CSIAM Trans. Appl. Math.
doi: 10.4208/csiam-am.2020-0021

Vol. **1**, No. 3, pp. 463-477
September 2020

A Multilevel Spectral Indicator Method for Eigenvalues of Large Non-Hermitian Matrices

Ruihao Huang[1], Jiguang Sun[1,*] and Chao Yang[2]

[1] *Michigan Technological University, Houghton, MI 49931, USA.*
[2] *Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA.*

Received 15 May 2020; Accepted 29 June 2020

Abstract. Recently a novel family of eigensolvers, called spectral indicator methods (SIMs), was proposed. Given a region on the complex plane, SIMs first compute an indicator by the spectral projection. The indicator is used to test if the region contains eigenvalue(s). Then the region containing eigenvalues(s) is subdivided and tested. The procedure is repeated until the eigenvalues are identified within a specified precision. In this paper, using Cayley transformation and Krylov subspaces, a memory efficient multilevel eigensolver is proposed. The method uses less memory compared with the early versions of SIMs and is particularly suitable to compute many eigenvalues of large sparse (non-Hermitian) matrices. Several examples are presented for demonstration.

AMS subject classifications: 65F15, 47A10, 65F10

Key words: Eigenvalue problems, spectral indicator method, non-Hermitian matrix.

1 Introduction

Consider the generalized eigenvalue problem

$$Ax = \lambda Bx, \tag{1.1}$$

where A, B are $n \times n$ large sparse non-Hermitian matrices. In particular, we are interested in the computation of all eigenvalues in a region $R \subset \mathbb{C}$, which contains p eigenvalues such that $1 \ll p \ll n$ or $1 \ll p \sim n$.

Many efficient eigensolvers are proposed in literature for large sparse Hermitian (or symmetric) matrices (see, e.g., [11]). In contrast, for non-Hermitian matrices, there exist

*Corresponding author. *Email addresses:* ruihaoh@mtu.edu (R. Huang), jiguangs@mtu.edu (J. Sun), cyang@lbl.gov (C. Yang)

much fewer methods including the Arnoldi method and Jacobi-Davidson method [2, 9]. Unfortunately, these methods are still far from satisfactory as pointed out in [12]:

"*In essence what differentiates the Hermitian from the non-Hermitian eigenvalue problem is that in the first case we can always manage to compute an approximation whereas there are non-symmetric problems that can be arbitrarily difficult to solve and can essentially make any algorithm fail*."

Recently, a family of eigensolvers, called the spectral indicator methods (SIMs), was proposed [6, 7, 14]. The idea of SIMs is different from the classical eigensolvers. In brief, given a region $R \subset \mathbb{C}$ whose boundary ∂R is a simple closed curve, an indicator I_R is defined and then used to decide if R contains eigenvalue(s). When the answer is positive, R is divided into sub-regions and indicators for these sub-regions are computed. The procedure continues until the size of the sub-region(s) is smaller than the specified precision, e.g., 10^{-6}. The indicator I_R is defined using the spectral projection P, i.e., Cauchy contour integral of the resolvent of the matrix pencil (A, B) on ∂R [8]. In particular, one can construct I_R based on the spectral projection of a random vector $\boldsymbol{f}$. It is well-known that P projects $\boldsymbol{f}$ to the generalized eigenspace associated to the eigenvalues enclosed by ∂R [8]. $P\boldsymbol{f}$ is zero if there is no eigenvalue(s) inside R, and nonzero otherwise. Hence $P\boldsymbol{f}$ can be used to decide if R contains eigenvalues(s) or not. Evaluation of $P\boldsymbol{f}$ needs to solve linear systems at quadrature points on ∂R. In general, it is believed that computing eigenvalues is more difficult than solving linear systems of equations [5]. The proposed method converts the eigenvalue problem to solving a number of related linear systems.

Spectral projection is a classical tool in functional analysis to study, e.g., the spectrum of operators [8] and the finite element convergence theory for eigenvalue problems of partial differential equations [14]. It has been used to compute matrix eigenvalue problems in the method by Sakurai-Sugiura [13] and FEAST by Polizzi [10]. For example, FEAST uses spectral projection to build subspaces and thus can be viewed as a subspace method [15]. In contrast, SIMs use the spectral projection to define indicators and combines the idea of bisection to locate eigenvalues. Note that the use of other tools such as the condition number to define the indicator is possible.

In this paper, we propose a new SIM, called SIM-M. Firstly, by proposing a new indicator, the memory requirement is significantly reduced and thus the computation of many eigenvalues of large matrices becomes realistic. Secondly, a new strategy to speedup the computation of the indicators is presented. Thirdly, other than the recursive calls in the first two members of SIMs [6, 7], a multilevel technique is used to further improve the efficiency. Moreover, a subroutine is added to find the multiplicities of the eigenvalues. The rest of the paper is organized as follows. Section 2 presents the basic idea of SIMs and two early members of SIMs. In Section 3, we propose a new eigensolver SIM-M with the above features. The algorithm and the implementation details are discussed as well. The proposed method is tested by various matrices in Section 4. Finally, in Section 5, we draw some conclusions and discuss some future work.

