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

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

[a] *The Institute for Solid State Physics, The University of Tokyo, Kashiwa-shi, Chiba, 277-8581,Japan*
[b] *Department of Physics, The University of Tokyo, Bunkyo-ku, Tokyo, 113-8656, Japan*
[c] *Quantum-Phase Electronics Center (QPEC), 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. A wide range of quantum lattice hamiltonians such as Hubbard and Heisenberg models, exchange couplings that break SU(2) symmetry of quantum spins, and Kondo lattice models describing itinerant electrons coupled with quantum spins can be analyzed. $\mathcal{H}\Phi$ 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.

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

PROGRAM SUMMARY

Manuscript Title: Quantum Lattice Model Solver $\mathcal{H}\Phi$
Authors: Mitsuaki Kawamura, Kazuyoshi Yoshimi, Takahiro Misawa, Youhei Yamaji, Synge Todo, Naoki Kawashima
Program Title: $\mathcal{H}\Phi$
Journal Reference:
Catalogue identifier:
Program summary URL: http://ma.cms-initiative.jp/en/application-list/hphi?set_language=en
Licensing provisions: GNU General Public License, version 3

*Corresponding author.
E-mail address: mkawamura@issp.u-tokyo.ac.jp

Programming language: C
Computer: Any architecture with suitable compilers including PCs and clusters.
Operating system: Unix, Linux, OS X.
RAM: bytes
Number of processors used:
Keywords: Full diagonalization method, Exact diagonalization method, Lanczos method, Thermal pure quantum states, quantum lattice model.
Classification:
External routines/libraries: MPI, LAPACK
Nature of problem:
This package has the purpose of computing $\cdots$.
Solution method:
Software application based on full diagonalization method, exact diagonalization using Lanczos method, microcanonical thermal pure quantum state for quantum lattice model such as Hubbard model, Heisenberg model and Kondo model.
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. Overview of $\mathcal{H}\Phi$

2.1. Target model

2.2. Method

2.3. Physical quantities

2.4. Download and Install

2.5. Calculation flow

The following 4 steps are done in $\mathcal{H}\Phi$ for actual calculations.
Step 1. Read input files $\mathcal{H}\Phi$ has two calculation modes; standard mode and expert mode. The difference between expert and standard modes is the format of input files. For Expert mode, the files categorized by following four parts are needed.
(1) Parameter files for specifying the model to give the transfer integrals, interactions and electronic state (local spin or itinerant electron state).
(2) Parameter files for specifying the calculation condition such as the calculation method, the target of model (Spin, Kondo or electronic model under canonical/ grand canonical ensemble), electron number and the conversion condition.
(3) Parameter files for target correlated functions (one body Green's functions and two body Green's functions) to calculate.
(4) File for giving name of above input files.

For Standard mode, $\mathcal{H}\Phi$ makes input files for Expert mode by using an input file which gives the basic information such as lattice type, model, method, the strength of transfer integrals and interactions, and the conditions of calculations.

Step 2. Make target Hilbert space

The target model is specified by the input file for calculation condition. $\mathcal{H}\Phi$ represents the Hilbert space for each models by bits. For electronic system, the local site has four states and thus the state can be represented by 4-bit, i.e., $\{0,1,2,3,4\} = \{0,\uparrow,\downarrow,\uparrow\downarrow\}$. For Kondo system, the local sites are classified into itinerant electron part and localized spin part. For simplicity, we use 4-bit state with the restriction of localized spin part only using $\{1,2\} = \{\uparrow,\downarrow\}$. For spin system, the spin angular momentum can be assigned by each sites. Thus, at i-th site with the spin angular momentum S_i, the Hilbert space at i-th site is represented by $2S_i + 1$ bit and the Hilbert space of the system is represented by $|\phi_0, \phi_1, \cdots \phi_{N_s}\rangle$, where $\phi_i = \{-2S_i, -2S_i - 1, \cdots, 2S_i\}$ and N_s is the whole number of sites.

When the canonical ensemble due to the conservation of particle numbers or/and S_z is selected by an input file, $\mathcal{H}\Phi$ automatically construct the limited Hilbert space $\{\boldsymbol{\phi}\}$ which preserves the conservation conditions.

Step 3. Calculation

In $\mathcal{H}\Phi$, there are three calculation modes, exact diagonalization by the Lanczos method (hereafter, Lanczos methods), calculation physical properties by using thermal quantum pure state (hereafter, TPQ method), and full diagonalization. For Lanczos and TPQ methods, the matrix components of Hamiltonian $\mathcal{H}_{ij} \equiv \langle\phi_i|\mathcal{H}|\phi_j\rangle$ is not stored, since the huge Hilbert space is considered to be needed. Thus, matrix multiplication between Hamiltonian and the vector is done by operating the each components of Hamiltonian to the vector. The details of each methods are shown in Sec. 4.

Step 4. Output results

For Lanczos and full diagonalization methods, $\mathcal{H}\Phi$ calculates and outputs the energy $\langle\mathcal{H}\rangle$, the one-body green's functions $\langle c^{\dagger}_{i\sigma_i} c_{j\sigma_j}\rangle$ and two-bodies green's functions $\langle c^{\dagger}_{i\sigma_i} c_{j\sigma_j} c^{\dagger}_{k\sigma_k} c_{l\sigma_l}\rangle$ for obtained eigen vectors. For TPQ method, the inverse temperature, the energy $\langle\mathcal{H}\rangle$ and its variance $\langle\mathcal{H}^2\rangle$ are also obtained per TPQ steps. The one-body green's functions and two-bodies green's functions are outputted at each TPQ steps specified by the input file.

3. Hamiltonian

3.1. Standard mode

In standard mode, standard models can be selected by specifying a keyword `model`. The list of models are shown as follows.

i) `Fermion Hubbard/ Fermion HubbardGC`
The Hamiltonian is given by

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

where μ is the chemical potential, t_{ij} is the transfer integral between i and j sites, U is the on-site Coulomb energy, V_{ij} is the long-range Coulomb energy between i and j sites. $c^\dagger_{i\sigma}$ ($c_{i\sigma}$) is the creation (annihilation) operator of an electron on site i with a spin $\sigma =\uparrow$ or $\downarrow$, $n_{i\sigma} = c^\dagger_{i\sigma}c_{i\sigma}$ is the charge density at site i with a spin σ and $n_i = n_{i\uparrow} + n_{i\downarrow}$.

ii) `Spin/ SpinGC`
The Hamiltonian is given by

$$\begin{aligned} H &= -h\sum_i S_{iz} - \Gamma\sum_i S_{ix} + D\sum_i S_{iz}S_{iz} \\ &+ \sum_{ij,\sigma_1} J_{ij\sigma_1}S_{i\sigma_1}S_{j\sigma_1} + \sum_{ij,\sigma_1\neq\sigma_2} J_{ij\sigma_1\sigma_2}S_{i\sigma_1}S_{j\sigma_2}, \end{aligned} \tag{2}$$

where h is the longitudinal magnetic field, Γ is the transverse magnetic field, and D is the single-site anisotropy parameter. $J_{ij\sigma_1}$ is the spin-coupling constant between (i, σ_1) and (j, σ_1) sites and $J_{ij\sigma_1\sigma_2}$ is the spin-coupling constant between (i, σ_1) and (j, σ_2) sites, where $\{\sigma_1, \sigma_2\} = \{x, y, z\}$. $S_{i\sigma}$ is the spin operator on a site i with a spin σ.

iii) `Kondo/ KondoGC`
The Hamiltonian is given by

$$\begin{aligned} \mathcal{H} &= -\mu\sum_{i\sigma} c^\dagger_{i\sigma}c_{i\sigma} - t\sum_{\langle ij\rangle\sigma} c^\dagger_{i\sigma}c_{j\sigma} + \sum_i U n_{i\uparrow}n_{i\downarrow} + \sum_{i\neq j} V_{ij}n_i n_j \\ &+ \sum_{ij,\sigma_1} J_{ij\sigma_1}S_{i\sigma_1}S_{j\sigma_1} + \sum_{ij,\sigma_1\neq\sigma_2} J_{ij\sigma_1\sigma_2}S_{i\sigma_1}S_{j\sigma_2}, \end{aligned} \tag{3}$$

where μ is the chemical potential, t_{ij} is the transfer integral between i and j sites, U is the on-site Coulomb energy, V_{ij} is the long-range Coulomb energy between i and j sites. $c^\dagger_{i\sigma}$ ($c_{i\sigma}$) is the creation (annihilation) operator of an electron on site i with a spin $\sigma =\uparrow$ or $\downarrow$, $n_{i\sigma} = c^\dagger_{i\sigma}c_{i\sigma}$ is the charge density at site i with a spin σ and $n_i = n_{i\uparrow} + n_{i\downarrow}$. $J_{ij\sigma_1}$ is the spin-coupling constant between (i, σ_1) and (j, σ_1) sites and $J_{ij\sigma_1\sigma_2}$ is the spin-coupling constant between (i, σ_1) and (j, σ_2) sites, where $\{\sigma_1, \sigma_2\} = \{x, y, z\}$. $S_{i\sigma}$ is the spin operator on a site i with a spin σ.

