Bergische Universität Wuppertal

Fakultät für Mathematik und Naturwissenschaften

Institute of Mathematical Modelling, Analysis and Computational Mathematics (IMACM)

Preprint BUW-IMACM 20/52

Sarah Huber, Yasunori Futamura, Martin Galgon, Akira Imakura, Bruno Lang and Tetsuya Sakurai

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

November 2, 2020

http://www.imacm.uni-wuppertal.de

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

Sarah Huber[1,a], Yasunori Futamura[b], Martin Galgon[a], Akira Imakura[b], Bruno Lang[a], Tetsuya Sakurai[b]
[a] University of Wuppertal, Faculty of Mathematics and Natural Sciences
[b] University of Tsukuba, Department of Computer Science

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

I. 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 SSM [3]–[11] and FEAST [12]–[17] have been extended and further developed. These methods are in particular applicable to the generalized interior eigenproblem in the positive definite case, i.e., to finding the eigenpairs $(\lambda, \mathbf{x})$ of

$$A\mathbf{x} = B\mathbf{x}\lambda, \qquad \lambda \in I_\lambda = \left[\underline{\lambda}, \overline{\lambda}\right], \tag{1}$$

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_λ, but no other eigenvalue, and then computes approximations to integrals of the form

$$\frac{1}{2\pi i}\int_C z^k (zB - A)^{-1} BY \,\mathrm{d}z \,. \tag{2}$$

From these, an approximate subspace, U, for the desired eigenspace, X_λ, 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 [18] and solve the reduced eigenproblem

$$A_U W = B_U W \Lambda \tag{3}$$

where $A_U = U^* A U, B_U = U^* B U$ and $\tilde{X} := UW$.
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 ω_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

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

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

$$U = [U_0, U_1, U_2, ..., U_{s-1}]. \tag{4}$$

[1] E-mail: shuber@math.uni-wuppertal.de

In FEAST, only the 0^{th} moment is considered, and U is formed as

$$U \equiv U_0 = \sum_{j=1}^{q_{\text{FEAST}}} \omega_j (z_j B - A)^{-1} BY. \tag{5}$$

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, $\text{width}(U)$, is by construction $\text{width}(U) = \text{RHS}_1 \cdot s$, 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 $1/s^{\text{th}}$ of the desired number of columns of U.

SS–RR has traditionally used a large overall subspace size $\text{width}(U)$, 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 behaviour of the algorithms is in this case necessary to develop good heuristics. Having previously explored adaptivity with respect to quadrature nodes [19], 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.

II. Flexible subspace iteration

FEAST is, by definition, a subspace iteration scheme. Multiple iterations of the sequence

1: Construct subspace U with contour integration according to (5)
2: Rayleigh–Ritz extraction of eigenvalues and vectors according to (3)
3: $Y := \tilde{X}$

result in a set of vectors, $\tilde{X}$, inside (or close to) the desired eigenspace.

SSM, on the other hand, was not originally defined as an iterative scheme. The basic SS–RR roughly works as follows:

1: Construct subspace U with contour integration and moments according to (4)
2: Orthogonalize U and reduce subspace by removing (almost) linearly dependent columns
3: Rayleigh–Ritz extraction of eigenvalues and vectors according to (3) ,

without iteration. More recently, the "inner iteration" has been introduced [8],

1: $U_0 := \sum_{j=1}^{q_{\text{SSM}}} \omega_j (z_j B - A)^{-1} BY$
2: $Y := U_0$,

involving multiple passes of a set of initial vectors through (5) before the construction of the full subspace using multiple moments.

Here we consider the use of full iterations with SSM, using a linear combination of the approximate eigenvectors as the initial vectors in the next iteration, [9] $Y := \tilde{X}R$ with $R \in \mathbb{C}^{\text{width}(\tilde{X}) \times \text{RHS}_1}$. We will call this technique the "outer iteration." 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 behaviour 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 behaviour, 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 have been observed to stagnate over iterations before all residuals of the computed eigenpairs have reached the prescribed tolerance, in particular for very small tolerances. A more flexible method might therefore allow a switch from a multi-moment to single-moment scheme if stagnation 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, adaptive techniques such as the locking of 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 these two techniques will provide the main contribution in this work.

