Martin Galgon, Lukas Krämer, Bruno Lang

Adaptive choice of projectors in projection based eigensolvers

January 30, 2015

Adaptive choice of projectors in projection based eigensolvers

Martin Galgon, Lukas Krämer, Bruno Lang[*][†]

January 30, 2015

We present a general framework for algorithms for the solution of Hermitian eigenvalue problems, where eigenvalues in a given interval are sought. One instance of this framework is Polizzi's FEAST algorithm [E. Polizzi: Density-matrix-based algorithm for solving eigenvalue problems. *Phys. Rev. B* 2009; **79**:115112], which is based on numerical integration. Another instance is based on polynomial approximation. We propose adaptive strategies for the choice of the polynomial degree and tolerance of the linear solver, respectively. Numerical experiments reveal that these strategies are able to improve the robustness of the resulting methods, while at the same time achieving near-to-optimum efficiency.

Keywords. Eigenvalue method; projection method; Chebyshev approximation; FEAST; numerical integration, iterative linear solver

1 Introduction

Matrix eigenvalue problems arise in a multitude of applications, ranging from physics to chemistry to mechanical engineering, to name just a few [7, 17, 3]. Here the eigenvalues typically represent energy levels or resonances, and the associated eigenvectors describe the corresponding states or "modes". Often the matrices are very large and sparse, and therefore methods based on factorizations (such as variants of the QR algorithm; see, e. g., [24]) cannot be applied.

In these cases one often does not need to know *all* eigenvalues and associated eigenvectors. Some applications require the knowledge of a fixed number of eigenvalues closest to a prescribed "target" $\tau \in \mathbb{C}$ or the smallest or largest ones, whereas in other

[*]Bergische Universität Wuppertal, Fachbereich C – Mathematik und Naturwissenschaften, 42097 Wuppertal, Germany, E-mail: lang@math.uni-wuppertal.de

[†]This work was supported by the Deutsche Forschungsgemeinschaft through the priority programme 1648 "Software for Exascale Computing" (SPPEXA) under project ESSEX

applications all eigenvalues in a given "band" must be determined. The latter situation is addressed in the present paper. More precisely, let the equation

$$\mathsf{AX} = \mathsf{X}\Lambda \tag{1}$$

be given, where $\mathsf{A} \in \mathbb{C}^{n\times n}$ is a Hermitian matrix. The matrix $\mathsf{X} \in \mathbb{C}^{n\times m}$, $m \le n$, is a matrix whose columns are formed by eigenvectors of A, the corresponding eigenvalues are supposed to be on the diagonal of the diagonal matrix Λ. We suppose that the eigenvalues in the real interval $I_\lambda = [\underline{\lambda}, \overline{\lambda}]$ are sought.

Methods for the solution of (1) under these circumstances typically aim at computing an approximation to the (orthogonal) spectral projector P_Λ onto the space $\operatorname{span}(\mathsf{X})$ corresponding to the eigenvalues collected in Λ. The projector is not computed explicitly but rather its effect on a set of test vectors Y, resulting in a basis $\widetilde{\mathsf{U}}$ of the so called search space in which the eigenvectors are sought.

A widely used class of such methods is based on Krylov subspaces, i. e., those of Lanczos or Arnoldi type [2, 14, 23]. Similar from a conceptual point of view are methods of Jacobi–Davidson type [22]. Krylov and Jacobi–Davidson type methods form the space $\operatorname{span}(\widetilde{\mathsf{U}})$ incrementally, i. e., column by column.

Recently, methods based on spectral projectors which are not of Krylov or Jacobi–Davidson type have been studied. They include those based on numerical integration [16, 20, 11] or rational approximation [10]. Another class of methods, known for a long time, is based on polynomial approximation, e. g., [21, 27]. These methods have in common that they aim at forming the whole space $\operatorname{span}(\widetilde{\mathsf{U}})$ together. In the sequel, we will refer to such a method as a *projection based* method.

All methods mentioned before aim at approximating the eigenspace $\operatorname{span}(\mathsf{X})$. The algorithms can be implemented such that they access the matrix only via matrix–vector products. This makes them amenable for easy parallelization, supposed a suitable library for the sparse matrix–vector products is available. Such techniques are also developed in the context of the ESSEX project [1].

The article is structured as follows. Section 2 deals with the general structure of projection based algorithms for eigenvalue problems. In Section 2.1 we review the polynomial approach, in Section 2.2 the numerical integration approach. In Section 3, we introduce the methodology for the adaptive approaches, accompanied by numerical results. Section 4 presents the resulting algorithm and concludes the article.

2 Structure of projection based algorithms

Having an approximation to the projector P_Λ at hand (formed by a Jacobi–Davidson, polynomial or integration method), a simple eigenvalue algorithm can be formulated. It makes use of a suitable Rayleigh–Ritz approach [24]. The prototype of the arising algorithm, called ProjAlg in the following, is listed in Algorithm 2.1. It will be substantiated later. Note that P_Λ is not needed explicitly, but only its product with a set of test vectors.

Typically, the computation in line 3 is done by the same rule in every iteration of the algorithm. However, there is no need to do so. The projector P_Λ can be chosen *dynamically* in each iteration of Algorithm 2.1. In practice this amounts to choosing certain parameters of the linear solver, the integration scheme or the polynomial approximation dynamically. The reasons for doing so are efficiency and robustness, as will be shown later. For instance, the required polynomial degree is not known in advance, but by increasing the degree dynamically one can achieve an overall workload that is close to the lowest possible. Sakurai and co-workers also worked on techniques for parameter estimation in integration based algorithms [19].

