Rational spectral filters with optimal convergence rate

SIAM J. SCI. COMPUT.
Vol. 43, No. 4, pp. A2660–A2684

RATIONAL SPECTRAL FILTERS WITH OPTIMAL CONVERGENCE RATE*

KONRAD KOLLNIG†, PAOLO BIENTINESI‡, AND EDOARDO A. DI NAPOLI§

Abstract. In recent years, contour-based eigensolvers have emerged as a standard approach for the solution of large and sparse eigenvalue problems. Building upon recent performance improvements through nonlinear least-squares optimization of so-called rational filters, we introduce a systematic method to design these filters by minimizing the worst-case convergence rate and eliminate the parametric dependence on weight functions. Further, we provide an efficient way to deal with the box-constraints which play a central role for the use of iterative linear solvers in contour-based eigensolvers. Indeed, these parameter-free filters consistently minimize the number of iterations and the number of FLOPs to reach convergence in the eigensolver. As a byproduct, our rational filters allow for a simple solution to load balancing when the solution of an interior eigenproblem is approached by the slicing of the sought after spectral interval.

Key words. Hermitian eigenvalue problem, contour-based eigensolver, worst-case convergence rate, load balancing, nonlinear least squares, BFGS

AMS subject classifications. 65F15, 15A18, 65F50

DOI. 10.1137/20M1313933

1. Introduction.

For the Hermitian eigenproblem $Ax = \lambda x$ with $\lambda \in [a, b] \subset \mathbb{R}$, the last decade has seen the emergence of a new class of eigensolvers based on spectral projectors. Such eigensolvers are typically expressed as integrals of the spectral resolvent $(A - zI)^{-1}$ over a contour in the complex plane that encloses the interval $[a, b]$ [34, 35, 32, 8, 21]. Numerical quadrature transforms the contour integral into a matrix-valued rational function with complex coefficients β_i and poles z_i. In this form, the problem of finding an efficient spectral projector is mapped to that of finding a rational function—often referred to as rational filter—that approximates the indicator function

$$\mathbb{1}_{(a,b)}(x) = \begin{cases} 1 & \text{if } x \in [a, b], \\ 0 & \text{otherwise.} \end{cases} \tag{1.1}$$

This is a discontinuous function, often termed the "ideal filter," because it exactly maps the desired eigenvalues in the interval to 1 and the rest of the spectrum to 0.

The algorithmic structure of eigensolvers based on rational filters has the advantage of lending itself to parallel implementations with multiple levels of nested parallelism [7, 5]. On the other hand, several factors make load balancing for these parallel eigensolvers a potential nightmare [9, 11]. Among them, the design of the filter is an important element that influences the convergence of the eigensolver with direct consequences on the load balancing of any parallel implementation based on slicing $[a, b]$ in subintervals. In this paper, we focus on the design of filters with the

*Submitted to the journal's Methods and Algorithms for Scientific Computing section January 21, 2020; accepted for publication (in revised form) March 31, 2021; published electronically July 29, 2021.

https://doi.org/10.1137/20M1313933

†Computer Science, RWTH Aachen University, 52074, Aachen, Germany (konrad.kollnig@rwth-aachen.de).

‡Department of Computing Science, Umeå University, Umeå, 901 87, Sweden (pauldj@cs.umu.se).

§Juelich Supercomputing Centre, Forschungszentrum Juelich, 52425, Juelich, Germany (e.di.napoli@fz-juelich.de).

aim of resolving this open issue. We build upon the results presented in [40] and introduce an optimization framework that is versatile and fast, eliminates parameter dependencies, and ultimately produces highly accurate rational filters with respect to a metric tightly bound to the quality of the ideal filter. Numerical tests show that an eigensolver equipped with our spectral projector converges with a rate that is practically independent of the search space size, the number of poles of the rational filter, and the number of iterations required.

When A is a Hermitian matrix, the corresponding rational filter is real-valued and symmetric with respect to the mapping $(x - x_0) \leftrightarrow (x_0 - x)$, where x_0 is the center point of the interval $[a, b]$. Taking into consideration the complex conjugation and parity symmetries, Winkelmann and Di Napoli [40] write r as a rational function of order $(4m - 1, 4m)$,

$$r(x) := r_{\beta,z}(x) := \sum_{i=1}^{m} \frac{\beta_i}{x - z_i} + \frac{\overline{\beta_i}}{x - \overline{z_i}} - \frac{\beta_i}{x + z_i} - \frac{\overline{\beta_i}}{x + \overline{z_i}}, \quad x \in \mathbb{R}, \tag{1.2}$$

where $m \in \mathbb{N}$, $\beta = (\beta_1, \ldots, \beta_m) \in \mathbb{C}^m$, and $z = (z_1, \ldots, z_m) \in (\mathbb{H}^{+R})^m$ with $\mathbb{H}^{+R}$ being the right quadrant of the upper half of $(\mathbb{C} \backslash \mathbb{R})$ with origin in x_0. With this setup, the problem to be addressed is how to select, for a fixed degree m,[1] the coefficients β_i and the poles z_i such that the corresponding rational function $r(x)$ approximates the ideal filter $\mathbb{1}_{(a,b)}$ according to a predetermined metric. Our aim is to build an optimization framework and select an appropriate metric such that the outcome is a filter r stabilizing the convergence of the eigensolver.

