Kω – Open-source library for the shifted Krylov subspace method

Takeo Hoshi\textsuperscript{a,}\textsuperscript{*}, Mitsuaki Kawamura\textsuperscript{b}, Kazuyoshi Yoshimi\textsuperscript{b}, Yuichi Motoyama\textsuperscript{b}, Takahiro Misawa\textsuperscript{b}, Youhei Yamaji\textsuperscript{c}, Synge Todo\textsuperscript{a,}\textsuperscript{b}, Naoki Kawashima\textsuperscript{b}, Tomohiro Sogabe\textsuperscript{c}

\textsuperscript{a}Department of Applied Mathematics and Physics, Tottori University, Tottori-shi, Tottori 680-8552, Japan
\textsuperscript{b}Institute for Solid State Physics, University of Tokyo, Kashiwa-shi, Chiba 277-8581 Japan
\textsuperscript{c}Department of Applied Physics, University of Tokyo, Bunkyo-ku, Tokyo 113-8656, Japan

\textsuperscript{*}Department of Applied Physics, Nagoya University, Chikusa-ku, Nagoya 464-8601, Japan

Abstract

We develop Kω, an open-source linear algebra library for the shifted Krylov subspace methods. The methods solve a set of shifted linear equations for a given matrix and a vector, simultaneously. The leading order of the operational cost is the same as that for a single equation. The shift invariance of the Krylov subspace is the mathematical foundation of the shifted Krylov subspace methods. Applications in materials science are presented to demonstrate the advantages of the algorithm over the standard Krylov subspace methods such as the Lanczos method. We introduce benchmark calculations of (i) an excited (optical) spectrum and (ii) intermediate eigenvalues by the contour integral on the complex plane. In combination with the quantum lattice solver $\Phi\Phi$, Kω can realize parallel computation of excitation spectra and intermediate eigenvalues for various quantum lattice models.

Keywords: Numerical linear algebra, Shifted linear equations, Krylov subspace method, Quantum lattice models

1. Introduction

The response of physical observables to an external field or a perturbation is an essential probe in the experimental and theoretical studies of quantum many-body systems. Theoretically, such a quantity can be formulated as a selected element of the Green’s function,

$$a^\dagger G(z) b,$$  \hspace{1cm} (1)

with given states (or vectors) $a$ and $b$, where $z$ is the complex energy parameter and the Green’s function $G(z)$ is defined as the inverse of the shifted Hamiltonian $H$,

$$G(z) \equiv (zI - H)^{-1}.$$ \hspace{1cm} (2)

In computational quantum physics, the Hamiltonian $H$ is usually represented by a real-symmetric or Hermitian $M \times M$ sparse matrix. A typical application of the Green’s function formalism is calculating spectra in which the elements $a^\dagger G(z)b$ are calculated at sampling points $z_k$ located near the real axis $(z_k \equiv \omega_k + i\delta)$ with a tiny imaginary part $\delta$ or where the elements $a^\dagger G(z)b$ ($a \neq b$) are calculated at sampling points $z_k$ for the numerical contour integral on the complex plane. The dimension $M$ is large, e.g., $M \geq 10^5$, and efficient numerical methods are essential to treat such large matrices.

Formally, the Green’s function (2) can be decomposed as

$$G(z) = \sum_j \frac{y_j^\dagger y_j}{z - \lambda_j}.$$ \hspace{1cm} (3)

---

\textsuperscript{*}Corresponding author.
E-mail address: hoshi@tottori-u.ac.jp

Preprint submitted to Computer Physics Communications

January 24, 2020
where $\lambda_j$ and $y_j$ are the $j$-th eigenvalue and the corresponding eigenvector of the matrix $H$, respectively, i.e.,

$$Hy_j = \lambda_jy_j. \quad (4)$$

In practice, however, the numerical evaluation of all the eigen-pairs is too expensive [the operational cost is $O(M^2)$] and the memory cost is $O(M^2)$] to treat large-scale matrices ($M \geq 10^9$).

In computational quantum physics, a Lanczos-based algorithm combined with the continued fraction technique has been proposed and widely used to calculate the Green’s function [1][2]. The most expensive operation of the Lanczos-based algorithm is the matrix–vector product, with an operational cost of $O(M^2)$. Moreover, the sparseness of the Hamiltonian matrix reduces this cost to $O(m)$ where $m$ is the number of nonzero elements in the matrix. However, the calculation of the Green’s function is restricted to diagonal components and it is not straightforward to evaluate the convergences of the obtained results.

In the present paper, we describe the numerical library Kω (https://github.com/issp-center-dev/Komega), which solves linear equations defined by

$$z_kI - H)x^{(k)} = b \quad (k = 0, 1, 2, \ldots, N_{eq} - 1). \quad (5)$$

instead of Eq. (4). Here, $b$ is a given vector, $\{z_k\}$ are given scalars, $I$ is the identity matrix, and $N_{eq}$ is the number of linear equations. Eq. (5) is called the shifted linear equation since the matrix on the left hand side is different only by the ‘shift’ term, $z_kI$. In general, the scalar $z$ is a complex number and the matrix of $(zI - H)$ is non-Hermitian. The solution vector $x^{(k)}$ is written as

$$x^{(k)} = G(z_k)b, \quad (6)$$

and satisfies $a^*G(z_k)b = a^*x^{(k)}$. An iterative algorithm is adopted in Kω, based on the property that the result of the $n$-th iteration, regardless of $z_k$, can be spanned in the Krylov subspace. The operational cost is usually dominated by the matrix–vector multiplication procedure for a vector $v$ ($v \Rightarrow Hv$).

The library Kω is based on efficient algorithms, called shifted Krylov subspace algorithms, designed to efficiently solve the shifted linear equations Eq. (5) with $N_{eq} > 1$. The shifted Krylov algorithms and related techniques are briefly explained in this paper and the details can be found in [3][7][17] and references therein. Currently, shifted Krylov algorithms are used in many computational science fields such as quantum chromodynamics [3], electronic structure calculations [4][8][15][19], excited electron calculations [20], nuclear physics [21][22], transport calculations with non-equilibrium Green’s function theory [23], and nano-structured superconducting systems [24]. However, no open-source numerical library/solver of the shifted Krylov subspace methods has yet been developed to our knowledge.