Algorithm 2.1 Skeleton of projection based algorithm ProjAlg

```
Input: An interval $I_\lambda = [\underline{\lambda}, \overline{\lambda}]$ and an initial estimate $\widetilde{m}$ of the number of eigenvalues
    in $I_\lambda$.
Output: $\hat{m} \le m$ eigenpairs with eigenvalues in $I_\lambda$ ($m$ is chosen somewhat larger than
    $\widetilde{m}$).
1: Choose $\mathsf{Y} \in \mathbb{C}^{n\times m}$ of rank $m$.
2: while not converged do
3:     Compute an approximation $\widetilde{\mathsf{U}}$ to
           $\mathsf{U} := \mathsf{P}_\Lambda \mathsf{Y}.$
4:     Form the Rayleigh quotients $\mathsf{A}_{\widetilde{\mathsf{U}}} := \widetilde{\mathsf{U}}^\star \mathsf{A}\widetilde{\mathsf{U}}$, $\mathsf{B}_{\widetilde{\mathsf{U}}} := \widetilde{\mathsf{U}}^\star \widetilde{\mathsf{U}}$.
5:     Solve the size-$m$ generalized eigenproblem $\mathsf{A}_{\widetilde{\mathsf{U}}}\widetilde{\mathsf{W}} = \mathsf{B}_{\widetilde{\mathsf{U}}}\widetilde{\mathsf{W}}\widetilde{\Lambda}$.
6:     Compute the approximate Ritz pairs $(\widetilde{\Lambda}, \widetilde{\mathsf{X}} := \widetilde{\mathsf{U}} \cdot \widetilde{\mathsf{W}})$.
7:     Check for convergence and set $\mathsf{Y} := \widetilde{\mathsf{X}}$.
```

The goal of this paper is to identify and study potential for adaptivity in such a projection based algorithm. We study two different ways of approximating the projector P_Λ, namely the approximation by polynomials and numerical integration of the resolvent where the occurring linear systems are solved iteratively. The dynamic in the computation of P_Λ then is reached by choosing respectively the polynomial degree or the accuracy of the linear solver adaptively. Besides the described adaptivity approach, we study the effects of "locking" converged eigenpairs (i. e., they are saved and not further considered in the iteration) and the resulting effect on the orthogonality of the eigenvectors. Further, we include schemes for reducing the subspace dimension to a reasonable value.

2.1 Polynomial approximation

A simple and well established way to approximate the projector P_Λ in line 3 of Algorithm 2.1 is by polynomials. For a given integer $d > 0$, a polynomial p_d of degree d is constructed that should fulfill $p_d(\mathsf{A}) \approx \mathsf{P}_\Lambda$ as well as possible. The polynomial p_d itself has to be chosen such that it approximates the characteristic function χ_{I_λ} of the search interval I_λ,

$$t \mapsto \chi_{I_\lambda}(t) := \begin{cases} 1, & t \in I_\lambda \\ 0, & t \notin I_\lambda \end{cases}. \tag{2}$$

The matrix $p_d(\mathsf{A})$ then has the same eigenvectors as A, where those belonging to eigenvalues of A inside I_λ [outside I_λ, resp.] belong to eigenvalues ≈ 1 [≈ 0] of $p_d(\mathsf{A})$. Typically, Chebyshev polynomials [15] are used to approximate (2). These polynomials have been used for a long time in matrix computations due to their ability to magnify certain parts of the spectrum. For examples, see [6, 21]. The approximation of (2) then takes the form

$$\chi_{I_\lambda}(t) \approx C_d(t) := \sum_{\ell=0}^{d} c_\ell T_\ell(t), \tag{3}$$

where T_ℓ denotes the Chebyshev polynomial of first kind [15] of degree ℓ and c_ℓ is a certain coefficient. Other orthogonal polynomials such as Legendre polynomials can also be used but are not as effective as Chebyshev polynomials [12]. Since the approximation (3) only works on the interval $[-1, 1]$, the spectrum of the matrix and the interval boundaries have to be transformed to this interval via a simple linear transformation [6]. In the sequel, we suppose that this preprocessing step has already been done. The coefficients c_ℓ in (3) can be computed via the formula

$$c_\ell = \begin{cases} \dfrac{\arccos(\underline{\lambda}) - \arccos(\overline{\lambda})}{\pi}, & \ell = 0 \\ \frac{2}{\ell\pi}(\sin(\ell \arccos(\underline{\lambda})) - \sin(\ell \arccos(\overline{\lambda}))), & \ell > 0 \end{cases};$$

see, e. g., [6] for a derivation of these coefficients. In order to suppress the unwanted oscillations of C_d near the boundaries of I_λ, the coefficients c_ℓ can be multiplied by certain other coefficients g_ℓ. There are several choices for these coefficients, we will make use of those called Jackson coefficients [21, 12, 26]. The polynomials T_ℓ fulfill the three term recursion

$$T_0(t) = 1,\ T_1(t) = t,\ T_{\ell+1}(t) = 2tT_\ell(t) - T_{\ell-1}(t).$$

Hence, computing $C_d(\mathsf{A})\mathsf{y}$ for a vector y requires only d matrix–vector multiplications with A.

In [12], a numerical study concerning the use of polynomials in the projection based algorithm was conducted. The parameters included different kinds of factors g_ℓ, locations of the search interval, matrix sizes and polynomial degrees. It turned out that the Jackson coefficients mentioned above work best and that the necessary polynomial degree grows with the matrix size and nearness of the search interval to the center of $[-1, 1]$. These findings can be found, to some extent, also in [21].

The problem still is the choice of the polynomial degree d. Although some facts about possible degrees are known in theory [12, 21], it is not known in advance how it should be chosen. If it is chosen too small, Algorithm 2.1 might not deliver all wanted eigenpairs to the desired accuracy. Chosen too large, computation time is wasted. The remedy is to use an adaptive approach, increasing d over the iterations of Algorithm 2.1. This amounts to choosing different (approximate) projectors P_Λ in line 3 of the algorithm. The approach was briefly described in [12], here we will present a further study. Increasing the degree d can lead to convergence if the initial value of d was chosen too low and it sometimes even can decrease the overall number of matrix–vector multiplications. The number of matrix–vector multiplications $d \cdot m$ in each iteration of Algorithm 2.1 is a useful quantity to assess the runtime of the algorithm. This is due to the fact that most of the operations of ProjAlg with polynomial approximation are spent in matrix–vector products. (Here, m denotes the number of columns in the "current" Y, which may be lower than the initial m; see Sections 3.1 and 3.2.)

