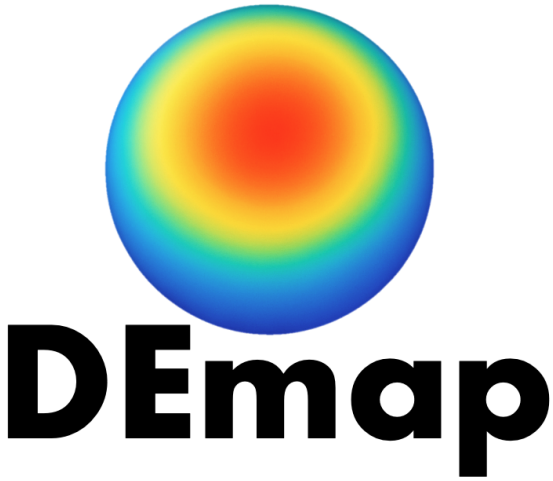

Manual

Version 1

Ashley Dickson

November 3, 2025

Contents

1 Introduction

This document details how to use DEmap to determine threshold displacement energy surfaces for a given Primary Knock-on Atom (PKA).

2 Install

DEmap can simply be installed using pip:

```
pip install demap
```

DEmap has a number of dependencies that are automatically installed when running this command. These dependencies are listed on the GitHub page.

3 Initial Set-up

The user must manually perform a couple of tests before running DEmap simulations. The first test is the size of cell required for simulations. One must find the smallest cell possible that contains a PKA of the highest Threshold Displacement Energy (TDE) expected from simulations. In general, a cell of size 50Å in all directions is sufficient, though the user should run their own tests.

Second, it is important to determine the cutoff time of the simulation. Perhaps the easiest and most intuitive way of doing this is to perform the aforementioned simulation, and measure the potential energy as a function of time. The cutoff time is then determined as the time at which the potential energy stabilises. An example of the potential energy of displacement event over time is shown in Figure 1. The cutoff time can be determined as 2000 ps in this case.

IMPORTANT NOTE: The user should first energy minimise the cell (box size and atom positions) before running these simulations. Furthermore, the simulation should be performed at zero kelvin, such that it is comparable to our DEmap simulations. An example simulation script is included

on the github (https://github.com/Ash-Dickson/DEmap/tree/main/initial_sanity_checks).

4 Creating the input

DEmap has a straight-forward function that generates the input file for TDE simulations. An example use is as follows:

```
from DEmap import *

generate_lammps_input(
    masses=[(1, 55.845)],
    pair_styles=['pair_style hybrid/overlay eam/fs tabgap'],
    pair_coeffs=[
        'pair_coeff * * eam/fs FULL/PATH/TO/POT/Fe-2021-04-06.eam.fs Fe',
        'pair_coeff * * tabgap FULL/PATH/TO/POT/Fe-2021-04-06.tabgap Fe no yes'
    ],
    units='metal',
    atom_style='atomic',
    read_data='Fe.data',
    run_steps=2000,
    PKA_id=4367
)
```

The definitions of each keyword are as follows:

- masses: list of tuples, defining mapping from LAMMPS atom type to mass.
- pair_styles: list of strings corresponding to the LAMMPS pair style you are using.
- pair_coeffs: list of strings corresponding to interatomic potential details. Each element of the list is a new line in the lammps input. **NOTE: when specifying the file path to the potential, please use the full path.**
- units: LAMMPS units.
- atom_style: LAMMPS atom_style.