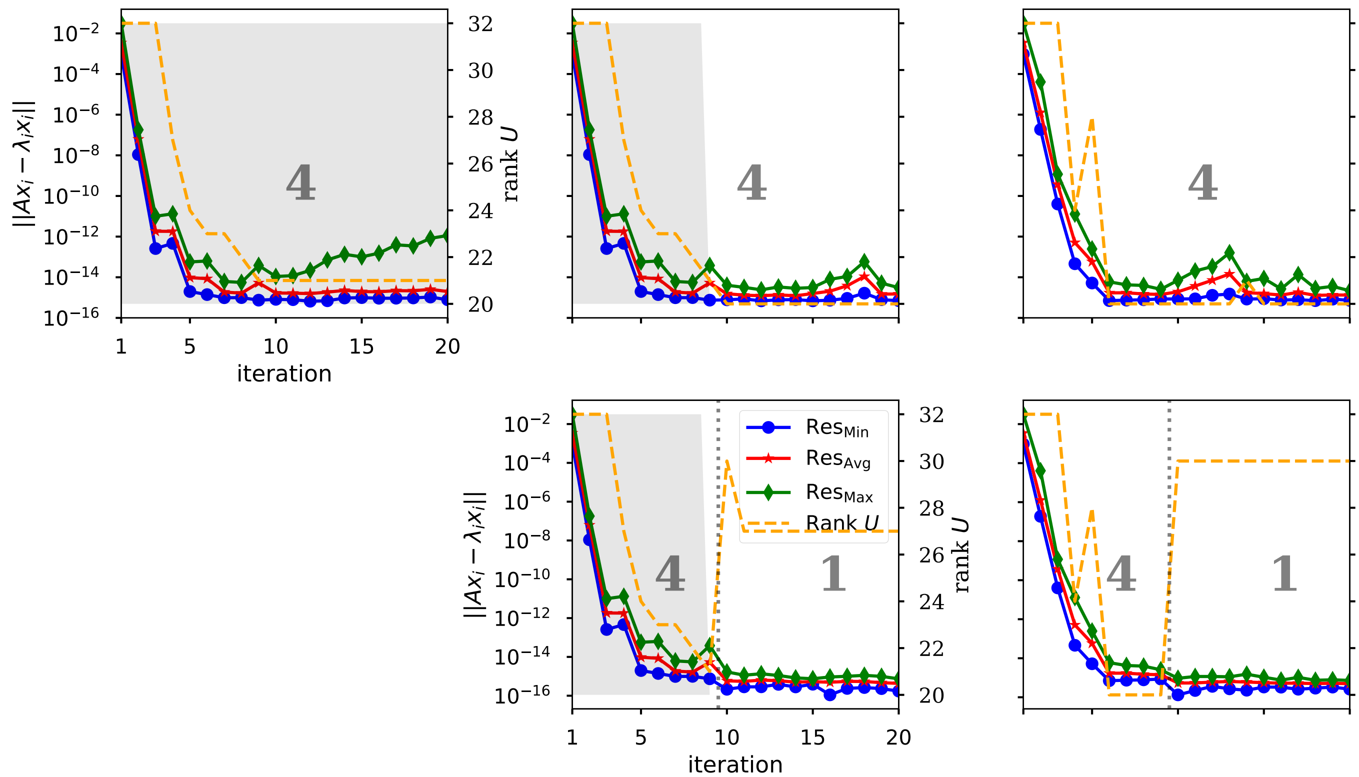


Fig. 1: Rank and convergence (maximum, minimum, and average of the 20 smallest residuals) of the contour integration-based iterative schemes for problem (6) with initial subspace size 32 and 4 moments for the first iteration. The number of moments is denoted in the background, with a vertical line indicating a change in number of moments. Background hashed for inner iterations. Bottom row: Switch from SSM-type multi-moment (`BEAST-M` implementation, cf. Section IV) to FEAST-type single-moment (`BEAST-C`, cf. Section IV) after the 9th iteration. (I.e., single moment used in 10th and subsequent iterations.) Top row: No such switching, multi-moment throughout. Left column: All inner iterations. Middle column: Inner iterations for the first 8 iterations, then outer iterations. (I.e., initial subspace in 10th iteration is constructed from approximate eigenvectors of previous iteration.) Right column: All outer iterations. Bottom left: Left blank as inner iterations are not consistent with FEAST-type single moment.

III. Inner vs. outer iterations and multi- vs. single-moment

In this section we compare the convergence behaviour of iteration schemes using either only inner iterations or only outer iterations, or switching from inner to outer after iteration 8, 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

$$A\mathbf{x}_i = \lambda_i \mathbf{x}_i, \quad \lambda_i \in I_\lambda = [-1, 1], \qquad \text{where } A = \operatorname{diag}(-2.99, -2.89, ..., 6.91) \in \mathbb{R}^{100\times 100}, \tag{6}$$

containing $m = 20$ eigenvalues in the search interval.

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 $\text{width}(U) = 32 \approx 1.5 \times m$. 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 behaviour for the inner and outer iteration types with moments, which further varies with the number of moments (and size of U_0).