2.2 Numerical integration

Another way of approximating the projector P_Λ from line 3 of Algorithm 2.1 is by using numerical integration. Let $\mathcal{C}$ be a curve in the complex plane surrounding the interval I_λ, such that no other eigenvalues than those in I_λ reside in the interior of $\mathcal{C}$. Then, it can be shown that

$$\mathsf{P}_\Lambda = \frac{1}{2\pi\mathbf{i}} \int_{\mathcal{C}} (z\mathsf{I} - \mathsf{A})^{-1} \mathrm{d}z, \tag{4}$$

i. e., the integral is exactly the projector onto the eigenspace belonging to the eigenvalues Λ. Hence, it is reasonable to approximate (4) with a numerical integration scheme (see, e. g., [5]), resulting in an approximate projector. This approach was, for instance, followed in [16, 20]. The integral in (4) can be approximated by a simple integration scheme as the trapezoidal rule or the Gauß–Legendre rule, as was done in [16]. The resulting algorithm from [16] is known by the name FEAST.

The formula for the approximation of $\mathsf{P}_\Lambda\mathsf{Y}$ takes the form

$$\mathsf{P}_\Lambda\mathsf{Y} \approx \widetilde{\mathsf{P}}_\Lambda\mathsf{Y} := \frac{1}{2\pi\mathbf{i}} \sum_{k=1}^{p} \omega_k \varphi'(t_k)(\varphi(t_k)\mathsf{I} - \mathsf{A})^{-1}\mathsf{Y}, \tag{5}$$

where $\varphi : [0, 2\pi] \to \mathbb{C}$ is a parametrization of $\mathcal{C}$. The numbers $t_k \in [0, 2\pi],\ k = 1, \dots, p$, are the integration points, the numbers $\omega_k,\ k = 1, \dots, p$, are the integration weights. Together they define the integration scheme. For each k, a linear system of the form

$$(z_k\mathsf{I} - \mathsf{A})\mathsf{V} = \mathsf{Y}, \text{where } z_k = \varphi(t_k), \tag{6}$$

with m right hand sides has to be solved.

For large system sizes n, the systems (6) cannot be solved with a method based on factorization, i. e., on some form of the Gauß method. This is at least the case unless the matrix A has some special structure, e. g., banded form. (For solution of the systems (6) in the banded case, see [7].)

The typical way for solution of (6) then is to use an iterative solver. A variety of those is available [18]. The system matrix $\mathsf{M}_k := z_k\mathsf{I} - \mathsf{A}$ in (6) is typically not Hermitian since the shifts z_k are complex, and since the z_k may be close to the spectrum of A, the matrices may be ill-conditioned. In [8], a method named CARP-CG was identified that is capable to solve the systems. For details, see [8] and [9]. When assessing the efficiency of the integration based method in combination with an iterative solver, it is sensible to count the overall number of iterations taken by the linear solver.

When computing (5) with an iterative linear solver, the overall error in the approximation to the projector—and thus the achievable quality of the computed eigenpairs—is determined by the discretization error of the integration method, together with the error in the solution of the linear systems [12]. Besides, rounding errors are present. In the following, we will also denote the *computed* quantity in the right hand side of (5) by $\widetilde{\mathsf{P}}_\Lambda$. This means that $\widetilde{\mathsf{P}}_\Lambda$ now also comprises the errors that are introduced by the linear solver and by rounding errors. In [11] it was shown how the errors in the computed eigenvalues and eigenvectors of A are bounded by the error $\|\mathsf{PY} - \widetilde{\mathsf{P}}_\Lambda \mathsf{Y}\|$. If this error is too large, no small residuals $\|\mathsf{A}\widetilde{\mathsf{x}} - \widetilde{\mathsf{x}}\widetilde{\lambda}\|$ can be expected. Hence it is reasonable to decrease $\|\mathsf{PY} - \widetilde{\mathsf{P}}_\Lambda \mathsf{Y}\|$ by increasing the accuracy of $\widetilde{\mathsf{P}}_\Lambda$. This can be achieved by adjusting the accuracy of the linear solver, which terminates when a certain residual in the linear systems $\|\mathsf{M}_k\widetilde{\mathsf{v}} - \mathsf{y}\|$ (either in relative or absolute sense) is reached. Here, $\widetilde{\mathsf{v}}$ denotes the computed counterpart to a column of V and y denotes a column of Y. This procedure is the analog to increasing the polynomial degree in Section 2.1.

Another way of introducing adaptivity is of course to increase the integration order p in (5). However, a fixed value of $p = 8$ proved to be sufficient in most of our experiments. This is why we keep p fixed in the following.

3 Adaptivity: Techniques and results

There are two ways to reduce the work spent in one iteration of the algorithm. The first is to use a smaller search space, with fewer vectors to work on, and the other is to spend less effort on each vector. Vectors may be removed from the search space either if they are already converged ("locking") or based on rank information. These two techniques are discussed in Sections 3.1 and 3.2, respectively. They are independent from the projector being applied via a (polynomial) approximation or with a contour integral. The work spent on each vector is dominated by the degree of the polynomial or the accuracy required for the solution of the linear systems. In Sections 3.3 and 3.4 we will discuss techniques for selecting these parameters suitably. All techniques will be illustrated with numerical examples.