3.2. Expert mode

In expert mode, we can define the generalized Hamiltonian given by

$$\mathcal{H} = \mathcal{H}_0 + \mathcal{H}_{\mathrm{I}}, \tag{4}$$

$$\mathcal{H}_0 = \sum_{i,j}\sum_{\sigma_1,\sigma_2} t_{ij\sigma_1\sigma_2} c_{i\sigma_1}^{\dagger} c_{j\sigma_2}, \tag{5}$$

$$\mathcal{H}_{\mathrm{I}} = \sum_{i,j,k,l}\sum_{\sigma_1,\sigma_2,\sigma_3,\sigma_4} I_{ijkl\sigma_1\sigma_2\sigma_3\sigma_4} c_{i\sigma_1}^{\dagger} c_{j\sigma_2} c_{k\sigma_3}^{\dagger} c_{l\sigma_4}, \tag{6}$$

where $t_{ij\sigma_1\sigma_2}$ is a generalized transfer integral between i site with σ_1 spin and j site with σ_2 spin and $I_{ijkl\sigma_1\sigma_2\sigma_3\sigma_4}$ is the generalized two-body interaction which changes the state from $(j\sigma_2,\ l\sigma_4)$ to $(i\sigma_1,\ k\sigma_3)$. For Hubbard and Kondo models, the 1/2-spin is only allowed. For spin model, the spin angular moment can be modified at each sites.

4. Algorithm

4.1. Lanczos method

4.1.1. Algoritm of Lanczos method

Some parts of this section are based on the manual of titpack [**?**] and textbook by M. Sugihara and K. Murota [**?**] (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 $\mathcal{K}_{n+1}(\hat{\mathcal{H}},\boldsymbol{x}_0) =$

[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.

$\{\boldsymbol{x}_0, \hat{\mathcal{H}}^1\boldsymbol{x}_0, \dots, \hat{\mathcal{H}}^n\boldsymbol{x}_0\}$ 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

$$\boldsymbol{x}_0 = \sum_i a_i \boldsymbol{e}_i. \tag{7}$$

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

$$\hat{\mathcal{H}}^n \boldsymbol{x}_0 = E_0^n \Big[a_0 \boldsymbol{e}_0 + \sum_{i \neq 0} \left(\frac{E_i}{E_0} \right)^n a_i \boldsymbol{e}_i \Big]. \tag{8}$$

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 $\boldsymbol{v}_0, \dots, \boldsymbol{v}_{n-1}$ 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 as follows:

$$\alpha_k = (\hat{\mathcal{H}}\boldsymbol{v}_k, \boldsymbol{v}_k), \tag{9}$$

$$\boldsymbol{w} = \hat{\mathcal{H}}\boldsymbol{v}_k - \beta_k \boldsymbol{v}_{k-1} - \alpha_k \boldsymbol{v}_k, \tag{10}$$

$$\beta_{k+1} = |\boldsymbol{w}|, \tag{11}$$

$$\boldsymbol{v}_{k+1} = \frac{\boldsymbol{w}}{|\boldsymbol{w}|}. \tag{12}$$

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

$$T_n = V_n^\dagger \hat{\mathcal{H}} V_n. \tag{13}$$

Here, V_n is matrix whose column vectors are $\boldsymbol{v}_i (i = 0, 1, \dots, n-1)$. 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. [**?**]).

4.1.2. 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.

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

$$\boldsymbol{y}_k = (\hat{\mathcal{H}} - E_n)\boldsymbol{y}_{k+1}. \tag{14}$$

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.

4.1.3. 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.

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

- For canonical ensemble and `initial_iv` ≥ 0

  A component of a target of Hilbert space is given by

  $$(N_{\text{dim}}/2 + r_s)\% N_{\text{dim}}, \tag{15}$$

  where N_{dim} is a total number of the Hilbert space and $N_{\text{dim}}/2$ 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

  $$123432 + |r_s|, \tag{16}$$

  where r_s is a number given by an input file and n_{run} is a number of runs. A maximum value of n_{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.

4.2. Full Diagonalization method

4.2.1. Over view

We generate matrix of $\hat{H}$ by using the real space configuration $|\psi_j\rangle (j = 1 \cdots d_{\rm H}$, $d_{\rm H}$ 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 $|\Phi_i\rangle$ $(i = 1 \cdots d_{\rm H})$. 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.

4.2.2. 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

$$\langle \hat{A}\rangle = \frac{\sum_{i=1}^{N}\langle A_i\rangle {\rm e}^{-\beta E_i}}{\sum_{i=1}^{N}{\rm e}^{-\beta E_i}}. \tag{17}$$

In the actual calculation are performed as the post scripts.

4.3. 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

$$|\psi_k\rangle \equiv \frac{(l - \hat{H}/N_s)|\psi_{k-1}\rangle}{|(l - \hat{H}/N_s)|\psi_{k-1}\rangle|}. \tag{18}$$

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

$$\beta_k \sim \frac{2k/N_s}{l - u_k}, \quad u_k = \langle\psi_k|\hat{H}|\psi_k\rangle/N_s, \tag{19}$$

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

$$\langle\hat{A}\rangle_{\beta_k} = \langle\psi_k|\hat{A}|\psi_k\rangle/N_s. \tag{20}$$

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 $|\psi_0\rangle$.

4.3.1. 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

$$123432 + (n_{\mathrm{run}} + 1) \times |r_s|, \tag{21}$$

where r_s is a number given by an input file and n_{run} is a number of runs. r_s and a maximum value of n_{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.

5. Parallelization

5.1. Distribution of wavefunction

In the calculation of the exact diagonalization or the thermally pure quantum state, we have to store the many body wavefunction on the random access memory (RAM). This wavefunction sometimes becomes very large. For examples, when we compute the 40-site 1/2 spin system in the S_z unconserved condition, its dimension is $2^{40} = 1,099,511,627,776$ and its size in RAM becomes about 17.6 TB (double precision complex number). For

	Grobal ID			Site					
	for a state	1	2		N_{Local}	$N_{\mathrm{Local}}+1$			N_{Total}
	1	$\downarrow$	$\downarrow$	$\cdots$	$\downarrow$	$\downarrow$	$\downarrow$	$\cdots$	$\downarrow$
	2	$\uparrow$	$\downarrow$	$\cdots$	$\downarrow$	$\downarrow$	$\downarrow$	$\cdots$	$\downarrow$
Proc. 1	3	$\downarrow$	$\uparrow$	$\cdots$	$\downarrow$	$\downarrow$	$\downarrow$	$\cdots$	$\downarrow$
	4	$\uparrow$	$\uparrow$	$\cdots$	$\downarrow$	$\downarrow$	$\downarrow$	$\cdots$	$\downarrow$
	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$		$\vdots$
	$2^{N_{\mathrm{Local}}}$	$\uparrow$	$\uparrow$	$\cdots$	$\uparrow$	$\downarrow$	$\downarrow$	$\cdots$	$\downarrow$
	$2^{N_{\mathrm{Local}}}+1$	$\downarrow$	$\downarrow$	$\cdots$	$\downarrow$	$\uparrow$	$\downarrow$	$\cdots$	$\downarrow$
Proc. 2	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$		$\vdots$
	$2^{N_{\mathrm{Local}}+1}$	$\uparrow$	$\uparrow$	$\cdots$	$\uparrow$	$\uparrow$	$\downarrow$	$\cdots$	$\downarrow$
	$2^{N_{\mathrm{Local}}+1}+1$	$\downarrow$	$\downarrow$	$\cdots$	$\downarrow$	$\downarrow$	$\uparrow$	$\cdots$	$\downarrow$
Proc. 3	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$		$\vdots$
	$2^{N_{\mathrm{Local}}+2}$	$\uparrow$	$\uparrow$	$\cdots$	$\uparrow$	$\downarrow$	$\uparrow$	$\cdots$	$\downarrow$
	$2^{N_{\mathrm{Local}}+2}+1$	$\downarrow$	$\downarrow$	$\cdots$	$\downarrow$	$\uparrow$	$\uparrow$	$\cdots$	$\downarrow$
Proc. 4	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$		$\vdots$
	$2^{N_{\mathrm{Local}}+3}$	$\uparrow$	$\uparrow$	$\cdots$	$\uparrow$	$\uparrow$	$\uparrow$	$\cdots$	$\downarrow$
$\vdots$									
	$2^{N_{\mathrm{Total}}-1}+1$	$\downarrow$	$\downarrow$	$\cdots$	$\downarrow$	$\uparrow$	$\uparrow$	$\cdots$	$\uparrow$
Proc. $2^{N_{\mathrm{Total}}-N_{\mathrm{Local}}}$		$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$		$\vdots$
	$2^{N_{\mathrm{Total}}}$	$\uparrow$	$\uparrow$	$\cdots$	$\uparrow$	$\uparrow$	$\uparrow$	$\cdots$	$\uparrow$

