A Contour-Integral Based Method for Counting the Eigenvalues Inside a Region

Guojian Yin

Received: 4 December 2017 / Revised: 12 July 2018 / Accepted: 20 September 2018 / Published online: 26 September 2018
© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract
In many applications, the information about the number of eigenvalues inside a given region is required. In this work, we develop a contour-integral based method for this purpose. Our method is motivated by two findings. There exist methods for estimating the number of eigenvalues inside a region in the complex plane, but our method is able to compute the number exactly. Our method has a good potential to be implemented on a high-performance parallel architecture. Numerical experiments are reported to show the viability of our method.

Keywords
Eigenvalue · Generalized eigenvalue problem · Contour integral · Spectral projection

Mathematics Subject Classification 15A18 · 58C40 · 65F15

1 Introduction
Consider the generalized eigenvalue problem
\[ Ax = \lambda Bx, \]
where \( A, B \in \mathbb{C}^{n \times n} \). The scalar \( \lambda \in \mathbb{C} \) and the associated nonzero vector \( x \in \mathbb{C}^n \) are called eigenvalue and eigenvector, respectively. Let \( \mathcal{D} \) be a disk in the complex plane enclosed by a circle \( \Gamma \). In this paper, we develop an approach for computing the number of eigenvalues of (1) inside \( \mathcal{D} \) exactly. Due to the Möbius transformation, the resulting approach can be adapted to the union or intersections of arbitrary half plane and (complemented) disks, and so a rather general region [3].

In many applications, it is required to know the number of eigenvalues, rather than the eigenvalues themselves, inside a prescribed region in the complex plane [16,17]. For example, choosing the number of clusters, say \( g \), is a general problem for all clustering algorithms; for the spectral clustering based on eigengap heuristic which is one of most popular clustering algorithms, to get the number \( g \) it needs to know the multiplicities of zero eigenvalues or the
number of zero or close to zero eigenvalues of a graph Laplacian matrix [28]. In the context of control theory, it needs to know whether there exists eigenvalue of the state (system) matrix inside the right half-plane [14,25]. If it is, then the system is unstable. The number of eigenvalues inside the target region is a prerequisite of the eigensolvers based on divide-and-conquer techniques [2,9]. To get the number of the eigenvalues inside a given region, the most straightforward way is to compute all eigenvalues inside the target region by some existing eigensolver, such as the rational Krylov subspace method [20]. However, this is always time-consuming and not effective, because we are only interested in the number of eigenvalues rather than the eigenvalues themselves.

When $A$ and $B$ are Hermitian matrices with $B$ being positive definite, i.e., $z B - A$ is a definite matrix pencil [2], it is well-known that the eigenvalues of (1) are real-valued [21,26]. Assume that we want to know the number of eigenvalues inside an interval, say $[a, b]$, the standard method works as follows. Compute two LDL decompositions: $A - aB = L_a D_a L_a^*$ and $A - bB = L_b D_b L_b^*$. Then, the difference between the numbers of negative entries on the diagonals of $D_a$ and $D_b$ is exactly the number of eigenvalues inside the interval $[a, b]$. The derivation of this method is based on the Sylvester law of inertia, see [9,13,21] for more details. Obviously, the efficiency of this method depends on the accurate computation of two related LDL decompositions [9]. Computing each of the LDL decomposition requires floating point operations of order $O(n^3)$ [12]; as a result, it becomes computationally prohibitive for large-scale problems. When (1) comes to the non-Hermitian case, to the best of our knowledge, there is no effective method for exactly computing the number of eigenvalues inside a given disk.

The contour-integral based eigensolvers [19,23,24,29] are recent efforts for computing the eigenvalues inside a prescribed region. The information about the number of eigenvalues inside the region of interest is crucial to their practical implementations of this kind of eigensolvers. Methods based on contour integral were proposed to get estimations of this number, where they make use of the stochastic estimation of the trace of the spectral operator associated with the eigenvalues inside a given region [9,10,29]. However, these estimation approaches may be unreliable in some cases, for instance, when the matrices $A$ and $B$ are ill-conditioned [29]. In [27], another type of contour-integral based estimation method was presented under the assumption that matrices $A$ and $B$ are Hermitian and $B$ is positive definite. It should be pointed out that all these existing contour-integral based estimation methods can only provide approximations for the number of eigenvalues inside the given region.

In this paper, we present a contour-integral based method for exactly computing the number of eigenvalues of (1) inside a disk. The derivation of the proposed method requires that the eigenvalues of (1) are semi-simple, namely, there are $n$ independent eigenvectors, which is almost always the case in practice. Our method is motivated by two findings. The first comes from using the Gauss–Legendre quadrature rule to approximately compute the spectral projector constructed by a particular contour integral, more details will be discussed in Sect. 2. The second one is devoted to avoiding the computation of the Weierstrass canonical form of matrix pencil $z B - A$ when using the first finding to count the eigenvalues inside the target disk, we will detail the second finding in Sect. 4. Since our method is also based on contour integral, it keeps the promising features of the usual contour-integral based eigensolvers, such as having a good potential to be implemented on a high-performance parallel architecture. Moreover, it can integrate with the contour-integral based eigensolvers [19,23,24,29] to help them determine whether all desired eigenvalues are found when the algorithms stop.

The paper is organized as follows. In Sect. 2, we present the first finding, which is derived from using the Gauss–Legendre quadrature rule to approximately compute the spectral projector constructed by a contour integral. Since our method needs the help of a contour-integral
based technique proposed in [29], we briefly describe the technique in Sect. 3. In Sect. 4, we first detail the second finding and then give the resulting method for counting the eigenvalues inside the disk \(D\). Numerical experiments are reported in Sect. 5 to show the viability of our method.

Throughout the paper, the following notation and terminology are used. The real part of a complex number \(a\) is denoted by \(\Re(a)\). We use \(\sqrt{-1}\) to denote the imaginary unit. The subspace spanned by the columns of matrix \(X\) is denoted by \(\text{span}\{X\}\). The rank of \(X\) is denoted by \(\text{rank}(X)\). The algorithms are presented in MATLAB style.

## 2 Approximate Spectral Projector

Our discussion starts with the spectral projector associated with the eigenvalues inside \(\Gamma\), which is constructed by the contour integral defined as

\[
Q = \frac{1}{2\pi \sqrt{-1}} \oint_{\Gamma} (zB - A)^{-1} Bdz. \tag{2}
\]

A matrix pencil \(zB - A\) is called regular if \(\det(zB - A)\) is not identically zero for all \(z \in \mathbb{C}\) [8,18]. Below is a generalization of the Jordan canonical form to the regular matrix pencil case.

**Theorem 1** (Weierstrass canonical form [11]) Let \(zB - A\) be a regular matrix pencil of order \(n\). Then there exist nonsingular matrices \(S\) and \(T \in \mathbb{C}^{n \times n}\) such that

\[
TAS = \begin{bmatrix} J_d & 0 \\ 0 & I_{n-d} \end{bmatrix} \quad \text{and} \quad TBS = \begin{bmatrix} I_d & 0 \\ 0 & N_{n-d} \end{bmatrix}, \tag{3}
\]

where \(J_d\) is a \(d \times d\) matrix in Jordan canonical form with its diagonal entries corresponding to the eigenvalues of \(zB - A\), \(N_{n-d}\) is an \((n-d) \times (n-d)\) nilpotent matrix also in Jordan canonical form, and \(I_d\) denotes the identity matrix of order \(d\).