2 Spectral indicator methods

In this section, we give an introduction of SIMs and refer the readers to [6,7,14] for more details. For simplicity, assume that R is a square and $\Gamma:=\partial R$ lies in the resolvent set $\mathcal{R}$ of (A,B), i.e., the set of $z\in\mathbb{C}$ such that $(A-zB)$ is invertible. The key idea of SIMs is to find an indicator that can be used to decide if R contains eigenvalue(s).

One way to define the indicator is to use the spectral projection, a classical tool in functional analysis [8]. Specifically, the matrix P defined by

$$P=\frac{1}{2\pi i}\int_{\Gamma}(A-zB)^{-1}dz \tag{2.1}$$

is the spectral projection of a vector $\boldsymbol{f}$ onto the generalized eigenspace associated with the eigenvalues of (1.1) inside Γ. If there are no eigenvalues inside Γ, then $P=0$, and hence $P\boldsymbol{f}=\mathbf{0}$ for all $\boldsymbol{f}\in\mathbb{C}^n$. If Γ does enclose one or more eigenvalues, then $P\boldsymbol{f}\neq\mathbf{0}$ with probability 1 for a random vector $\boldsymbol{f}$.

To improve robustness, in RIM (recursive integral method) [6], the first member of SIMs, the indicator is defined as

$$I_R:=\left\|P\left(\frac{P\boldsymbol{f}}{\|P\boldsymbol{f}\|}\right)\right\|. \tag{2.2}$$

Analytically, I_R=1 if there exists at least one eigenvalue in Γ. Note that when a quadrature rule is applied, $I_R\neq 1$ in general. The RIM algorithm is very simple and listed as follows [6].

RIM$(A,B,R,h_0,\delta_0,\boldsymbol{f})$

Input: matrices A,B, region R, precision h_0, threshold δ_0, random vector $\boldsymbol{f}$.

Output: generalized eigenvalue(s) λ inside R

1. Compute I_R.
2. If $I_R<\delta_0$, exit (no eigenvalues in R).
3. Otherwise, compute the diameter h of R.

   - If $h>h_0$, partition R into subregions $R_j, j=1,\cdots N$.
     for $j=1$ to N
     RIM$(A,B,R_j,h_0,\delta_0,\boldsymbol{f})$.
     end
   - else,
     set λ to be the center of R. output λ and exit.

The major task of RIM is to compute the indicator I_R defined in (2.2). Let the approximation to $P\boldsymbol{f}$ be given by

$$P\boldsymbol{f} \approx \frac{1}{2\pi i}\sum_{j=1}^{n_0}\omega_j \boldsymbol{x}_j, \tag{2.3}$$

where ω_j's are quadrature weights and $\boldsymbol{x}_j$'s are the solutions of the linear systems

$$(A-z_jB)\boldsymbol{x}_j = \boldsymbol{f}, \quad j=1,\cdots,n_0. \tag{2.4}$$

Here $z_j's$ are quadrature points on Γ. The total number of the linear systems (2.4) for RIM to solve is at most

$$2n_0\lceil \log_2(h/h_0)\rceil p, \tag{2.5}$$

where p is the number of eigenvalues in R, n_0 is the number of the quadrature points, h is the size of the R, h_0 is the required precision, and $\lceil\cdot\rceil$ denotes the least larger integer. Given R, p is a fixed number. The complexity of RIM is proportional to the complexity of solving the linear system (2.4).

The computational cost of RIM mainly comes from solving the linear systems (2.4) to approximate the spectral projection $P\boldsymbol{f}$. It is clear that the cost will be greatly reduced if one can take advantage of the parametrized linear systems of the same structure. In [7], a new member RIM-C (recursive integral method using Cayley transformation) is proposed. The idea is to construct some Krylov subspaces and use them to solve (2.4) for all quadrature points z_j's. Since the method we shall propose is based on RIM-C, a description of RIM-C is included as follows.

Let M be an $n\times n$ matrix, $\boldsymbol{b}\in\mathbb{C}^n$ be a vector, and m be a non-negative integer. The Krylov subspace is defined as

$$K_m(M;\boldsymbol{b}) := \text{span}\{\boldsymbol{b}, M\boldsymbol{b}, \cdots, M^{m-1}\boldsymbol{b}\}. \tag{2.6}$$

It has the shift-invariant property

$$K_m(\gamma_1 M+\gamma_2 I;\boldsymbol{b}) = K_m(M;\boldsymbol{b}), \tag{2.7}$$

where γ_1 and γ_2 are two scalars.

