SLEPc Technical Report **STR-11**
Available at http://slepc.upv.es

# Contour Integral Spectrum Slicing Method in SLEPc

Y. Maeda
T. Sakurai
J. E. Roman

Last update: March, 2016 (SLEPc 3.7)
Previous updates: –

## Contents

## 1 Introduction

The Contour Integral Spectrum Slicing method (CISS) is an eigensolver based on the Sakurai-Suigura method [Sakurai and Sugiura, 2003] in the SLEPc library. Users can compute the eigenvalues inside a specified region on the complex plane and corresponding eigenvectors using CISS. CISS can be used for generalized eigenvalue problems such as $A\boldsymbol{x} = \lambda B\boldsymbol{x}$ (`EPSCISS`) and nonlinear eigenvalue problems such as $F(\lambda)\boldsymbol{x} = \mathbf{0}$ (`NEPCISS`). In `EPSCISS` and `NEPCISS`, PETSc's complex scalars are used. However, users can utilize `EPSCISS` in real arithmetic by building the real version of PETSc when matrices $A$ and $B$ are real symmetric and $B$ is positive-definite (i.e., all eigenvalues are real). This report describes the utilization of `EPSCISS` (Section 2), `EPSCISS` with real scalars (Section 3), and `NEPCISS` (Section 4).

Additional details of the algorithms can be found in [Sakurai and Tadano, 2007], [Ikegami *et al.*, 2010], [Ikegami and Sakurai, 2010], [Asakura *et al.*, 2009], and [Maeda *et al.*, 2011].

## 2 Utilization of CISS for generalized eigenvalue problems

### 2.1 Execution with CISS

CISS for the generalized eigenvalue problems is embedded in the `EPS` object of SLEPc (`EPSCISS`). `EPSCISS` solves generalized eigenvalue problems such as $A\boldsymbol{x} = \lambda B\boldsymbol{x}$. Users can select the solver in the `EPS` object by means of the `EPSSetType` function (or `-eps_type` from the command line). When selecting `EPSCISS`, users should also set a region object (`RG`) to the `EPS` object using the `EPSSetRG` function (or `-rg_type` from the command line). The following is an example of running an executable with `EPSCISS`:

```
$ mpirun -np (#process) ./(executable) -eps_type ciss -rg_type ellipse
```

### 2.2 Setting the region object in `EPSCISS`

`EPSCISS` computes the eigenvalues inside a specified region and corresponding eigenvectors. Three types of regions, `RGELLIPSE`, `RGINTERVAL`, and `RGRING`, can be used in `EPSCISS`.

**RGELLIPSE** In `RGELLIPSE`, the region is defined as an ellipse on the complex plane. To indicate the position and shape of the ellipse, the center, radius, and vertical scale (vscale) are set with the input arguments of the `RGEllipseSetParameters` function. The default values of the center, radius, and vertical scale are 0.0, 1.0, and 1.0, respectively (defining a precise circle). Figure 1 illustrates the region and parameters of `RGELLIPSE`. A complex value of the center can be provided in the command line as [±][value] [±][value `i`], with no spaces (e.g., `-rg_ellipse_center 1.0+2.0i`). In `RGELLIPSE`, the closed Jordan curve $\Gamma$ is the boundary of the region. The detail of $\Gamma$ is described in Section 2.4.1.

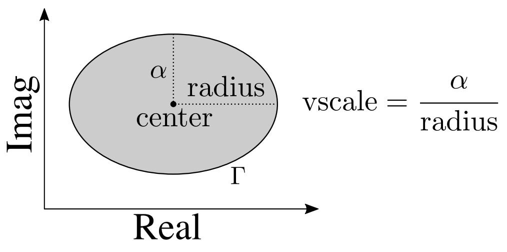

*Figure 1:* `RGELLIPSE` and parameters

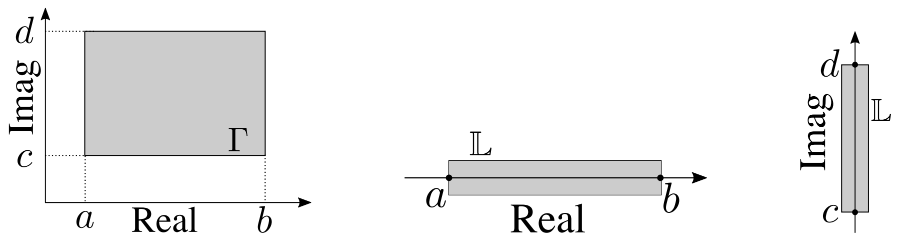

*Figure 2:* `RGINTERVAL` and parameters for the general case (left), for an interval on the real axis (center), and for an interval on the imaginary axis (right)

**RGINTERVAL** In `RGINTERVAL`, the region is defined as a rectangle $[a, b] \times [c, d]$ on the complex plane. To indicate the position of the rectangle, $a, b, c$, and $d$ are set with the input arguments of the `RGIntervalSetEndpoints` function. Figure 2 illustrates the region and parameters of `RGINTERVAL`. In `EPSCISS`, the eigenvalues (and corresponding eigenvectors) inside the gray region are computed. In `RGINTERVAL`, the closed Jordan curve $\Gamma$ is the boundary of the region, as shown in Figure 2. When $c = d = 0$, `EPSCISS` computes the eigenvalues (and corresponding eigenvectors) inside the interval $[a, b]$ on the real axis. When $a = b = 0$, `EPSCISS` computes the eigenvalues (and corresponding eigenvectors) inside the interval $[c, d]$ on the imaginary axis. Figure 2 also illustrates the region of `RGINTERVAL` and the parameters for the particular cases $c = d = 0$ and $a = b = 0$. The quadrature points of `EPSCISS` are set on segment $\mathbb{L}$ in these cases. The detail of the quadrature points is described in Section 2.4.1.

