Improved Coefficients for Polynomial Filtering in ESSEX

Martin Galgon, Lukas Krämer, Bruno Lang, Andreas Alvermann, Holger Fehske, Andreas Pieper, Georg Hager, Moritz Kreutzer, Faisal Shahzad, Gerhard Wellein, Achim Basermann, Melven Röhrig-Zöllner, and Jonas Thies

Abstract The ESSEX project is an ongoing effort to provide exascale-enabled sparse eigensolvers, especially for quantum physics and related application areas. In this paper we first briefly summarize some key achievements that have been made within this project.

Then we focus on a projection-based eigensolver with polynomial approximation of the projector. This eigensolver can be used for computing hundreds of interior eigenvalues of large sparse matrices. We describe techniques that allow using lower-degree polynomials than possible with standard Chebyshev expansion of the window function and kernel smoothing. With these polynomials, the degree, and thus the number of matrix–vector multiplications, typically can be reduced by roughly one half, resulting in comparable savings in runtime.

1 The ESSEX Project

ESSEX—Equipping Sparse Solvers for Exascale—is one of the projects within the German Research Foundation (DFG) Priority Programme "Software for Exascale Computing" (SPPEXA). It is a joint effort of physicists, computer scientists and mathematicians to develop numerical methods and programming concepts for the solution of large sparse eigenvalue problems on extreme-scale parallel machines.

Martin Galgon, Lukas Krämer and Bruno Lang
University of Wuppertal, School of Mathematics and Natural Sciences, Wuppertal, Germany

Andreas Alvermann, Holger Fehske and Andreas Pieper
University of Greifswald, Institute of Physics, Greifswald, Germany

Georg Hager, Moritz Kreutzer, Faisal Shahzad and Gerhard Wellein
Erlangen Regional Computing Center, Erlangen, Germany

Achim Basermann, Melven Röhrig-Zöllner, and Jonas Thies
German Aerospace Center (DLR), Simulation and Software Technology, Cologne, Germany

ESSEX' goal is not to provide a general-purpose eigenvalue library directly usable in the full range of eigenvalue computations. Rather, in the first funding period the focus was on appropriate methods for selected physical applications featuring rather different characteristics, such as the need for a few extremal eigenpairs, a bunch of interior eigenpairs, information about the whole eigenvalue distribution, and dynamics, of symmetric or Hermitian matrices. These methods include Jacobi–Davidson iteration, projection-based algorithms, the kernel polynomial method, and Chebyshev time propagation, respectively.

To obtain scalable high performance, ESSEX takes a holistic performance engineering approach encompassing the applications, algorithmic development, and underlying building blocks. One of the outcomes is an "Exascale Sparse Solver Repository" (ESSR) that provides high-performance implementations of several methods. In addition, experiences gained through ESSEX work can give guidelines for addressing structurally similar problems in numerical linear algebra. In the following we briefly summarize some of ESSEX' results so far.

Up to now, block variants of the Jacobi–Davidson algorithm have been considered worthwhile mainly for robustness reasons. Our investigations [20] have revealed that they also can be faster than non-blocked variants, provided that all basic operations (in particular sparse matrix times multiple vector multiplication and operations on tall and skinny matrices) achieve optimal performance. This is typically not the case if block vectors are stored in column-major ordering, and we showed that some care has to be taken when implementing algorithms using row-major block vectors instead. A blocked GMRES solver has been developed for solving the multiple Jacobi–Davidson correction equations occurring in the block algorithm.

An adaptive framework for projection-based eigensolvers has been developed [8], which allows the projection to be carried out with either polynomials or a contour integration as in the FEAST method [19]. The latter approach requires the solution of highly indefinite, ill-conditioned linear systems. A robust solver for these has been identified and implemented [9].

The eigensolvers in ESSEX are complemented by domain-specific algorithms for quantum physics computations such as the kernel polynomial method (KPM) [26] for the computation of eigenvalue densities and spectral functions. These algorithms, which are based on simple schemes for the iterative evaluation of (Chebyshev) polynomials of sparse matrices, are very attractive candidates for our holistic performance engineering approach. For example, in a large-scale KPM computation [14, 15] we could achieve 11% of LINPACK efficiency on 4096 heterogeneous CPU–GPU nodes of Piz Daint at Swiss National Supercomputing Centre (CSCS). This is an unusually high value for sparse matrix computations whose performance is normally much more restricted by the main memory bandwidth.

Such progress is possible because in ESSEX the algorithmic work on the above methods goes hand-in-hand with the model-guided development of high-performance MPI+X hybrid parallel kernels for the computational hot spots. They are included in the "General, Hybrid, and Optimized Sparse Toolkit" (GHOST) [16]. Besides implementations of sparse matrix–(multiple) vector multiplication, optionally chained with vector updates and inner products to reduce data transfers, operations with

"tall skinny" dense matrices, and other basic building blocks, GHOST also provides mechanisms for thread-level task management to utilize the aggregate computational power of heterogeneous nodes comprising standard multicore CPUs, Intel Xeon Phi manycore CPUs, and NVIDIA GPUs, and to enable asynchronous checkpointing with very low overhead.

A "Pipelined Hybrid-parallel Iterative Solver Toolkit" (PHIST) [25] provides an interface to the GHOST kernels and adapters to the Trilinos libraries Anasazi [1] and Belos [2]. It also contains higher-level kernels and a comprehensive test framework.

A unified sparse matrix storage format for heterogeneous compute environments, SELL-C-σ, has been proposed [13]. It is a generalization of the existing formats Compressed Row Storage and (Sliced) ELLPACK with row sorting and allows near-to-optimum performance for a wide range of matrices on CPU, Phi, and GPU, thus often obviating the former need to use different formats on different platforms.

While the methods and software developed in ESSEX are applicable to general eigenvalue problems our project specifically addresses quantum physics research applications. Among these, the recent fabrication of graphene [4] and topological insulator [11] materials has renewed the interest in the solution of large scale interior eigenvalue problems. Realistic modeling of structured or composite devices made out of these materials directly leads to large sparse matrix representations of the Hamilton operator in the Schrödinger equation, and the eigenvalues and eigenvectors deep inside the spectrum, near the Fermi energy, determine the electronic structure and topological character and thus the functioning of such devices.

