Generalized-restart Lanczos Method for Schrödinger Equation in Semiconductor Simulation

Hantian Zhang*[1] and Xiaolin Guo†[1]

[1]Department of Mathematics, ETH Zurich, 8092 Zurich, Switzerland

October 18, 2015

Abstract

In this paper, we develop the generalized-restart Lanczos iteration method and its inverse variant for solving the bounded time-independent Schrödinger equation under the finite element formalism. It avoids matrix inversion and enables the direct computation for several eigenpairs simultaneously, which makes generalized large-scale eigendecomposition problem fast solvable. Its inverse variant further reduces the algorithm complexity by almost a half, which leads to an efficient scheme for multiple lowest quantum states extraction. The numerical experiment has been conducted on 2D quantum harmonic oscillator, with the conjugate gradient method employed as the linear system equations solver inside the generalized Lanczos algorithm. Finally, the memory-distributed parallelism by MPI has been introduced to achieve the high performance computing.[1]

Introduction

In semiconductor simulation, it is required to solve the time-independent Schrödinger equation to extract its ground and excited states simultaneously [1]. Under finite element method (FEM), this spectral decomposition results in a generalized Hermitian eigenvalue problem (GHEP), which is the most computationally difficult part of the entire semiconductor simulation.

The commonly used approach to solve GHEP in form of $Ax = \lambda Mx$ is to compute the matrix inversion to get the form $M^{-1}Ax = \lambda x$, which can be solved by several strandard eigenvalue solver, e.g. restarted Arnoldi methods explained by Seed [2, Ch.7] and Krylov-Schur method proposed by Stewart [4]. While these techniques are not possible to solve the large-scale GHEP from PDEs, because the matrices A and M are sparse, but $M^{-1}A$ is a dense matrix. Thus, the single Lanczos iteration algorithm for GHEP explained by Ruhe [3, Ch.5] should be taken into account.

*M.Sc. Student; Email: hantian.zhang@student.ethz.ch

†M.Sc. Student; Email: guoxi@student.ethz.ch

[1]Code repository: https://github.com/hantian01/Generalized-restart-Lanczos-for-Schrodinger-equation

Wu and Simon [5] introduced the thick-restart Lanczos algorithm for standard eigenvalue problem, also they [6] proposed the single Lanczos iteration algorithm for GHEP. While the combination of these two techniques are somehow missing in literatures. Thus, we develop the generalized-restart Lanczos iteration algorithm and its inverse iteration variant to simultaneously extract several eigenpairs from large-scale GHEP.

In first section, we briefly present the finite element formalism of Schrödinger equation. In second section, the generalized-restart Lanczos iteration algorithm and its inverse variant is introduced. The detailed pseudocodes with comments are also provided. In third section, the efficient scheme for extracting several lowest quantum states is discussed. In last section, we conduct the numerical experiment on 2D quantum harmonic oscillator by means of linear FEM with conjugate gradient method employed as linear system equation solver, alongside with a specific memory-distributed parallelism approach as an illustration.

1 Finite Element Method for Schrödinger Equation

The time-independent Schrödinger equation of this problem with homogeneous Dirichlet boundary condition is in the form

$$\left[-\frac{\hbar^2}{2m}\nabla^2 + V(\vec{r})\right]\Psi(\vec{r}) = E\Psi(\vec{r}) \text{ in } \Omega \ , \quad \Psi = 0 \text{ on } \partial\Omega \tag{1}$$

Take the test and trial functions in the sobolov space of

$$H_0^1(\Omega) = \{v \in L^2(\Omega) \mid \operatorname{grad} v \in L^2(\Omega)^3, v_{|\partial\Omega} = 0\} \ ,$$

we can easily obtain the variational form by

$$\Psi \in H_0^1(\Omega) : \int_\Omega \frac{\hbar^2}{2m}\nabla\Psi(\vec{r})\cdot\nabla v(\vec{r})d\vec{r} + \int_\Omega V(\vec{r})\Psi(\vec{r})v(\vec{r})d\vec{r} = \int_\Omega E\Psi(\vec{r})v(\vec{r})d\vec{r} \ , \quad \forall v \in H_0^1(\Omega) \ . \tag{2}$$

Denote the nodal basis by $\mathfrak{B}_N = \{b_N^1, \ldots, b_N^N\}$, then the trial function can be expanded as

$$\Psi_N = \sum_{i=1}^{N} \mu_i b_N^i \ .$$

We finally obtain the discretized matrix form

$$\mathbf{A}\vec{\mu} = \lambda\mathbf{M}\vec{\mu} \ , \tag{3}$$

where

$$\begin{aligned}
&\mathbf{A}_{ij} = \frac{\hbar^2}{2m}\int_\Omega \nabla b_N^i(\boldsymbol{x})\nabla b_N^j(\boldsymbol{x})d\boldsymbol{x} + \int_\Omega V(\boldsymbol{x})b_N^i(\boldsymbol{x})b_N^j(\boldsymbol{x})d\boldsymbol{x} \ , \\
&\mathbf{M}_{ij} = \int_\Omega b_N^i(\boldsymbol{x})b_N^j(\boldsymbol{x})d\boldsymbol{x} \ , \\
&\vec{\mu} = (\mu_i)_{i=1}^N \in \mathbb{R}^N \ , \quad \lambda = E \ .
\end{aligned}$$

So far, solving Schrödinger equation has been converted to a real-valued generalized Hermitian eigenvalue problem.

2 Generalized Hermitian Eigenvalue Problem Solver

The solvable GHEP is constructed in symmetric form of

$$Ax = \lambda M x \ ,$$

where $A \in \mathbb{R}^{n \times n}$ is sparse symmetric, and $M \in \mathbb{R}^{n \times n}$ is sparse symmetric positive definite.

In semiconductor simulation, only several lowest eigenpairs need to be solved in order to approximate the electronic gas density. Thus, we develop the following algorithm to extract few eigenpairs from the large-scale problem.

