Martin Galgon, Lukas Krämer, Bruno Lang

Counting eigenvalues and improving the integration in the FEAST algorithm

October 2012

http://www-ai.math.uni-wuppertal.de

Counting eigenvalues and improving the integration in the FEAST algorithm

Martin Galgon[a] and Lukas Krämer[a]*and Bruno Lang[a]

[a]Fachbereich C – Mathematik und Naturwissenschaften, Bergische Universität Wuppertal, 42097 Wuppertal, Germany

5 October 2012

Two techniques for improving the robustness and efficiency of the FEAST eigensolver are presented. The FEAST algorithm relies on a Rayleigh–Ritz procedure and on contour integration to determine eigenpairs such that the eigenvalues are contained in a prescribed region I_λ. In this paper the focus is on the symmetric standard eigenvalue problem $Ax = \lambda x$, with a search interval $I_\lambda \subset \mathbb{R}$. We present and compare four methods for estimating the number of eigenvalues contained in I_λ. These estimators can be used to adjust the size of the search space in FEAST dynamically, or as stand-alone methods for counting eigenvalues. We also take a closer look at the numerical integration. We compare two different integration schemes and introduce the stretching of the interval I_λ, leading to faster convergence and more accurate results.

Keywords: Symmetric eigenvalue problem; Counting eigenvalues; Numerical integration; FEAST algorithm

AMS subject classification: 65D30, 65F15, 65F35

1 Introduction

The FEAST algorithm [9] is a method for finding those eigenpairs (λ, x) of a generalised eigenvalue problem $Ax = \lambda Bx$ where the eigenvalue λ is contained in some prescribed search area I_λ.

A recent analysis [6] showed that FEAST essentially implements a Rayleigh–Ritz process with a particular subspace $\mathcal{U} := \operatorname{span}(U)$, which is obtained via a contour integration,

$$U := \frac{1}{2\pi i} \int_{\mathcal{C}} (zB - A)^{-1} \mathrm{d}z\, Y.$$

*Corresponding author. Email: lkraemer@math.uni-wuppertal.de

Here, $\mathcal{C}$ is a contour in the complex plane containing the search area I_λ, and Y is a matrix of initial vectors with $\widetilde{M}$ columns, where $\widetilde{M}$ is an estimate of the number of eigenvalues lying in I_λ. This number must be provided before starting FEAST.

Although the idea behind FEAST and related methods is appealingly simple, there has not been too much work related to those methods. Roughly a decade ago, Sakurai and Sugiura introduced the first method based on contour integration that was solely dedicated to the solution of eigenvalue problems [12]; see also [11]. At about the same time a method for counting eigenvalues was published [1]. The FEAST method as stated below was introduced by E. Polizzi in 2009 [9]. A recent study [8] describes the application of the method to a problem from computational chemistry. Beyn [2] published a similar method for the solution of nonlinear eigenvalue problems. Integrating the resolvent also appears in the context of matrix functions; see, e. g., [5].

In [6] several issues were identified that may arise when trying to use FEAST as a robust black box solver. In the present article we give hints on how to overcome some of these problems. For simplicity, we restrict ourselves to the (real) symmetric standard eigenvalue problem,

- given a symmetric matrix $A \in \mathbb{R}^{n\times n}$ and an interval $I_\lambda = [\underline{\lambda}, \overline{\lambda}] \subset \mathbb{R}$,
- find all eigenpairs (x_i, λ_i) of A (i. e., $Ax_i = \lambda_i x_i$) such that $\lambda_i \in I_\lambda$. Furthermore we want the eigenvectors to be orthonormal, $x_i^T x_j = \delta_{ij}$.

If not stated otherwise we assume the eigenvalues to be ordered ascendingly, $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$.

The FEAST method for this problem is summarised in Algorithm 1.

Algorithm 1 Skeleton of the FEAST algorithm

Input: An interval $I_\lambda = [\underline{\lambda}, \overline{\lambda}]$ and an estimate $\widetilde{M}$ for the number of eigenvalues in I_λ.
Output: $\hat{M} \leq \widetilde{M}$ eigenpairs in I_λ.

```
1: Choose $Y \in \mathbb{R}^{n\times\widetilde{M}}$ of rank $\widetilde{M}$ and compute
       $$U := \frac{1}{2\pi i}\int_{\mathcal{C}} (zI - A)^{-1} \mathrm{d}z\, Y. \qquad (1)$$
2: Form the Rayleigh quotients $A_U := U^T A\, U$, $B_U := U^T U$.
3: Solve the size-$\widetilde{M}$ generalised eigenvalue problem $A_U\, W = B_U\, W\Lambda$.
4: Compute the approximate Ritz pairs $(\Lambda,\ X := U \cdot W)$.
5: If convergence is not reached then go to Step 1, with $Y := X$.
```

The integral (1) has to be evaluated using numerical quadrature, the basics of which can be found many textbooks; see, e. g., [3, 7]. The main computational effort of the FEAST algorithm lies in the solution of the linear systems $(z_j I - A)^{-1}\, \widetilde{U}_j = Y_j$. Each node z_j of the quadrature rule leads to a system of this form, and these systems have to be solved either separately for each column in Y_j or with a special block method. This is the subject of current work and not discussed in the present article.