A more comprehensive description of the results obtained within the ESSEX project can be found in [25] and the references therein, and on the ESSEX homepage https://blogs.fau.de/essex/.

In the following sections we focus on the BEAST-P eigensolver that is available in the ESSR and in particular propose approaches for increasing its efficiency.

2 Accelerated Subspace Iteration With Rayleigh–Ritz Extraction

Eigenvalue problems (EVPs) $\mathsf{A}\mathsf{v} = \lambda\mathsf{v}$ with real symmetric or complex Hermitian matrices A arise in many applications, e.g., electronic structure computations in physics and chemistry [3].

Often the matrix is very large and sparse, and only a few extreme or interior eigenpairs (λ, v) of the spectrum are required. In this case iterative solvers based on Krylov subspaces, such as Lanczos or Jacobi–Davidson-type methods [17, 24], or block variants of these [21, 20], tend to be most efficient. A common feature of such algorithms is a subspace that is expanded by a single vector or a block of vectors in each iteration, thus increasing its dimension.

In this work we focus on the situation when (i) the eigenpairs in a given interval are sought, $\lambda \in I_\lambda = [\alpha, \beta]$, (ii) these eigenvalues are in the interior of the spectrum, and (iii) their number is moderately large (some hundreds, say). Then subspace iteration, possibly coupled with a Rayleigh–Ritz extraction, may be competitive or

superior to the afore-mentioned methods. The basic procedure is summarized in Fig. 1; cf. also [22, 8].

Fig. 1 Key algorithmic steps for subspace iteration with Rayleigh–Ritz extraction.

Given: $\mathsf{A} \in \mathbb{C}^{n\times n}$, $I_\lambda = [\alpha,\beta] \subset \mathbb{R}$
Sought: Those eigenpairs (λ,v) of A such that $\lambda \in I_\lambda$

start with a subspace $\mathsf{Y} \in \mathbb{C}^{n\times m}$ of suitable dimension m
while not yet converged
  compute $\mathsf{U} = f(\mathsf{A})\cdot\mathsf{Y}$ for a suitable function f
  compute $\mathsf{A}_\mathsf{U} = \mathsf{U}^*\mathsf{A}\mathsf{U}$ and $\mathsf{B}_\mathsf{U} = \mathsf{U}^*\mathsf{U}$ and solve the size-m generalized EVP $\mathsf{A}_\mathsf{U}\mathsf{W} = \lambda\mathsf{B}_\mathsf{U}\mathsf{W}$
  replace Y with $\mathsf{U}\cdot\mathsf{W}$

The function f can be chosen in many different ways, ranging from $f \equiv \mathrm{id}$ (i.e., $\mathsf{U} = \mathsf{A}\cdot\mathsf{Y}$, "power iteration") to more sophisticated "filter functions;" cf., e.g., [22]. In particular, consider the "window function"

$$f(x) \equiv \chi_{I_\lambda}(x) = \begin{cases} 1, & x \in I_\lambda, \\ 0, & \text{otherwise} \end{cases}$$

and a column y_j of Y, expanded w.r.t. an orthonormal system $\mathsf{v}_1,\ldots,\mathsf{v}_n$ of A's eigenvectors, $\mathsf{y}_j = \sum_{k=1}^n \eta_k \mathsf{v}_k$. Then

$$f(\mathsf{A})\cdot\mathsf{y}_j = \sum_{k=1}^n \eta_k f(\lambda_k)\mathsf{v}_k = \sum_{\lambda_k\in I_\lambda} \eta_k \mathsf{v}_k \ ,$$

i.e., y_j is projected onto the invariant subspace spanned by the desired eigenvectors. With this choice of f, the procedure in Fig. 1 would terminate after just one iteration. Especially for large matrices, however, $\chi_{I_\lambda}(\mathsf{A})\cdot\mathsf{y}_k$ can only be approximated, either by using specialized algorithms for matrix functions $f(\mathsf{A})\cdot\mathsf{b}$ [12, 23] or by approximating the function: $f \approx \chi_{I_\lambda}$. We will focus on the latter approach, in particular on using polynomials for approximating f. Very good approximation can be achieved with rational functions, and methods for choosing these optimally have been investigated [10] (cf. also [19] for the FEAST algorithm, where the rational approximation is done via a numerical contour integration). However, rational functions require the solution of shifted linear systems $(\mathsf{A}-\sigma\mathsf{I})\mathsf{x}_j = \mathsf{y}_j$, and these can be very challenging if direct solvers are not feasible.

In this work we will instead consider polynomial functions $f(x) = p(x)$, where $p(x)$ is a polynomial of degree d. Then $f(\mathsf{A})\cdot\mathsf{y}$ is easy to evaluate even if A is not available explicitly but its action $\mathsf{A}\cdot\mathsf{v}$ on any vector v can be determined (e.g., for very large sparse matrices whose nonzero entries are not stored but re-computed whenever they are needed).

An important observation is that, in order for the overall algorithm in Fig. 1 to work, we must be able to determine the number of eigenvalues that are contained in

I_λ, e.g., to decide whether all sought eigenpairs have been found and to adjust the dimension of the search space Y. This can be done by counting the singular values of the current matrix U that are larger than some given bound τ_{inside}. Thus we must have

$$p(x) \geq \tau_{\text{inside}} \quad \text{for } x \in [\alpha, \beta] \tag{1}$$

in order not to miss one of the sought eigenpairs. For compatibility with the FEAST realization of the projector we choose $\tau_{\text{inside}} = 0.5$ throughout. By contrast, *linearity* of the filter is not an issue: it is not necessary that $p(x) \approx 1$ in the interior of I_λ. (The values should, however, not be too large to avoid numerical problems in the SVD computation.)