Consider a family of linear systems

$$(A-zB)\boldsymbol{x} = \boldsymbol{f}, \tag{2.8}$$

where z is a complex number. Assume that σ is not a generalized eigenvalue and $\sigma\neq z$. By Cayley transformation, multiplying both sides of (2.8) by $(A-\sigma B)^{-1}$, we have that

$$\begin{aligned}(A-\sigma B)^{-1}\boldsymbol{f} &= (A-\sigma B)^{-1}(A-zB)\boldsymbol{x}\\ &= (A-\sigma B)^{-1}(A-\sigma B+(\sigma-z)B)\boldsymbol{x}\\ &= (I+(\sigma-z)(A-\sigma B)^{-1}B)\boldsymbol{x}.\end{aligned}$$

Let $M=(A-\sigma B)^{-1}B$ and $\boldsymbol{b}=(A-\sigma B)^{-1}\boldsymbol{f}$. Then (2.8) becomes

$$(I+(\sigma-z)M)\boldsymbol{x}=\boldsymbol{b}. \tag{2.9}$$

From (2.7), the Krylov subspace $K_m(I+(\sigma-z)M;\boldsymbol{b})$ is the same as $K_m(M;\boldsymbol{b})$. We shall use $K_m^{\sigma}(M;\boldsymbol{b})$ when it is necessary to indicate its dependence on the shift σ.

Arnoldi's method is used by RIM-C to solve the linear systems. First, consider the orthogonal projection method for

$$M\boldsymbol{x}=\boldsymbol{b}.$$

Let the initial guess be $\boldsymbol{x}_0=\boldsymbol{0}$. One seeks an approximate solution $\boldsymbol{x}_m$ in $K_m(M;\boldsymbol{b})$ by imposing the Galerkin condition [11]

$$(\boldsymbol{b}-M\boldsymbol{x}_m)\perp K_m(M;\boldsymbol{b}). \tag{2.10}$$

The Arnoldi's method (Algorithm 6.1 of [12]) is as follows.

1. Choose a vector $\boldsymbol{v}_1$ of norm 1 ($\boldsymbol{v}_1=\boldsymbol{b}/\|\boldsymbol{b}\|_2$).
2. for $j=1,2,\cdots,m$
   - $h_{ij}=(M\boldsymbol{v}_j,\boldsymbol{v}_i)$, $i=1,2,\cdots,j$.
   - $\boldsymbol{w}_j=M\boldsymbol{v}_j-\sum_{i=1}^{j}h_{ij}\boldsymbol{v}_i$.
   - $h_{j+1,j}=\|\boldsymbol{w}_j\|_2$. If $h_{j+1,j}=0$, stop.
   - $\boldsymbol{v}_{j+1}=\boldsymbol{w}_j/h_{j+1,j}$.

Let V_m be the $n\times m$ orthogonal matrix with column vectors $\boldsymbol{v}_1,\cdots,\boldsymbol{v}_m$ and H_m be the $m\times m$ Hessenberg matrix whose nonzero entries are $h_{i,j}$. Proposition 6.5 of [12] implies that

$$MV_m=V_mH_m+\boldsymbol{v}_{m+1}h_{m+1,m}\boldsymbol{e}_m^T \tag{2.11}$$

and

$$\text{span}\{\text{col}(V_m)\}=K_m(M;\boldsymbol{b}).$$

Let $\boldsymbol{x}_m=V_m\tilde{\boldsymbol{y}}$ such that the Galerkin condition (2.10) holds, i.e.,

$$V_m^T\boldsymbol{b}-V_m^TMV_m\tilde{\boldsymbol{y}}=\boldsymbol{0}. \tag{2.12}$$

Using (2.11), the residual is given by

$$\|\boldsymbol{b}-M\boldsymbol{x}_m\|_2=h_{m+1,m}|\boldsymbol{e}_m^T\tilde{\boldsymbol{y}}|. \tag{2.13}$$

Next, we consider the linear system (2.9). For $I+(\sigma-z)M$, due to the shift invariant property, one has that

$$\{I+(\sigma-z)M\}V_m=V_m(I+(\sigma-z)H_m)+(\sigma-z)\boldsymbol{v}_{m+1}h_{m+1,m}\boldsymbol{e}_m^T. \tag{2.14}$$

The Galerkin condition (2.10) becomes

$$V_m^T \boldsymbol{b} - V_m^T \{I + (\sigma - z)M\} V_m \boldsymbol{y} = 0. \tag{2.15}$$

It implies that

$$\{I + (\sigma - z)H_m\}\boldsymbol{y} = \beta \boldsymbol{e}_1, \tag{2.16}$$

where $\beta = \|\boldsymbol{b}\|_2$. Combination of (2.14) and (2.16) gives the residual

$$\|b - \{I + (\sigma - z)M\}\boldsymbol{x}_m\|_2 = (\sigma - z)h_{m+1,m}|\boldsymbol{e}_m^T \boldsymbol{y}|. \tag{2.17}$$

Let z_j be a quadrature point and one need to solve

$$(I + (\sigma - z_j)M)\boldsymbol{x}_j = \boldsymbol{b}, \tag{2.18}$$

where $M = (A - \sigma B)^{-1}B$ and $\boldsymbol{b} = (A - \sigma B)^{-1}\boldsymbol{f}$.