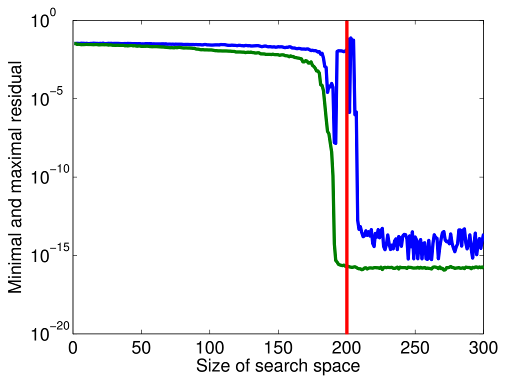

Figure 1: Minimal (lower line) and maximal (upper line) residuals among all eigenpairs with eigenvalue in I_λ. The search interval was chosen such that it contains exactly the 200 lowest eigenpairs, and FEAST was allowed to do at most 20 iterations.

The remainder of the article is structured as follows. In Section 2 we address the problem of estimating the number of eigenvalues that are contained in I_λ and the control of the size of the search space and discuss solution strategies to the problems that were pointed out in [6]. We show that the presented methods are also applicable as stand-alone methods for counting eigenvalues. In Section 3 we treat the numerical integration schemes in more detail and propose a modification that improves convergence for eigenpairs with eigenvalues close to the boundaries of I_λ. Section 4 summarises our findings.

All methods described apply likewise for the generalised eigenvalue problem $Ax = \lambda Bx$ with A Hermitian and B Hermitian positive definite.

2 Counting eigenvalues and controlling the size of the search space

In Algorithm 1, the user must supply, besides the matrix A and the search interval I_λ, a number $\widetilde{M}$ that is an estimate for the number M of eigenvalues lying in I_λ. This number determines the (initial) size of the subspace $\mathcal{U}$, and it should be neither much too large in order to avoid unnecessary computational effort, nor too small because the algorithm cannot compute more than $\widetilde{M}$ eigenpairs.

The phenomenon occurring with an undersized starting base is illustrated by Figure 1. Here, we considered an interval I_λ containing the $M = 200$ lowest eigenpairs of a size-1357 matrix, while varying the value $\widetilde{M}$. As can be seen, $\widetilde{M}$ should be slightly larger than M in order to reach convergence for all sought eigenpairs. If $\widetilde{M} < M$ then possibly even none of the eigenpairs may converge. For a more detailed discussion we refer to [6].

Typically the count of eigenpairs with eigenvalues inside an interval I_λ is not known prior to computation, making it difficult to select a suitable search space size. However, as we will show in this section, good estimates for this number become available quite early during the FEAST algorithm. This makes it possible to check whether the initial base was large enough and to shrink the base if it was too large.

2.1 Estimating the number of eigenvalues in an interval

In the following we discuss four different methods for estimating M, the number of eigenvalues in I_λ.

1. The simplest method consists of counting, in Line 4 of Algorithm 1, those Ritz values of the reduced generalised eigenvalue problem (i. e., diagonal entries of Λ) that are contained in I_λ. This estimate comes almost for free, taking just $\mathcal{O}(\widetilde{M})$ operations.

2. According to [6], the number of eigenvalues in $\mathcal{C}$ is given by the rank of U (and B_U), and thus can be estimated by counting the singular values of U or B_U that exceed a certain threshold. Computing the singular values of B_U requires roughly $8\widetilde{M}^3/3$ operations (Golub–Reinsch SVD, Σ only) [4, Section 5.4.5].

3. An alternative way to reliably determine the rank consists of computing a so-called rank-revealing QR decomposition (rrQR), $X\Pi = QR$, where X is either U or B_U, and Π is a suitable permutation of X's columns. The matrix Q is orthogonal, and R is upper triagonal with diagonal entries r_{ii} ordered descendingly according to their absolute value. Thus the rank of U or B_U can be estimated by counting those entries on the diagonal of R that exceed a certain threshold. An rrQR is less expensive than the SVD, taking $4\widetilde{M}^2 r - 4r^2\widetilde{M} + 4r^3/3$ operations when applied to a matrix B_U with rank r [4, Section 5.4.1].

4. Additional savings are possible by making use of the fact that $\|X\|_F = (\sum_{j=1}^{r} \sigma_j)^{1/2}$ for any matrix X with rank r and singular values σ_j. Furthermore, with exact integration, the matrices U and B_U only have singular values 0 and 1 if Y was orthonormal. (Even with a random starting base, this condition will be reached at least approximately after one FEAST iteration.) It follows that $\|U\|_F = \|B_U\|_F = \sqrt{r}$, and thus (except for the first FEAST iteration) the rank can be obtained by computing the Frobenius norm of U or B_U, requiring approximately $2\widetilde{M}^2$ operations in the latter case.

The behaviour of the different methods is exhibited by three increasingly demanding experiments. In each experiment, an *interval progression* is performed (meaning that a fixed-size search interval I_λ is moved step by step over a known group of eigenvalues), and for each position of the search interval two FEAST iterations are performed and the estimates provided by the above methods are monitored. All experiments where performed with an 8-point Gauß–Legendre integration scheme.