Due to the discontinuity of the indicator function $\mathbb{1}_{(a,b)}$, the problem of determining the best coefficients and poles for $r(x)$ is tackled using a nonlinear weighted least-squares approach. For a given interval $[a, b]$, one aims to minimize the objective function

$$f_\omega(\beta, z) := \int_{-\infty}^{\infty} \omega(x) \left(\mathbb{1}_{(a,b)}(x) - r_{\beta,z}(x)\right)^2 \, \mathrm{d}x, \quad \text{where } \beta \in \mathbb{C}^m, z \in (\mathbb{H}^{+R})^m, \tag{1.3}$$

over β and z for some fixed $m \in \mathbb{N}$ and a weight function $\omega(x) : \mathbb{R} \to [0, \infty)$, which is even with respect to x_0 and piecewise constant. This optimization framework, termed Symmetric nonLinear Optimized Least-Squares (SLiSe) in [40], provides a comprehensive parameterization of rational filters. The resulting SLiSe filters have proven to be competitive with previous rational filters, such as Gauss–Legendre [32] and Zolotarev [11].

The SLiSe framework is independent of the specific eigensolver in which the function r is plugged into and used as a spectral filter. At a glance, a filter optimized through this framework should perform well independently of the target eigenproblem. In practice, the effectiveness of a filter depends indirectly on the eigenvalue distribution around the interval $[a, b]$ through the choice of the weight function ω. In other words, despite its versatility, the SLiSe framework outputs filters whose quality is sensitive to the ad hoc choice of weight functions and the piecewise intervals defining them: small changes in the choice of $\omega(x)$ greatly influence the effectiveness of the resulting filter.

Contributions. Building on top of the SLiSe framework, this work addresses problematic aspects of such optimization and ultimately provides a solution to the open

[1] Strictly speaking the degree of r is $4m$. In the rest of the paper we will stick to a more intuitive notion of degree which refers to the number of poles in $\mathbb{H}^{+R}$ corresponding to the range of the index i in (1.2).

issue of how spectral filters influence load balancing. In detail, we identify a number of main contributions. We improve the performance of the unconstrained minimization process by substituting the Levenberg–Marquardt with the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [31, Chapter 6]. Likewise, when SLiSe is used in combination with box-constraints, it is natural to extend BFGS to the L-BFGS-B algorithm [2, 42, 25]. Using the BFGS family of algorithms results in a substantial reduction of the memory footprint and time-to-solution, which in turn is a necessary requirement to reduce the objective function residual and, at the same time, increases the accuracy of the SLiSe filters. We increase the accuracy by casting the problem of selecting β_is and z_is in terms of finding the corresponding rational function $r(x)$ that minimizes the worst-case convergence rate (WCR).[2] The relevance of this metric resides in the fact that the ideal filter $\mathbb{1}_{(a,b)}$ has the lowest possible value for WCR, which is 0.

In order to use the WCR metric effectively, we embed the SLiSe framework, equipped with the BFGS algorithm, within a second minimization process. This process has the explicit goal of minimizing the WCR metric with respect to the weight function ω. We attain this target by using the derivative-free Nelder–Mead algorithm. The byproduct of this process is eliminating the dependence on the arbitrary choice of ω in the definition of the objective function $f_\omega(\beta, z)$. The net result is a parameter-free minimization framework with an enhanced usability and productivity. When used in interior eigensolvers based on subspace iteration, we observe that the rational filters obtained with the new minimization framework outperform state-of-the-art filters. The convergence rate of the eigensolver becomes almost independent of the size of the search subspace and the number of poles used. Consequently, the eigensolver is more robust in terms of convergence rate and does not require tweaking of the parameters associated with the spectral projection. In turn, this enhanced behavior of the eigensolver facilitates the load balancing when executed on parallel platforms. We termed this enhanced minimization framework, and the corresponding rational filters it produces, Worst-Case Optimized Least-Squares (WiSe).