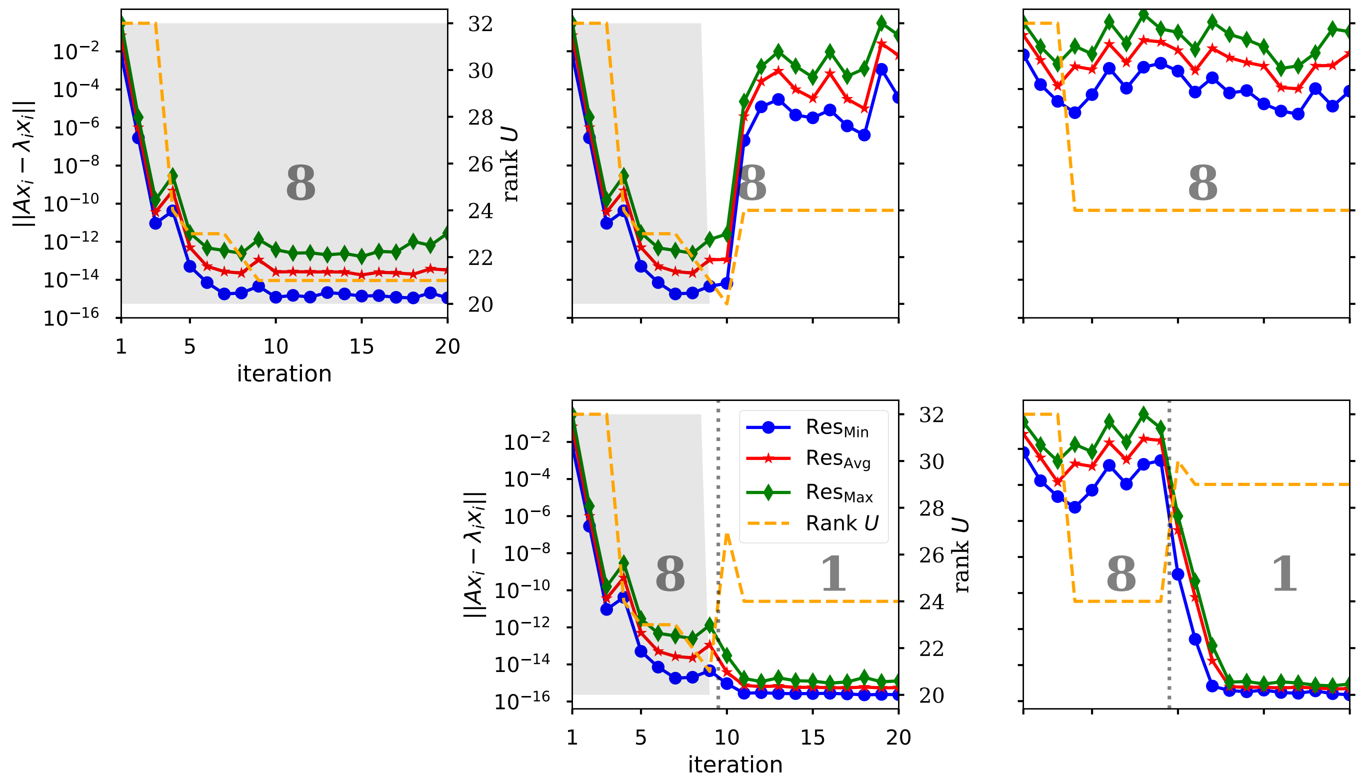


Fig. 2: As Figure 1, but with 8 moments.

A. *Inner and outer iterations*

In the top row of each figure, we observe the different behaviour for outer and inner iterations. Though computation of the approximate eigenpairs is not strictly necessary for the inner iteration, we include it in this work in order to observe intermediate convergence behaviour and utilize further advantageous strategies dependant on this behaviour. 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.

We see that for early iteration counts, convergence is noticeably improved for the inner iteration compared to the outer iteration. These effects 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.

B. *Switching from multi- to single-moment*

In the bottom row 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 $1.5\times$ the approximate number of eigenpairs, this means that more right hand sides are involved in the solution of the linear systems. We see the increase in stability of convergence with this result, for previous iterations of both the outer and inner type.

IV. The BEAST framework

`BEAST` provides a framework for subspace iteration with Rayleigh–Ritz extraction of eigenvalues and vectors [20], [21]. 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 $A\mathbf{x} = \mathbf{x}\lambda$ 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`.

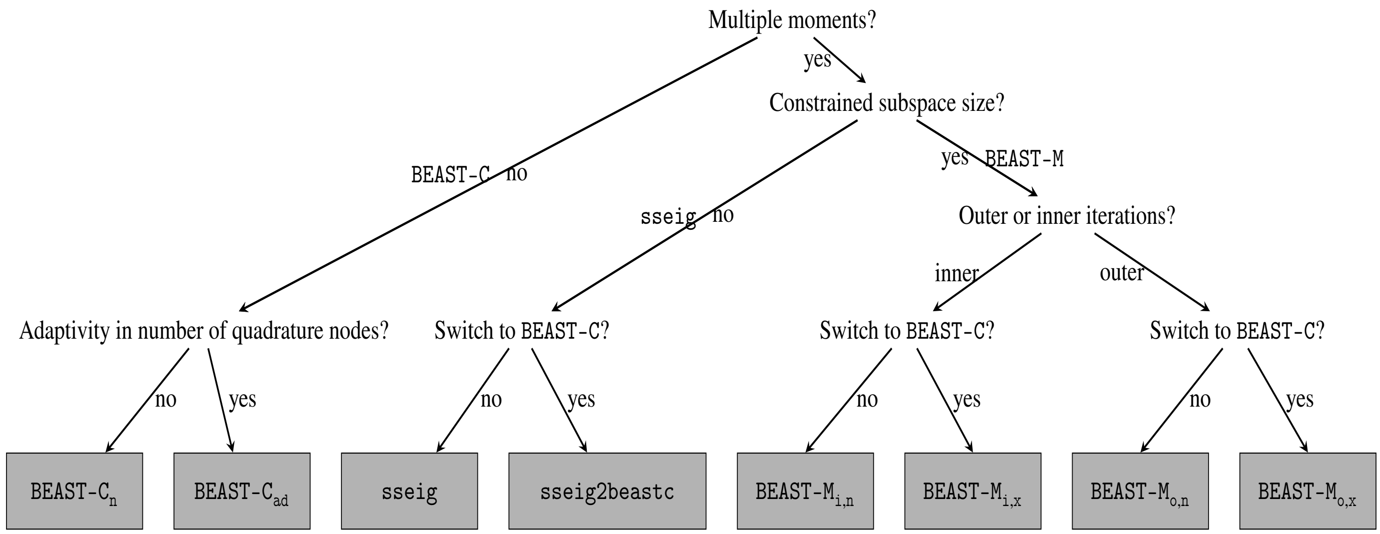


Fig. 3: Overview of all computation schemes and choices considered in section V.

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 $\texttt{BEAST-M}_{\texttt{i},*}$ and $\texttt{BEAST-M}_{\texttt{o},*}$, resp.), and we may or may not allow switching to `BEAST-C` whenever stagnation of convergence is detected ($\texttt{BEAST-M}_{*,\texttt{x}}$ and $\texttt{BEAST-M}_{*,\texttt{n}}$, resp.). Thus, $\texttt{BEAST-M}_{\texttt{o},\texttt{x}}$ 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, $r_{\text{SNC}} = \max_j\{\|A\tilde{\mathbf{x}}_j - B\tilde{\mathbf{x}}_j\tilde{\lambda}_j\|$: this residual is above tolerance$\}$, since the previous iteration; that is, $r^i_{\text{SNC}}/r^{i-1}_{\text{SNC}}$. Whenever this ratio rose above the threshold of 0.01, stagnation was determined to have set in.
- For `BEAST-C` iterations, the variant $\texttt{BEAST-C}_{\texttt{ad}}$ adjusts the number of quadrature nodes q (and the nodes and coefficients) over the iterations, whereas $\texttt{BEAST-C}_{\texttt{n}}$ does not.