The numerical experiments have been performed with matrices of size $WL \times WL$, resulting from modeling graphene strips with $W \times L$ atoms [7, 4]. Our TestSet comprises twelve test problems Gra{I,II,III}-{1k,11k}-{A,B}, where the first parameter refers to one of three underlying models, the second parameter indicates the matrix size ($n = 1152$ for $W = 12$, $L = 96$ or $n = 11604$ for $W = 12$, $L = 967$), and the third specifies the location of the search interval I_λ (centered at $c = 0.1725$ for A, and at $c = 1.0$ for B). All search intervals have been chosen to contain roughly 300 eigenvalues each, and we always started with the estimate $\widetilde{m} = 300$ and $m = 450$. Most of the experimental data given below were obtained with the two problems GraI-11k-A and GraI-11k-B, that is, with the same "type-I", size-11604 matrix GraI-11k whose eigenvalues are shown in Figure 1. The two problems only differ in the location of the search interval; see Table 1 for more details. Note that in GraI-11k-B the eigenvalues are much denser and come closer to the boundaries of I_λ than in GraI-11k-A, making the former problem much harder to solve. The number of iterations of the eigensolver

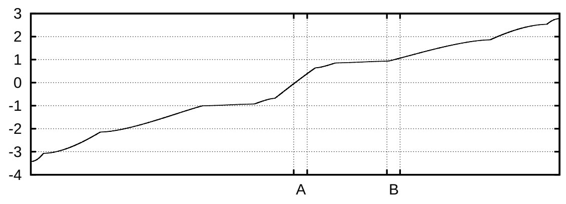

Figure 1: Eigenvalues and search intervals for the matrix GraI-11k.

Table 1: The problems GraI-11k-A and GraI-11k-B.

Problem	GraI-11k-A		GraI-11k-B	
$[\underline{\lambda}, \overline{\lambda}]$	$[-0.0475, 0.3925]$		$[0.935, 1.065]$	
Eigenvalues closest to $\underline{\lambda}$	-0.049569	$-\ 0.046663$	0.934905	0.935025
Eigenvalues closest to $\overline{\lambda}$	0.390239	0.392900	1.064481	1.065235
Eigenvalues in I_λ	298		289	

was limited to 15, and the threshold for considering a Ritz pair $(\widetilde{\mathsf{x}}_j, \widetilde{\lambda}_j)$ converged was set to $\|\mathsf{A}\widetilde{\mathsf{x}}_j - \widetilde{\mathsf{x}}_j\widetilde{\lambda}_j\| \leq 10^{-12} \cdot n \cdot \max\{|\underline{\lambda}|, |\overline{\lambda}|\}$ as suggested in [13]. All problems $(\mathsf{A}, \underline{\lambda}, \overline{\lambda})$ were scaled by $1/4$ before calling the eigensolver to ensure $\mathrm{spec}(\mathsf{A}) \subset [-1, 1]$.

3.1 Locking

If the degree of the Chebyshev polynomial is very high or the linear systems occurring in the contour integration are solved to very high accuracy (e. g., with a direct solver) then all Ritz pairs belonging to the sought eigenvalues may reach the residual threshold in the same (typically the third or fourth) iteration of the algorithm. Otherwise, convergence will occur during several iterations. Then it is natural to "lock" the converged pairs, i. e., to freeze them and remove the vectors from the search space to be used in later iterations.

Table 2 shows the convergence history for the problem GraI-11k-B when using Chebyshev approximation with fixed degree $d = 1600$. Starting at iteration 8, convergence sets in, leading to considerably smaller search spaces thereafter. This also reduces the overall number of matrix–vector multiplications by roughly 27%. (It turns out that reducing the search space also tends to slightly reduce the number of iterations, in this case from 14 to 13.)

Locking, however, essentially amounts to computing the remaining eigenpairs independently from the locked ones, and thus the orthogonality may suffer; cf. also [13]. Figure 2 shows the orthogonality of the vectors computed in the above example if no measures for maintaining orthogonality are taken. Note that the final sorting of the eigenvalues has been suppressed to better expose the sets of vectors that have been locked in the same iteration. While these show excellent orthogonality

Table 2: Dimension of search space and number of converged and locked eigenpairs for problem GraI-11k-B (without SVD-based resizing, cf. Section 3.2).

Iteration	1–5	6	7	8	9	10	11	12	13	14
Without locking:										
Search space dim	450	450	450	450	450	450	450	450	450	450
Converged pairs				183	247	267	281	280	288	289
With locking:										
Search space dim	450	450	450	450	267	202	181	168	165	
Pairs to be locked				183	65	21	13	3	4	

($|\widetilde{\mathsf{x}}_i^\star \cdot \widetilde{\mathsf{x}}_j| < 2 \cdot 10^{-15}$ for $i \neq j$ from the same set), the orthogonality between vectors from different iterations can be worse by five orders of magnitude. Thus it is mandatory to orthogonalize the "current" vectors $\widetilde{\mathsf{x}}_j$ against all previously locked vectors. Doing this yielded $|\widetilde{\mathsf{x}}_i^\star \cdot \widetilde{\mathsf{x}}_j| < 10^{-14}$ for all $i \neq j$.

3.2 SVD-based resizing

The eigenvectors $\widetilde{\mathsf{x}}_j$ computed by the projection-based algorithm should be orthonormal. Because of $\widetilde{\mathsf{X}} = \widetilde{\mathsf{U}} \cdot \widetilde{\mathsf{W}}$ this implies that $\widetilde{\mathsf{U}}$ must have full rank. If this is not the case then $\widetilde{\mathsf{U}}$ should be replaced with a basis of range($\widetilde{\mathsf{U}}$).

A more aggressive resizing strategy also makes use of the facts that (i) the number of eigenvalues to be found in the search interval is very well predicted by the number of singular values of $\widetilde{\mathsf{U}}$ that are larger or equal to $1/2$ and that (ii) the corresponding left singular vectors determine a suitable space for the ensuing projection [7, 12, 25]. Note that the number $1/2$ depends on the implementation of Algorithm 2.1, e. g., on the used integration scheme. It can be shown that it is the correct choice for the Gauß–Legendre rule [25] and the trapezoidal rule [12]. For the polynomial approach, $1/2$ can also be used [12].

In practice, this approach should be modified in two aspects.

