Quantum Lattice Model Solver $\mathcal{H}\Phi$

Youhei Yamaji[a,*], Takahiro Misawa[b], Synge Todo[b,1], Mitsuaki Kawamura[b], Kazuyoshi Yoshimi[b], Naoki Kawashima[b]

[a]*Quantum-Phase Electronics Center (QPEC), The University of Tokyo, Bunkyo-ku, Tokyo, 113-8656, Japan*
[b]*The Institute for Solid State Physics, The University of Tokyo, Kashiwa-shi, Chiba, 277-8581,Japan*
[c]*Department of Physics, The University of Tokyo, Bunkyo-ku, Tokyo, 113-8656, Japan*

Abstract

$\mathcal{H}\Phi$ is a parallelized diagonalization package for solving many-body quantum lattice hamiltonians with arbitrary one-body potentials and two-body interactions.

Keywords: Lanczos method; Thermal pure quantum states; Hubbard model; Heisenberg model; Kondo model;

PROGRAM SUMMARY
Manuscript Title:
Authors:
Program Title: $\mathcal{H}\Phi$
Journal Reference:
Catalogue identifier:
Program summary URL: `http://ma.cms-initiative.jp/en/application-list/hphi?set_languag`
Licensing provisions: GNU General Public Licence (GPLv3)
Programming language: C
Computer: Any architecture with suitable compilers including PCs and clusters.
Operating system: Unix, Linux, OSX.
RAM: bytes
Number of processors used:
Supplementary material:
Keywords: Keyword one, Keyword two, Keyword three, etc.

[*]Corresponding author.
E-mail address: firstAuthor@somewhere.edu

Classification:
External routines/libraries:
Subprograms used:
*Catalogue identifier of previous version:**
*Journal reference of previous version:**
*Does the new version supersede the previous version?:**
Nature of problem:

Solution method:

*Reasons for the new version:**

*Summary of revisions:**

Restrictions:

Unusual features:

Additional comments:

Running time:

[1] Reference 1

[2] Reference 2

[3] Reference 3

* Items marked with an asterisk are only required for new versions of programs previously published in the CPC Program Library.

1. Introduction

Comparison between experimental observation and theoretical analysis is a crucial step in condensed-matter physics research. Temperature dependence of specific heat and magnetic susceptibility, for example, have been studied to extract nature of low energy excitations of and magnetic interactions among electrons, respectively, through comparison with theories such as Landau's Fermi liquid theory and Curie-Weiss law.

For the flexible and quantitative comparison with experimental data, an exact diagonalization approach [1] is one of the most reliable numerical tools without any approximation or inspiration of genius. For last few decades, a numerical diagonalization package for quantum spin hamiltonians, TITPACK developed by Prof. Hidetoshi Nishimori in Tokyo Institute of Technology, has been widely used in the condensed-matter physics community. [Other apps for exact diagonalization methods] Nevertheless, limitation of computational resources had hindered the non-expert users from applying the package to quantum systems with large number of electrons or spins.

In contrast, recent and rapid development of parallel computing infrastructure opens up new avenues for user-friendly larger scale diagonalizations up to 18 site Hubbard clusters or 36 S=1/2 quantum spins. In addition, recent advances in quantum statistical mechanics [2, 3, 4, 5] enable us to calculate finite temperature properties of quantum many-body systems with computational costs similar to calculations of ground state properties, which also enables us to compare theoretical results for temperature dependence of, for example, specific heat and magnetic susceptibility with experimental results quantitatively [6]. To utilize the parallel computing infrastructure with narrow bandwidth and distributed-memory architectures, efficient, user-friendly, and highly parallelized diagonalization packages are highly desirable.

$\mathcal{H}\Phi$, a flexible diagonalization package for solving quantum lattice hamiltonians, has been developed to be such a descendant of the pioneering package TITPACK. The Lanczos method for calculations of the ground state and a few excited states properties, and finite temperature calculations based on thermal pure quantum states [5] are implemented in the package $\mathcal{H}\Phi$, with an easy-to-use and flexible user interface. By using $\mathcal{H}\Phi$, you can analyze a wide range of quantum lattice hamiltonians including simple Hubbard and Heisenberg models, multi-band extensions of the Hubbard model, exchange couplings that break SU(2) symmetry of quantum spins such as Dzyaloshinskii-Moriya and Kitaev interactions, and Kondo lattice models describing itinerant electrons coupled with quantum spins. $\mathcal{H}\Phi$ calculates a variety of physical quantities such as internal energy at zero temperature or finite temperatures, temperature dependence of specific heat, charge/spin structure factors, and so on. A broad spectrum of users including experimental scientists is cordially welcome.