For the approximate projector $p(\mathsf{A})$ to be effective, it must dampen the components v_j corresponding to unwanted eigenvalues $\lambda_j \notin I_\lambda$. Thus we require

$$|p(x)| \leq \tau_{\text{outside}} \quad \text{for } x \notin [\alpha - \delta, \beta + \delta] \tag{2}$$

for some threshold τ_{outside}, e.g., $\tau_{\text{outside}} = 0.01$. The margin $\delta > 0$ is necessary because a continuous function p cannot achieve (2) for all $x \notin I_\lambda$ while fulfilling (1). Note that if there are unwanted eigenvalues $\lambda_j \in (\alpha - \delta, \alpha) \cup (\beta, \beta + \delta)$ then the corresponding components v_j may not be sufficiently damped to yield satisfactory convergence of the overall method. In this case it may be necessary to increase the degree of the polynomial; cf. [8]. Anyway p should be chosen such that a small margin can be achieved. This is the main focus of this work.

The remainder of the article is organized as follows. In Sect. 3 we will discuss how to reduce the margin while still trying to approximate the window function, $p(x) \approx \chi_{I_\lambda}$. In Sect. 4 we will see that the margin for $[\alpha, \beta]$ can be reduced further by approximating the window function for a *smaller* interval. Giving up the linearity constraint within the interval yields another type of filter that is discussed in Sect. 5. Numerical results presented in Sect. 6 show that the reduction of the margin also leads to a lower number of operations (measured by the number of matrix–vector multiplications) and faster execution for the overall eigensolver.

3 Polynomial Approximation of the Window Function

The Chebyshev approximation discussed in the following requires $x \in [-1, 1]$. To achieve this, the matrix A is shifted and scaled such that all its eigenvalues lie between -1 and 1. The search interval $[\alpha, \beta]$ is transformed accordingly. Throughout the following discussion we assume that this preprocessing has already been done.

It is well known [7] that the window function $\chi_{[\alpha,\beta]}$ can be approximated by a polynomial expansion

$$\chi_{[\alpha,\beta]}(x) \approx \sum_{k=0}^{d} c_k T_k(x) \ , \tag{3}$$

where the T_k are the Chebyshev polynomials of the first kind,

$$
\begin{aligned}
T_0(x) &\equiv 1 \,, \\
T_1(x) &= x \,, \\
T_k(x) &= 2x \cdot T_{k-1}(x) - T_{k-2}(x) \,, \quad k \geq 2,
\end{aligned}
$$

and the coefficients are given by

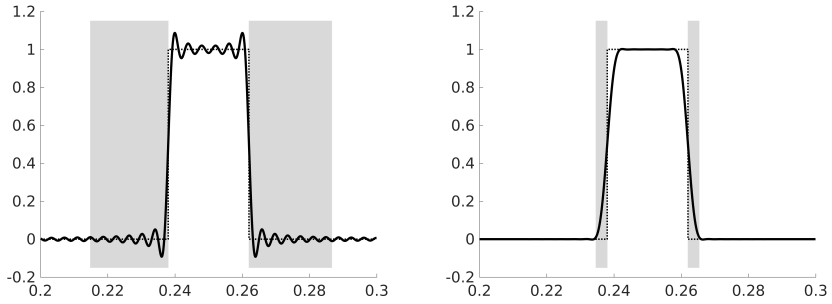
$$
\left.
\begin{aligned}
c_0 &= \tfrac{1}{\pi} \cdot (\arccos\alpha - \arccos\beta) \,, \\
c_k &= \tfrac{2}{k\pi} \cdot (\sin(k \cdot \arccos\alpha) - \sin(k \cdot \arccos\beta)) \,, \quad k \geq 1.
\end{aligned}
\right\} \tag{4}
$$

Using a finite expansion, however, leads to so-called Gibbs oscillations [26] close to the jumps of the function, clearly visible in the left picture in Fig. 2. These oscillations can be reduced by using appropriate kernels [26], e.g., of Jackson, Fejér, Lorentz, or Lanczos type. Incorporating a kernel amounts to replacing the c_k in (3) with modified coefficients $c'_k = g_k \cdot c_k$. For the Lanczos kernel, which has proved successful in the context of polynomial approximation [18], the corrections are given by

$$
g_k = \left(\operatorname{sinc}\frac{k}{d+1}\right)^{\mu} \,, \quad k \geq 0 \,, \quad \text{where} \quad \operatorname{sinc}\xi = \frac{\sin(\pi\xi)}{\pi\xi} \,.
$$

The parameter μ is assumed to be positive and integer.

Fig. 2 Dotted curves: window function $\chi_{[\alpha,\beta]}(x)$ for the interval $[\alpha,\beta] = [0.238, 0.262]$; solid curves: degree-1600 Chebyshev approximation $p(x)$ without Lanczos kernel (left picture) and with Lanczos kernel ($\mu = 2$; right picture). The "damping condition" $|p(x)| \leq \tau_{\text{outside}} = 0.01$ may be violated in the light gray areas.



The right picture in Fig. 2 demonstrates the smoothing effect of the Lanczos kernel ($\mu = 2$). As the oscillations are removed almost completely, the margin δ (width of the gray areas in Fig. 2) is reduced from approximately 0.02316 to 0.00334, even if the resulting p has a much lower steepness in the points α and β. Roughly speaking, the right picture in Fig. 2 suggests bad damping throughout the whole (smaller)

margin, whereas in the left picture good damping may be achieved even at some points within the (larger) margin.

To obtain a small margin δ, the filter polynomial should be as steep as possible at the points α and β and have oscillations $|p(x)| \geq \tau_{\text{outside}}$ only very close to the interval. In the following we will consider three different approaches for increasing the steepness while keeping the oscillations limited. The first approach—discussed below—still aims at approximating the window function $\chi_{[\alpha,\beta]}$. The other approaches are based on different target functions; they will be presented in Sections 4 and 5.