Figure 1: Distribution of the wavefunction of N_{Total}-site 1/2 spin system in the S_z unconserved condition. Each process has wavefunction having $2^{N_{\mathrm{Local}}}$ component; we can obtain the wavefunction of entire system by connecting local wavefunction in all processes.

treating such a large wavefunction, we distribute it in some processes. In $\mathcal{H}\Phi$, the number of processes have to be an exponent of $2S+1$ (for a spin system) or 4 (for an itinerant electron system); By using this processes, we produce a N_{Local}-site subsystem from the whole system which has N_{Total} sites; the remained $N_{\mathrm{Total}} - N_{\mathrm{Local}}$ sites are associated with the process ID.

For example, we show the distribution of the wavefunction of N_{Total}-site 1/2 spin system in the S_z unconserved condition in Fig. 1; the number of processes is $2^{N_{\mathrm{Total}}-N_{\mathrm{Local}}}$. Each process has wavefunction having $2^{N_{\mathrm{Local}}}$ component; we can obtain the wavefunction of entire system by connecting local wavefunction in all processes.

5.2. Parallelism of Hamiltonian-vector product

The most time-consuming procedure in a calculation of the exact diagonalization or the thermally pure quantum state is the production of the Hamiltonian and a vector ($|u\rangle = \hat{\mathcal{H}}|v\rangle$), where $|u\rangle$ and $|v\rangle$ are wavefunctions distributed to each processes. We parallelize this procedure with the aid of the Message Passing Interface(MPI)[**?**]. For reducing the size of arrays

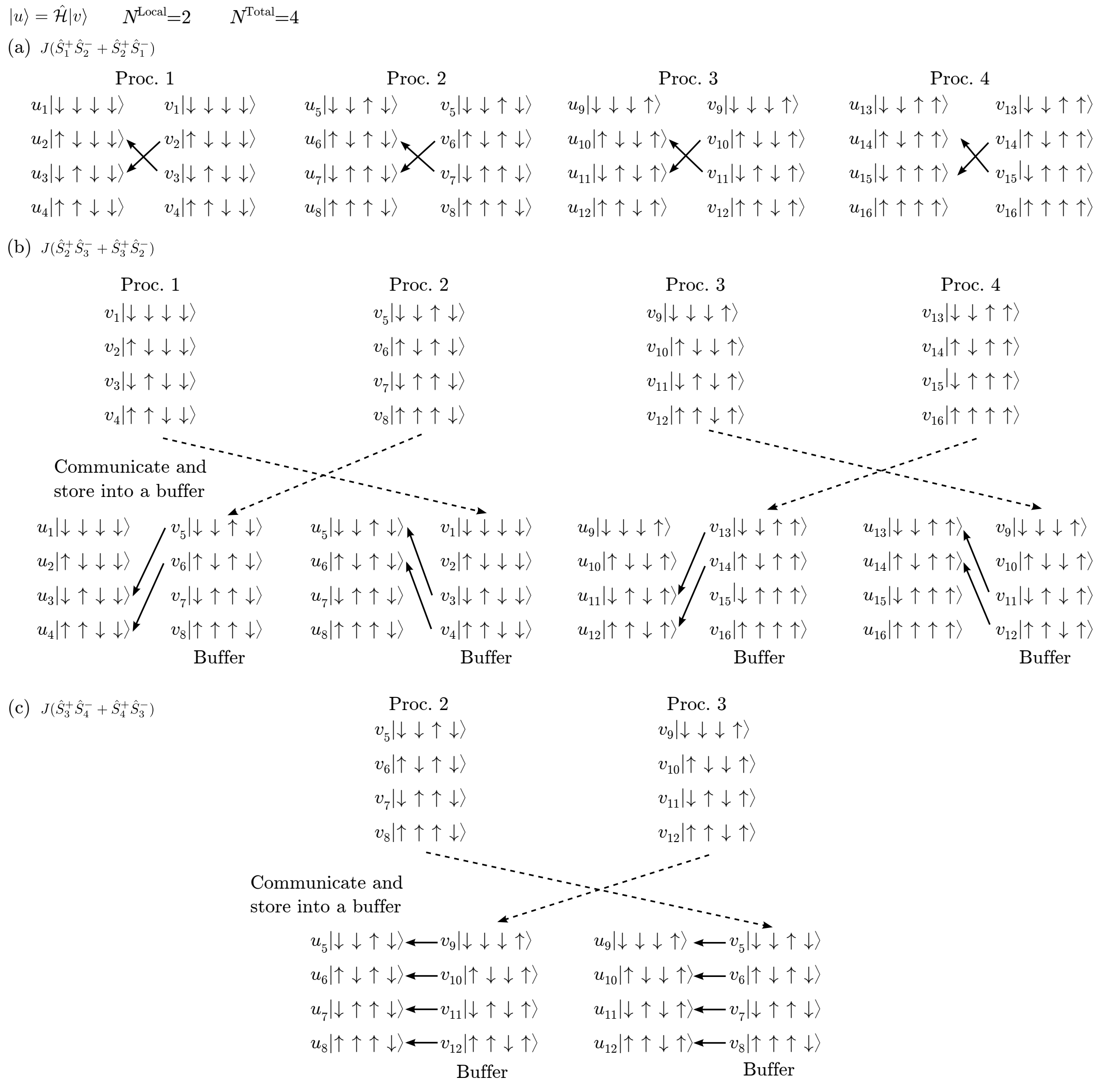
$|u\rangle = \hat{\mathcal{H}}|v\rangle$
$N^{\rm Local}$=2
$N^{\rm Total}$=4
(a) $J(\hat{S}_1^+\hat{S}_2^- + \hat{S}_2^+\hat{S}_1^-)$
Proc. 1
$u_1|\downarrow\downarrow\downarrow\downarrow\rangle$ $v_1|\downarrow\downarrow\downarrow\downarrow\rangle$
$u_2|\uparrow\downarrow\downarrow\downarrow\rangle$ $v_2|\uparrow\downarrow\downarrow\downarrow\rangle$
$u_3|\downarrow\uparrow\downarrow\downarrow\rangle$ $v_3|\downarrow\uparrow\downarrow\downarrow\rangle$
$u_4|\uparrow\uparrow\downarrow\downarrow\rangle$ $v_4|\uparrow\uparrow\downarrow\downarrow\rangle$
Proc. 2
$u_5|\downarrow\downarrow\uparrow\downarrow\rangle$ $v_5|\downarrow\downarrow\uparrow\downarrow\rangle$
$u_6|\uparrow\downarrow\uparrow\downarrow\rangle$ $v_6|\uparrow\downarrow\uparrow\downarrow\rangle$
$u_7|\downarrow\uparrow\uparrow\downarrow\rangle$ $v_7|\downarrow\uparrow\uparrow\downarrow\rangle$
$u_8|\uparrow\uparrow\uparrow\downarrow\rangle$ $v_8|\uparrow\uparrow\uparrow\downarrow\rangle$
Proc. 3
$u_9|\downarrow\downarrow\downarrow\uparrow\rangle$ $v_9|\downarrow\downarrow\downarrow\uparrow\rangle$
$u_{10}|\uparrow\downarrow\downarrow\uparrow\rangle$ $v_{10}|\uparrow\downarrow\downarrow\uparrow\rangle$
$u_{11}|\downarrow\uparrow\downarrow\uparrow\rangle$ $v_{11}|\downarrow\uparrow\downarrow\uparrow\rangle$
$u_{12}|\uparrow\uparrow\downarrow\uparrow\rangle$ $v_{12}|\uparrow\uparrow\downarrow\uparrow\rangle$
Proc. 4
$u_{13}|\downarrow\downarrow\uparrow\uparrow\rangle$ $v_{13}|\downarrow\downarrow\uparrow\uparrow\rangle$
$u_{14}|\uparrow\downarrow\uparrow\uparrow\rangle$ $v_{14}|\uparrow\downarrow\uparrow\uparrow\rangle$
$u_{15}|\downarrow\uparrow\uparrow\uparrow\rangle$ $v_{15}|\downarrow\uparrow\uparrow\uparrow\rangle$
$u_{16}|\uparrow\uparrow\uparrow\uparrow\rangle$ $v_{16}|\uparrow\uparrow\uparrow\uparrow\rangle$
(b) $J(\hat{S}_2^+\hat{S}_3^- + \hat{S}_3^+\hat{S}_2^-)$
Proc. 1
Proc. 2
Proc. 3
Proc. 4
Communicate and store into a buffer
Buffer
Buffer
Buffer
Buffer
(c) $J(\hat{S}_3^+\hat{S}_4^- + \hat{S}_4^+\hat{S}_3^-)$
Proc. 2
Proc. 3
Communicate and store into a buffer
Buffer
Buffer