In this paper, we first review the algorithm implemented in Kω briefly in section 2. Next, the basic information such as installation and usage is introduced in section 3. Then, some examples using Kω as a library or software are illustrated in section 4. Finally, we summarize the paper in section 5.

2. Algorithms

2.1. Overview

In this section, we briefly review the shifted Krylov subspace methods implemented in Kω. The shifted linear equations of Eq. (5) can be rewritten as

$$(A + \sigma I)x^{eq} = b \quad (7)$$

with $A = z_0I - H$ and $\sigma \equiv z_k - z_0$ for $k = 0, 1, 2, \ldots, N_{eq} - 1$. Here, the suffix $k$ in Eq. (5) is dropped for simplicity. Hereafter, Eq. (7) with $\sigma = 0$ is called the seed equation, while the other equations with $\sigma \neq 0$ are called the shifted equations. The accuracy of the approximate solution vector $x^{eq}_n$ can be checked by monitoring the residual vector

$$r^{eq}_n \equiv b - (A + \sigma I)x^{eq}_n, \quad (8)$$

where $n$ is the number of iteration steps.

Table 1 shows the four solver methods available in Kω. The methods are classified according to the type of matrix $A + \sigma I$. Users need to select an appropriate method depending on whether the matrix $H$ is real-symmetric or Hermitian and whether the shift constant values $\{z_k\}$ are complex or real [25].

The mathematical foundation of Kω is an iterative algorithm and the numerical solution of Eq. (7) at the $n$-th iteration is obtained in the Krylov subspace defined as

$$K_n(A, b) = \text{span}[b, Ab, A^2b, \ldots, A^{n-2}b, A^{n-1}b]. \quad (9)$$

The common mathematical foundation of the shifted Krylov subspace methods is the shift invariance property of the Krylov subspace

$$K_n(A + \sigma I, b) = K_n(A, b), \quad (10)$$

and the collinear residual theorem [3], which is explained later in this section.

This section explains the shifted conjugate gradient (CG) method in Table 1 as an example. The other methods, the shifted conjugate orthogonal CG method and the shifted BiCG method are explained in Appendix A and Appendix B respectively.

2.2. Shifted conjugate gradient method

The shifted CG method is based on the CG method [26] that is used for solving $Ax = b$, when the matrix $A$ is real-symmetric (or Hermitian) and positive definite.

2.2.1. Seed equation

The seed equation [Eq. (7) with $\sigma = 0$] is denoted as $Ax = b$ and the solution vector at the $n$-th iterative step is denoted as $x_n$, where $x_0$ is set to be $x_0 = 0$. The residual vector at the $n$-th step is denoted as $r_n = b - Ax_n$. The three vectors, $x_n$, $r_n$, and
where $\mathbf{p}_n$, are computed by the following recurrence equations:

$$ \rho_n = r_n^* r_n $$
$$ \alpha_n = \frac{\rho_n}{\rho_{n-1}} $$
$$ \beta_n = \frac{\rho_{n+1}}{\rho_n} $$
$$ \mathbf{x}_{n+1} = \mathbf{x}_n + \alpha_n \mathbf{p}_n $$
$$ \mathbf{r}_{n+1} = \mathbf{r}_n - \alpha_n \mathbf{A} \mathbf{p}_n $$
$$ \mathbf{p}_{n+1} = \mathbf{r}_{n+1} + \beta_n \mathbf{p}_n $$

with the initialization $\mathbf{p}_0 = \mathbf{r}_0 = \mathbf{b}$.

Kow uses the three-term recurrence formula [7]

$$ \mathbf{r}_{n+1} = \left(1 + \frac{\alpha_b \beta_{n-1}}{\alpha_{n-1}} - \alpha_n A \right) \mathbf{r}_n - \frac{\alpha_b \beta_{n-1}}{\alpha_{n-1}} \mathbf{r}_{n-1}, $$

which is obtained by eliminating $\beta_{n-1}/\alpha_{n-1} = 0$ because $\alpha_0 = \rho_0/(\rho_0^* A \rho_0) = \rho_0/(\mathbf{r}_0^* A \mathbf{r}_0)$. The reason for this implementation style will be presented in the following subsections.

It is crucial that the $n$-th residual $\mathbf{r}_n$ belongs to the $(n+1)$-th Krylov subspace $K_{n+1}(A, \mathbf{b})$ and is orthogonal to the $n$-th Krylov subspace $K_n(A, \mathbf{b})$.

$$ \mathbf{r}_n \in K_{n+1}(A, \mathbf{b}) \quad \text{and} \quad \mathbf{r}_n \perp K_n(A, \mathbf{b}). $$

These mean that $\mathbf{r}_n$ belongs to the one-dimensional orthogonal complementary space of $K_n(A, \mathbf{b})$ within $K_{n+1}(A, \mathbf{b})$, say $K_n^\perp(A, \mathbf{b})$.

### 2.2.2. Shifted equation and collinear residuals

When the CG method is applied to a shifted equation $(A + \sigma I) \mathbf{x} = \mathbf{b}, \sigma \neq 0$ with $\mathbf{x}_0^\sigma = \mathbf{0}$ and $\mathbf{r}_0 = \mathbf{b}$, the residual vector $\mathbf{r}_n^\sigma$ belongs to the same one-dimensional subspace $K_n^\perp(A, \mathbf{b})$ because of Eq. (10). Consequently, the residual $\mathbf{r}_n^\sigma$ is parallel to that of the seed equation ($\mathbf{r}_n$)

$$ \mathbf{r}_n^\sigma = \frac{1}{\pi_n^\sigma} \mathbf{r}_n, $$

where $\pi_n^\sigma$ is a constant [3] called the collinearity factor. Figure 1 is a schematic figure depicting the collinearity ($\mathbf{r}_n^\sigma \parallel \mathbf{r}_n$).