2.1 Generalized restart Lanczos algorithm for GHEP

Here, we develop the generalized restart Lanczos algorithm (1) for GHEP to obtain its m largest modulus eigenvalues ($m \ll n$). We construct the basis $V_m \in \mathbb{R}^{n \times m}$ by satisfying M-orthogonalization

$$\begin{aligned} V_m^T M V_m &= I_m \ , \\ V_m^T A V_m &= T_m \ , \\ W_m &= M V_m \ , \end{aligned}$$

where the matrix $T_m \in \mathbb{R}^{m \times m}$ is in form of

$$T_m = \begin{pmatrix} \alpha_1 & \beta_1 & 0 & \cdots & & 0 \\ \beta_1 & \alpha_2 & \beta_2 & & & \\ 0 & \beta_2 & \ddots & & & \vdots \\ \vdots & & & \ddots & & \\ & & & & \alpha_{m-1} & \beta_{m-1} \\ 0 & & \cdots & & \beta_{m-1} & \alpha_m \end{pmatrix} \tag{4}$$

The m eigenpairs $\{(\lambda_i, s_i)\}_m$ of T_m are called Ritz pairs, which can be solved simultaneously by QR algorithm (4 in Appendix). Then the approximate eigenpairs of the GHEP can be computed by

$$(\lambda_i, x_i) = (\lambda_i, V_m s_i) \ . \tag{5}$$

In particular the induced linear system equations (LSE) $Mx = b$ is required to be solved, which is the major difference from standard Lanczos algorithm.

2.1.1 Algorithm details

Each Lanczos iteration loop involves:

1. Gram-Schmidt M-orthogonalization
2. Linear system solver $Mx = y$, (for inverse iteration is $Ax = y$)
3. QR algorithm for finding m Ritz pairs of T_m

4. Sparse matrix-vector multiplication (SMVM)
5. Dot product and other vector operations

Among them, the LSE solver occupies most of computation time, thus the parallelism on this part is usually necessary.

The restarting technique employed in the algorithm can be read as, within k-th Lanczos iteration loop,

- if the k^{th} eigenpairs converges, then freeze the k^{th} column of V_m and T_m, and restart the $(k+1)^{th}$ iteration by using M-orthogonalized k^{th} eigenvector as initial condition.
- otherwise, directly restart the k^{th} iteration by using M-orthogonalized approximated k^{th} eigenvector as initial condition.

The convergence of k-th eigenpair $(\lambda_k, \mathbf{s}_k)$ is checked by the residual norm

$$\epsilon_k = \beta_m s_k^{(m)} \ ,$$

where $s_k^{(m)}$ is the last component of k-th Ritz vector $s_k \in \mathbb{R}^m$

2.1.2 Pseudocode

The pseudocode is listed below in Algorithm 1.

Algorithm 1: Generalized restart Lanczos algorithm for GHEP

```
1:  procedure COMPUTE EIGENPAIRS OF $\mathbf{A}x = \lambda\mathbf{M}x$
2:  Input: $\mathbf{A}, \mathbf{M}$.        Output: $T_m, V_m, (\lambda_i, \mathbf{s}_i)$ Ritz eigenpairs.
3:  Start: With $\mathbf{A} \in \mathbb{R}^{n\times n}, \mathbf{M} \in \mathbb{R}^{n\times n}$, choose subspace dimension $m$, and a normalized
    initial vector $\mathbf{q} \in \mathbb{R}^n$, initialize $k = 1, \mathbf{w}_0 = 0$. Take $T_m \in \mathbb{R}^{m\times m}$ and $V_m = [\mathbf{v}_1, \ldots, \mathbf{v}_m] \in$
    $\mathbb{R}^{n\times m}$.
4:      $\mathbf{r} = \mathbf{M}\mathbf{q}$;
5:      $\beta_0 = |\mathbf{q}^T\mathbf{r}|^{\frac{1}{2}}$;
6:  k-th eigenvalue loop:
7:      while $k \le m$ do
8:          for $j = k, \ldots, m$ do                              ▷ Lanczos iteration
9:              $\mathbf{w}_j = \mathbf{r}/\beta_{j-1}$;
10:             $\mathbf{v}_j = \mathbf{q}/\beta_{j-1}$;
11:             $\mathbf{r} = \mathbf{A}\mathbf{v}_j$;
12:             $\mathbf{r} = \mathbf{r} - \mathbf{w}_{j-1}\beta_{j-1}$;
13:             $\alpha_j = \mathbf{v}_j^T\mathbf{r}$;
14:             $\mathbf{r} = \mathbf{r} - \mathbf{w}_j\alpha_j$;
15:             for $i = 1, \ldots, j$ do            ▷ Gram-Schmidt M-orthogonalization
16:                 $y = \mathbf{v}_i^T\mathbf{r}$;
17:                 $\mathbf{r} = \mathbf{r} - \mathbf{w}_i y$;
18:             end for
19:             Solve $\mathbf{M}\mathbf{q} = \mathbf{r}$ for $\mathbf{q}$;                  ▷ Use LSE solver
20:             $\beta_j = |\mathbf{q}^T\mathbf{r}|^{\frac{1}{2}}$;
```

```
            $t_{j,j} = \alpha_j$, $t_{j,j+1} = t_{j+1,j} = \beta_j$;    ▷ Assemble $T_m$, see eq. (4)
            **if** $\beta_j = 0$ **then**    ▷ Invariant subspace found
                Call function QR($T_m$);
                **return** $V_m \in \mathbb{R}^{n\times j}, T_m \in \mathbb{R}^{j\times j}, \{(\lambda_i, \mathbf{s}_i)\}_j$;
            **end if**
        **end for**
        Solve $k$ Ritz eigenpairs $(\lambda_i, \mathbf{s}_i)$ of $T_m$;    ▷ Use QR Alg. (4)
        $flag = 1$;
        **while** flag == 1 **do**
            $\epsilon_k = |\beta_m s_k^{(m)}|$;    ▷ Check residual norm
            **if** $\epsilon_k \to 0$ **then** accept $(\lambda_k, \mathbf{s}_k)$ as eigenpair;
                Set $k = k + 1$;
                Set $\mathbf{u}_k = V_m \mathbf{s}_k$;    ▷ $k$-th approximate eigenvector
                $\mathbf{q} = \mathbf{u}_k / \|\mathbf{u}_k\|$;    $\mathbf{r} = \mathbf{M}\mathbf{q}$;    ▷ Initialize restarting vectors
                **for** $i = 1, \dots, k-1$ **do**
                    $y = \mathbf{v}_i^T \mathbf{r}$;    $\mathbf{r} = \mathbf{r} - \mathbf{w}_i y$;
                **end for**
                Solve $\mathbf{M}\mathbf{q} = \mathbf{r}$ for $\mathbf{q}$;    ▷ Use LSE solver
                $\beta_{k-1} = |\mathbf{q}^T \mathbf{r}|^{\frac{1}{2}}$;
                **if** $k \geq m$ **then**
                    **return** $V_m \in \mathbb{R}^{m\times n}, T_m \in \mathbb{R}^{m\times m}, \{(\lambda_i, \mathbf{s}_i)\}_m$;
                **end if**
            **else**
                Set $\mathbf{u}_k = V_m \mathbf{s}_k$;    ▷ Re-initialize starting vectors
                $\mathbf{q} = \mathbf{u}_k / \|\mathbf{u}_k\|$;    $\mathbf{r} = \mathbf{M}\mathbf{q}$;
                **for** $i = 1, \dots, k-1$ **do**
                    $y = \mathbf{v}_i^T \mathbf{r}$;    $\mathbf{r} = \mathbf{r} - \mathbf{w}_i y$;
                **end for**
                Solve $\mathbf{M}\mathbf{q} = \mathbf{r}$ for $\mathbf{q}$;    ▷ Use LSE solver
                $\beta_{k-1} = |\mathbf{q}^T \mathbf{r}|^{\frac{1}{2}}$;
                $flag = 0$;    ▷ Restart Lanczos iteration
            **end if**
        **end while**
    **end while**
**end procedure**
```