From (2.3) and (2.16),

$$\boldsymbol{y}_j = \beta(I + (\sigma - z_j)H_m)^{-1}\boldsymbol{e}_1, \tag{2.19}$$

$$\boldsymbol{x}_j \approx V_m \boldsymbol{y}_j,$$

$$P\boldsymbol{f} \approx \frac{1}{2\pi i}\sum w_j V_m \boldsymbol{y}_j. \tag{2.20}$$

The idea of RIM-C is to use the Krylov subspace for $M = (A - \sigma B)^{-1}B$ to solve (2.4) for as many z_j's as possible. The residual can be monitored with a little extra cost using (2.17).

Since the Krylov subspace method is used, the indicator defined in (2.2) is not appropriate since it projects $\boldsymbol{f}$ twice. RIM-C defines an indicator different from (2.2). Let $P\boldsymbol{f}|_{n_0}$ be the approximation of $P\boldsymbol{f}$ with n_0 quadrature points for the circle circumscribing R. It is well-known that the trapezoidal quadrature of a periodic function converges exponentially [3, Section 4.6.5], i.e.,

$$\|P\boldsymbol{f} - P\boldsymbol{f}|_{n_0}\| = \mathcal{O}(e^{-Cn_0}),$$

where C is a constant. For a large enough n_0, one has that

$$\frac{\|P\boldsymbol{f}|_{2n_0}\|}{\|P\boldsymbol{f}|_{n_0}\|} = \begin{cases} \dfrac{\|P\boldsymbol{f}\| + \mathcal{O}(e^{-C2n_0})}{\|P\boldsymbol{f}\| + \mathcal{O}(e^{-Cn_0})}, & \text{if there are eigenvalues inside } R, \\ \dfrac{\mathcal{O}(e^{-C2n_0})}{\mathcal{O}(e^{-Cn_0})} = \mathcal{O}(e^{-Cn_0}), & \text{no eigenvalue inside } R. \end{cases} \tag{2.21}$$

The indicator is then defined as

$$I_R := \frac{\|P\boldsymbol{f}_{2n_0}\|}{\|P\boldsymbol{f}_{n_0}\|} \approx \frac{\left\|\sum_{j=1}^{2n_0} w_j V_m \boldsymbol{y}_j\right\|}{\left\|\sum_{j=1}^{n_0} w_j V_m \boldsymbol{y}_j\right\|}. \tag{2.22}$$

3 Multilevel memory efficient method

In this section, we make several improvements of RIM-C and propose a multilevel memory efficient method, called SIM-M.

3.1 A new memory efficient indicator

In view of (2.22), the computation of the indicator needs to store V_m. When R contains a lot of eigenvalues, the method can become memory intensive.

Definition 3.1. *A (square) region R is resolvable with respect to (σ,ϵ_0) if the linear systems* (2.4) *associated with all the quadrature points can be solved up to the given residual ϵ_0 using the Krylov subspace related to a shift σ.*

Assume that R is resolvable with respect to (σ,ϵ_0). From (2.22), one has that

$$I_R \approx \frac{\|\sum_{j=1}^{2n_0} w_j V_m \boldsymbol{y}_j\|}{\|\sum_{j=1}^{n_0} w_j V_m \boldsymbol{y}_j\|} = \frac{\|V_m \sum_{j=1}^{2n_0} w_j \boldsymbol{y}_j\|}{\|V_m \sum_{j=1}^{n_0} w_j \boldsymbol{y}_j\|}. \tag{3.1}$$

Note that

$$\left\| V_m \sum_{j=1}^{n_0} w_j \boldsymbol{y}_j \right\|^2 = \left(\sum_{j=1}^{n_0} w_j \boldsymbol{y}_j \right)^T V_m^T V_m \sum_{j=1}^{n_0} w_j \boldsymbol{y}_j = \left\| \sum_{j=1}^{n_0} w_j \boldsymbol{y}_j \right\|^2 \tag{3.2}$$

since $V_m^T V_m$ is the identity matrix. Dropping V_m in (2.20), we define a new indicator

$$\tilde{I}_R = \frac{\|\sum_{j=1}^{2n_0} w_j \boldsymbol{y}_j\|}{\|\sum_{j=1}^{n_0} w_j \boldsymbol{y}_j\|}. \tag{3.3}$$

As a consequence, there is no need to store V_m's ($n \times m$ matrices) but to store much smaller $m \times m$ ($m = \mathcal{O}(1)$) matrices H_m's.

As before, we use a threshold to decide whether or not eigenvalues exist in R. From (2.21), if there are no eigenvalues in R, the indicator $I_R = \mathcal{O}(e^{-Cn_0})$. In the experiments, we take $n_0 = 4$. Assume that $C = 1$, we would have that $I_R \approx 0.018$. It is reasonable to take $\delta_0 = 1/20$ as the threshold. The choice is ad-hoc. Nonetheless, the numerical examples show that the choice is robust.

Definition 3.2. *A (square) region R is admissible if $I_R > \delta_0$.*

Remark 3.1. In practice, a region which is smaller than h_0 and not resolvable with respect to (σ,ϵ_0) is taken to be admissible.