#### Table 1: Classification of the Krylov subspace methods.

| Type of $A + \sigma I$ | Method | Solver name |
|------------------------|--------|-------------|
| Real symmetric         | Shifted conjugate gradient (CG) method (using real vector) | komega_cg_r |
| Hermitian              | Shifted conjugate gradient (CG) method (using complex vector) | komega_cg_c |
| Complex symmetric      | Shifted conjugate orthogonal conjugate gradient (COCG) method | komega_cocg |
| Otherwise              | Shifted bi-conjugate gradient (BiCG) method | komega_bicg |

#### Figure 1: Collinearity of residual vectors. The shaded bottom surface and the cube represent the $n$-th Krylov subspace $K_n(A, \mathbf{b})$ and the $(n+1)$-th Krylov subspace $K_{n+1}(A, \mathbf{b})$, respectively. The two arrows depict the $n$-th residuals, $\mathbf{r}_n$ and $\mathbf{r}_n^\sigma$.

The collinearity factor $\pi_n^\sigma$ satisfies the recurrence equation

$$ \pi_{n+1}^\sigma = \left(1 + \frac{\alpha_b \beta_{n-1}}{\alpha_{n-1}} + \alpha_n \sigma \right) \pi_n^\sigma - \frac{\alpha_b \beta_{n-1}}{\alpha_{n-1}} \pi_{n-1}^\sigma $$

with the initialization $\pi_0^\sigma = \pi_{-1}^\sigma = 1$, because of Eq. (17) and Eq. (20). By substituting $\pi_n^\sigma \mathbf{r}_n^\sigma$ into $\mathbf{r}_n$ in Eq. (17), the recurrence equations for the shifted equations are derived as

$$ \alpha_n^\sigma = \frac{\pi_n^\sigma}{\pi_{n+1}^\sigma} \alpha_n, $$
$$ \beta_n^\sigma = \frac{\pi_n^\sigma}{\pi_{n+1}^\sigma} \beta_n, $$
$$ \mathbf{x}_{n+1}^\sigma = \mathbf{x}_n^\sigma + \alpha_n \mathbf{p}_n^\sigma, $$
$$ \mathbf{p}_{n+1}^\sigma = \frac{1}{\pi_{n+1}^\sigma} \mathbf{r}_{n+1} + \beta_n^\sigma \mathbf{p}_n^\sigma $$

with $\mathbf{x}_0^\sigma = \mathbf{0}$ and $\mathbf{p}_0^\sigma = \mathbf{b}$. It should be noted that the recurrences of Eqs. (21), (22), (23), (24), and (25) require no expensive matrix–vector multiplication.

#### 2.2.3. Cost and projection

Table 2 compares the operational costs of the conventional Krylov method and the shifted Krylov methods [4]. The first row of Table 2 represents the conventional Krylov method, which solves the $N_{eq}$ linear equations independently in each Krylov subspace. Here $M$ and $M_{NZ}$ are the dimension and the average number of non-zero elements per column of matrix $A$, respectively ($M_{NZ} \leq M$). The second row represents the cost of the shifted Krylov method for the case where all the elements of the solution vectors $\mathbf{x}^\sigma$ are calculated. We find that the operational cost for the sparse matrix–vector product (SpMV) is
drastically reduced \((MM_{NZ}N_{eq} \rightarrow MM_{NZ})\), since the explicit SpMV appears only in the seed equation \((\sigma = 0)\). The third row represents the cost of the shifted Krylov subspace method for the case where we do not need all the elements of the solution \(x'\), but only its projection, \(y' = Px'\) with the \(M_{left} \times M\) projection matrix \(P\) \((1 \leq M_{left} \leq M)\). In the third case, we can replace the recurrence equations, Eqs. \((24)\) and \((25)\), by

\[
y'_{n+1} = y'_n + \alpha'_n u'_n
\]

\[
u'_{n+1} = \frac{1}{\pi'_n} Pr'_n + \beta'_n u'_n
\]

with \(y'_0 = 0\) and \(u'_0 = Pb\). By this replacement, the number of scalar–vector products (SV) is reduced from \(3MN_{eq}\) to \(3M + 3M_{left}(N_{eq} - 1)\). For example, the calculation of an element of the Green’s function by Eq. \((4)\)

\[
G_{ab}(\omega) \equiv a^\dagger (I - H)^{-1} b = a^\dagger x(\omega)
\]

is a case with \(M_{left} = 1\). In typical applications, both \(N_{eq}\) and \(M_{left}\) are much smaller than \(M (N_{eq}, M_{left} \ll M)\), and thus the operational costs in the third row in Table 2 are reduced to those in the first row with \(N_{eq} = 1\). In other words, the operational cost in the third case is reduced, typically, to that for solving a single linear equation.

\(K_\omega\) implements the third case in Table 2 i.e., the shifted Krylov subspace method with projection. For the seed equation, the residual vector \(r_n\) is updated by Eq. \((17)\) and the coefficients \(\rho_n, \alpha_n, \) and \(\beta_n\) are updated by Eqs. \((11), (15),\) and \((15)\), respectively. For the shifted equations, the projected solution vector \(y'_{n}\) and the projected search direction vector \(u'_{n}\) are updated by Eqs. \((26)\) and \((27)\), respectively, and the coefficients \(\pi'_n, \alpha'_n, \) and \(\beta'_n\) are updated by Eqs. \((21), (22),\) and \((23)\), respectively. The solution vectors \(x'_{n}\) for the shifted equations can be obtained by setting \(P = I\) or \(M_{left} = M\), but users should accept the additional memory cost, as explained below.

The present implementation of \(K_\omega\) offers a great advantage not only in the operational cost but also in the memory cost, because an \(M\)-dimensional vector \(v\) requires a large memory cost in typical applications. In the present implementation, only three \(M\)-dimensional vectors, the residual vectors for the seed equation \((r_{n+1}, r_n, \) and \(r_{n-1})\), are stored in the memory with a memory cost of \(O(M^1 M_{eq})\). The residual vectors for the shifted equations, such as \(r'_{n}\), can be obtained by Eq. \((20)\) with a negligible memory cost of \(O(M^p M'_{eq})\). To store the other \(M\)-dimensional vectors for the shifted equations, such as \(x'_{n}\) and \(p'_{n}\), the required memory size will be \(O(M^1 M'_{eq})\), which can be huge.