2.2 Inverse variant of generalized restart Lanczos algorithm

The inverse iteration algorithm (2) computes the Ritz pairs of $\{(\theta_i, s_i)\}_m$, with relation to the approximate eigenpairs of GHEP by

$$(\lambda_i, x_i) = \left(\frac{1}{\theta_i}, W_m s_i\right) ,$$

with $W_m = MV_m$.

Instead of solving LSE $Mx = b$ twice in Lanczos algorithm (1), the inverse variant only solve LSE $Ax = b$ once. Thus the inverse variant significant reduce the algorithm complexity by about a half, as the LSE part takes up most of time.

2.2.1 Pseudocode

The pseudocode of inverse variant is listed in Algorithm 2.

Algorithm 2: Inverse generalized restart Lanczos algorithm for GHEP

```
 1: procedure COMPUTE EIGENPAIRS OF $\mathbf{M}x = \theta\mathbf{A}x$
 2: Input: $\mathbf{A}, \mathbf{M}$.        Output: $T_m, W_m, (\theta_i, \mathbf{s}_i)$ Ritz eigenpairs.
 3: Start: With $\mathbf{A} \in \mathbb{R}^{n\times n}, \mathbf{M} \in \mathbb{R}^{n\times n}$, choose subspace dimension $m$, and a normalized initial vector $\mathbf{r} \in \mathbb{R}^n$, initialize $k = 1, \mathbf{w}_0 = 0$. Take $T_m \in \mathbb{R}^{m\times m}$ and $W_m = [\mathbf{w}_1, \dots, \mathbf{w}_m] \in \mathbb{R}^{n\times m}$.
 4:     $\mathbf{q} = \mathbf{M}\mathbf{r}$;
 5:     $\beta_0 = |\mathbf{q}^T\mathbf{r}|^{\frac{1}{2}}$;
 6: k-th eigenvalue loop:
 7:     while $k \le m$ do
 8:         for $j = k, \dots, m$ do                              ▷ Lanczos iteration
 9:             $\mathbf{w}_j = \mathbf{r}/\beta_{j-1}$;
10:             $\mathbf{v}_j = \mathbf{q}/\beta_{j-1}$;
11:             Solve $\mathbf{A}\mathbf{r} = \mathbf{v}_j$ for $\mathbf{r}$;                     ▷ Use LSE solver
12:             $\mathbf{r} = \mathbf{r} - \mathbf{w}_{j-1}\beta_{j-1}$;
13:             $\alpha_j = \mathbf{v}_j^T\mathbf{r}$;
14:             $\mathbf{r} = \mathbf{r} - \mathbf{w}_j\alpha_j$;
15:             for $i = 1, \dots, j$ do                ▷ Gram-Schmidt M-orthogonalization
16:                 $y = \mathbf{v}_i^T\mathbf{r}$;
17:                 $\mathbf{r} = \mathbf{r} - \mathbf{w}_i y$;
18:             end for
19:             $\mathbf{q} = \mathbf{M}\mathbf{r}$;
20:             $\beta_j = |\mathbf{q}^T\mathbf{r}|^{\frac{1}{2}}$;
21:             $t_{j,j} = \alpha_j$, $t_{j,j+1} = t_{j+1,j} = \beta_j$;          ▷ Assemble $T_m$, see eq. (4)
22:             if $\beta_j = 0$ then                           ▷ Invariant subspace found
23:                 Call function QR($T_m$);
24:                 return $V_m \in \mathbb{R}^{n\times j}, T_m \in \mathbb{R}^{j\times j}, \{(\lambda_i, \mathbf{s}_i)\}_j$;
25:             end if
26:         end for
27:         Solve $k$ Ritz eigenpairs $(\lambda_i, \mathbf{s}_i)$ of $T_m$;               ▷ Use QR Alg. (4)
28:         $flag = 1$;
29:         while flag == 1 do
30:             $\epsilon_k = |\beta_m s_k^{(m)}|$;                              ▷ Check residual norm
31:             if $\epsilon_k \to 0$ then accept $(\lambda_k, \mathbf{s}_k)$ as eigenpair;
32:                 Set $k = k + 1$;
33:                 Set $\mathbf{u}_k = W_m\mathbf{s}_k$;                   ▷ $k$-th approximate eigenvector
34:                 $\mathbf{r} = \mathbf{u}_k/\|\mathbf{u}_k\|$;                      ▷ Initialize restarting vectors
35:                 for $i = 1, \dots, k-1$ do
36:                     $y = \mathbf{v}_i^T\mathbf{r}$;    $\mathbf{r} = \mathbf{r} - \mathbf{w}_i y$;
37:                 end for
38:                 $\mathbf{q} = \mathbf{M}\mathbf{r}$;
39:                 $\beta_{k-1} = |\mathbf{q}^T\mathbf{r}|^{\frac{1}{2}}$;
40:                 if $k \ge m$ then
41:                     return $V_m \in \mathbb{R}^{m\times n}, T_m \in \mathbb{R}^{m\times m}, \{(\lambda_i, \mathbf{s}_i)\}_m$;
42:                 end if
43:             else
```

```
44:                 Set $\mathbf{u}_k = W_m\mathbf{s}_k$;                    ▷ Re-initialize starting vectors
45:                 $\mathbf{r} = \mathbf{u}_k/\|\mathbf{u}_k\|$;
46:                 for $i = 1, \ldots, k-1$ do
47:                     $y = \mathbf{v}_i^T\mathbf{r}$;   $\mathbf{r} = \mathbf{r} - \mathbf{w}_i y$;
48:                 end for
49:                 $\mathbf{q} = \mathbf{Mr}$;
50:                 $\beta_{k-1} = |\mathbf{q}^T\mathbf{r}|^{\frac{1}{2}}$;
51:                 $flag = 0$;                                   ▷ Restart Lanczos iteration
52:             end if
53:         end while
54:     end while
55: end procedure
```