The first modification is aimed at avoiding the time-consuming SVD of the large matrix $\widetilde{\mathsf{U}} \in \mathbb{C}^{n\times m}$. As we actually do not need the full accuracy provided by a "true" SVD, we instead consider the smaller matrix $\mathsf{B}_{\widetilde{\mathsf{U}}} := \widetilde{\mathsf{U}}^\star\widetilde{\mathsf{U}} \in \mathbb{C}^{m\times m}$ and its eigendecomposition $\mathsf{B}_{\widetilde{\mathsf{U}}} =: \mathsf{V}\Sigma^2\mathsf{V}^\star$. Thus, we take the eigenvalues $\geq 1/4$ of that matrix and the corresponding eigenvectors (say $\mathsf{V}(:, 1 : s)$, after sorting the eigenvalues descendingly) and replace $\widetilde{\mathsf{U}}$ with $\widetilde{\mathsf{U}} \cdot \mathsf{V}(:, 1 : s)$ [7, 12]. Relying on the eigendecomposition instead of the SVD is particularly attractive if each column of $\widetilde{\mathsf{U}}$ is distributed over several processors because the computation of $\mathsf{B}_{\widetilde{\mathsf{U}}}$ requires only one global synchronization and then the eigendecomposition can be determined redundantly in each processor.

Second, similarly to the initial situation, the search space should remain slightly oversized to enable convergence. Thus, if the SVD (or eigendecomposition) predicts that s eigenvalues still have to be found, we keep $m_{\text{new}} = \max\{s \cdot \varphi_{\text{mult}}, s + \varphi_{\text{add}}\}$

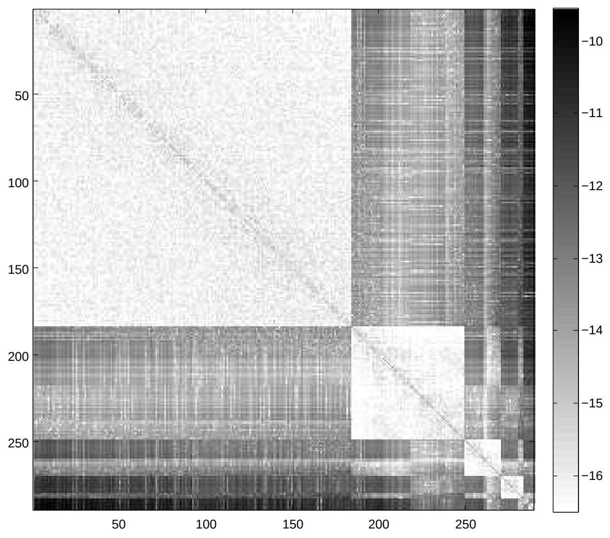

Figure 2: Orthonormality $|\widetilde{x}_i^\star \widetilde{x}_j - \delta_{ij}|$ (on the $\log_{10}$ scale) of the eigenvectors computed for problem GraI-11k-B (with locking, without SVD-based resizing, no orthogonalization against locked vectors).

vectors. In our experiments, we chose $\varphi_{\text{mult}} = 1.5$ and $\varphi_{\text{add}} = 10$.

There are situations where the SVD count is not correct. At least in the very first iteration, it may be zero, no matter how many eigenvalues there are in I_λ. Later on, there may be fluctuations ± 1 in the overall count, in particular if eigenvalues are close to the boundary of I_λ and if convergence happens during several iterations. Thus we apply SVD-based resizing only when we consider the SVD count to be reliable. In our experiments, this meant that (i) we are beyond the first three iterations, (ii) the count did not change during the previous two iterations, and (iii) the remaining Ritz pairs with Ritz values in the search interval already "have begun to converge", i. e. that the smallest of their residuals is $\leq 10^{-6}$. While this compound criterion may be overly pessimistic in most situations, it never allowed undersizing the search space.

Table 3 shows the convergence history for problem GraI-11k-B when SVD-based resizing is enabled. In this example, the number of matrix–vector multiplications is reduced by another 15%, as compared to the second run in Table 2.

3.3 Adjusting the degree of the Chebyshev polynomials

If the projector is applied to a size-m search space via a degree-d Chebyshev polynomial then the computational work in one iteration of the eigensolver is dominated by the $d{\cdot}m$ matrix–vector multiplications (*MVMs*). Figure 3 summarizes the convergence history for solving the problems GraI-11k-A and GraI-11k-B with various fixed degrees d. The pictures reveal that $d = 150$ is sufficient to find all eigenpairs of GraI-11k-A (taking 11 iterations) and that for very high degrees the overall number of MVMs grows linearly

Table 3: Dimension of search space, estimate for the number of remaining eigenpairs, and number of locked eigenpairs for problem GraI-11k-B (with SVD-based resizing).

Iteration	1	2	3–5	6	7	8	9	10	11	12
Search space dim	450	450	450	450	433	429	212	71	35	11
SVD-based eigenvalue count	0	288	289	289	289	285	70	36	17	1
SVD-based resizing to dim				433		427	105	54	27	
Pairs to be locked					4	215	34	19	16	1

with d. For moderate values of d this growth is to some extent balanced by a decreasing number of iterations and by increased success in locking during earlier iterations. In total, $d \sim 250$ gave the best performance.

By contrast, degrees up to 500 did not lead to convergence of a single eigenpair of GraI-11k-B within the maximum number of 15 iterations, and only for $d \geq 1400$ all eigenpairs were found.

This example shows that using the wrong degree may either lead to an unnecessarily high computational effort (number of MVMs) or prevent convergence, and "suitable" degrees can differ widely, even for the same matrix. (Note that locking and SVD-based resizing have a smoothing effect on the MVM count. Without these, the curves show significantly higher variation in the vicinity of the optimum.) An appropriate a priori choice of the degree is difficult without very detailed knowledge about the eigenvalue distribution.