2. What $\mathcal{H}\Phi$ does

User can choose algorithms for diagonalization (Lapack, Lanczos) or finite temperatures (Lapack, TPQ), the Hilbert space (with/without particle number/S_z conservation), and physical quantities calculated after/during the Lanczos steps.

2.1. Calculation mode

2.2. List of input and output files

2.3. Physical quantities

3. Hamiltonian

3.1. Standard models

$$H = -\mu \sum_{i\sigma} c_{i\sigma}^{\dagger} c_{i\sigma} - \sum_{i \neq j \sigma} t_{ij} c_{i\sigma}^{\dagger} c_{j\sigma} + \sum_{i} U n_{i\uparrow} n_{i\downarrow} + \sum_{i \neq j} V_{ij} n_i n_j, \tag{1}$$

$$\begin{aligned} H &= -h \sum_{i} S_{iz} + \Gamma \sum_{i} S_{ix} + D \sum_{i} S_{iz} S_{iz} \\ &\quad + \sum_{ij} \left(J_{ijx} S_{ix} S_{jx} + J_{ijy} S_{iy} S_{jy} + J_{ijz} S_{iz} S_{jz} \right), \end{aligned} \tag{2}$$

$$H = -\mu \sum_{i\sigma} c_{i\sigma}^{\dagger} c_{i\sigma} - t \sum_{\langle ij \rangle \sigma} c_{i\sigma}^{\dagger} c_{j\sigma} + \frac{J}{2} \sum_{i} \left\{ S_i^{+} c_{i\downarrow}^{\dagger} c_{i\uparrow} + S_i^{-} c_{i\uparrow}^{\dagger} c_{i\downarrow} + S_{iz} (n_{i\uparrow} - n_{i\downarrow}) \right\}, \tag{3}$$

3.2. Generalized Hamiltonians

Algorithm

4. Lanczos method

4.1. Algoritm of Lanczos method

Some parts of this section are based on the manual of titpack [7] and textbook by M. Sugihara and K. Murota [8] (These references are written in Japanese).

In the Lanczos method, by successively operating the Hamiltonian to the initial vector, we obtain the accurate eigenvalues around the maximum and minimum eigenvalues and associated eigenvectors. Because we can perform the Lanczos method by using only two vectors whose dimensions are the dimension of the total Hilbert space [1], Lanczos method is frequently used for the diagonalization of the large matrices where the full diagonalization is impossible. As we detail below, to obtain the eigenvector by the Lanczos method, one additional vector is necessary.

The principle of the Lanczos method is based on the power method. In the power method, by successively operating the Hamiltonian $\hat{\mathcal{H}}$ to the arbitrary vector $\boldsymbol{x}_0$, we generate $\hat{\mathcal{H}}^n\boldsymbol{x}_0$. The obtained space $\mathcal{K}_{n+1}(\hat{\mathcal{H}},\boldsymbol{x}_0)=\{\boldsymbol{x}_0,\hat{\mathcal{H}}^1\boldsymbol{x}_0,\ldots,\hat{\mathcal{H}}^n\boldsymbol{x}_0\}$ is called Krylov subspace. Initial vector is represented by the superposition of the eigenvectors $\boldsymbol{e}_i$ (corresponding eigenvalues are E_i) of $\hat{\mathcal{H}}$ as

$$\boldsymbol{x}_0=\sum_i a_i\boldsymbol{e}_i. \tag{4}$$

Here, E_0 is maximum absolute values of the eigenvalues. We note that all the eigenvalues are real number because Hamiltonian is Hermite. By operating $\hat{\mathcal{H}}^n$ to the initial vector, we obtain the relation as

$$\hat{\mathcal{H}}^n\boldsymbol{x}_0=E_0^n\Big[a_0\boldsymbol{e}_0+\sum_{i\neq 0}\left(\frac{E_i}{E_0}\right)^n a_i\boldsymbol{e}_i\Big]. \tag{5}$$

This relation shows that the eigenvector of E_0 becomes dominant for sufficiently large n. In the Lanczos method, we obtain the eigenvalues and eigenvectors by performing the proper transformation for obtained Krylov subspace.