Related work. The interpretation of spectral projectors as rational (filter) functions of matrices was discussed in [38] for FEAST and in [13, 12] for Sakurai–Sugiura-type eigensolvers. Rational filters were also proposed early on for signal processing by Murakami [30, 29, 26, 28, 27]. In recent years, filters have been treated as a parameter that can be designed via optimization methods. Van Barel [39] suggested a nonlinear least-squares approach for non-Hermitian filters to be used in the Sakurai–Sugiura framework, while Xi and Saad [41] described linear least-squares optimized filters for the Hermitian FEAST eigensolver. Van Barel's approach is based on the discrete ℓ_2 norm, not a functional approximation approach, and does not support constraints optimization. Xi and Saad presented a linear least-squares minimization method where only the coefficients of the rational function are optimized. For FEAST, Güttel et al. [11] presented a first approach to minimizing the WCR in (2.7). They derived a set of *generalized* Gauss–Legendre filters, parameterized by one variable only, with respect to which they minimized the WCR functional. The resulting WCR values were smaller than for unparameterized Gauss–Legendre filters, but not as small as for Zolotarev filters [32, 11, 1, 10] which offered the best WCR so far. This observation motivated a more rigorous parameterization of a subset of rational filters, that is, SLiSe filters [40], so as to benefit from a reduced number of parameters within WCR minimization.

[2] This metric is defined in the next section.

Organization. The remainder of this paper is organized as follows. In section 2 we introduce the reader to spectral filters and the general mathematical setup. In section 3, we review the SLiSe framework and introduce efficient imposition of box-constraints on rational filters through L-BFGS-B. In section 4, we illustrate the minimization scheme to reduce the WCR of SLiSe filters, which in turn eliminates the dependence on weight functions. In section 5, we present a set of numerical experiments comparing our new filters to the state of the art and illustrate their numerical properties and advantages. The last section summarizes our results and provides a perspective on their impact on the load balancing of parallel interior eigenvalue solvers.

2. Methodology. Contour-based eigensolvers were originally conceived for the solution of the generalized interior eigenvalue problem

$$Av = \lambda Bv, \quad \lambda \in [a, b], \tag{2.1}$$

where $A, B \in \mathbb{C}^{n\times n}$ are Hermitian and B is positive definite, $v \in \mathbb{C}^n \setminus \{0\}$, $a < b$, and $n \in \mathbb{N}$. A spectral projector can be defined as the integral of the matrix resolvent $(A - zB)^{-1}$ along a contour Γ in the complex plane $\mathbb{C}$ enclosing the interval $[a, b] \subset \mathbb{R}$. Without loss of generality, one can linearly map $[a, b]$ to the standard interval $[-1, 1]$ and select an integration contour around it. It is standard practice to compute the contour integral via numerical quadrature (e.g., Gauss–Legendre):

$$r(A, B) := \sum_i \beta_i (A - Bz_i)^{-1} B \approx \frac{1}{2\pi i} \oint_\Gamma \frac{\mathrm{d}z}{A - Bz} B \tag{2.2}$$

with $\beta_i,\ z_i \in \mathbb{C}$. When used in combination with a subspace iteration scheme, $r(A, B)$ projects a given set of vectors Y onto an invariant subspace of the spectrum corresponding to the eigenvalues within the interval $[-1, 1]$ [18]. In practice, spectral projection exchanges the direct solution of the eigenproblem for that of many independent linear systems, each with multiple right-hand sides:

$$(A - Bz_i)V = \beta_i BY. \tag{2.3}$$

Because each linear system can be solved independently of the others and the interval $[a, b]$ can be sliced into subintervals, this class of eigensolvers naturally lends itself to multiple layers of parallelism. Thanks to the natural parallelism of the underlying algorithm, contour-based eigensolvers are especially well suited for today's increasingly parallel computer architectures. This is demonstrated by the proliferation, in the last ten years, of parallel implementations in software packages and libraries [36, 17, 22, 33].

As shown in several recent publications, the performance of the eigensolver depends on the effectiveness of the spectral filter $r(A, B)$ [18, 38, 11, 1, 4, 8]. Recently, the authors of [40] proposed a numerical optimization approach alternative to the standard quadrature rules. By minimizing the objective function of (1.3), they propose a new class of rational filters, termed SLiSe, which perform better than the filters currently in use, on a large number of representative eigenproblems. Despite such an advance, the SLiSe framework showed a few shortcomings, such as slow convergence and lack of efficient support for box-constraints. These box-constraints—defined as upper and lower bounds on the imaginary parts of each z_i—can substantially influence the time-to-solution in iterative linear system solvers. Having a time-to-solution comparable across all linear systems is a crucial element to load balance a parallel eigensolver based on spectral projection.

The SLiSe filters. The SLiSe minimization framework aims to approximate the indicator function $\mathbb{1}_{(-1,1)}$ by rational filters $r(x)$ of a fixed degree m. This approximation is obtained by minimizing the objective function $f_\omega(\beta, z)$ from (1.3). In the SLiSe framework, a new filter is obtained as follows: Given a fixed weight function ω and an $m \in \mathbb{N}$, SLiSe takes an existing rational filter $r_{\bar{\beta},\bar{z}}$, where $\bar{\beta} \in \mathbb{C}^m$ and $\bar{z} \in (\mathbb{H}^{+R})^m$, and derives a new rational filter $r_{\hat{\beta},\hat{z}}$ such that $(\hat{\beta}, \hat{z})$ solves the minimization problem

$$\operatorname*{argmin}_{\beta \in \mathbb{C}^m, z \in (\mathbb{H}^{+R})^m} f_\omega(\beta, z). \tag{2.4}$$