2.2.4. Seed switching

A mathematical technique called seed switching \((7)\) is adopted in \(K_\omega\) for an efficient convergence, because the convergence speed of the CG method can be different among the energy points \(\{\sigma_k\}\). The residual vectors for several shifted equations \((r'_{n}, \sigma \neq 0)\) can sometimes remain large, while that for the seed equation is reduced to be negligible \((4)\). In this case, we can switch the seed equation as \(A' = A + \sigma \text{seed} I\), where \(\sigma \text{seed}\) is the shift that gives the largest residual \(|r'_{n}|\), or in other words, the smallest collinearity factor \(|\rho'_{n=0}|\). The residual vector for the new seed equation is obtained by Eq. \((20)\) to be \(r_{n=0} = (1/\pi'_{n=0})r_n\). It is noted that the present implementation style does not require the solution and search direction vectors for the new seed equation \((x'_{n=0}, p'_{n=0})\).

2.3. Comparison with Lanczos method

Here, we briefly describe the merits for calculating \(G_{ab}(\omega)\) of the shifted Krylov method compared to the traditional way based on the Lanczos-based algorithm \((1, 2)\). In the Lanczos-based algorithm, the Krylov subspace of \(K_{\omega}(A, b)\) is generated by the Lanczos-type recurrence formula and the diagonal component \(G_{bb}\) is given by a continued fraction form. In contrast, using the shifted Krylov method, both the diagonal components \(G_{bb}\) and the off-diagonal components \(G_{ab}\) can be calculated directly and simultaneously with the same order of operational cost as that of the Lanczos-based method. In addition, the accuracy of the obtained results can be evaluated by monitoring the residual vector of Eq. \((8)\).

3. Usage of \(K_\omega\)

In this section, the usage of \(K_\omega\) is introduced. We first introduce how to install \(K_\omega\). \(K_\omega\) provides libraries and a standalone program \(ShiftK\). out. In Sec. 3.2, the procedures for using \(K_\omega\) are schematically shown.

3.1. Installation

The stable version of \(K_\omega\) is distributed on the release page with the Lesser General Public License (LGPL) version 3. To build \(K_\omega\) a Fortran compiler and a BLAS library are required. The following is an example of compiling \(K_\omega\):

\[
\text{
$ ./configure --prefix=install_dir$
$ make$
$ make install$
}
\]

Here, \(install\_dir\) indicates the full path of the directory where the library is installed.
3.2. Schematic flow of $K\omega$ usage

In this subsection, the usage of $K\omega$ as a library or a standalone program is explained. In Fig. 2, the schematic flow of the library usage (a) and the corresponding flowchart in the standalone program (b) are shown.

3.2.1. Library

(i) Preparation of a routine for matrix–vector multiplication

$K\omega$ provides a reverse communication interface for the matrix–vector multiplication routine ($v \rightarrow Hv$). The interface requires the preparation of a routine to perform the matrix–vector multiplication, to be called in $K\omega$. This interface allows extremely large matrices to be handled, since the matrix elements are internally generated in the ‘matrix–vector multiplication’ and not stored in the memory.

(ii) Selection of an appropriate solver

$K\omega$ provides four kinds of numerical solvers. An appropriate solver should be selected depending on whether the type of Hamiltonian $H$ and the frequency $\omega$ are complex or real, as shown in Table 1. It is noted that $H$ must be Hermitian or symmetric for a complex or real matrix, respectively. For an efficient calculation, the seed switching function \(*_\text{getresidual}\) is introduced in all methods.

(iii) Calculation

The calculation is performed in the following steps:

(a) Initialization using \(*_\text{init}\) functions to set and initialize internal variables in the library. For restart calculations, the initial values of the coefficients and the vector should be inputted at this step.

(b) Updating of the results iteratively using \(*_\text{update}\) functions, which are called alternately with the matrix–vector product routine in the loop to update the solution. \(*_\text{update}\) also performs the seed switching. At each step, the values of the 2-norm of the residual vector at each shift point are obtained using \(*_\text{getresidual}\) functions.

(c) Finalization using \(*_\text{finalize}\) functions to release the memories of the arrays stored in the library. Here, \(*\) indicates the name of the solver such as \(komega\_cg\_r\), \(komega\_cg\_c\), \(komega\_cocg\), and \(komega\_bicg\).

We summarize this subsection by showing the pseudo code for solving \((\omega I - H)x = b\) with the shifted CG method \((komega\_cg\_r)\) implemented in $K\omega$ in Algorithm 1. Users can use the shifted COCG method \((komega\_cocg)\) in the same way. The shifted BiCG method \((komega\_bicg)\) requires the allocations of two extra vectors, $\tilde{r}$ and $\tilde{q}$, for the shadow equation.

\[\text{Algorithm 1} \]

\[ (\omega I - H)x = b \]

\[ (komega\_cg\_r) \]

\[ (komega\_cocg) \]

\[ (komega\_bicg) \]

---

1 This pseudo code omits deallocations for simplicity.
Algorithm 1 Solve \((zI - H)x = b\) by CG method with \(K\omega\)

**procedure** ShiftedCG\((H, [z]), b)\)

\[
\text{ndim} \leftarrow \text{dim}(H) \\
\text{nz} \leftarrow \text{dim}([z]) \\
\text{allocate } x[\text{ndim} \times \text{nz}] \\
\text{allocate } r[\text{ndim}] \\
r \leftarrow b \\
\text{allocate } q[\text{ndim}] \\
call komega_cg_r_init(\text{ndim}, \text{nz}, \text{ndim}, x, [z]) \\
\text{repeat} \\
\quad q = Hr \\
\quad call komega_cg_r_update(q, r, x) > \text{update and seed} \\
\quad \text{switching} \\
\quad \text{until converged} \\
\quad call komega_cg_r_finalize() \\
\text{end procedure}
\]

3.2.2. Standalone program

To show detailed usage of \(K\omega\), an example standalone program ShiftK.out is also provided. ShiftK.out computes the diagonal element of the Green’s function:

\[
G_{a_0}(z) = a^\dagger (zI - H)^{-1}a.
\]

Here, \(a\) is expanded as \(\sum_i a_i n_i\), where \(n_i\) is the \(i\)-th basis vector of the Hilbert space. In the following, the usage of the software is explained.

(i) Preparation of an input file

The input parameters for ShiftK.out are categorized into four sections: cg, dyn, filename, and ham.

