GPUQT:
Graphics Processing Units Quantum Transport

Reference Manual

Version 0.9
(change to 1.0 upon official release)

(Dec 21, 2018)

Authors:
Zheyong Fan (Aalto University)
Ville Vierimaa (Aalto University)
Ari Harju (Aalto University)

Contents

Chapter 1

Introduction

1.1 What is GPUQT?

`GPUQT` is an efficient and flexible implementation of the linear scaling quantum transport (LSQT) methods reviewed in Fan et al. (2018a). The `GPUQT` code is not fully finished yet, and we aim to complete version 1.0 within a few months.

Here are the major available features in the current code:

- It supports both pure CPU and GPU+CPU computations. One can build a CPU or a GPU version. Both versions give consistent results for a given simulation.
- It can be used to study large systems with millions or even tens of millions of atoms. The major limitation on the system size comes from the limited amount of CPU and/or GPU memory available.
- It can be used to calculate the total and local electronic density of states (DOS).
- It can be used to calculate the velocity auto-correlation (VAC) function and/or the mean square displacement (MSD), from which one can deduce other transport quantities such as electrical conductivity, propagating length, electrical conductance, diffusivity, group velocity, mobility, relaxation time, mean free path, localization length, etc.
- It can be used to calculate the spin polarization, from which one can deduce the spin relaxation time.

TODO list up to version 1.0:

- Improve the features related to model building
- Kubo-Bastin formula for Hall transport

TODO list for future versions:

- Optical conductivity
- Lattice thermal transport
- OpenMP acceleration for the CPU version
- MPI acceleration for both the CPU and the GPU versions

1.2 Citations

- If you use `GPUQT` in your published work, we kindly ask you to cite the review paper Fan et al. (2018a) which describes the various algorithms used in `GPUQT`.
- The most important original paper is Roche and Mayou (1997) which introduced the concept of MSD.
- When you use `GPUQT` to do calculations related to Hall transport (not implemented yet), you can cite Ortmann et al. (2015) and García et al. (2015).
- When you use `GPUQT` to do calculations related to spin transport, you can cite one or more of these papers: Van Tuan et al. (2014), Vierimaa et al. (2017), and Cummings et al. (2017).
- When you use the GPU version, you can cite Fan et al. (2018b) and/or Fan et al. (2014).

1.3 Feedbacks

You can email the first author if you find errors in the manual or bugs in the source code, or have any suggestions/questions about the manual and code. The following email addresses can be used:

- zheyong.fan(at)aalto.fi (valid at least up to the end of 2019)
- brucenju(at)gmail.com

1.4 Acknowledgments

We acknowledge the computational resources provided by Aalto Science-IT project and Finland's IT Center for Science (CSC).

Chapter 2

Theoretical formalisms

The theoretical formalisms for the LSQT methods have been comprehensively reviewed in Fan et al. (2018a). Here we only give a brief review on the formulas that are implemented in `LSQT`. The major purpose here is to set up the necessary notations.

2.1 Quantities to be calculated

2.1.1 Total density of states

The total density of states (DOS) is defined as

$$\rho(E) = \frac{2}{\Omega}\mathrm{Tr}\left[\delta(E-\hat{H})\right], \tag{2.1}$$

where Ω is the volume of the system, E is the Fermi energy, $\hat{H}$ is the system Hamiltonian. The factor 2 stands for spin degeneracy. The normalization by the volume is not necessary; one can also change Ω to the total number of orbitals N. The choice of our definition makes the later expressions for the conductivity simpler.

2.1.2 Local density of states

The local density of states (LDOS) for orbital $|i\rangle$ is defined as

$$\rho_i(E) = 2\langle i|\delta(E-\hat{H})|i\rangle \tag{2.2}$$

2.1.3 Dissipative conductivity from velocity auto-correlation

One can express the time-dependent electrical conductivity $\sigma^{\mathrm{VAC}}(E,t)$ as a time integral of the velocity auto-correlation (VAC) $C_{vv}(E,t)$,

$$\sigma^{\mathrm{VAC}}(E,t) = e^2\rho(E)\int_0^t C_{vv}(E,t)dt; \tag{2.3}$$

$$\rho(E)C_{vv}(E,t) = \frac{2}{\Omega}\mathrm{Re}\left[\mathrm{Tr}\left[\hat{U}(t)\hat{V}\delta(E-\hat{H})\hat{U}(t)^{\dagger}\hat{V}\right]\right]; \tag{2.4}$$

where $\hat{V}(t) = \hat{U}^{\dagger}(t)\hat{V}\hat{U}(t) = e^{i\hat{H}t/\hbar}\hat{V}e^{-i\hat{H}t/\hbar}$ is the velocity operator in the transport direction in the Heisenberg representation.

