2013).

Retrieved from "[https://gpumd.zheyongfan.org/index.php?title=The\\_REBO-LJ\\_potential\\_for\\_Mo-](https://gpumd.zheyongfan.org/index.php?title=The_REBO-LJ_potential_for_Mo-)

S\_systems&oldid=22313"

---

- This page was last edited on 24 July 2022, at 17:47.

# The restart.out output file

From GPUMD

[Jump to navigation](#)[Jump to search](#)

## Contents

- 1 Brief Description
- 2 Version
- 3 The keyword which produces the current file
- 4 File format
- 5 Tips

## Brief Description

This is the restart file for `src/gpumd`.

## Version

- This is only used before GPUMD-v3.4.

## The keyword which produces the current file

- `dump_restart`

## File format

- This file has the same format as the `xyz.in` file.

## Tips

- The output mode for this file is **overwrite**.
- By changing the file name to `xyz.in`, it can be used for the next simulation.

Retrieved from "[https://gpumd.zheyongfan.org/index.php?title=The\\_restart.out\\_output\\_file&oldid=22410](https://gpumd.zheyongfan.org/index.php?title=The_restart.out_output_file&oldid=22410)"

- 
- This page was last edited on 28 August 2022, at 08:24.



# The Stillinger-Weber potential

From GPUMD

[Jump to navigation](#)[Jump to search](#)

## Contents

- 1 Brief descriptions
- 2 Potential form
- 3 Parameters
- 4 Potential file
  - 4.1 Single-element system
  - 4.2 Double-element system
  - 4.3 Three-element system
- 5 References

## Brief descriptions

- This is the Stillinger-Weber (SW) potential corresponding to [Stillinger 1985].
- It applies to systems with one, two, or three atom types.
- This potential will be removed starting from GPUMD-v3.4.

## Potential form

The total potential energy consists of a two-body part and a three-body part. The site potential is

$$U_i = \frac{1}{2} V_2(r_{ij}) + \frac{1}{2} \sum_{j \neq i} \sum_{k \neq i, j} h_{ijk},$$

where the two-body part is

$$V_2(r_{ij}) = \epsilon A \left[ B \left( \frac{\sigma}{r_{ij}} \right)^4 - 1 \right] \exp \left( \frac{1}{r_{ij}/\sigma - a} \right)$$

and the three-body part is

$$h_{ijk} = \epsilon \lambda \exp \left[ \frac{\gamma}{r_{ij}/\sigma - a} + \frac{\gamma}{r_{ik}/\sigma - a} \right] (\cos \theta_{ijk} - h)^2.$$

## Parameters

Parameter	Units
$A$	eV
$B$	dimensionless
$\epsilon$	dimensionless
$\sigma$	Å
$a$	dimensionless
$\lambda$	eV
$\gamma$	dimensionless
$h$	dimensionless

## Potential file

### Single-element system

- For one-element systems, the potential file reads

```
sw_1985 1
epsilon lambda A B a gamma sigma h
```

### Double-element system

- For two-element systems, the two-body parameters can generally have 3 different values and the three-body parameters can generally have 8 different values. The potential file reads

```
sw_1985 2
A_00 B_00 a_00 sigma_00 gamma_00
A_01 B_01 a_01 sigma_01 gamma_01
A_11 B_11 a_11 sigma_11 gamma_11
lambda_000 cos0_000
lambda_001 cos0_001
lambda_010 cos0_010
lambda_011 cos0_011
lambda_100 cos0_100
lambda_101 cos0_101
lambda_110 cos0_110
lambda_111 cos0_111
```

- Note that we have removed the redundant parameter  $\epsilon$  in the potential file here and  $A$  and  $\lambda$  actually stand for  $\epsilon A$  and  $\epsilon \lambda$ .