The cg section sets the numerical condition for the CG (or COCG, BiCG) method. maxloops is the maximum number of iterations and convfactor is the threshold value for the convergence criterion of the residual norm, i.e., \(\max_r |r^n_x| < 10^{-\text{convfactor}}\).

The dyn section specifies the parameters for computing the spectrum. By setting the parameters omegamin, omegamax, and nomega, the target frequencies are given as \(\omega_i = \text{omegamin} + i \times (\text{omegamax} - \text{omegamin})/\text{nomega}, (i = 0, \ldots, \text{nomega} - 1)\). In ShiftK.out, the calculation mode is chosen from normal, recalc, and restart by specifying the parameter calctype. normal is the mode for computing with the Krylov subspace from scratch. recalc is the mode for computing with the Krylov subspace generated in the previous calculation (See (iv) for details). restart is the mode for restarting the calculation from the previous run. In the computation of the Green’s function, the shifted BiCG or shifted COCG method is automatically selected when \(H\) is real or complex, respectively.

(a) Input of the matrix of the Hamiltonian

The filename section specifies the name of the input files for the matrix of the Hamiltonian inham or the initial excited vector invec. The file format of both files is the Matrix Market format \cite{27}. If invec is not specified, a random vector is used as the initial vector. An example input file for reading the Hamiltonian matrix and excited vector is as follows:

```plaintext
&filename
  inham = "Ham.dat"
  invec = "Excited.dat"
/
&cg
  maxloops = 100
  convfactor = 6
/
&dyn
  calctype = "normal"
  nomega = 100
  omegamin = (-2d0, 0.1d0)
  omegamax = (1d0, 0.1d0)
/

(b) Generation of the matrix of the Hamiltonian

For the trial use, the matrix of the Hamiltonian is generated in ShiftK.out mode. In this mode, the ham section is used instead of inham in the filename section. In the ham section, model parameters are specified to generate the Hamiltonian matrix for a one-dimensional spin chain model:

\[
H = \sum_{i=1}^{\text{nsite}} \begin{pmatrix}
  S_x^{(i)} & S_y^{(i)} & S_z^{(i)} \\
  -S_y^{(i)} & -S_z^{(i)} & 0 \\
  0 & 0 & S_z^{(i)}
\end{pmatrix},
\]

where \(S_j^{(i)}\) is a spin-1/2 operator at a site \(i\) with component \(j = x, y, z\). In the ham section, the following parameters are specified: the total site of the spin chain nsite and the parameters of the strength for spin–spin interactions \(J_x, J_y, J_z, \text{ and } D_z\). An example of using the mode to generate internally the Hamiltonian matrix is as follows:

```plaintext
&filename
/
&ham
  Jx = 1d0
  Jy = 1d0
  Jz = 1d0
  Dz = 1d0
/
&cg
  maxloops = 100
  convfactor = 6
/
&dyn
  calctype = "normal"
  nomega = 100
  outrestart = .TRUE.
/
```

(ii) Run

After preparing the input file, an executable ShiftK.out in terminal is run as follows:
\$ ShiftK.out namelist.def

Here, namelist.def is the name of the input file. The residual values at each step are output to the residual.dat file in the working directory.

(iii) Results

After running ShiftK.out, the output directory is automatically generated. In this directory, dynamical Green’s functions, the residual vector, and the coefficients are output to dynamicalG.dat, ResVec.dat, and TriDiagComp.dat, respectively.

(iv) Recalculation for additional data (optional)

The standalone program ShiftK.out provides, as an optional function, a recalculation function for additional data, as explained below. After the successful completion of ShiftK.out, users might like to obtain the solution at additional points $\sigma$ that are not obtained in the completed calculation. In such cases, users can calculate the solution at these points for the shifted equations with negligible operational costs, because the coefficients of $\{\alpha_n, \beta_n\}$ and the projected residual vectors $\{P_n\}$ are saved in the files TriDiagComp.dat and ResVec.dat, respectively, and the recurrence relations for the shifted equation Eqs. (24), (25), (26), and (27) can be solved without any expensive matrix–vector multiplications.

4. Applications in material science

In this section, we present several numerical results of K$\omega$ applied to quantum lattice models, to demonstrate typical applications in computational quantum physics. The quantum lattice model plays a crucial role for quantum many-body systems and is described by a large sparse matrix $H$. The applied studies in this section use the quantum lattice solver package $\mathcal{HF}$ [28]. $\mathcal{HF}$ was developed as an exact diagonalization solver and can treat a wide range of quantum lattice models, such as the Hubbard model, the Kondo model, and the Heisenberg model. K$\omega$ can be called from $\mathcal{HF}$ and the users of $\mathcal{HF}$ can use the shifted Krylov subspace solvers. Here, three typical examples are explained.

4.1. Calculation of spectra using ShiftK.out

The first example is a spectral calculation of Eq. (29) by the standalone program ShiftK.out explained in Sec. 3.2.2. A typical application for a quantum lattice model is the calculation of the dynamical correlation factor, since it is often used to analyze low-energy structures. The dynamical correlation factor is defined as

$$D(\omega, \eta) = -\frac{1}{\pi} \text{Im} \left[ \phi_0^\dagger A [(\omega - E_0 - i\eta)I - H]^{-1} B \phi_0 \right].$$

(31)

where $\phi_0$ is the ground state vector, $A$ and $B$ are the matrices to generate excited states, $\omega$ represents the frequency, and $\eta$ represents the smearing factor. For example, by taking $A = B = S^z$, we can calculate the dynamical spin structure factors, which can be directly measured by the neutron scattering experiments.

The key part in the calculation of the dynamical correlation factors is solving the linear equation defined as

$$\phi' = [(\omega - E_0 - i\eta)I - H]^{-1} B \phi_0.$$  

(32)

As an example of a calculation of dynamical Green’s functions by ShiftK.out, we present the calculation of the dynamical spin structure factors on a one-dimensional Heisenberg chain, whose Hamiltonian is defined as

$$H = J \sum_{i=0}^{L-1} S_i \cdot S_{i+1},$$

(33)

where $S$ represents the spin 1/2 operator and we take a magnetic interaction of $J = 1$ and system size of $L = 12$. The matrix data file of the Hamiltonian $H$ is generated by $\mathcal{HF}$ in the Matrix Market format. The dynamical spin structure factors are defined as

$$S(q, \omega, \eta) = -\frac{1}{\pi} \text{Im} \left[ \phi_0^\dagger S^{\dagger}(q) \left[(\omega - E_0 - i\eta)I - H\right]^{-1} S(q)\phi_0 \right].$$

(34)

where $S^{\dagger}(q) = \sum_{\alpha \beta} e^{iq\alpha} S^{\alpha \beta}$ and $q$ is the wave vector. The input file for ShiftK.out is given as follows ($E_0 = 0, -5.5 \leq \omega \leq 0.0, \eta = 0.02$):