Therefore we propose a strategy that starts with a low degree d_{start} in the first iterations and increases d when the convergence is not "as expected". It is based on the following observations. If the degree is high enough (e. g., $d = 2000$ for GraI-11k-A) then up to four orders of magnitude reduction can be achieved for the residuals in a single iteration. Close to the optimum, however, convergence is somewhat slower (more iterations at lower cost), roughly two orders of magnitude per iteration. This is the convergence rate we are aiming at with our adaptive strategy. Starting with $d_{\text{start}} = 100$, d is increased if the previous iteration did not reduce a particular residual ρ_j by at least a factor of 100. If the reduction was "just too small" (by a factor between 10 and 100) then d is increased only moderately (to $\lfloor \sqrt{2} \cdot d \rfloor$), otherwise d is doubled. Thus we try to avoid over-shooting too much, as well as taking too many iterations to reach a suitable degree range. The index j is chosen to refer to the minimum residual among the not-yet-converged pairs with Ritz value inside the interval. For the example shown in Figure 4, $j = 1$ during the first nine iterations, and $j = 250$ in the last iteration because 249 vectors were locked in iteration 9. Note that in this run all remaining 60 pairs converged in the tenth iteration, but 20 of them were discarded because their Ritz values were outside I_λ. It can be seen that the degree is increased until the desired reduction rate for the residuals is reached.

The overall numbers of MVMs taken to solve the problems GraI-11k-A and GraI-

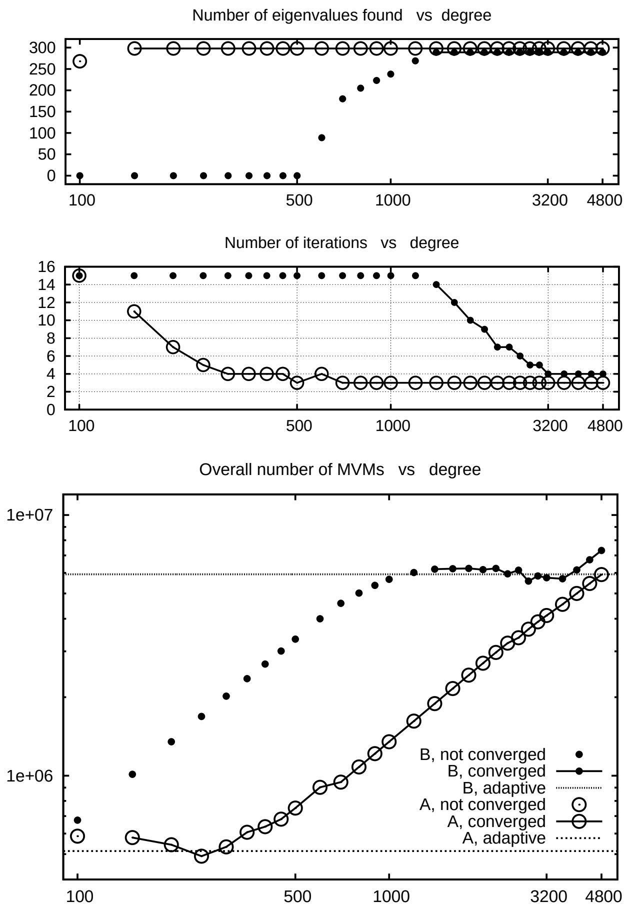

Figure 3: Convergence statistics and overall number of matrix–vector multiplications in the solution of the problems GraI-11k-A (empty circles) and GraI-11k-B (filled circles) using Chebyshev approximation of various fixed degrees. Isolated circles mark runs that did not yield all the eigenpairs in I_λ, connecting lines mark runs with correct results. The number of iterations was limited to 15. The dotted horizontal lines in the bottom picture represent the two runs with the adaptive strategy described in the main text.

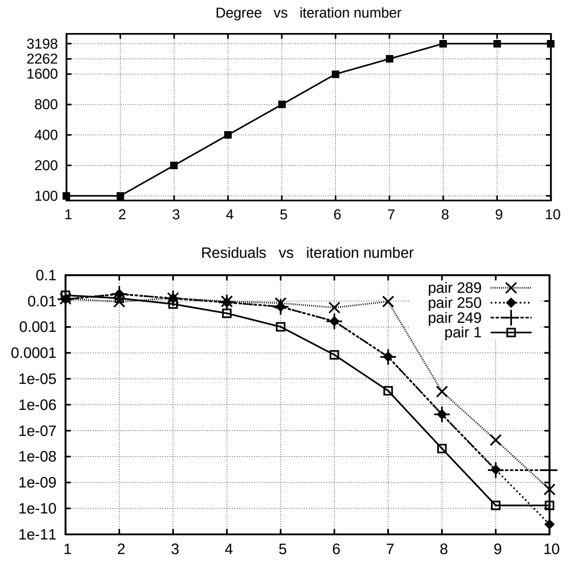

Figure 4: Degrees chosen (top) and residuals of selected pairs (bottom) for GraI-11k-B with the adaptive strategy for the Chebyshev degree. The curves trace the residuals of the first and last pair locked in iteration 9 (nos. 1 and 249) and of the first and last remaining pair (nos. 250 and 289).

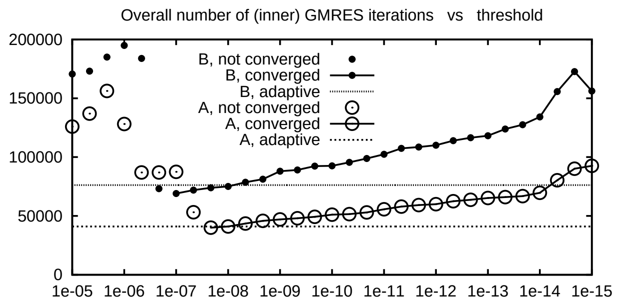

Figure 5: Overall number of GMRES iterations in the solution of the problems GraI-11k-A and GraI-11k-B for various fixed accuracy thresholds. The dotted horizontal lines represent the two runs with the adaptive strategy.

11k-B with this strategy are also marked in Figure 3 (dotted horizontal lines). In both cases the adaptive strategy comes close to the MVM count of the best fixed-degree run.