### Three-element system

- If there are three atom types, the potential file reads:

```

sw_1985 3
A_00 B_00 a_00 sigma_00 gamma_00
A_01 B_01 a_01 sigma_01 gamma_01
A_02 B_02 a_02 sigma_02 gamma_02
A_10 B_10 a_10 sigma_10 gamma_10
A_11 B_11 a_11 sigma_11 gamma_11
A_12 B_12 a_12 sigma_12 gamma_12
A_20 B_20 a_20 sigma_20 gamma_20
A_21 B_21 a_21 sigma_21 gamma_21
A_22 B_22 a_22 sigma_22 gamma_22
lambda_000 cos0_000
lambda_001 cos0_001
lambda_002 cos0_002
lambda_010 cos0_010
lambda_011 cos0_011
lambda_012 cos0_012
lambda_020 cos0_020
lambda_021 cos0_021
lambda_022 cos0_022
lambda_100 cos0_100
lambda_101 cos0_101
lambda_102 cos0_102
lambda_110 cos0_110
lambda_111 cos0_111
lambda_112 cos0_112
lambda_120 cos0_120
lambda_121 cos0_121
lambda_122 cos0_122
lambda_200 cos0_200
lambda_201 cos0_201
lambda_202 cos0_202
lambda_210 cos0_210
lambda_211 cos0_211
lambda_212 cos0_212
lambda_220 cos0_220
lambda_221 cos0_221
lambda_222 cos0_222

```

- Do we need to consider more than three atom types? I don't think so. So stop here.

## References

- [Stillinger 1985] Frank H. Stillinger and Thomas A. Weber, *Computer simulation of local order in condensed phases of silicon* (<https://doi.org/10.1103/PhysRevB.31.5262>), Phys. Rev. B **31**, 5262 (1985).

Retrieved from "[https://gpumd.zheyongfan.org/index.php?title=The\\_Stillinger-Weber\\_potential&oldid=22314](https://gpumd.zheyongfan.org/index.php?title=The_Stillinger-Weber_potential&oldid=22314)"

- This page was last edited on 24 July 2022, at 17:47.

# The Tersoff-1988 potential

From GPUMD

[Jump to navigation](#)[Jump to search](#)

## Contents

- 1 Brief descriptions
- 2 Potential form
- 3 Parameters
- 4 Potential file format
- 5 References

## Brief descriptions

- This is the Tersoff-1988 potential corresponding to [Tersoff 1988].
- It is made to mimic the Tersoff potential in LAMMPS ([https://lammps.sandia.gov/doc/pair\\_tersoff.html](https://lammps.sandia.gov/doc/pair_tersoff.html)).
- The Tersoff-1988 potential is more general than the Tersoff-1989 potential, but when the Tersoff-1989 potential applies, it is better to use the Tersoff-1989 potential, because it is faster.

## Potential form

- The site potential can be written as

$$U_i = \frac{1}{2} \sum_{j \neq i} f_C(r_{ij}) [f_R(r_{ij}) - b_{ij} f_A(r_{ij})].$$

- The function  $f_C$  is a cutoff function, which is 1 when  $r_{ij} < R$  and 0 when  $r_{ij} > S$  and takes the following form in the intermediate region:

$$f_C(r_{ij}) = \frac{1}{2} \left[ 1 + \cos \left( \pi \frac{r_{ij} - R}{S - R} \right) \right].$$

- The repulsive function  $f_R$  and the attractive function  $f_A$  take the following forms:

$$f_R(r) = A e^{-\lambda r_{ij}};$$

$$f_A(r) = B e^{-\mu r_{ij}}.$$

- The bond-order is

$$b_{ij} = \left( 1 + \beta^n \zeta_{ij}^n \right)^{-\frac{1}{2n}},$$

where

$$\zeta_{ij} = \sum_{k \neq i,j} f_C(r_{ik}) g_{ijk} e^{\alpha(r_{ij}-r_{ik})^m};$$

$$g_{ijk} = \gamma \left( 1 + \frac{c^2}{d^2} - \frac{c^2}{d^2 + (h - \cos \theta_{ijk})^2} \right).$$