In the Lanczos method, we successively generate the normalized orthogonal basis $\boldsymbol{v}_0,\ldots,\boldsymbol{v}_{n-1}$ from the Krylov subspace $\mathcal{K}_n(\hat{\mathcal{H}},\boldsymbol{x}_0)$. We defines initial vector and associated components as $\boldsymbol{v}_0=\boldsymbol{x}_0/|\boldsymbol{x}_0|$, $\beta_0=0,\boldsymbol{x}_{-1}=0$. From this initial condition, we can obtain the normalized orthogonal basis

[1] In $\mathcal{H}\Phi$, to reduce the numerical cost, we use some additional vectors; vector for accumulating the real-space diagonal elements of the Hamiltonian, vector for specifying the given S_z space and given particle space. The dimension of these vectors is that of the Hilbert space.

as follows:

$$\alpha_k = (\hat{\mathcal{H}}\boldsymbol{v}_k, \boldsymbol{v}_k), \tag{6}$$

$$\boldsymbol{w} = \hat{\mathcal{H}}\boldsymbol{v}_k - \beta_k \boldsymbol{v}_{k-1} - \alpha_k \boldsymbol{v}_k, \tag{7}$$

$$\beta_{k+1} = |\boldsymbol{w}|, \tag{8}$$

$$\boldsymbol{v}_{k+1} = \frac{\boldsymbol{v}_k}{|\boldsymbol{v}_k|}. \tag{9}$$

From these definitions, it it obvious that α_k, β_k are real number.

In the subspace spanned by these normalized orthogonal basis, the Hamiltonian is transformed as

$$T_n = V_n^{\dagger}\hat{\mathcal{H}}V_n. \tag{10}$$

Here, V_n is matrix whose column vectors are $\boldsymbol{v}_i (i = 0, 1, \ldots, n-1)$. T_n is tridiagonal matrix and its diagonal elements are α_i and subdiagonal elements are β_i. It is known that the eigenvalues of $\hat{\mathcal{H}}$ are well approximated by the eigenvalues of T_n for sufficiently large n. (We note that $V^{\dagger}V = I$, I is identity matrix). The original eigenvectors of $\hat{\mathcal{H}}$ is obtained by $\boldsymbol{e}_i = V\tilde{\boldsymbol{e}}_i$, where $\tilde{\boldsymbol{e}}_i$ are the eigenvectors of T_n. From V, we can obtain the eigenvectors of $\hat{\mathcal{H}}$ by performing the Lanczos method. However, in the actual calculations, it is difficult to keep V because its dimension is large [dimension of V = (dimension of the total Hilbert space) $\times$ (# of Lanczos iterations)]. Thus, to obtain the eigenvectors, we again perform the same Lanczos calculations after we obtain the eigenvalues from the Lanczos methods. In the first Lanczos calculation, we keep $\tilde{\boldsymbol{e}}_i$ because its dimension is small. [2]. From this procedure, we obtain the eigenvectors from V.

In the Lanczos method, within a few hundred or thousand Lanczos iterations, we obtain the accurate eigenvalues near the maximum and minimum values of eigenvalues. The necessary number of iterations is small enough compared to the dimensions of the total Hilbert space. We note that it is shown that the errors of the maximum and minimum eigenvalues becomes exponentially small as a function of Lanczos iteration (for details, see Ref. [8]).

4.2. Inverse iteration method

From the approximate value of the eigenvalues (E_n), by successively operating $(\hat{\mathcal{H}} - E_n)^{-1}$ to the initial vector $\boldsymbol{y}_0$, we can obtain the accurate eigenvector for E_n.

[2]upper bound of the dimensions of $\tilde{\boldsymbol{e}}_i$ is # of Lanczos iterations.