Experiment 2.1. This experiment assesses the methods' ability to correctly detect single eigenvalues that are well separated from the remainder of the spectrum. To this end we chose the eigenvalue $\lambda = \lambda_{1755}$ of the matrix `bcsstkm13`[1] as isolated target. The top picture in Figure 2 shows this eigenvalue as well as the first and the last position of the search interval in the interval progression. The nearest eigenvalue (marked by a "+") is far enough away so that, throughout the progression, I_λ always covers either no eigenvalue or just λ. The step size was chosen such that for two of the positions in the progression the target eigenvalue lies on the left or right, resp., boundary of I_λ. Two iterations of the FEAST algorithm were applied for each interval with a random orthogonal starting base and a search space size of 10.

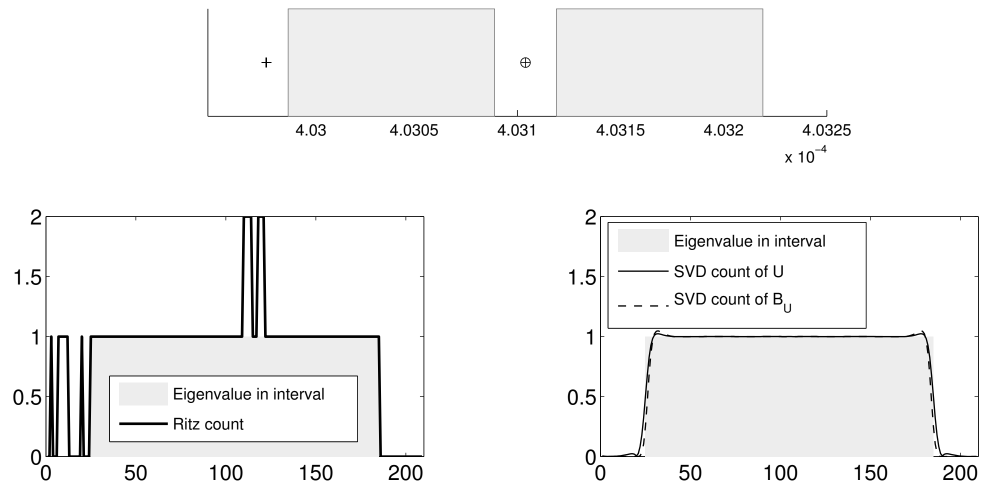

Figure 2: Top: First and last interval of the interval progression over λ (indicated by $\oplus$). Bottom: Ritz value count of the reduced eigenproblem (left) and Frobenius norms of U and B_U (right) for each interval in the progression. The values on the abscissae indicate the number of the interval within the progression. Light gray background implies that the considered eigenvalue is inside the respective interval.

As can be seen in the bottom left picture of Figure 2, the Ritz count tends to overestimate the true number of eigenvalues even in this simple situation. (This behaviour was also confirmed in other experiments.)

By contrast, the three remaining methods yield accurate estimates, at least if the eigenvalue is not too close to the boundaries of I_λ; see the bottom right picture. Only the data for the SVD count are shown, the curves for the Frobenius norm of U and B_U and for the largest entry $|r_{11}|$ of the rrQR coincide with these. (Since all other singular values and diagonal entries of R were considerably smaller, the rank is determined by just σ_1 and $|r_{11}|$.) ■

[1] A modified version of the matrix with the same name from matrix market, see http://http://math.nist.gov/MatrixMarket/. The spectrum now is quite challenging with large clusters of eigenvalues.

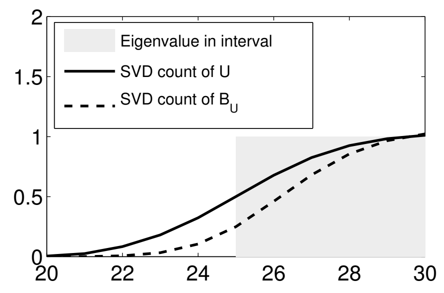

Figure 3: Zoom into Figure 2.

Since counting the Ritz values often gives wrong results even in the case of a single eigenvalue this approach should not be taken if very good estimates are needed. In other situations Ritz counts may prove useful; see the discussion at the end of this subsection.

To get a closer look on the singular values of U and B_U when the interval boundaries are close to the target value, we give in Figure 3 a detail of the right plot in Figure 2.

This behaviour is typical. It reveals that each time when an eigenvalue λ enters (leaves) the search interval, one more singular value σ_λ becomes larger (smaller) than 0.5 for U and 0.25 for B_U. Thus, singular values

$$\sigma(U) > 0.5 \qquad \text{and} \qquad \sigma(B_U) > 0.25, \quad \text{resp.,}$$

correspond to eigenvalues in I_λ, and counting them gives an estimate for the number of eigenvalues in the search interval. Singular values that are very close to these thresholds correspond to eigenvalues at the interval boundaries, whereas singular values close to one come with eigenvalues "well in the interior."

Thus we expect the Frobenius norms of U and B_U to be approximately $\sqrt{M}$ if M eigenvalues are located inside the interval with a certain distance to the interval borders, as is confirmed by the following experiment.