3 Extraction of m lowest eigenstates

Equipped with the above algorithms, we provide an efficient scheme for finding m lowest quantum eigenstates:

1. Apply Lanczos iteration (1) for $\mathbf{A}\vec{\mu} = \lambda\mathbf{M}\vec{\mu}$ to extract the largest eigenvalue λ, the upper bound is in form of
$$\tau = \lambda + 1 \ .$$

2. Apply the following transformation to ensure the positive definite property of left-hand side
$$\begin{aligned}(\tau\mathbf{M} - \mathbf{A})\vec{\mu} &= (\tau - \lambda)\mathbf{M}\vec{\mu} \ ,\\ \tilde{\mathbf{A}}\vec{\mu} &= \tilde{\lambda}\mathbf{M}\vec{\mu} \ .\end{aligned}$$

3. Apply Lanczos iteration (1) again for $\tilde{\mathbf{A}}\vec{\mu} = \tilde{\lambda}\mathbf{M}\vec{\mu}$ to extract lowest eigenvalue (ground state energy) by
$$\lambda_{gnd} = \tau - \tilde{\lambda} \ .$$

4. Determine the minimum shift by
$$\sigma = \begin{cases} 2|\lambda_{gnd}| \ , & \lambda < 0 \\ 0 \ , & \lambda \geq 0 \end{cases} \ ,$$
and make transformation
$$\begin{aligned}(\sigma\mathbf{M} + \mathbf{A})\vec{\mu} &= (\sigma + \lambda)\mathbf{M}\vec{\mu} \ ,\\ \hat{\mathbf{A}}\vec{\mu} &= \hat{\lambda}\mathbf{M}\vec{\mu} \ .\end{aligned}$$

5. Apply inverse Lanczos iteration (2) for $\hat{\mathbf{A}}\vec{\mu} = \hat{\lambda}\mathbf{M}\vec{\mu}$ to extract m lowest eigenstates by
$$\begin{aligned}\lambda_i &= \frac{1}{\hat{\theta}_i} - \sigma \ ,\\ \vec{\mu}_i &= W_m\mathbf{s}_i \ , \quad i = 1, \ldots, m \ .\end{aligned}$$

Remark. *The convergence rate of computing eigenvalues is proportional to* $(\lambda_i/\lambda_{i+1})^2$*, which makes extracting the lowest eigenvalue in Step 2 become very difficult, because the largest eigenvalue will increase dramatically with growing N size. While Step 1-4 is to make sure the positive definite property of* $\hat{A}$ *in the last step, thus the accuracy of lowest eigenvalue before Step 5 is not necessary. Therefore, we employ larger threshold and fewer mesh nodes for Lanczos iteration before Step 5, then extract accurate results from the last step with a faster inverse iteration algorithm.*

4 Numerical Experiment on 2D

Recall the Schrödinger equation in matrix form as a generalized eigenvalue problem of

$$\mathbf{A}\vec{\mu} = \lambda \mathbf{M}\vec{\mu} \ ,$$

here due to the homogeneous Dirichlet boundary condition, we restrict $\vec{\mu}$ only to the interior nodes.

4.1 Linear FEM on 2D triangular mesh

The domain $\Omega = [x_a, x_b] \otimes [y_a, y_b]$ is discretized by triangle elements as shown in Fig. 1.

