Main Page

From GPUMD
Jump to navigationJump to search

Contents

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. Using a single, powerful GPU, such as **Tesla P100**, it can run **100 MD steps for a one-million-atom system within one second**. See the tutorials for details.

- **Particularly good for heat transport applications**. See the theoretical formulations and publications (ht tps://github.com/brucefan1983/GPUMD/tree/master/publications) that developed or used `GPUMD`.

- **Installation of `GPUMD` is easy**. If you have a working CUDA development environment, it just requires a single `make` to install `GPUMD` in both Linux and Windows.

Versions of `GPUMD`

- This website documents the **developing version** of `GPUMD`.
- Because both the code and the documentation are developing, they can be inconsistent before a new release.
- We suggest use the latest released version, which is `GPUMD-v2.5` now.
- For each released version, there is a corresponding user manual in a PDF file.
- For a full description of the various versions of GPUMD, see the following table.

Released versions of GPUMD

Version	Date of release	Major changes relative to the previous version
GPUMD-v1.0 (https://github.com/brucefan1983/GPUMD/releases/tag/v1.0)	Aug 13, 2017	The first version of GPUMD.
GPUMD-v1.1 (https://github.com/brucefan1983/GPUMD/releases/tag/v1.1)	Aug 30, 2017	Added a potential describing single-layer black phosphorene (but the implementation is incorrect and was corrected in GPUMD-v1.5 (https://github.com/brucefan1983/GPUMD/releases/tag/v1.5)).
GPUMD-v1.2 (https://github.com/brucefan1983/GPUMD/releases/tag/v1.2)	Sep 25, 2017	Zero the angular momentum when initializing the velocities.
GPUMD-v1.3 (https://github.com/brucefan1983/GPUMD/releases/tag/v1.3)	Oct 12, 2017	Added a REBO (without LJ) potential for Mo-S systems.
GPUMD-v1.4 (https://github.com/brucefan1983/GPUMD/releases/tag/v1.4)	Dec 24, 2017	Added the Vashishta potential.
GPUMD-v1.5 (https://github.com/brucefan1983/GPUMD/releases/tag/v1.5)	Jan 24, 2018	Added the general two-element SW potential.
GPUMD-v1.6 (https://github.com/brucefan1983/GPUMD/releases/tag/v1.6)	Mar 4, 2018	Improved the performance of some potentials.
GPUMD-v1.7 (https://github.com/brucefan1983/GPUMD/rel	Jun 30, 2018	Added the HNEMD method for the Tersoff and SW potentials.

eases/tag/v1.7)		
GPUMD-v1.8 (https://github.com/brucefan1983/GPUMD/releases/tag/v1.8)	Jul 26, 2018	1) Added the HNEMD method for all the potentials. 2) Added support for hybrid potentials.
GPUMD-v1.9 (https://github.com/brucefan1983/GPUMD/releases/tag/v1.9)	Oct 9, 2018	Corrected a couple of memory bugs.
GPUMD-v2.0 (https://github.com/brucefan1983/GPUMD/releases/tag/v2.0)	Nov 23, 2018	Added the Langevin thermostat and the BDP thermostat.
GPUMD-v2.1 (https://github.com/brucefan1983/GPUMD/releases/tag/v2.1)	Dec 14, 2018	Implemented a more general version of the SHC method.
GPUMD-v2.2 (https://github.com/brucefan1983/GPUMD/releases/tag/v2.2)	Jan 19, 2019	1) Fixed a few memory bugs introduced in GPUMD-v2.0 (https://github.com/brucefan1983/GPUMD/releases/tag/v2.0) and GPUMD-v2.1 (https://github.com/brucefan1983/GPUMD/releases/tag/v2.1). 2) Changed the data format in the `xyz.in` file and removed the `layer.in` file. 3) Changed the output of the `dump_position` command from `xyz.out` to `movie.xyz`. 4) Added a `dump_restart` command which can produce a `restart.out` file. 5) The `compute_temp` command has been replaced by the more general `compute` command. 6) Added the `deform` command, which can be used to compute the stress-strain relation.
GPUMD-v2.3 (https://github.com/brucefan1983/GPUMD/releases/tag/v2.3)	Feb 20, 2019	1) Added the general Tersoff potential, which is applicable to systems with an arbitrary number of atom types. The added version is as general as the Tersoff potential in LAMMPS. 2) Added the support of triclinic box (but it currently cannot be used together with the NPT ensemble).
GPUMD-v2.4 (https://github.com/brucefan1983/GPUMD/releases/tag/v2.4)	Apr 20, 2019	1) Added a main program for phonon calculations based on harmonic lattice dynamics. 2) Improved the calculation of the vibrational density of states from the velocity autocorrelation function.
GPUMD-v2.4.1 (https://github.com/brucefan1983/GPUMD/releases/tag/v2.4-correction)	Apr 20, 2019	Corrected `makefile.phonon`, which is incorrect as in GPUMD-v2.4 (https://github.com/brucefan1983/GPUMD/releases/tag/v2.4).