For the further use as a filter in the eigensolver the integrality restriction for μ is not necessary. Indeed, non-integer μ values may lead to filters with smaller margin; see Fig. 3 for $\mu = \sqrt{2}$, with $\delta \approx 0.00276$. Thus the margin was reduced by another factor of 1.21 with respect to $\mu = 2$. In the remainder of the paper, this factor will be called the gain of a filter:

$$\mathsf{gain} = \frac{\delta(\text{Chebyshev approximation with Lanczos kernel, } \mu = 2)}{\delta(\text{filter under consideration})} \,. \tag{5}$$

(A close look reveals that the amplitude of the oscillations outside the margin has increased w.r.t. $\mu = 2$, but not enough to violate the condition $|p(x)| \leq \tau_{\text{outside}}$.)

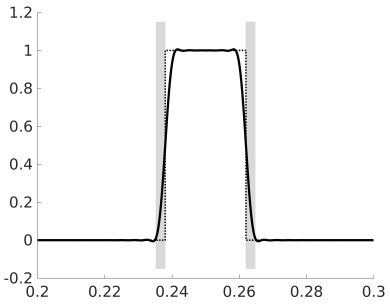


Fig. 3 Window function $\chi_{[\alpha,\beta]}(x)$ for the interval $[\alpha,\beta] = [0.238, 0.262]$ (dotted line) and degree-1600 Chebyshev approximation $p(x)$ with $\mu = \sqrt{2}$ Lanczos kernel (solid line).

According to Fig. 4, a gain of roughly 1.4 may be achieved with an approximation to the window function if the kernel parameter μ is chosen appropriately, and "appropriately" depends on the degree d. (The optimal value for μ also depends on α and β; this dependence is not shown in the picture.) Note that μ can be optimized *before* applying the filter $p(\mathsf{A})$ to some vectors v because δ depends only on the interval $[\alpha,\beta]$ and the degree d, but not on A and v.

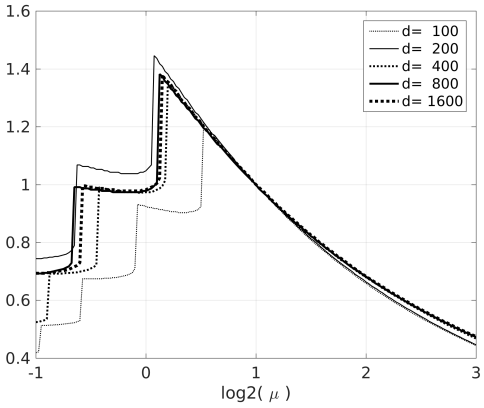


Fig. 4 gain, as defined in (5), by using different values for the parameter μ in the Lanczos kernel for $[\alpha,\beta] = [0.238, 0.262]$.

4 Shrinking the Interval

The second approach to improving the filter is based on shrinking the interval to a smaller one, $[\alpha,\beta] \mapsto [\widetilde{\alpha},\widetilde{\beta}] = [\alpha+\Delta_1, \beta-\Delta_2] \subseteq [\alpha,\beta]$. If the Lanczos parameter μ is kept fixed, the shoulders of p will follow the interval boundaries and move inwards. Thus, the resulting function values $\widehat{p}(\alpha)$ and $\widehat{p}(\beta)$ will drop below 0.5. We then try to restore the property (1) by scaling the polynomial (via its coefficients $\widehat{c}_k$):

$$\widetilde{p} = \varphi \cdot \widehat{p}\,, \quad \text{where} \quad \varphi = \frac{0.5}{\min\{\widehat{p}(\alpha), \widehat{p}(\beta)\}}\,. \tag{6}$$

Figure 5 shows the resulting polynomial for $\widetilde{\alpha} = 0.24032$ and $\widetilde{\beta} = 0.25969$.

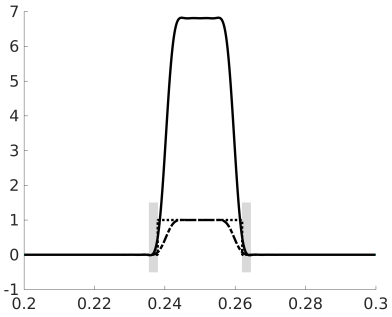


Fig. 5 Window function $\chi_{[\alpha,\beta]}(x)$ for $[\alpha,\beta] = [0.238, 0.262]$ (dotted line) and degree-1600 Chebyshev approximation $\widetilde{p}(x)$ with $\mu = 2$ Lanczos kernel for the shrunken interval $[\widetilde{\alpha},\widetilde{\beta}] = [0.24032, 0.25969]$ before (dash-dotted) and after (solid) scaling.

It remains to determine by which amount the interval should be shrunk. The shift Δ_1 is determined by the slope $p'(\alpha)$: we choose it proportional to $p(\alpha)/p'(\alpha)$, with a proportionality factor $\sigma \geq 0$, and analogously for β. Note that for low degrees d

and narrow intervals $[\alpha,\beta]$ the shoulder of p may not be very steep, and thus the resulting interval can become empty: $\widetilde{\alpha} = \alpha + \sigma\frac{p(\alpha)}{p'(\alpha)} \geq \beta + \sigma\frac{p(\beta)}{p'(\beta)} = \widetilde{\beta}$. To avoid this situation and to preserve a certain width of the interval, the shifts are limited to $\Delta_i \leq 0.9 \cdot r$, where $r = (\beta-\alpha)/2$ is the radius of the original interval. In particular, none of the endpoints is moved over the original midpoint.

Figure 6 reveals that a gain of almost 3 can be obtained with suitable combinations (μ,σ) (note the logarithmic color coding). As $\sigma = 0$ disables shrinking, the left border of the plot corresponds to the "$d = 1600$" curve in Fig. 4.