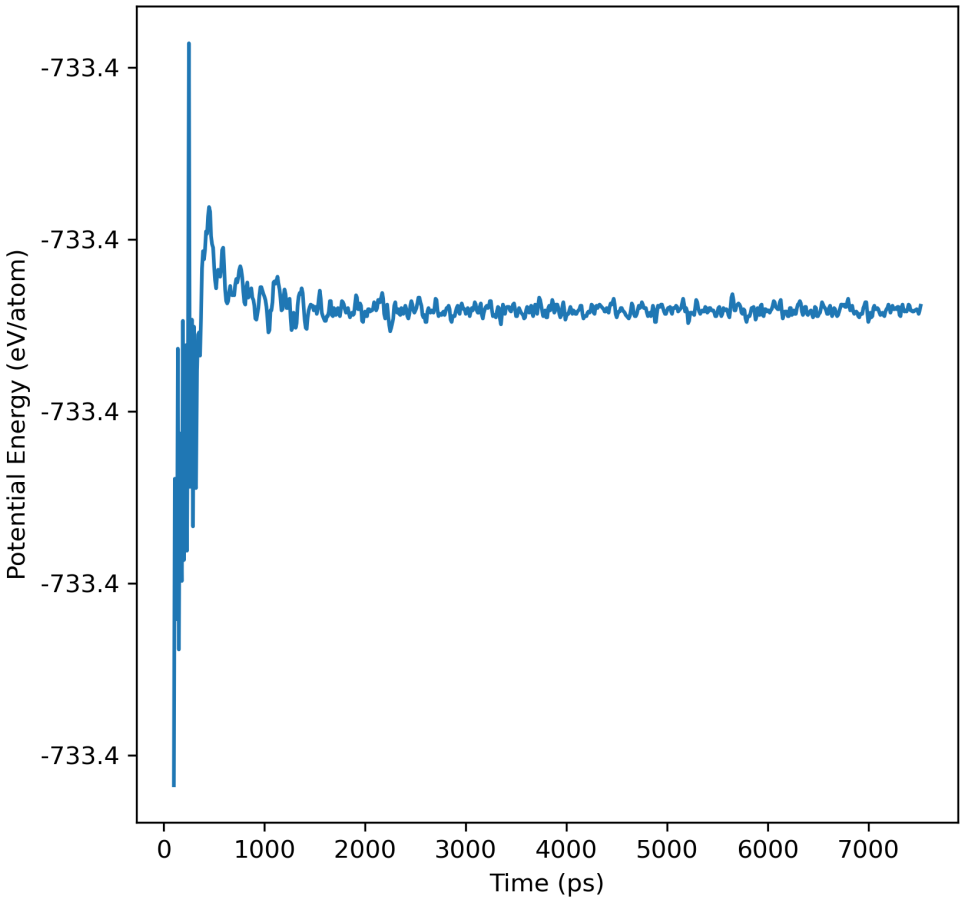

Figure 1: Potential energy of a threshold displacement event versus time.

- read_data: LAMMPS data file containing structure details. **NOTE: This should be the relaxed supercell that TDE simulations are performed within.**
- run_steps: number of LAMMPS steps to run before stopping. This should correspond to when the potential energy stabilises for high displacement energy event (see section 3).
- PKA_id: the LAMMPS atom id of the PKA atom. **NOTE: this should be selected as close to the center of the cell as possible (in order to minimise chance of the PKA crossing the periodic boundaries).**

5 Running DEmap

The simulation setup is as follows. Create a new directory to run your simulations. Into this directory you will place the relaxed LAMMPS data file specified in read_data. The user will then run generate_lammps_input (as above) in this directory. An example of the subsequent code to run DEmap is shown below:

```
from DEmap import *
import numpy as np

generate_lammps_input(
    masses=[(1, 55.845)],
    pair_styles=['pair_style hybrid/overlay eam/fs tabgap'],
    pair_coeffs=[
        'pair_coeff * * eam/fs FULL/PATH/TO/POT/Fe-2021-04-06.eam.fs Fe',
        'pair_coeff * * tabgap FULL/PATH/TO/POT/Fe-2021-04-06.tabgap Fe no yes'
    ],
    units='metal',
    atom_style='atomic',
    read_data='Fe.data',
    run_steps=2000,
    PKA_id=4367
)

cfg = Config(
run_line='mpirun /PATH/TO/LMP/BINARY/lmp -in tde.in',
                                init_n=50, max_iters=1400)

demap(theta_range=(0, np.pi/4),
phi_range=(0, np.pi/4), restart=True, cfg=cfg, plot=True)
```