From $(\hat{\mathcal{H}} - E_n)^{-1}\boldsymbol{y}_0$, we obtain the linear simultaneous equations such as

$$\boldsymbol{y}_k = (\hat{\mathcal{H}} - E_n)\boldsymbol{y}_{k+1}. \tag{11}$$

By solving this equation by using the conjugate gradient method (CG method), we obtain the eigenvector. From the obtained eigenvector, we can calculate the eigenvalues and correlation functions. We note that additional four vectors are necessary to perform the CG method.

4.3. Details of implementation

Initial vector

In the Lanczos method, an initial vector is specified with `initial_iv`($\equiv r_s$) defined in an input file for Standard mode or a ModPara file for Expert mode. A type of an initial vector can be selected from a real number or complex number by using `InitialVecType` in a ModPara file.

- For canonical ensemble and `initial_iv` ≥ 0

  A component of a target of Hilbert space is given by

  $$(N_{\rm dim}/2 + r_s)\%N_{\rm dim}, \tag{12}$$

  where $N_{\rm dim}$ is a total number of the Hilbert space and $N_{\rm dim}/2$ is added to avoid selecting the special configuration for a default value `initial_iv` $= 1$. When a type of an initial vector is selected as a real number, a coefficient value is given by 1, while as a complex number, the value is given by $(1 + i)/\sqrt{2}$.

- For grand canonical ensemble or `initial_iv` < 0

  An initial vector is given by using a random generator, i.e. coefficients of all components for the initial vector is given by random numbers. The seed is calculated as

  $$123432 + |r_s|, \tag{13}$$

  where r_s is a number given by an input file and $n_{\rm run}$ is a number of runs. A maximum value of $n_{\rm run}$ is defined by `NumAve` in an input file for Standard mode or a ModPara file for Expert mode. Random numbers are generated by using SIMD-oriented Fast Mersenne Twister (dSFMT)[**?**].

Convergence condition

In $\mathcal{H}\Phi$, we use `dsyev` (routine of lapack) for diagonalization of T_n. We use the energy of the first excited state of T_n as a criteria of convergence. In the standard setting, after five Lanczos step, we diagonalize T_n every two Lanczos step. If the energy of the first excited states agrees with the previous energy within the specified accuracy, the Lanczos iteration finishes. The accuracy of the convergence can be specified by `CDataFileHead` (ModPara file in the expert mode).

After obtaining the eigenvalues, we again perform the Lanczos iteration to obtain the eigenvector. From the eigenvectors $|n\rangle$, we calculate energy $E_n = \langle n|\hat{\mathcal{H}}|n\rangle$ and variance $\Delta = \langle n|\hat{\mathcal{H}}^2|n\rangle - (\langle n|\hat{\mathcal{H}}|n\rangle)^2$. If E_n agrees with the eigenvalues obtained by the Lanczos iteration and Δ is smaller than the specified value, we finish the diagonalization.

If the accuracy of Lanczos method is not enough, we perform the CG method to obtain the eigenvector. As an initial vector of the CG method, we use the eigenvectors obtained by the Lanczos method in the standard setting. This often accelerates the convergence.

5. Full Diagonalization method

5.1. Over view

We generate matrix of $\hat{H}$ by using the real space configuration $|\psi_j\rangle(j = 1 \cdots d_{\mathrm{H}}$, d_{H} is dimension of the Hilbert space): $H_{ij} = \langle\psi_i|\hat{H}|\psi_j\rangle$. By diagonalizing this matrix, we can obtain all the eigenvalues E_i and eigenvectors $|\Phi_i\rangle$ $(i = 1 \cdots d_{\mathrm{H}})$. In the diagonalization, we use lapack routine such as `dsyev` or `zheev`. We also calculate and out put the expectation values $\langle A_i\rangle \equiv \langle\Phi_i|\hat{A}|\Phi_i\rangle$. These values are used for the finite-temperature calculations.

5.2. Finite-temperature calculations

From $\langle A_i\rangle \equiv \langle\Phi_i|\hat{A}|\Phi_i\rangle$, we calculate finite-temperature properties by using the relation

$$\langle\hat{A}\rangle = \frac{\sum_{i=1}^{N}\langle A_i\rangle \mathrm{e}^{-\beta E_i}}{\sum_{i=1}^{N}\mathrm{e}^{-\beta E_i}}. \tag{14}$$

In the actual calculation are performed as the post scripts.