3.2 Speedup the computation of indicators

To check if a linear system (2.4) can be solved effectively using a Krylov space $K_m^\sigma(M;\boldsymbol{b})$, one need to compute the residual (2.17) for many z_j's. In the following, we propose a fast method. First rewrite (2.16) as

$$\left(\frac{1}{\sigma-z_j}I+H_m\right)\boldsymbol{y}_j=\frac{\beta}{\sigma-z_j}\boldsymbol{e}_1. \tag{3.4}$$

Assume that H_m has the following eigen-decomposition $H_m=PDP^{-1}$ where

$$D=\text{diag}\{\lambda_1,\lambda_1,\cdots,\lambda_m\}.$$

Then (3.4) can be written as

$$P\left(\frac{1}{\sigma-z_j}I+D\right)P^{-1}\boldsymbol{y}_j=\frac{\beta}{\sigma-z_j}\boldsymbol{e}_1,$$

whose solution is simply

$$\begin{aligned}\boldsymbol{y}_j&=P\left(\frac{1}{\sigma-z_j}I+D\right)^{-1}P^{-1}\frac{\beta}{\sigma-z_j}\boldsymbol{e}_1\\&=P\big(I+(\sigma-z_j)D\big)^{-1}P^{-1}\boldsymbol{e}_1.\end{aligned}$$

Hence

$$\begin{aligned}\boldsymbol{e}_m^T\boldsymbol{y}_j&=\boldsymbol{e}_m^TP\big(I+(\sigma-z_j)D\big)^{-1}P^{-1}\boldsymbol{e}_1\\&=\boldsymbol{r}_m\Lambda\boldsymbol{c}_1,\end{aligned} \tag{3.5}$$

where $\boldsymbol{r}_m$ is the last row of P, $\boldsymbol{c}_1$ is the first column of P^{-1}, and

$$\Lambda=\text{diag}\left\{\frac{1}{1+(\sigma-z_j)\lambda_1},\frac{1}{1+(\sigma-z_j)\lambda_2},\cdots,\frac{1}{1+(\sigma-z_j)\lambda_m}\right\}.$$

In fact, this further reduces the memory requirement since only three $m\times1$ vectors, $\boldsymbol{r}_m$, $\boldsymbol{c}_1$, and Λ are stored for each shift σ.

3.3 Multilevel technique

Now we propose a multilevel technique, which is more efficient and suitable for parallelization. In SIM-M, the following strategy is employed.

At level 1, R is divided uniformly into smaller squares R_j^1, $j=1,\cdots,N^1$. Collect all quadrature points z_j^1's and solve the linear systems (2.4) accordingly. The indicators of R_j^1's are computed and squares containing eigenvalues are chosen. Indicators of the

resolvable squares are computed. Squares containing eigenvalues are subdivided into smaller square. Squares that are not resolvable are also subdivided into smaller squares. These squares are left to the next level. At level 2, the same operation is carried out. The process stops at level K when the size of the squares is smaller than the given precision h_0.

3.4 Multiplicities of eigenvalues

The first two members of SIMs only output the eigenvalues. A function to find the multiplicities of the eigenvalues can be integrated into SIM-M.

Definition 3.3. *An eigenvalue λ is said to be resolved by a shift σ if the small square at level K containing λ is resolvable using the Krylov subspace K_m^σ.*

When the eigenvalues are computed, a mapping from the set of eigenvalues Λ to the set of shifts Σ is also established. Hence, for a shift σ, one can find the set of all eigenvalues that are resolved by σ, denoted by

$$\Lambda_\sigma = \{\lambda_1, \cdots, \lambda_n\}.$$

For k random vectors $\boldsymbol{f}_1, \cdots, \boldsymbol{f}_k$, generate k Krylov subspaces $K_m^\sigma(M, \boldsymbol{b}_i)$, $i = 1, \cdots, k$. For each $\lambda \in \Lambda_\sigma$, compute the spectral projections of $\boldsymbol{f}_1, \cdots, \boldsymbol{f}_k$ using the above Krylov subspaces. Then the number of significant singular values of the matrix $[P\boldsymbol{f}_1, \cdots, P\boldsymbol{f}_k]$ is the multiplicity of λ.

Remark 3.2. In fact, the associated eigenvectors can be obtained with little extra cost by adding more quadrature points. However, it can be expected that it needs a lot of more time and memory to find the multiplicities since more Krylov subspaces are generated.

3.5 Algorithm for SIM-M

Now we are ready to present the new algorithm SIM-M.