Assume that there are \(n\) independent eigenvectors, which implies \(J_d\) is a diagonal matrix and \(N_{n-d}\) is a zero matrix. Let \(J_d = \text{diag}({\lambda}_1, {\lambda}_2, \ldots, {\lambda}_d)\), with \(\lambda_i\) being the (finite) eigenvalues [8]. Here the \(\lambda_i\) are not necessarily distinct and can be repeated according to their multiplicities.

For \(z \neq \lambda_i\), the matrix \((zI_d - J_d)\) is invertible. Hence, according to (3), the resolvent operator

\[
(zB - A)^{-1}B = S \begin{bmatrix} (zI_d - J_d)^{-1} & 0 \\ 0 & (zN_{n-d} - I_{n-d})^{-1} \end{bmatrix} T B
\]

\[
= S \begin{bmatrix} (zI_d - J_d)^{-1} & 0 \\ 0 & (zN_{n-d} - I_{n-d})^{-1} \end{bmatrix} \begin{bmatrix} I_d & 0 \\ 0 & N_{n-d} \end{bmatrix} S^{-1}
\]

\[
= SD(z)S^{-1}, \tag{4}
\]

where

\[
D(z) = \begin{bmatrix} (zI_d - J_d)^{-1} & 0 \\ 0 & 0 \end{bmatrix}. \tag{5}
\]
and the diagonal block \((zI_d - J_d)^{-1}\) is of the form:

\[
(zI_d - J_d)^{-1} = \begin{bmatrix}
\frac{1}{z - \lambda_1} & 0 & \cdots & 0 \\
0 & \frac{1}{z - \lambda_2} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \frac{1}{z - \lambda_d}
\end{bmatrix}.
\]  

(6)

Assume that there are \(s\) eigenvalues enclosed by \(\Gamma\), without loss of generality, let them be \(\{\lambda_1, \ldots, \lambda_s\}\). Then according to the residue theorem in complex analysis [1], it follows from (4)–(6) that

\[
Q = S \left[ \frac{1}{2\pi \sqrt{-1}} \oint_{\Gamma} D(z) dz \right] S^{-1} = S \begin{bmatrix} I_s & 0 \\ 0 & 0 \end{bmatrix} S^{-1} = S(\mathcal{S}(\{1:s\})) (S^{-1})(\{1:s\}).
\]  

(7)

It is easy to verify that \(Q^2 = Q\), which implies that \(Q\) is a spectral projector onto the eigenspace \(\text{span}\{\mathcal{S}(\{1:s\})\}\) corresponding to \(\{\lambda_1, \lambda_2, \ldots, \lambda_s\}\) [29].

In view of (7), the spectral projector \(Q\) can be obtained via computing the Weierstrass canonical form of \(zB - A\) (see Theorem 1). However, it is well-known that the Weierstrass canonical form is not suitable for numerical computation [2,8]. According to the expression (2), an alternative way is to compute \(Q\) numerically by a quadrature scheme. In our method, the quadrature scheme is restricted to the Gauss–Legendre quadrature rule [7]. Let \(c\) and \(\rho\) be the center and the radius of circle \(\Gamma\), respectively. Applying the \(q\)-point Gauss–Legendre quadrature rule to (2) yields

\[
Q \approx \tilde{Q} = \frac{1}{2} \sum_{j=1}^{q} \omega_j (z_j - c)(z_j B - A)^{-1} B = S \left[ \frac{1}{2} \sum_{j=1}^{q} \omega_j (z_j - c) D(z_j) \right] S^{-1}.
\]  

(8)

Here \(z_j = c + \rho e^{\sqrt{-1} \theta_j}\), \(\theta_j = (1 + t_j)\pi\), and \(t_j\) is the \(j\)th Gaussian node with associated weight \(\omega_j\). We remark that \(\tilde{Q}\) is an approximate spectral projector. Let

\[
D = \frac{1}{2} \sum_{j=1}^{q} \omega_j (z_j - c) D(z_j).
\]  

(9)

Comparing to (7) and (8), we see that \(D\) is an approximation to \(\mathcal{S}(\{1:s\})\).

Let \(\mu = c + r e^{\sqrt{-1} \theta}\), where \(r \in [0, \infty)\) and \(\theta \in (-\pi, \pi]\). Define

\[
\psi(\mu) = \frac{1}{2\pi \sqrt{-1}} \oint_{\Gamma} \frac{1}{z - \mu} dz.
\]  

(10)

According to the residue theorem, we know that \(\psi(\mu) = 1\) when \(\mu\) is located inside \(\Gamma\), and \(\psi(\mu) = 0\) when \(\mu\) is located outside \(\Gamma\). If \(\psi(\mu)\) is computed approximately by the \(q\)-point
Fig. 1 This figure illustrates the function $\Re[\tilde{\psi}(\mu)]$ (cf. (12)) when $r \in [0, 4]$. The circle $\Gamma$ has radius $\rho = 1$ with center at the origin. We set the degree of Gauss–Legendre quadrature rule $q = 16$. The left picture shows the general shape of the function, and the right one shows the logarithmic scale.

Gauss–Legendre quadrature rule, then we have

$$
\psi(\mu) \approx \tilde{\psi}(\mu) = \frac{1}{2} \sum_{j=1}^{q} \omega_j (z_j - c) \frac{1}{z_j - \mu}
$$

$$
= \frac{1}{2} \sum_{j=1}^{q} \omega_j \left( \frac{-\rho \cos(t_j \pi) - \sqrt{-1} \rho \sin(t_j \pi)}{-(\rho \cos(t_j \pi) + r \cos \theta) - \sqrt{-1}(\rho \sin(t_j \pi) + r \sin \theta)} \right)
$$

$$
= \frac{1}{2} \sum_{j=1}^{q} \omega_j \left( \frac{\rho^2 + \rho r \cos(t_j \pi - \theta)}{\rho^2 + r^2 + 2\rho r \cos(t_j \pi - \theta)} \right). \tag{11}
$$

It was shown that $\tilde{\psi}(\mu) \geq \frac{1}{2}$ if $\mu$ is real-valued and located inside $\Gamma$ [27]. This observation sheds light on the following theorem.

**Theorem 2** If $\mu$ is enclosed by $\Gamma$, then the real part of $\tilde{\psi}(\mu)$ satisfies

$$
\Re[\tilde{\psi}(\mu)] > \frac{1}{2}.
$$

If $\mu$ is located outside $\Gamma$, we have

$$
\Re[\tilde{\psi}(\mu)] < \frac{1}{2}.
$$

**Proof** According to (11), we have

$$
\Re[\tilde{\psi}(\mu)] = \frac{1}{2} \sum_{j=1}^{q} \omega_j \frac{\rho^2 + \rho r \cos(t_j \pi - \theta)}{\rho^2 + r^2 + 2\rho r \cos(t_j \pi - \theta)}. \tag{12}
$$

Let

$$
g_j(r, \theta) = \frac{\rho^2 + \rho r \cos(t_j \pi - \theta)}{\rho^2 + r^2 + 2\rho r \cos(t_j \pi - \theta)}, \quad j = 1, \ldots, q.
$$
For any given \( j \), one can show that

\[
g_j(r, \theta) - \frac{1}{2} = \frac{\rho^2 - r^2}{2\left[\rho + r \cos(t_j \pi - \theta)\right]^2 + (r \sin(t_j \pi - \theta))^2}.
\]

(13)

Note that the denominator of the right hand in (13) is positive. It is readily to see that

\[
g_j(r, \theta) > \frac{1}{2}
\]

for \( r \in [0, \rho) \) and \( \theta \in (-\pi, \pi] \), in which case \( \mu \) is enclosed by \( \Gamma \); and

\[
g_j(r, \theta) < \frac{1}{2}
\]

for \( r \in (\rho, +\infty) \) and \( \theta \in (-\pi, \pi] \), in which case \( \mu \) is located outside \( \Gamma \).

