Flexible subspace iteration with moments for an effective contour integration-based eigensolver

Sarah Huber¹ | Yasunori Futamura² | Martin Galgon¹ | Akira Imakura² | Bruno Lang¹ | Tetsuya Sakurai²

¹Faculty of Mathematics and Natural Sciences, University of Wuppertal, Wuppertal, Germany
²Department of Computer Science, University of Tsukuba, Tsukuba, Japan

Correspondence
Sarah Huber, Faculty of Mathematics and Natural Sciences, University of Wuppertal, Gaussstr. 20, Wuppertal 42119, Germany
Email: shuber@math.uni-wuppertal.de

Funding information
Deutsche Forschungsgemeinschaft, Grant/Award Number: SPP 1648; Japan Society for the Promotion of Science, Grant/Award Numbers: 17K12690, 18H03250, 19K20280

Abstract
Contour integration schemes are a valuable tool for the solution of difficult interior eigenvalue problems. However, the solution of many large linear systems with multiple right hand sides may prove a prohibitive computational expense. The number of right hand sides, and thus, computational cost may be reduced if the projected subspace is created using multiple moments. In this work, we explore heuristics for the choice and application of moments with respect to various other important parameters in a contour integration scheme. We provide evidence for the expected performance, accuracy, and robustness of various schemes, showing that good heuristic choices can provide a scheme featuring good properties in all three of these measures.

KEYWORDS
contour-integral based eigensolver, Sakurai–Sugiura methods, subspace iteration

1 | INTRODUCTION

Contour integration schemes have become popular in recent years as a broadly applicable spectral filtering approach to interior eigenvalue problems. Two of the most well-known of these are Sakurai–Sugiura methods (SSM), the first of which was introduced by Sakurai and Sugiura in 2003,¹ and FEAST, introduced by Polizzi in 2009.² Both SSM³⁻¹¹ and FEAST¹²⁻¹⁷ have been extended and further developed. These methods are in particular applicable to the generalized interior eigenproblem in the positive definite case, that is, to finding the eigenpairs \((\lambda, \mathbf{x})\) of

\[
A\mathbf{x} = B\mathbf{x}\lambda, \quad \lambda \in I_{\lambda} = \left[\underline{\lambda}, \overline{\lambda}\right].
\]  

with hermitian \(A\) and hermitian positive definite \(B\).

The identifying and typically most computationally costly component of contour integration schemes is the evaluation of a contour integral. More precisely, for (1) one chooses a contour \(C\) in the complex plane such that \(C\) contains the eigenvalues in \(I_{\lambda}\), but no other eigenvalue, and then computes approximations to integrals of the form

\[
\frac{1}{2\pi i} \int_{C} z^{k}(zB - A)^{-1}BYdz .
\]  

From these, an approximate subspace, \(U\), for the desired eigenspace, \(X_{\lambda}\), is obtained. In both FEAST and Sakurai–Sugiura Rayleigh–Ritz (SS–RR),³ the type of SSM considered in this work, the approximate eigenvalues and vectors are then
obtained from $U$ with a Rayleigh–Ritz reduction and solving the reduced eigenproblem

$$A_U W = B_U W \Lambda,$$

(3)

where $A_U = U^* AU, B_U = U^* BU$ and $X := UW$.

Solvers use numerical quadrature to approximate the integral (2), acting on a set of initial vectors $Y$ to construct the approximate subspace $U$. Given a quadrature rule with $q$ quadrature nodes $z_j$ and coefficients $\omega_j$, $q$ linear systems of the form $(z_j B - A)^{-1} BY$ with multiple right hand sides must be solved. The defining difference between FEAST and SSM is the use of moments in the contour integrals.

In Sakurai–Sugiura methods, the approximate integral contains a term $z^k$ representing the $k$th moment, leading to the subspace components

$$U_k = \sum_{j=1}^{q_{SSM}} \omega_j z^k_j (z_j B - A)^{-1} BY,$$

which are combined, given a total of $s$ moments, to form the complete approximate subspace as

$$U = [U_0, U_1, U_2, \ldots, U_{s-1}].$$

(4)

In FEAST, only the 0th moment is considered, and $U$ is formed as

$$U \equiv U_0 = \sum_{j=1}^{q_{FEAST}} \omega_j (z_j B - A)^{-1} BY.$$

(5)

Often the most expensive step of a contour integration scheme is the solution of linear systems $(z_j B - A)^{-1} BY$ involved in the construction of $U$. Discounting differences based on the type of linear solver used, the cost for one integration depends on $q$, the number of quadrature nodes (or block linear systems solved), and on RHS$_1$, the number of right hand sides (or columns of $Y$) in each block linear system. The overall cost will therefore be dependant on RHS$_{ovl}$ and BLS$_{ovl}$, the overall number of right hand sides and of block linear systems solved, resp., over all potential iterations.

Note that the number of columns of $U$, width($U$), is by construction width($U$) = RHS$_1 \cdot s$, where RHS$_1$ is the number of columns of $Y$, and $s$ is the number of moments. Therefore, a subspace $U$ constructed with $s$ moments requires the input matrix $Y$ to have only $1/s$th of the desired number of columns of $U$.

SS–RR has traditionally used a large overall subspace size width($U$), resulting in rapid convergence rates and thus requiring few iterations. However, allowing for more iterations can give more possibilities for adaptive techniques and overall cost reduction. Predicting the expected behavior of the algorithms is in this case necessary to develop good heuristics. Having previously explored adaptivity with respect to quadrature nodes, we now consider the benefits of allowing the solver type itself to vary between iterations. This work considers the use of multiple moments in these schemes, as in SS–RR, as well as when a single-moment strategy (FEAST) is suitable, for a given subspace size. Our goal is to increase the efficiency of the eigensolver by reducing RHS$_{ovl}$ while maintaining accuracy and robustness. To this end, we introduce and compare various strategies, utilizing a software framework for projected subspace iterative eigensolvers, BEAST.

The results of this article are also used in the dissertation of the first author.