6. Finite-temperature calculations by TPQ method

Sugiura and Shimizu show that it is possible to calculate the finite-temperature properties from a few wavefunctions (in the thermodynamic limit, only one wave function is necessary) [5]. The wavefunction is called thermal pure quantum (TPQ) state. Because TPQ state can be generated by operating the Hamiltonian to the random initial wavefunction, we directly use the routine Lanczos method to the TPQ calculations. Here, we explain how to construct micro canonical TPQ (mTPQ) state, which offers the simplest way for finite-temperature calculations.

Let $|\psi_0\rangle$ be a random initial vector. By operating $(l - \hat{H}/N_s)^k$ (l is constant, N_s represents number of sites) to $|\psi_0\rangle$, we obtain the kth TPQ states as

$$|\psi_k\rangle \equiv \frac{(l - \hat{H}/N_s)|\psi_{k-1}\rangle}{|(l - \hat{H}/N_s)|\psi_{k-1}\rangle|}. \tag{15}$$

From $|\psi_k\rangle$, we estimate corresponding inverse temperature β_k as

$$\beta_k \sim \frac{2k/N_s}{l - u_k}, \quad u_k = \langle\psi_k|\hat{H}|\psi_k\rangle/N_s, \tag{16}$$

where u_k is the internal energy. Arbitrary local physical properties at β_k is also estimated as

$$\langle\hat{A}\rangle_{\beta_k} = \langle\psi_k|\hat{A}|\psi_k\rangle/N_s. \tag{17}$$

In finite-size system, error is caused by the choice of the initial random vector. To estimate the average value and error of the physical properties, we perform some independent calculations by changing $|\psi_0\rangle$.

6.1. Details of implementation

Initial vector

In the TPQ method, an initial vector is given by using a random generator, i.e. coefficients of all components for the initial vector is given by random numbers. The seed is calculated as

$$123432 + (n_{\rm run} + 1) \times |r_s|, \tag{18}$$

where r_s is a number given by an input file and $n_{\rm run}$ is a number of runs. r_s and a maximum value of $n_{\rm run}$ are defined by `initial_iv` and `NumAve` in an input file for Standard mode or a ModPara file for Expert mode, respectively. Random numbers are generated by using SIMD-oriented Fast Mersenne Twister (dSFMT)[**?**]. We can select a type of initial vector from a real number or complex number by using `InitialVecType` in a ModPara file.

7. Physical quantities

7.1. Internal energy and variance

7.1.1. Ground state

7.1.2. Finite temperatures

7.2. Green's functions

8. Parallelization

8.1. Butterfly

8.2. Continuous access

8.3. Green's functions

9. Performance

10. Getting started

10.1. Prerequisite

$\mathcal{H}\Phi$ requires the following packages:

- C compiler (intel, Fujitsu, GNU, etc.)
- LAPACK library (intel MKL, Fujitsu TCL, ATLAS, etc.)
- MPI library (If you do not use MPI, it is not necessary.)

Tips

E. g. / Settings of intel compiler

When you use the intel compiler, you can use easily scripts attached to the compiler. In the case of the bash in 64 bit OS, write the following in your `~/.bashrc`:

```
source /opt/intel/bin/compilervars.sh intel64
```

or

```
source /opt/intel/bin/iccvars.sh intel64
source /opt/intel/mkl/bin/mklvars.sh
```

Please read manuals of your compiler/library for more information.

10.2. Installation

You can download $\mathcal{H}\Phi$ in the following place.
`https://github.com/QLMS/HPhi/releases`
You can obtain the $\mathcal{H}\Phi$ directory by tiping

```
$ tar xzvf HPhi-xxx.tar.gz
```

There are two kind of procedures to install $\mathcal{H}\Phi$.