## Parameters

Parameter	Units
$A$	eV
$B$	eV
$\lambda$	$\text{\AA}^{-1}$
$\mu$	$\text{\AA}^{-1}$
$\beta$	dimensionless
$n$	dimensionless
$c$	dimensionless
$d$	dimensionless
$h$	dimensionless
$R$	$\text{\AA}$
$S$	$\text{\AA}$
$m$	dimensionless
$\alpha$	$\text{\AA}^{-m}$
$\gamma$	dimensionless

## Potential file format

- We have adopted a file format similar (but not identical) to that used by LAMMPS [1].
- The potential file for a single-element system reads:

```
tersoff_1988 1
A_000 B_000 lambda_000 mu_000 beta_000 n_000 c_000 d_000 h_000 R_000 S_000 m_000 alpha_000 gamma_000
```

- The potential file for a double-element system reads:

```
tersoff_1988 2
A_000 B_000 lambda_000 mu_000 beta_000 n_000 c_000 d_000 h_000 R_000 S_000 m_000 alpha_000 gamma_000
A_001 B_001 lambda_001 mu_001 beta_001 n_001 c_001 d_001 h_001 R_001 S_001 m_001 alpha_001 gamma_001
A_010 B_010 lambda_010 mu_010 beta_010 n_010 c_010 d_010 h_010 R_010 S_010 m_010 alpha_010 gamma_010
A_011 B_011 lambda_011 mu_011 beta_011 n_011 c_011 d_011 h_011 R_011 S_011 m_011 alpha_011 gamma_011
A_100 B_100 lambda_100 mu_100 beta_100 n_100 c_100 d_100 h_100 R_100 S_100 m_100 alpha_100 gamma_100
A_101 B_101 lambda_101 mu_101 beta_101 n_101 c_101 d_101 h_101 R_101 S_101 m_101 alpha_101 gamma_101
A_110 B_110 lambda_110 mu_110 beta_110 n_110 c_110 d_110 h_110 R_110 S_110 m_110 alpha_110 gamma_110
A_111 B_111 lambda_111 mu_111 beta_111 n_111 c_111 d_111 h_111 R_111 S_111 m_111 alpha_111 gamma_111
```

- Can you guess the file format for a triple-element system?

## References

- [Tersoff 1988] J. Tersoff, *New empirical approach for the structure and energy of covalent systems* (<https://doi.org/10.1103/PhysRevB.37.6991>), Phys. Rev. B **37**, 6991 (1988).

Retrieved from "[https://gpumd.zheyongfan.org/index.php?title=The\\_Tersoff-1988\\_potential&oldid=21265](https://gpumd.zheyongfan.org/index.php?title=The_Tersoff-1988_potential&oldid=21265)"

- 
- This page was last edited on 22 August 2020, at 17:29.

# The Tersoff-1989 potential

From GPUMD

[Jump to navigation](#)[Jump to search](#)

## Contents

- 1 Brief descriptions
- 2 Potential form
- 3 Parameters
- 4 Potential file format
  - 4.1 Tersoff-1989 potential for single-element systems
  - 4.2 Tersoff-1989 potential for double-element systems
- 5 References

## Brief descriptions

- This is the Tersoff-1989 potential corresponding to [Tersoff 1989].
- It only applies to systems with one or two atom types. For systems with more atom types, one needs to use the Tersoff-1988 potential.
- Even for systems with one or two atom types, the Tersoff-1989 potential is less general than the Tersoff-1988 potential, but the Tersoff-1989 potential is faster.

## Potential form

- Conventions:
  - Use  $i, j, k, \dots$  for atom **indices**.
  - Use  $I, J, K, \dots$  for atom **types**.
- The **site potential** can be written as

$$U_i = \frac{1}{2} \sum_{j \neq i} f_C(r_{ij}) [f_R(r_{ij}) - b_{ij} f_A(r_{ij})].$$

- The function  $f_C$  is a **cutoff function**, which is 1 when  $r_{ij} < R_{IJ}$  and 0 when  $r_{ij} > S_{IJ}$  and takes the following form in the intermediate region:

$$f_C(r_{ij}) = \frac{1}{2} \left[ 1 + \cos \left( \pi \frac{r_{ij} - R_{IJ}}{S_{IJ} - R_{IJ}} \right) \right].$$