**RGRING** In `RGRING`, the region is defined as a partial ring on the complex plane. To indicate the position and shape of the ring, the parameters for the center, radius, vertical scale (vscale), start angle, end angle, and width are set with the input arguments of the `RGRingSetParameters` function. The angles are provided in turns, i.e., a value ranging from 0.0 to 1.0 (1.0 means the full circumference). The default values of the center, radius, and vertical scale are $0.0, 1.0$, and 1.0, respectively, and the default values of start_ang, end_ang, and width are $0.0, 1.0$, and 0.5, respectively. Figure 3 illustrates the region and the parameters of `RGRING`. In `EPSCISS`, the eigenvalues inside the gray region and the corresponding eigenvectors are computed. In `RGRING`, the closed Jordan curve $\Gamma$ is the boundary of the region.

## 2.3 Setting parameters for `EPSCISS`

Users can set parameters in `EPSCISS` using the following 6 functions:

- `EPSCISSSetSizes`
- `EPSCISSSetUseST`
- `EPSCISSSetQuadRule`
- `EPSCISSSetThreshold`

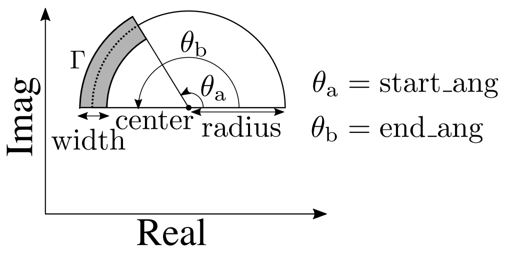

*Figure 3:* `RGRING` and parameters

- `EPSCISSSetRefinement`
- `EPSCISSSetExtraction`

The components of these functions indicate the input parameters for fine-tuning `EPSCISS` and are described in the following subsections.

## 2.4 Input parameters in `EPSCISS`

Tables 1–6 show the input parameters of `EPSCISSSetSizes`, `EPSCISSSetUseST`, `EPSCISSSetQuadRule`, `EPSCISSSetThreshold`, `EPSCISSSetRefinement` and `EPSCISSSetExtraction`, respectively.

In `EPSCISS`, the solutions of several linear systems are required. To solve linear systems, preconditioner (`PC`) objects and Krylov Subspace Methods (`KSP`) objects in PETSc are used in `EPSCISS`. Users can select `PC` objects and `KSP` objects as shown in Table 7.

In the following sections, the detail of the above parameters is discussed by describing the algorithms of `EPSCISS`.

### 2.4.1 `ip`, `bs`, `ms`

To compute the eigenvalues (and corresponding eigenvectors) inside a specified region, a two-step procedure is used in `EPSCISS`. The first step is to construct the subspace filtered for eigenvectors, and the second step is to extract the eigenvalues (and corresponding eigenvectors) inside the closed Jordan curve. The parameters `ip`, `bs`, and `ms` are used in the procedure for constructing the subspace.

Let $\Gamma$ be a positively oriented closed Jordan curve on the complex plane, and let $S_k,\ k = 0, 1, \ldots, M-1$ be $n \times L$ matrices which are determined through the contour integration,

$$S_k = \frac{1}{2\pi \mathrm{i}} \oint_\Gamma z^k \left(zB - A\right)^{-1} BV \mathrm{d}z, \ \text{ for } \ k = 0, 1, \ldots, M-1, \tag{1}$$

where $zB - A$ is a regular matrix pencil on $z \in \Gamma$, $M$ is the moment size, and $V$ is an $n \times L$ matrix whose column vectors are linearly independent.

*Table 1:* Input parameters of `EPSCISSSetSizes`

| Input | Description | Condition | Default | Command line option |
|---|---|---|---|---|
| ip | Number of integration points ($N$) | ip $> 0$ and is an even value | 32 | `-eps_ciss_integration_points` |
| bs | Block size ($L$) | bs $> 0$ | 16 | `-eps_ciss_blocksize` |
| ms | Moment size ($M$) | $0 <$ ms $\leq$ ip | 8 | `-eps_ciss_moments` |
| npart | Number of partitions | npart $\geq 1$ | 1 | `-eps_ciss_partitions` |
| bsmax | Maximum block size | bsmax $\geq$ bs | 128 | `-eps_ciss_maxblocksize` |
| realmats | $A$ and $B$ are real | {true\|false} | false | `-eps_ciss_realmats` |

*Table 2:* Input parameters of `EPSCISSSetUseST`

| Input | Description | Condition | Default | Command line option |
|---|---|---|---|---|
| usest | Use the `ST` object or not | {true\|false} | true | `-eps_ciss_usest` |

*Table 3:* Input parameters of `EPSCISSSetQuadRule`

| Input | Description | Condition | Command line option |
|---|---|---|---|
| quad | Quadrature rule | `{EPS_CISS_QUADRULE_TRAPEZOIDAL\| EPS_CISS_QUADRULE_CHEBYSHEV}` | `-eps_ciss_quadrule` |

*Table 4:* Input parameters of `EPSCISSSetThreshold`

| Input | Description | Condition | Default | Command line option |
|---|---|---|---|---|
| delta | Threshold for numerical rank | delta $> 0$ | $10^{-12}$ | `-eps_ciss_delta` |
| spur | Threshold to discard spurious eigenpairs | spur $> 0$ | $10^{-4}$ | `-eps_ciss_spurious_threshold` |