2 | FLEXIBLE SUBSPACE ITERATION

FEAST is, by definition, a subspace iteration scheme. The iterative scheme, as presented in Algorithm 1, results in a set of vectors, $\bar{X}$, spanning a subspace close to the desired eigenspace. Bounds on the distance between the spaces and errors in the approximate eigenpairs have been previously explored; in, for example, the analysis provided by Krämer in thms. 2.43 and 2.46, the space spanned by the computed $U$ differs from the exact subspace by an amount that depends on the quadrature formula (5) used to approximate (1), as well as on possible errors in the solution of the linear systems. Then, the errors in the computed eigenspaces depend on the angular perturbation of the subspace and on the separation of suitable (sub)spectra, as discussed in thm. 2.15 of the same work.
Algorithm 1. FEAST

**Input:** \(A, B \in \mathbb{C}^{n \times n}, Y \in \mathbb{C}^{n \times \text{RHS}_1}\)

1. **while** not converged **do**
2. Construct subspace \(U \equiv U_0 = \sum_{j=1}^{\text{FEAST}} \omega_j (z_j B - A)^{-1} BY\)
3. Solve reduced eigenproblem \(A_U W = B_U W \Lambda\) where \(A_U = U^* AU, B_U = U^* BU\)
4. Obtain full-sized eigenvectors \(\tilde{X} = UW\)
5. **end while**

SSM, on the other hand, was not originally defined as an iterative scheme. A method using a linear combination of the approximate eigenvectors, constructed by multiplying these vectors with an appropriately sized random matrix, \(R\), as the initial vectors in the next iteration has, however, previously been defined. This “outer iteration” variant of SSM is presented in Algorithm 2.

Algorithm 2. SS–RR with outer iteration type

**Input:** \(A, B \in \mathbb{C}^{n \times n}, Y \in \mathbb{C}^{n \times \text{RHS}_1}\)

1. **while** not converged **do**
2. Construct subspace \(U = \left[U_0, U_1, U_2, \ldots, U_{s-1}\right]\) with \(U_k = \sum_{j=1}^{\text{SSM}} \omega_j^k (z_j B - A)^{-1} BY\)
3. Orthogonalize \(U\) and remove rank deficient columns
4. Solve reduced eigenproblem \(A_U W = B_U W \Lambda\) where \(A_U = U^* AU, B_U = U^* BU\)
5. Obtain full-sized eigenvectors \(\tilde{X} = UW\)
6. Compress subspace to dimension \(\text{RHS}_1\) for next iteration \(Y = \tilde{X} R\) with \(R \in \mathbb{C}^{\text{width}(\tilde{X}) \times \text{RHS}_1}\)
7. **end while**

More recently, the “inner iteration” has been introduced involving multiple passes of a set of initial vectors through (5) before the construction of the full subspace using multiple moments. The “outer” and “inner” iteration types are mainly distinguished by the choice of starting vectors in subsequent iterations. The “outer” iteration uses the approximate eigenvectors and the “inner” iteration uses the result of numerical quadrature with the 0th moment, \(U_0\). In Algorithm 3 we introduce a presentation of the latter scheme including an optional convergence check. This check, and in particular the extraction of the approximate eigenpairs in each iteration, is not strictly necessary from an algorithmic point of view, as the starting vectors for the next iteration are computed directly in the numerical quadrature step, but it allows us to observe convergence behavior at each iteration, and enables adaptivity in the algorithm. Further details will be discussed in Section 4. For large problems, the cost of computing \(U\) vastly outweighs the cost of the Rayleigh–Ritz step, so performance should not be significantly affected by the additional work, which may also be offset by savings through adaptive strategies.

Algorithm 3. SS–RR with inner iteration type

**Input:** \(A, B \in \mathbb{C}^{n \times n}, Y \in \mathbb{C}^{n \times \text{RHS}_1}\)

1. **while** not converged **do**
2. Construct subspace \(U = \left[U_0, U_1, U_2, \ldots, U_{s-1}\right]\) with \(U_k = \sum_{j=1}^{\text{SSM}} \omega_j^k (z_j B - A)^{-1} BY\)
3. Optional convergence check:
4. Orthogonalize \(U\) and remove rank deficient columns
5. Solve reduced eigenproblem \(A_U W = B_U W \Lambda\) where \(A_U = U^* AU, B_U = U^* BU\)
6. Obtain full-sized eigenvectors \(\tilde{X} = UW\)
7. Compute residuals as norms of columns of \(AX - BX\Lambda\)
8. Set starting vectors for next iteration \(Y = U_0\) (\(U_0\) as in line 2)
9. **end while**
When the approximation of the contour integral dominates computational expense, the inner and outer iteration for a similar size of the subspace $U_0$ will have similar cost.

Previous work\(^8\) has considered the theoretical filtering and convergence behavior of the inner iteration. This is substantially more nebulous for the outer iteration, and this work will focus on a heuristic analysis of best practices for convergence and adaptivity. As adaptivity relies on iterative behavior, we focus on situations where inner and outer iterations are truly relevant; using a smaller subspace, and thus fewer RHS (single linear solves) per iteration.

Under a constrained subspace size, however, moment-based schemes may experience stagnation or regression of convergence for some of the desired eigenpairs, in particular for very small tolerances. An example of this behavior for a model problem can be seen in the following section. Previous studies of SSM have typically involved larger subspace sizes and fewer subspace iterations, where it is expected that the desired eigenspace will be well represented. The connection between improved subspace filtering for the desired directions with decreased number of moments is intuitive and has been previously observed.\(^9\) An iterative method might therefore allow a flexible switch from a multi-moment to single-moment scheme if stagnation or loss of convergence is detected. The single-moment scheme may be more costly, as it requires more RHS in each linear system for the same size of subspace $U$, but may converge to a smaller tolerance due to the increased numerical stability of the projected subspace. Since this switch is typically only required when residuals are already rather small, we do not expect to require many of the more expensive single-moment iterations. Furthermore, techniques such as a standard locking scheme\(^24\) for converged eigenpairs may reduce the cost of a more expensive single-moment iteration by reducing the number of RHS considered. The heuristics and advantages of the flexible choice of moments within an iterative projected subspace method will provide the main contribution in this work.

## 3 INNER VERSUS OUTER ITERATIONS AND MULTI- VERSUS SINGLE-MOMENT

In this section we compare the convergence behavior of iteration schemes using either only inner iterations or only outer iterations, or switching from inner to outer iterations, as well as consider the effect of switching from multiple moments (SSM-type) to single-moment (FEAST-type). This motivates the adaptive strategy described in the following section.

We considered the toy problem
\[
A x_i = \lambda_i x_i, \quad \lambda_i \in I_\lambda = [-1, 1], \quad \text{where } A = \text{diag}(-2.99, -2.89, \ldots, 6.91) \in \mathbb{R}^{100 \times 100},
\]
containing $m = 20$ eigenvalues in the search interval. We have chosen this very simple setting because it allows for explicit observation of the phenomena described below without disturbances from other matrix effects.