- The **repulsive function**  $f_R$  and the **attractive function**  $f_A$  take the following forms:

$$f_R(r) = A_{IJ} e^{-\lambda_{IJ} r_{ij}};$$

$$f_A(r) = B_{IJ} e^{-\mu_{IJ} r_{ij}}.$$

- The **bond-order function** is

$$b_{ij} = \chi_{IJ} \left( 1 + \beta_I^{n_I} \zeta_{ij}^{n_I} \right)^{-\frac{1}{2n_I}},$$

where

$$\zeta_{ij} = \sum_{k \neq i, j} f_C(r_{ik}) g_{ijk};$$

$$g_{ijk} = 1 + \frac{c_I^2}{d_I^2} - \frac{c_I^2}{d_I^2 + (h_I - \cos \theta_{ijk})^2}.$$

## Parameters

Parameter	Units
$A_{IJ}$	eV
$B_{IJ}$	eV
$\lambda_{IJ}$	$\text{\AA}^{-1}$
$\mu_{IJ}$	$\text{\AA}^{-1}$
$\beta_I$	dimensionless
$n_I$	dimensionless
$c_I$	dimensionless
$d_I$	dimensionless
$h_I$	dimensionless
$R_{IJ}$	$\text{\AA}$
$S_{IJ}$	$\text{\AA}$
$\chi_{IJ}$	dimensionless

## Potential file format

### Tersoff-1989 potential for single-element systems

- In this case,  $\chi_{IJ}$  is irrelevant. The potential file reads

```
tersoff_1989 1
A B lambda mu beta n c d h R S
```

### Tersoff-1989 potential for double-element systems

- In this case, there are two sets of parameters, one for each atom type. The following mixing rules are used to determine some parameters between the two atom types  $i$  and  $j$ :

$$A_{IJ} = \sqrt{A_{II} A_{JJ}};$$



$$B_{IJ} = \sqrt{B_{II}B_{JJ}};$$

$$R_{IJ} = \sqrt{R_{II}R_{JJ}};$$

$$S_{IJ} = \sqrt{S_{II}S_{JJ}};$$

$$\lambda_{IJ} = (\lambda_{II} + \lambda_{JJ})/2;$$

$$\mu_{IJ} = (\mu_{II} + \mu_{JJ})/2.$$

- Here, the parameter  $\chi_{01} = \chi_{10}$  needs to be provided.  $\chi_{00} = \chi_{11} = 1$  by definition.
- The potential file reads

```
tersoff_1989 2
A_0 B_0 lambda_0 mu_0 beta_0 n_0 c_0 d_0 h_0 R_0 S_0
A_1 B_1 lambda_1 mu_1 beta_1 n_1 c_1 d_1 h_1 R_1 S_1
chi_01
```

## References

- [Tersoff 1989] J. Tersoff, *Modeling solid-state chemistry: Interatomic potentials for multicomponent systems* (<https://doi.org/10.1103/PhysRevB.39.5566>), Phys. Rev. B **39**, 5566(R) (1989).

Retrieved from "[https://gpumd.zheyongfan.org/index.php?title=The\\_Tersoff-1989\\_potential&oldid=21260](https://gpumd.zheyongfan.org/index.php?title=The_Tersoff-1989_potential&oldid=21260)"

- 
- This page was last edited on 22 August 2020, at 17:27.

# The Tersoff-mini potential

From GPUMD

[Jump to navigation](#)[Jump to search](#)

## Contents

- 1 Brief descriptions
- 2 Potential form
- 3 Parameters
- 4 Potential file format
  - 4.1 Single-element systems
- 5 References

## Brief descriptions

- This is the Tersoff-mini potential corresponding to [Fan 2020].
- It currently only applies to systems with a single atom type.
- One can use the GPUGA package (<https://github.com/brucefan1983/GPUGA>) to fit this potential for new systems.