*Table 5:* Input parameters of `EPSCISSSetRefinement`

| Input | Description | Condition | Default | Command line option |
|---|---|---|---|---|
| inner | Maximum refinement iteration (inner loop) | inner $\geq 0$ | 0 | `-eps_ciss_refine_inner` |
| blsize | Maximum refinement iteration to increase block size | blsize $\geq 0$ | 0 | `-eps_ciss_refine_blocksize` |

*Table 6:* Input parameters of `EPSCISSSetExtraction`

| Input | Description | Condition | Command line option |
|---|---|---|---|
| extraction | Extraction technique | `{EPS_CISS_EXTRACTION_RITZ\| EPS_CISS_EXTRACTION_HANKEL}` | `-eps_ciss_extraction` |

*Table 7:* Parameters for solving linear systems

| Parameter | Description | Default | Command line option |
|---|---|---|---|
| `PC` object | `PC` object for solving linear systems | `PCLU` | `-st_pc_type` (usest = true) `-eps_ciss_pc_type` (usest = false) |
| `KSP` object | `KSP` object for solving linear systems | `KSPPREONLY` | `-st_ksp_type` (usest = true) `-eps_ciss_ksp_type` (usest = false) |

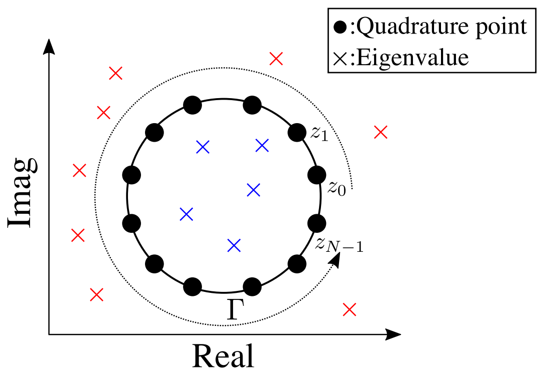

*Figure 4:* Quadrature points and eigenvalues on the complex plane

To compute Eq. (1) numerically, an $N$-points trapezoidal quadrature rule is applied, that is,

$$S_k \approx \hat{S}_k = \sum_{j=0}^{N} w_j \zeta_j^k X_j, \tag{2}$$

where $z_j, \zeta_j$, and $w_j$ are quadrature points, normalized quadrature points, and corresponding weights, respectively, and $X_j,\ j = 0, 1, \ldots, N-1$ are the solutions of linear systems with multiple right-hand side vectors,

$$(z_j B - A)\, X_j = BV, \quad j = 0, 1, \ldots, N-1. \tag{3}$$

`EPSCISS` uses a `PC` object and a `KSP` object to solve these linear systems. Users can select a `PC` object and a `KSP` object, as shown in Table 7. `EPSCISS` constructs $\hat{S} = [\hat{S}_0, \hat{S}_1, \ldots, \hat{S}_{M-1}] \in \mathbb{C}^{n\times(LM)}$. As described in Figure 4, quadrature points are located on $\Gamma$, and `EPSCISS` computes the eigenvalues located inside $\Gamma$ (blue cross marks). The components of $\hat{S}_k$ in the direction of eigenvectors with eigenvalues outside $\Gamma$ are small.

The values of `ip`, `bs`, and `ms` in the input parameters of `EPSCISS` are the same as $N, L$, and $M$ in Eq. (1), respectively. The values of `bs` and `ms` provided by the user should be sufficiently large for the method to work correctly: the value of `bs` should be greater than the maximum multiplicity of eigenvalues inside the region, and `bs` $*$ `ms` should be greater than the number of eigenvalues inside the region.

### 2.4.2 `npart`

`EPSCISS` has the potential for hierarchical parallelism:

**(I)** Each region can be computed independently.

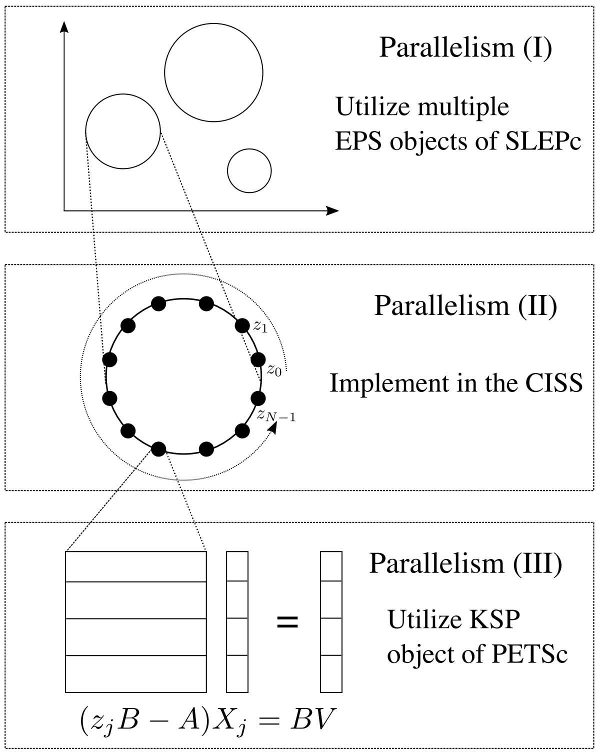

*Figure 5:* Hierarchical parallelism of `EPSCISS`

**(II)** Linear systems at each quadrature point can be solved independently.

**(III)** The linear systems in Eq. (3) can be computed in parallel.

Parallelism (I) can be implemented with multiple `EPS` objects of SLEPc, each of them having different region parameters. Hence, users can manage Parallelism (I) themselves. Parallelism (III) is implemented within the `KSP` object of PETSc. Finally, `EPSCISS` implements Parallelism (II) using MPI communicators in the `EPS` object. The above descriptions are illustrated in Figure 5.