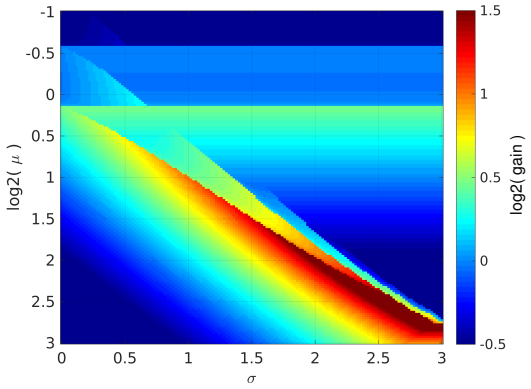


Fig. 6 $\log_2(\mathsf{gain})$, as defined in (5), by using different values for the parameter μ in the Lanczos kernel and different shrink factors σ for $[\alpha,\beta] = [0.238, 0.262]$ and $d = 1600$.

The search for a suitable (μ,σ) combination is simplified by the observation that there are just three "essentially different" patterns for the dependence $\mathsf{gain}(\mu,\delta)$: while Fig. 6 gives a typical pattern for "high" degrees, the patterns for "critical" and "low" degrees are shown in Fig. 7. Whether a given degree d is to be considered high, critical, or low, depends mainly on the width of the interval, $\beta-\alpha$, and to a lesser degree on the location of the interval's midpoint with respect to $[-1,1]$: intervals close to the origin require higher values of d than intervals near the boundaries ± 1; see Fig. 8 for very similar patterns corresponding to different interval widths and locations.

No matter if the degree is low, critical or high, the (μ,σ) combination yielding the optimal gain is located close to the diagonal $\log_2(\mu) = \sigma$. Therefore our search considers only those combinations on a $(\Delta\log_2\mu, \Delta\sigma)$-equispaced grid that lie within a specified band along the diagonal (see Fig. 9) and selects the one giving the highest gain.

This BAND search may be followed by a closer look at the vicinity of the selected combination $P_{\text{best}} = (\log_2\mu_{\text{best}}, \sigma_{\text{best}})$, either by considering the points on an equispaced GRID centered at P_{best} (with smaller step sizes $\Delta'\log_2\mu \ll \Delta\log_2\mu$, $\Delta'\sigma \ll \Delta\sigma$), or by following a PATH originating at P_{best}: consider the eight neighbors of P_{best} at distances $\pm\frac{1}{2}\Delta\log_2\mu$, $\pm\frac{1}{2}\Delta\sigma$, go to the one giving the best gain, and repeat until none of the neighbors is better (then halve the step sizes and repeat until a prescribed minimum step size is reached).

Fig. 7 Typical pattern of $\log_2(\mathsf{gain})$, as defined in (5), for $[\alpha,\beta] = [0.238, 0.262]$. Left picture: "critical" degree ($d = 565$); right picture: "low" degree ($d = 141$).

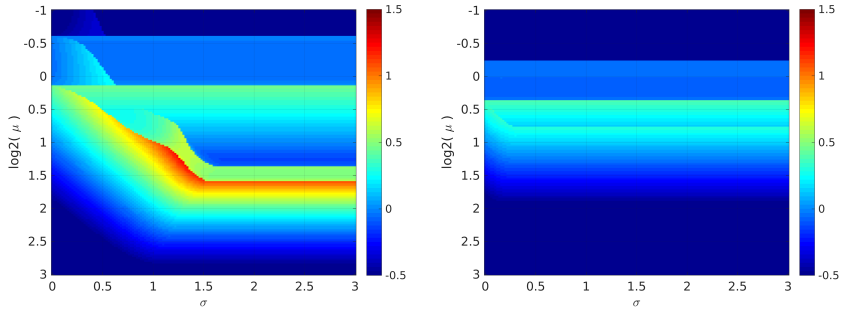


Fig. 8 Intervals $[\alpha,\beta]$ with the same width, but different location, or different width may lead to almost the same pattern as in Fig. 6 if a suitable degree d is considered. Top left: $[\alpha,\beta] = [-0.984, -0.960]$, $d = 400$; top right: $[\alpha,\beta] = [0.560, 0.584]$, $d = 1131$; bottom left: $[\alpha,\beta] = [-0.012, 0.012]$, $d = 1600$; bottom right: $[\alpha,\beta] = [0.150, 0.350]$, $d = 200$.

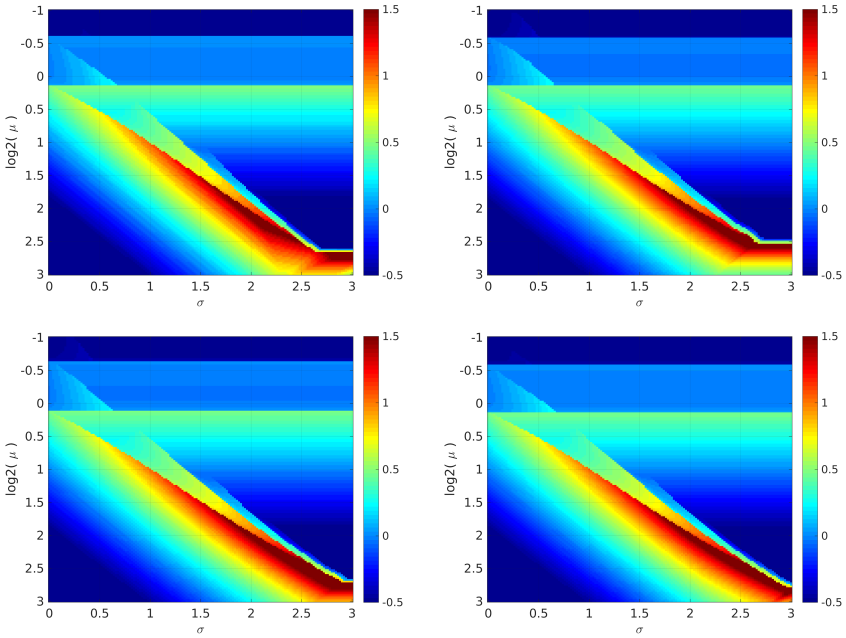
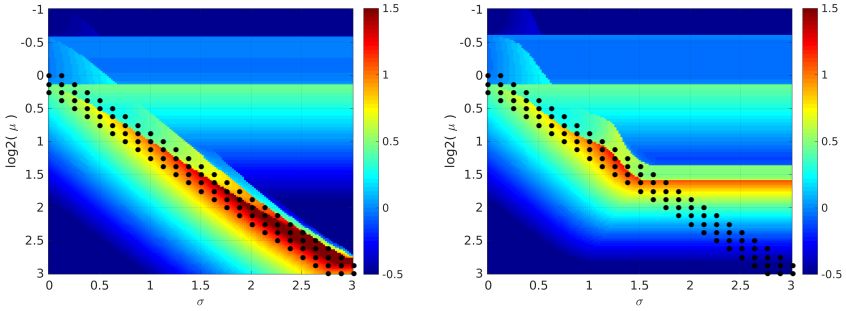


Fig. 9 The points mark the $(\log_2 \mu, \sigma)$ combinations that are considered in the BAND search; cf. Fig. 6 and Fig. 7 for the underlying $\log_2(\mathsf{gain})$ patterns.