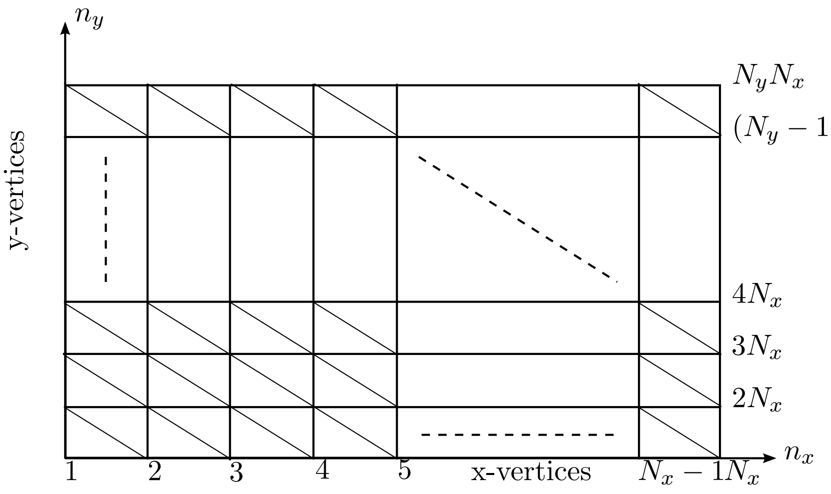


Figure 1: Rectangle Mesh global vertices index ordering

The nodal basis $\mathfrak{B}_N$ is chosen by

$$\mathfrak{B} = \{b_N^1, \dots, b_N^N\} \ ,$$
$$b_N^j(\boldsymbol{x}_i) = \delta_{ij} := \begin{cases} 1, \ i = j \\ 0, \ i \neq j \end{cases} , \quad i, j \in \{1, \dots, N\}.$$

4.1.1 Local matrix computation

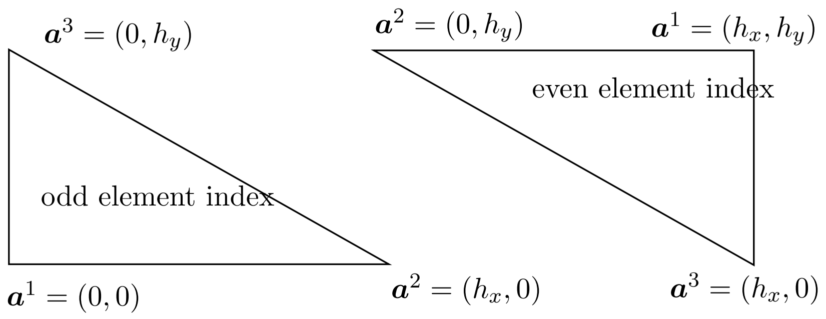
The local vertices in the reference frame is shown in Fig. 2. The element width and height are given by

$$h_x = \frac{x_b - x_a}{N_x - 1} \ , \quad h_y = \frac{y_b - y_a}{N_y - 1} \ ,$$

and the area of triangle element K is

$$|K| = \frac{1}{2} h_x h_y \ .$$

Fortunately, in this local vertices ordering, the local Galerkin matrix is the same with odd-/even element index.



$\boldsymbol{a}^3 = (0, h_y)$
$\boldsymbol{a}^2 = (0, h_y)$
$\boldsymbol{a}^1 = (h_x, h_y)$
even element index
odd element index
$\boldsymbol{a}^1 = (0, 0)$
$\boldsymbol{a}^2 = (h_x, 0)$
$\boldsymbol{a}^3 = (h_x, 0)$

Figure 2: Local vertices in reference frame

The first bilinear term on triangle $K \in \mathcal{M}$ can be analytically obtained by using barycentric function on triangle element

$$a_{1|K}(b_N^i, b_N^j) = \sigma \int_K \nabla b_{N|K}^i \cdot \nabla b_{N|K}^j d\boldsymbol{x} = \frac{\sigma}{2} h_x h_y \begin{pmatrix} \frac{1}{h_x^2} + \frac{1}{h_y^2} & -\frac{1}{h_x^2} & -\frac{1}{h_y^2} \\ -\frac{1}{h_x^2} & \frac{1}{h_x^2} & 0 \\ -\frac{1}{h_y^2} & 0 & \frac{1}{h_y^2} \end{pmatrix} . \tag{6}$$

The second bilinear form on triangle $K \in \mathcal{M}$ can be computed by

$$a_{2|K}(b_N^i, b_N^j) = \int_K V_{|K}(\boldsymbol{x}) b_{N|K}^i b_{N|K}^j d\boldsymbol{x} \ .$$

We employ the mid-point Gaussian quadrature rule on standard triangle element by

$$\begin{aligned} &T_{st} = \{(\xi, \eta) : 0 \leq \xi, \eta, \ \xi + \eta \leq 1\} \ , \\ &\int_{T_{st}} g(\xi, \eta) d\xi d\eta = \frac{1}{6} \left[g\left(0, \frac{1}{2}\right) + g\left(\frac{1}{2}, 0\right) + g\left(\frac{1}{2}, \frac{1}{2}\right) \right] \ . \end{aligned}$$

Then we obtain the second local Galerkin matrix on the reference frame by

$$a_{2|K}(b_N^i, b_N^j) = \frac{|K|}{12} \begin{pmatrix} V_{12} + V_{13} & V_{12} & V_{13} \\ V_{12} & V_{12} + V_{23} & V_{23} \\ V_{13} & V_{23} & V_{13} + V_{23} \end{pmatrix} \ , \tag{7}$$

where
$$V_{12} = V(\frac{h_x}{2}, 0) \ , \quad V_{13} = V(0, \frac{h_y}{2}) \ , \quad V_{23} = V(\frac{h_x}{2}, \frac{h_y}{2}) \ .$$

The third bilinear form on the right-hand side on triangle $K \in \mathcal{M}$ can be analytically calculated by

$$m_{|K}(b_N^i, b_N^j) = \int_K b^i_{N|K} b^j_{N|K} d\boldsymbol{x} = \frac{|K|}{12} \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix} \tag{8}$$