To implement Parallelism (II), `EPSCISS` creates subcommunicators (from groups of the MPI processes participating in the `EPS` object) and then assigns quadrature points to these subcommunicators. Then, a copy of the matrices $A$ and $B$ is created redundantly in each of the subcommunicators, enabling the simultaneous solution of linear systems in each subcommunicator. The results of the linear systems are communicated from the subcommunicators to the parent communicator. The number of subcommunicators can be set using `npart`. An example of the above descriptions is illustrated in Figure 6.

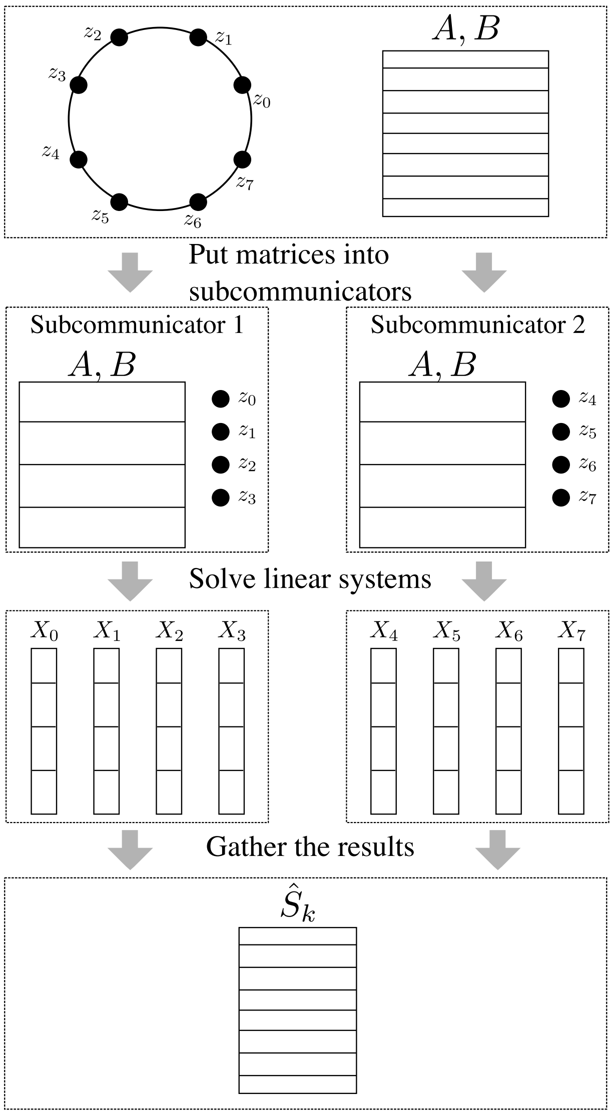

*Figure 6:* Example of Parallelism (II) (#process = 8, `ip` = 8, and `npart` = 2)

### 2.4.3 `bsmax`

To obtain good accuracy of eigenpairs, `EPSCISS` computes a stochastic estimation of the eigenvalue count. The estimated value is used for auto-tuning of the parameter `bs`.

Let $m$ be the number of eigenvalues inside a specified region. The number of eigenvalues inside the region $m$ can be estimated by

$$m \approx \frac{1}{L}\sum_{j=0}^{N-1} w_k \mathrm{tr}(V^{\mathrm{T}} X_j),$$

where $V$ is an $n \times L$ matrix whose elements take the value 1 or $-1$ with equal probability, and $X_j$ is the same as in Eq. (3). It is important to note that the accuracy of the estimation is problem-dependent. The error could become arbitrarily large. To prevent the extreme increase of `bs`, `bsmax` can be used to maintain `bs` $\leq$ `bsmax`. The `bsmax` parameter can also be specified indirectly via the `ncv` parameter in the `EPSSetDimensions` function. If users set the `ncv` parameter, `bsmax` is set to `ncv/ms`. The `ncv` parameter can be set using `-eps_ncv` from the command line.

### 2.4.4 `realmats`

When matrices $A$ and $B$ are real, and `RGELLIPSE` with a real center is used in `EPSCISS`, the results of linear systems $X_j$ and $X_{N-1-j}$, $(j = 0, 1, \ldots, N/2-1)$ are a conjugate pair. Thus, `EPSCISS` can reduce the computational cost by reusing the results of the linear systems. In this case, users specify the `realmats` option to indicate that `EPSCISS` should use this property.

### 2.4.5 `usest`

If `usest` is true, the `ST` object in SLEPc is used for solving linear systems in Eq. (3). Using the `ST` object, the memory usage associated with linear system solution is greatly reduced, since no new `KSP` objects are created internally to `EPSCISS`. It is important to note that Parallelism (II) is not applicable (i.e., `npart` becomes 1) when `usest` is true.

### 2.4.6 `quad`

As explained in Section 2.4.1, `EPSCISS` applies an $N$-points trapezoidal rule as the quadrature rule. Users can select the quadrature rule by setting `quad` in the `EPSCISSSetQuadRule` function. If the "`chebyshev`" option is specified (`EPS_CISS_QUADRULE_CHEBYSHEV`), Chebyshev points are used as quadrature points. It is important to note that the "`chebyshev`" option can be specified when using `RGINTERVAL` with $c = d = 0$, `RGINTERVAL` with $a = b = 0$, or `RGRING`.

### 2.4.7 `delta`

After constructing the subspace, `EPSCISS` extracts the eigenvalues inside the region using the Rayleigh-Ritz approach.