## Potential form

- The **site potential** can be written as

$$U_i = \frac{1}{2} \sum_{j \neq i} f_C(r_{ij}) [f_R(r_{ij}) - b_{ij} f_A(r_{ij})] .$$

- The function  $f_C$  is a **cutoff function**, which is 1 when  $r_{ij} < R_{IJ}$  and 0 when  $r_{ij} > S_{IJ}$  and takes the following form in the intermediate region:

$$f_C(r_{ij}) = \frac{1}{2} \left[ 1 + \cos \left( \pi \frac{r_{ij} - R}{S - R} \right) \right] .$$

- The **repulsive function**  $f_R$  and the **attractive function**  $f_A$  take the following forms:

$$f_R(r_{ij}) = \frac{D_0}{S - 1} \exp(\alpha r_0 \sqrt{2S}) e^{-\alpha \sqrt{2S} r_{ij}} ;$$

$$f_A(r_{ij}) = \frac{D_0 S}{S - 1} \exp(\alpha r_0 \sqrt{2/S}) e^{-\alpha \sqrt{2/S} r_{ij}} .$$

- The **bond-order function** is

$$b_{ij} = \left( 1 + \zeta_{ij}^n \right)^{-\frac{1}{2n}} ,$$

where

$$\zeta_{ij} = \sum_{k \neq i,j} f_C(r_{ik}) g_{ijk};$$

$$g_{ijk} = \beta (h - \cos \theta_{ijk})^2.$$

## Parameters

Parameter	Units
$D_0$	eV
$\alpha$	$\text{\AA}^{-1}$
$r_0$	$\text{\AA}$
$S$	dimensionless
$n$	dimensionless
$\beta$	dimensionless
$h$	dimensionless
$R$	$\text{\AA}$
$S$	$\text{\AA}$

## Potential file format

### Single-element systems

- The potential file reads

```
tersoff_mini 1
D alpha r0 S beta n h R S
```

## References

- [Fan 2020] Zheyong Fan, Yanzhou Wang, Xiaokun Gu, Ping Qian, Yanjing Su, and Tapio Ala-Nissila, *A minimal Tersoff potential for diamond silicon with improved descriptions of elastic and phonon transport properties* (<https://doi.org/10.1088/1361-648X/ab5c5f>), J. Phys.: Condens. Matter **32**, 135901 (2020).

Retrieved from "[https://gpumd.zheyongfan.org/index.php?title=The\\_Tersoff-mini\\_potential&oldid=21276](https://gpumd.zheyongfan.org/index.php?title=The_Tersoff-mini_potential&oldid=21276)"

- This page was last edited on 22 August 2020, at 17:34.

# The Vashishta potential

From GPUMD

[Jump to navigation](#)[Jump to search](#)

## Contents

- 1 Brief descriptions
- 2 Potential form
- 3 Parameters
- 4 Potential file format
- 5 References

## Brief descriptions

- This is the Vashishta potential corresponding to [Vashishta 2007].
- It only applies to systems with two atom types.
- This potential will be removed starting from GPUMD-v3.4.

## Potential form

- The Vashishta potential is essentially a pairwise potential plus a modified form of the three-body part of the Stillinger-Weber potential. Therefore, the site potential can be written in the same form as the Stillinger-Weber potential:

$$U_i = \frac{1}{2} V_2(r_{ij}) + \frac{1}{2} \sum_{j \neq i} \sum_{k \neq i, j} h_{ijk}.$$

- The two-body part reads

$$V_2(r_{ij}) = \frac{H}{r_{ij}^\eta} + \frac{1}{4\pi\epsilon_0} \frac{q_i q_j}{r_{ij}} e^{-r_{ij}/\lambda} - \frac{1}{4\pi\epsilon_0} \frac{D}{2r_{ij}^4} e^{-r_{ij}/\xi} - \frac{W}{r_{ij}^6}.$$

The four terms on the right hand side of the above equation correspond to steric size effects, charge-charge interactions, charge-dipole interactions, and dipole-dipole interactions, respectively. The original paper has used Gauss units for the middle two terms and we have used the SI units.