The GRID and PATH search might also be used without a preceding BAND search, but the two-phase approach tends to be more efficient.

In a parallel setting, the evaluation of the gain at the different points in a BAND or GRID search can be done concurrently, thus requiring only one global reduction operation to determine the optimum combination $(\log_2 \mu, \sigma)$. Then each process recomputes the coefficients $\widetilde{c}_k$ corresponding to this combination to avoid global communication involving a length-$(d+1)$ vector. The PATH search has lower potential for parallelization, but tends to be more efficient serially, in particular if we avoid to re-evaluate gain for combinations that already had been considered before along the path.

5 Iteratively Compensating Filters

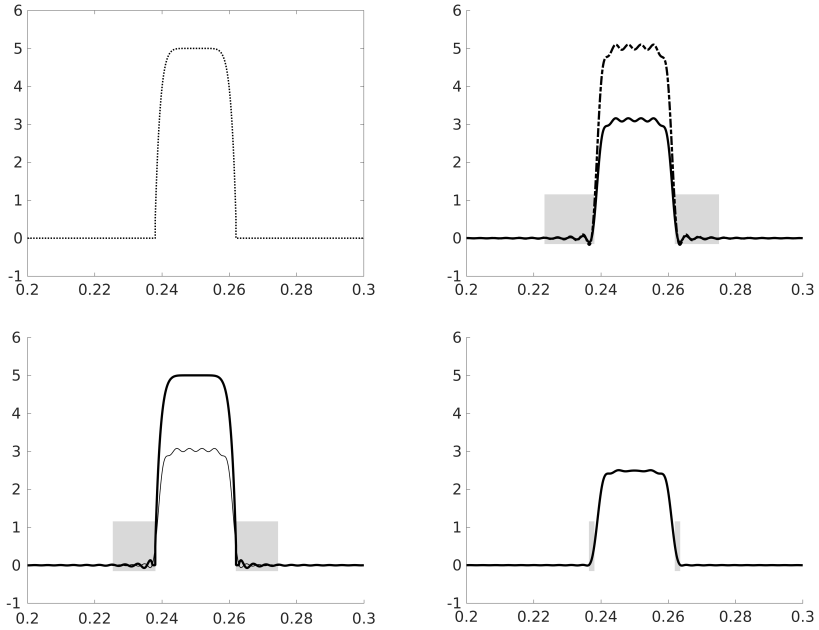
As already mentioned in Sect. 2, it is not necessary that $f(x) \approx 1$ within $[\alpha, \beta]$. Consider instead the function

$$f_1(x) = \begin{cases} f_{\max} - (f_{\max} - 0.5) \cdot \left(\dfrac{x-m}{r}\right)^{d_f}, & x \in [\alpha, \beta] \\ 0, & \text{otherwise} \end{cases},$$

where $r = (\beta - \alpha)/2$ is the radius of the interval and $m = (\alpha + \beta)/2$ is its midpoint. Thus, $f_1(x)$ is a degree-d_f monomial within $[\alpha, \beta]$, taking its maximum $f_{\max}$ at the midpoint and $f_1(\alpha) = f_1(\beta) = 0.5$ at the boundaries, and $f_1(x) \equiv 0$ outside the interval. The two parameters may be chosen freely such that $f_{\max} > \tau_{\text{inside}}$ and $d_f \geq 0$ is an even integer. See the top left picture in Fig. 10 for the function f_1 with $f_{\max} = 5$ and $d_f = 8$.

We then determine a degree-d Chebyshev approximation p_1 to f_1 and scale it to achieve $\min\{p_1(\alpha), p_1(\beta)\} = 0.5$ (top right picture in Fig. 10). To reduce the

Fig. 10 Top left: The function f_1 for $[\alpha,\beta] = [0.238, 0.262]$, $f_{\max} = 5$ and $d_f = 8$. Top right: The resulting degree-1600 approximation $p_1(x)$ before (dash-dotted) and after (solid) scaling to achieve $\min\{p(\alpha), p(\beta)\} = 0.5$. Bottom left: "Compensating" filter function f_2 (thick line) and resulting approximation p_2 (thin line). Bottom right: Final filter polynomial $p = p_{34}$.



oscillations outside $[\alpha,\beta]$, we "compensate" for them by taking the negative error $-p_1(x)$ as a target in the second step, i.e., we now approximate the function

$$f_2 = \begin{cases} f_1(x), & x \in [\alpha,\beta] \\ -\rho \cdot p_1(x), & \text{otherwise} \end{cases}$$

with a relaxation parameter $\rho > 0$ (we used $\rho = 0.75$). This is repeated until a prescribed number of iterations (e.g., 50) is reached or the margin did not improve during the last 3, say, iterations. In the example in Fig. 10 this procedure takes $34+3$ iterations to reduce the margin from $\delta \approx 0.01305$ for y_1 to 0.00150 for $y_{\text{best}} = y_{34}$.