Experiment 2.2. In this experiment an interval I_λ, large enough to cover the first five eigenvalues of A, is used for the interval progression. Starting at the left of the spectrum, the interval is moved to the right until it contains five eigenvalues. The positions are chosen so that every 10 steps another eigenvalue enters the interval and is exactly located at the right interval boundary; thus the step size is constant only between two eigenvalues, i. e., for 10 successive steps. The results are depicted in Figure 4. ■

Furthermore, if the M-th eigenvalue enters the interval then we have

$$\|U\|_F^2 \geq (M-1) + 0.5^2 \qquad \text{and} \qquad \|B_U\|_F^2 \geq (M-1) + 0.25^2.$$

We thus can give an estimate for the eigenvalue count as

$$M_{est} = \left\lceil \|U\|_F^2 - 0.5^2 \right\rceil \qquad \text{resp.} \qquad M_{est} = \left\lceil \|B_U\|_F^2 - 0.25^2 \right\rceil. \tag{2}$$

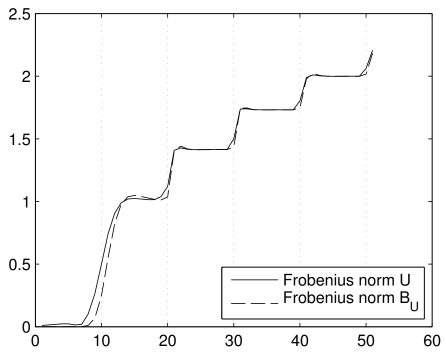

Figure 4: Frobenius norms of U and B_U during the interval progression of Experiment 2.2. The values on the abscissa indicate the number of the interval in the progression.

While working well in most circumstances, the estimate (2) may fail in the presence of tight clusters of eigenvalues because, e. g., $\|U\|_F \approx 1$ may come either from one eigenvalue in the interior of I_λ (as suggested by $M_{est} = 1$) or from four eigenvalues on the boundary. This problem indeed led to wrong results in some of our experiments, under-estimating the total number of eigenvalues while over-estimating the number of those lying "far enough" in the interior. From the above, one would expect the norm-based method to under-estimate the number of eigenvalues. The following experiment shows that this expectation cannot be relied on. This is due to the fact that singular values corresponding to eigenvalues within (outside of) I_λ are not exactly 1 (0, resp.) if the integration is done numerically, cf. Section 3.

Experiment 2.3. To address the general case, where intervals might contain an arbitrary number of eigenvalues, we set up an interval progression where the interval width was chosen arbitrarily but fixed for all intervals in the progression. Further, the initial position and (constant) step size were chosen arbitrarily. In particular all parameters were chosen in such a way that no special behaviour could be expected. We then tested all four methods with that progression. This was done with several matrices and configurations of interval progressions. In Figure 5 we present the results for one run with the matrix $A =$ `bsstkm13`, which is exceptionally demanding due to the structure of its spectrum (very strongly clustered eigenvalues, eigenvalues of small absolute value).

The SVD and rrQR estimations were only performed for B_U, and the results for rrQR are not shown because they are identical to those for SVD. These two methods provided excellent estimations while the Ritz value count gave a larger number than the correct one. The Frobenius norms of U and B_U also deviated from the exact value, but not in a predictable way. ■

The behaviour shown in Figure 5 is typical. In most of our experiments, the estimators based on the rrQR and SVD (of U or B_U) gave correct results up to at most one or two eigenvalues. The advantage of the more expensive SVD and rrQR of U is that they also

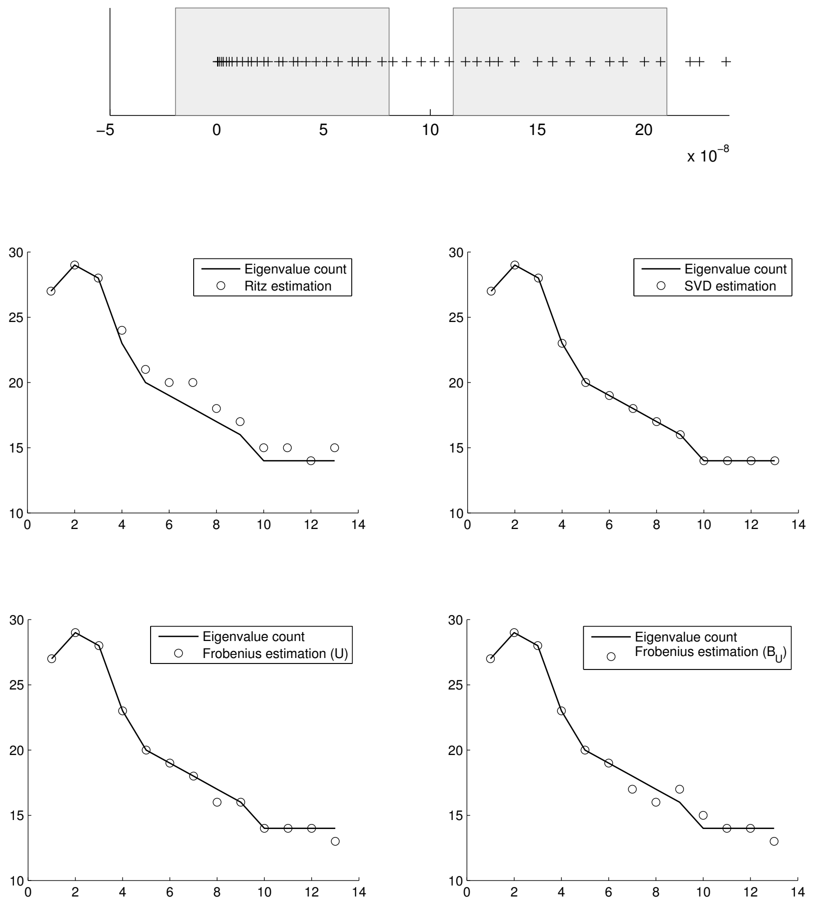

Figure 5: Top: Interval progression for Experiment 2.3. Gray shading indicates the first and last interval of the progression, respectively. Below: Results. For each interval we plotted the exact number of eigenvalues in it (solid lines) and the number estimated with the method indicated in the legend.