```bash
&filename
  inham = "Ham.dat"
  invec = "excitedvec.dat"
/ &cg
  maxloops = 1000
  convfactor = 6
/ &dyn
  calctype = "normal"
  nomega = 1000
  omegamin = (-5.5, -0.02d0)
  omegamax = ( 0.0, -0.02d0)
/ Here, Ham.dat and excitedvec.dat are the input files for the Hamiltonian matrix $H$ and the excited vector $S^{\dagger}(q)\phi_0$. These files can be obtained using the quantum lattice solver $\mathcal{HF}$. The details are shown in Appendix C.

In Fig. 3 we show the numerical result of $S(q = \pi, \omega, \eta)$, Here, we shift $\omega$ by the ground state energy $E_0 = -5.387$. The frequencies where $S(q = \pi, \omega, \eta)$ have sharp peaks correspond to the excitation energies induced by $S_i(q = \pi)$. The convergence of the residuals can be checked in the standard output.

4.2. Calculation of internal eigenpairs

The second example is the calculation of internal eigenvalues by a contour-integral method called the Sakurai–Sugiura (SS) method [29]. The SS method is an efficient method for obtaining eigenpairs (eigenvalues and eigenvectors) in a specified eigenvalue range. Although an SS method software package has already been developed [30], this example illustrates how to use K$\omega$ in a standalone program.
Then, the matrix $S$ with vectors $s_{i,j}$ is constructed as

$$ S = (s_{0,0}, \ldots, s_{N_1-1,0}, s_{0,1}, \ldots, s_{N_1-1,1}, \ldots, s_{0,N_1-1}, \ldots, s_{N_1-1,N_1-1}). $$

By performing singular value decomposition for $S$, we obtain

$$ S = U \Sigma V^\dagger. $$

The number of non-zero singular values in $\Sigma$ is just that of the independent vectors $(M_{nz})$ in $N_{N_1 \times N_1}$ spanned by $s_{i,j}$. By using only the left-singular vectors with non-zero singular values, we can construct the matrix $\hat{U}$,

$$ \hat{U} = (u_0, \ldots, u_{M_{nz} - 1}). $$

Using $\hat{U}$, we project the original Hamiltonian on the small matrix (dimension $M_{nz} \times M_{nz}$) whose eigenvalues exist within $\Gamma$ as

$$ \hat{H} = \hat{U}^\dagger H \hat{U}. $$

By diagonalizing $\hat{H}$, we can obtain all the eigenvalues and eigenvectors which exist within $\Gamma$.

As an example, we apply the SS method to the 12-site Heisenberg chain model defined in Eq. (33). For simplicity, we obtain the eigenvalues around the ground state (the lowest eigenvalue). The matrix data file in the Matrix Market format (Ham.dat) is generated using $H\Phi$ as in Appendix C. Using Ham.dat, we can perform the SS method using Koo[31]. To perform the integration on the complex plane, we use the following points:

$$ z_j = \gamma + \rho \exp\left[\frac{2\pi i}{N_c} \left(j + \frac{1}{2}\right)\right], $$

where we take $\gamma = -5$, $\rho = 0.8$, and $N_c = 100$ as an example.

In Table B, we show the result of the SS method for the $L = 12$ Heisenberg chain, whose Hilbert dimension is 924. To see the convergence of the SS method, we show the obtained eigenvalues for several different conditions. By taking about 50 different bases, we can obtain the correct eigenvalues including degeneracies.

4.3. Calculation of optical conductivity using $H\Phi$

The third example is the calculation of a spectrum or dynamical correlation function using $H\Phi$, demonstrating that using Koo in $H\Phi$ can be a powerful tool for various problems. As an example, we show the optical conductivity in the extended Hubbard model, which is a typical model for strongly correlated electron systems. We note that the optical conductivity is often used for examining the metallic or insulating behavior of correlated electron systems. The optical conductivity can be calculated from the current–current correlation $I(\omega, \eta)$, which is defined as

$$ J_x = i \sum_{r}\langle c_{r+e_x,\sigma}^c c_{r,\sigma} - c_{r,\sigma}^c c_{r+e_x,\sigma}\rangle, $$

$$ I(\omega, \eta) = \text{Im} \left[ \phi_0^\dagger J_x (\omega - E_0 - i\eta) H^{-1} J_x \phi_0 \right]. $$
Table 3: Eigenvalues obtained by $\mathcal{H}\Phi$ (full diagonalization using LAPACK) and the SS method. For the SS method, we show results for several different $N_s$ and $N_l$, denoted as SS($N_s$,$N_l$). When the rank of SS($N_s$,$N_l$) is smaller than $n$, we do not obtain the $n$-th eigenvalues. This case is indicated as NA (not available) in the table. In the actual calculation, we discard singular values that are smaller than $10^{-5}$.

| $E_n$ | $\mathcal{H}\Phi$ | SS(10,1) | SS(10,2) | SS(10,5) |
|-------|------------------|----------|----------|----------|
| 0     | -5.387391        | -5.387391| -5.387391| -5.387391|
| 1     | -5.031543        | -5.031543| -5.031543| -5.031543|
| 2     | -4.777389        | -4.777389| -4.777389| -4.777389|
| 3     | -4.569374        | -4.569374| -4.569374| -4.569374|
| 4     | -4.569374        | -4.297689| -4.569374| -4.569374|
| 5     | -4.297689        | NA       | -4.297689| -4.297689|
| 6     | -4.297689        | NA       | -4.297689| -4.297689|

where $\mathbf{e}_i$ is the unit translational vector in the $x$ direction. The ground state vector $\Phi_0$ is calculated by an exact diagonalization solver built in $\mathcal{H}\Phi$. From the current–current correlation, the regular part (i.e., without the Drude part at $\omega = 0$) of the optical conductivity is defined as

$$\sigma_{\text{reg}}(\omega) = \frac{I(\omega, \eta) + I(-\omega, -\eta)}{\omega N_s},$$

(49)

where $N_s$ is the number of sites.