2.1.4 Dissipative conductivity from mean square displacement

Equivalently, one can expresses the time-dependent electrical conductivity as a time derivative of the mean square displacement (MSD) $\Delta X^2(E,t)$,

$$\sigma^{\mathrm{MSD}}(E,t) = e^2\rho(E)\frac{d}{2dt}\Delta X^2(E,t); \tag{2.5}$$

$$\rho(E)\Delta X^2(E,t) = \frac{2}{\Omega}\mathrm{Tr}\left[[\hat{X},\hat{U}(t)]^\dagger\delta(E-\hat{H})[\hat{X},\hat{U}(t)]\right], \tag{2.6}$$

where $\hat{X}$ is the position operator in the transport direction.

2.1.5 Spin polarization

The time-dependent spin polarization $S(E,t)$ is defined as

$$S(E,t) = \frac{\mathrm{Tr}\left[(\hat{I}_N\otimes\hat{s}_z(t))\delta(E-\hat{H}_{2N})\right]}{\mathrm{Tr}\left[\delta(E-\hat{H}_{2N})\right]}, \tag{2.7}$$

where $\hat{I}_N\otimes\hat{s}_z(t) = U^\dagger(t)(\hat{I}_N\otimes\hat{s}_z)U(t)$, $\hat{I}_N$ is an identity matrix of size $N\times N$ (N is the number of orbitals for one spin), and $\hat{s}_z(t) = \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$ is the Pauli z-matrix. Here we assume that the initial spin polarization is along the z direction. Note that the Hamiltonian $\hat{H}_{2N}$ is defined in the compound Hilbert space formed by the tensor product of the N-dimensional orbital space and the 2-dimensional spin space.

2.1.6 Kubo-Bastin conductivity

To be written.

2.1.7 Optical conductivity

To be written.

2.1.8 Lattice thermal conductivity

To be written.

2.2 Linear-scaling techniques

2.2.1 Linear-scaling evaluation of the trace

We use the random vector approximation method Weiße et al. (2006) to evaluate the trace in the above equations. In this method, the trace of a general operator $\hat{A}$ of size $N\times N$ is written as

$$\mathrm{Tr}\left[\hat{A}\right] \approx N\langle\phi|\hat{A}|\phi\rangle, \tag{2.8}$$

where $|\phi\rangle$ is a random vector normalized to 1, $\langle\phi|\phi\rangle = 1$. The error introduced by this approximation decreases with increasing N, scaling as $\sim 1/\sqrt{N}$ Weiße et al. (2006). For a

given N, the accuracy can also be increased by taking average over independent random vectors. Usually, it is enough to use a few random vectors.

With this approximation, we have

$$\rho(E) \approx \frac{2}{\Omega}\langle\phi|\delta(E-\hat{H})|\phi\rangle; \tag{2.9}$$

$$\rho(E)C_{vv}(E,t) \approx \frac{2}{\Omega}\mathrm{Re}\left[\langle\phi|\hat{U}(t)V\delta(E-\hat{H})\hat{U}(t)^{\dagger}V|\phi\rangle\right]; \tag{2.10}$$

$$\rho(E)\Delta X^2(E,t) \approx \frac{2}{\Omega}\langle\phi|[X,\hat{U}(t)]^{\dagger}\delta(E-\hat{H})[X,\hat{U}(t)]|\phi\rangle. \tag{2.11}$$