10.2.1. Using `HPhiconfig.sh`

Please run `HPhiconfig.sh` script in the $\mathcal{H}\Phi$ directory as follow (for ISSP system-B "sekirei"):

```
$ bash HPhiconfig.sh sekirei
```

Then environmental configuration file `make.sys` is generated in `src/` directory. The command-line argment of `HPhiconfig.sh` is as folows:

- `sekirei` : ISSP system-B "sekirei"
- `maki` : ISSP system-C "maki"
- `intel` : intel compiler + Linux PC
- `mpicc-intel` : intel compiler + MPI (excepting intelMPI) + Linux PC

- `gcc` : GCC + Linux PC
- `gcc-mac` : GCC + Mac

`make.sys` is as follows (for ISSP-system-B ”sekirei”):

```
CC = mpiicc
LAPACK_FLAGS = -Dlapack -mkl=parallel
FLAGS = -qopenmp  -O3 -xCORE-AVX2 -mcmodel=large -shared-intel -D MPI
MTFLAGS = -DDSFMT_MEXP=19937 $(FLAGS)
INCLUDE_DIR=./include
```

We explatin macros of this file as:

- `CC` : The Compilation command (`icc`, `gcc`, `fccpx`)
- `LAPACK_FLAGS` : Compilation options for LAPACK. `-Dlapack` can not be removed.
- `FLAGS` : Other compilation options. OpenMP utilization option (`-openmp`, `-fopenmp`, `-qopenmp`, etc.) must be specified. When you use MPI, please set `-D MPI`.
- `MTFLAGS`, `INCLUDE_DIR` : Options for the Mersenne Twister and additional include directory. You do not have to modify them.

Then you are ready to compile HPhi. Please type

```
$ make HPhi
```

and obtain an executable `HPhi` in `src/` directory; you should add this directory to the `$PATH`.

Tips

You can make a PATH to $\mathcal{H}\Phi$ as follows:
`$ export PATH=${PATH}:`*HPhi_top_directory*`/src/`
If you keep this PATH, you should write above in `~/.bashrc` (for `bash` as a login shell)

10.2.2. Using `cmake`

Tips

Before using cmake for sekirei, you must type

```
source /home/issp/materiapps/tool/env.sh
```

while for maki, you must type

```
source /global/app/materiapps/tool/env.sh
```

We can compile Hphi as

```
cd $HOME/build/hphi
cmake -DCONFIG=gcc $PathTohphi
make
```

Here, we set a path to $\mathcal{H}\Phi$ as `$PathTohphi` and to a build directory as `$HOME/build/hphi`. After compiling, a src folder is constructed below a `$HOME/build/hphi` folder and obtain an executable `HPhi` in `src/` directory. When there is not a MPI library in your system, an executable `HPhi` is automatically compiled without a MPI library.

In the above example, we compile $\mathcal{H}\Phi$ by using a gcc compiler. We can select a compiler by using following options

- `sekirei` : ISSP system-B "sekirei"
- `fujitsu` : Fujitsu compiler (ISSP system-C "maki")
- `intel` : intel compiler + Linux PC
- `gcc` : GCC compiler + Linux PC.

An example for compiling $\mathcal{H}\Phi$ by an intel compiler is shown as follows,

```
mkdir ./build
cd ./build
cmake -DCONFIG=intel ../
make
```

After compiling, a `src` folder is made below the `build` folder and an execute $\mathcal{H}\Phi$ is made in the `src` folder. It is noted that we must delete the `build` folder and do the above works again when we change the compilers.