To directly compare the optical conductivity with previous studies, we calculate the optical conductivity in the extended Hubbard model defined as

$$H = -t \sum_{\langle i,j \rangle} \left[ c_{i\sigma}^\dagger c_{j\sigma} + c_{j\sigma}^\dagger c_{i\sigma} \right] + U \sum_i n_{i\uparrow} n_{i\downarrow} + V \sum_{\langle i,j \rangle} N_i N_j$$

(50)

where $c_{i\sigma}$ and $c_{i\sigma}^\dagger$ denote the annihilation and creation operators of an electron at site $i$ with spin $\sigma$, and $n_{i\sigma} = c_{i\sigma}^\dagger c_{i\sigma}$ represents the number operator of an electron at site $i$ with spin $\sigma$. We note that this model at quarter filling is an effective model for organic conductors. We perform a calculation for an $N_s = 4 \times 4$ system with $t = 1$, $U = 4$, $V = 3$, and $V' = 5$.

In Fig. 4, we show the result of the optical conductivity in the extended Hubbard model. This result is consistent with a previous study. We also show the residuals of $I(\omega, \eta)$ in Fig. 5. By calculating the residuals, we can check whether the obtained dynamical correlation functions are well converged. As mentioned above, this is one of the main advantages of the shifted Krylov method. Furthermore, by examining the $\omega$ dependence of the residuals, we can also identify bottlenecks for the convergence.

It is noteworthy that the matrix size becomes huge for the calculation of realistic models. In such cases, the matrix–vector product dominates the numerical cost of the shifted-Krylov method. To address this problem, we designed the $K\omega$ library to not directly include the matrix–vector product in the function, as explained in Sec. 3.2. If users can prepare the matrix–vector product function with high parallelization efficiency, the calculation time is greatly suppressed. In fact, the calculation of dynamical Green’s functions by $\mathcal{H}\Phi$ shows high parallelization efficiency, since the matrix–vector product is well parallelized. From this point of view, this library is considered to be suitable for large-scale calculations.

5. Summary

We developed the numerical library $K\omega$ for solving shifted linear equations with the shifted Krylov subspace methods. The present paper details applications for quantum many-body problems that appear in condensed matter physics. As a demonstration, numerical results including dynamical Green’s functions, optical conductivity, and eigenvalues through the SS method are shown. The method is also applicable to other computational physics areas, and hence $K\omega$ could be a useful numerical library in computational physics.

Figure 4: Optical conductivity for the extended Hubbard model. In this calculation, we take $\eta = 0.1$, $\omega_{\text{min}} = 0$, $\omega_{\text{max}} = 15.0$, and the number of $\omega$ as 2000.

Figure 5: Residual norm $\max_{\omega, \eta} |r_{\omega, \eta}^n|$ of $I(\omega, \eta)$ for several iteration steps. For the extended Hubbard model, we find that the residual for the high-energy part remains large. This result indicates that it is necessary to perform more than $10^4$ iterations to obtain a well-converged correlation function when the threshold value is set to $10^{-12}$. 
These vectors are iteratively generated by the following recurrence equations, as for the CG method,

\begin{align}
\rho_n &= \tilde{r}_n^\dagger r_n, \\
\alpha_n &= \frac{\rho_n}{\tilde{p}_n^\dagger\tilde{p}_n}, \\
x_{n+1} &= x_n + \alpha_n p_n, \\
r_{n+1} &= r_n - \alpha_n A^\dagger \tilde{p}_n, \\
\tilde{r}_{n+1} &= r_n - \alpha_n A^\dagger \tilde{p}_n, \\
\beta_n &= \frac{\rho_{n+1}}{\rho_n}, \\
p_{n+1} &= r_{n+1} + \beta_n p_n, \\
\tilde{p}_{n+1} &= r_{n+1} + \beta_n^\dagger \tilde{p}_n,
\end{align}

with the initialization \( r_0 = p_0 = b - Ax_0 \) and \( \tilde{p}_0 = \tilde{b} \).

The residuals \( r \) and \( \tilde{r} \) and the coefficient \( \alpha \) follow three term recurrences:

\begin{align}
r_{n+1} &= \left(1 + \frac{\alpha_n \beta_{n-1}}{\alpha_{n-1}}\right) r_n - \frac{\alpha_n \beta_{n-1}}{\alpha_{n-1}} r_{n-1}, \\
\tilde{r}_{n+1} &= \left(1 + \frac{\alpha_n \beta_{n-1}}{\alpha_{n-1}}\right)^* \tilde{r}_n - \frac{\alpha_n \beta_{n-1}}{\alpha_{n-1}}^* \tilde{r}_{n-1}, \\
\alpha_n &= \frac{\rho_n}{\tilde{r}_n^\dagger A r_n - \frac{\beta_{n-1}}{\alpha_{n-1}} \rho_n},
\end{align}

with \( \beta_{-1}/\alpha_{-1} = 0 \). \( \text{Ko} \) adopts these three term recurrences.

**Appendix B.2. Shifted equation**

When starting the BiCG iteration for the seed equation \( Ax = b \) and some shifted equation \( (A + \sigma \mathbf{I}) x = b \) with the same initial guess \( x_0 = x_0^0 = 0 \) and shadow residuals \( \tilde{r}_0 \), the \( n \)-th residuals \( r_n \) and \( \tilde{r}_n \) are parallel to each other [3],

\[ r_n^\sigma = \frac{1}{\pi_n} r_n. \]

The approximate solutions for the shifted equations are calculated via the same equations in the shifted CG method, Eqs. (21)-(25).