provide approximate bases for the spaces spanned by the eigenvectors corresponding to eigenvalues in I_λ. These bases can be used in subsequent iterations or as starting base in a restart of the complete algorithm; see Section 2.2. The Ritz estimator always counted a number of Ritz values that was larger or equal than the actual number of eigenvalues in the interval. Given the negligible cost of this method, its results are not too bad, and they may be sufficient for certain purposes; see Section 2.2. The behaviour of the Frobenius estimators is quite unpredictable when eigenvalues cluster at the interval boundaries.

Note that over-estimating the number of eigenvalues during the FEAST algorithm is not a real problem. This is because the search space should be kept somewhat larger than the true eigenspace, and Ritz pairs with Ritz values outside I_λ can be removed once all Ritz pairs have converged.

In addition, all results can be improved when using higher precision in the numerical integration, i. e., more integration points.

The methods presented can also be used as stand-alone methods for counting eigenvalues. Another integration-based method for counting eigenvalues in a given domain was proposed by Bertrand and Philippe [1].

2.2 Using the estimates to control the size of the search space

If the dimension of the search space is less than M, the number of eigenvalues contained in I_λ, then FEAST will not converge [6]. If, on the other hand, too many vectors are carried through the algorithm then the number of right-hand sides for the linear systems as well as the dimensions of the reduced eigenvalue problem and of other computations are unnecessarily large. The objective therefore is to reduce the size of the search space as far as possible. In practice, a value slightly larger than M gives best convergence [6]. As was shown in the preceding subsection, good estimates for M become available during the second iteration of FEAST.

The most critical situation occurs right at the start of the algorithm. Since the dimension of the initial search space is determined by some initial guess $\widetilde{M}$ for M, a space too small may be chosen if $\widetilde{M} < M$. We cannot exclude this possibility, but this situation can at least be detected during the two first iterations using the above estimates. (The Ritz count can be used even in the first iteration. Although slightly inaccurate at this point, it will report an upper bound for the eigenvalue count early in the algorithm.) If the current estimate M_{est} is equal to the initial $\widetilde{M}$ then this strongly hints at $\widetilde{M}$ having been too small. In this case one should increase $\widetilde{M}$ and restart the algorithm. Polizzi's FEAST implementation [10] does this, based on the Ritz count.

If M_{est} is well below $\widetilde{M}$ then the current search space is too large and may be reduced (to a dimension slightly larger than $\widetilde{M}$). To not discard part of the information already computed, one should not remove vectors whose corresponding eigenvalues are contained in I_λ. Using the SVD or rrQR of U allows us to identify such vectors, since these decompositions also provide orthonormal bases of the space spanned by U. The selection of "good" vectors is also possible when the rank of B_U and the Ritz count coincide. In this case one would keep those Ritz vectors belonging to Ritz values in I_λ. In Section 3 we will see that the selection function implies that vectors associated with certain Ritz values outside I_λ may also influence overall convergence and should hence be kept as well.

In addition to shrinking the search space, already converged eigenpairs may be "locked," i. e., no longer be included in the computations. Locking further reduces the number of linear systems to solve and the size of the reduced eigenvalue problem without hindering still missing pairs to converge in later iterations.

2.3 Detection of empty intervals

One method for using multiple processors in the eigenvalue computation consists of subdividing I_λ into subintervals and running FEAST in parallel on these subintervals. Since the initial interval subdivision cannot take the (yet unknown) structure of the spectrum into account, probably some of the subintervals will not contain any eigenvalue. If it is possible to detect such subintervals cheaply then they can be discarded before running the more expensive "full" FEAST algorithm on them. We propose the following method as a preprocessing step.

1. Perform two FEAST iterations with a search space of size ≥ 1.
2. Check if $\sigma(B_U) > 0.25$ for at least one singular value (or apply one of the other methods from Section 2).

This will deliver the correct answer "interval empty/not empty," since the singular values of B_U will always be larger than the threshold as long as one eigenvalue is on the boundary of the subinterval or inside.

Several numerical experiments confirm the reliability of this method. Only in the case that all starting vectors are orthogonal to the eigenvectors belonging to the considered subinterval, this method will fail. For this reason one should not use just a one-dimensional search space; in our experiments $\widetilde{M} = 3$ gave a reasonable compromise between reliability and efficiency.

3 Numerical integration

In this section we analyse the numerical integration that is necessary in Step 1 of Algorithm 1 to some detail. It is clearly one of the most vital aspects in the algorithm concerning numerical robustness and effectiveness. We propose and evaluate a modification yielding improvement with respect to both aspects.