On the other hand, it is well-known that \( \sum_{j=1}^{q} \omega_j = 2 \) [7,27]. Therefore,

\[
\Re[\tilde{\psi}(\mu)] = \frac{1}{2} \sum_{j=1}^{q} \omega_j g_j(r, \theta) > \frac{1}{2}
\]

when \( \mu \) is enclosed by \( \Gamma \), and

\[
\Re[\tilde{\psi}(\mu)] < \frac{1}{2}
\]

when \( \mu \) is located outside \( \Gamma \). \qedhere

We use Fig. 1 to depict the function \( \Re[\tilde{\psi}(\mu)] \). The figure well demonstrates the conclusion presented in Theorem 2. It is shown that \( \Re[\tilde{\psi}(\mu)] \) approximates to 1 when \( \mu \) is located inside but not close to the circle, and approximates to 0 when \( \mu \) is located outside but not close to the circle, as expected.

From (5), (6), and (9)–(11), we see that

\[
D_{(i,i)} = \tilde{\psi}(\lambda_i), \quad i = 1, \ldots, d.
\]

(14)

Due to this, in the following we always refer to \( D_{(i,i)} \) as the diagonal entry of \( D \) that corresponds to the eigenvalue \( \lambda_i \), \( i = 1, \ldots, d \). In view of Theorem 2, the diagonal entries of \( D \) can be divided into two categories, that is, \( \{ \Re[D_{(i,i)}] > \frac{1}{2}\}_{i=1}^{s} \) in one group and \( \{ \Re[D_{(i,i)}] < \frac{1}{2}\}_{i=s+1}^{n} \) in the other. This fact is the first finding of our work, which implies that the sought-after number \( s \) can be obtained by counting the diagonal entries of \( D \) whose real parts are larger than \( \frac{1}{2} \). According to (9), getting the diagonal matrix \( D \) requires to compute the Weierstrass canonical form of \( zB - A \). However, as was suggested in [2], the Weierstrass canonical form is not suitable for numerical computation.

In our work, we present an alternative way to obtain \( \{D_{(i,i)}\}_{i=1}^{s} \) as our second finding, which does not need to compute the Weierstrass canonical form of \( zB - A \). As a result, when combined with the first finding, we can exactly compute the number of eigenvalues of (1) inside \( \Gamma \). Since it needs the help of a contour-integral based technique proposed in [29], we briefly introduce this technique in next section before presenting our second finding.
3 Finding an Upper Bound for the Number of Eigenvalues Inside $\Gamma$

In [29], a method based on contour integral was proposed for finding an upper bound of the number of eigenvalues inside a given region. Meanwhile, it can produce an approximate projection onto the eigenspace corresponding to the eigenvalues inside the given region.

The method first uses a stochastic estimation of the trace of spectral projector $Q$ (cf. (2)) to obtain an estimation of $s$. By $Y_p \sim N_{n \times p}(0, 1)$, we mean that $Y_p$ is an $n \times p$ random matrix with independent and identically distributed entries drawn from standard normal distribution $N(0, 1)$. Let $U_p = QY_p$, making use of (7) one can easily verify that

$$
\frac{1}{p} \mathbb{E}[\text{trace}(Y_p^*U_p)] = \text{trace}(Q) = \text{trace}(S_{(\cdot,1:3)}(S^{-1}_{(1:3,\cdot)}))
$$

$$
= \text{trace}((S^{-1}_{(1:3,\cdot)})S_{(\cdot,1:3)}) = \text{trace}(I_3)
$$

$$
= s.
$$

Therefore,

$$
s_0 := \left\lfloor \frac{1}{p} \text{trace}(Y_p^*U_p) \right\rfloor
$$

provides an estimation for $s$, see [10,22,29] for more details. With this knowledge on hand, a method was then given in [29] to seek an upper bound, say $s_1$, of $s$. In order to derive the method, we need the following lemma.

**Lemma 1** ([29]) Let $Y \in \mathbb{R}^{n \times m}$. If the entries of $Y$ are random numbers from a continuous distribution and that they are independent and identically distributed (i.i.d.), then the matrix $(S^{-1}_{(1:3,\cdot)})Y$ is almost surely nonsingular.

Let $s^\dagger$ be a positive integer and $Y_{s^\dagger} \sim N_{n \times s^\dagger}(0, 1)$. Consider

$$
U_{s^\dagger} = QY_{s^\dagger} = \frac{1}{2\pi \sqrt{-1}} \oint_{\Gamma} (zB - A)^{-1} BdzY_{s^\dagger} = S_{(\cdot,1:3)}(S^{-1}_{(1:3,\cdot)})Y_{s^\dagger}.
$$

Recall that $Q$ is a spectral projector onto the the eigenspace span$\{S_{(\cdot,1:3)}\}$. Therefore, $U_{s^\dagger}$ is the projection of $Y_{s^\dagger}$ onto span$\{S_{(\cdot,1:3)}\}$, which implies rank$(U_{s^\dagger}) \leq s$. With this in mind, if rank$(U_{s^\dagger}) = s^\dagger$, then we have $s^\dagger \leq s$. Otherwise, if rank$(U_{s^\dagger}) < s^\dagger$, we can conclude that $s = \text{rank}(U_{s^\dagger})$ with the help of Lemma 1, and thereby $s < s^\dagger$. Based on these arguments, the following algorithm was proposed in [29] for finding an upper bound for $s$. Meanwhile, a projection matrix onto the eigenspace span$\{S_{(\cdot,1:3)}\}$ is generated, which will play an important role in the resulting algorithm presented in next section.

**Algorithm 1** Input an increasing factor $\alpha > 1$, the size $p$ of sample vectors, and a threshold $\varepsilon$. The function “SEARCH” outputs $s_1$, an upper bound of the number of eigenvalues $s$ inside $\Gamma$, and a projection matrix $U_1 \in \mathbb{C}^{n \times s_1}$ onto span$\{S_{(\cdot,1:3)}\}$.

Function $[U_1, s_1] = \text{SEARCH}(A, B, \Gamma, \alpha, p, \varepsilon)$

1. Pick $Y_p \sim N_{n \times p}(0, 1)$ and compute $U_p = \frac{1}{2\pi \sqrt{-1}} \oint_{\Gamma} (zB - A)^{-1} BdzY_p$

by the $q$-point Gauss–Legendre quadrature rule.

2. Set $s_0 = \left\lfloor \frac{1}{p} \text{trace}(Y_p^*U_p) \right\rfloor$ and $s^\dagger = \min(\max(p, s_0), n)$.

3. If $s^\dagger > p$,

4. Pick $\hat{Y} \sim N_{n \times (s^\dagger - p)}(0, 1)$ and compute $\hat{U} = \frac{1}{2\pi \sqrt{-1}} \oint_{\Gamma} (zB - A)^{-1} Bdz\hat{Y}$
by the \( q \)-point Gauss–Legendre quadrature rule.