Figure 3 shows the different choices for the methods, as well as SSM-based methods included in section V, and Algorithm 1 gives an overview of the computations done in each `BEAST` case.

V. Numerical experiments

Numerical results are shown for 37 test problems as detailed in Table I. The problems arise from graphene modelling [22], and the SuiteSparse Matrix Collection [23].

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 RHS_{ovl} counts over several test problems. We used an elliptical contour with eccentricity 0.1 and 16 quadrature nodes. Testing $\texttt{BEAST-M}_{\texttt{o},\texttt{x}}$ and $\texttt{BEAST-M}_{\texttt{i},\texttt{n}}$ 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 RHS_{ovl} was similar or better for Gauss–Legendre with subspace factor 1.5, and trapezoidal was superior with subspace factor 4; cf. Table II. 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 `sseig` (a Matlab implementation of SS–RR [9]), 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, RHS_{ovl}, as well as all block linear solves, BLS_{ovl}, 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 `svd` function. Small eigenproblems were solved directly using the `eig` function. In accordance with FEAST [2] we chose a subspace size $\text{width}(U) \approx 1.5 \times \tilde{m}$, 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, `BEAST-M` and `BEAST-C` began with overall subspace sizes $\text{width}(U)$ 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, $\texttt{BEAST-C}_{\texttt{ad}}$, and with the *best* fixed-order method, $\texttt{BEAST-C}_{\texttt{n}}$ (i.e., the fixed q leading to the minimum RHS_{ovl} among all q considered, such that that all

TABLE I: Sizes, number of nonzeros per row (nnzr; rounded), smallest ($\lambda_{\min}$) and largest ($\lambda_{\max}$) eigenvalues (rounded to three significant digits), and the two search intervals (with eigenvalue counts m) for 9 test matrices from graphene modelling and 10 test matrices from the GHS_indef (laser, linverse, brainpc2), PARSEC (SiH4, Si5H12, SiO), ACUSIM (Pres_Poisson), Boeing (bcsstk37), DIMACS10 (rgg_n_2_15_s0), and Andrews (Andrews) groups of the SuiteSparse Matrix Collection. [23] For each problem, the test matrix and interval were taken as A and I_λ in (1), $B = I$. The problems are also problem numbers 1:25, 27:34, 36:39 in Table 1 of Galgon et al. [20].

Name	Size	nnzr	[$\lambda_{\min}$,$\lambda_{\max}$]	Interval A	m	No.	Interval B	m	No.
laser	3002	3	$[-1.10, 4.25]$	$[-0.100, 0.357]$	307	1	$[1.0000, 4.2389]$	304	NA
SiH4	5041	34	$[-0.996, 36.8]$	$[25.0, 28.4]$	301	2	$[15.3, 16.4]$	290	3
linverse	11999	8	$[-4.70, 15.5]$	$[0.00, 0.62]$	308	4	$[2.62, 2.77]$	304	5
Pres_Poisson	14822	48	$[1.28e\text{-}5, 26.0]$	$[3.7, 10.0]$	302	6	$[1.000, 1.182]$	300	7
Si5H12	19896	37	$[-0.996, 58.6]$	$[24.2, 24.7]$	320	8	$[41, 42]$	274	9
bcsstk37	25503	45	$[-7.04e\text{-}5, 8.41e7]$	$[8.0e5, 9.3e5]$	297	10	$[1.15e7, 1.30e7]$	305	11
brainpc2	27607	6	$[-2000, 4460]$	$[275, 345]$	313	12	$[1900, 1920]$	309	13
rgg_n_2_15_s0	32768	10	$[-5.12, 17.4]$	$[-2.00, -1.95]$	282	14	$[5.0, 5.5]$	337	15
SiO	33401	39	$[-1.67, 84.3]$	$[32.0, 32.4]$	316	16	$[57.0, 57.8]$	283	17
Andrews	60000	13	$[3.64e\text{-}16, 36.5]$	$[21.0, 21.4]$	300	18	$[11.20, 11.26]$	298	19
GraI-1k	1152	13	$[-3.43, 2.78]$	$[-0.7575, 1.1025]$	301	20	$[0.42, 1.58]$	301	21
GraI-11k	11604	13	$[-3.43, 2.78]$	$[-0.0475, 0.3925]$	298	22	$[0.935, 1.065]$	289	23
GraI-119k	119908	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	11604	13	$[-3.43, 2.79]$	$[-0.1375, 0.4825]$	299	28	$[0.949, 1.051]$	299	29
GraII-119k	119908	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	11604	12	$[-3.35, 2.73]$	$[-0.0975, 0.4425]$	305	34	$[0.952, 1.048]$	319	35
GraIII-119k	119908	13	$[-3.43, 2.78]$	$[0.1395, 0.2055]$	310	36	$[0.996, 1.004]$	311	37

TABLE II: Average $\mathrm{RHS}_{\mathrm{ovl}}$ with different quadrature rules for 31 of the 37 problems listed in Table I. Problems for which all not schemes found all eigenpairs (7,14,20,26,34) are not included in the average.

Iteration Type	Quadrature Rule	Subspace Factor	Average $\mathrm{RHS}_{\mathrm{ovl}}$
BEAST-$\mathtt{M_{i,x}}$	Gauss Legendre	1.5	3587
BEAST-$\mathtt{M_{i,x}}$	Trapezoidal	1.5	3514
BEAST-$\mathtt{M_{i,x}}$	Gauss Legendre	4.0	5229
BEAST-$\mathtt{M_{i,x}}$	Trapezoidal	4.0	4545
BEAST-$\mathtt{M_{o,x}}$	Gauss Legendre	1.5	3201
BEAST-$\mathtt{M_{o,x}}$	Trapezoidal	1.5	3723
BEAST-$\mathtt{M_{o,x}}$	Gauss Legendre	4.0	5218
BEAST-$\mathtt{M_{o,x}}$	Trapezoidal	4.0	4366

TABLE III: Number of problems for each method for which not all eigenpairs were found (from a total of 37 problems).