3.1 The selection function

For the numerical integration of some function f defined on a compact interval $[a, b]$, a numerical integration scheme $(\omega_j, t_j)_{j=1:p}$ can be employed, e. g., a Gauß–Legendre scheme [3]. Then the integral of f is approximated as

$$\int_a^b f(t)\mathrm{d}t \approx \sum_{j=1}^{p} \omega_j f(t_j)$$

with weights $\omega_j > 0$ and integration points $t_j \in [a, b]$. To compute U, this is combined with a parametrisation map $\varphi : [a, b] \to \mathcal{C}$, yielding

$$\begin{aligned}
U &= \frac{1}{2\pi i} \int_{\mathcal{C}} (zI - A)^{-1} \mathrm{d}z\, Y \\
&= \frac{1}{2\pi i} \int_a^b (\varphi(t)I - A)^{-1}\, \varphi'(t)\mathrm{d}t\, Y \\
&\approx \frac{1}{2\pi i} \sum_{j=1}^{p} \omega_j\, (\varphi(t_j)I - A)^{-1}\, \varphi'(t_j)\, Y \\
&= \sum_{j=1}^{p} \omega'_j\, (z_j I - A)^{-1}\, Y, \qquad (3)
\end{aligned}$$

where $z_j = \varphi(t_j)$ and $\omega'_j = \frac{1}{2\pi i}\varphi'(t_j)\omega_j$.

Let $\{x_1, \ldots, x_n\}$ be a complete system of orthonormal eigenvectors to the eigenvalues $\lambda_1, \ldots, \lambda_n$. Then

$$(z_j I - A)^{-1} = \sum_{k=1}^{n} \frac{1}{z_j - \lambda_k} \cdot x_k x_k^T,$$

and therefore (3) can be written as

$$U \approx \sum_{k=1}^{n} \left[\sum_{j=1}^{p} \frac{\omega'_j}{z_j - \lambda_k} \right] x_k x_k^T\, Y.$$

The same representation occurs in [8], in the context of non-Hermitian complex eigenproblems stemming from computational chemistry. Therein, Laux calls the number in brackets the *selection function* and interprets it as a function of λ,

$$S(\lambda) := \sum_{j=1}^{p} \frac{\omega'_j}{z_j - \lambda}. \qquad (4)$$

(More precisely, it is considered to be a function of the eigenvalues λ_k.) It only depends on $\mathcal{C}$ and the chosen integration scheme.

If A is real symmetric, as assumed in this work, and if in addition the curve $\mathcal{C}$ is symmetric with respect to the real axis, i. e., $z \in \mathcal{C} \Leftrightarrow \bar{z} \in \mathcal{C}$, then the integral in (1) simplifies to [9]

$$U = \frac{1}{\pi} \int_{\mathcal{C}^+} \mathrm{Im}((zI - A)^{-1})\, \mathrm{d}z\, Y. \qquad (5)$$

Here, $\mathcal{C}^+$ denotes the upper half of the contour. Since we must only integrate over one half of the contour, we can hope to obtain better results with the same amount of work invested. In the following we therefore consider parametrisations of $\mathcal{C}^+$ instead of $\mathcal{C}$.

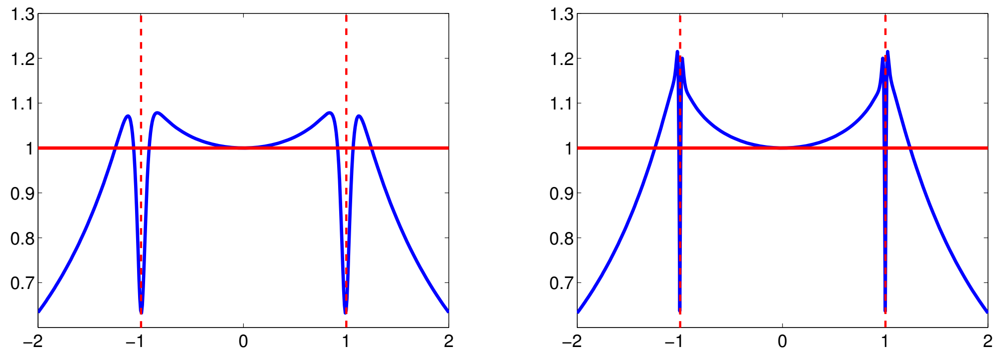

Figure 6: Absolute values of S for midpoint rule (left) and Gauß–Legendre (right), each with 16 integration points.

3.2 Stretching I_λ

In [8] it is also noted that the integration is exact if S is 1 at every eigenvalue inside $\mathcal{C}$ and 0 at all others. Since S is a continuous function at all points $\lambda \neq z_j$, we cannot optimise it in such a way that it acts as a perfect "indicator" (1 inside $\mathcal{C}$, 0 outside). However, we still should seek for a function S, i. e., for an integration contour and integration scheme, such that this property is met as closely as possible.

The integration schemes typically used are Gauß–Legendre and trapezoidal or midpoint type rules. For a general overview on numerical integration, see, e. g., [3]. In the following we use a circle as the contour for the integration. This is not the only choice one can make; see, e. g., [8].

The use of trapezoidal and midpoint rules in combination with a circle contour is motivated by the fact that these schemes converge exponentially for periodic functions [3, Section 2.9]. Gauß–Legendre integration, i. e., Gauß quadrature with weight function 1, is widely used, and its use in FEAST was proposed by Polizzi in [9].