Let the singular value decomposition (SVD) of $\hat{S} = [\hat{S}_0, \hat{S}_1, \ldots, \hat{S}_{M-1}] \in \mathbb{C}^{n\times(LM)}$ be

$$\hat{S} = Q\Sigma W^{\mathrm{H}},$$

where

$$\begin{array}{ll} Q = [\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_{LM}] \in \mathbb{C}^{n\times LM}, & \Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_{LM}), \\ \sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{LM}, & W \in \mathbb{C}^{LM\times LM}. \end{array}$$

To refine the approximation of the eigenpairs, `EPSCISS` omits singular values less than $\delta$ and constructs $\hat{Q} = [\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_K] \in \mathbb{C}^{n\times K}$, where $K \geq m$, and $\sigma_K \geq \sigma_1\delta \geq \sigma_{K+1}$. The parameter `delta` is the same as the threshold $\delta$.

### 2.4.8 `spur`

`EPSCISS` computes all eigenvalues of the small eigenvalue problem

$$(\alpha_i\hat{Q}^{\mathrm{H}}B\hat{Q} - \hat{Q}^{\mathrm{H}}A\hat{Q})\boldsymbol{u}_i = \boldsymbol{0}, \quad \hat{Q}^{\mathrm{H}}A\hat{Q}, \hat{Q}^{\mathrm{H}}B\hat{Q} \in \mathbb{C}^{K\times K},$$

where $\alpha_i$ is the eigenvalue of the matrix pencil $\alpha_i\hat{Q}^{\mathrm{H}}B\hat{Q} - \hat{Q}^{\mathrm{H}}A\hat{Q}$ and $\boldsymbol{u}_i$ is the eigenvector corresponding to $\alpha_i$. Then the eigenvalues of the matrix pencil $A - \lambda B$ are given by $\lambda_i = \alpha_i$, and the corresponding eigenvectors are given by $\boldsymbol{x}_i = \hat{Q}\boldsymbol{u}_i$ for $i = 1, 2, \ldots, K$.

Some of the approximated eigenvalues may lie outside the region. `EPSCISS` keeps the eigenvalue $\lambda_i$ inside the region for $i = 1, 2, \ldots, \tilde{m}$, where $\tilde{m}$ is the number of approximated eigenvalues inside the region, and discards the rest.

Spurious eigenvalues may be contained in the approximated eigenvalues. `EPSCISS` removes the eigenvalues with $\mathrm{res}_i \geq \tau$ inside $\Gamma$ as spurious eigenvalues, where $\mathrm{res}_i$ is the relative residual norm for the eigenpairs $(\lambda_i, \boldsymbol{x}_i)$. The `spur` parameter is the same as the threshold $\tau$.

### 2.4.9 `inner`

To obtain eigenpairs with good accuracy, `EPSCISS` applies three types of recurrence refinements as described below. It is important to note that these recurrence refinements are the aids for improving accuracy. To improve the accuracy of eigenpairs clearly, it is recommended that users change the parameters `ip`, `bs`, and `ms` instead of `inner`, `blsize`, and `maxit`.

Let $\hat{S}_k^{(r)}$ be the $r$th refinement of $\hat{S}_k$,

$$\hat{S}_k^{(r)} = \hat{F}_k\hat{S}_0^{(r-1)}, \quad k = 0, 1, \ldots, M-1,$$

where $\hat{F}_k = \sum_{j=0}^{N} w_j\zeta_j^k(z_jB - A)^{-1}B$. When $r > 0$, the components of $\hat{S}_k^{(r)}$ in the direction of eigenvectors with eigenvalues outside $\Gamma$ are smaller than $\hat{S}_k$. `EPSCISS` constructs $\hat{S} = [\hat{S}_0^{(r)}, \hat{S}_1^{(r)}, \ldots \hat{S}_{M-1}^{(r)}]$ and extracts eigenpairs. The refinement is terminated if the smallest singular value of $\hat{S}^{(r)}$ becomes sufficiently small with a threshold of $\delta$ or the iteration number $r$ becomes `inner`.

### 2.4.10 Number of outer iterations

In case some residuals of the approximated eigenpairs are not small enough for a given tolerance, `EPSCISS` can improve the resulting approximate eigenpairs by setting $V$ as

$$V = [\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_{\tilde{m}}]C$$

where $C \in \mathbb{R}^{\tilde{m}\times L}$, the elements of which are given by random numbers, and $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_{\tilde{m}}$ are the approximated eigenvectors inside the region. After setting $V$, `EPSCISS` reconstructs $\hat{S}_k$ and extracts eigenpairs again. The refinement is terminated if the largest residual of the approximated eigenpairs becomes sufficiently small with a given tolerance `tol` in the `EPSSetTolerances` function. The refinement is also terminated if the iteration number becomes `maxit` in the `EPSSetTolerances` function. The `tol` and `maxit` parameters can be set using `-eps_tol` and `-eps_max_it` from the command line.

### 2.4.11 `blsize`

Section 2.4.3 describes how `EPSCISS` may increase the value of $L$ by estimating the number of eigenvalues inside $\Gamma$. `EPSCISS` can also perform additional increasing of $L$ iteratively, as follows.

Let $H \in \mathbb{C}^{LM\times LM}$ be the block Hankel matrix,

$$H = \begin{pmatrix} T_0 & T_1 & \dots & T_{M-1} \\ T_1 & T_2 & \dots & T_M \\ \vdots & \vdots & \ddots & \vdots \\ T_{M-1} & T_M & \dots & T_{2M-2} \end{pmatrix},$$

where $T_k = V^{\mathrm{H}} S_k \in \mathbb{C}^{L\times L}$. The smallest singular value of $H$ becomes small when $LM$ is greater than the number of eigenvalues inside $\Gamma$. If the smallest singular value of $H$ is not sufficiently small, `EPSCISS` increases $L$ and constructs $H$ iteratively. The refinement is terminated if the smallest singular value of $H$ becomes sufficiently small with a threshold of $\delta$, or the iteration number becomes `blsize`.