The unit circle was chosen as the contour of integration, and we used Gauss–Legendre quadrature with 16 quadrature nodes on the contour. Note that only the 8 linear systems along the upper half of the contour must be solved due to symmetry; this is the value originally defined in FEAST.\(^2\) The subspace was constrained to yield a truly iterative scheme. More precisely, the number of initial vectors after each iteration was chosen as to give an overall subspace size width($U$) = $32 \approx 1.5 \times m$. Testing was done in Matlab, with the initial $Y$ chosen randomly using the “twister” generator from a standard normal distribution, and linear systems were solved using Matlab’s backslash function.

The minimum, average, and maximum of the smallest $m = 20$ residuals for the approximate eigenpairs are plotted in Figure 1 (using 4 moments and therefore 8 columns in $U_0$) and Figure 2 (8 moments, 4 columns). We observe different behavior for the inner and outer iteration types with moments, which further varies with the number of moments (and size of $U_0$).

In these examples, we also explore switching from inner to outer iterations and from multi to single moment schemes, with an artificial switch set approximately halfway through the 20 iterations viewed. As we have not defined inner iterations with a FEAST-type single moment scheme, we choose to switch from inner to outer iterations in the 9th iteration before a possible switch from a multi- to single-moment scheme in the 10th iteration.

### 3.1 Inner and outer iterations

In the top row of Figures 1 and 2, we observe the different behavior for outer and inner iterations. We see that for early iteration counts, convergence is noticeably improved for the inner iteration compared to the outer iteration. These effects
FIGURE 1  Rank and convergence (maximum, minimum, and average of the 20 smallest residuals) of the contour integration-based iterative schemes for problem (6) with initial subspace size 32 and 4 moments for the first iteration. The number of moments is denoted in the background, with a vertical line indicating a change in number of moments. Background hashed for inner iterations. Bottom row: Switch from SSM-type multi-moment (BEAST-M implementation, cf. Section 4) to FEAST-type single-moment (BEAST-C, cf. Section 4) over iterations 10 and 11. Top row: No such switching, multi-moment throughout. Left column: All inner iterations. Middle column: Switch from inner to outer iterations over iterations 9 and 10. Right column: All outer iterations. Bottom left: Left blank as inner iterations are not consistent with FEAST-type single moment.

FIGURE 2  As Figure 1, but with 8 moments.
are amplified as the starting number of moments increases, as seen in the difference between Figures 1 and 2. In later iterations, however, some convergence may be lost as error creeps back in to the largest residual. The random recombination of approximate eigenvectors used to generate Y for outer iterations may be responsible for some oscillatory behavior at low residual thresholds, as has previously been explored.21

### 3.2 Switching from multi- to single-moment

In the bottom row of Figures 1 and 2 we observe the effect of switching from a multi- to single-moment scheme. After the 9th iteration, we switch from a SS–RR- to a FEAST-type iteration, with only one moment. As the overall subspace size remains the same 1.5x the approximate number of eigenpairs, this means that more right hand sides are involved in the solution of the linear systems. Numerical instability may increase with the number of moments; we observe a significant decrease in numerical rank of $U$, especially with multiple moments. We expect this is a significant factor in the convergence of these methods. We see the increase in stability of convergence with a reduction in the number of moments, for previous iterations of both the outer and inner type. Observing this trend, we anticipate that a small residual error may be more easily achieved with single moment iterations. However, in early iterations, single moment schemes are more expensive (due to the additional right hand sides) and have similar convergence rates to an equivalent multi-moment scheme. A well-timed switch to a single moment scheme could reduce the overall cost of the eigen solver, while allowing smaller overall residuals to be obtained. The iteration in which to perform this switch may be chosen by monitoring the convergence rate, for example, as in Reference 19, by tracking the smallest not yet converged residual. When this metric shows signs of stagnation, as will be discussed later, the switch to a single moment scheme may take place. The iterative nature of the eigensolver makes this a straightforward change, and simply requires constructing an appropriately sized block vector $Y$ for the starting vectors in the next iteration such that the subspace may be constructed with the desired number of moments.

### 4 THE BEAST FRAMEWORK

BEAST provides a framework for subspace iteration with Rayleigh–Ritz extraction of eigenvalues and vectors.25,26 Testing and development are often done with a Matlab version of BEAST, and a C version with multiple levels of parallelization is available for performance-critical runs.

BEAST provides two contour-based ways to construct the subspace $U$; our SS–RR- and FEAST-type implementations are referred to as BEAST-M and BEAST-C, respectively. (For standard eigenvalue problems $Ax = \lambda x$ we also provide BEAST-P, which uses a polynomial filter: $U = p(A)Y$. This component is not considered in the present paper.) The framework contains algorithmic strategies discussed in previous works for improving the foundational methods, such as the locking of converged eigenvectors,19 as well as the components integral to the reference algorithms, such as the orthogonalization of the subspace for BEAST-M.

One particular feature of BEAST is that it allows the method (or its parameters) to be changed from one iteration to the next, and some of the resulting overall strategies are discussed in the remainder of this section. We first explain the naming conventions.

- **BEAST-M** and **BEAST-C** refer to the method used in a given iteration.
- For **BEAST-M**, we can do inner or outer iterations (denoted as **BEAST-M_{i,x}** and **BEAST-M_{o,x}**, resp.), and we may or may not allow switching to **BEAST-C** whenever stagnation of convergence is detected (**BEAST-M_{o,x}** and **BEAST-M_{o,n}**, resp.). Thus, **BEAST-M_{o,x}** starts with multi-moment outer iterations and may switch to single-moment upon stagnation.

  Stagnation was measured in the “drop rate” ratio of the smallest residual among the not-yet-converged eigenpairs, $r_{SNC}^i = \max_j \frac{\|A\bar{x}_j - B\bar{x}_j\|}{\|A\bar{x}_j - B\bar{x}_j\|_{t-1}}$ : this residual is above tolerance $\epsilon$, since the previous iteration; that is, $r_{SNC}^i/r_{SNC}^{i-1}$. Whenever this ratio rose above the threshold of 0.01, stagnation was determined to have set in.

- For **BEAST-C** iterations, the variant **BEAST-C_{ad}** adjusts the number of quadrature nodes $q$ (and the nodes and coefficients) over the iterations, whereas **BEAST-C_n** does not.

Figure 3 shows the different choices for the methods, as well as SSM-based methods included in Section 5, and Algorithm 4 gives an overview of the computations done in each BEAST case.
FIGURE 3  Overview of all computation schemes and choices considered in Section 5.

Algorithm 4. BEAST framework, including BEAST-M and BEAST-C algorithmic variants. The option of adaptive quadrature node choice for BEAST-C is shown as BEAST-Cad. The option of adaptive method switching for BEAST-M is shown as BEAST-Ms,x. The option of inner or outer iterations for BEAST-M are shown as BEAST-Mo,x and BEAST-Mi,x.