- The two-body part is shifted in terms of both potential and force:

$$V_2^{\text{shifted}}(r_{ij}) = V_2(r_{ij}) - V_2(r_c) - (r - r_c) \frac{dV_2(r_{ij})}{dr_{ij}} \Big|_{r=r_c}.$$

Therefore, both the potential and the force for the two-body part are continuous at the cutoff distance  $r_c$ .

- The three-body part is

$$h_{ijk} = B \exp \left[ \frac{\gamma}{r_{ij} - r_0} + \frac{\gamma}{r_{ik} - r_0} \right] \frac{(\cos \theta_{ijk} - h)^2}{1 + C(\cos \theta_{ijk} - h)^2}.$$

The parameter  $\gamma$  is always 1 Å and is thus redundant.

Parameters

Parameter	Units
$B$	eV
$h$	dimensionless
$C$	dimensionless
$r_0$	Å
$r_c$	Å
$H$	eV Å <sup><math>\eta</math></sup>
$\eta$	dimensionless
$q$	e
$\lambda$	Å
$D$	e <sup>2</sup> Å <sup>3</sup>
$\xi$	Å
$W$	eV Å <sup>6</sup>

Potential file format

- The potential file for this potential model reads

```
vashishta 2
B_0 B_1 h_0 h_1 C r0 rc
H_00 eta_00 q0*q0 lambda_00 D_00 xi_00 W_00
H_01 eta_01 q0*q1 lambda_01 D_01 xi_01 W_01
H_11 eta_11 q1*q1 lambda_11 D_11 xi_11 W_11
```

- The parameter  $\eta$  should be entered as an integer in the potential file.

References

- [1] Priya Vashishta, Rajiv K. Kalia, and Aiichiro Nakano, *Interaction potential for silicon carbide: A molecular dynamics study of elastic constants and vibrational density of states for crystalline and amorphous silicon carbide* (<https://doi.org/10.1063/1.2724570>), J. Appl. Phys. **101**, 103515 (2007).

Retrieved from "[https://gpumd.zheyongfan.org/index.php?title=The\\_Vashishta\\_potential&oldid=22311](https://gpumd.zheyongfan.org/index.php?title=The_Vashishta_potential&oldid=22311)"

- This page was last edited on 24 July 2022, at 17:46.

# The xyz.in input file

From GPUMD

[Jump to navigation](#)[Jump to search](#)

## Contents

- 1 Brief descriptions
- 2 File format
  - 2.1 Format A
  - 2.2 Format B
  - 2.3 Line 0 (we prefer to **count from 0**)
  - 2.4 Line 1
  - 2.5 Line 2
  - 2.6 The remaining lines
- 3 Units
- 4 An example
- 5 Tips

## Brief descriptions

- This file defines the **simulation model**, such as the **initial conditions** and **boundary conditions**.

## File format

- This file should have one of the following formats (empty lines and comments are **not** allowed):

### Format A

```
N M cutoff 0 has_velocity number_of_grouping_methods
pbc_x pbc_y pbc_z L_x L_y L_z
t0 x0 y0 z0 m0 [vx0] [vy0] [vz0] [group0_0] [group1_0] [group2_0]...
t1 x1 y1 z1 m1 [vx1] [vy1] [vz1] [group0_1] [group1_1] [group2_1]...
t2 x2 y2 z2 m2 [vx2] [vy2] [vz2] [group0_2] [group1_2] [group2_2]...
...
```

### Format B

```
N M cutoff 1 has_velocity number_of_grouping_methods
pbc_a pbc_b pbc_c a_x a_y a_z b_x b_y b_z c_x c_y c_z
t0 x0 y0 z0 m0 [vx0] [vy0] [vz0] [group0_0] [group1_0] [group2_0]...
t1 x1 y1 z1 m1 [vx1] [vy1] [vz1] [group0_1] [group1_1] [group2_1]...
t2 x2 y2 z2 m2 [vx2] [vy2] [vz2] [group0_2] [group1_2] [group2_2]...
...
```