2.2.2 Linear-scaling evaluation of the time evolution

The time evolution operator $\hat{U}(t)$ in the above equations can be evaluated by using Chebyshev polynomial expansion. For one time step Δt, we have the following Chebyshev polynomial expansions Tal-Ezer and Kosloff (1984):

$$\hat{U}(\pm\Delta t)|\psi\rangle \approx \sum_{m=0}^{N_p-1}(2-\delta_{0m})(\mp i)^m J_m\left(\widetilde{\Delta t}\right)T_m(\widetilde{H})|\psi\rangle; \tag{2.12}$$

$$[X,\hat{U}(\Delta t)]|\psi\rangle \approx \sum_{m=0}^{N_p-1}(2-\delta_{0m})(-i)^m J_m\left(\widetilde{\Delta t}\right)[X,T_m(\widetilde{H})]|\psi\rangle, \tag{2.13}$$

where J_m is the mth order Bessel function of the first kind and T_m is the mth order Chebyshev polynomial of the first kind. Note that T_m is defined in the interval $[-1,1]$ and the Hamiltonian and time step have to be scaled in the opposite way:

$$\widetilde{H} = \hat{H}/\Delta E; \tag{2.14}$$

$$\widetilde{\Delta t} = \Delta E \Delta t, \tag{2.15}$$

where ΔE is sufficiently large such that the spectrum of the scaled Hamiltonian $\widetilde{H}$ lies within the interval $[-1,1]$. The Chebyshev polynomial expansions of the time evolution operators can be evaluated up to machine precision and the order of expansion N_p needed for achieving this is proportional to the time interval.

The above summations can be efficiently evaluated by using the following recurrence relations ($m \geq 2$) ($T_m(\widetilde{H})$ is written as T_m for simplicity):

$$T_m = 2\widetilde{H}T_{m-1} - T_{m-2}; \tag{2.16}$$

$$[X,T_m] = 2[X,\widetilde{H}]T_{m-1} + 2\widetilde{H}[X,T_{m-1}] - [X,T_{m-2}]; \tag{2.17}$$

$$T_0 = 1 \quad T_1 = \widetilde{H}; \tag{2.18}$$

$$[X,T_0] = 0, \quad [X,T_1] = [X,\widetilde{H}]. \tag{2.19}$$

2.2.3 Linear-scaling evaluation of the quantum resolution operator

We use the kernel polynomial method (KMP) Weiße et al. (2006) to approximate the quantum resolution operator:

$$\rho(E) \approx \frac{2}{\pi\Omega\Delta E\sqrt{1-\widetilde{E}^2}} \sum_{n=0}^{N_m-1} g_n(2-\delta_{n0})T_n(\widetilde{E})C_n^{\mathrm{DOS}}; \tag{2.20}$$

$$\rho(E)C_{vv}(E,t) \approx \frac{2}{\pi\Omega\Delta E\sqrt{1-\widetilde{E}^2}} \sum_{n=0}^{N_m-1} g_n(2-\delta_{n0})T_n(\widetilde{E})C_n^{\mathrm{VAC}}(t); \tag{2.21}$$

$$\rho(E)\Delta X^2(E,t) \approx \frac{2}{\pi\Omega\Delta E\sqrt{1-\widetilde{E}^2}} \sum_{n=0}^{N_m-1} g_n(2-\delta_{n0})T_n(\widetilde{E})C_n^{\mathrm{MSD}}(t). \tag{2.22}$$

Here, C_n^{DOS}, $C_n^{\mathrm{VAC}}(t)$, and $C_n^{\mathrm{MSD}}(t)$ are the Chebyshev moments:

$$C_n^{\mathrm{DOS}} \approx \langle\phi|T_n(\widetilde{H})|\phi\rangle; \tag{2.23}$$

$$C_n^{\mathrm{VAC}}(t) \approx \mathrm{Re}\left[\langle\phi|\hat{U}(t)\hat{V}T_n(\widetilde{H})\hat{U}(t)^\dagger\hat{V}|\phi\rangle\right]; \tag{2.24}$$

$$C_n^{\mathrm{MSD}}(t) \approx \langle\phi|[\hat{X},\hat{U}(t)]^\dagger T_n(\widetilde{H})[\hat{X},\hat{U}(t)]|\phi\rangle. \tag{2.25}$$

and

$$g_n = (1-n\alpha)\cos(\pi n\alpha) + \alpha\sin(\pi n\alpha)\cot(\pi\alpha);\ \alpha = \frac{1}{N_m+1} \tag{2.26}$$

is the Jackson damping factor used to suppress Gibbs oscillations. The energy resolution achieved scales as $\delta \sim 1/N_m$ Weiße et al. (2006). Therefore, to achieve a finer energy resolution, one needs to use a larger N_m. Usually, a value of $N_m = 3000$ is more than enough.