This minimization problem is nonlinear and nonconvex and therefore difficult to solve due to the nonexistence of closed-form solutions. Yet, the objective function f_ω, as well as its gradient

$$\nabla f_\omega = (\nabla_{\beta_1} f_\omega, \ldots, \nabla_{\beta_m} f_\omega, \nabla_{z_1} f_\omega, \ldots, \nabla_{z_m} f_\omega)^\top, \tag{2.5}$$

are differentiable and can be computed through a small number of matrix operations [40]. In this setup, one can make use of a wide range of existing numerical minimization methods. Winkelmann and Di Napoli obtain SLiSe filters by employing two such minimization methods, gradient descent and Levenberg–Marquardt (LM).

While LM makes for an effective minimization scheme, it may require up to thousands of iterations to converge to a satisfactory value for the residual level of f_ω. Executing an efficient minimization becomes a pressing problem in the case of box-constrained optimization, when the LM algorithm cannot be used and gradient descent requires up to millions of iterations to converge, which translates to a significantly larger amount of computing time over the unconstrained case. In addition, and most importantly, the quality of a resulting filter depends on the choice of weight function ω, which is not automatic and requires an experienced user to follow a set of guidelines. In the following, we illustrate a minimization scheme that ensures speed of convergence, supports box-constraints, and eliminates the dependence on the custom choice of weight functions ω.

The new minimization scheme. In the rest of the paper, we refer to r as a rational filter and, without loss of generality, consider only the case $r(A, B = I) = r(A)$. As seen in the previous section, if A is a Hermitian matrix, the corresponding rational function $r(x)$ is forced to be real and symmetric and can be expressed with a subset of poles and coefficients as in (1.2). Since the minimization of the objective function in (1.3) is completely general, the resulting filter is independent of the specific subspace iteration eigensolver and can be plugged into any eigensolver of this type. Nonetheless, for practical purposes, we use the FEAST eigensolver [5] as a reference algorithm. Given an exact value $\lambda_j \in [-1, 1]$, FEAST computes an approximate eigenpair $(\boldsymbol{q}_j, \hat{\lambda}_j)$ with a residual vector norm equal to $\|A\boldsymbol{q}_j - \hat{\lambda}_j \boldsymbol{q}_j\|$. Such a residual converges linearly with a convergence rate given by $|\gamma_{\text{out}}/\gamma_{\text{in}}|$, where γ_{out} (γ_{in}) is related to the maximum (minimum) value of the filter outside (inside) a neighborhood enclosing the $[-1, 1]$ interval [38, Theorem 5.2]. Consequently, the convergence rate depends on both the spectrum of the given matrix A and the spectral filter of choice.

Although the actual convergence rate will vary for different spectra, a filter-dependent upper bound is given by the WCR. The WCR applies to a variety of other eigensolvers based on spectral projection such as the block Sakurai–Sugiura–Rayleigh–Ritz method [35] and its noniterative variant [14]. As defined in [11], the WCR satisfies the following theorem.

THEOREM 2.1 ([11, Theorem 2.2]). *Given a rational filter r and a fixed gap parameter $G \in (0,1)$, the FEAST method converges linearly, with probability one, at a convergence rate no larger than*

$$w_G(r) = \frac{\max_{x\in[-\infty,-G^{-1}]\cup[G^{-1},\infty]} |r(x)|}{\min_{x\in[-G,G]} |r(x)|}, \tag{2.6}$$

as long as no eigenvalues lie within $[-G^{-1},-G]\cup[G,G^{-1}]$. The occurring probability stems from choosing the initial subspace within the FEAST method at random.

Since Theorem 2.1 implies that $|\gamma_{\text{out}}/\gamma_{\text{in}}| \le w_G(r)$, for an appropriate G, a smaller WCR value $w_G(r)$ implies faster worst-case convergence. As we already mentioned in the Introduction, minimizing WCR for the SLiSe filters points out which filters best approximate the ideal filter $\mathbb{1}_{(-1,1)}$ (that has indeed the optimal bound $w_G(\mathbb{1}_{(-1,1)}) = 0$). Based on the considerations above, we can now define the following optimization problem.