The flowchart of the three types of recurrence refinement is shown in Figure 7.

### 2.4.12 `extraction`

After constructing the subspace, `EPSCISS` extracts the eigenvalues inside the region. In Section 2.4.7, `EPSCISS` uses the Rayleigh-Ritz approach for extracting eigenvalues. `EPSCISS` can also extract the eigenvalues inside the region using the block Hankel matrix. Users can specify the extraction technique using `extraction` in the `EPSCISSSetExtraction` function. The default value of `extraction` is "`ritz`" (`EPS_CISS_EXTRACTION_RITZ`). If the "`hankel`" option is specified (`EPS_CISS_EXTRACTION_HANKEL`), then a block Hankel matrix is used as the extraction technique. This section describes the extraction technique using the block Hankel matrix.

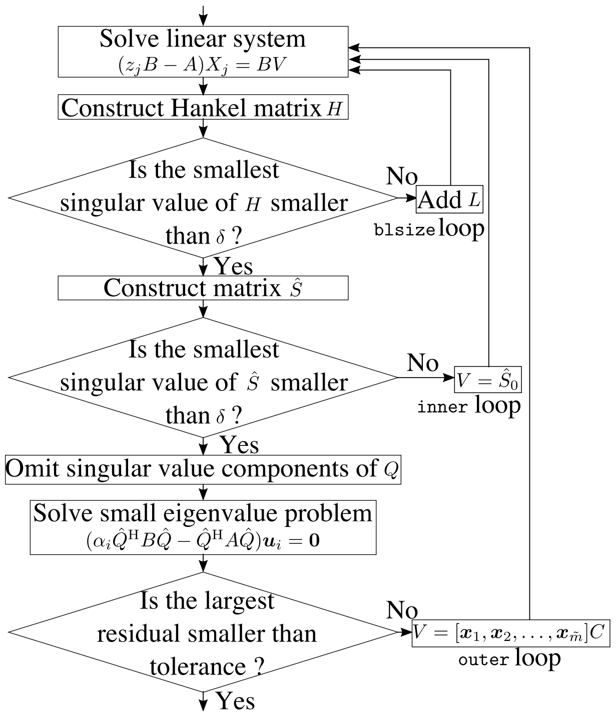
Solve linear system
$(z_jB - A)X_j = BV$
Construct Hankel matrix $H$
Is the smallest singular value of $H$ smaller than $\delta$ ?
No
Add $L$
blsize loop
Yes
Construct matrix $\hat{S}$
Is the smallest singular value of $\hat{S}$ smaller than $\delta$ ?
No
$V = \hat{S}_0$
inner loop
Yes
Omit singular value components of $Q$
Solve small eigenvalue problem
$(\alpha_i\hat{Q}^{\mathrm{H}}B\hat{Q} - \hat{Q}^{\mathrm{H}}A\hat{Q})\boldsymbol{u}_i = \boldsymbol{0}$
Is the largest residual smaller than tolerance ?
No
$V = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{\tilde{m}}]C$
outer loop
Yes

*Figure 7:* The recurrence refinement part of the flowchart of the EPSCISS algorithm using the Rayleigh-Ritz approach

Let $H, H^< \in \mathbb{C}^{LM \times LM}$ be the block Hankel matrices,

$$H = \begin{pmatrix} T_0 & T_1 & \ldots & T_{M-1} \\ T_1 & T_2 & \ldots & T_M \\ \vdots & \vdots & \ddots & \vdots \\ T_{M-1} & T_M & \ldots & T_{2M-2} \end{pmatrix}, \quad H^< = \begin{pmatrix} T_1 & T_2 & \ldots & T_M \\ T_2 & T_3 & \ldots & T_{M+1} \\ \vdots & \vdots & \ddots & \vdots \\ T_M & T_{M+1} & \ldots & T_{2M-1} \end{pmatrix},$$

where $T_k = V^{\mathrm{H}}S_k \in \mathbb{C}^{L \times L}$, and let the singular values of $H$ be

$$\sigma_1, \sigma_2, \ldots, \sigma_{LM}, \quad \sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{LM}.$$

To refine the approximation of eigenpairs, `EPSCISS` omits singular values less than $\delta$ and constructs $\hat{H} = H(1:K, 1:K)$ and $\hat{H}^< = H^<(1:K, 1:K)$, where $K$ is greater than the number of eigenvalues inside $\Gamma$ and $\sigma_K \geq \sigma_1\delta \geq \sigma_{K+1}$. The parameter `delta` is the same as threshold $\delta$ in Section 2.4.7.

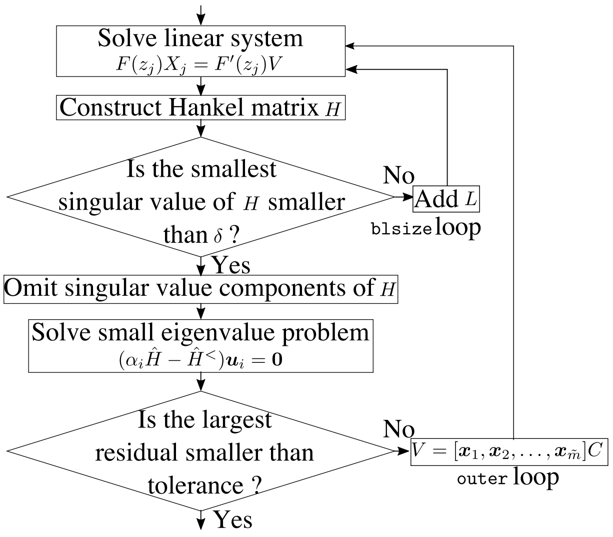
Solve linear system
$F(z_j)X_j = F'(z_j)V$
Construct Hankel matrix $H$
Is the smallest singular value of $H$ smaller than $\delta$ ?
No
Add $L$
blsize loop
Yes
Omit singular value components of $H$
Solve small eigenvalue problem
$(\alpha_i\hat{H} - \hat{H}^{<})\boldsymbol{u}_i = \boldsymbol{0}$
Is the largest residual smaller than tolerance ?
No
$V = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{\tilde{m}}]C$
outer loop
Yes