The `Config()` function specifies the setup of the Gaussian Process (GP) code. Key parameters include:

- **init_n:** The total number of initial points to sample from the Fibonacci sphere. A minimum of 50 points is required. The appropriate number depends on system complexity:
  - Simple systems: 50 points are usually sufficient.
  - Complex systems: 200 points are recommended for better coverage.
- **run_line:** The LAMMPS execution command. This should be set to the path of your LAMMPS binary. By default, `generate_lammps_input` creates an input file called `tde.in`, so the LAMMPS input file should be specified as `-in tde.in`.

- **max_iters:** The maximum number of directions to sample (after `init_n`) before the simulation exits (integer).
- **probe_n**: total number of fibonacci points to sample with trained GP model to output final data (integer, defaults to 10000).
- **random_seed**: start seed for numpy and torch to ensure consistency between runs (integer).

The `demap()` function allows customization of simulation execution:

- **Restart:** Boolean flag (`True/False`). If set to `True`, the simulation will attempt to restart from an existing run. This is useful if the job hits the HPC wall time limit.
- **cfg: Required.** Pass the `Config()` object here so that `demap()` knows how to run LAMMPS.
- **Plot:** Boolean flag. If set to `True`, `demap()` will generate `TDEMAP.png` and `VARMAP.png` after simulations are complete.
- **theta_range** and **phi_range:** Specify the segment of the unit sphere to sample.
  - `theta` ranges from 0 (north pole, +*z*) to π (south pole, $-z$).
  - `phi` ranges from 0 to 2π, where ϕ = 0 corresponds to the +*x* axis.

Figure 2 illustrates diagrammatically the meaning of the spherical angles ϕ and θ. One should try to pick the θ and ϕ ranges that correspond to the symmetrically distinct set of PKA vectors in the material of interest. This will greatly aid the efficiency of the Gaussian Process sampling.

Running the Script

To execute the workflow:

1. Save the Python script in the same directory as the LAMMPS data file.
2. Run the script either locally or on an HPC cluster.

The script will automatically generate the LAMMPS input, run simulations, and optionally produce TDE and variance (TDEMAP.png and VARMAP.png). The units for these plots are degrees on both axes, with ϕ running around the

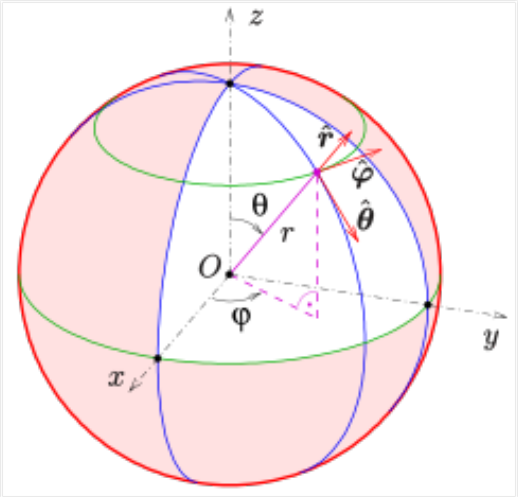

Figure 2: Spherical angles in the unit sphere (credit to https://en.wikipedia.org/wiki/Spherical_coordinate_system).

circular axis and θ along the radial axis.

On completion, DEmap will also create probe_points.txt, tde_points.txt, and var_points.txt. These text files contain numpy arrays for the entire list of vectors that were sampled (all n points specified by **probe_n**), the TDE values at these vectors, and the variance at these vectors, respectively. One can use these files to create their own plots - for inspiration on how to construct plots on these data consult the plot_tools.py module in DEmap.