5. Augment \( \hat{U} \) to \( U_p = [U_p, \hat{U}] \in \mathbb{C}^{n \times s^\dagger} \).

6. Else

7. Set \( s^\dagger = p \).

8. End

9. Compute the rank-revealing QR decomposition \([6, 12]\) of \( U_p \) with column
   pivoting strategy: \( U_p \Pi = [U_1, U_2] \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix} \), here \( \|R_{22}\|_2 \leq \varepsilon \).

10. Set \( s_1 = \text{rank}(R_{11}) \).

11. If \( s_1 < s^\dagger \), stop. Otherwise, set \( p = s_1 \) and \( s^\dagger = \lceil \alpha s_1 \rceil \). Then go to Step 3.

In Step 9, we perform the rank-revealing QR decomposition to compute the \( \varepsilon \)-rank \([12]\) of \( U_p \), by which we detect the rank deficiency of \( U_p \). Algorithm 1 provides an upper bound \( s_1 \) for the sought-after number \( s \) and an approximate projection \( U_1 \) onto \( \text{span}\{S_{(\cdot,:):I}\} \). With its help, we develop our method in the next section, which computes the number of eigenvalues inside \( \Gamma \) exactly.

### 4 Counting the Eigenvalues Inside \( \Gamma \)

In this section, we present a contour-integral based method for counting the eigenvalues inside a given disk in the complex plane. We start with our second finding.

**Theorem 3** Let \( V_1 \) be the orthonormal basis of the subspace \( \text{span}\{S_{(\cdot,:):I}\} \), where \( \mathcal{I} \subseteq \{1, \ldots, d\} \) is an index set with cardinality \( t \). Compute the approximated projection matrix \( V_2 = QV_1 \) by the \( q \)-point Gauss–Legendre quadrature rule

\[
V_2 \approx \tilde{V}_2 = \frac{1}{2} \sum_{j=1}^{q} \omega_j (z_j - c)(z_j B - A)^{-1} BV_1. \tag{16}
\]

Define the \( t \times t \) matrix

\[
M' = V_1^* \tilde{V}_2, \tag{17}
\]

the eigenvalues of \( M' \) are \( \{D_{(i,i)}:\mathcal{I}\} \).

**Proof** Since \( V_1 \) is the orthonormal basis of \( \text{span}\{S_{(\cdot,:):I}\} \), there exists a nonsingular matrix, say \( W \), such that

\[ S_{(\cdot,:):I} = V_1 W. \]

By (8) and (17), we have

\[
M' = V_1^* \tilde{V}_2 = V_1^* (SDS^{-1}) V_1 \\
= V_1^* (SDS^{-1}) (S_{(\cdot,:):I} W^{-1}) \\
= V_1^* S_{(\cdot,:):I} D_{(\cdot,:):\mathcal{I}} W^{-1} \\
= V_1^* (V_1 W) D_{(\cdot,:):\mathcal{I}} W^{-1} \\
= WD_{(\cdot,:):\mathcal{I}} W^{-1}.
\]

Since \( W \) is nonsingular, the matrices \( M' \) and \( D_{(\cdot,:):\mathcal{I}} \) have the same eigenvalues, which are \( \{D_{(i,i)}:\mathcal{I}\} \).

\( \square \)
Our first finding tells us that the sought-after number $s$ can be obtained by counting $D(i,i)$, $i = 1, \ldots, n$, whose real parts are larger than $\frac{1}{2}$. However, to get the matrix $D$, we have to compute the Weierstrass canonical form of the matrix pencil $zB - A$, which is not suitable for numerical computation. Our second finding (Theorem 3) tells us that if $\{1, \ldots, s\} \subseteq \mathcal{I}$, then there are $s$ eigenvalues of $M'$ whose real parts are larger than $\frac{1}{2}$ and the real parts of the other $t - s$ eigenvalues of $M'$ are less than $\frac{1}{2}$, which sheds light on an alternative way to get the number $s$. To this end, the most important task is to seek an orthonormal basis $V$ such that
\[
\text{span}\{S(c,:\mathcal{I})\} \subseteq \text{span}\{V_1\} = \text{span}\{S(c,:\mathcal{I})\}.
\]

Fortunately, we find that the approximate projection matrix $U_1$ computed by Algorithm 1 meets the requirement. Now we will justify our choice for $V_1$ below.

By (8), we know that $U_p$ (line 5) formed in the last iteration in Algorithm 1 is
\[
U_p = \tilde{Q}Y_{s^+} = S \left[ \frac{1}{2} \sum_{j=1}^{q} \omega_j (z_j - c) D(z_j) \right] S^{-1}Y_{s^+} = [D(1,1)S(1,:), \ldots, D(d,d)S(d,:), 0, \ldots, 0]S^{-1}Y_{s^+} = [\tilde{\psi}(\lambda_1)S(1,:), \ldots, \tilde{\psi}(\lambda_d)S(d,:), 0, \ldots, 0]S^{-1}Y_{s^+},
\]
where $Y_{s^+} \sim N_{n \times s^+}(0, 1)$ with $s^+ > s_1$, $s_1$ is the $\varepsilon$-rank of $U_p$.

It has been shown in Sect. 2 that if $\lambda_i$ is located inside $\Gamma$ we have $\Re[\tilde{\psi}(\lambda_i)] > \frac{1}{2}$, otherwise $\Re[\tilde{\psi}(\lambda_i)] < \frac{1}{2}$. Let $\mu = c + re^{\sqrt{-1}\theta}$, where $r \in [0, \infty)$ and $\theta \in (-\pi, \pi]$, it follows from that (11), for any given $\theta$, $|\tilde{\psi}(\mu)|$ is a decreasing function of $r$ in $[\rho, \infty]$ and approximates to 0 as $r$ approaches to positive infinity (Fig. 1 gives us an intuitive impression). For instance, taking $\mu = c + 3\rho$, we get $|\tilde{\psi}(\mu)| = 1.84 \times 10^{-10}$. Without loss of generality, we can assume that
\[
|\tilde{\psi}(\lambda_1)| \geq \cdots \geq |\tilde{\psi}(\lambda_s)| \geq \cdots \geq |\frac{1}{2} \geq \cdots \geq |\tilde{\psi}(\lambda_{d})|.
\]

In Step 9 in Algorithm 1, we use the rank-revealing QR decomposition (RRQR) to compute the $\varepsilon$-rank of $U_p$. The termination criterion is $\|R_{22}\|_2 \leq \varepsilon$. The RRQR technique can detect rank deficiency of $U_p$ if $\varepsilon$ is small [12, Chapter 5.4.3]. A typical choice of $\varepsilon$ is $\varepsilon_0 \|U_p\|_2$ for some small machine-dependent parameter $\varepsilon_0$ [6,12]. According to Lemma 1, it is known that $S^{-1}Y_{s^*}$ is full column-rank. Now the $\varepsilon$-rank $s_1$ is less than $s^*$, which means that $U_p$ is rank-deficient. Due to this fact, in view of (19), we can conclude that $|\tilde{\psi}(\lambda_{s_1+1})|, \ldots, |\tilde{\psi}(\lambda_d)|$ are sufficiently close to zero. Otherwise, the numerical rank of $U_p$ must be larger than $s_1$. If $|\tilde{\psi}(\lambda_j)| = 0$, $j = s_1 + 1, \ldots, d$, then it follows from (19) that
\[
\text{span}\{U_p\} = \text{span}\{U_1\} = \text{span}\{S(c,:s_1)\},
\]
which inspires that $U_1$ computed in Algorithm 1 can play the role of $V_1$ in Theorem 3. Compute the refined projection matrix of $U_1$ numerically by the $q$-point Gauss–Legendre quadrature rule, that is
\[
U_2 = QU_1 \approx \tilde{U}_2 = \frac{1}{2} \sum_{j=1}^{q} \omega_j (z_j - c)(z_jB - A)^{-1}BU_1.
\]
Define the $s_1 \times s_1$ matrix
\[
M = U_1^* \tilde{U}_2.
\]
By Theorem 3, we have that the eigenvalues of $M$ are $\{D_{(i,i)}\}_{i=1}^{s_1}$, which are the $s_1$ largest diagonal entries of $D$. We can obtain the sought-after number $s$ by counting these eigenvalues whose real parts are larger than 0.5.