Solver	Total failed
BEAST-$\mathtt{C_{ad}}$	7
sseig	7
sseig2beastc	1
BEAST-$\mathtt{M_{i,n}}$	8
BEAST-$\mathtt{M_{i,x}}$	2
BEAST-$\mathtt{M_{o,n}}$	8
BEAST-$\mathtt{M_{o,x}}$	1

Algorithm 1 BEAST framework, including BEAST−M and BEAST−C algorithmic variants. The option of adaptive quadrature node choice for BEAST−C is shown as BEAST−$\mathtt{C_{ad}}$. The option of adaptive method switching for BEAST−M is shown as BEAST−$\mathtt{M}_{*,\mathtt{x}}$. The option of inner or outer iterations for BEAST−M are shown as BEAST−$\mathtt{M}_{\mathtt{o},*}$ and BEAST−$\mathtt{M}_{\mathtt{i},*}$.

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, ω_j)
$Y \in \mathbb{C}^{n \times \mathrm{RHS}_1}$

while not converged **do**
  if in BEAST−$\mathtt{C_{ad}}$ mode **then**
    Choose q, z_j, ω_j ▷ otherwise use parameters as provided with inputs
  end if
  Construct subspace U according to (4) (BEAST−M mode) or (5) (BEAST−C mode)
  if in BEAST−$\mathtt{M}_{\mathtt{i},*}$ mode **then**
    Set $Y := U_0$ ▷ inner iteration
  end if
  Compute singular values of U
  if in BEAST−M mode **then**
    Orthogonalize U
  end if
  Resize subspace
  Rayleigh–Ritz extraction of eigenvalues Λ and eigenvectors $\tilde{X}$ according to (3)
  Orthogonalize $\tilde{X}$ against locked eigenpairs, lock newly converged eigenpairs
  if in BEAST−$\mathtt{M}_{*,\mathtt{x}}$ mode and stagnation has been detected **then**
    switch to BEAST−C
  end if
  if in BEAST−$\mathtt{M}_{\mathtt{o},*}$ mode **then**
    Set $Y := \tilde{X}R$ with $R \in \mathbb{C}^{\mathrm{width}(U) \times \mathrm{RHS}_1}$
  else if in BEAST−C mode **then**
    Set $Y := \tilde{X}$
  end if
end while

eigenpairs were found). In the adaptive case, as described by Galgon et al. [20], q is increased by a factor of $\sqrt{1.5}$ or at least by 1 if the smallest non-converged residual decreases by a factor of $\left(10^{-1.5}, 10^{-0.75}\right]$ and by a factor of 1.5, and by at least 1, if the decrease factor is greater than $10^{-0.75}$. A comparison was also made to sseig, which was also run with 4 moments, and the subspace was allowed to grow expansively, starting with $\mathrm{width}(U_0) = 16$ right hand sides and growing to a maximum of 256 for the problems considered, for an overall subspace size of $\mathrm{width}(U) = 1024$. The final comparison was with sseig2beastc, 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 and sseig2beastc will converge quickly, i.e., 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 III. 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 observe that all methods obtaining the minimum, or near minimum number of single linear solves, $\mathrm{RHS}_{\mathrm{ovl}}$, included multiple moments, and the difference in $\mathrm{RHS}_{\mathrm{ovl}}$ between methods involving and not involving moments is substantial. We also observe that methods involving a switch to BEAST−C behaved more robustly, with fewer failures to find the minimum number of linear solves, $\mathrm{RHS}_{\mathrm{ovl}}$. We see that both BEAST−$\mathtt{M}_{*,\mathtt{x}}$ and BEAST−$\mathtt{M}_{*,\mathtt{n}}$ outperformed sseig and BEAST−C, with BEAST−$\mathtt{M}_{*,\mathtt{n}}$ and BEAST−$\mathtt{M}_{*,\mathtt{x}}$ having the same minimum $\mathrm{RHS}_{\mathrm{ovl}}$ in most cases.

When switching was allowed, BEAST−$\mathtt{M}_{\mathtt{o},\mathtt{x}}$ and BEAST−$\mathtt{M}_{\mathtt{i},\mathtt{x}}$ behaved robustly, failing to find all eigenvalues for only one and two problems respectively. In comparison, BEAST−$\mathtt{C_{ad}}$, BEAST−$\mathtt{M}_{*,\mathtt{n}}$, 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 behaviour, 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