Figure 2: Hamiltonian-vector product

stored on RAM, $\mathcal{H}\Phi$ does not use the parallelism associated with the index of each operator, such as $J(\hat{S}_i^+\hat{S}_j^- + \hat{S}_j^+\hat{S}_i^-)$ or $t(\hat{c}_{i\sigma}^\dagger\hat{c}_{j\sigma} + \hat{c}_{j\sigma}^\dagger\hat{c}_{i\sigma})$, in a Hamiltonian. We use the following three methods according to the sites associated with an operator(See Fig. 2). When indices of both sites are smaller than or equal to $N_{\rm Local}$, we can perform this operation independently in each processes; there is no inter-process communication [Fig. 2(a)]. If one of these site has an index larger than N_{Local}, first the local wavefunction is communicated between two processes connected with the operator and stored on a buffer. Then we compute the remained operator [Fig. 2(b)]. We employ this procedure also when both sites are larger than N_{Local} [Fig. 2(c)]. In this case, some processes do no participate the calculation; it causes a load invalance. However, the computation time is short because the arrays are accessed sequentially.

5.3. Continuous access

As explained in the previous section devoted to the butterfly algorithm, random memory access has been expected to be inevitable during a Lanczos step. In $\mathcal{H}\Phi$, we implement an algorithm to realize continuous memory access and to enhance computing efficiency ratios, at the cost of additional data transfer. The continuous-memory-access algorithm is available as the Boost mode for $S = 1/2$ spins without total S_z conservation.

Here, we explain the continuous-memory-access algorithm. In $\mathcal{H}\Phi$, the Hilbert space of a L site cluster of $S = 1/2$ spin without total S_z conservation is expanded by eigenstates of $\hat{S}^z_j$, where $\hat{S}^z_j$ is the z component of $S = 1/2$-spin operator of the jth site. The eigenstates of $\hat{S}^z_j$ are implemented as bit arrays by assigning $\uparrow$ ($\downarrow$) spin at the jth site to the bit $I_j = 0$ ($I_j = 1$). Then, the wave function of the L spin cluster is written as

$$|\Phi\rangle = \sum_{I_0=0,1} \sum_{I_1=0,1} \cdots \sum_{I_{L-1}=0,1} C_{I_0 I_1 \cdots I_{L-1}} |I_{L-1} \cdots I_1 I_0\rangle \,. \tag{22}$$

The coefficients $C_{I_0 I_1 \cdots I_{L-1}}$ are stored in a one-dimensional array v as

$$v\left[\sum_{\nu=0}^{L-1} I_\nu \cdot 2^\nu\right] = C_{I_0 I_1 \cdots I_{L-1}}. \tag{23}$$

5.3.1. Continuous access in shared memory architecture

Although the continuous-memory-access algorithm is implemented for both shared memory and distributed memory architectures, we start with the continuous-memory-access algorithm for the shared memory architecture. To realize a continuous-access algorithm during a Lanczos step, we divide the Hamiltonian $\hat{H}$ into R partial Hamiltonians acting on small clusters $\Gamma_C^{(\lambda)} = \{\sigma_0^{(\lambda)}, \sigma_1^{(\lambda)}, \ldots, \sigma_{C-1}^{(\lambda)}\}$ ($\lambda = 0, 1, \ldots, R-1$) that overlap each other:

$$\hat{H} = \sum_{\lambda=0}^{R-1} \hat{H}_{\Gamma_C^{(\lambda)}}. \tag{24}$$

Here, the overlapping clusters cover the entire system as $\{0, 1, \ldots, L-1\} = \Gamma_C^{(0)} \cup \Gamma_C^{(1)} \cup \cdots \cup \Gamma_C^{(R-1)}$. The number of the small clusters R, the size of the clusters C, and the system size L satisfy the inequality $W \equiv L/R < C$ for any connected lattices that cannot be divided into two independent subsystems. If the divided small clusters $\Gamma_C^{(\lambda)} = \{\sigma_0^{(\lambda)}, \sigma_1^{(\lambda)}, \ldots, \sigma_{C-1}^{(\lambda)}\}$ satisfy the following conditions, by storing the part of the coefficients $C_{I_0 I_1 \cdots I_{L-1}}$ in a small matrix and utilizing bit shifts, an efficient continuous-memory-access algorithm is achievable.

Before giving our practical conditions on the cluster division, first, we explain simple observations on how the partial Hamiltonian $\hat{H}_{\Gamma_C^{(\lambda)}}$ acts on bit arrays. If $\Gamma_C^{(0)}$ consists of C successive bits as $\Gamma_C^{(0)} = \{0, 1, \ldots, C-1\}$, then, $\hat{H}_{\Gamma_C^{(0)}}$ is represented by a $2^C \times 2^C$ matrix acting on a 2^C dimensional vector for each given bit array $I_C I_{C+1} \cdots I_{L-1}$. In other word, multiplication of $\hat{H}_{\Gamma_C^{(0)}}$ to the wave function is represented by matrix-vector operations with an outer loop indexed by the bit array $I_C I_{C+1} \cdots I_{L-1}$. Then, if we set $\sigma_0^{(\lambda)} = \lambda W$ ($\lambda = 0, 1, \ldots, R-1$) and shift the W front bits to the end of the array after multiplication of $\hat{H}_{\Gamma_C^{(\lambda)}}$ for each λ, the partial Hamiltonian $\hat{H}_{\Gamma_C^{(\lambda)}}$ always acts on a 2^C dimensional vector successively stored in memory.

The above division of the system is simple but not flexible and efficient. In $\mathcal{H}\Phi$, more practical divisions of the system are implemented. To make the division more flexible, we just impose conditions,

$$\sigma_0^{(\lambda)} = \lambda W \ (\lambda = 0, 1, \ldots, R-1) \tag{25}$$

and

$$\sigma_0^{(\lambda)} \leq \sigma_1^{(\lambda)} \leq \cdots \leq \sigma_{C-1}^{(\lambda)}, \tag{26}$$

and do not require $\sigma_\nu^{(0)} = \nu$ ($\nu = 0, 1, \ldots, C-1$). Instead, we introduce a buffer set of bits $\Gamma_D^{(\lambda)}$

$$\Gamma_D^{(\lambda)} = \{\lambda W, \lambda W + 1, \ldots, \lambda W + C + D - 1\} \setminus \Gamma_C^{(\lambda)}, \tag{27}$$

where D is the order of $\Gamma_D^{(\lambda)}$ that satisfies $C + D \geq \sigma_{C-1}^{(\lambda)} - \sigma_0^{(\lambda)} + 1$. Then, the 2^C dimensional vector introduced in the above naive division is replaced by a 2^C by 2^D matrix.