Summarizing the above discussions, we give the complete algorithm for computing the number of eigenvalues inside $\Gamma$.

**Algorithm 2** Input an increasing factor $\alpha > 1$, the size $p$ of sample vectors, and a threshold $\varepsilon$. The function “COUNT_EIGS” computes the number of eigenvalues of (1) that are located inside circle $\Gamma$.

Function $s = \text{COUNT_EIGS}(A, B, \Gamma, \alpha, p, \varepsilon)$

1. Call $[U_1, s_1] = \text{SEARCH}(A, B, \Gamma, \alpha, p, \varepsilon)$.
2. Compute $U_2 = QU_1$ by the $q$-point Gauss–Legendre quadrature rule to get $\tilde{U}_2$.
3. Compute the eigenvalues of $M = U_1^* \tilde{U}_2$, and set $s$ to be the number of the computed eigenvalues whose real parts are larger than $\frac{1}{2}$.

Now we analyse the computational cost of Algorithm 2. The computational work of Algorithm 2 mainly includes two parts. The first part is computing two approximate projection matrices $U_1$ and $\tilde{U}_2$ in the first two steps, where they need to compute three contour integrals approximately by the $q$-point Gauss–Legendre quadrature scheme. More precisely, the first contour integral is devoted to obtaining the estimation $s_0$ (see (15)); the second is devoted to finding the upper bound $s_1$ and the approximate projection matrix $U_1$; the last one is computed for obtaining $\tilde{U}_2$, the refined projection matrix of $U_1$. In view of (22), to obtain the approximation of each contour integral, it needs to solve $q$ generalized shifted linear systems of the form

$$(z_j B - A)X_j = BY,$$

where $z_j$ are the quadrature nodes. The second part is computing the eigenvalues of matrix $M$ (see (23)), whose size is $s_1 \times s_1$. Note that $s_1$ is an upper bound of the sought-after number $s$, its value is small if the value of $s$ is small. Therefore, the expense of computing the eigenvalues of $M$ is negligible comparing to the overhead of solving the linear systems (24). Therefore, solving $3q$ linear systems of the form (24) is the dominant computational work of our method, and they can be solved by any method of the user’s choice. Since the quadrature nodes $z_j$ and the columns of the right-hand sides of (24) are independent, our method (Algorithm 2) has good scalability in modern parallel architectures.

Our method can help the recently developed contour-integral based eigensolvers [19,23, 24,29], which were formulated for computing the eigenvalues inside a given region, to design stopping criteria. For clarity, here we take the non-Hermitian FEAST algorithm (GFEAST) proposed in [29] as an example to show how to integrate our method with the contour-integral based eigensolvers. Note that the approximate projection matrices $U_1$ and $\tilde{U}_2$ computed in our method (Algorithm 2) are also computed at the first iteration and the second, respectively, in the GFEAST algorithm. Therefore to get the number $s$, the extra price has to be paid in the GFEAST algorithm is constructing the matrix $M = U_1^* \tilde{U}_2$ and computing its eigenvalues, which is always negligible in comparison with the expense of computing $U_1$ and $\tilde{U}_2$. Since we know the exact number $s$, if $s$ computed eigenpairs attain the prescribed accuracy at some iteration, we stop the GFEAST algorithm.
5 Numerical Experiments

In this section, we give some numerical experiments to illustrate the viability of our method. All computations are carried out in MATLAB version R2012b on a MacBook with an Intel Core i5 2.5 GHz processor and 8 GB RAM.

Experiment 5.1 Our method (Algorithm 2) is motivated by two findings: (i) for matrix $D$ (cf. (9)), $\Re[D_{(i,i)}] > \frac{1}{2}$ if $D_{(i,i)}$ correspond to the eigenvalues enclosed by $\Gamma$, and $\Re[D_{(i,i)}]$ are less than $\frac{1}{2}$ if $D_{(i,i)}$ correspond to the eigenvalues outside $\Gamma$; (ii) the eigenvalues of $M$ (cf. (23)) are $\{D_{(i,i)}\}_{i=1}^{s_1}$, which are the largest $s_1$ diagonal entries of $D$. This experiment is devoted to illustrating these two findings.

Let $A = \text{diag}([0.1 : 0.1 : 0.8])$, $S = \text{randn}(8)$, the matrices $A$ and $B$ are given by

$$A = SAS^{-1}, \quad B = \text{eye}(8).$$

Here $\text{diag}$, $\text{randn}$, and $\text{eye}$ are MATLAB commands. Obviously, for the problem under consideration, the eigenvalues are 0.1, 0.2, ... , 0.8. Let $\Gamma$ be a circle with center at origin and radius $p = 0.401$. Suppose that we are interested in the number of eigenvalues inside $\Gamma$, it is clear that there are 4 eigenvalues inside $\Gamma$. Note that the eigenvalue 0.4 is located inside $\Gamma$ and close to boundary of the disk surrounded by $\Gamma$.

In this experiment, we select the number of quadrature points $q = 32$. According to (9), now $D$ is given by

$$D = \frac{1}{2} \sum_{i=1}^{32} \omega_i z_i (z_i I_8 - A)^{-1},$$

where $\omega_i$ are the weights associated with quadrature nodes $z_i$. We take the size of sample vectors $p = 6$, thus the starting basis in function $\text{SEARCH}$ is $Y_6 = \text{randn}(8, 6)$. Since the size of test problem is small and the number of columns of $Y_6$ is already larger than the number of eigenvalues inside $\Gamma$, we just run one iteration when preforming function $\text{SEARCH}$ (Algorithm 1) to get the approximate projection matrix $U_1$. As a result, the matrix $M$ defined in (23) is of size $6 \times 6$.

Since we are only interested in the real parts of the diagonal entries of $D$, in the second column in Table 1 we list $\Re[D_{(i,i)}]_{i=1}^{s_1}$ that are computed by (25). It can be seen that $\Re[D_{(i,i)}]_{i=1}^{s_1}$, corresponding to the eigenvalues inside $\Gamma$, are larger than 0.5. More precisely, $\Re[D_{(i,i)}]$, $i = 1, 2, 3$, corresponding to eigenvalues 0.1, 0.2, 0.3, respectively, approximate to the theoretical value 1 sufficiently well. But $\Re[D_{(i,i)}]$ is about 0.8, this is because it corresponds to the eigenvalue 0.4, which is close to the boundary of the target disk. On the
other hand, the real parts of \( \{D_{i,i}\}_{i=5}^8 \), corresponding to the eigenvalues outside \( \Gamma \), are less than 0.5 and very close to zero, as predicted by our first finding.

The third column in Table 1 shows the real parts of the eigenvalues of \( M \) in descending order. The same digits of \( \Re[D_{i,i}] \) and the \( i \)th largest \( \Re[\text{eig}(M)] \), \( i = 1, \ldots, 6 \) are underlined. We can see that \( \Re[D_{i,i}] \), \( i = 1, \ldots, 6 \), agree at least fourteen digits to their counterparts in the third column. Therefore the real parts of the eigenvalues of \( M \) are almost equal to \( \Re[D_{i,i}] \), \( i = 1, \ldots, 6 \), i.e., the six largest real parts of the diagonal entries of \( D \), which justifies the second finding of our work.