4.1.2 Global matrices

The global Galerkin matrices are symmetric sparse block-tridiagonal. The left-hand side Hamiltonian matrix has the form of

$$\mathbf{A} = \begin{pmatrix} \boldsymbol{A}_1 & \boldsymbol{B}_1 & & \cdots & 0 \\ \boldsymbol{B}_1^T & \boldsymbol{A}_2 & \boldsymbol{B}_2 & & \vdots \\ & \boldsymbol{B}_2^T & \ddots & \ddots & \\ \vdots & & \ddots & \boldsymbol{A}_{Ny-1} & \boldsymbol{B}_{Ny-1} \\ 0 & \cdots & & \boldsymbol{B}_{Ny-1}^T & \boldsymbol{A}_{Ny} \end{pmatrix} \in \mathbb{R}^{N \times N} \ , \tag{9}$$

where $N = N_x N_y$, and $\boldsymbol{A}_k, \boldsymbol{B}_k$ have the form of

$$\boldsymbol{A}_k = \begin{pmatrix} \times & \times & & \cdots & 0 \\ \times & \times & \times & & \vdots \\ & \times & \ddots & \ddots & \\ \vdots & & \ddots & \times & \times \\ 0 & \cdots & & \times & \times \end{pmatrix} \in \mathbb{R}^{N_x \times N_x} \ , \quad \boldsymbol{B}_k = \begin{pmatrix} \times & 0 & & \cdots & 0 \\ \times & \times & 0 & & \vdots \\ & \times & \ddots & \ddots & \\ \vdots & & \ddots & \times & 0 \\ 0 & \cdots & & \times & \times \end{pmatrix} \in \mathbb{R}^{N_x \times N_x} \ . \tag{10}$$

The right-hand side matrix also has the similar form

$$\mathbf{M} = \begin{pmatrix} \boldsymbol{M}_1 & \boldsymbol{C}_1 & & \cdots & 0 \\ \boldsymbol{C}_1^T & \boldsymbol{M}_2 & \boldsymbol{C}_2 & & \vdots \\ & \boldsymbol{C}_2^T & \ddots & \ddots & \\ \vdots & & \ddots & \boldsymbol{M}_{Ny-1} & \boldsymbol{C}_{Ny-1} \\ 0 & \cdots & & \boldsymbol{C}_{Ny-1}^T & \boldsymbol{M}_{Ny} \end{pmatrix} \in \mathbb{R}^{N \times N} \ , \tag{11}$$

where $\boldsymbol{M}_k, \boldsymbol{C}_k$ have the same form as $\boldsymbol{A}_k, \boldsymbol{B}_k$.

4.2 Convergence evaluation

To validate the finite element method, we evaluate the discretized error with respect to L2 norm by
$$\|u_N - u\|_{L^2} \ ,$$
where u is exact analytical solution, and u_N is the discretized solution obtained from FEM with N interior nodes. The second-order algebraic convergence of L2 error norm is expected for linear FEM in case of h-refinement.

4.2.1 Test on 2D quantum harmonic oscillator

We test the 2D harmonic oscillator model with

$$V(\boldsymbol{x}) = \frac{1}{2} m\omega^2 |\boldsymbol{x}|^2 \ .$$

Then the analytic energy levels are

$$E_n = \hbar\omega(n_x + n_y + 1) \ , \quad n_x, n_y = 0, 1, 2, \dots \ , \tag{12}$$

Here we employ the Conjugate Gradient (CG) algorithm (5) as the LSE solver, and set $m = 1$, $\hbar = 1$, $\omega = 1$, then the L2 error norm of eigen energies are shown in Fig.3, where the second-order algebraic convergence has been observed successfully. The ground and excited state wave functions are shown in Fig. 4.

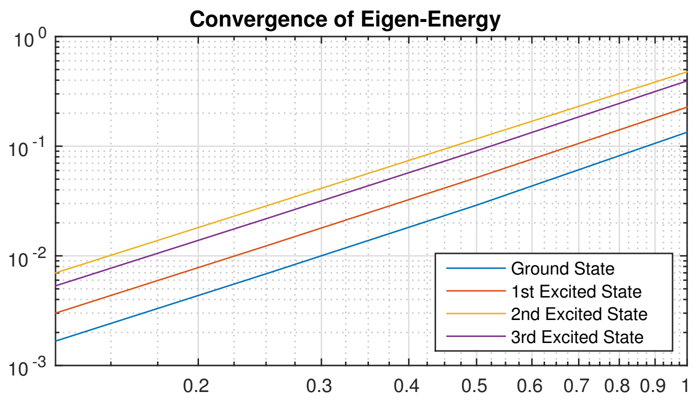


Figure 3: Error plot of eigen energy w.r.t L2-norm, 2nd order algebraic convergence observed