In Figure 6 the values of $|S|$ are depicted for the midpoint and Gauß–Legendre rules. Here, the integral (5) was used, i. e., only the upper half of the circle was taken into account. The curves are produced by taking for each integration point also its conjugate value with the same integration weight. This amounts to integrating over the whole circle $\mathcal{C}$, but over the upper and lower part separately. The relatively high peaks of the curve in the right plot can be explained by the fact that the integration points in Gauß–Legendre are closer to the boundary of the considered interval. Thus φ maps them closer to the real axis, and the denominator in (4) becomes small. This effects becomes more pronounced with more integration points.

The plots in Figure 6 reveal a potential problem with the numerical integration. They show a decrease of $|S|$ to about 0.65 toward the boundaries of I_λ, but still inside the interval. This decrease leads to problems with eigenvalues that lie very close to the boundaries of the interval, in particular, if they are clustered there. Several numerical

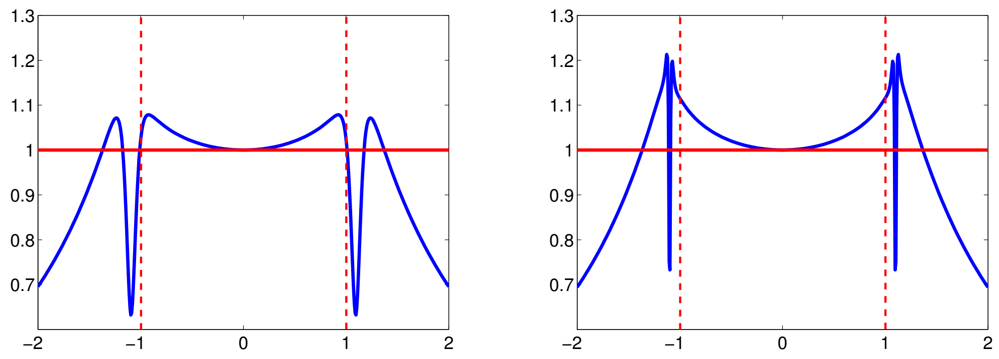

Figure 7: Absolute values of S for midpoint rule (left) and Gauß–Legendre (right), each with 16 integration points. Integration is on $[-1.1, 1.1]$, i. e., $\gamma = 1.1$.

experiments confirmed exactly this expected behaviour. In particular it can happen that eigenpairs with eigenvalue well inside I_λ have already converged while those at the boundary require some additional iterations or even fail to converge.

The remedy for this problem is to stretch the interval in such a way that $|S| \approx 1$ inside I_λ. In practice, this is done by using *one* interval I_λ for searching eigenvalues in *another* (larger) interval $I'_\lambda = [\underline{\lambda}', \overline{\lambda}'] \supset I_\lambda$ for integration.

For simplicity, I'_λ is chosen symmetrically around I_λ. For different integration schemes and different integration curves, suitable stretching factors

$$\gamma := \frac{|\overline{\lambda}' - \underline{\lambda}'|}{|\overline{\lambda} - \underline{\lambda}|}$$

can be determined systematically via experiments. In our experience $\gamma = 1.1$ is adequate for 8 to 16 integration points on $\mathcal{C}^+$. We then have $|S| \geq 1$ inside I_λ and a steep descent to about 0.65 outside. The resulting curves for $|S|$ can be seen in Figure 7.

Stretching the integration interval can improve the convergence of some of the desired eigenpairs, in particular if the eigenvalues are piled around the boundaries of I_λ. We observed that in this case, with the original interval I_λ for integration, the inner eigenvalues (where $|S| \approx 1$) always converge some iterations earlier than the outer ones. In the extreme case, where all eigenvalues lie at the two interval boundaries, stretching the integration interval led to significant savings in terms of iteration numbers. The following experiment illustrates the benefits of stretching.

Experiment 3.1. We took again the size-2003 matrix `bcsstkm13` from Section 2 and chose an interval I_λ containing 295 eigenvalues in four clusters. Two of these clusters are very close to the boundaries of I_λ, the remaining two are well inside. We ran the FEAST algorithm using the midpoint and Gauß–Legendre rule, respectively, for different values for γ. We always used a subspace of dimension 310. We then counted the necessary number of iterations and measured the relative residuals

Table 1: Results of Experiment 3.1. The symbol "—" indicates that the algorithm failed to converge.

γ	**Midpoint**	**Gauß–Legendre**
1.0	3 iterations $\mathsf{orth} = 1.2 \cdot 10^{-13}$ $\mathsf{res} = 1.2 \cdot 10^{-12}$	—
1.1	3 iterations $\mathsf{orth} = 5.7 \cdot 10^{-15}$ $\mathsf{res} = 1.5 \cdot 10^{-13}$	3 iterations $\mathsf{orth} = 1.1 \cdot 10^{-14}$ $\mathsf{res} = 9.5 \cdot 10^{-13}$
1.2	4 iterations $\mathsf{orth} = 6.4 \cdot 10^{-15}$ $\mathsf{res} = 8.2 \cdot 10^{-14}$	4 iterations $\mathsf{orth} = 9.4 \cdot 10^{-14}$ $\mathsf{res} = 8.4 \cdot 10^{-13}$
1.3	2 iterations $\mathsf{orth} = 6.6 \cdot 10^{-15}$ $\mathsf{res} = 8.9 \cdot 10^{-16}$	3 iterations $\mathsf{orth} = 6.6 \cdot 10^{-15}$ $\mathsf{res} = 1.0 \cdot 10^{-12}$
1.4	2 iterations $\mathsf{orth} = 3.5 \cdot 10^{-14}$ $\mathsf{res} = 6.2 \cdot 10^{-15}$	2 iterations $\mathsf{orth} = 6.6 \cdot 10^{-15}$ $\mathsf{res} = 9.8 \cdot 10^{-15}$