Chapter 3

Using LSQT

The code has only been tested in linux operating systems and we assume that the user is using a linux operating system to compile and run this code.

3.1 Compile the code and run the examples

3.1.1 Compiling

After downloading and unpacking `GPUQT`, one can see three folders: `src`, `doc`, and `examples`. The folder `src` contains all the source files. The folder `examples` contains all the examples. The folder `doc` contains the pdf file you are reading and the source files generating it.

To compile the code, first go to the `src` folder. Then one can build either a CPU version or a GPU version or both.

- To build the CPU version, type

  ```
  make -f makefile.cpu
  ```

  in the command line. This will produce an executable called `lsqt_cpu` in the `src` folder. The second line of `makefile.cpu` reads

  ```
  CFLAGS = -O3 -std=c++11 -x c++ -DDEBUG -DCPU_ONLY
  ```

  The flag `-DDEBUG` means that a fixed seed for the random number generator will be used in different runs. This will result in identical results from two independent runs with the same inputs. This is useful for debugging, but usually is not good for production runs. Remember to remove `-DDEBUG` if you want to build a version which uses different seeds for the random number generator in different runs.

- To build the GPU version, type

  ```
  make -f makefile.gpu
  ```

  in the command line. This will produce an executable called `lsqt_gpu` in the `src` folder. To build the GPU version, you need to have a CUDA toolkit and a CUDA-enabled GPU with compute capability of 2.0 or higher. The second line of `makefile.gpu` reads

```
CFLAGS = -O3 -std=c++11 -arch=sm_35 -use_fast_math -DDEBUG
```

The option `-arch=sm_35` means that the code is built with a target compute capability of 3.5. It is usually a good idea to set the compute capability as the same one for your GPU.

3.1.2 Running

To do calculations, one has to first prepare a "driver input file" to specify the paths of the working directories containing the actual input files. For example, the following "driver input file" specifies two working directories:

```
examples/cpc2018/lattice/diffusive
examples/cpc2018/lattice/localized
```

Here I used relative paths starting from the directory where you can see the `src` folder. One can also use absolute paths.

Suppose that the "driver input file" is named as `input.txt` and is in the `examples` folder, one can run the code by typing

```
src/lsqt_gpu examples/input.txt
```

or

```
src/lsqt_cpu examples/input.txt
```

We now turn to describe the actual input files needed for `GPUQT`.

3.2 Input files for GPUQT

The input files are used to specify the Hamiltonian and related quantities of a simulated system and some controlling parameters. All the input files for a simulation should be in a single folder. We currently provide two ways to specify the Hamiltonian and related quantities. One way is to use **the general model** and the other is to use **the lattice model**. In both cases, one needs to prepare a file called `para.in` which contains the controlling parameters, a file called `energy.in` which contains the Fermi energies, and possibly a file called `time_step.in` which contains the time steps.

3.2.1 The para.in input file

This file contains the controlling parameters defining the simulation process. In this input file, blank lines are ignored. Each non-empty line starts with a keyword possibly followed by one or more parameters. The valid keywords and their parameters are listed below.

1. `model`
   This keyword needs one parameter, which can only be 0 or 1, where 0 means using the general model to construct the Hamiltonian and related quantities and 1 means using the lattice model instead.

2. `anderson_disorder` W
   This keyword can only be used when using the lattice model. It needs one parameter, which is the strength W of the Anderson disorder. This keyword is used to add the Anderson disorder to the system. It cannot be used together with the `charged_impurity` keyword.

3. `charged_impurity` N_i W ξ
   This keyword can only be used when using the lattice model. It needs three parameters, which are consecutively the number of charged impurities N_i, the impurity strength W, and the impurity range ξ. This keyword is used to add charged impurities to the system. It cannot be used together with the `anderson_disorder` keyword.

4. `vacancy_disorder` N_v
   This keyword can only be used when using the lattice model. It needs one parameter, which is the number of vacancies N_v. This keyword is used to create vacancies in the system.

5. `calculate_vac`
   This keyword does not need any parameter. If this keyword appears, the VAC will be calculated. Otherwise, the VAC will not be calculated. It cannot be used simultaneously with `calculate_spin`.