SIM-M$(A, B, R, \boldsymbol{f}, h_0, \epsilon, \delta_0, m, n_0)$

Input:

- A, B: $n \times n$ matrices
- R: search region in $\mathbb{C}$
- $\boldsymbol{f}$: a random vector
- h_0: precision
- ϵ: residual tolerance
- δ_0: indicator threshold

- m: size of Krylov subspace
- n_0: number of quadrature points

Output:

- generalized eigenvalues λ's inside R

1. use the center of R as the first shift and generate the associated Krylov subspaces.
2. pre-divide R into small squares of size h_0: $R_j, j=1,\cdots,J$ (these are selected squares at the initial level).
3. for $j=1:J$ do
   - For all quadrature points for R_j, check if the related linear systems can be solved using any one of the existing Krylov subspaces up to the given residual ϵ_0. If yes, associate R_j with that Krylov subspace. Otherwise, set the shift to be the center of R_j and construct a Krylov subspace.
4. calculate the number of the levels, denoted by K, needed to reach the precision h_0.
5. for $k=1:K$
   - for each selected square R_j^k at level k, check if R_j^k is resolvable.
     * if R_j^k is resolvable, compute the indicator for R_j^k and mark it when the indicator is larger than δ_0, i.e., R_j^k contains eigenvalues.
     * if R_j^k is not solvable, mark R_j^k and leave it to next level.
   - divide each marked square into four squares uniformly and move to next level.
6. post-processing the marked squares at level K, merge eigenvalues when necessary, show warnings if there exist unsolvable squares.
7. output eigenvalues.

In the implementation, we choose $m=50$. Similar values such as $m=30$ do not change the performance significantly. The indicator threshold is set to be $\delta_0=1/20$ as discussed in Section 3.1. The number of quadrature points is $n_0=8$, which is effective for the examples. The choices of these parameters affect the efficiency and robustness of the algorithm in a subtle way and deserve more study for different problems.

4 Numerical examples

We show some examples for SIM-M. All the test matrices are from the University of Florida Sparse Matrix Collection [4] except the last example. The computations are done using MATLAB R2017a on a MacBook Pro with 16 GB memory and a 3-GHz Intel Core i7 CPU.

4.1 Directed weighted graphs

The first group contains four non-symmetric matrices, HB/gre_115, HB/gre_343, HB/gre_512, HB/gre_1107. These matrices represent directed weighted graphs.

Table 1: Time (in second) used for all eigenvalues by SIM-M.

N (size of the matrix)	115	343	512	1107
T (time in seconds)	3.4141s	10.2917s	14.7461s	40.2252s
T/N	0.0297	0.0300	0.0288	0.0363

We compute all eigenvalues using SIM-M in Table 1. The first row represents sizes of the four matrices. The second row shows the CPU times (in seconds) used by SIM-M. The numbers in the third row are the ratios of the seconds used by SIM-M and the sizes of the matrices, i.e., the average time to compute one eigenvalue. It seems that the ratio is stable for matrices of different sizes. In Fig. 1, we show the eigenvalues computed by SIM-M and by Matlab *eig*, which coincide each other.