For $\lambda = 0$ and a given bit array $I_{C+D} \cdots I_{L-1}$, the 2^C by 2^D matrix is defined as

$$\widetilde{X}_{J_{\{I_\mu|\mu\in\Gamma_C^{(0)}\}} J_{\{I_\nu|\nu\in\Gamma_D^{(0)}\}}} = C_{I_0 \cdots I_{C+D-1} \underbrace{I_{C+D} \cdots I_{L-1}}_{\text{given}}}, \tag{28}$$

where $J_{\{I_\mu|\mu\in\Gamma_C^{(0)}\}}$ and $J_{\{I_\nu|\nu\in\Gamma_D^{(0)}\}}$ are decimal representations of sets of bits $\Gamma_C^{(0)}$ and $\Gamma_D^{(0)}$, respectiviely, defined as follows. After odering the elements in a set of bit Γ as

$$\Gamma = \{\mu_0, \mu_1, \cdots, \mu_{G-1}\} \ (\mu_0 \leq \mu_1 \leq \cdots \leq \mu_{G-1}), \tag{29}$$

where G is the order of the set Γ, the decimal representation of the bit set Γ is given by

$$J_{\{I_\mu|\mu\in\Gamma\}} = \sum_{a=0}^{G-1} I_{\mu_a} \cdot 2^a, \tag{30}$$

where $\{I_\mu|\mu \in \Gamma\} = \{I_{\mu_0}, I_{\mu_1}, \ldots, I_{\mu_{G-1}}\}$. The partial Hamiltonian $\widetilde{H}_{J_{\{I_\mu|\mu\in\Gamma_C^{(0)}\}} J_{\{I'_\mu|\mu\in\Gamma_C^{(0)}\}}}$ is correspondingly represent as the following $2^C \times 2^C$ matrix,

$$\widetilde{H}_{J_{\{I_\mu|\mu\in\Gamma_C^{(0)}\}} J_{\{I'_\mu|\mu\in\Gamma_C^{(0)}\}}} = \left\langle I_{\mu_{C-1}} \cdots I_{\mu_1} I_{\mu_0} \right| \hat{H}_{\Gamma_C^{(0)}} \left| I'_{\mu_{C-1}} \cdots I'_{\mu_1} I'_{\mu_0} \right\rangle, \tag{31}$$

where the bit array $I_{L-1} \cdots I_{C+D}$ is omitted in the above equation for simplicity. The multiplication of $\hat{H}_{\Gamma_C^{(0)}}$ is now represented by a matix-matrix operation,

$$\widetilde{X'} = \widetilde{H}\widetilde{X}, \tag{32}$$

where the matrix elements of $\widetilde{X'}$ represent the updated coefficients of the wave function as

$$C'_{I_0 \cdots I_{C-1} I_C \cdots I_{C+D-1} \underbrace{I_{C+D} \cdots I_{L-1}}_{\text{given}}} = \widetilde{X'}_{J_{\{I_\mu|\mu\in\Gamma_C\}} J_{\{I_\nu|\nu\in\Gamma_D\}}}. \tag{33}$$

Consequently, by executing the outer loop indexed by $J_{\{I_\mu|\mu\in\Gamma_\Delta^{(0)}\}}$, where $\Gamma_\Delta^{(\lambda)}$ is defined as

$$\Gamma_\Delta^{(\lambda)} = \{0, 1, \ldots, L-1\} \setminus \{0, 1, \ldots, C+D-1\}, \tag{34}$$

the multiplication of $\hat{H}_{\Gamma_C^{(0)}}$ summarized as

$$|\Phi'\rangle \equiv \hat{H}_{\Gamma_C^{(0)}}|\Phi\rangle = \sum_{I_0=0,1}\sum_{I_1=0,1}\cdots\sum_{I_{L-1}=0,1} C'_{I_0 I_1\cdots I_{L-1}}|I_{L-1}\cdots I_1 I_0\rangle \tag{35}$$

is completed. As achieved in the above simple division of the system, therefore, the present division with the buffers $\Gamma_D^{(\lambda)}$ indeed realizes the continuous memory access during the multiplication of $\hat{H}_{\Gamma_C^{(0)}}$.

It is straightforward to implement the multiplication of $\hat{H}_{\Gamma_C^{(\lambda)}}$ for $\lambda \geq 1$ in a continuous-memory-access fashion by utilizing index shifts in the partial Hamiltonian $\hat{H}_{\Gamma_C^{(\lambda)}}$ and W $(= L/R < C)$ bit shifts in the wave function. First, we shift the indices as

$$\Gamma_C^{(\lambda)} = \{\sigma_0^{(\lambda)}, \sigma_1^{(\lambda)}, \ldots, \sigma_{C-1}^{(\lambda)}\} \xrightarrow{\text{index shift}} \overline{\Gamma}_C^{(\lambda)} = \{0, \sigma_1^{(\lambda)} - \lambda W, \ldots, \sigma_{C-1}^{(\lambda)} - \lambda W\}, \tag{36}$$

and replace $\hat{H}_{\Gamma_C^{(\lambda)}}$ with $\hat{H}_{\overline{\Gamma}_C^{(\lambda)}}$. Next, we shift the bits as

$$\begin{aligned} |\Phi\rangle &= \sum_{I_0=0,1}\sum_{I_1=0,1}\cdots\sum_{I_{L-1}=0,1} C_{I_0 I_1\cdots I_{L-1}}|\overbrace{I_{L-1}\cdots I_W}^{L-W}\overbrace{I_{W-1}\cdots I_0}^{W}\rangle \\ \xrightarrow{\text{bit shift}} \hat{\mathcal{B}}_W|\Phi\rangle &= \sum_{I_0=0,1}\sum_{I_1=0,1}\cdots\sum_{I_{L-1}=0,1} C_{I_0 I_1\cdots I_{L-1}}|\overbrace{I_{W-1}\cdots I_0}^{W}\overbrace{I_{L-1}\cdots I_W}^{L-W}\rangle, \end{aligned} \tag{37}$$

where the operator $\hat{\mathcal{B}}_W$ representing the bit shift is introduced for the following discussion. Then, the multiplication of $\hat{H}_{\Gamma_C^{(\lambda)}}$ to $|\Phi\rangle$ is replaced with the multiplication of $\hat{H}_{\overline{\Gamma}_C^{(\lambda)}}$ to $(\hat{\mathcal{B}}_W)^\lambda|\Phi\rangle$. As explained in the multiplication of $\hat{H}_{\Gamma_C^{(0)}}$ to $|\Phi\rangle$, the multiplication of $\hat{H}_{\overline{\Gamma}_C^{(\lambda)}}$ to $(\hat{\mathcal{B}}_W)^\lambda|\Phi\rangle$ is represented by a matrix-matrix operation with continuous memory access.

Then, the whole process of the continuous-memory-access multiplication is given by

$$\hat{H}|\Phi\rangle = |\widetilde{\Phi}_R\rangle, \tag{38}$$

where $|\widetilde{\Phi}_R\rangle$ is achieved through the recursive steps,

$$|\widetilde{\Phi}_\lambda\rangle = \hat{\mathcal{B}}_W|\widetilde{\Phi}_{\lambda-1}\rangle + \hat{H}_{\overline{\Gamma}_C^{(\lambda)}}(\hat{\mathcal{B}}_W)^\lambda|\Phi\rangle, \tag{39}$$

with the initial condition $|\widetilde{\Phi}_0\rangle = \hat{H}_{\overline{\Gamma}_C^{(0)}}|\Phi\rangle$.

5.3.2. Bit shift in distributed memory architecture

Here, we illustrate our bit-shift procedure in distributed memory architecture.

$$\begin{aligned}
&|\Phi\rangle = \sum_{I_0,I_1,\dots,I_{L-1}} C_{I_0 I_1\cdots I_{L-1}} \\
&\times|\underbrace{I_{L-1}\cdots I_{L-L_\mathrm{p}}}_{L_\mathrm{p}}\underbrace{I_{L-L_\mathrm{p}-1}\cdots I_W}_{L-W-L_\mathrm{p}}\underbrace{\underbrace{I_{W-1}\cdots I_{W-L_\mathrm{p}}}_{L_\mathrm{p}}\underbrace{I_{W-L_\mathrm{p}-1}\cdots I_1 I_0}_{W-L_\mathrm{p}}}_{W}\rangle \\
\xrightarrow{\text{LBS}}\ &\hat{\mathcal{B}}_W^{\mathrm{loc}}|\Phi\rangle = \sum_{I_0,I_1,\dots,I_{L-1}} C_{I_0 I_1\cdots I_{L-1}} \\
&\times|\underbrace{I_{L-1}\cdots I_{L-L_\mathrm{p}}}_{L_\mathrm{p}}\underbrace{\underbrace{I_{W-1}\cdots I_{W-L_\mathrm{p}}}_{L_\mathrm{p}}\overbrace{\underbrace{I_{W-L_\mathrm{p}-1}\cdots I_1 I_0}}^{W-L_\mathrm{p}}}_{W}\underbrace{I_{L-L_\mathrm{p}-1}\cdots I_W}_{L-W-L_\mathrm{p}}\rangle \\
\xrightarrow{\text{A2A}}\ &\hat{\mathcal{P}}_{2^{L_\mathrm{p}}}\hat{\mathcal{B}}_W^{\mathrm{loc}}|\Phi\rangle = \sum_{I_0,I_1,\dots,I_{L-1}} C_{I_0 I_1\cdots I_{L-1}} \\
&\times|\underbrace{\underbrace{I_{W-1}\cdots I_{W-L_\mathrm{p}}}_{L_\mathrm{p}}\underbrace{I_{W-L_\mathrm{p}-1}\cdots I_1 I_0}_{W-L_\mathrm{p}}}_{W}\underbrace{I_{L-1}\cdots I_{L-L_\mathrm{p}}}_{L_\mathrm{p}}\underbrace{I_{L-L_\mathrm{p}-1}\cdots I_W}_{L-W-L_\mathrm{p}}\rangle
\end{aligned} \tag{40}$$

Here, the leftmost L_p bits always define the process to which the bit array belongs.