Note that in this approach the coefficients c_k cannot be computed cheaply with a closed formula such as (4). Using the orthogonality of the Chebyshev polynomials w.r.t. the inner product

$$\langle p, q \rangle = \int_{-1}^{+1} w(\xi) p(\xi) q(\xi)\, \mathrm{d}\xi \quad \text{with} \quad w(\xi) = \frac{1}{\pi\sqrt{1-\xi^2}}\,,$$

the coefficients are given by

$$c_k = \frac{\langle f, T_k \rangle}{\langle T_k, T_k \rangle}$$

and can be obtained by numerical integration. Here, f is the target function for the current iteration. This makes determining iteratively compensating filters more expensive than optimized shrunken Lanczos filters. (Cf. also [5] for an alternative approach for approximating functions working with a modified inner product.)

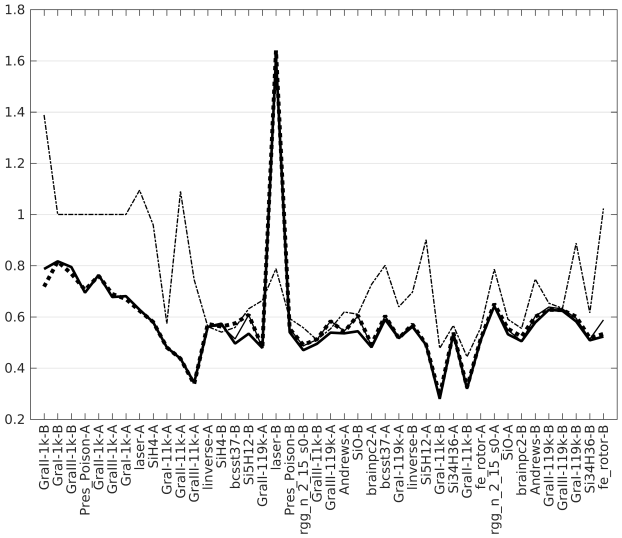
6 Numerical Experiments

So far we have focused on the gain of the improved filters, i.e. on the reduction of the margin δ where sufficient damping of unwanted eigenvalues may be violated. When applying these filters in, e.g., iterative eigensolvers then the ultimate goal is to speed up the computations. In our experiments we use the filters in a polynomial-accelerated subspace iteration with Rayleigh–Ritz extraction and adaptive control of the polynomial degrees; see [8] for a detailed description of this algorithm (BEAST-P in the ESSR).

As the overall work is typically dominated by the matrix–vector multiplications (MVMs), our first set of experiments determines if the improved filters lead to a reduction of the overall number of MVMs. We use a test set comprising 21 matrices with dimensions ranging from 1152 to 119908. Twelve matrices are taken from the University of Florida matrix collection [6] and nine come from graphene modeling. For each matrix we consider two search intervals I_λ containing roughly 300 eigenvalues. In Fig. 11 the resulting 42 problems are sorted by "hardness," i.e., by the overall number of MVMs taken by Chebyshev approximation with Lanczos kernel ($\mu = 2$). The plots show the reduction of the MVM count w.r.t. this reference filter if we use (i) the shrunken Lanczos filters described in Sect. 4 with BAND optimization alone or with BAND optimization, followed by PATH search, or (ii) the iteratively compensating filters described in Sect. 5, or (iii) a combination of both. In the latter case, we first determine the best shrunken Lanczos filter by a BAND+PATH search, and only if this did not yield a gain ≥ 2.0 then an iteratively compensating filter is determined as well, and the gain-maximal of the two filters is taken.

The data indicate that roughly 40–50% of the MVMs can be saved in most cases with the combined approach and that most of the improvement can already be achieved with just GRID optimization of shrunken Lanczos filters turned on. The effectiveness of the iteratively compensating filters as a stand-alone method tends to be inferior to filters involving shrunken Lanczos. (In general, the latter are better for critical and high degrees, whereas iteratively compensating filters may be superior for low degrees.) In a single case the improved filters involving shrunken Lanczos led to an increase of the MVMs (by 60%). This seems to be an artefact of our adaptive scheme: All variants take seven iterations to find all 304 eigenpairs contained in the search interval. With improved coefficients, 40% fewer MVMs are needed so far, but then the adaptive control fails to detect completeness and triggers additional iterations with increasing degree; we will investigate this issue further.

Fig. 11 Ratios of the overall number of matrix-vector products w.r.t. Lanczos ($\mu = 2$) for iteratively compensating filters (dash-dotted thin line), shrunken Lanczos filters with BAND optimization (dotted line) BAND and PATH optimization (solid thin line), and combined filters (see main text; solid thick line).



The above experiments were done with Matlab on machines with varying load and therefore do not provide reliable timing information. Timings were obtained with substantially larger matrices on the Emmy cluster (two 2.2GHz 10-core Xeon 2260v2 per node) at Erlangen Regional Computing Center with a parallel implementation featuring the performance-optimized kernels described in [14]. The data in Tab. 1 indicate that using the improved coefficients allowed the adaptive scheme to settle at a much lower degree for the polynomials, yielding a reduction of the MVM count and overall runtime by roughly one half. Even with the complete optimization of the coefficients done redundantly in each node, their computation took only a small amount of the overall time. Parallelizing this step as described at the end of Sect. 4 will reduce its time consumption even further.

7 Conclusions

After a very brief overview of the ESSEX project and some of its results in the first funding period, we have focused on one method used for computing a moderate number (some hundreds, say) of interior eigenpairs for very large symmetric or Hermitian matrices: subspace iteration with polynomial acceleration and Rayleigh-

Table 1 Final degree in the adaptive scheme (starting with $d = 100$ and increasing by factors of 2 or $\lfloor\sqrt{2}\rfloor$), overall number of MVMs, overall time, and time required for computing the coefficients of the polynomials, for two problems from modeling topological insulators, using standard Chebyshev approximation with Lanczos kernel ($\mu = 2$) or the improved coefficients for the filters.