6. `calculate_msd`
   This keyword does not need any parameter. If this keyword appears, the MSD will be calculated. Otherwise, the MSD will not be calculated. It cannot be used simultaneously with `calculate_spin`.

7. `calculate_spin`
   This keyword does not need any parameter. If this keyword appears, the spin polarization will be calculated. Otherwise, the spin polarization will not be calculated. It cannot be used simultaneously with `calculate_vac` or `calculate_msd`. Currently, one cannot calculate the spin polarization when using the lattice mode.

8. `number_of_random_vectors` N_r
   This keyword needs one parameter, which is the number of random vectors N_r used in the simulation. If this keyword is absent, the default value $N_r = 1$ will be used. If you want to use 10 random vectors for a given problem, you can either set this number to 10, or set it to 1 and then run the simulation 10 times. Increasing N_r can improve the accuracy of the results.

9. `number_of_moments` N_m
   This keyword needs one parameter, which is the number of Chebyshev moments N_m used in the kernel polynomial method. If this keyword is absent, the default value $N_m = 1000$ will be used. A larger N_m gives a finer energy resolution and one usually needs to test the effects of this parameter.

10. `energy_max` ΔE
    This keyword needs one parameter, which is a scaling parameter ΔE used to scale the Hamiltonian. The scaled Hamiltonian $\hat{H}/\Delta E$ must have all of its eigenvalues lying within the interval $[-1, 1]$. If this keyword is absent, the default value $\Delta E = 10$

will be used. Using a value larger than needed will only effectively reduce the energy resolution, but using a value smaller than needed will cause big problems as this will lead to calculating the square roots of negative numbers.

3.2.2 The energy.in input file

This file contains the Fermi energy points to be considered in the simulations. There is a single column in this file. The first line should be an integer, which is the number of energy points to be read in. Starting from the second line, the nth line contains the $(n-1)$th energy value. Note that the method is parallel in energy and using one thousand energy points takes roughly as much time as using a single energy point. Here is an example:

```
601
-3.00
-2.99
...
0.00
...
3.00
```

This file tells that there would be 601 energy points to be calculated, from -3 to 3, with a spacing of 0.01.

3.2.3 The time_step.in input file

This file contains the time steps in the VAC, MSD, and spin polarization calculations. There is a single column in this fie. The first line should be an integer, which is the number of time steps to be read in. Starting from the second line, the nth line is the $(n-1)$th time step. Here is an example:

```
20
1
2
...
19
20
```

This file tells that there would be 20 (non-uniform) time steps, from t_0 to 20 t_0, with a spacing of t_0. One should note that the data here are the time steps, not the cumulative times. The cumulative times for this example should be t_0, $3t_0$, $6t_0$, $10t_0$, $\cdots$. The unit of time, t_0 is fixed by the energy unit, as we set the reduced Planck constant to 1. Suppose the unit of energy is γ, the unit of time is then $t_0 = \hbar/\gamma$.

3.2.4 Other input files when using the general model

Essentially, to build the simulation model, we need information regarding the Hamiltonian and the velocity operators.

In the tight-binding approximation, the Hamiltonian can be written as

$$\hat{H} = \sum_m \sum_n H_{mn}|m\rangle\langle n| + \sum_m U_m|m\rangle\langle m|, \tag{3.1}$$

where H_{mn} is the hopping integral between orbitals m and n and U_m is the on-site potential of site m. Similarly, the position and velocity operators can be expressed as

$$\hat{X} = \sum_m X_m |m\rangle\langle m|; \tag{3.2}$$

$$\hat{V} = \frac{i}{\hbar}[\hat{H}, \hat{X}] = \frac{i}{\hbar} \sum_m \sum_n (X_n - X_m) H_{mn} |m\rangle\langle n|. \tag{3.3}$$

When using the general model, one has to specify four sets of data: 1) the neighbor list structure that determines which hopping integrals are non-vanishing, 2) the non-vanishing hopping integrals, 3) the on-site potentials, and 4) the positions X_m of the sites projected onto the transport direction.