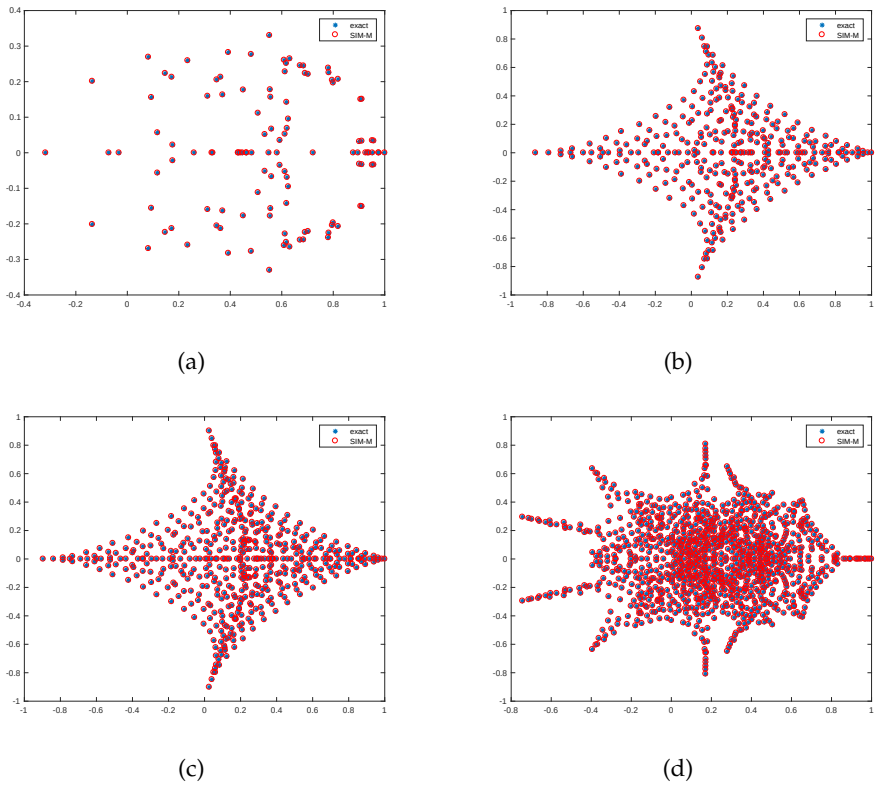
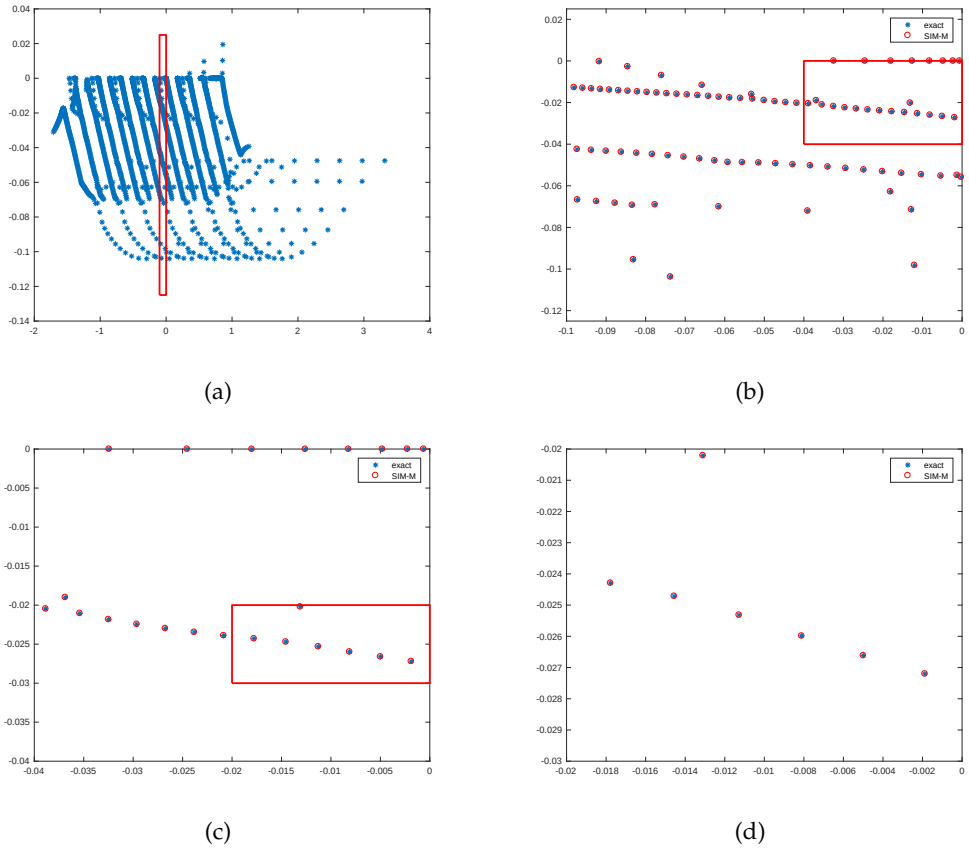


Figure 1: Eigenvalues computed by SIM-M and Matlab *eig* coincide. (a) HB/gre_115. (b) HB/gre_343. (c): HB/gre_512. (d): HB/gre_1107.

4.2 A quantum chemistry problem

The second example, Bai/qc2534, is a sparse 2534×2534 matrix from modeling H2+ in an electromagnetic field. The full spectrum, computed by Matlab *eig*, is shown in Fig. 2(a), in which the red rectangle is $R_1=[-0.1,0]\times[-0.125,0.025]$. In Fig. 2(b), the eigenvalues are computed by SIM-M in R_1, which coincide with those computed by Matlab *eig*. The red rectangle is Fig. 2(b) is $R_2=[-0.04,0]\times[-0.04,0]$. Eigenvalues in R_2 computed by SIM-M are shown in Fig. 2(c). The rectangle in Fig. 2(c) is $R_3=[-0.02,0]\times[-0.03,-0.02]$. Eigenvalues in R_3 computed by SIM-M are shown in Fig. 2(d).



(a) (b)

(c) (d)

Figure 2: QC2534. (a): Full spectrum by Matlab *eig* (the rectangle is R_1). (b): Eigenvalues by SIM-M in R_1 (the rectangle is R_2). (c): Eigenvalues by SIM-M in R_2 (the rectangle is R_3). (d): Eigenvalues by SIM-M in R_3.

The second row of Table 2 shows that there are 88, 23 and 7 eigenvalues in R_1,R_2 and R_3, respectively. The third row shows the time used by SIM-M to compute all eigenvalues in R_1,R_2 and R_3. The fourth row shows the average time to compute one eigenvalue, which seems to be consistent.

Table 2: Time (in second) used by SIM-M for different regions.

	R_1	R_2	R_3
N (# of eigenvalues)	88	23	7
T (time in seconds)	14.7445s	3.7005s	0.54645s
T/N	0.1676	0.1609	0.0781

4.3 DNA electrophoresis

The third example is a $39{,}082\times 39{,}082$ matrix, vanHeukelum/cage11, arising from DNA electrophoresis. We consider a series of nested domains

$$\begin{aligned} R_1 &= [0.230,0.270]\times[-0.0005,0.0005],\\ R_2 &= [0.250,0.270]\times[-0.0005,0.0005],\\ R_3 &= [0.250,0.260]\times[-0.0005,0.0005],\\ R_4 &= [0.254,0.256]\times[-0.0005,0.0005]. \end{aligned}$$

In Table 3, the time and number of eigenvalues found in each domain are shown. Again, the average time to compute one eigenvalue is stable.