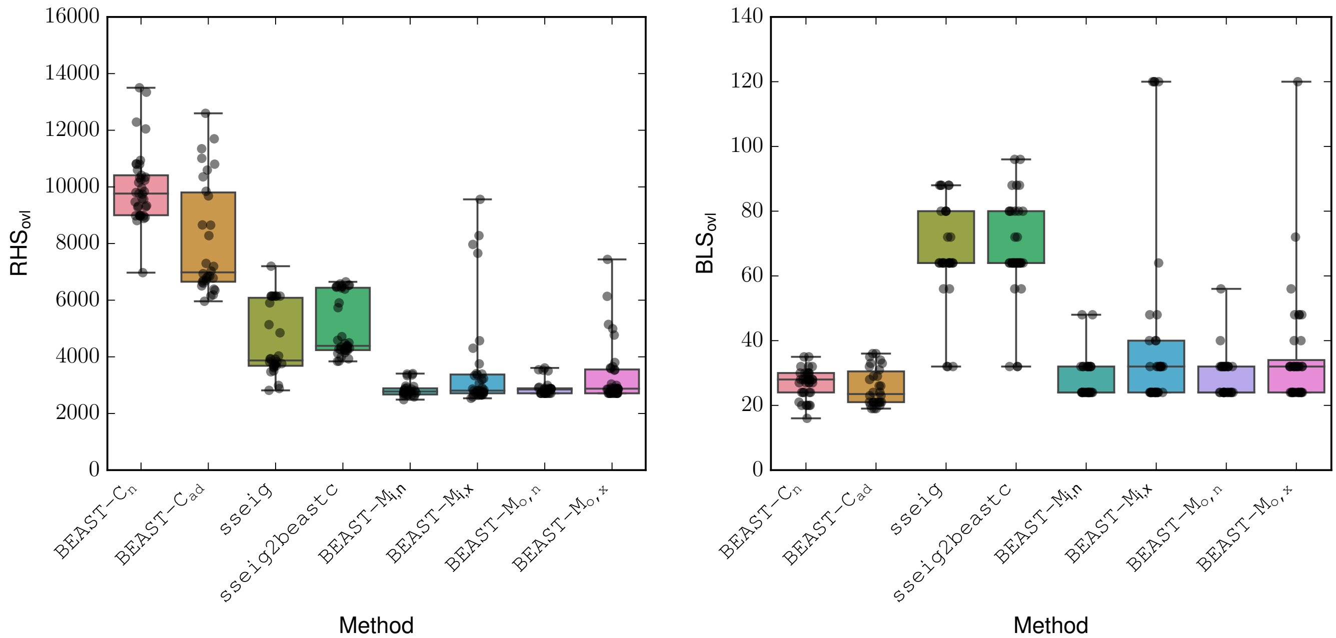


Fig. 4: $\mathrm{RHS_{ovl}}$ (left) and $\mathrm{BLS_{ovl}}$ (right) for different solution techniques. Values are only shown for systems where all eigenpairs were found with a given method. Each box extends from the first to third quartiles of the respective data, with the median (second quartile) shown as a horizontal black line in the box. The whiskers cover the entire range of observations. Distinct counts for each problem are visible as translucent circles. Full numerics are given in the appendix.

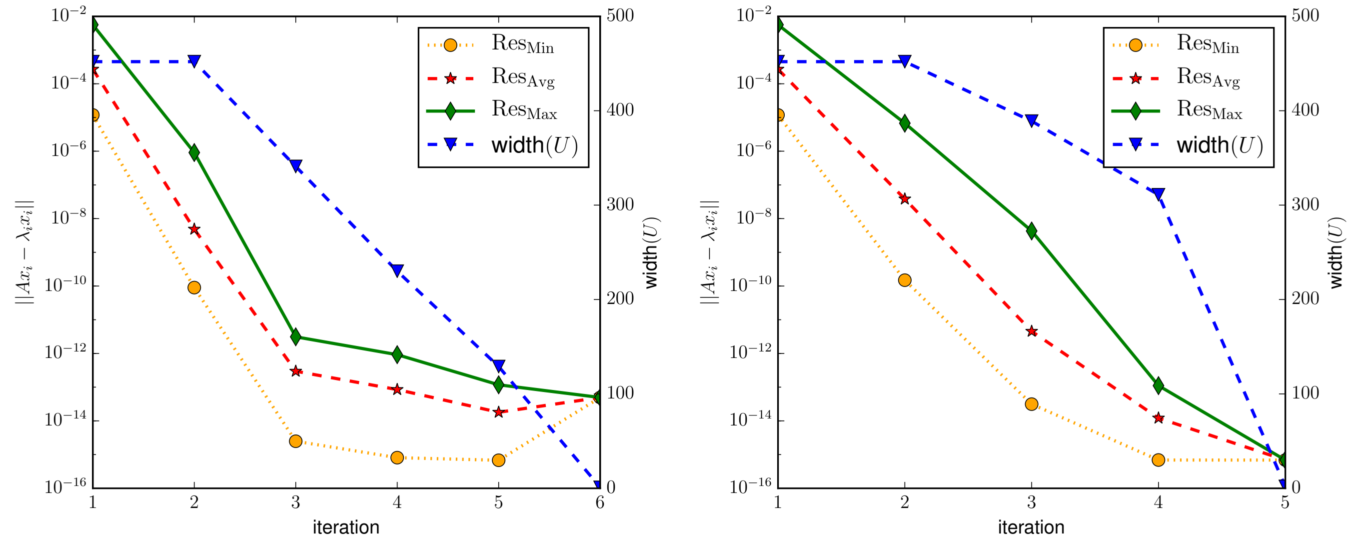
a considerable expense. We also observe the increased $\mathrm{BLS_{ovl}}$ for BEAST-M compared to BEAST-C and sseig, due to a higher number of iterations. We predict, and demonstrate, in the following tests, that the reduction of $\mathrm{RHS_{ovl}}$ is enough to offset this cost, but acknowledge that 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-$\mathrm{M_{o,*}}$ and BEAST-$\mathrm{M_{i,*}}$, we see similar behaviour between the two methods, with a few larger outliers in $\mathrm{RHS_{ovl}}$ for BEAST-$\mathrm{M_{i,x}}$. 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-$\mathrm{M_{i,x}}$, which started out with much faster convergence than BEAST-$\mathrm{M_{o,x}}$.

Out of the various methods considered so far, we choose to highlight BEAST-$\mathrm{M_{o,x}}$ 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 $\mathrm{RHS_{ovl}}$ corresponds to savings in time. We compared parallelized, distributed memory implementations of BEAST-C (specifically BEAST-$\mathrm{C_n}$) and BEAST-$\mathrm{M_{o,x}}$ for two larger systems. The scalable solver strumpack [24] was used to solve all linear systems directly, and the kernel library ghost [25] 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 IV. 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 $1.5 \times \tilde{m}$, and locking was enabled. BEAST-$\mathrm{M_{o,x}}$ began with 4 moments and was allowed to switch to BEAST-C when stagnation was detected.