5.4. Green's functions

6. Performance

In this section, benchmark results of $\mathcal{H}\Phi$ are shown. First, we give a demonstration of finite-temperture simulation implemented in $\mathcal{H}\Phi$. Accuracy of the recently introduced TPQ algorithm [5] used in the finite-temperture simulation can be examined by users themselves for small systems as demostrated below. Second, we show parallelization efficiency of $\mathcal{H}\Phi$. We carried out TPQ simulations of two typical Hamiltonians, namely, a 18-site Hubbard model on a square lattice and a 36-site Heisenberg model on a kagome lattice, with changing numbers of threads and processes. We also show performance of the newly introduced continuous-access parallelization algorithm.

6.1. Benchmark results of TPQ simulations

As an example of finite-temperature TPQ simulations by using $\mathcal{H}\Phi$, we compute the temperature dependence of the doublon density in a Hubbard model ($U/t = 8$) for 8, 10, 12, 14, 16 site clusters with perodic boundary conditions. The Hubbard model is defined by setting $\mu = 0$, $V_{ij} = 0$, $t_{ij} = -t$ for nearest-neighbor pairs of sites, $\langle i, j\rangle$, and $t_{ij} = 0$ for further neighbor pairs of sites. The shapes of the clusters are illustrated in Fig. 3. The users can easily generate the input files that defines these Hamiltonians by using the standard mode of $\mathcal{H}\Phi$ detailed in Sec.3.

Figure 4 shows the temperature dependence of the doublon density calculated from the TPQ state and the canonical ensemble obtained by the full

Table 1: Specification of the single node of the supercomputer system B "sekirei" [**?**] in The Institute for Solid State Physics and the K-computer[**?**] in RIKEN.

	sekirei	K-computer
CPU	Xeon E5-2680v3×2	SPARC 64$^{\mathrm{TM}}$VIIIfx
Numer of cores per node	24	8
Peak performance	960 GFlops	128 GFlops
Main memory	128 GB	16 GB
Memory band width	136.4 GB/s	64 GB/s
Network topology	Enhanced hypercube	Six-dimensional mesh/torus
Network band width	7 GB/s × 2	5 GB/s × 2

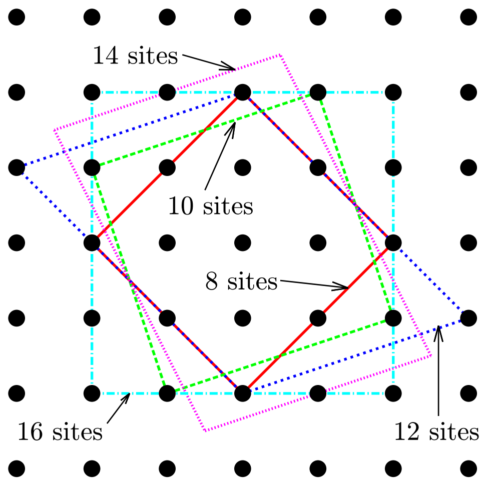

Figure 3: Shape of 8-, 10-, 12-, 14-, 16-site clusters with periodic boundary condition. Tiling of these clusters is simply defined by translation with vectors representing two adjacent sides of the parallelogram clusters.

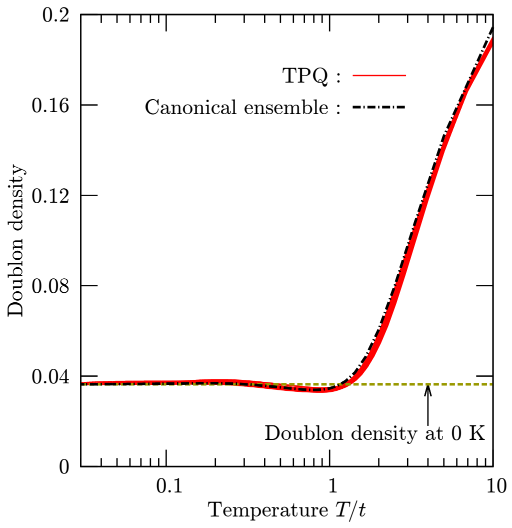

Figure 4: Temperature dependence of the doublon density calculated from the TPQ state and the canonical ensemble obtained by the full diagonalization method on a 8-site Hubbard model. We perform the independent TPQ calculations 20 times and depict every result. The yellow horizontal line indicates the doublon density at zero temperature, which is calculated with the Lanczos method.

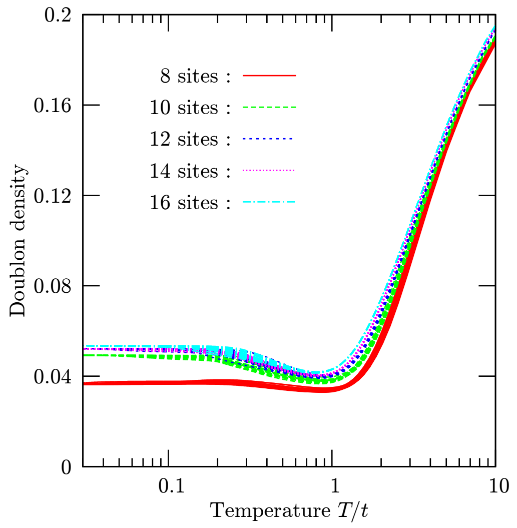

Figure 5: The doublon density calculated from TPQ states on Hubbard model with 8, 10, 12, 14, 16 sites; we perform the TPQ calculation 20 times in each size and plot all of them.

diagonalization method on the Hubbard model with 8 sites. We perform the TPQ calculations with 20 different initial vectors and show all results in Fig. 4. The doublon densities obtained with the TPQ states and the canonical ensemble coincide with each other.

Then, we also show the size dependence of the doublon density calculated by the TPQ states. In Fig. 5, we show the temperature dependence of the doublon density of Hubbard model with 8, 10, 12, 14, 16 sites. We perform the TPQ calculation 20 times for each size and found that the result almost converges about the number of sites.

6.2. Parallelization Efficiency

Here, we carried out TPQ simulations with changing numbers of threads and processes to examine parallelization efficiency of $\mathcal{H}\Phi$. We choose two typical Hamiltonians, namely, a 18-site Hubbard model on a square lattice and a 36-site Heisenberg model on a kagome lattice. The speedup of the simulation for the 18-site Hubbard model with up to 3,072 cores is carried out in the System B of the Supercomputer Center, the Institute for Solid State Physics (ISSP), the University of Tokyo (Table 1). The speedup of the large-scale simulation for the 36-site Heisenberg model with up to 32,768 cores is examined by using the K computer in RIKEN Advanced Institute

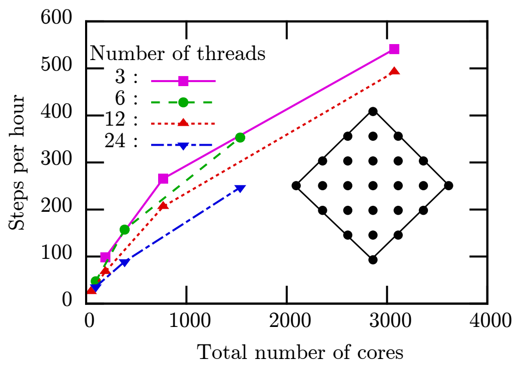

Figure 6: Speedup of TPQ calculations with hybrid parallelization by using up to 3,072 cores. Here, we show TPQ steps per hour for the 18-site Hubbard cluster with finite U/t (U/t=8) and total S_z conservation ($\sum_{i=0}^{N_s-1} S_{iz} = 0$) as functions of the total numner of cores. We note that, due to changes in size of data transfer during MPI process, the thread number affects the TPQ steps per hour even when the total number of cores is fixed. The squares, circles, upward triangles, and downward triangles represent the results with 3 threads, 6 threads, 12 threads, and 24 threads, respectively. The inset shows the shape of the 18-site cluster used in this benchmark calculations.

for Computational Science (Table 1).