DEFINITION 2.2. *Given $G \in (0,1)$, $m \in \mathbb{N}$, and $r_{\beta,z}$, a rational filter as defined in* (1.2)*, an optimal rational filter is one solving the minimization problem*

$$\operatorname*{argmin}_{\beta\in\mathbb{C}^m, z\in(\mathbb{H}^{+R})^m} w_G(r_{\beta,z}). \tag{2.7}$$

In general, the WCR is a nonlinear, derivative-free function. Its formulation makes it difficult to determine further mathematical properties, such as convexity or continuity. Conventional methods, like steepest descent, cannot be applied. Additionally, in derivative-free minimization, the number of w_G function evaluations may become intractable very quickly, even for a modest increase of the rational filter degree m. These observations cause this minimization problem to be especially challenging. Instead of solving the problem as formulated in (2.7), we propose a modified minimization problem that combines the existing SLiSe framework, solving for (β, z) while ω is fixed as in (2.4), with the minimization of the WCR with respect to ω, seeking a better ω while (β, z) is fixed. These two minimization problems,

$$\begin{cases} \beta, z & \leftarrow \operatorname*{argmin}_{\beta,z} f_\omega(\beta, z) \qquad \text{for a fixed } \omega, \\ \omega & \leftarrow \operatorname*{argmin}_{\omega} w_G(r_{\beta,z}(\omega)) \quad \text{for an initial pair } (\beta, z), \end{cases} \tag{2.8}$$

are clearly not independent. The WCR is minimized solely with respect to the weight function ω—where we have indicated explicitly the dependence of the rational filter on the weight function in (2.8)—but it is a nonlinear, derivative-free function. As such, the WCR depends on the whole r which, in turn, depends on the minimization of the objective function f_ω. In other words, we now have to solve two nonlinearly dependent minimization problems, which need to be solved self-consistently. We will see in section 4 how we implement this process in a nested loop fashion, where the SLiSe process is executed within the WCR minimization and convergence is reached self-consistently by continuously swapping between the two minimizations. Solving (2.8) is now a tractable problem, even if it calls for sophisticated algorithms, and is computationally very intensive, requiring many repeated invocations of the SLiSe minimization.

In order to increase the performance of the self-consistent minimization, we introduce the BFGS algorithm [31, Chapter 6] within the unconstrained SLiSe minimization process. Similarly, for box-constraints, we present an embedding of the

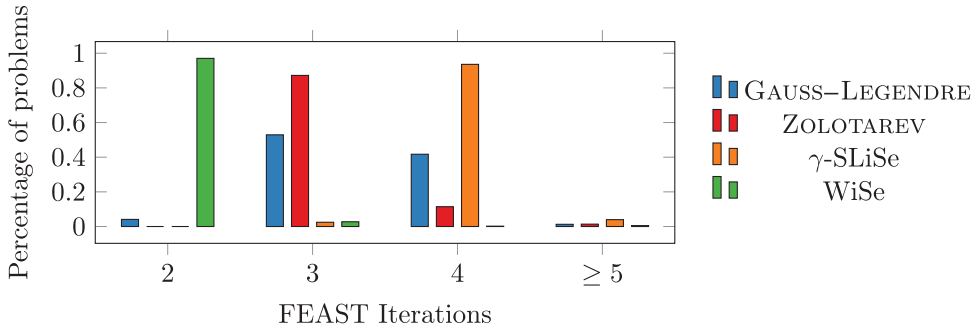


FIG. 1. *FEAST iterations for different filters with gap parameter* $G = 0.95$ *to solve* 2117 *benchmark eigenproblems, for an eigencount multplier of* $C = 1.1$. *Our new WiSe filter outperforms the others, i.e., generalized* GAUSS–LEGENDRE [32, 11], ZOLOTAREV [11], *and* γ*-SLiSe* [40]. *Details are discussed in section* 5.3.2.

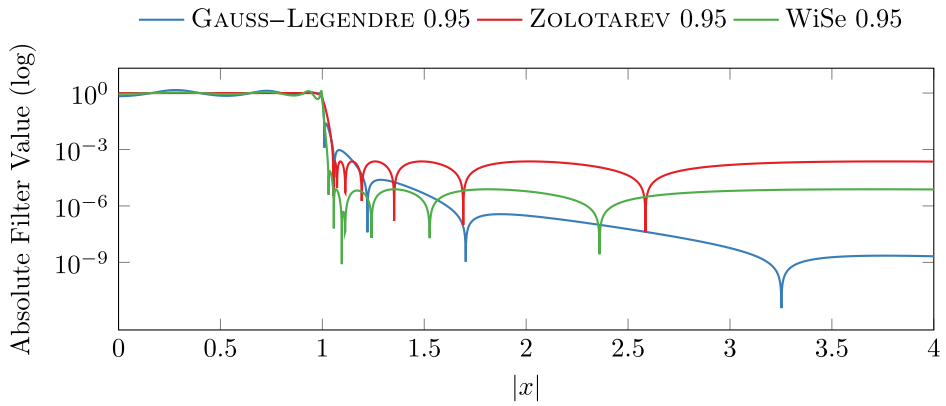


FIG. 2. *Logarithmic plot of different* 16*-pole rational filters, showing the state-of-the-art* GAUSS–LEGENDRE *and* ZOLOTAREV *rational filters* [11, 32], *alongside our new filter candidate, WiSe. A moderate gap parameter* $G = 0.95$ *was chosen for* ZOLOTAREV, *WiSe, and* GAUSS–LEGENDRE. *Our new WiSe filter provides the best WCR, which can be seen in offering the sharpest slump for* x *around* 1, *and maintaining a constantly close approximation of the indicator function* $\mathbb{1}_{(-1,1)}$ *across the whole domain of real numbers.*