**Experiment 5.2** This experiment is devoted to testing the viability of our method. The test matrices are downloaded from the Matrix Market collection [4]. They are the real-world problems from scientific and engineering applications. The descriptions of the matrices are presented in Table 2, where \( \text{nnz} \) denotes the number of non-zero entries and \( \text{cond} \), the condition numbers of the test matrices, are computed by the MATLAB function `condest`. The test matrices of different problems vary in size, spectrum and other properties.

In this experiment, as for computing the generalized shifted linear systems of the form (24), we first use the MATLAB function `lu` to compute the LU decomposition of \( z_j B - A \), \( j = 1, 2, \ldots, q \), and then perform the triangular substitutions to get the corresponding solutions.

We set the parameter \( q \) to be 16. The parameters \( c \) and \( \rho \) are the center and the radius of circle \( \Gamma \), respectively. Note that the test problems 3, 4, 6, and 8 are Hermitian problems, which means their (finite) eigenvalues are real-valued. Due to this, we choose the circles with centers lying on the real line for these test problems.

Table 3 presents the numerical results, in which \( s \) is the actual number of eigenvalues inside \( \Gamma \). We first use the MATLAB built-in function `eig` to compute all eigenvalues of the test problems and get the true value of \( s \). We also show the estimation \( s_0 \) computed by the
Experiment 5.3 The dominant work of our method (Algorithm 2) is to solve 3q generalized shifted linear systems of form (24). In the previous experiment, we use the direct method to solve the related linear systems. This experiment is devoted to showing the numerical performance of our method when the iterative method is used.

In this experiment, the matrix data of test problems are from ELSES Matrix Library (http://www.elses.jp/matrix/). It was generated by ELSES [15], a quantum mechanical nano-material simulator. The test problem is VCNT22500, whose size is 22,500 × 22,500. It is for simulating a vibrating carbon nanotube within a supercell with symmetric positive definite orbitals [5].

We consider using the iterative methods to compute (24). Here we use the preconditioned GMRES method (P-GMRES) with an ILU factorization using a drop tolerance of droptol = 0.01 and a pivoting threshold of 0.05 as defined by the ilu function from MATLAB to solve the linear systems

\[(z_iB - A)(X_i)_{(c,j)} = BY_{(c,j)} \quad i = 1, \ldots, q, \ j = 1, \ldots, s_1.\] (26)

For all the runs with P-GMRES, we use a restart dimension of m = 10 and limit the number of steps to 200. The iteration process is stopped when the residual norm defined by

\[r_i^j = \frac{\|BY_{(c,j)} - (z_iB - A)(X_i)_{(c,j)}\|_2}{\|BY_{(c,j)}\|_2}\] (27)

is less than a certain convergence tolerance (tol). In order to test the numerical performance when the linear systems (26) solved by iterative method with different accuracy, we use the values for tol: 1.0 × 10^{-4}, 1.0 × 10^{-6}, and 1.0 × 10^{-8}.

In this experiment, we seek the number of eigenvalues inside circle Γ with center c = −1 and radius ρ = 0.001. There are 12 eigenvalues enclosed by Γ. In Table 4, we report the

| No. | (c, ρ) | s     | s_0  | s_1  | C_Eigs |
|-----|--------|-------|------|------|--------|
| 1   | (−6.0 × 10^5) + √−1(2.0 × 10^5), 3.0 × 10^5 | 120   | 136  | 177  | 120    |
| 2   | (−5.0 × 10^5) + √−1(1.0 × 10^5), 2.0 × 10^5 | 165   | 161  | 228  | 165    |
| 3   | (5.0 × 10^5, 3.0 × 10^5) | 178   | 199  | 190  | 178    |
| 4   | (5.0 × 10^3, 3.0 × 10^3) | 160   | 134  | 192  | 160    |
| 5   | (5.0 × 10^5, 1.5 × 10^{-1}) | 301   | 297  | 397  | 301    |
| 6   | (6.0 × 10^5, 3.5 × 10^6) | 232   | 229  | 262  | 232    |
| 7   | (−5.0 × 10^2) + √−1(2.0 × 10^2), 4.0 × 10^2 | 212   | 545  | 293  | 212    |
| 8   | (5.0 × 10^5, 2.0 × 10^5) | 1663  | 1628 | 1749 | 1663   |

trace formula (15) and the upper bound s_1 computed by Algorithm 1. C_Eigs is the result computed by our new method.

The results for all eight test problems are reported in Table 3. From these data, we see that the estimation s_0 computed by the trace formula (15) always is a good estimation to s for all test problems except for test problem 7, due to its ill-condition. Algorithm 1 always gives a good upper bound s_1 for s. It is remarkable that the result computed by our method is the same with the exact number s for each test problem, even for test problem 7. Therefore, our new method is numerically efficient and reliable.
eighteen largest real parts of the eigenvalues of matrix $M$, which is generated by Algorithm 2, in decreasing order. It can be seen that the number of the eigenvalues of $M$ whose real parts are larger than 0.5 are 12 for all three convergence tolerance, they all agree with the true number of eigenvalues inside $\Gamma$. There are obvious magnitude gaps between the twelfth and the thirteenth largest real parts of the eigenvalues of $M$ for three different convergence tolerance. Moreover, we can see that the results obtained with $tol = 1.0 \times 10^{-4}$ have about five-digit accuracy when compared with those obtained with $tol = 1.0 \times 10^{-6}$; while the results corresponding to the case that $tol = 1.0 \times 10^{-8}$ have seven the same digits with the results listed in the last column.

Experiment 5.4 It has been shown that when an eigenvalue, say $\lambda_i$, is close to the boundary of the target disk, the value of $\Re[D_{i,i}]$ approaches 0.5. As a consequence, the reliability and validity of our method may be effected due to the presence of rounding errors. The present experiment is devoted to showing the numerical performance of our method in the case where there is an eigenvalue very close to the boundary of the target disk.

We choose problem 2 as our test problem, which serves as a classical testbed for generalized eigenvalue problem [2]. To achieve our goal, we first use the MATLAB function `eig` to compute all eigenvalues of problem 2 and then select one of its eigenvalues, which is $\bar{\lambda} = -3.296886162426114 \times 10^5$. Different disks are constructed as follows: the centers $c$ are taken the same as the one in Table 3; while the radii vary such that the selected eigenvalue $\bar{\lambda}$ are close to the boundaries of the associated regions. In this experiment, we choose six different radii, which are 0.999, 0.9999, 0.99999, 1.001, 1.0001, and 1.00001 times of the
distance $|\lambda - c|$. As a result, the eigenvalue $\lambda$ is close to the boundaries of all constructed target regions.