10.3. Directory structure

When HPhi-xxx.tar.gz is unzipped, the following directory structure is composed.

```
├──── CMakeLists.txt
├──── COPYING
├──── config/
│       ├──── fujitsu.cmake
│       ├──── gcc.cmake
│       ├──── intel.cmake
│       └──── sekirei.cmake
├──── doc/
│       ├──── en/
│       ├──── jp/
│       ├──── userguide_en.pdf
│       └──── userguide_jp.pdf
├──── HPhiconfig.sh
├──── samples/
│      ├──── Expert/
│      │      ├──── Hubbard/
│      │      │      ├── square/
│      │      │      │      ├────*.def
│      │      │      │      ├──── output_FullDiag/
│      │      │      │      │         └────**.dat
│      │      │      │      ├──── output_Lanczos/
│      │      │      │      │         └────**.dat
│      │      │      │      └──── output_TPQ/
│      │      │      │               └────**.dat
│      │      │      └── triangular/
│      │      │             └──── . . .
│      │      ├──── Kondo/
│      │      │      └── chain/
│      │      │             └──── . . .
│      │      └──── Spin/
│      │             ├── HeisenbergChain/
│      │             │      └──── . . .
│      │             ├── HeisenbergSquare/
│      │             │      └──── . . .
```

```
|      |           ├── Kagome/
|      |           |     └──── ...
|      |           ├── Kitaev/
|      |           |     └──── ...
|      |           └── S1Chain/
|      |                 └──── ...
|      └───── Standard/
|              ├───── Hubbard/
|              |       ├── square/
|              |       |     ├───── StdFace.def
|              |       |     ├───── output_FullDiag/
|              |       |     |        └────**.dat
|              |       |     ├───── output_Lanczos/
|              |       |     |        └────**.dat
|              |       |     └───── output_TPQ/
|              |       |              └────**.dat
|              |       └── triangular/
|              |             └──── ...
|              ├───── Kondo/
|              |       └── chain/
|              |             └──── ...
|              └───── Spin/
|                        ├── HeisenbergChain/
|                        |     └──── ...
|                        ├── HeisenbergSquare/
|                        |     └──── ...
|                        ├── Kitaev/
|                        |     └──── ...
|                        └── S1Chain/
|                               └──── ...
└───── src/
         ├─────**.c
         ├───── CMakeLists.txt
         ├───── include/
         |       └────**.h
         └───── makefile_src
```

10.4. Basic usage

$\mathcal{H}\Phi$ has two modes; standard mode and expert mode. Here, the basic flows of calculations by standard and expert modes are shown.

10.4.1. Standard mode

The procedure of calculation through the standard mode is shown as follows:

1. Make a directory for a calculation scenario.
   First, you make a working directory for the calculation.
2. Make input files for standard mode
   In the standard mode, you can choose a model (the Heisenberg model, the Hubbard model, etc.) and a lattice (the square lattice, the triangular lattice, etc.) from ones provided; you can specify some parameters (such as the first/second nearest neighbor hopping integrals, the on-site Coulomb integral, etc.) for them. Finally, you have to specify the numerical method (such as the Lanczos method) employed in this calculation. The input file format is described in the Sec. **??**.
3. Run
   Run a executable `HPhi` in terminal by setting option "`-s`" (or "`--standard`") and the name of input file written in previous step.

   - Serial/OpenMP parallel

     `$` *Path*`/HPhi -s` *Input_file_name*

   - MPI parallel/ Hybrid parallel

     `$ mpiexec -np` *number_of_processes* *Path*`/HPhi -s` *Input_file_name*

     When you use a queuing system in workstations or super computers, sometimes the number of processes is specified as an argument for the job-submitting command. If you need more information, please refer manuals for your system. A number of processes depend on a target of system for models. The details of setting a number of processes are shown in 10.4.3.

4. Watch calculation logs
   Log files are outputted in the "output" folder which is automatically made in the directory for a calculation scenario. The details of output files are shown in **??**.

5. Results
   If the calculation is finished normally, the result files are outputted in the "output" folder. The details of output files are shown in **??**.

Tips

The number of threads for OpenMP

If you specify the number of OpenMP threads for $\mathcal{H}\Phi$, you should set it as follows (in case of 16 threads) before the running:

```
export OMP_NUM_THREADS=16
```

10.4.2. Expert mode

The procedure of calculation for expert mode is shown as follows.

1. Make a directory for a calculation scenario.
   First, you make a directory named as a calculation scenario (you can attach an arbitrary name to a directory).
2. Make input files for expert mode
   For expert mode, you should make input files for constructing Hamiltonian operators, calculation condition and a list file for the filenames of input files (see the file formats shown in **??**).
   Note: A List file can be easily made by using standard mode.
3. Run
   Run $\mathcal{H}\Phi$ in terminal by setting option "`-e`" (or "`--expert`") and a file name of a list file.

   - Serial/OpenMP
     `$` *Path*`/HPhi -e` *Input_List_file_name*
   - MPI/Hybrid
     `$ mpiexec -np` *number_of_processes* *Path*`/HPhi -e` *Input_List_file_name*
     A number of processes depend on a target of system for models. The details of setting a number of processes are shown in 10.4.3.