**Input:** Matrix pair $A, B$ of size $n \times n$
- $\tilde{m}$ estimate for number of eigenpairs in interval
- $q$ quadrature nodes and coefficients $(z_j, \omega_j)$
- $Y \in \mathbb{C}^{n \times \text{RHS}_1}$

1: while not converged do
2: if in BEAST-Cad mode then
3: Choose $q, z_j, \omega_j$  \(\triangleright\) otherwise use parameters as provided with inputs
4: end if
5: Construct subspace $U$ according to (4) (BEAST-M mode) or (5) (BEAST-C mode)
6: if in BEAST-Mi,x mode then
7: Set $Y := U_0$  \(\triangleright\) inner iteration
8: end if
9: Compute singular values of $U$
10: if in BEAST-M mode then
11: Orthogonalize $U$ (rank-revealing factorization)
12: end if
13: Resize subspace
14: Rayleigh–Ritz extraction of eigenvalues $\Lambda$ and eigenvectors $\tilde{X}$ according to (3)
15: Orthogonalize $\tilde{X}$ against locked eigenpairs, lock newly converged eigenpairs
16: if in BEAST-Ms,x mode and stagnation has been detected then
17: switch to BEAST-C
18: end if
19: if in BEAST-Mo,x mode then
20: Set $Y := \tilde{X}R$ with $R \in \mathbb{C}^{\text{width}(U) \times \text{RHS}_1}$
21: else if in BEAST-C mode then
22: Set $Y := \tilde{X}$
23: end if
24: end while
5 NUMERICAL EXPERIMENTS

Numerical results are shown for 37 test problems as detailed in Table 1. The problems arise from graphene modeling, and the SuiteSparse Matrix Collection. All contour-based schemes require selecting a quadrature rule. Gauss–Legendre is a typical choice for FEAST, and the trapezoidal rule is a typical choice for SSM. The reasoning for these standards becomes clear when comparing the \( \text{RHS}_{\text{olv}} \) counts over several test problems. We used an elliptical contour with eccentricity 0.1 and 16 quadrature nodes. Testing BEAST–\( M_{G, n} \) and BEAST–\( M_{I, n} \) with both the large subspace size typical for SSM (dimension of overall subspace four times larger than the expected number of eigenvalues) and the constrained size typical for FEAST (factor 1.5), in both cases, the average number of \( \text{RHS}_{\text{olv}} \) was similar or better for Gauss–Legendre with subspace factor 1.5, and trapezoidal was superior with subspace factor 4; compare Table 2. The difference was especially marked in the case of the outer iteration. Investigating the causes of this difference is left for future work. In the following tests, we used Gauss–Legendre for BEAST and the trapezoidal rule for \texttt{sseig} (a Matlab implementation of SS–RR), in both cases for an elliptical contour with eccentricity 0.1 and 16 quadrature nodes on the whole contour.

We first compared the total number of right hand sides used in all linear system solves, \( \text{RHS}_{\text{olv}} \), as well as all block linear solves, \( \text{BL}_{\text{olv}} \), over the iterations required to reach a tolerance of \( 10^{-13} \). Testing was done in Matlab, with the initial \( Y \) chosen randomly and linear systems solved using Matlab’s backslash function. In BEAST, orthogonalization and the computation of singular values was done in one step using Matlab’s \texttt{svd} function. Small eigenproblems were solved directly using the \texttt{eig} function. In accordance with FEAST\textsuperscript{2} we chose a subspace size width(\( U \)) \( \approx 1.5 \times \tilde{m} \), where \( \tilde{m} \) is the expected number of eigenpairs in the interval. Given an estimated number of eigenpairs \( \tilde{m} = 300 \) for all test problems.

| Name          | Size    | nnrz | \( [\lambda_{\text{min}}, \lambda_{\text{max}}] \) | Interval A | \( m \) | No. | Interval B | \( m \) | No. |
|---------------|---------|------|-------------------------------------------------|------------|------|-----|------------|------|-----|
| laser         | 3002    | 3    | \([-1.10, 4.25]\)                               | \([-0.100, 0.357]\) | 307  | 1   | \([1.000, 4.238]\) | 304  | NA |
| SiH4          | 5041    | 34   | \([-0.996, 36.8]\)                              | \([25.0, 28.4]\) | 301  | 2   | \([15.3, 16.4]\) | 290  | 3   |
| linverse      | 11,999  | 8    | \([-4.70, 15.5]\)                               | \([0.00, 0.62]\) | 308  | 4   | \([2.62, 2.77]\) | 304  | 5   |
| Pres_Poisson  | 14,822  | 48   | \([-1.28e-5, 26.0]\)                            | \([3.7, 10.0]\) | 302  | 6   | \([1.00, 1.182]\) | 300  | 7   |
| SiH12         | 19,896  | 37   | \([-0.996, 58.6]\)                              | \([24.2, 24.7]\) | 320  | 8   | \([41.42]\) | 274  | 9   |
| bcsstk37      | 25,503  | 45   | \([-7.04e-5, 8.41e7]\)                          | \([8.0e5, 9.3e5]\) | 297  | 10  | \([1.15e7, 1.30e7]\) | 305  | 11  |
| brainpc2      | 27,607  | 6    | \([-2000, 4460]\)                               | \([275, 345]\) | 313  | 12  | \([1900, 1920]\) | 309  | 13  |
| rgg_n_2_15_s0 | 32,768  | 10   | \([-5.12, 17.4]\)                               | \([-2.00, -1.95]\) | 282  | 14  | \([5.0, 5.5]\) | 337  | 15  |
| SiO           | 33,401  | 39   | \([-1.67, 84.3]\)                               | \([32.0, 32.4]\) | 316  | 16  | \([57.0, 57.8]\) | 283  | 17  |
| Andrews       | 60,000  | 13   | \([-3.64e-16, 36.5]\)                           | \([21.0, 21.4]\) | 300  | 18  | \([11.20, 11.26]\) | 298  | 19  |
| GraI-1k       | 1152    | 12   | \([-3.43, 2.78]\)                               | \([-0.7575, 1.1025]\) | 301  | 20  | \([0.42, 1.58]\) | 301  | 21  |
| GraI-11k      | 11,604  | 13   | \([-3.43, 2.78]\)                               | \([-0.0475, 0.3925]\) | 298  | 22  | \([0.935, 1.065]\) | 289  | 23  |
| GraI-119k     | 119,908 | 13   | \([-3.43, 2.78]\)                               | \([0.1375, 0.2075]\) | 306  | 24  | \([0.9957, 1.0043]\) | 304  | 25  |
| GraII-1k      | 1152    | 13   | \([-3.43, 2.78]\)                               | \([-0.7575, 1.1025]\) | 292  | 26  | \([0.42, 1.58]\) | 304  | 27  |
| GraII-11k     | 11,604  | 12   | \([-3.43, 2.79]\)                               | \([-0.1375, 0.4825]\) | 299  | 28  | \([0.949, 1.051]\) | 299  | 29  |
| GraII-119k    | 119,908 | 13   | \([-3.43, 2.79]\)                               | \([0.0975, 0.2475]\) | 313  | 30  | \([0.9956, 1.0044]\) | 303  | 31  |
| GraIII-1k     | 1152    | 12   | \([-3.35, 2.73]\)                               | \([-0.7575, 1.1025]\) | 300  | 32  | \([0.42, 1.58]\) | 331  | 33  |
| GraIII-11k    | 11,604  | 12   | \([-3.35, 2.73]\)                               | \([-0.0975, 0.4425]\) | 305  | 34  | \([0.952, 1.048]\) | 319  | 35  |
| GraIII-119k   | 119,908 | 13   | \([-3.43, 2.78]\)                               | \([0.1395, 0.2055]\) | 310  | 36  | \([0.996, 1.004]\) | 311  | 37  |