## Line 0 (we prefer to count from 0)

- **N** is the **number of atoms**.
- **M** is the **maximum number of neighbors** any atom can ever has. It is the responsibility of the user to use a large enough value of **M**. If some atom can have more than **M** neighbors during the simulation, unexpected error may occur and there may be no useful error message. If **M** is too large, it just wastes

some memory. For programmers: the Verlet neighbor list will be allocated a memory chunk of size  $\text{sizeof(int)} * N * M$ . If you really don't know how to choose the value of  $M$ , you can simply set it to the maximal allowed value: 1024. **This item will be removed starting from GPUMD-v3.4.**

- cutoff is the **initial cutoff distance used for building the neighbor list**. If you really don't know how to choose the value of cutoff, you can simply set it to cutoff distance of the maximal force cutoff in your system plus 1.0 Å. You may want to have a look at the neighbor keyword. **This item will be removed starting from GPUMD-v3.4.**
- The next number can be either 0 (**Format A**) or 1 (**Format B**). If it is 0, the simulation box will be **orthogonal**; if it is 1, the simulation box will be **triclinic**.
- has\_velocity can be 1 or 0, which means this file contains or does not contain the **initial velocities**, respectively.
- number\_of\_grouping\_methods can be 0 or any positive integer, which is the **number of grouping methods** that need to be defined in this file.

## Line 1

- If the box is **orthogonal**, the first three items, pbc\_x, pbc\_y, and pbc\_z, can only be 1 or 0. If pbc\_x is 1, it means that periodic boundary conditions will be applied to the x direction; if pbc\_x is 0, it means that free boundary conditions will be applied to the x direction. Similar descriptions apply to the other two directions. The next three items, L\_x, L\_y, and L\_z, are the initial lengths of the (rectangular) simulation box along the *x*, *y*, and *z* directions, respectively.
- If the box is **triclinic**, the first three numbers correspond to pbc\_a, pbc\_b, and pbc\_c, which have similar meanings as pbc\_x, pbc\_y, and pbc\_z, but for the ***a***, ***b***, and ***c*** directions as specified by the remaining nine numbers in this line:

$$\mathbf{a} = a_x \mathbf{e}_x + a_y \mathbf{e}_y + a_z \mathbf{e}_z;$$

$$\mathbf{b} = b_x \mathbf{e}_x + b_y \mathbf{e}_y + b_z \mathbf{e}_z;$$

$$\mathbf{c} = c_x \mathbf{e}_x + c_y \mathbf{e}_y + c_z \mathbf{e}_z;$$

These vectors define the simulation box and there is no restriction on the values of the vector components, but the user is responsible for the compatibility between the initial atom coordinates and the simulation box.

- About periodic boundary conditions:
  - We use the **minimum-image convention** to account for the periodic boundary conditions for **non-NEP potentials**. The user needs to make sure that the box size is large enough (the thickness in each direction is larger than twice of the force cutoff) to incorporate enough neighbors.
  - However, starting from **GPUMD-v3.0**, for the **NEP potential**, we do not use the **minimum-image convention** and there is no requirement for the box size. That said, we only recommend to use small boxes for the purpose of doing active learning. Otherwise, we recommend to use a relatively large box.

## Line 2

- t0 is the type of atom 0. The atom type will be used to determine which potential parameters to use. **In GPUMD-v2.6, for the NEP machine learning potential**, the atom type should be the **atomic number** (number of protons) of the element; for other potentials, **we use integers to record the atom types and the indices start from 0**. In GPUMD-v2.7, we also use integers (0, 1, 2, ...) for the NEP potential. In GPUMD-2.8, one needs to use atom symbols (such as H, He, Li) when the code is compiled to use NEP potentials.
- x0, y0, and z0 are the coordinates of this atom.
- m0 is the mass of this atom.
- if has\_velocity is 1, the next 3 items are the initial velocity components of this atom.
- if number\_of\_grouping\_methods is larger than zero, the next items are the group labels of this atom in these grouping methods. That is, group0\_0 specifies which group this atom belongs to using grouping

method 0; group1\_0 specifies which group this atom belongs to using the grouping method 1; group2\_0 specifies which group this atom belongs to using the grouping method 2. **Both grouping methods and group labels within a given grouping method are represented as integers and start from 0.** Some keywords in the run.in file will only use grouping method 0 and there is no need (or no way) to choose a grouping method for these keywords.