GPUMD-v2.5	Sep 1, 2020	1) Added the Tersoff-mini potential. 2) Added the FCP potential. 3) Added a modal analysis method. 4) Changed the data format in the `xyz.in` file by removing the `has_layer` item in the first line. 5) Changed the grammar of the `potential` keyword and removed the `potentials` keyword. 6) Added the `minimize` keyword. 7) Changed the grammar of the `compute_shc` keyword. 8) Enriched the features of the `compute_dos` keyword by allowing for calculating the DOS for all the groups. 9) Added support for dumping the NETCDF file. 10) Added support for Windows with MSVC.

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, `gpumd` and `phonon`, will be generated in the `src` directory.
- Please check the comments in the beginning of the makefile for some compiling options.

Run GPUMD

The driver input file

- To run either 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/gpumd` or `src/phonon`) 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.

Prepared examples

- Go to the directory where you can see `src`.
- Type `src/gpumd < examples/input_gpumd.txt` to run the examples in `examples/gpumd`.
- Type `src/phonon < examples/input_phonon.txt` to run the examples in `examples/phonon`.

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
xyz.in	Define the **simulation model**
`run.in`	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 ...
```

- 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	**Yes**	N/A
potential	Set up a single potential	**Yes**	N/A
minimize	Perform an energy minimization	**Yes**	N/A
ensemble	Specify an integrator for a run	No	No
time_step	Specify the time step for integration	No	**Yes**
neighbor	Require 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
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

- The overall structure of a `run.in` file is as follows:
  - First, set up the initial velocities using the velocity keyword and set up the potential model using the potential keyword.
  - Then, if needed, use the minimize keyword to minimize the energy of the whole system.
  - 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.

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	dump_restart	The restart file	**Overwrite**
v.out	dump_velocity	The velocity file	Append
f.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

Inputs for the `src/phonon` executable

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

Input files for the `src/phonon` executable

Input filename	Brief description
xyz.in	Define the simulation model
basis.in	Define the mapping from the atom label to the basis label
kpoints.in	Specify the k-points
`phonon.in`	Define the simulation protocol

- The `phonon.in` file is used to define the simulation protocol for `sr/phonon`. 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 ...
```

- Here is the complete list of the keywords:

Keywords for the `phonon.in` input file

Keyword	Brief description	Take action immediately?
potential	Set up a single potential	Yes
minimize	Perform an energy minimization	Yes
cutoff	The cutoff distance used for calculating the force constants	No
delta	The finite displacement used in calculating the force constants	No

- The overall structure of a `phonon.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 use the other keywords to set up some parameters.

Output files for the `src/phonon` executable

Output files for the `src/phonon` executable

Output filename	Brief description	Output mode
D.out	Dynamical matrices $D(\vec{k})$ for the input k points	Overwrite
omega2.out	Phonon frequency square $\omega^2(\vec{k})$ for the input k points	Overwrite

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

- Hybrid potentials can be constructed from these individual ones. See the potential keyword for details.

Tutorials for using `GPUMD`

Tutorials for using the `src/gpumd` executable

Tutorials for using the `src/gpumd` executable

Tutorial name	Brief description
Tutorial: Thermal expansion	Study thermal expansion of silicon crystal from 100 K to 1000 K
Tutorial: Density of states	Calculate the vibrational density of states of graphene at 300 K
Tutorial: Thermal conductivity from EMD	Calculate the lattice thermal conductivity of graphene at 300 K using the EMD method
Tutorial: Thermal transport from NEMD and HNEMD	Calculate the spectral conductance and conductivity of graphene using the NEMD, HNEMD, and spectral decomposition methods

Tutorials for using the src/phonon executable

Tutorials for using the src/phonon executable

Tutorial name	Brief description
Tutorial: Phonon dispersion	Calculate the phonon dispersion of silicon crystal

Python interface of GPUMD

- The thermo (https://github.com/AlexGabourie/thermo) Python package developed by Alex Gabourie can be used to pre-process and post-process GPUMD.

Theoretical formulations of GPUMD

- Here are the Theoretical formulations of GPUMD.
- Zheyong Fan is writing a book on MD, which is in Chinese.

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(at)freelists.org
  - Here is the archive (public): https://www.freelists.org/archive/gpumd/
- Discussion group
  - There is a Chinese discussion group based on the QQ 群 887975816.

Want to develop GPUMD?

Developers of GPUMD

- GPUMD was first developed by **Zheyong Fan** (Previously Aalto University; brucenju(at)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(at)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 from compute capability no less than 3.5 and in both Linux and Windows systems.
- We use the .clang-format file within src/ to format any CUDA C++ source code.

Units system adopted within GPUMD

- We use the following basic units
  - Energy: eV (electron volt)
  - Length: A (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.nsfc.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/)