*Figure 8:* The recurrence refinement part of the flowchart of the `EPSCISS` algorithm using a block Hankel matrix

`EPSCISS` computes all eigenvalues of the small eigenvalue problem

$$(\alpha_i\hat{H} - \hat{H}^{<})\boldsymbol{u}_i = \boldsymbol{0}, \quad \hat{H}, \hat{H}^{<} \in \mathbb{C}^{K\times K},$$

where $\alpha_i$ is the eigenvalue of the matrix pencil $\alpha_i\hat{H} - \hat{H}^{<}$ and $\boldsymbol{u}_i$ is the eigenvector corresponding to $\alpha_i$. Then the eigenvalues of the matrix pencil $A - \lambda B$ are given by $\lambda_i = \alpha_i$, and the corresponding eigenvectors are given by $\boldsymbol{x}_i = \hat{S}(:, 1 : K)\boldsymbol{u}_i$ for $i = 1, 2, \ldots, K$.

Some of the approximated eigenvalues may lie outside the region. `EPSCISS` keeps the eigenvalue $\lambda_i$ inside the region for $i = 1, 2, \ldots, \tilde{m}$, where $\tilde{m}$ is the number of approximated eigenvalues inside the region, and discards the rest.

Spurious eigenvalues may be contained in the approximated eigenvalues. `EPSCISS` removes the eigenvalues with $\text{res}_i \geq \tau$ inside $\Gamma$ as spurious eigenvalues, where $\text{res}_i$ is a relative residual for the eigenpairs $(\lambda_i, \boldsymbol{x}_i)$. `spur` is the same as threshold $\tau$ in Section 2.4.8.

To obtain good accuracy of eigenpairs, `EPSCISS` using a block Hankel matrix applies two types of recurrence refinement, as described in Section 2.4.10 and Section 2.4.11. The inner loop does not apply in this case because $\hat{S}$ is not constructed in `EPSCISS` using a block Hankel matrix. The flowchart of the two types of recurrence refinement is shown in Figure 8. It is important to note that these recurrence refinements are aids for improving good accuracy. To improve the accuracy properly, it is recommended that users change the parameters `ip`, `bs`, and `ms` instead of `maxit`, and `blsize`.

## 3 Utilization of CISS for linear problems with real scalars

The description of Section 2 assumes that users should build the complex version of PETSc for utilizing `EPSCISS`. When matrices $A$ and $B$ are real symmetric and $B$ is positive-definite (i.e., all eigenvalues are real), users can also utilize `EPSCISS` by building the real version of PETSc.

### 3.1 Setting the region object in real-type-`EPSCISS`

In real-type-`EPSCISS`, two types of regions can be used: `RGINTERVAL` and `RGELLIPSE`.

**RGINTERVAL** In `RGINTERVAL`, the region is defined as an interval $[a, b]$ on the real axis. To indicate the position, $a$ and $b$ are set with the input arguments of the `RGIntervalSetEndpoints` function. The complex values $c$ and $d$ in the `RGIntervalSetEndpoints` function are not used in this case. Figure 9 illustrates the region and the parameters of `RGINTERVAL`. In real-type-`EPSCISS`, all eigenvalues (and the corresponding eigenvectors) inside the interval $[a, b]$ are computed. The quadrature points of `EPSCISS` are set on the interval $[a, b]$. The detail of the quadrature points is described in Sections 2.4.1 and 2.4.6.

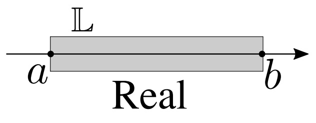

*Figure 9:* `RGINTERVAL` and parameters in real-type-`EPSCISS`

**RGELLIPSE** In `RGELLIPSE`, the region is defined as an ellipse on the complex plane. To indicate the position, the center and radius are set with the input arguments of the `RGEllipseSetParameters` function. The vscale parameter in the `RGEllipseSetParameters` function is not used in real-type-`EPSCISS`. The default values of the center and radius are 0.0 and 1.0, respectively. Figure 10 illustrates the region and the parameters of `RGELLIPSE`. In real-type-`EPSCISS`, all eigenvalues (and the corresponding eigenvectors) inside the interval [center − radius, center + radius] are computed. The quadrature points of `EPSCISS` are set on the interval [center − radius, center + radius]. The detail of the quadrature points is described in Sections 2.4.1 and 2.4.6.

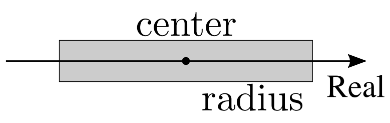

*Figure 10:* `RGELLIPSE` and parameters in real-type-`EPSCISS`

### 3.2 Setting parameters for real-type-EPSCISS

Users can set parameters in real-type-`EPSCISS` in the same way as that in complex-type-`EPSCISS` in Section 2.3. However, the parameter `realmats` is not used in real-type-`EPSCISS`.