- The `neighbor.in` input file
  This file specifies the topology of the problem using a neighbor list. This will be used to build the sparse Hamiltonian. The first line should have two integer numbers. The first number is the total number of sites in the simulated system. The second number is the maximum possible number of nonzero hopping integrals originated from a given site. For example, in a square lattice with nearest-neighbor hopping only, this number can be set as 4. Using a larger number than needed will just waste memory, while using a smaller number than needed will cause an error. Starting from the second line, the nth line contains the number of neighbors and the indices of the neighboring sites of the $(n-1)$th site.

- The `hopping.in` input file
  This is an optional input file, which contains the hopping integrals (the off-diagonal terms in the Hamiltonian). If this file is not prepared, it is assumed that all the hopping integrals between pairs of neighboring sites (specified in the `neighbor.in` file) are -1. The first line should be either the word `real` or `complex`. If the word is `real`, it means that all the hopping integrals are real numbers. Then, starting from the second line, the nth line contains the real hopping integrals between the $(n-1)$th site and its neighboring sites, and the order should be consistent with that in the `neighbor.in` file. The file will look like this:

  ```
  real
  real_1 real_2 real_3 ...
  ...
  ```

  If the word is `complex`, it means that not all the hopping integrals are real numbers. Then, each real hopping integral as described above should be substituted by two real numbers, the real and imaginary parts of the complex hopping integral. The file will look like this:

  ```
  complex
  real_1 imag_1 real_2 imag_2 real_3 imag_3 ...
  ...
  ```

  The unit of energy is determined by the user. One should consistently use the same unit in other input files such as `potential.in`, `energy.in` and `para.in`.

- The `potential.in` input file
  This is an optional input file, which contains the on-site potentials (the diagonal terms in the Hamiltonian). If this file is not prepared, it is assumed that all the on-site potentials are zero. The nth line is the on-site potential of the nth site.

- The `position.in` input file
  This file specifies the coordinates of the sites in the simulated system. This will be used to build the velocity (current) operator. The first line should have two numbers, which are the length of the simulated system in the transport direction and the volume of the system. Be careful with periodic boundary conditions. For example, consider a $1\ 000 \times 1\ 000$ regular square lattice with a lattice constant of $a = 1$, the length in the x direction should be 1 000, even though the distance between a leftmost site and a rightmost site is only 999. Starting from the second line, the nth line is the position component of the $(n-1)$th site in the transport direction. The unit of length is determined by the user. One can either set the lattice constant to 1 or some values in unit of nm or Å. What is important is to be consistent when reporting the results.

3.2.5 Other input files when using the lattice model

The general model approach is flexible, but is not convenient for "simple" systems. We therefore provide an alternative method for constructing the simulation model. In this "lattice model" method, one needs to prepare a file called `lattice.in` of the following form:

```
2500 2000 1          # Nx Ny Nz
1 1 0 0              # pbc_x pbc_y pbc_z transport_direction
1.7321 3 1           # ax ay az
4 3                  # N_orbital  N_hopping
0.866 0    0         # coordinates of orbital 0
0     0.5  0         # coordinates of orbital 1
0     1.5  0         # coordinates of orbital 2
0.866 2    0         # coordinates of orbital 3
3                    # there are 3 hoppings from orbital 0
1  0 0 1 -2.7 0      # hop to the orbital 1 in the positive x cell
0  0 0 1 -2.7 0      # hop to the orbital 1 in the current cell
0 -1 0 3 -2.7 0      # hop to the orbital 3 in the negative y cell
3                    # there are 3 hoppings from orbital 1
0  0 0 0 -2.7 0      # hop to the orbital 0 in the current cell
-1 0 0 0 -2.7 0      # hop to the orbital 0 in the negative x cell
0  0 0 2 -2.7 0      # hop to the orbital 2 in the current cell
3                    # there are 3 hoppings from orbital 2
0  0 0 3 -2.7 0      # hop to the orbital 3 in the current cell
-1 0 0 3 -2.7 0      # hop to the orbital 3 in the negative x cell
0  0 0 1 -2.7 0      # hop to the orbital 1 in the current cell
3                    # there are 3 hoppings from orbital 3
1 0 0 2 -2.7 0       # hop to the orbital 2 in the positive x cell
0 0 0 2 -2.7 0       # hop to the orbital 2 in the current cell
0 1 0 0 -2.7 0       # hop to the orbital 0 in the positive y cell
```

Here are some explanations:

- Line 1. `Nx` is the number of unit cells in the x direction.
- Line 1. `Ny` is the number of unit cells in the y direction.
- Line 1. `Nz` is the number of unit cells in the z direction.
- When simulating two-dimensional materials, one can set `Nz` to 1.
- Line 2. `pbc_x` specifies the boundary conditions in the x direction. It can be either 1 or 0, corresponding to periodic and open boundary conditions, respectively.
- Line 2. `pbc_y` is similar to `pbc_x`, but for the y direction.
- Line 2. `pbc_z` is similar to `pbc_x`, but for the z direction.
- When all three directions have periodic boundary conditions, the system is considered as a bulk material. When one direction is open, the system is considered as a film or a 2D material. When two directions are open, the system is considered as a wire or a ribbon. When all the directions are open, the system is considered as a dot, or a particle/cluster.
- Line 2. `transport_direction` specifies the transport direction in dissipative transport. It can be 0, 1, or 2, which corresponds to the x, y, and z directions, respectively. The transport direction must have periodic boundary conditions.
- Line 3. `ax` is the lattice constant in the x direction, which is the length of the unit cell in this direction. The units of the length are chosen by the user. The user need to make sure that all the length parameters are of the same units for a given simulation.
- Line 3. `ay` is the lattice constant in the y direction.
- Line 3. `az` is the lattice constant in the z direction.
- Line 4. `N_orbital` is the number of orbitals in the unit cell. The unit cell must be rectangular.
  - Example 1. In the single-orbital square lattice model, the smallest unit cell only contains one site, and `N_orbital` is 1.
  - Example 2. Consider graphene. The smallest rectangular unit cell contains 4 carbon atoms. When considering the p_z-orbital tight-binding model, each atom contributes one orbital. In this case, `N_orbital` is 4.
- Line 4. `N_hopping` is the maximum number of hoppings between one orbital and others.
  - Example 1. In the nearest-neighbor single-orbital square lattice model, `N_hopping` is 4.
  - Example 2. In the nearest-neigthbor p_z-orbital tight-binding model for graphene, `N_hopping` is 3.

- The next `N_orbital` lines gives the coordinates of the `N_orbital` orbitals in the unit cell. The first, second, and third columns correspond to the x, y, and z coordinates, respectively. Note the different orbitals can have the same coordinates. That is, one atom/site can have multiple orbitals.

- Starting from the next line, there are `N_orbital` blocks of data. Each block of data correspond to an orbital. For each block, there are one plus `N_hopping` lines, where the first line gives the number of hopping integrals M for the current orbital, and the remaining M lines give the details of the hopping integrals. To specify a hopping integral from the current orbital in the unit cell, we have to specify the (three-dimensional) cell index and orbital index of the neighboring orbital. The data format is as follows:

  ```
  nx ny nz n_orbital h_real h_imag
  ```

  This means that there is a hopping integtal with real part `h_real` and imaginary part `h_imag`, hopping to an orbital with index `n_orbital` in a unit cell that are offset by `nx`, `ny`, and `nz` compared to the current unit cell in the x, y, and z directions.

3.3 Output files of LSQT

We now describe the data format of the output files produced by running the code. We note that for all the output files, results from a new simulation will append to, rather than overwrite existing data.

3.3.1 The dos.out file

The nth column of this file corresponds to the value of $\rho(E_n)$ at the nth energy point E_n specified in the `energy.in` file. Each row corresponds to the results obtained by using one random vector. The unit of DOS is $1/\gamma/a^2$ in 2D and $1/\gamma/a^3$ in 3D, where γ is the unit of energy and a is the unit of length.

3.3.2 The vac.out file

The nth column of this file corresponds to the value of $\rho(E_n)C_{vv}(E_n,t)$ at the nth energy point E_n specified in the `energy.in` file. If the number of time steps specified in the `time_step.in` file is N_t, the first N_t rows correspond to the results obtained by using one random vector. Integrating this quantity with respect to time gives the running electrical conductivity. In 2D, the unit of conductivity is $e^2/\hbar$. In 3D, the unit is $e^2/\hbar/a$, where a is the unit of length. As expected, the unit in 3D can be converted to S/cm.

3.3.3 The vac0.out file

Similar to the`vac.out` file, but only for $\rho(E_n)C_{vv}(E_n,t=0)$.