In Figure 6, we again see the reduction in $\mathrm{RHS_{ovl}}$ of BEAST-$\mathrm{M_{o,x}}$ compared to 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 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 $\mathrm{RHS_{ovl}}$ for BEAST-M compared to BEAST-C was not so dramatic, and at least one additional iteration was required, as can be seen in the $\mathrm{BLS_{ovl}}$ count. However, even with the additional iteration(s), BEAST-$\mathrm{M_{o,x}}$ was still faster than BEAST-C.



(a) BEAST-$\text{M}_{\text{i,x}}$

(b) BEAST-$\text{M}_{\text{o,x}}$

Fig. 5: Convergence and subspace size $\text{width}(U)$ with BEAST-$\text{M}_{\text{i,x}}$ and BEAST-$\text{M}_{\text{o,x}}$ for Problem 4, Table 1 in Galgon et al. [20].

TABLE IV: Sizes, average number of nonzeros per row (nnzr; rounded), Approximate smallest (λ_{min}) and largest (λ_{max}) eigenvalues, search interval, starting approximate eigenvalue count $\tilde{m}$, and number of eigenvalues found $\hat{m}$ for test matrices from graphene [22] and topological insulator modelling [26]. Related matrices may be generated from the ScaMaC library. [27] The spectral properties of matrices of these forms with regard to computation have been previously studied. [28] For each problem, the test matrix and interval were taken as A and I_λ in (1), $B = I$.

Name	Size	nnzr	$[\lambda_{\text{min}}, \lambda_{\text{max}}]$	Interval	$\tilde{m}$	$\hat{m}$
Graph16M	16000000	4	$[-3.0, 3.0]$	$[-0.0025, 0.0025]$	320	318
Topi1M	1638400	12	$[-4.8, 4.8]$	$[-0.06, 0.06]$	120	116

VI. 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, particularily 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 heuristical 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 best strategies for choosing the number of moments on a given iteration, and the choice of starting vectors. 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 [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.

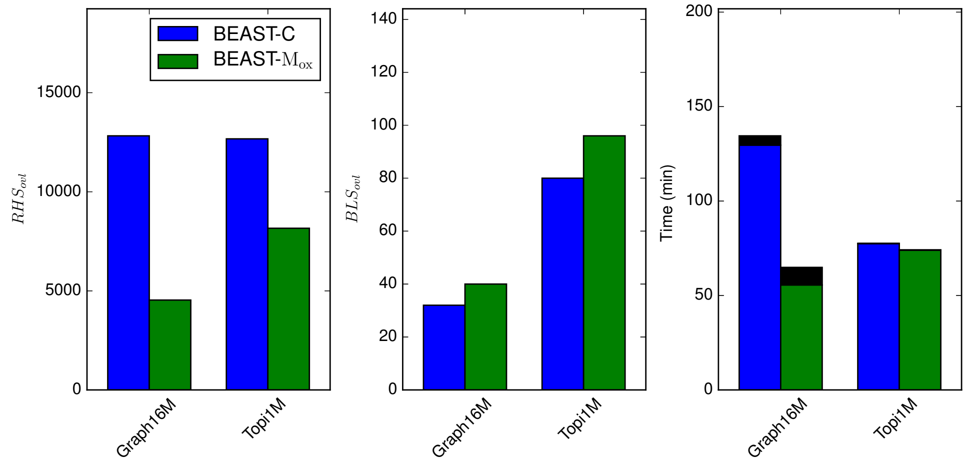


Fig. 6: Linear solve count and timings of `BEAST-M`$_{\texttt{o,x}}$ vs. `BEAST-C`. In right most figure, time for building U shown in color, time for all other components shown in black.

Acknowledgements

The present study has been supported in part by the Deutsche Forschungsgemeinschaft 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)