3.4 Adjusting the accuracy of the iterative linear solver

If the projector is applied via contour integration then the solution of the linear systems $\mathsf{M}_k \mathsf{v}_j = \mathsf{y}_j$ with $\mathsf{M}_k = z_k \mathsf{I} - \mathsf{A}$ dominates the time for each iteration of the eigensolver. The situation is similar as with polynomial approximation, the requirement for terminating the linear solves (typically given as a threshold on the absolute or relative residuals, $\|\mathsf{y}_j - \mathsf{M}_k \mathsf{v}_j\| \leq \tau$ and $\|\mathsf{y}_j - \mathsf{M}_k \mathsf{v}_j\| / \|\mathsf{y}_j\| \leq \tau$, resp.) taking the role of the polynomial degree. If the requirement is set too low (large τ) then the resulting approximation to the projector is not good enough to make the eigensolver converge. On the other hand, with small τ the linear solvers take more iterations—and more time—to reach the solution; see Figure 5, which shows the total number of (inner) iterations in the solution of the linear systems for the problems GraI-11k-A and GraI-11k-B. Here we used restarted GMRES(16) with an incomplete LU preconditioner [18] (droptol = 0.001) to solve the linear systems because for these problems this combination was feasible and performed better than our default solver CARP-CG; cf. Section 2.2. If the accuracy requirement is set too ambitiously then the linear solver may run up to its maximum allowed number of iterations, leading to a sharp increase in the overall time.

Our adaptive strategy for adjusting the accuracy threshold is similar to the one for the Chebyshev degree: Starting with $\tau_{\text{start}} = 10^{-8}$, τ is decreased by a factor of 4 if the previous iteration did reduce the residual ρ_j by a factor of less than 10, and by a factor of 2 if the residual was reduced by a factor between 10 and 100, with j chosen as in Section 3.3. Again, the adaptive strategy came close to the best fixed-accuracy runs for GraI-11k-A and GraI-11k-B; see the horizontal lines in Figure 5.

4 Conclusions

Combining the techniques described so far we arrive at the procedure given in Algorithm 4.1. The algorithm comprises the polynomial based as well as the integration based algorithm.

Both instantiations (Chebyshev polynomials with adaptive degree, as well as GMRES with variable tolerance) were able to solve all twelve problems from the TestSet. While these problems are not very large, they cover a broad range of “hardness”. Together with a few (successful) runs with larger problems ($W = 124$, $L = 967$, $n = 119908$, and $W = 124$, $L = 9677$, $n = 1199948$) this indicates that the adaptive strategies are successful in automatically finding suitable degrees or residual thresholds, yielding a robust overall method. According to the results from Section 3, they may also come close to the performance that could be obtained with the best fixed parameters, provided these were known beforehand.

However, the robustness of the integration based algorithm depends strongly on the ability of the iterative solver to achieve small residuals, which in turn requires a powerful preconditioner. In fact, relaxing droptol to 0.01 leads the adaptive GMRES-based method to fail on one of the problems from the TestSet (GraII-11k-B), due to integration points z_k being very close to eigenvalues of the matrix. Therefore we will continue our work on robust iterative schemes and preconditioners for the nearly singular systems arising in the integration based algorithm.

In the future we will also work on more sophisticated adaptive schemes, including the integration order p as a parameter. We also will consider the use of mixed precision arithmetic. For instance, in earlier iterations of the methods, some or all computations may be performed in single precision to save further computation time.

Acknowledgements. We thank Andreas Alvermann, Holger Fehske and Andreas Pieper from Ernst-Moritz-Arndt-Universität Greifswald for providing us with the graphene matrices and Jonas Thies and Achim Basermann from DLR Köln for providing us with the CARP-CG code, which is the default iterative linear solver in our ProjAlg algorithm.

References

[1] Andreas Alvermann, Achim Basermann, Holger Fehske, Martin Galgon, Georg Hager, Moritz Kreutzer, Lukas Krämer, Bruno Lang, Andreas Pieper, Melven Röhrig-Zöllner, Faisal Shahzad, Jonas Thies, and Gerhard Wellein. ESSEX: Equipping Sparse Solvers for Exascale. Preprint BUW-IMACM 14/31, http://www.imacm.uni-wuppertal.de, 2014.

[2] W. E. Arnoldi. The principle of minimized iteration in the solution of the matrix eigenvalue problem. *Qart. Appl. Math.*, 9:17–29, 1951.

[3] Martin Braun. *Differential Equations and Their Applications*, volume 11 of *Texts in Applied Mathematics*. Springer, New York, 1993.

Algorithm 4.1 Projection based algorithm ProjAlg

```
Input: An interval $I_\lambda = [\underline{\lambda}, \overline{\lambda}]$ and an estimate $\widetilde{m}$ of the number of eigenvalues in $I_\lambda$
Output: $\hat{m} \le m$ eigenpairs with eigenvalues in $I_\lambda$ ($m$ is chosen somewhat larger than $\widetilde{m}$)
$m := \max\{\widetilde{m} \cdot \varphi_{\text{mult}}, \widetilde{m} + \varphi_{\text{add}}\}$
choose $\mathsf{Y} \in \mathbb{C}^{n \times m}$ with orthonormal columns
while not done do
    if degree/threshold adaptivity is enabled
        adjust degree of approximation polynomial or threshold of linear solver
    compute approximation $\widetilde{\mathsf{U}}$ to $\mathsf{U} := \mathsf{P}_\Lambda \mathsf{Y}$, using either polynomial approximation
            or contour integration
    { SVD-based resizing: }
    determine the singular values $\sigma$ of the $n$-by-$m$ matrix $\widetilde{\mathsf{U}}$ via the
            $m$-by-$m$ eigendecomposition $\widetilde{\mathsf{U}}^\star \widetilde{\mathsf{U}} =: \mathsf{V}\Sigma^2\mathsf{V}^\star$
    let $r$ and $s$ be the number of singular values $\ge \text{thresh}_{\text{rank}}$ and $\ge 1/2$, resp.
    $m_{\text{new}} := r$
    if SVD-based resizing is enabled and the SVD count is considered reliable
        $m_{\text{new}} := \min\{m_{\text{new}}, \max\{s \cdot \varphi_{\text{mult}}, s + \varphi_{\text{add}}\}\}$
    if $m_{\text{new}} < m$
        replace $\widetilde{\mathsf{U}}$ with $\widetilde{\mathsf{U}} \cdot \mathsf{V}(:, 1 : m_{\text{new}})$ and recompute (smaller) $\mathsf{B}_{\widetilde{\mathsf{U}}} := \widetilde{\mathsf{U}}^\star \widetilde{\mathsf{U}}$
        $m := m_{\text{new}}$
    { project and solve small problem: }
    compute $\mathsf{A}_{\widetilde{\mathsf{U}}} := \widetilde{\mathsf{U}}^\star \mathsf{A} \widetilde{\mathsf{U}}$
    solve the size-$m$ generalized eigenproblem $\mathsf{A}_{\widetilde{\mathsf{U}}} \widetilde{\mathsf{W}} = \mathsf{B}_{\widetilde{\mathsf{U}}} \widetilde{\mathsf{W}} \widetilde{\Lambda}$
    compute the approximate Ritz pairs $(\widetilde{\mathsf{X}} := \widetilde{\mathsf{U}} \cdot \widetilde{\mathsf{W}}, \widetilde{\Lambda})$
    orthogonalize $\widetilde{\mathsf{X}}$ against $\mathsf{X}_{\text{locked}}$
    for $j = 1 : m$
        compute residual $\rho_j = \|\mathsf{A}\widetilde{\mathsf{x}}_j - \widetilde{\mathsf{x}}_j \widetilde{\lambda}_j\|$
        if $\widetilde{\lambda}_j \in I_\lambda$ and $\rho_j \le$ residual threshold and locking is enabled
            remove $\widetilde{\mathsf{x}}_j$ from $\widetilde{\mathsf{X}}$ and append $(\widetilde{\mathsf{x}}_j, \widetilde{\lambda}_j)$ to $(\mathsf{X}_{\text{locked}}, \Lambda_{\text{locked}})$
    check for convergence and set $\mathsf{Y} := \widetilde{\mathsf{X}}$
end while
sort computed eigenvalues and rearrange converged eigenvectors accordingly
```