Our method solves the numbers of eigenvalues inside all six disks successfully, although the eigenvalue $\lambda$ is very close to the boundaries of target disks. Remarkably, when the values of $\sigma$ are equal to 0.9999 and 1.00001, which means that $\lambda$ is close to the boundary sufficiently, our method is still able to obtain the exact number for $s$. Table 5 shows the values of $\Re[\sigma_{i-1}], \Re[\sigma_i], \Re[\sigma_{i+1}]$ and $\Re[\sigma_{i+2}]$ corresponding to different values of $\alpha$, where $\{\sigma_i\}$ are the eigenvalues of $\mathbf{M}$ (cf. (23)) sorted in descending order. When the parameter $\alpha$ is less than 1, the eigenvalue $\lambda$ is outside the target regions. In this case we see from Table 5 that the larger the values of $\alpha$ are, the closer the values of $\Re[\sigma_i]$ will be to 0.5, although the values of $\Re[\sigma_i]$ are less that 0.5. We have shown that $\Re[\sigma_i]$ should be equal to 0.5 when the eigenvalue is on the boundary. When $\alpha = 1.0001$, in which case the eigenvalue $\lambda$ is inside the associated disk and is very close to the boundary, we see that the value of $\Re[\sigma_i]$ approaches to 0.5 sufficiently close and is greater than 0.5, as expected.

Experiment 5.5 Our method is a contour-integral based method, the contour integrals involved must be computed by a numerical quadrature rule. In our method, we use the $q$-point Gauss–Legendre quadrature scheme. The parameter $q$ is the number of quadrature nodes, which may effect the numerical performance of the present method. The aim of this experiment is to illustrate the influence of the size of $q$ to the validity of our method.

Here we choose problem 5 as the test problem, whose eigenvalues appear in pairs. The test region is taken to the disk with center at $c = \sqrt{-1}(5.0 \times 10^{-1})$ and radius $\rho = 1.5 \times 10^{-1}$. There are $s = 376$ eigenvalues inside the selected test region.

In this experiment, we want to demonstrate the influence of the size of $q$ on the numerical performance of the proposed method. To this end, we select four different values for the parameter $q$, which are 4, 12, 24, and 36. In Table 6 we present the values of $\Re[\sigma_{i-1}], \Re[\sigma_i]$ and $\Re[\sigma_{i+1}]$, respectively, corresponding to different values of $q$. From Table 6 we see that $\Re[\sigma_i]$ are greater than 0.5 and $\Re[\sigma_{i+1}]$ are less than 0.5 for all four cases, which means that our method finds the exact number for $s$ even in the case of $q$ equals to 4. In fact, from the derivation of our method, we know that our method is irrelevant to the parameter $q$ but is plays a crucial role in finding an upper bound $s_1$ (see Algorithm 1). In Table 6 we also list the result of $s_1$ corresponding to different values of $q$. Recall that in Algorithm 1 we compute the projection matrix $\mathbf{QY}$, which is the projection of random matrix $\mathbf{Y}$ onto the desired eigenspace, by the $q$-point Gauss–Legendre quadrature scheme and then get an approximation projection matrix $\mathbf{U}_p$. The rank of $\mathbf{QY}$ is $s$, which is the dimension of the desired eigenspace. $s_1$ is the numerical rank of $\mathbf{U}_p$, which is an upper bound of $s$. Since $\mathbf{U}_p$ is the numerical approximation of $\mathbf{QY}$ computed by the $q$-point Gauss–Legendre quadrature scheme, the larger the values of $q$ is, the closer $s_1$ will be to $s$. This fact is verified by the results presented in Table 6. From the formulation of our method we can see that what our method needs is the subspace spanned by $\mathbf{U}_p$ containing the desired eigenspace, which is monitored by observing whether $\mathbf{U}_p$ is rank-deficient. Of course if the value of $q$ is small, the size of $s_1$ will be large, which means that the size of matrix $\mathbf{M}$ is also large. As a result, it needs higher computational cost to compute the eigenvalues of $\mathbf{M}$ in the last step in Algorithm 1.

Experiment 5.6 In many practical applications, the number of eigenvalues, rather than the eigenvalues themselves, inside a specific region is of interests. To the best of our knowledge, there is no specific approach for this purpose when the eigenproblem is non-Hermitian. An intuitive way to get the number is first computing the eigenvalues inside the target region and then counting them. Our work aims to develop a method for computing the number of eigenvalues inside a given region, but it does not need to compute the eigenvalues. This
Table 5 $\tilde{\lambda}$ is an eigenvalue of Problem 2

$\tilde{\lambda} = -3.296886162426114 \times 10^5, c = (-6.0 \times 10^5) + \sqrt{-1}(2.0 \times 10^5), \rho = \alpha|\tilde{\lambda} - c|

|   | 164   | 164   | 164   | 165   | 165   | 165   |
|---|-------|-------|-------|-------|-------|-------|
| $s$ | 0.99  | 0.999 | 0.999 | 1.001 | 1.000 | 1.000 |
| $\alpha$ | 0.60429279 | 0.61289712 | 0.61375172 | 0.56590363 | 0.55611204 | 0.55512957 |
| $\Re[\sigma_{s-1}]$ | 0.54406455 | 0.55392744 | 0.55491104 | 0.51168507 | 0.50116953 | 0.50011679 |
| $\Re[\sigma_s]$ | 0.48830316 | 0.4983042 | 0.4998308 | 0.30590641 | 0.29525526 | 0.29420324 |
| $\Re[\sigma_{s+1}]$ | 0.28256847 | 0.29292028 | 0.29396982 | 0.28760021 | 0.27717055 | 0.27614160 |

The parameters $c$ and $\rho$ are the center and the radii of target regions, respectively. $\{\sigma_s\}$ are the eigenvalues of $M$ (cf. (23)) sorted in descending order.
Table 6  $q$ is the number of quadrature nodes, $s_1$ is an upper bound of $s$ computed by Algorithm 2, $\{\sigma_i\}$ are the eigenvalues of $M$ (cf. (23)) sorted in descending order.

| $q$ | 4   | 12  | 24  | 36  |
|-----|-----|-----|-----|-----|
| $s_1$ | 1919 | 1119 | 558 | 500 |
| $\Re[\sigma_{s-1}]$ | 0.5102963292306995 | 0.5157456315250075 | 0.530759551032785 | 0.5457198025413754 |
| $\Re[\sigma_s]$ | 0.5000409851774534 | 0.500062665648214 | 0.5001225904309401 | 0.5001825121512026 |
| $\Re[\sigma_{s+1}]$ | 0.4976523775362070 | 0.4964090591079866 | 0.4929784638334438 | 0.4895472918707309 |
Table 7 Numerical comparison: timing required by the \texttt{eig}, CIRR, GFEAST and our method (Algorithm 2) to compute the number of eigenvalues inside the region given in Table 3

| No. | Size  | \texttt{Time (s) eig} | CIRR | GFEAST | Our methods |
|-----|-------|----------------------|------|--------|-------------|
| 1   | 398   | 1.11                 | 3.34 | 3.29   | 1.20        |
| 2   | 782   | 10.70                | 4.76 | 5.79   | 4.08        |
| 3   | 1074  | 0.58                 | 3.31 | 4.00   | 2.97        |
| 4   | 1224  | 27.84                | 14.69| 23.96  | 10.25       |
| 5   | 1919  | 108.76               | 41.21| 46.99  | 26.73       |
| 6   | 2003  | 171.33               | 73.64| 113.29 | 58.98       |
| 7   | 4800  | 1845.73              | 113.67| 297.08 | 36.56       |
| 8   | 15,439| 41288.43             | –    | 4362.55| 1552.61     |

experiment is devoted to comparing our method with the idea of obtaining the number \( s \) via solving eigenvalues.