## The remaining lines

- Similarly, the  $(m + 2)$ th line gives the information for the  $m$ th atom. This file should have  $(N+2)$  lines.

## Units

- The mass should be given in units of the unified **atomic mass unit** (amu).
- The cutoff distance, box lengths and atom coordinates should be given in units of **angstrom**.
- Velocities should be in the natural units adopted in GPUMD, i.e.,  $\text{eV}^{1/2} \text{amu}^{-1/2}$ . This is usually guaranteed, because one will probably use the dump\_restart keyword to produce an xyz.in file which contains the velocity data. Otherwise, one would probably not prepare velocity data in the xyz.in file.

## An example

Assume we have the following xyz.in file:

```
10 2 1.5 0 0 3
1 0 0 4 1 1
0 0 0 0 1 0 0 0
1 1 0 0 1 0 1 0
0 2 0 0 1 0 2 0
1 3 0 0 1 0 3 0
0 4 0 0 1 0 4 0
1 5 0 0 1 1 5 0
0 6 0 0 1 1 6 0
1 7 0 0 1 1 7 0
0 8 0 0 1 1 8 0
1 9 0 0 1 1 9 0
```

It means:

- There are 10 atoms (or particles)
- The maximum number of neighbors is set to 2, which is acceptable if we are sure that one atom can have at most 2 neighbors.
- The initial cutoff distance for building the neighbor list is 1.5 Å.
- The simulation box is orthogonal.
- There are no velocity data in this file.
- There are 3 grouping methods defined in this file (see below).
- Use periodic boundary conditions in the  $x$  direction and free boundary conditions in the other directions.
- The box lengths in the three directions are respectively 4 Å, 1 Å, and 1 Å.
- Atoms 0, 2, 4, 6, and 8 are of type 0 and atoms 1, 3, 5, 7, and 9 are of type 1.
- The 10 atoms are located along a line in the  $x$  direction with equal spacing, from 0 Å to 9 Å.
- All the 10 atoms have mass 1.
- In grouping method 0, atom 0 to atom 4 have group label 0 (which means they are in group 0), and atom 5 to atom 9 have group label 1 (which means they are in group 1).
- In grouping method 1, atom  $m$  ( $0 \leq m \leq 9$ ) has group label  $m$ . That is, each group consists of a single atom.



- In grouping method 2, all the atoms have group label 0. That is, all the atoms are in the same group.

## Tips

- A valid xyz.in file is a valid XYZ file ([https://en.wikipedia.org/wiki/XYZ\\_file\\_format](https://en.wikipedia.org/wiki/XYZ_file_format)) that can be visualized by VMD (<https://www.ks.uiuc.edu/Research/vmd/>). Before this, one has to rename the file, giving it a suffix of .xyz.
- There are no functionalities for building simulation models within GPUMD, but the Python package thermo (<https://github.com/AlexGabourie/thermo>) developed by Alexander J. Gabourie can be used to pre-process and post-process data related to GPUMD.
- Recently, ASE (<https://gitlab.com/ase/ase>) also supported reading and writing the xyz.in input file for GPUMD. Check the read\_gpumd and write\_gpumd functions in the io module of ASE.

Retrieved from "[https://gpumd.zheyongfan.org/index.php?title=The\\_xyz.in\\_input\\_file&oldid=22315](https://gpumd.zheyongfan.org/index.php?title=The_xyz.in_input_file&oldid=22315)"

- 
- This page was last edited on 24 July 2022, at 17:48.