6.2.1. *18-site Hubbard model*

Figure 6 shows the speedup of the TPQ simulations for the 18-site Hubbard model on the square lattice illustrated in the inset of fig.6. Here, we employ the subspace of the Hilbert space that satisfies $\sum_{i=0}^{N_s-1} S_{iz} = 0$. Then, the dimension of the subspace is $2,363,904,400$.

Here, we note that, due to changes in size of data transfer during MPI process, the thread number affects the TPQ steps per hour even when the total number of cores is fixed. The strong thread number dependence may be attributed to the moderate throughput of the internode data transfer in the System B of the Supercomputer Center, ISSP. Although the TPQ steps per hour increases as the total number of cores increases, the speedup for 3,072 cores is insufficient possibly due to intricacy of communication.

6.2.2. *36-site Heisenberg model*

Figure 7 shows the speedup of TPQ calculations for the 36-site Heisenberg model on the kagome lattice with hybrid parallelization by using up to 32,768 cores. Here, we employ the 2^{36}-dimensional Hilber space. Due to

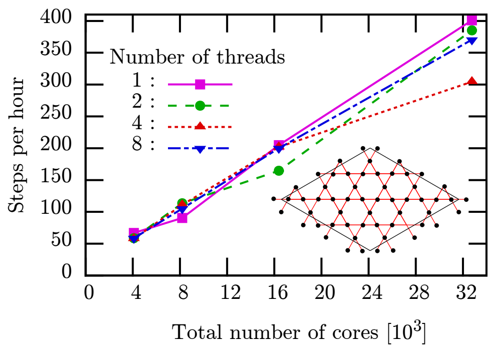

Figure 7: Speedup of TPQ calculations with hybrid parallelization by using up to 32,768 cores. Here, we show TPQ steps per hour for the 36-site Hubbard cluster without total S_z conservation as functions of the total numner of cores. The squares, circles, upward triangles, and downward triangles represent the results with 1 thread, 2 threads, 4 threads, and 8 threads, respectively. The inset shows the shape of the 36-site cluster used in this benchmark calculations.

the high throughput of the internode data transfer of the K computer, the parallelization efficiency from 4096 cores to 32,768 cores reaches 82%.

6.3. Benchmark of boost hybrid

In this section, we show benchmark results of the continuous-access parallelization algorithm called as boost hybrid algorithm in $\mathcal{H}\Phi$, which is introduced in Sec. $*$. The boost hybrid is faster than the conventional parallelization scheme detailed in Sec. $*$ when inter-site interactions are dense and complicated. To compare performance of the boost hybrid and conventional parallelization scheme, here, we carried out TPQ simulations of a spin Hamiltonian with complicated spin-spin interactions.

One of the aims of developing $\mathcal{H}\Phi$ is to make a flexible diagonalization program that enables us to compare the theoretical calculations and experimental data directly. In contrast to simple model Hamiltonians introduced in textbooks of quantum statistical physics, complicated Hamiltonians are inevitably introduced to describe electronic properties of the real compounds quantitatively. Thus, we develop the boost hybrid parallelization scheme suitable for handling complicated Hamiltonians.

Here, we show the benchmark results of the TPQ simulation for a spin Hamiltonians of iridium oxide, Na_2IrO_3, which was derived by utilizing *ab initio* electronic band structures and many-body perturbation theory. The

Hamiltonian is defined on a honeycomb structure and has complicated inter-site spin-spin interactions [6] that can be classified into five parts as follows:

$$\mathcal{H} = \mathcal{H}_{\mathrm{K}} + \mathcal{H}_{\mathrm{J}} + \mathcal{H}_{\mathrm{off}} + \mathcal{H}_2 + \mathcal{H}_3, \tag{41}$$

where the first term $\mathcal{H}_{\mathrm{K}}$ is given by a so-called Kitaev Hamiltonian, and the second term $\mathcal{H}_{\mathrm{J}}$ describes diagonal exchange couplings between nearest-neighbor spins, as follows:

$$\begin{aligned} \mathcal{H}_{\mathrm{K}} &= K' \sum_{\langle i,j\rangle\in\Gamma_{\mathrm{X}}} S_{ix}S_{jx} + K' \sum_{\langle i,j\rangle\in\Gamma_{\mathrm{Y}}} S_{iy}S_{jy} + K \sum_{\langle i,j\rangle\in\Gamma_{\mathrm{Z}}} S_{iz}S_{jz} && (42) \\ \mathcal{H}_{\mathrm{J}} &= \sum_{\langle i,j\rangle\in\Gamma_{\mathrm{X}}} (J''S_{iy}S_{jy} + J'S_{iz}S_{jz}) + \sum_{\langle i,j\rangle\in\Gamma_{\mathrm{Y}}} (J'S_{iz}S_{jz} + J''S_{ix}S_{jx}) \\ &+ \sum_{\langle i,j\rangle\in\Gamma_{\mathrm{Z}}} J(S_{ix}S_{jx} + S_{iy}S_{jy}). && (43) \end{aligned}$$

Here, $\langle i,j\rangle$ denotes a nearest-neighbor (n.n.) pair of sites, and Γ_{X}, Γ_{Y}, and Γ_{Z} are sets of n.n. pairs as illustrated in the upper panel of Fig. 8. The honeycomb structure has three kinds of We classified the n.n. bonds into three sets, Γ_{X}, Γ_{Y}, and Γ_{Z}, depending on their bond directions. The third term in Eq.(41), $\mathcal{H}_{\mathrm{off}}$, describes complicated off-diagonal spin-spin interactions given as,

$$\mathcal{H}_{\mathrm{off}} = \sum_{\langle i,j\rangle\in\Gamma_{\mathrm{X}}} \boldsymbol{S}_i^\dagger \mathcal{J}_{\mathrm{X}}^{\mathrm{off}} \boldsymbol{S}_j + \sum_{\langle i,j\rangle\in\Gamma_{\mathrm{Y}}} \boldsymbol{S}_i^\dagger \mathcal{J}_{\mathrm{Y}}^{\mathrm{off}} \boldsymbol{S}_j + \sum_{\langle i,j\rangle\in\Gamma_{\mathrm{Z}}} \boldsymbol{S}_i^\dagger \mathcal{J}_{\mathrm{Z}}^{\mathrm{off}} \boldsymbol{S}_j, \tag{44}$$

where $\boldsymbol{S}_i^\dagger = (S_{ix}, S_{iy}, S_{iz})$, and the matrices $\mathcal{J}_{\mathrm{X}}^{\mathrm{off}}$, $\mathcal{J}_{\mathrm{Y}}^{\mathrm{off}}$ and $\mathcal{J}_{\mathrm{Z}}^{\mathrm{off}}$ are defined as

$$\begin{aligned} \mathcal{J}_{\mathrm{X}}^{\mathrm{off}} &= \begin{bmatrix} 0 & I_2'' & I_2' \\ I_2'' & 0 & I_1' \\ I_2' & I_1' & 0 \end{bmatrix}, && (45) \\ \mathcal{J}_{\mathrm{Y}}^{\mathrm{off}} &= \begin{bmatrix} 0 & I_2'' & I_1' \\ I_2'' & 0 & I_2' \\ I_1' & I_2' & 0 \end{bmatrix}, && (46) \\ \mathcal{J}_{\mathrm{Z}}^{\mathrm{off}} &= \begin{bmatrix} 0 & I_1 & I_2 \\ I_1 & 0 & I_2 \\ I_2 & I_2 & 0 \end{bmatrix}. && (47) \end{aligned}$$

The fourth term $\mathcal{H}_2$ describes the second neighbor (2nd n.) interactions defined as,

$$\mathcal{H}_2 = \sum_{\langle\langle i,j\rangle\rangle \in \Gamma_{\rm Z}^{\rm 2nd}} \boldsymbol{S}_i^{\dagger} \mathcal{J}_{\rm Z}^{\rm 2nd} \boldsymbol{S}_j, \tag{48}$$

where $\Gamma_{\rm Z}^{\rm 2nd}$ is the set of the 2nd n. bonds orthogonal to the bonds in $\Gamma_{\rm Z}$ and the matrix $\mathcal{J}_{\rm Z}^{\rm 2nd}$ is given by

$$\mathcal{J}_{\rm Z}^{\rm 2nd} = \left[\begin{array}{ccc} J^{\rm 2nd} & I_1^{\rm 2nd} & I_2^{\rm 2nd} \\ I_1^{\rm 2nd} & J^{\rm 2nd} & I_2^{\rm 2nd} \\ I_2^{\rm 2nd} & I_2^{\rm 2nd} & K^{\rm 2nd} \end{array} \right]. \tag{49}$$

The third neighbor interactions are described by the fifth term $\mathcal{H}_3$,

$$\mathcal{H}_3 = J^{\rm 3rd} \sum_{\langle\langle\langle i,j\rangle\rangle\rangle \in \Gamma^{\rm 3rd}} (S_{ix}S_{jx} + S_{iy}S_{jy} + S_{iz}S_{jz}). \tag{50}$$

The fifteen coupling constants, K, K', J, J', J'', $\ldots$, and $J_{\rm 3rd}$ are given in the previous study [6].