Note: For each problem, the test matrix and interval were taken as \( A \) and \( I \) in (1), \( B = I \). The problems are also problem numbers 1:25, 27:34, 36:39 in Table 1 of Galgon et al.\textsuperscript{25}
**TABLE 2** Average RHS\textsubscript{ovl} with different quadrature rules for 31 of the 37 problems listed in Table 1.

| Iteration type | Quadrature rule | Subspace factor | Average RHS\textsubscript{ovl} |
|----------------|-----------------|-----------------|-------------------------------|
| BEAST-M\textsubscript{i,x} | Gauss Legendre | 1.5 | 3587 |
| BEAST-M\textsubscript{i,x} | Trapezoidal | 1.5 | 3514 |
| BEAST-M\textsubscript{i,x} | Gauss Legendre | 4.0 | 5229 |
| BEAST-M\textsubscript{i,x} | Trapezoidal | 4.0 | 4545 |
| BEAST-M\textsubscript{o,x} | Gauss Legendre | 1.5 | 3201 |
| BEAST-M\textsubscript{o,x} | Trapezoidal | 1.5 | 3723 |
| BEAST-M\textsubscript{o,x} | Gauss Legendre | 4.0 | 5218 |
| BEAST-M\textsubscript{o,x} | Trapezoidal | 4.0 | 4366 |

*Note*: Problems for which all not schemes found all eigenpairs (7, 14, 20, 26, 34) are not included in the average.

BEAST-M and BEAST-C began with overall subspace sizes width(\(U\)) of 452 and 450, respectively. BEAST-M began in all cases with 4 moments.

We compared with the BEAST-C solver with an adaptive number of quadrature nodes \(q\), BEAST-C\textsubscript{ad}, and with the best fixed-order method, BEAST-C\textsubscript{n} (i.e., the fixed \(q\) leading to the minimum RHS\textsubscript{ovl} among all \(q\) considered, such that all eigenpairs were found). In the adaptive case, as described by Galgon et al.,\textsuperscript{25} a minimal initial value of \(q\) is increased if the smallest non-converged residual decreases by an unsatisfactory amount; by a factor of \(\sqrt{1.5}\) if the smallest non-converged residual decreases by a factor of \((10^{-1.5}, 10^{-0.75})\) and by a stronger factor of 1.5 if the decrease factor is greater than \(10^{-0.75}\).

In either case, the number of quadrature nodes must be increased by at least 1. When the reduction factor is smaller than \(10^{-1.5}\), \(q\) is left unchanged. A comparison was also made to sseig, which was also run with 4 moments, and the subspace was allowed to grow expansively, starting with width(\(U_0\)) = 16 right hand sides and growing to a maximum of 256 for the problems considered, for an overall subspace size of width(\(U\)) = 1024. The final comparison was with sseig\textsuperscript{2beastc}, switching from sseig after a single outer iteration to BEAST-C, using all approximate eigenvectors as the initial vectors \(Y\) for BEAST-C. This allowed us to test the scheme-switching heuristic in the context of large subspace sizes. With a large subspace size, it is expected that sseig\textsuperscript{and sseig\textsuperscript{2beastc}} will converge quickly, that is, with just a few iterations.

We measured success in terms of the number of linear solves (single or block) required for all eigenpairs to converge (efficiency), as well as the number of times the method found all eigenpairs inside the interval (robustness). The respective results are summarized in Figure 4, as well as Table 3 and Table A1. We point out that the methods were deemed to have “failed” when not all eigenvalues inside the interval were found to the desired tolerance. This may be an overly strict measure, especially for sseig, which we observed often had a few eigenvalues slightly above the residual tolerance at completion.

We notice that all methods obtaining the minimum, or near minimum observed number of single linear solves, RHS\textsubscript{ovl}, included multiple moments, and the difference in RHS\textsubscript{ovl} between methods involving and not involving moments is substantial. Methods involving a switch to BEAST-C appear to behave more robustly, with fewer failures to find all eigenpairs in the interval to the desired tolerance. We see that both BEAST-M\textsubscript{i,x} and BEAST-M\textsubscript{o,n} outperformed sseig and BEAST-C, with BEAST-M\textsubscript{n,n} and BEAST-M\textsubscript{i,x} having the same minimum RHS\textsubscript{ovl} in most cases.

When switching was allowed, BEAST-M\textsubscript{o,x} and BEAST-M\textsubscript{i,x} behaved robustly, failing to find all eigenvalues for only one and two problems respectively. In comparison, BEAST-C\textsubscript{ad}, BEAST-M\textsubscript{n,n}, and sseig featured an observably higher number of cases where not all eigenpairs were found to the desired tolerance. For BEAST, even if this were an acceptable behavior, a failed convergence check (performed using singular values of \(U\)) will result in continued iterations until all eigenpairs have reached the desired tolerance or some maximum number of iterations is reached. For stagnant eigenpairs this may be a considerable expense. We also observe the increased BLS\textsubscript{ovl} for BEAST-M compared to BEAST-C\textsuperscript{and sseig}, due to a higher number of iterations. We predict, and demonstrate, in the following tests, that the reduction of RHS\textsubscript{ovl} is enough to offset this cost, but acknowledge that this depends on the linear solver used, the computational set-up, including the distribution of work and memory, and the problem itself.