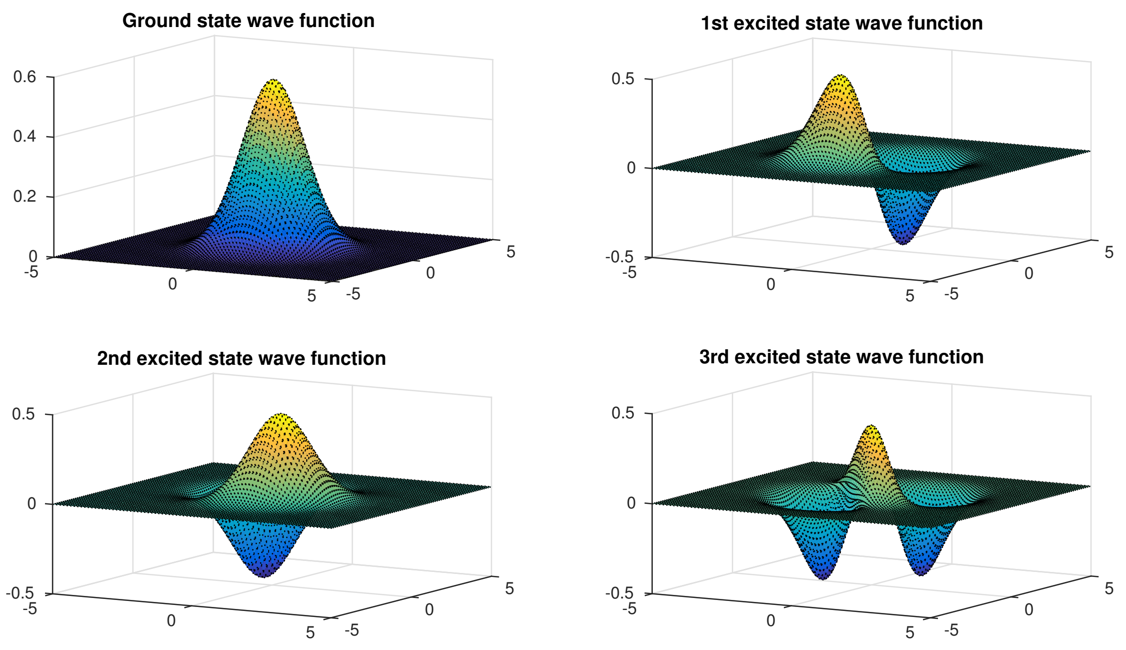


Figure 4: Wave function of bound states

4.3 Distributed-memory parallelism

Since CG algorithm (5)) for solving LSE takes up most of the time, we introduce the parallelism of this part with MPI.

CG algorithm mainly consists of sparse matrix-vector multiplication (SMVM) and dot product, which need to be both conducted distributively. For SMVM, the left-hand side matrix in our problem is always tridiagonal. This leads to a straight-forward partitioning strategy: partitioning by rows. Correspondingly, vector `d` which is used in the matrix-vector multiplication is divided as is illustrated in Figure 5. Particularly, the communication of ghost region and the SMVM computation on non-ghost part can be performed at the same time.

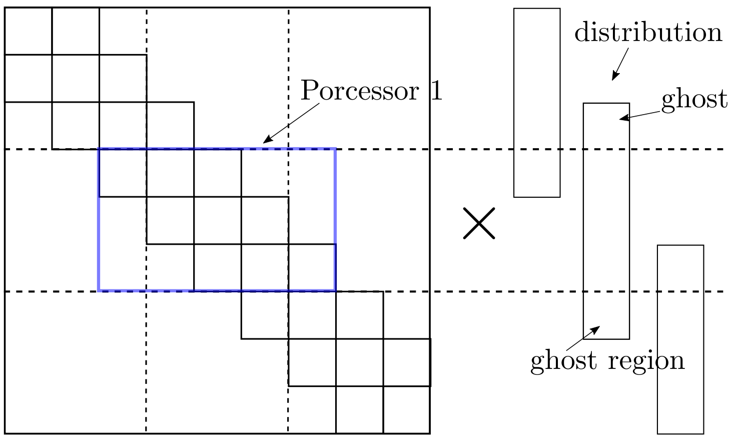


Figure 5: Block partition in MPI for sparse matrix-vector multiplication

This partition strategy results in the following MPI parallel version of CG algorithm (5)

Algorithm 3: CG algorithm with MPI

```
 1: procedure THE MAIN LOOP IN CG
 2:     for $k = 0, 1, \ldots, MAXITER$ do
 3:         rr0 = rr1
 4:         Adloc = smvm_mpi(A, d_loc)
 5:         dAd = dot_product_mpi(d_loc, Adloc)
 6:         $\alpha$ = rr0/dAd
 7:         $x_{loc} = x_{loc} + \alpha d_{loc}$
 8:         $r_{loc} = r_{loc} - \alpha$Adloc
 9:         rr1 = dot_product_mpi(r_loc, r_loc)
10:         if rr1 $< tol^2$ then
11:             return
12:         end if
13:         $\beta$ = rr1/'rr0
14:         $d_{loc,nonghost} = r_{loc} + \beta d_{loc,nonghost}$
15:         update $d_{loc,ghost}$ with MPI_Isend() and MPI_Irecv()
16:     end for
17: end procedure
18: procedure DP = DOT_PRODUCT_MPI(X_LOC, Y_LOC)
19:     dp_loc = dot_product(x_loc, y_loc)
20:     dp = MPI_Allreduce(dp_loc,MPI_SUM)
21: end procedure
```

```
22: procedure Y = SMVM_MPI()(A, d_loc)
23:     y = smvm(A_nonghost, d_loc_nonghost)
24:     MPI_Waitall() for d_loc_ghost to arrive
25:     y += smvm(A_ghost, d_loc_ghost)
26: end procedure
```

4.3.1 Scaling

Here we tend to provide a strong scaling plot on MPI speedup as an illustration. To avoid the degeneracy of eigen energies, we modify the potential term by $V(x,y) = \frac{1}{2}m\omega^2(x^2+8y^2)$. In order to obtain an approximate reference speedup, the experimental sequential percentage results of given problem size is list in Table 1.

N size	64^2	128^2	256^2	512^2	1024^2
Sequential percent	17.05%	11.61%	9.19%	5.53%	3.01%

Table 1: Sequential percentage of tests with single processor, run on Piz Daint, CSCS, Switzerland. flag: CC -O3 -Wall

Then the reference speedup is estimated by Amdahl's law by

$$speedup(P) = \frac{1}{s + (1-s)/P} .$$

The strong scaling plot of MPI speedup is shown in Fig. 6. The occasionally excessive speedup results from $N = 256^2, 512^2$ cases are due to the restriction of cache size. The matrix size cannot fit into cache of single processor, but fit into those of multiple processors, which results in a much better locality and higher performance.