We compare our method with the \texttt{MATLAB} function \texttt{eig}, CIRR [22,24] and GFEAST [29] in terms of timing. It should pointed out that both CIRR and GFEAST are developed for computing the eigenvalues inside a given region, so for \texttt{eig}, CIRR and GFEAST to get the number \( s \), we first use them to compute the eigenvalues inside the target disk. We have shown that the dominant computational work of our method is solving 3\( q \) linear systems of the form (24). It is known that the contour-integral based eigensolvers need to solve \( q \) linear systems of the form (24) at each iteration [22,29]. If \( n_c \) iterations are needed to achieve the prescribed accuracy, then the dominant computational cost of the contour-integral based eigensolvers is solving \((n_c + 1)q\) linear systems of the form (24) (including solving \( q \) linear systems for getting the estimation \( s_0 \) (see (15))). CIRR and GFEAST are two typical contour-integral based eigensolvers. The iteration numbers \( n_c \) required by CIRR and GFEAST are always not less than two [22,27,29]. Therefore, our method requires less overhead than using the contour-integral based eigensolvers to compute the eigenvalues. On the other hand, it was shown in [22,29] that CIRR and GFEAST are always more efficient than the \texttt{MATLAB} built-in function \texttt{eig}, even though the parallelism offered by the contour-integral based eigensolvers was not used.

Table 7 presents the clock time used by the four test methods to compute the number of the eigenvalues inside the given region. The test problems and the target regions are taken to the ones listed in Tables 2 and 3, respectively. The accuracy of eigenpairs \((\lambda_i, x_i)\) computed by CIRR and GFEAST are measured by

\[
r_i = \frac{\|A x_i - \lambda_i B x_i\|_2}{\|A x_i\|_2 + \|B x_i\|_2}, \quad 1 \leq i \leq s.
\]

The convergence tolerance is set to \( \eta = 1.0 \times 10^{-6} \) for these two contour-integral based eigensolvers.

We can see from Table 7 that our method requires less time than its counterparts, except for Problem 1 and Problem 3, whose sizes are relatively small. Therefore, if just the number of eigenvalues inside a given disk is of interest, our methods is more efficient than the ways of getting the number by using \texttt{eig} or the contour-integral based eigensolvers to compute the eigenvalues, especially when the problems are large.
6 Conclusion

In this work, we developed an approach for computing the number of eigenvalues of a generalized eigenproblem inside a given disk in the complex plane. Our method is a contour-integral based method and motivated by two findings. The computational advantage of our method is that it is easily parallelizable. Its another promising feature is that it can integrate with the recently proposed contour-integral based eigensolvers to provide them the information of the number of eigenvalues inside the target region. Error analysis when the related linear systems solved with different accuracy deserves further investigation. How to adapt the resulting method to the nonlinear problems will also be our future work.

Acknowledgements

I would like to thank Professor Raymond H. Chan, my thesis advisor, for his help in preparing this paper. I also would like to thank the anonymous reviewers for their useful suggestions which have greatly improved this paper. This work was supported by the National Natural Science Foundation of China (NSFC) under Grant 11701593.

References

1. Ahlfors, L.: Complex Analysis, 3rd edn. McGraw-Hill, Inc., New York City (1979)
2. Bai, Z., Demmel, J., Dongarra, J., Ruhe, A., Van der Vorst, H.: Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide. SIAM, Philadelphia (2000)
3. Bai, Z., Demmel, J., Gu, G.: An inverse free parallel spectral divide and conquer algorithm for nonsymmetric eigenproblem. Numer. Math. 76, 279–308 (1997)
4. Boisvert, R.F., Pozo, R., Remington, K., Barrett, R., Dongarra, J.: The matrix market: a web resource for test matrix collections. In: Boisvert, R.F. (ed.) Quality of Numerical Software: Assessment and Enhancement, IFIP Advances in Information and Communication Technology, pp. 125–137. Chapman & Hall, London (1997)
5. Cerdá, J., Soria, F.: Accurate and transferable extended Hückel-type tight-binding parameters. Phys. Rev. B 61, 7965 (2000)
6. Chan, T.T.: Rank revealing QR factorizations. Linear Algebra Appl. 88–89, 67–82 (1987)
7. Davis, P.J., Rabinowitz, P.: Methods of Numerical Integration, 2nd edn. Academic Press, Orlando (1984)
8. Demmel, J.: Applied Numerical Linear Algebra. SIAM, Philadelphia (1997)
9. Di Napoli, E., Polizzi, E., Saad, Y.: Efficient estimation of eigenvalue counts in an interval. arXiv:1308.4275
10. Futamura, Y., Tadano, H., Sakurai, T.: Parallel stochastic estimation method of eigenvalue distribution. JSIAM Lett. 2, 127–130 (2010)
11. Gantmacher, F.R.: The Theory of Matrices. Chelsea, New York (1959)
12. Golub, G.H., Van Loan, C.F.: Matrix Computations, 3rd edn. Johns Hopkins University Press, Baltimore (1996)
13. Grimes, R.G., Lewis, J.D., Simon, H.D.: A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems. SIAM J. Matrix Anal. Appl. 15, 228–272 (1994)
14. Hinrichsen, D., Pritchard, A.J.: Mathematical Systems Theory I: Modelling, State Space Analysis and Robustness. Springer, Berlin (2005)
15. Hoshi, T., Yamamoto, S., Fujiwara, T., Sogabe, T., Zhang, S.L.: An order-N electronic structure theory with generalized eigenvalue equations and its application to a ten-million-atom system. J. Phys. Condens. Matter 24, 165502 (2012)
16. Kamgnia, E.R., Philippe, B.: Counting eigenvalues in domains of the complex field. arXiv:1110.4797
17. Lin, L., Saad, Y., Yang, C.: Approximating spectral densities of large matrices. arXiv:1308.5467
18. Moler, C.B., Stewart, G.W.: An algorithm for generalized matrix eigenvalue problems. SIAM J. Numer. Anal. 10, 241–256 (1973)
19. Polizzi, E.: Density-matrix-based algorithm for solving eigenvalue problems. Phys. Rev. B 79, 115112 (2009)
20. Ruhe, A.: Rational Krylov: a practical algorithm for large sparse nonsymmetric matrix pencils. SIAM J. Sci. Comput. 19, 1535–1551 (1998)
21. Saad, Y.: Numerical Methods for Large Eigenvalue Problems. SIAM, Philadelphia (2011)
22. Sakurai, T., Futamura, Y., Tadano, H.: Efficient parameter estimation and implementation of a contour integral-based eigensolver. J. Algorithms Comput. Technol. 7, 249–269 (2013)
23. Sakurai, T., Sugiura, H.: A projection method for generalized eigenvalue problems using numerical integration. J. Comput. Appl. Math. 159, 119–128 (2003)
24. Sakurai, T., Tadano, H.: CIRR: a Rayleigh–Ritz type method with contour integral for generalized eigenvalue problems. Hokkaido Math. J. 36, 745–757 (2007)
25. Sontag, E.D.: Mathematical Control Theory: Deterministic Finite Dimensional Systems, 2nd edn. Springer, New York (1998)
26. Stewart, G.W.: Matrix Algorithms, Vol. II, Eigensystems. SIAM, Philadelphia (2001)
27. Tang, P., Polizzi, E.: FEAST as a subspace iteration eigensolver accelerated by approximate spectral projection. SIAM J. Matrix Anal. Appl. 35, 354–390 (2014)
28. von Luxburg, U.: A tutorial on spectral clustering. Stat. Comput. 17, 395–416 (2007)
29. Yin, G., Chan, R., Yueng, M.-C.: A FEAST algorithm with oblique projection for generalized eigenvalue problems. Numer. Linear Algebra Appl. 24, e2092 (2017)