## 4 Utilization of CISS for nonlinear eigenvalue problems

CISS for nonlinear eigenvalue problems is embedded in the `NEP` object of SLEPc (`NEPCISS`). `NEPCISS` solves nonlinear eigenvalue problems such as $F(\lambda)\boldsymbol{x} = \boldsymbol{0}$. Users can select the solver in the `NEP` object using the `NEPSetType` function (or `-nep_type` from the command line). When selecting `NEPCISS`, users should also set a region object (`RG`) to the `NEP` object using the `NEPSetRG` function (or `-rg_type` from the command line). The following is one example of running an executable with `NEPCISS`:

```
$ mpirun -np (#process) ./(executable) -nep_type ciss -rg_type ellipse
```

### 4.1 Setting the region object in NEPCISS

`NEPCISS` computes all eigenvalues (and corresponding eigenvectors) inside a specified region. Two types of regions can be used in `NEPCISS`: `RGELLIPSE` and `RGINTERVAL`. The parameters of `RGELLIPSE` and `RGINTERVAL` are the same as the parameters described in Section 2.2.

### 4.2 Setting parameters for NEPCISS

Users can set parameters in `NEPCISS` using the following three functions:

- `NEPCISSSetSizes`
- `NEPCISSSetThreshold`
- `NEPCISSSetRefinement`

The components of these functions indicate the input parameters for fine-tuning `NEPCISS`. Tables 8, 9, 10 show the input parameters of `NEPCISSSetSizes`, `NEPCISSSetThreshold` and `NEPCISSSetRefinement`, respectively.

All parameters of `NEPCISS` are the same as the parameters of `EPSCISS`, as described in Section 2.3. In `NEPCISS`, the block Hankel matrix is used as an extraction technique, as described in Section 2.4.12. The Rayleigh-Ritz approach cannot be used with `NEPCISS`. The difference in the algorithms between `EPSCISS` and `NEPCISS` is the procedure for constructing the subspace. In the procedure for constructing the subspace, `NEPCISS` solves the following linear systems with multiple right-hand side vectors instead of Eq. (3),

$$F(z_j)X_j = F'(z_j)V, \quad j = 0, 1, \dots, N-1,$$

*Table 8:* Input parameters of `NEPCISSSetSizes`

| Input | Description | Condition | Default | Command line option |
|---|---|---|---|---|
| ip | Number of integration points | ip $> 0$ and is an even value | 32 | `-nep_ciss_integration_points` |
| bs | Block size | bs $> 0$ | 16 | `-nep_ciss_blocksize` |
| ms | Moments size | $0 <$ ms $\leq$ ip | 8 | `-nep_ciss_moments` |
| npart | Number of partitions | npart $\geq 1$ | 1 | `-nep_ciss_partitions` |
| bsmax | Maximum block size | bsmax $\geq$ bs | 128 | `-nep_ciss_maxblocksize` |
| realmats | $F(z)$ is real for real $z$ | {true\|false} | false | `-nep_ciss_realmats` |

*Table 9:* Input parameters of `NEPCISSSetThreshold`

| Input | Description | Condition | Default | Command line option |
|---|---|---|---|---|
| delta | Threshold for numerical rank | delta $> 0$ | $10^{-12}$ | `-nep_ciss_delta` |
| spur | Threshold to discard spurious eigenpairs | spur $> 0$ | $10^{-4}$ | `-nep_ciss_spurious_threshold` |

*Table 10:* Input parameters of `NEPCISSSetRefinement`

| Input | Description | Condition | Default | Command line option |
|---|---|---|---|---|
| inner | Maximum refinement iteration (inner loop) | inner $\geq 0$ | 0 | `-nep_ciss_refine_inner` |
| blsize | Maximum refinement iteration to increase block size | blsize $\geq 0$ | 0 | `-nep_ciss_refine_blocksize` |

where $F(z)$ is the matrix function that satisfies $F(\lambda)\boldsymbol{x} = \mathbf{0}$, and

$$F'(z_j) = \left.\frac{\mathrm{d}F(z)}{\mathrm{d}z}\right|_{z=z_j}.$$

Users set the matrix function $F(z)$ and $F'(z)$ using `NEPSetSplitOperator` or `NEPSetFunction` and `NEPSetJacobian` functions. To solve linear systems, `PC` objects and `KSP` objects in PETSc are used in `NEPCISS`. Users can select `PC` objects and `KSP` objects as shown in Table 7, with the prefix `-nep_ciss` instead of `-eps_ciss`.

## References

Asakura, J., T. Sakurai, H. Tadano, T. Ikegami, and K. Kimura (2009). A numerical method for nonlinear eigenvalue problems using contour integrals. *JSIAM Letters*, 1:52–55.

Ikegami, T. and T. Sakurai (2010). Contour integral eigensolver for non-Hermitian systems: a Rayleigh-Ritz-type approach. *Taiwan. J. Math.*, 14(3A):825–837.

Ikegami, T., T. Sakurai, and U. Nagashima (2010). A filter diagonalization for generalized eigenvalue problems based on the Sakurai-Sugiura projection method. *J. Comput. Appl. Math.*, 233(8):1927–1936.

Maeda, Y., Y. Futamura, and T. Sakurai (2011). Stochastic estimation method of eigenvalue density for nonlinear eigenvalue problem on the complex plane. *JSIAM Letters*, 3:61–64.

Sakurai, T. and H. Sugiura (2003). A projection method for generalized eigenvalue problems using numerical integration. *J. Comput. Appl. Math.*, 159(1):119–128.

Sakurai, T. and H. Tadano (2007). CIRR: a Rayleigh-Ritz type method with contour integral for generalized eigenvalue problems. *Hokkaido Math. J.*, 36(4):745–757.