[4] 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.*, 81:109–162, 2009.

[5] P. J. Davis and P. Rabinowitz. *Methods of numerical integration.* Academic Press, Orlando, FL, second edition, 1984.

[6] V. L. Druskin and L. A. Knizhnerman. Two polynomial methods of calculating functions of symmetric matrices. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 29(6):112–121, 1989.

[7] Martin Galgon, Lukas Krämer, Bruno Lang, Andreas Alvermann, Holger Fehske, and Andreas Pieper. Improving robustness of the FEAST algorithm and solving eigenvalue problems from graphene nanoribbons. *PAMM*, 14(1):821–822, 2014.

[8] Martin Galgon, Lukas Krämer, Jonas Thies, Achim Basermann, and Bruno Lang. On the parallel iterative solution of linear systems arising in the FEAST algorithm for computing inner eigenvalues. Preprint BUW-IMACM 14/35, http://www.imacm.uni-wuppertal.de/, 2014.

[9] Dan Gordon and Rachel Gordon. CARP-CG: A robust and efficient parallel solver for linear systems, applied to strongly convection dominated PDEs. *Parallel Comput.*, 36(9):495–515, 2010.

[10] Stefan Güttel, Eric Polizzi, Peter Tang, and Gautier Viaud. Zolotarev quadrature rules and load balancing for the FEAST eigensolver. The University of Manchester, MIMS EPrint 2014.39, http://www.manchester.ac.uk/mims/eprints, 2014.

[11] Lukas Krämer. Convergence of integration based methods for the solution of standard and generalized Hermitian eigenvalue problems. Preprint BUW-IMACM 14/30, http://www.imacm.uni-wuppertal.de, 2014.

[12] Lukas Krämer. *Integration based solvers for standard and generalized Hermitian eigenvalue problems.* PhD thesis, Bergische Universität Wuppertal, 2014. http://nbn-resolving.de/urn/resolver.pl?urn=urn:nbn:de:hbz:468-20140701-112141-6.

[13] Lukas Krämer, Edoardo Di Napoli, Martin Galgon, Bruno Lang, and Paolo Bientinesi. Dissecting the FEAST algorithm for generalized eigenproblems. *J. Comput. Appl. Math.*, 244:1–9, 2013.

[14] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Nat. Bur. Stand.*, 45(4):255–282, 1950.

[15] G. G. Lorentz. *Approximation of Functions.* Chelsea Publishing Company, New York, NY, 1986.

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

[17] Aubrey B. Poore. A model equation arising from chemical reactor theory. *Arch. Rat. Mech. Anal.*, 52(4):358–388, 1973.

[18] Y. Saad. *Iterative Methods for Sparse Linear Systems.* SIAM, Philadelphia, PA, 2nd edition, 2003.

[19] T. Sakurai, Y. Futamura, and H. Tadano. Efficient parameter estimation and implementations of a contour integral-based eigensolver. *J. Algo. Comput. Tech.*, 7:249–269, 2013.

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

[21] Grady Schofield, James R. Chelikowsky, and Yousef Saad. A spectrum slicing method for the Kohn–Sham problem. *Comput. Phys. Comm.*, 183(3):497–505, 2012.

[22] G. L. G. Sleijpen and H. A. van der Vorst. A Jacobi–Davidson iteration method for linear eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, 17(2):401–425, 1996.

[23] D. C. Sorensen. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM J. Matrix Anal. Appl.*, 13:357–385, 1992.

[24] G. W. Stewart. *Matrix Algorithms*, volume II, Eigensystems. SIAM, Philadelphia, PA, 2001.

[25] Ping Tak Peter Tang and Eric Polizzi. FEAST as a subspace iteration eigensolver accelerated by approximate spectral projection. *SIAM J. Matrix Anal. Appl.*, 35(2):354–390, 2014.

[26] Alexander Weiße, Gerhard Wellein, Andreas Alvermann, and Holger Fehske. The kernel polynomial method. *Rev. Mod. Phys.*, 78:275–306, 2006.

[27] Y. Zhou and Y. Saad. A Chebyshev–Davidson algorithm for large symmetric eigenproblems. *SIAM J. Matrix Anal. Appl.*, 29(3):954–971, 2007.