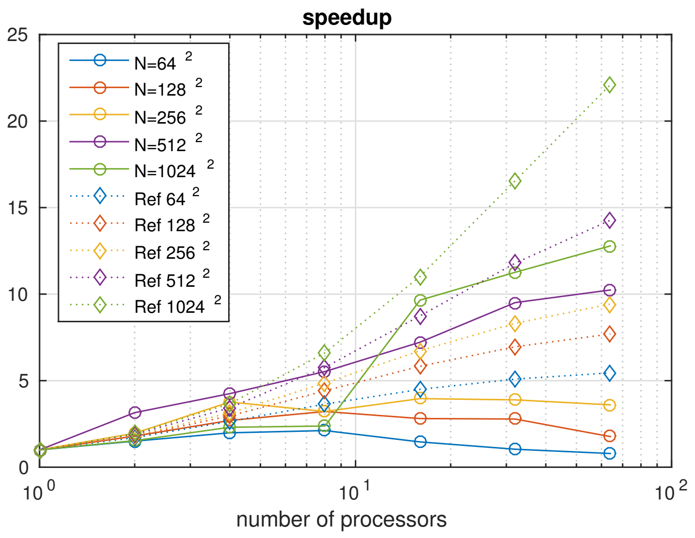


Figure 6: MPI speedup plot of strong scaling, run on Piz Daint, CSCS, Switzerland. Solid lines represent experiment results, and dashed lines are theoretical estimation. flag: CC -O3 -Wall

5 Conclusion

The efficient scheme for multiple quantum states extraction and the generalized-restart Lanczos iteration successfully solve the bounded time-independent Schrödinger equation, which can be directly embedded into the large-scale semiconductor simulation under finite element formalism. The inverse generalized-restart Lanczos algorithm further accelerate the generalized eigendecomposition significantly.

The parallel numerical simulation has been tested on 2D quantum harmonic oscillator successfully, with the memory-distributed MPI parallelism introduced for the conjugate gradient method as the parallel linear system equations solver inside the generalized Lanczos method.

Appendix

Algorithm 4: QR algorithm

```
1: procedure COMPUTE RITZ EIGENPAIRS $(\lambda_i, \mathbf{s}_i)$ OF $H_m$
2: Input: $H_m \in \mathbb{F}^{m\times m}$
3: Output: $(\lambda_i, \mathbf{s}_i)$
4: Start:
5:     Set $T_0 = H_m, U_0 = I$, $n_{iter}$ the number of iteration
6:     while $i <= n_{iter}$ do
7:         $T_{i-1} = Q_i R_i$;        ▷ QR factorization via modified Gram-Schmidt method
8:         Set $Q_i = 0 \in \mathbb{F}^{m\times m}, \mathbf{q}_1 = \mathbf{t}_1$;
9:         Set $R_i = 0 \in \mathbb{F}^m, r_{1,1} = 1$;
10:        while $k = 1, \dots m$ do
11:            $r_{k,k} = \|\mathbf{t}_k\|$;
12:            $\mathbf{q}_k = \mathbf{t}_k / r_{k,k}$;
13:            while $j = k+1, \dots, m$ do
14:                $r_{k,j} = \mathbf{q}_k^T \mathbf{t}_j$;
15:                $\mathbf{t}_j = \mathbf{t}_j - \mathbf{q}_k r_{k,j}$;
16:            end while
17:        end while
18:        $T_i = R_i Q_i$;
19:        $U_i = U_{i-1} Q_i$;
20:    end while
21:    Set $\lambda_i = t_{i,i}, \mathbf{s}_i = \mathbf{u}_i$;
22:    return $(\lambda_i, \mathbf{s}_i)$
23: end procedure
```

Algorithm 5: Conjugate Gradient algorithm

```
1: procedure SOLVE LINEAR SYSTEM EQUATIONS $Mx = b$
2: Input: $M, y$.        Output: $x$.
3: Start: With $M \in \mathbb{R}^{n\times n}$, and a normalized initial vector $x_0 \in \mathbb{R}^n$, initialize $Iter_{max} = n$.
4:     $r_0 = b - Mx_0$;
5:     $d_0 = r_0$;
6:     for $k = 0, \dots, Iter_{max} - 1$ do        ▷ CG iteration
7:         $\alpha_k = \frac{r_k^T r_k}{d_k^T M d_k}$;
8:         $x_{k+1} = x_k + \alpha_k d_k$;
9:         $r_{k+1} = r_k - \alpha_k M d_k$;
10:        $\epsilon = \|r_{k+1}\|$;
11:        if $\epsilon_k \to 0$ then accept $x_{k+1}$ as solution;
12:            Exit loop;
13:        else
14:            $\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$
15:            $d_{k+1} = r_{k+1} + \beta_k d_k$;
16:        end if
17:    end for
18: end procedure
```

References

[1] C.P. May, *Realistic Simulation of Semiconductor Nanostructure.* Sc.D. Dissertation, ETH Zurich, Zurich, 2009.

[2] Y. Saad, *Numerical Methods for Large Eigenvalue Problems, Revised Edition.* SIAM, Philadelphia, 2011.

[3] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide.* SIAM, 2000.

[4] G.W. Stewart, "A Krylov-Schur Algorithm for Large Eigenproblems." *SIAM Journal on Matrix Analysis and Applications*, Vol. 23, No. 3, pp. 601–614, 2001.

[5] K. Wu, and H. Simon, "Thick-restart Lanczos method for large symmetric eigenvalue problems." *SIAM Journal on Matrix Analysis and Applications*, Vol. 22, No. 2, pp. 602–616, 2000.

[6] K. Wu, and H. Simon, "A parallel Lanczos method for symmetric generalized eigenvalue problems." *Computing and Visualization in Science*, Vol. 2, No. 1, pp. 37–46, 1999, Springer.

[7] A. Quarteroni, R. Sacco, and F. Saleri, *Numerical Mathematics.* Springer-Verlag Berlin Heidelberg, 2007.

[8] P. Arbenz, *Solving Large Scale Eigenvalue Problems.* Lecture Notes, ETH Zurich.

[9] R. Hiptair, *Numerical Methods for Partial Differential Equations.* Lecture Notes, ETH Zurich.