4. Under running
   Log files are outputted in the "output" folder which is automatically made in the directory for a calculation scenario. The details of output files are shown in **??**.

5. Results
   If the calculation is finished normally, the result files are outputted in the “output” folder. The details of output files are shown in **??**.

10.4.3. Setting a process number for MPI/Hybrid parallelization

For using MPI/Hybrid parallelization、a process number must be set as follows.

1. Standard mode
   - Hubbard / Kondo model
     When `model` in an input file for Standard mode is set as `"Fermion Hubbard"`, `"Kondo Lattice"` or `"Fermion HubbardGC"`, a process number must be equal to 4^n.
   - Spin model
     When `model` in an input file for Standard mode is set as `"Spin"` or `"SpinGC"`, a process number must be equal to $(2S+1)^n$ where `2S` is set in the input file (a default value is 1).
2. Expert mode
   - Hubbard / Kondo model

     When a model is selected as fermion Hubbard model or Kondo model by setting `CalcModel` in a **CalcMod** file, a process number must be equal to 4^n. See **??** for details of a `CalcModel` file.
   - Spin model

     When a model is selected as Spin model by setting `CalcModel` in a **CalcMod** file, a process number is fixed by a **LocSpin** file. A process number must be equal to a number calculated by multiplying a state number of localized spin (2S+1) in descending order about site numbers. See **??** for details of a **LocSpin** file.

     For example, when a **LocSpin** file is given as follws, a process number must be equal to $2 = 1+1,\ 6 = 2\times(2+1),\ 24 = 6\times(3+1)$.

```
================================
NlocalSpin     3
================================
========i_0IteElc_2S ======
================================
    0      3
    1      2
    2      1
```

[1] E. Dagotto, Correlated electrons in high-temperature superconductors, Rev. Mod. Phys. 66 (1994) 763–840. doi:10.1103/RevModPhys.66.763.
URL `http://link.aps.org/doi/10.1103/RevModPhys.66.763`

[2] M. Imada, M. Takahashi, Quantum transfer monte carlo method for finite temperature properties and quantum molecular dynamics method for dynamical correlation functions, Journal of the Physical Society of Japan 55 (10) (1986) 3354–3361. arXiv:http://dx.doi.org/10.1143/JPSJ.55.3354, doi:10.1143/JPSJ.55.3354.
URL `http://dx.doi.org/10.1143/JPSJ.55.3354`

[3] J. Jaklič, P. Prelovšek, Lanczos method for the calculation of finite-temperature quantities in correlated systems, Phys. Rev. B 49 (1994) 5065–5068. doi:10.1103/PhysRevB.49.5065.
URL `http://link.aps.org/doi/10.1103/PhysRevB.49.5065`

[4] A. Hams, H. De Raedt, Fast algorithm for finding the eigenvalue distribution of very large matrices, Phys. Rev. E 62 (2000) 4365–4377. doi:10.1103/PhysRevE.62.4365.
URL `http://link.aps.org/doi/10.1103/PhysRevE.62.4365`

[5] S. Sugiura, A. Shimizu, Thermal pure quantum states at finite temperature, Phys. Rev. Lett. 108 (2012) 240401. doi:10.1103/PhysRevLett.108.240401.
URL `http://link.aps.org/doi/10.1103/PhysRevLett.108.240401`

[6] Y. Yamaji, Y. Nomura, M. Kurita, R. Arita, M. Imada, First-principles study of the honeycomb-lattice iridates na_2iro_3 in the presence of strong spin-orbit interaction and electron correlations, Phys. Rev. Lett. 113 (2014) 107201. doi:10.1103/PhysRevLett.113.107201.
URL `http://link.aps.org/doi/10.1103/PhysRevLett.113.107201`

[7] http://www.stat.phys.titech.ac.jp/ nishimori/titpack2_new/index-e.html.

[8] M. Sugihara, K. Murota, Theoretical Numerical Linear Algebra, Iwanami Studies in Advanced Mathematics, Iwanami Shoten, Publishers, 2009.