Taking a closer look at BEAST-M\textsubscript{o,x} and BEAST-M\textsubscript{i,x}, we see similar behavior between the two methods, with a few larger outliers in RHS\textsubscript{ovl} for BEAST-M\textsubscript{i,x}. Based on our results from the previous section, we might have expected noticeably better performance from the inner iteration as opposed to the outer. In fact, the initial improvement in convergence...
for the inner iteration is a significant reason for these outliers. We consider the numerical example shown in Figure 5. After convergence begins, eigenpairs are locked, and the subspace shrinks; perhaps overly much, as convergence subsequently slows down, even after a switch to BEAST–C. Eventually, an additional iteration is required for BEAST–M\textsubscript{i,x}, which started out with much faster convergence than BEAST–M\textsubscript{o,x}.

Out of the various methods considered so far, we choose to highlight BEAST–M\textsubscript{o,x} for its combination of robustness, adaptability, and low cost. We now turn to numerical results showing the possible savings in cost available with this scheme.

We also considered how a reduction in RHS\textsubscript{ovl} corresponds to savings in time. We compared parallelized, distributed memory implementations of BEAST–C (specifically BEAST–C\textsubscript{n}) and BEAST–M\textsubscript{o,x} for two larger systems. The scalable solver strumpack\textsuperscript{29} was used to solve all linear systems directly, and the kernel library ghost\textsuperscript{30} provided fast matrix operations as a background library. Testing was performed with 32 nodes of the Emmy HPC cluster at Friedrich-Alexander-Universität Erlangen–Nürnberg. The eigenproblems are detailed in Table 4. They were solved to a tolerance of $10^{-12}$, using 16 quadrature nodes and Gauss–Legendre quadrature on a circular contour. The number of columns of $U$ was set to $1.5 \times n$, and locking was enabled. BEAST–M\textsubscript{o,x} began with 4 moments and was allowed to switch to BEAST–C when stagnation was detected.
FIGURE 5  Convergence and subspace size width(U) with BEAST-M₁ₓ and BEAST-M₀ₓ for Problem 4, Table 1 in Galgon et al.²⁵

TABLE 4  Sizes, average number of nonzeros per row (nnrz; rounded), Approximate smallest (λₘᵋᵣ) and largest (λᵣᵢₐₓ) eigenvalues, search interval, starting approximate eigenvalue count ̃ₘ, and number of eigenvalues found ̂ₘ for test matrices from graphene²⁷ and topological insulator modeling.³¹

| Name   | Size      | nnzr | [λₘᵋᵣ,λᵣᵢₐₓ]       | Interval           | ̃ₘ   | ̂ₘ   |
|--------|-----------|------|----------------------|--------------------|------|------|
| Graph16M | 16,000,000 | 4    | [−3.0, 3.0]          | [−0.0025, 0.0025]  | 320  | 318  |
| Topi1M   | 1,638,400  | 12   | [−4.8, 4.8]          | [−0.06, 0.06]      | 120  | 116  |

Note: Related matrices may be generated from the ScaMaclibrary.³² The spectral properties of matrices of these forms with regard to computation have been previously studied.³³ For each problem, the test matrix and interval were taken as A and Iₐ in (1), B = I.

FIGURE 6  Linear solve count and timings of BEAST-M₀ₓ versus BEAST-C. In rightmost figure, time for building U shown in color, time for all other components shown in black.
In Figure 6, we again see the reduction in \( \text{RHS}_{\text{ovl}} \) of \( \text{BEAST-M}_{0, x} \) compared to \( \text{BEAST-C} \), as well as the respective times to solution. As expected, the construction of \( U \) was by far the most time consuming portion of \( \text{BEAST} \), due to the expense of solving linear systems. The time reduction was more pronounced for the graphene problem than for the topological insulator. For the latter, the decrease in \( \text{RHS}_{\text{ovl}} \) for \( \text{BEAST-M} \) compared to \( \text{BEAST-C} \) was not so dramatic, and at least one additional iteration was required, as can be seen in the \( \text{BLS}_{\text{ovl}} \) count. However, even with the additional iteration(s), \( \text{BEAST-M}_{0, x} \) was still faster than \( \text{BEAST-C} \).

We have exclusively considered projected subspace methods in this work. These methods have proven well suited to interior eigenvalue problems, in particular for large problems and when a reasonably large number of eigenpairs are sought in an interval. The comparison could also be expanded to include, for example, Krylov subspace methods, such as Krylov–Schur.\(^{34}\) Initial investigations show that these methods may require significantly fewer \( \text{RHS}_{\text{ovl}} \) than the projected subspace methods explored here. However, the fundamental nature of a subspace expansion scheme such as Krylov–Schur requires the orthogonalization of each new vector against the rest of the subspace. Operations such as these may result in an overall less efficient algorithm for problem sizes beyond a certain threshold, and may also be inherently less scalable. Furthermore, the use of these methods for interior eigenvalue problems typically involves a shift-invert technique with a real-valued shift, which may result in a very ill-conditioned linear system of equations if close to an eigenvalue, having a major impact on its solution. However, since the focus of the present paper is on \( \text{RHS}_{\text{ovl}} \) rather than time, we do not further elaborate on this.

6 | CONCLUSION

Contour integration schemes are a powerful tool for interior eigenvalue problems, but the repeated solution of large linear systems with many right hand sides is a bottleneck in time and energy. The judicious choice of various parameters has a noticeable effect on the potential overall cost of the solver, particularly those parameters corresponding to the cost of linear system solves. Obvious parameters, such as the choice of quadrature degree, as considered in previous work,\(^{19}\) will have a noticeable effect on the overall cost of linear system solves per iteration, and may be chosen adaptively for an overall reduction in cost over iterations. However, this parameter, and others like it, have a direct effect on the convergence rate. We have explored the heuristic choice of the number of moments, which, while relevant to convergence, may have a less direct effect, especially for initial iterations. The choice of the number of moments is a powerful parameter for controlling the cost of linear system solves, and thus the overall eigensolver cost. We explored further heuristics within the context of an iterative eigensolver, including how the number of moments and the starting vectors may be chosen. In a broad comparison between methods, we provide evidence that a multi-moment flexible iterative method may reduce the number of single linear solves over all iterations, and thus the potential overall cost of an eigensolver. Furthermore, the flexibility of the method ensures comparative robustness and accuracy. We provide further evidence for performance capabilities in initial larger experiments.

