<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Installation</td>
<td>3</td>
</tr>
<tr>
<td>2 Usage</td>
<td>5</td>
</tr>
<tr>
<td>3 Example for a-Si</td>
<td>7</td>
</tr>
<tr>
<td>3.1 Prerequisites</td>
<td>7</td>
</tr>
<tr>
<td>4 API documentation</td>
<td>9</td>
</tr>
<tr>
<td>Python Module Index</td>
<td>13</td>
</tr>
</tbody>
</table>
This package contains Python tools for calculating the spectral decomposition of heat current from the data produced by non-equilibrium molecular dynamics simulation with LAMMPS software. The relevant theory was published in:


The package is meant to help anyone interested in implementing the spectral heat current calculations for their own applications. If you want to use the code for research purposes, please cite the above-mentioned publications and let me know.

*Note that I’m currently rewriting both the documentation and code so this is still work in progress!*
Clone the GitHub repository and install the package from the repository root:

```
$ pip install -e .
```

The command installs the library and its dependencies listed in `setup.py`.

If you wish to use Python package to also compute the force constants, you need to have Python library interface for LAMMPS set up.

You will also need to build `compactify_vels.cpp` found in `scripts` folder and have `compactify_vels` available in your `$PATH`. If you’re lucky, you only need to do `make` in `scripts` folder.

I may later also add the package to PyPI so one can simply do `pip install sdhc` to use the package to compute spectral decompositions.
Example:

```python
from sdhc import SHCPostProc
import numpy as np

postprocessor = SHCPostProc(*args, **kwargs)
postprocessor.postProcess()

# Save frequencies and smoothened spectral heat currents as NumPy files
np.save('angular_frequencies.npy', postprocessor.oms_fft)
np.save('heat_currents.npy', postprocessor.SHC_smooth)

# Save the frequencies and smoothened spectral heat currents to text file
np.savetxt('frequencies_and_currents.txt', np.column_stack((oms, postprocessor.SHC_smooth))
```
CHAPTER 3

Example for a-Si

Folder example contains a self-contained example for calculating the spectral decomposition of heat current flowing across a slab of amorphous Si. The script to be run is called silicon_example.py. It performs the following steps:

1. prepare a box of atoms,
2. call LAMMPS to perform the quenching procedure contained in LAMMPS input file quench_Si.lmp,
3. call LAMMPS to perform the actual NEMD calculation for a-Si using amorphous_interface.lmp, and
4. perform the post-processing using sdhc

3.1 Prerequisites

- Using LAMMPS from Python requires that you have built LAMMPS as a dynamically shared library as instructed in the LAMMPS manual
- Simulation uses the sw pair style, which is included in the MANYBODY package. See here how to include packages in your LAMMPS build.
class _sdhc.fcCalc (fileprefix, restartfile)
Class for computing force constants between atoms. Uses the Python library interface of LAMMPS so you need to have (1) lammps in your PYTHONPATH and (2) liblammps.so available for the Python package.

Parameters

• fileprefix (str) – File prefix (TODO What is this)
• restartfile (str) – LAMMPS restart file (TODO What is this)

fcCalc (hstep)
Compute force constants and store to self.Kij.

Parameters hstep (float) – Step to use in finite differences

Returns None

call preparelammps (pair_style=None, pair_coeff=None, x_interface=0.5, w_interface=3.0)
Prepare the LAMMPS object for computing force constants.

Parameters

• pair_style (str, optional) – LAMMPS pair style to set
• pair_coeff (str, optional) – LAMMPS pair_coeff to set
• x_interface (float, optional) – Position of the interface relative to boxlo and boxhi, defaults to 0.5
• w_interface (float, optional) – Width of the area of atoms to include in the interface, defaults to 3.0

Returns None

writeToFile ()
Write self.Kij to files starting with self.fileprefix.

Returns None
class sdhc.SHCPPostProc(compactVelocityFile, KijFilePrefix, reCalcVels=False, reCalcFC=False, **args)

Compute the spectral decomposition of heat current from the data produced using LAMMPS MD simulation. The velocities are read from the “compact” file produced with the C++-code compactify_vels.cpp from a LAMMPS dump file. If the dump file does not exist, it is produced by calling the binary compactify_vels, which must be found in the environment’s $PATH.

Minimal usage in Python:

```python
pP = SHCPPostProc(compactVelocityFile, KijFilePrefix)
pP.postProcess()  # Calculate the heat current spectrum
```

Public attributes:

- SHC_smooth (numpy float array): The chunk-averaged, smoothened spectral heat current
- SHC_smooth2 (numpy float array): Square of the chunk-averaged, smoothened spectral heat current, used for estimating the error from the between-chunk variance
- SHC_average (numpy float array): The chunk-averaged spectral heat current without smoothing
- SHC_error (numpy float array): The estimated error from the between-chunk variance, None if only one chunk evaluated
- oms_fft (numpy float array): The angular frequency grid (in the units of Hz if dt_md is given in the units of seconds in the initialization)

Parameters

- **compactVelocityFile (str)** – Path to file from where the velocities are read. Produced using the binary compactify_vels if the file does not exist. In this case, you must also supply the keyword argument LAMMPSDumpFile containing the velocities produced using LAMMPS.
- **KijFilePrefix (str)** – The prefix used in trying to find the force constant matrix file KijFilePrefix.Kij.npy. If the file does not exist, the force constant calculator fcCalc is called using the keyword argument argument LAMMPSRestartFile (which must be supplied in this case).
- **reCalcVels (boolean, optional)** – Force recreation of the compact velocity file if True, defaults to False
- **reCalcFC (boolean, optional)** – Re-calculate force constants if True, defaults to False
- **args** – Additional keyword arguments

Additional keyword arguments:

- **dt_md (float)**: Timestep used in the NEMD simulation (seconds), used for inferring the sampling timestep and the frequency grid (default 1.0)
- **scaleFactor (float)**: Multiply the spectral heat current by this factor to convert to correct units (default 1.0)
- **LAMMPSDumpFile (str)**: Use this velocity dump file produced by LAMMPS for post-processing, needed only if the compact velocity file cannot be found (default None)
- **LAMMPSRestartFile (str)**: Use this restart file for calculating the force constants if the force constant file cannot be found (default None)
- **widthWin (float)**: Use this width for the smoothing window (Hz) (default 1.0)
- **chunkSize (int)**: Used chunk size for reading the velocities, affects the frequency grid (default 50000). Performing FFT is faster if chunkSize is a power of 2.
- NChunks (int): The number of chunks to be read, this should be set to a sufficiently large value if the whole velocity file should be read (default 20)
- backupPrefix (str): Prefix for pickling the post-processing object to a file after the read of each chunk (default None)
- hstep (float): The displacement used in calculating the force constants by the finite-displacement method (default 0.001)

`postProcess()`

Calculate the spectral decomposition and store to `self.SHC_smooth`.

Returns None
s
sdhc, 9
Index

F
fcCalc (class in sdhc), 9
fcCalc() (sdhc.fcCalc method), 9

P
postProcess() (sdhc.SHCPostProc method), 11
preparelammps() (sdhc.fcCalc method), 9

S
sdhc (module), 9
SHCPostProc (class in sdhc), 9

W
writeToFile() (sdhc.fcCalc method), 9