$\mathsf{res} = \|AX - X\Lambda\| \,/\, \|A\|$ as well as the maximum deviation from orthogonality,

$$\mathsf{orth} = \max_{\lambda_i, \lambda_j \in I_\lambda,\, i \neq j} \left| x_i^T x_j \right| .$$

The results are given in Table 3.1. They indicate that the midpoint rule was more robust when integrating over the original interval I_λ. Using a slightly larger integration interval I'_λ with $\gamma = 1.1$, both schemes led to convergence of the FEAST algorithm. For values $\gamma > 1.4$, the results in terms of the three indicators given became worse again. The best overall result was obtained by the combination of midpoint rule and $\gamma = 1.3$. ■

3.3 Considering S outside of I_λ

By stretching the interval we can achieve that $|S|$ is not smaller than 1 inside I_λ and also not much larger, but $|S|$ also takes values considerably larger than 0 outside I_λ, such that vectors belonging to unwanted eigenvalues enter the calculation. Laux states [8] "M_0 [$\widetilde{M}$ in our notation] must be as large as the number of convergent eigenvalues within $\mathcal{C}$ and its surrounding fringe." One possibility is then to adapt $\widetilde{M}$ and take the value of $|S|$ into account when determining $\widetilde{M}$ with the techniques from Section 2. One still can delete vectors from the calculation that belong to Ritz values outside I'_λ. This is essentially the Ritz count technique from Section 2, but here the Ritz values in I'_λ and not in I_λ are counted.

4 Conclusion

This paper contributes in two ways to making the FEAST algorithm faster and more robust.

The first contribution are four methods for estimating the number of eigenvalues in a given interval I_λ. The methods are based on counting Ritz values, singular value decomposition, rank-revealing QR decomposition, and the Frobenius norm, respectively. The estimate can be used to check whether the setup of the algorithm, where the user had to supply a number $\widetilde{M}$ of eigenvalues that are presumed to be in I_λ, was appropriate. In the course of the algorithm, the estimates can be used to reduce the dimension of the search space, leading to higher efficiency. The methods based on SVD and rrQR of the small-scale matrix B_U show a good balance of reliability and cost. They can also be used as stand-alone methods for counting eigenvalues. The advantage of the more expensive estimates based on the SVD or rrQR of U is that they also provide approximate bases for the spaces spanned by the eigenvectors corresponding to eigenvalues in I_λ. These can be used in subsequent iterations or as starting base in a restart of the complete algorithm with the estimated number of eigenvalues. For a very rough estimation within the FEAST algorithm, the extremely cheap Ritz count may be sufficient. We further proposed a scheme for the detection of intervals that contain no eigenvalues. This scheme is much cheaper than that one for counting eigenvalues.

The second aspect proposed is taking a closer look at the numerical integration. We evaluated two different integration schemes and introduced the stretching of the interval I_λ, leading to faster convergence and more accurate results.

References

[1] O. Bertrand and B. Philippe. Counting the eigenvalues surrounded by a closed curve. *Sib. Zh. Ind. Mat.*, 4(2):73–94, 2001.

[2] Wolf-Jürgen Beyn. An integral method for solving nonlinear eigenvalue problems. *Linear Algebra Appl.*, 436(10):3839–3863, 2012.

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

[4] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

[5] Nicholas Hale, Nicholas J. Higham, and Lloyd N. Trefethen. Computing A^α, $\log(A)$ and related matrix functions by contour integrals. *SIAM J. Numer. Anal.*, 46(5):2505–2523, 2008.

[6] Lukas Krämer, Edoardo Di Napoli, Martin Galgon, Bruno Lang, and Paolo Bientinesi. Dissecting the FEAST algorithm for generalized eigenproblems. Preprint BUW-IMACM 12/09, http://www.imacm.uni-wuppertal.de/imacm/research/preprints.html, 2012.

[7] Arnold R. Krommer and Christoph W. Ueberhuber. *Computational Integration.* SIAM, Philadelphia, PA, 1998.

[8] S. E. Laux. Solving complex band structure problems with the FEAST eigenvalue algorithm. *Phys. Rev. B*, 86:075103, 2012.

[9] E. Polizzi. Density-matrix-based algorithm for solving eigenvalue problems. *Phys. Rev. B*, 79:115112, 2009.

[10] Eric Polizzi. A high-performance numerical library for solving eigenvalue problems: FEAST solver v2.0 user's guide, 2012. `arXiv:1203.4031v1`.

[11] T. Sakurai, Y. Kodaki, H. Tadano, D. Takahashi, M. Sato, and U. Nagashima. A parallel method for large sparse generalized eigenvalue problems using a GridRPC system. *Future Generation Computer Systems*, 24:613–619, 2008.

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