Future topics of exploration include the adaptive choice of quadrature degree in combination with multiple moments, extending the work of Reference 19. Furthermore, the capacity for performance improvement with multiple moments and reduced right hand sides clearly depends on the linear solver used. In these experiments, due to the difficulty of solving these shifted linear systems, we tested with direct solvers. In future work, the capacity for improvement dependant on the properties of the linear solver used may be investigated. Results and exploration for an iterative linear solver would be of particular interest as present problems may exceed the capacity of direct solvers.

ACKNOWLEDGMENTS

The present study has been supported in part by the Deutsche Forschungsgemeinschaft (No. SPP 1648) through the priority programme 1648 "Software for Exascale Computing" (SPPEXA) under the project ESSEX-II, as well as by the Japan Society for the Promotion of Science (JSPS), Grants-in-Aid for Scientific Research (Nos. 17K12690, 18H03250, 19K20280).

CONFLICT OF INTEREST

This study does not have any conflicts to disclose.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.
REFERENCES

1. Sakurai T, Sugiura H. A projection method for generalized eigenvalue problems using numerical integration. J Comput Appl Math. 2003;159(1):119–28.
2. Polizzi E. Density-matrix-based algorithm for solving eigenvalue problems. Phys Rev B Condens Matter. 2009;79(11):115112.
3. Sakurai T, Tadano H. CIRR: a Rayleigh-Ritz type method with contour integral for generalized eigenvalue problems. Hokkaido Math J. 2007;36(4):745–57.
4. Imakura A, Du L, Sakurai T. Relationships among contour integral-based methods for solving generalized eigenvalue problems. Jpn J Ind Appl Math. 2016;33(3):721–50.
5. Imakura A, Du L, Sakurai T. A block Arnoldi-type contour integral spectral projection method for solving generalized eigenvalue problems. Appl Math Lett. 2014;32:22–7.
6. Ikegami T, Sakurai T. Contour integral eigensolver for non-Hermitian systems: a Rayleigh–Ritz-type approach. Taiwan J Math. 2010;14(3A):825–37.
7. Ikegami T, Sakurai T, Nagashima U. A filter diagonalization for generalized eigenvalue problems based on the Sakurai–Sugiura projection method. J Comput Appl Math. 2010;233(8):1927–36.
8. Imakura A, Du L, Sakurai T. Error bounds of Rayleigh–Ritz type contour integral-based eigensolver for solving generalized eigenvalue problems. Numer Algorithms. 2016;71(1):103–20.
9. Sakurai T, Futamura Y, Tadano H. Efficient parameter estimation and implementation of a contour integral-based eigensolver. J Algorithms Comput Technol. 2013;7(3):249–69.
10. Sakurai T, Futamura Y, Imakura A, Imamura T. Scalable eigen-analysis engine for large-scale eigenvalue problems. In: Sato M, editor. Advanced software technologies for post-petascale computing. New York, NY: Springer; 2019. p. 37–57.
11. Imakura A, Sakurai T. Block SS–CAA: a complex moment-based parallel nonlinear eigensolver using the block communication-avoiding Arnoldi procedure. Parallel Comput. 2018;74:34–48.
12. Peter Tang PT, Polizzi E. FEAST as a subspace iteration eigensolver accelerated by approximate spectral projection. SIAM J Matrix Anal Appl. 2014;35(2):354–90.
13. Guettel S, Polizzi E, Tang P, Viaud G. Zolotarev quadrature rules and load balancing for the FEAST eigensolver. SIAM J Sci Comput. 2015;37(4):A2100–22.
14. Yin G, Chan RH, Yeung MC. A FEAST algorithm for generalized non-Hermitian eigenvalue problems. Numer Algorithms. 2016;71(1):103–20.
15. Kestyn J, Polizzi E, Peter Tang PT. FEAST eigensolver for non-Hermitian problems. SIAM J Sci Comput. 2016;38(5):S772–99.
16. Alappat CL, Alvermann A, Basermann A, Fehske H, Futamura Y, Galgon M, et al. ESSEX: equipping sparse solvers for exascale. In: Bungartz H-J, Reitz S, Uekermann B, Neumann P, Nagel WE, editors. Software for exascale computing—SPPEXA 2016-2019. Lecture Notes in Computational Science and Engineering. Volume 136. New York, NY: Springer; 2020. p. 143–87.
17. Davis TA, Hu Y. The university of Florida sparse matrix collection. ACM Trans Math Softw. 2011;38(1):1:1–1:25.
29. Rouet FH, Li XS, Ghysels P, Napov A. A distributed-memory package for dense hierarchically semi-separable matrix computations using randomization. ACM Trans Math Softw. 2016;42(4):27:1–27:35.
30. Kreutzer M, Thies J, Röhrig-Zöllner M, Pieper A, Shahzad F, Galgon M, et al. GHOST: building blocks for high performance sparse linear algebra on heterogeneous systems. Int J Parallel Prog. 2017;45(5):1046–72.
31. Hasan MZ, Kane CL. Colloquium: topological insulators. Rev Mod Phys. 2010;82:3045–67.
32. Alvermann A. Scalable matrix collection. Available from: https://bitbucket.org/essex/matrixcollection.
33. Pieper A, Kreutzer M, Alvermann A, Galgon M, Fehske H, Hager G, et al. High-performance implementation of Chebyshev filter diagonalization for interior eigenvalue computations. J Comput Phys. 2016;11(325):226–43.
34. Stewart GW. A Krylov–Schur algorithm for large eigenproblems. SIAM J Matrix Anal Appl. 2002;23(3):601–14.

**How to cite this article:** Huber S, Futamura Y, Galgon M, Imakura A, Lang B, Sakurai T. Flexible subspace iteration with moments for an effective contour integration-based eigensolver. Numer Linear Algebra Appl. 2022;29(6):e2447. [https://doi.org/10.1002/nla.2447](https://doi.org/10.1002/nla.2447)