L-BFGS-B minimization method [2, 42, 25] into SLiSe. By formulating the WCR minimization as a nested process, we additionally solve the issue of weight function selection, which is one of the open issues of SLiSe. The net result is an extension of the SLiSe framework toward rational filters for faster convergence, without the need to select the weight functions by hand. Our new rational filters, termed WiSe, outperform state-of-the-art filters (see Figure 1 for experimental results and Figure 2 for filter plots). In particular, we prove that ZOLOTAREV filters do not provide best worst-case convergence, despite their optimality in approximating the indicator function with respect to the ∞-norm.

3. Efficient computation of WiSe filters. In this section, we introduce the use of the BFGS algorithm, which yields faster convergence and better box-constrained rational filters than previous implementations. Moreover, the extended L-BFGS-B successfully addresses open issues that appear in box-constrained filters [11, 40, 8].

3.1. Accelerating SLiSe. When using the BFGS algorithm to solve the minimization problem in (2.4), we end up reducing substantially the number of function evaluations needed to reach convergence. Seemingly minor, this improvement is actually essential for an effective embedding of SLiSe into a scheme that is based on the minimization of the WCR. The BFGS algorithm belongs to the class of quasi-Newton methods. It approximates a local minimizer iteratively, in a manner similar to the popular Gauss–Newton algorithm, which is Hessian-based. However, unlike Gauss–Newton, BFGS does not require the exact Hessian $\nabla^2 f$ and uses an approximation instead. The minimum requirement for the algorithm to work is that the function f has a quadratic Taylor series expansion about its minimum. Thanks to this weaker condition, BFGS guarantees convergence also for nonsmooth and non-convex functions.

The standard implementation of the BFGS variant in Algorithm 3.1 does not offer support for real-valued objective functions of complex arguments, such as our f_ω from (1.3). This problem can be overcome by a conversion of f_ω and ∇f_ω to real arguments. In the case of a generic function of complex variables $g : \mathbb{C}^n \to \mathbb{R}$, one can separate the real from the imaginary parts [37] and instead minimize the function $\tilde{g} : \mathbb{R}^{2n} \to \mathbb{R}$, defined as

$$\tilde{g}\left(\begin{pmatrix} a \\ b \end{pmatrix}\right) := g(a + ib) \quad \text{for } a, b \in \mathbb{R}^n, \tag{3.1a}$$

by computing descent directions from its gradient

$$\nabla\tilde{g}\left(\begin{pmatrix} a \\ b \end{pmatrix}\right) = \begin{pmatrix} \operatorname{Re}\nabla g(a + ib) \\ \operatorname{Im}\nabla g(a + ib) \end{pmatrix} \quad \text{for } a, b \in \mathbb{R}^n. \tag{3.1b}$$

The same mapping can be applied to the SLiSe functional $f_\omega : \mathbb{C}^m \times (\mathbb{H}^{+R})^m \to \mathbb{R}$ because it operates on a subset of $\mathbb{C}^{2m}$, where $m \in \mathbb{N}$ is the degree of the rational filter. In this case, one can think of the complex vectors β and z as being part of a vector $v = (\beta\ z)^\top$ and define $\tilde{f} : \mathbb{R}^{4m} \to \mathbb{R}$ such that

$$\tilde{f}\begin{pmatrix} \begin{pmatrix} \operatorname{Re}(\beta^\top) \\ \operatorname{Re}(z^\top) \end{pmatrix} \\ \begin{pmatrix} \operatorname{Im}(\beta^\top) \\ \operatorname{Im}(z^\top) \end{pmatrix} \end{pmatrix} := f(\operatorname{Re}(\beta) + i\operatorname{Im}(\beta), \operatorname{Re}(z) + i\operatorname{Im}(z)). \tag{3.2}$$

Starting at an initial point $x_0 = (\operatorname{Re}(\beta)\ \operatorname{Re}(z)\ \operatorname{Im}(\beta)\ \operatorname{Im}(z))^\top$, BFGS computes iterates x_k that converge to a local minimizer of $\tilde{f}$ as $k \in \mathbb{N}$ increases, employing the descent directions

$$p_k := -H_k\ \nabla\tilde{f}(x_k) \tag{3.3a}$$

and a line search which guarantees that the secant condition is satisfied (see `line 7` of Algorithm 3.1). H_k is an approximation to the inverse Hessian of $\tilde{f}$ and is recursively defined as

$$H_0 := I_{4m}, \quad H_{k+1} := \left(I_{4m} - \frac{s_k y_k^T}{y_k^T s_k}\right)\ H_k\ \left(I_{4m} - \frac{y_k s_k^T}{y_k^T s_k}\right) + \frac{s_k s_k^T}{y_k^T s_k} \tag{3.3b}$$

with

$$s_k := x_{k+1} - x_k, \quad y_k := \nabla\tilde{f}(x_{k+1}) - \nabla\tilde{f}(x_k), \tag{3.3c}$$