References

[1] T. Sakurai and H. Sugiura, "A projection method for generalized eigenvalue problems using numerical integration," *Journal of computational and applied mathematics*, vol. 159, no. 1, pp. 119–128, 2003.
[2] E. Polizzi, "Density-matrix-based algorithm for solving eigenvalue problems," *Physical Review B*, vol. 79, no. 11, p. 115112, 2009.
[3] T. Sakurai, H. Tadano *et al.*, "CIRR: a Rayleigh-Ritz type method with contour integral for generalized eigenvalue problems," *Hokkaido mathematical journal*, vol. 36, no. 4, pp. 745–757, 2007.
[4] A. Imakura, L. Du, and T. Sakurai, "Relationships among contour integral-based methods for solving generalized eigenvalue problems," *Japan Journal of Industrial and Applied Mathematics*, vol. 33, no. 3, pp. 721–750, 2016.
[5] ——, "A block Arnoldi-type contour integral spectral projection method for solving generalized eigenvalue problems," *Applied Mathematics Letters*, vol. 32, pp. 22–27, 2014.
[6] T. Ikegami, T. Sakurai *et al.*, "Contour integral eigensolver for non-hermitian systems: a Rayleigh–Ritz-type approach," *Taiwanese Journal of Mathematics*, vol. 14, no. 3A, pp. 825–837, 2010.
[7] T. Ikegami, T. Sakurai, and U. Nagashima, "A filter diagonalization for generalized eigenvalue problems based on the Sakurai–Sugiura projection method," *Journal of Computational and Applied Mathematics*, vol. 233, no. 8, pp. 1927–1936, 2010.
[8] A. Imakura, L. Du, and T. Sakurai, "Error bounds of Rayleigh–Ritz type contour integral-based eigensolver for solving generalized eigenvalue problems," *Numerical Algorithms*, vol. 71, no. 1, pp. 103–120, Jan 2016. [Online]. Available: https://doi.org/10.1007/s11075-015-9987-4
[9] T. Sakurai, Y. Futamura, and H. Tadano, "Efficient parameter estimation and implementation of a contour integral-based eigensolver," *Journal of Algorithms & Computational Technology*, vol. 7, no. 3, pp. 249–269, 2013. [Online]. Available: https://doi.org/10.1260/1748-3018.7.3.249
[10] T. Sakurai, Y. Futamura, A. Imakura, and T. Imamura, "Scalable eigen-analysis engine for large-scale eigenvalue problems," in *Advanced Software Technologies for Post-Peta Scale Computing*. Springer, 2019, pp. 37–57.
[11] A. Imakura and T. Sakurai, "Block SS–CAA: a complex moment-based parallel nonlinear eigensolver using the block communication-avoiding Arnoldi procedure," *Parallel Computing*, vol. 74, pp. 34–48, 2018.
[12] P. T. Peter Tang and E. Polizzi, "FEAST as a subspace iteration eigensolver accelerated by approximate spectral projection," *SIAM Journal on Matrix Analysis and Applications*, vol. 35, no. 2, pp. 354–390, 2014.
[13] S. Guettel, E. Polizzi, P. Tang, and G. Viaud, "Zolotarev quadrature rules and load balancing for the FEAST eigensolver," *SIAM Journal on Scientific Computing*, vol. 37, 07 2014.
[14] G. Yin, R. H. Chan, and M.-c. Yeung, "A FEAST algorithm for generalized non-Hermitian eigenvalue problems," *arXiv preprint arXiv:1404.1768*, 2014.
[15] J. Kestyn, E. Polizzi, and P. T. Peter Tang, "FEAST eigensolver for non-hermitian problems," *SIAM Journal on Scientific Computing*, vol. 38, no. 5, pp. S772–S799, 2016.
[16] B. Gavin, A. Międlar, and E. Polizzi, "FEAST eigensolver for nonlinear eigenvalue problems," *Journal of computational science*, vol. 27, pp. 107–117, 2018.
[17] B. Gavin and E. Polizzi, "Krylov eigenvalue strategy using the FEAST algorithm with inexact system solves," *Numerical Linear Algebra with Applications*, vol. 25, no. 5, p. e2188, 2018.
[18] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. Society for Industrial and Applied Mathematics, 2003. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9780898718003
[19] M. Galgon, L. Krämer, and B. Lang, "Improving projection-based eigensolvers via adaptive techniques," *Numerical Linear Algebra with Applications*, pp. e2124–n/a, 2017, e2124 nla.2124. [Online]. Available: http://dx.doi.org/10.1002/nla.2124
[20] M. Galgon, L. Krämer, B. Lang, A. Alvermann, H. Fehske, and A. Pieper, "Improving robustness of the FEAST algorithm and solving eigenvalue problems from graphene nanoribbons," *PAMM*, vol. 14, no. 1, pp. 821–822, 2014.
[21] M. Galgon, L. Krämer, J. Thies, A. Basermann, and B. Lang, "On the parallel iterative solution of linear systems arising in the FEAST algorithm for computing inner eigenvalues," *Parallel Computing*, vol. 49, pp. 153–163, 2015.
[22] A. H. Castro Neto, F. Guinea, N. M. R. Peres, K. S. Novoselov, and A. K. Geim, "The electronic properties of graphene," *Rev. Mod. Phys.*, vol. 81, pp. 109–162, Jan 2009. [Online]. Available: https://link.aps.org/doi/10.1103/RevModPhys.81.109

[23] T. A. Davis and Y. Hu, "The University of Florida sparse matrix collection," *ACM Trans. Math. Softw.*, vol. 38, no. 1, pp. 1:1–1:25, Dec. 2011. [Online]. Available: http://doi.acm.org/10.1145/2049662.2049663

[24] F.-H. Rouet, X. S. Li, P. Ghysels, and A. Napov, "A distributed-memory package for dense hierarchically semi-separable matrix computations using randomization," *ACM Trans. Math. Softw.*, vol. 42, no. 4, pp. 27:1–27:35, Jun. 2016. [Online]. Available: http://doi.acm.org/10.1145/2930660

[25] M. Kreutzer, J. Thies, M. Röhrig-Zöllner, A. Pieper, F. Shahzad, M. Galgon, A. Basermann, H. Fehske, G. Hager, and G. Wellein, "GHOST: building blocks for high performance sparse linear algebra on heterogeneous systems," *International Journal of Parallel Programming*, vol. 45, no. 5, pp. 1046–1072, 2017.

[26] M. Z. Hasan and C. L. Kane, "Colloquium: Topological insulators," *Rev. Mod. Phys.*, vol. 82, pp. 3045–3067, Nov 2010. [Online]. Available: https://link.aps.org/doi/10.1103/RevModPhys.82.3045

[27] A. Alvermann *et al.*, "a scalable matrix collection," available at https://bitbucket.org/essex/matrixcollection.

[28] A. Pieper, M. Kreutzer, A. Alvermann, M. Galgon, H. Fehske, G. Hager, B. Lang, and G. Wellein, "High-performance implementation of chebyshev filter diagonalization for interior eigenvalue computations," *J. Comp. Phys.*, vol. 325, pp. 226–243, 11 2016.

Appendix
Detailed results for Figure 4

TABLE V: Matlab results $\mathrm{RHS}_{\mathrm{ovl}} : \mathrm{BLS}_{\mathrm{ovl}}$. Problems labelled by number as listed in Table I. Value given as "-" when the given method did not find all eigenpairs in interval.

	BEAST-C		sseig		BEAST-M			
No.	n	ad	no switch	2beastc	i,n	i,n	o,n	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 : 64	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	6576 : 96	3336 : 32	3336 : 32	3504 : 56	3584 : 48
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
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
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