### APPENDIX A. DETAILED RESULTS FOR FIGURE 4

| Table A1 | Matlab results RHS\textsubscript{rot} : BLS\textsubscript{rot}. |
|----------|--------------------------------------------------|
| **No.**  | \texttt{BEAST-C} | \texttt{sseig} | \texttt{BEAST-M} |
|      | \texttt{n} | \texttt{ad} | \texttt{no switch} | \texttt{2beastc} | \texttt{i,n} | \texttt{i,x} | \texttt{o,n} | \texttt{o,x} |
| 1      | 6972 : 16 | - | 5904 : 80 | 6392 : 80 | - | 8280 : 120 | - | 4992 : 40 |
| 2      | 10320 : 24 | - | 3744 : 64 | 4280 : 64 | 2968 : 32 | 7968 : 120 | 2992 : 32 | 2992 : 32 |
| 3      | 9585 : 27 | 6703 : 21 | 7200 : 80 | 4120 : 64 | 2672 : 24 | 2672 : 24 | 2712 : 24 | 2712 : 24 |
| 4      | 10234 : 28 | 6781 : 21 | 3680 : 60 | 4304 : 64 | 2776 : 24 | 2776 : 24 | 2856 : 24 | 2856 : 24 |
| 5      | 8885 : 20 | 6784 : 21 | 3920 : 64 | 4392 : 64 | 2752 : 24 | 2752 : 24 | 2712 : 24 | 2712 : 24 |
| 6      | 9835 : 35 | 10803 : 33 | 6144 : 88 | 4320 : 64 | 2584 : 24 | 2584 : 24 | 2712 : 24 | 2712 : 24 |
| 7      | 9765 : 27 | 6697 : 21 | 3648 : 64 | 4224 : 64 | 2728 : 24 | 2728 : 24 | 2712 : 24 | 2712 : 24 |
| 8      | 10410 : 30 | - | 4584 : 64 | 2808 : 24 | 2808 : 24 | 2848 : 32 | 2848 : 32 | 2848 : 32 |
| 9      | 9800 : 30 | 6637 : 20 | 3472 : 64 | 3928 : 64 | 2488 : 24 | 2536 : 32 | 2888 : 24 | 2888 : 24 |
| 10     | 8980 : 30 | - | 3712 : 64 | 4232 : 64 | 2584 : 24 | 2584 : 24 | 2712 : 24 | 2712 : 24 |
| 11     | 10818 : 27 | 6189 : 19 | 3824 : 64 | 4352 : 64 | 2808 : 32 | 2808 : 32 | 2880 : 32 | 2880 : 32 |
| 12     | 10936 : 32 | 7194 : 24 | 3936 : 72 | 4472 : 72 | 2792 : 24 | 2792 : 24 | 2712 : 24 | 2712 : 24 |
| 13     | 9297 : 27 | 6390 : 20 | 3824 : 64 | 4384 : 64 | 2696 : 24 | 2696 : 24 | 2712 : 24 | 2712 : 24 |
| 14     | 8980 : 30 | 6839 : 23 | - | - | - | - | - | - |
| 15     | 8995 : 20 | 8649 : 29 | 4032 : 64 | 4712 : 64 | 2904 : 32 | 2904 : 32 | 2928 : 32 | 3040 : 48 |
| 16     | 10596 : 30 | 8657 : 29 | - | 4496 : 64 | 2832 : 24 | 2832 : 24 | 2856 : 32 | 2856 : 32 |
| 17     | 8922 : 24 | 6348 : 19 | - | 3992 : 64 | 2648 : 24 | 2648 : 24 | 2712 : 24 | 2712 : 24 |
| 18     | 9744 : 28 | 7299 : 24 | 3920 : 72 | 4360 : 72 | 2744 : 24 | 2744 : 24 | 2712 : 24 | 2712 : 24 |
| 19     | 9963 : 27 | 7029 : 23 | 3696 : 64 | 4232 : 64 | 2728 : 48 | 3072 : 40 | 2880 : 32 | 2880 : 32 |
| 20     | 10255 : 35 | 12598 : 36 | - | 6456 : 80 | - | 4568 : 40 | - | 5144 : 72 |
| 21     | 8810 : 20 | 6936 : 26 | 2880 : 32 | 3848 : 32 | 2664 : 24 | 2664 : 24 | 2848 : 32 | 2848 : 32 |
| 22     | 9000 : 20 | 6848 : 21 | 6144 : 88 | 6480 : 88 | 2952 : 32 | 7656 : 120 | 2712 : 24 | 2712 : 24 |
| 23     | 13500 : 30 | 10594 : 32 | 6144 : 88 | 6520 : 96 | 3392 : 32 | 3392 : 32 | - | 3656 : 48 |
| No. | sseig | BEAST-C | BEAST-M |
|-----|-------|---------|---------|
|     | i,n   | i,x     | o,n     | o,x     |
|     | no switch | 2beastc |         |         |
| 24  | 8975 : 20 | 9845 : 31 | 3616 : 64 | 4256 : 64 | 2864 : 32 | 2864 : 32 | 2872 : 32 | 2872 : 32 |
| 25  | 9471 : 28 | 6505 : 21 | 3744 : 64 | 4304 : 64 | -         | 3192 : 40 | 2888 : 24 | 2888 : 24 |
| 26  | 12047 : 28 | -       | -         | 6432 : 80 | -         | 4304 : 48 | 3528 : 40 | 3528 : 40 |
| 27  | 8976 : 24 | -       | -         | 2816 : 32 | 3840 : 32 | 2656 : 24 | 2656 : 24 | -         | 3800 : 32 |
| 28  | 10800 : 24 | 9682 : 28 | 6144 : 88 | 6488 : 88 | -         | 3232 : 64 | 3544 : 32 | 3544 : 32 |
| 29  | 13344 : 32 | 11347 : 34 | 6144 : 80 | 6488 : 80 | 3360 : 32 | 3360 : 32 | -         | 4768 : 56 |
| 30  | 9338 : 21 | 10355 : 35 | 4848 : 56 | 5736 : 56 | -         | 9560 : 120 | -         | 7440 : 120 |
| 31  | 9562 : 28 | 6599 : 21 | 3760 : 64 | 4304 : 64 | 2672 : 24 | 2672 : 24 | 2712 : 24 | 2712 : 24 |
| 32  | 12288 : 32 | 11698 : 33 | -         | 6496 : 80 | 3408 : 32 | 3408 : 32 | 3608 : 32 | 3608 : 32 |
| 33  | 10157 : 28 | 6150 : 21 | 2992 : 32 | 4144 : 32 | 2640 : 24 | 2640 : 24 | 2904 : 32 | 2904 : 32 |
| 34  | 10800 : 24 | 11008 : 36 | 6144 : 88 | 6536 : 88 | -         | -         | -         | 6136 : 48 |
| 35  | 10346 : 28 | 8279 : 26 | 6144 : 80 | 6648 : 80 | 2808 : 32 | 2808 : 32 | 2840 : 32 | 2840 : 32 |
| 36  | 9317 : 28 | -       | 5136 : 64 | 5904 : 64 | 2736 : 24 | 2736 : 24 | 2712 : 24 | 2712 : 24 |
| 37  | 9289 : 28 | 5961 : 19 | 3504 : 56 | 4240 : 56 | 2888 : 48 | 3752 : 48 | 2712 : 24 | 2712 : 24 |

Note: Problems labeled by number as listed in Table 1. Value given as “-” when the given method did not find all eigenpairs in interval.