Algorithm 3.1 (unconstrained SLiSe through BFGS algorithm).

procedure SLiSe(β, z, ω)
  Define $x \leftarrow (\mathrm{Re}(\beta), \mathrm{Re}(z), \mathrm{Im}(\beta), \mathrm{Im}(z))^\top \in \mathbb{R}^{4m}$
  Compute the objective function $\tilde{f}(x)$ and its gradient $\nabla\tilde{f}(x)$
  $H \leftarrow I_{4m}$
  while $||\nabla\tilde{f}(x)|| > \varepsilon$ **do** ▷ Default value $\varepsilon = 10^{-8}$
    $p \leftarrow -H\ \nabla\tilde{f}(x)$ ▷ Obtain descent direction
    Choose an $\alpha \in \mathbb{R}^+$ to minimize $\tilde{f}(x + \alpha p)$ over α ▷ Ensuring $s^\top y > 0$
    $s \leftarrow \alpha\, p$
    $w \leftarrow x + s$
    $y \leftarrow \nabla\tilde{f}(w) - \nabla\tilde{f}(x)$
    $H \leftarrow (I_{4m} - \frac{sy^\top}{y^\top s})\ H\ (I_{4m} - \frac{ys^\top}{y^\top s}) + \frac{ss^\top}{y^\top s}$ ▷ Approximate inverse Hessian
    $x \leftarrow w$
  end while
  $\beta' \leftarrow (x_{1:m} + i x_{2m+1:3m})^\top, \quad z' \leftarrow (x_{m+1:2m} + i x_{3m+1:4m})^\top$
  return $r_{\beta',z'}$
end procedure

where $x_k, s_k, y_k, p_k \in \mathbb{R}^{4m}$ and $H_k \in \mathbb{R}^{4m\times 4m}$ for some $m \in \mathbb{N}$. The formulation through the BFGS algorithm converges faster than the previous minimization algorithms used by the SLiSe framework (see Figure 3 for the box-constrained case that is discussed in the following subsection). The conversion of the minimization functional to real arguments allows one to use not only the BFGS scheme but also various other minimization algorithms (such as those in the minimization algorithm collection NLOpt [15]). Despite such an advantage, most alternatives do not yield any substantial improvements over BFGS.

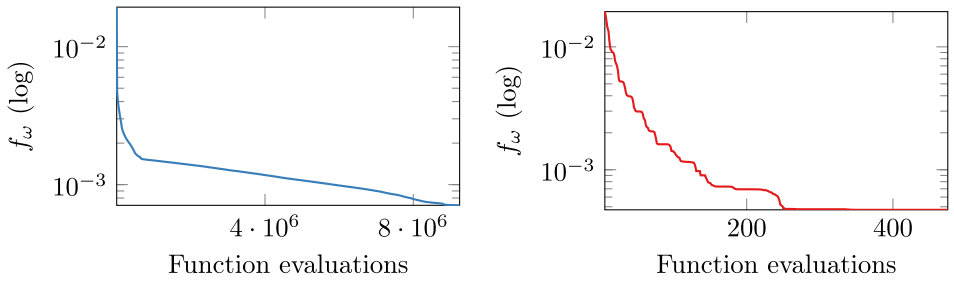
3.2. Imposing box-constraints efficiently. As described at the beginning of section 2, the spectral projection at the base of the FEAST eigensolver leads to the solution of several independent linear systems with multiple right-hand sides (see (2.3)). In the case of very large and sparse systems, the use of direct solvers is not feasible due to memory requirements. In this case, iterative solvers, such as GMRES or CG, are the natural choice. For these methods, time-to-solution and accuracy depend substantially on the condition number of the resolvent matrices $(A - z_i I)$. When A is Hermitian, such a condition number is, up to a constant factor, equal to

$$\kappa(A - z_i I) = \frac{\max_{\lambda_a\in\sigma(A)} |\lambda_a - z_i|}{\min_{\lambda_b\in\sigma(A)} |\lambda_b - z_i|}. \tag{3.4}$$

Since the filter is built to approximate the indicator function $\mathbb{1}_{(-1,1)}$, the numerator of this equation is bound from above by $(\max_{\lambda_a\in\sigma(A)} |\lambda_a| + 1)$, while the denominator is bound from below by $|\mathrm{Im}(z_i)|$. Consequently, if the poles of the rational function $r_{\beta,z}$ are close to the real axis, the condition number of some of the resolvent matrices can be quite high. This consideration motivated the introduction of the box-constraints $|\mathrm{Im}(z_i)| \geq \mathtt{lb} > 0$ $(i = 1, \ldots, m)$ to the SLiSe minimization process, where `lb` is a positive constant representing the minimum distance of any pole from the real axis.