3.3.4 The msd.out file

The nth column of this file corresponds to the value of $\rho(E_n)\Delta X^2(E_n, t)$ at the nth energy point E_n specified in the `energy.in` file. If the number of time steps specified in the `time_step.in` file is N_t, the first N_t rows correspond to the results obtained by using one random vector. Taking derivative of this quantity with respect to time and then dividing by 2 gives the running electrical conductivity.

Bibliography

Aron W Cummings, Jose H. García, Jaroslav Fabian, and Stephan Roche. Giant Spin Lifetime Anisotropy in Graphene Induced by Proximity Effects. *Physical Review Letters*, 119(20):206601, 2017. ISSN 0031-9007. doi: 10.1103/PhysRevLett.119.206601. URL https://link.aps.org/doi/10.1103/PhysRevLett.119.206601.

Z Fan, A Uppstu, T Siro, and A Harju. Efficient linear-scaling quantum transport calculations on graphics processing units and applications on electron transport in graphene. *Computer Physics Communications*, 185(1):28 – 39, 2014. ISSN 0010-4655. doi: https://doi.org/10.1016/j.cpc.2013.08.009. URL http://www.sciencedirect.com/science/article/pii/S0010465513002701.

Zheyong Fan, Jose Hugo Garcia, Aron W Cummings, Jose-Eduardo Barrios, Michel Panhans, Ari Harju, Frank Ortmann, and Stephan Roche. Linear Scaling Quantum Transport Methodologies. *arXiv preprint arXiv:1811.07387*, 2018a. URL https://arxiv.org/abs/1811.07387.

Zheyong Fan, Ville Vierimaa, and Ari Harju. GPUQT: An efficient linear-scaling quantum transport code fully implemented on graphics processing units. *Computer Physics Communications*, 230:113 – 120, 2018b. ISSN 0010-4655. doi: https://doi.org/10.1016/j.cpc.2018.04.013. URL http://www.sciencedirect.com/science/article/pii/S0010465518301280.

Jose H. García, Lucian Covaci, and Tatiana G. Rappoport. Real-Space Calculation of the Conductivity Tensor for Disordered Topological Matter. *Phys. Rev. Lett.*, 114:116602, Mar 2015. doi: 10.1103/PhysRevLett.114.116602. URL https://link.aps.org/doi/10.1103/PhysRevLett.114.116602.

Frank Ortmann, Nicolas Leconte, and Stephan Roche. Efficient linear scaling approach for computing the Kubo Hall conductivity. *Phys. Rev. B*, 91:165117, Apr 2015. doi: 10.1103/PhysRevB.91.165117. URL https://link.aps.org/doi/10.1103/PhysRevB.91.165117.

S. Roche and D. Mayou. Conductivity of Quasiperiodic Systems: A Numerical Study. *Phys. Rev. Lett.*, 79:2518–2521, Sep 1997. doi: 10.1103/PhysRevLett.79.2518. URL https://link.aps.org/doi/10.1103/PhysRevLett.79.2518.

H. Tal-Ezer and R. Kosloff. An accurate and efficient scheme for propagating the time dependent Schrödinger equation. *The Journal of Chemical Physics*, 81(9):3967–3971, 1984. doi: 10.1063/1.448136. URL https://doi.org/10.1063/1.448136.

Dinh Van Tuan, Frank Ortmann, David Soriano, Sergio O. Valenzuela, and Stephan Roche. Pseudospin-driven spin relaxation mechanism in graphene. *Nature Physics*,

10:857–863, 2014. doi: 10.1038/nphys3083. URL http://dx.doi.org/10.1038/nphys3083.

V Vierimaa, Z Fan, and A Harju. Scattering from spin-polarized charged impurities in graphene. *Phys. Rev. B*, 95:041401, Jan 2017. doi: 10.1103/PhysRevB.95.041401. URL https://link.aps.org/doi/10.1103/PhysRevB.95.041401.

Alexander Weiße, Gerhard Wellein, Andreas Alvermann, and Holger Fehske. The kernel polynomial method. *Rev. Mod. Phys.*, 78:275–306, Mar 2006. doi: 10.1103/RevModPhys.78.275. URL https://link.aps.org/doi/10.1103/RevModPhys.78.275.