In the lower panel of Fig. 8, elapsed time per TPQ step for the conventional and boost hybrid parallelization scheme is shown for a Kitaev Hamiltonian, $\mathcal{H}_{\rm K}$, a Kitaev-Heisenberg Hamiltonian, $\mathcal{H}_{\rm K}+\mathcal{H}_{\rm J}$, a n.n. Hamiltonian, $\mathcal{H}_{\rm nn} = \mathcal{H}_{\rm K}+\mathcal{H}_{\rm J}+\mathcal{H}_{\rm off}$, a Hamiltonian including 2nd n. interactions, $\mathcal{H}_{\rm nn}+\mathcal{H}_2$, and the *ab initio* Hamiltonian $\mathcal{H} = \mathcal{H}_{\rm nn}+\mathcal{H}_2+\mathcal{H}_3$. Here, we use 16 nodes (24 cores per node) in the Supercomputer Center of ISSP and perform the TPQ simulation with 3 threads and 128 processes. As the number of the coupling constant increases, from the sparse Kitaev Hamiltonian $\mathcal{H}_{\rm K}$ to the dense and complicated Hamiltonian of $\rm Na_2IrO_3$, $\mathcal{H}$, the elpased time per TPQ step of the boost hybrid scheme does not show significant increase while the elapsed time of the conventional scheme shows significant increase.

Although, for the sparse Kitaev Hamiltonian $\mathcal{H}_{\rm K}$, the conventional scheme shows slightly better performance than that of the boost hybrid scheme, the boost hybrid scheme is three times faster than the conventional one even for the n.n. Hamiltonian $\mathcal{H}_{\rm nn}$. Furthermore, even though, from $\mathcal{H}_{\rm nn}$ to $\mathcal{H}_{\rm nn}+\mathcal{H}_2$, the number of the bonds clearly increases, the elapsed time per TPQ step of the boost hybrid scheme only shows slight increase.

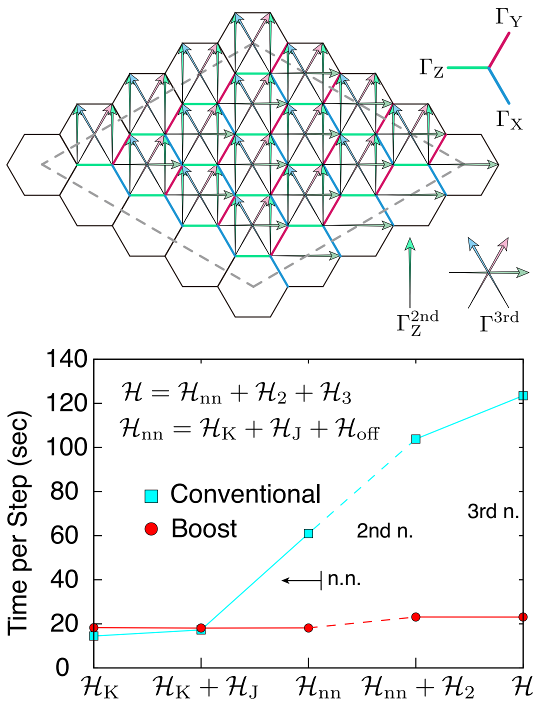

Figure 8: Lattice structure and bonds for $\mathcal{H}$ (upper panel) and elapsed time per TPQ step for conventional and boost hybrid paralleliztion schemes (lower panel). The three different nearest-neighbor (n.n.) bonds, Γ_X, Γ_Y, and Γ_Z, the second neighbor (2nd n.) bonds Γ_Z^{2nd}, and the third neighbor (3rd n.) bonds Γ^{3rd} are illustrated in the upper panel. In the lower panel, the (cyan) squares denote the elapsed time per TPQ step of the conventional scheme and the (red) circles denote the elapsed time per TPQ step of the boost hybrid scheme. Here, we use 16 nodes (24 cores per node) in the Supercomputer Center of ISSP and perform the TPQ simulation with 3 threads and 128 processes. As the number of the coupling constant increases, from a sparse Kitaev Hamiltonian $\mathcal{H}_K$ to the dense and complicated Hamiltonian of Na_2IrO_3, $\mathcal{H}$, the elpased time per TPQ step of the boost hybrid scheme does not show significant increase while the elapsed time of the conventional scheme shows significant increase.

7. Getting started

7.1. Download and build

We can obtain the tarball at the $\mathcal{H}\Phi$ download site; manuals are also found there. For building the executable of $\mathcal{H}\Phi$, we should prepare the C compiler and BLAS/LAPACK library[**?**]. If we perform the parallel computation, the MPI library is also required.

We can use `cmake` for building $\mathcal{H}\Phi$ as follows

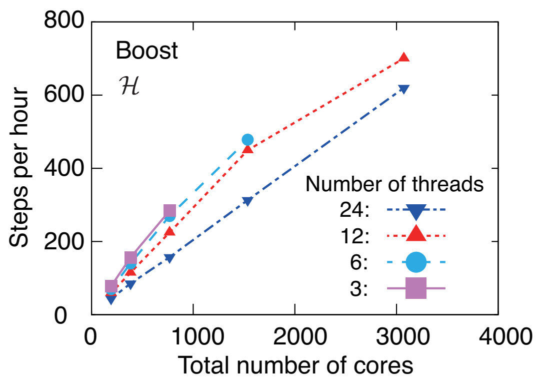

Figure 9: Speedup of TPQ calculations with hybrid parallelization by using up to 3,072 cores. Here, we show TPQ steps per hour for the 32-site spin Hamiltonian of Na_2IrO_3 without total S_z conservation as functions of the total numner of cores. The squares, circles, upward triangles, and downward triangles represent the results with 1 thread, 2 threads, 4 threads, and 8 threads, respectively.

```
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.

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.

If the system does not have `cmake`, there is an alternative way for building $\mathcal{H}\Phi$ as follows: `HPhiconfig.sh` script in the $\mathcal{H}\Phi$ top directory can be run as

```
$ bash HPhiconfig.sh gcc
$ make HPhi
```

Then we can obtain an executable `HPhi` in `src/` directory.

7.2. Running samples

There are some sample input files and reference results in `samples/` directory. For examples, `samples/Standard/Hubbard/square/` is the directory containing files for the computation of the Hubbard model on the 2×4-site square lattice. We can perform the calculation of the ground state by running $\mathcal{H}\Phi$ as the "Standard mode",

```
$ ../../../../src/HPhi -s StdFace.def
```

Then $\mathcal{H}\Phi$ starts a calculation with some information on the standard output. We can check the geometry of the calculated system by using `lattice.gp` which is generated by $\mathcal{H}\Phi$; it is read by `gnuplot`[**?**] as follows:

```
$ gnuplot lattice.gp
```

The calculated results are written to files in `output/`; this directory is also generated by $\mathcal{H}\Phi$. `output/zvo_energy.dat` contains the total energy, the number of doublon, and the magnetization along the z axis; we can compare it with the reference data in `output_Lanczos/zvo_energy.dat` . Correlation functions are obtained in `output/zvo_cisajs.dat` and `output/zvo_cisajscktalt.dat`; more detailed analysis can be performed with these files. We can create an original model by editing generated new input files (`calcmod.def`, `greenone.def`, `greentwo.def`, `modpara.def`, `zInterAll.def`, `zTrans.def`, and `zlocspn.def`) and run $\mathcal{H}\Phi$ as

```
$ ../../../../src/HPhi -e namelist.def
```

This "Expert mode" can perform more flexible calculation.

[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`