**Appendix C. Generate input files by \( \mathcal{H}\Phi \)**

This appendix shows the Hamiltonian matrix in the Matrix Market format and the excited vector generated using \( \mathcal{H}\Phi \). These files can be read by shiftk.out. As an example, we focus on the 12-site one-dimensional Heisenberg chain, whose Hamiltonian is defined in Eq. (13) and the excited vector \( S^\dagger(q) \Phi_0 \) in Eq. (34).

**Appendix C.1. Output Hamiltonian matrix**

An input file for \( \mathcal{H}\Phi \) is given as follows:

\begin{verbatim}
12 1

\end{verbatim}
L = 12
model = "Spin"
lattice = "chain"
method = "FullDiag"
J = 1.0
2Sz = 0
HamIO = "out"

Using this input file, we can obtain the output file of the Hamiltonian matrix in the Matrix Market format (zvo_Ham.dat).

To obtain the excited state, we first calculate the ground state by the following input file:

L = 12
model = "Spin"
lattice = "chain"
method = "CG"
J = 1.0
2Sz = 0
EigenVecIO = "Out"

The input file for calculating the excited state $S^z(q)\phi_0$ is given as follows:

L = 12
model = "Spin"
lattice = "chain"
method = "CG"
J = 1.0
2Sz = 0
CalcSpec="Normal"
SpectrumQL = 0.5
EigenVecIO = "In"
OutputExcitedVec = "Out"

Here, the wave vector is specified using $q = \pi$ in the above case, $S^z(q = \pi)\phi_0$ is obtained. For details, see the user's manual. After the calculations are finished, the output file of the excited vector zvo_excitedvec.dat (renamed as excitedvec.dat in Sec. C.2) is obtained.

**References**

[1] R. Haydock, Solid State Phys. 35, 215 (1980).
[2] E. Dagotto, Rev. Mod. Phys. 66, 763 (1994).
[3] A. Frommer, Computing 70, 87 (2003).
[4] R. Takayama, T. Hoshi, T. Fujiwara, T. Sogabe, and S.-L. Zhang, Phys. Rev. B 73, 165108/1 (2006).
[5] S. Yamamoto, T. Fujiwara, and Y. Hatsugai, Phys. Rev. B 76, 165114/1 (2007).
[6] T. Sogabe, T. Hoshi, S.-L. Zhang, and T. Fujiwara, Electron. Trans. Numer. Anal. 31, 126 (2008).
[7] S. Yamamoto, T. Sogabe, T. Hoshi, S.-L. Zhang, and T. Fujiwara, J. Phys. Soc. Jpn. 77, 114713 (2008).
[8] H. Teng, T. Fujiwara, T. Hoshi, T. Sogabe, S.-L. Zhang, and S. Yamamoto, Phys. Rev. B 83, 165103/1 (2011).
[9] L. Du, T. Sogabe, B. Yu, Y. Yamamoto, and S.-L. Zhang, J. Comput. Appl. Math. 235, 4095 (2011).
[10] T. Sogabe and S.-L. Zhang, East Asia J. on Appl. Math. 1, 97 (2011).
[11] T. Sogabe, T. Hoshi, S.-L. Zhang, and T. Fujiwara, J. Comput. Phys. 231, 5669 (2012).
[12] L. Du, T. Sogabe, and S.-L. Zhang, J. Comput. Appl. Math. 231, 5669 (2012).
[13] A. Imakura, T. Sogabe, and S.-L. Zhang, J. Math. Res. Appl. 33, 127 (2013).
[14] X.-M. Gu, T.-Z. Huang, J. Meng, T. Sogabe, H.-B. Li, and L. Li, Comput. Math. Appl. 68, 746 (2014).
[15] S. Saito, H. Tadano, and A. Imakura, Trans. JSIAM (in Japanese) 26, 157170 (2016).
[16] K. Soodhalter, SIAM J. on Sci. Comput. 38, A302 (2016).
[17] D.-L. Sun, T.-Z. Huang, B. Carpentieri, and Y.-F. Jing, J. Sci. Comput. 78, 746 (2018).
[18] Y. Futamura, H. Tadano, and T. Sakurai, JSIAM Letters 2, 127 (2010).
[19] K. Takai, M. Ikeda, T. Yamasaki, and C. Kaneta, J. Phys. Commun. 1, 045010 (2017).
[20] F. Giustino, M. L. Cohen, and S. G. Louie, Phys. Rev. B 81, 115105/1 (2010).
[21] T. Mizusaki, K. Kaneko, M. Homma, and T. Sakurai, Phys. Rev. C 82, 024310/1 (2010).
[22] S. Jin, A. Bulgac, K. Roche, and G. Wijewick, Phys. Rev. C 95, 044302 (2017).
[23] S. Iwase, T. Hoshi, and T. Ono, Phys. Rev. E 91, 063305 (2015).
[24] Y. Nagai, Y. Shinohara, Y. Futamura, and T. Sakurai, J. Phys. Soc. Jpn. 86, 014708/1 (2017).
[25] It is noted that the method for real-symmetric matrix $H$ can be applied to the problem for complex Hermitian matrix $H(zvq)$, because the eigenvalue problem of Eq. (3) with an $M \times M$ complex Hermitian matrix $H(zvq)$ is transformed into one with a $2M \times 2M$ real-symmetric matrix.
[26] M. R. Hestenes and E. Stiefel, J. Res. Nat. Bur. Stand. 49, 409 (1952).
[27] https://math.nist.gov/MatrixMarket/.
[28] M. Kawamura, K. Yoshimi, T. Misawa, Y. Yamaji, S. Todo, and N. Kawashima, Comput. Phys. Commun. 217, 180 (2017).
[29] T. Sakurai and H. Sugiura, J. Comput. Appl. Math. 159, 119 (2003).
[30] http://zpares.cs.tsukuba.ac.jp/.
[31] A sample code for the SS method is available at https://github.com/issp-center-dev/KOmega-Gallery.
[32] J. Merino, H. Seo, and M. Ogata, Phys. Rev. B 71, 125111 (2005).
[33] H. van der Vorst and J. Melissen, IEEE Trans. Mag. 26, 706 (1990).
[34] R. Fletcher, in *Numerical Analysis*, edited by G. A. Watson (Springer-Verlag Berlin Heidelberg, 1976), vol. 506 of *Lectures Notes in Mathematics*, pp. 73–89.