Filter	Final degree	Overall MVMs	Overall time	Time for coeffs
Topological insulator, $n = 268\,435\,456$, 148 evals, 128 nodes à 20 cores				
Lanczos ($\mu = 2$)	4525	5598502	7.11 h	0.00 h
Improved (combined)	2255	2602360	3.44 h	0.02 h
Topological insulator, $n = 67\,108\,864$, 148 evals, 64 nodes à 20 cores				
Lanczos ($\mu = 2$)	2262	2726112	1.97 h	0.00 h
Improved (combined)	1127	1482035	1.10 h	0.01 h

Ritz extraction. We have presented two techniques for reducing the degree of the polynomials. One of them was based on determining standard Chebyshev approximations with suitable Lanczos kernels to a window function, but for a *shrunken* interval. The other technique was iterative, starting with a polynomial-shaped target function and trying to compensate the error made in the previous approximation. In both cases the optimization of the coefficients of the final polynomial was done with respect to the margin, i.e., the width of the area where sufficient damping of unwanted eigenpairs cannot be guaranteed. Numerical experiments showed that the polynomials thus obtained indeed also reduce the overall number of matrix–vector multiplications in the eigensolver, and thus its runtime.

Acknowledgements This work was supported by the German Research Foundation (DFG) through the Priority Programme 1648 "Software for Exascale Computing" (SPPEXA) under project ESSEX.

References

1. C. G. Baker, U. L. Hetmaniuk, R. B. Lehoucq, and H. K. Thornquist. Anasazi software for the numerical solution of large-scale eigenvalue problems. *ACM Trans. Math. Software*, 36(3):13:1–13:23, July 2009.
2. E. Bavier, M. Hoemmen, S. Rajamanickam, and H. Thornquist. Amesos2 and Belos: Direct and iterative solvers for large sparse linear systems. *Sci. Prog.*, 20(3):241–255, 2012.
3. V. Blum, R. Gehrke, F. Hanke, P. Havu, V. Havu, X. Ren, K. Reuter, and M. Scheffler. Ab initio molecular simulations with numeric atom-centered orbitals. *Comput. Phys. Comm.*, 180(11):2175–2196, 2009.
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. J. Chen, M. Anitescu, and Y. Saad. Computing $f(A)b$ via least squares polynomial approximations. *SIAM J. Sci. Comput.*, 33(1):195–222, 2011.
6. T. A. Davis and Y. Hu. The University of Florida sparse matrix collection. *ACM Trans. Math. Softw.*, 38(1):1:1–1:25, 2011.

7. V. Druskin and L. Knizhnerman. Two polynomial methods to compute functions of symmetric matrices. *U.S.S.R. Comput. Maths. Math. Phys.*, 29(6):112–121, 1989.
8. M. Galgon, L. Krämer, and B. Lang. Adaptive choice of projectors in projection based eigensolvers. Preprint BUW-IMACM 15/07, University of Wuppertal, 2015.
9. 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 Comput.*, 49:153–163, 2015.
10. S. Güttel, E. Polizzi, P. T. P. Tang, and G. Viaud. Zolotarev quadrature rules and load balancing for the FEAST eigensolver. *SIAM J. Sci. Comput.*, 37(4):A2100–A2122, 2015.
11. M. Z. Hasan and C. L. Kane. Topological insulators. *Rev. Mod. Phys.*, 82:3045–3067, Nov 2010.
12. N. J. Higham. *Functions of Matrices: Theory and Computation*. SIAM, Philadelphia, PA, 2008.
13. M. Kreutzer, G. Hager, G. Wellein, H. Fehske, and A. R. Bishop. A unified sparse matrix data format for efficient general sparse matrix-vector multiplication on modern processors with wide SIMD units. *SIAM J. Sci. Comput.*, 36(5):C401–C423, 2014.
14. M. Kreutzer, G. Hager, G. Wellein, A. Pieper, A. Alvermann, and H. Fehske. Performance engineering of the kernel polynomial method on large-scale CPU-GPU systems. In *Proc. IDPDS 2015, 29th International Parallel and Distributed Processing Symposium*, pages 417–426. IEEE Computer Society, 2015.
15. M. Kreutzer, A. Pieper, A. Alvermann, H. Fehske, G. Hager, G. Wellein, and A. R. Bishop. Efficient large-scale sparse eigenvalue computations on heterogeneous hardware, 2015. Poster at the 2015 ACM/IEEE Int. Conf. for High Performance Computing, Networking, Storage and Analysis.
16. 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. Preprint arXiv:1507.08101v2, Dec. 2015.
17. C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Nat. Bur. Standards*, 45(4):255–282, Dec. 1950.
18. A. Pieper, M. Kreutzer, M. Galgon, A. Alvermann, H. Fehske, G. Hager, B. Lang, and G. Wellein. High-performance implementation of Chebyshev filter diagonalization for interior eigenvalue computations. Preprint arXiv:1510.04895, 2015.
19. E. Polizzi. Density-matrix-based algorithm for solving eigenvalue problems. *Phys. Rev. B*, 79(11):115112, Mar. 2009.
20. M. Röhrig-Zöllner, J. Thies, M. Kreutzer, A. Alvermann, A. Pieper, A. Basermann, G. Hager, G. Wellein, and H. Fehske. Increasing the performance of the Jacobi–Davidson method by blocking. *SIAM J. Sci. Comput.*, 37(6):C697–C722, Nov. 2015.
21. A. Ruhe. Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices. *Math. Comp.*, 33(146):680–687, Apr. 1979.
22. Y. Saad. *Numerical Methods for Large Eigenvalue Problems*. SIAM, Philadelphia, PA, 2nd edition, 2011.
23. M. Schweitzer. *Restarting and error estimation in polynomial and extended Krylov subspace methods for the approximation of matrix functions*. PhD thesis, University of Wuppertal, 2015.
24. 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.
25. J. Thies, M. Galgon, F. Shahzad, A. Alvermann, M. Kreutzer, A. Pieper, M. Röhrig-Zöllner, A. Basermann, H. Fehske, G. Hager, B. Lang, and G. Wellein. Towards an exascale enabled sparse solver repository. Preprint, 2015. http://elib.dlr.de/100211/.
26. A. Weiße, G. Wellein, A. Alvermann, and H. Fehske. The kernel polynomial method. *Rev. Mod. Phys.*, 78(1):275–306, Jan. 2006.