Table 3: Time (in second) used by SIM-M for different regions.

	R_1	R_2	R_3	R_4
N (# of eigenvalues)	105	31	31	8
T (time in seconds)	588.3552s	299.4242s	214.0637s	47.8098s
T/N	5.6034	9.6588	6.9053	5.9762

Remark 4.1. Note that it is not possible to use Matlab *eig* to find all eigenvalues due the memory constraint. However, SIM-M does not have this limitation. In fact, numerical results in the above two subsections indicate that a parallel version of SIM-M has the potential to be faster than the classical methods.

4.4 Quantum states in disordered media

The test matrices are sparse and symmetric arising from localized quantum states in random or disordered media [1]. We would like to use this example to show that the method can treat rather large problems on a laptop. The matrices A and B are of $1{,}966{,}080\times 1{,}966{,}080$. We consider three nested domains given by

$$\begin{aligned} R_1 &= [0.00,0.60]\times[-0.05,0.05],\\ R_2 &= [0.00,0.50]\times[-0.05,0.05],\\ R_3 &= [0.00,0.40]\times[-0.05,0.05]. \end{aligned}$$

In Table 4, time and number of eigenvalues in each domain are shown. Again, we observe that the average time to compute one eigenvalue is stable.

Table 4: Time (in second) used by SIM-M for different regions.

	R_1	R_2	R_3
N (# of eigenvalues)	36	7	3
T (time in seconds)	573.1088s	112.1876s	58.9957s
T/N	15.9197	16.0268	19.6652

5 Conclusions and future work

Given a region on the complex plane, SIMs first compute an indicator, which is used to test if the region contains eigenvalues. Then the region is subdivided and tested until all the eigenvalues are isolated with a specified precision. Hence SIMs can be viewed as a bisection technique.

We propose an improved version SIM-M to compute many eigenvalues of large matrices. Several examples are presented for demonstrations. However, to make the method practically competitive, a parallel implementation on super computers is necessary. Currently, SIMs use the spectral projection to compute the indicators. Other ways to define the indicators should be investigated in future.

Acknowledgments

The work of Jiguang Sun was partially supported by a REF-SCG, Michigan Technological University. The work of Chao Yang was supported by the Scientific Discovery through Advanced Computing (SciDAC) program funded by U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research under Contract No. DE-AC02-05CH11231.

References

[1] D.N. Arnold, G. David, D. Jerison, S. Mayboroda and M. Filoche, *Effective confining potential of quantum states in disordered media.* Phys. Rev. Lett. 116(5), 056602, 2016.

[2] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst (editors), Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide, Society for Industrial and Applied Mathematics, Philadelphia, 2000.

[3] P.J. Davis and P. Rabinowitz, Methods of Numerical Integration, 2nd Ed., Academic Press, Inc., Orlando, FL, 1984.

[4] T.A. Davis and Y. Hu, *The University of Florida sparse matrix collection*, ACM Transaction on Mathematical Software, Vol. 38(2011), Iss. 1, Article No. 1.

[5] V. Hernandez, J.E. Roman, and V. Vidal, *SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems*, ACM Transactions on Mathematical Software (TOMS), Vol. 31(2005), Iss. 3, 351–362.
[6] R. Huang, A. Struthers, J. Sun and R. Zhang, *Recursive integral method for transmission eigenvalues.* J. Comput. Phys. 327, 830–840, 2016.
[7] R. Huang, J. Sun and C. Yang, *Recursive Integral Method with Cayley Transformation*, Numer. Linear Algebra Appl. 25(6), e2199, 2018.
[8] T. Kato, Perturbation Theory of Linear Operators, Classics in Mathematics, Springer-Verlag, Berlin, 1995.
[9] R.B. Lehoucq, D.C. Sorensen and C. Yang, ARPACK User's Guide – Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1998.
[10] E. Polizzi, *Density-matrix-based algorithms for solving eigenvalue problems.* Phys. Rev. B., 79, 115112, 2009.
[11] Y. Saad, Iterative Methods for Sparse Linear Systems, 2nd Ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, 2003.
[12] Y. Saad, Numerical Methods for Large Eigenvalue Problems, 2nd Ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, 2011.
[13] T. Sakurai and H. Sugiura, *A projection method for generalized eigenvalue problems using numerical integration.* J. Comput. Appl. Math, 159(1), 119–128, 2003.
[14] J. Sun and A. Zhou, Finite Element Methods for Eigenvalue Problems, Chapman and Hall/CRC, Boca Raton, FL, 2016.
[15] P. Tang and E. Polizzi, *FEAST as a subspace iteration eigensolver accelerated by approximate spectral projection.* SIAM J. Matrix Anal. Appl. 35 (2014), no. 2, 354–390.