In SLiSe, the box-constrained minimization was implemented through a simple projected gradient descent method, where each new step is computed by the simple gradient projection

$$x_{k+1} = \mathcal{P}(x_k - t\nabla_x \tilde{f}(x)|_{x=x_k})$$



(a) Projected gradient descent (b) L-BFGS-B

FIG. 3. *Box-constrained minimization of the functional f_ω using projected gradient descent and L-BFGS-B, respectively. The setup is taken from the original publication* [40]*, using the* 16*-degree* ZOLOTAREV *filter as the starting point and a lower bound of* $\texttt{lb} = 0.0022$ *on the absolute value of the imaginary parts of the poles. The L-BFGS-B method settles at a smaller residual and converges substantially faster, requiring only a few function evaluations.*

with $t > 0$. In the BFGS algorithm, box-constraints can be included by projecting the search direction onto the constraints. This is accomplished through the same gradient projection $\mathcal{P}$, followed by a BFGS update treating the bounded components of x as equality constraints. In our case, the operator $\mathcal{P} : \mathbb{R}^{4m} \to \mathbb{R}^{4m}$ projects only the imaginary part of the poles $\mathrm{Im}(z_i)$ and takes consequently the following form when acting on a vector $y \in \mathbb{R}^{4m}$:

$$\mathcal{P}(y)_j := \begin{cases} \mathrm{sign}(y_j) \cdot \texttt{lb} & \text{if } |y_j| < \texttt{lb} \text{ and } j \in \{\, 3m+1, \ldots, 4m \,\}, \\ y_j & \text{otherwise} \end{cases} \tag{3.5}$$

for $j = 1, \ldots, 4m$.

This approach is encoded in the L-BFGS-B algorithm, which extends projected gradient descent to the Hessian approximations from BFGS and can be used to realize box-constrained minimization efficiently in SLiSe, similar to what is done in Algorithm 3.1. The L-BFGS-B algorithm has shown to converge quickly in our experiments when compared with projected gradient descent. In terms of both speed and accuracy, the use of L-BFGS-B places the constrained SLiSe method on par with the unconstrained BFGS algorithm. To illustrate the increase in performance caused by L-BFGS-B, we compare box-constrained minimization through our L-BFGS-B implementation against the projected gradient descent implemented in the original SLiSe framework. Figure 3(a) and (b) show the number of function evaluations carried out by the projected gradient descent and the L-BFGS-B algorithms, respectively. L-BFGS-B requires four orders of magnitude fewer evaluations than projected gradient descent and converges to a smaller residual.

So far, the procedure used to obtain the SLiSe filters depends on the specific form of a given weight function ω. For some such weight functions, the computed filters were shown to outperform state-of-the-art rational filters. Yet, the only criterion known to determine suitable weight functions is comparing hand-crafted weight functions on a large set of representative interior eigenproblems. While guidelines for the construction of ω have been devised [40], the choice of weight functions remains a complex issue. In the following section, we propose an algorithm to obtain weight functions which yield SLiSe filters with reduced WCR and overcome the necessity of selecting weight functions manually.

4. SLiSe filters with reduced WCR. In this section, we illustrate how to reduce the WCR of a given SLiSe filter by improving on the choice of weight function ω. We achieve this by minimizing a new objective function, closely related to the WCR of rational filters.

4.1. Parameterization of weight functions. Weight functions are even, nonnegative, piecewise constant functions that are used in the definition of the SLiSe functional f_ω in (1.3). This means that a weight function can be characterized in terms of $n \in \mathbb{N}$ intervals $[x_i, x_{i+1}) \subseteq [0, \infty]$ and corresponding function values $\omega(x \in [x_i, x_{i+1})) = \omega_i$, called weights, where $\omega_i \in [0, \infty)$ for $i = 1, 2, \ldots, n$. In their original contribution, Winkelmann and Di Napoli obtained weight functions for the SLiSe framework by following three guiding principles, derived from experience: (i) gradual decrease in weights outside the search interval $[-1, 1]$, (ii) sufficient magnitude of weights inside $[-1, 1]$, and (iii) symmetry in weights about the interval endpoints of $[-1, 1]$. While SLiSe filters following these guidelines could outperform state-of-the-art Gauss–Legendre and Zolotarev filters, some manual adjustment based on experience remained necessary.

An example of a weight function $\omega_{\gamma\text{-SLiSe}}$, which yields SLiSe filters outperforming Gauss–Legendre filters, is given in Table 1. This choice of weights suggests a natural way of parameterizing weights and interval boundaries so they can be treated without distinction. For this purpose, we introduce a set V_s of parameter vectors $v = (v_1, \ldots, v_{2s-3})$, where $s \geq 2$ equals the number of intervals to the right of 0:

$$V_s = \{\, v \in [0, \infty)^{2s-3} \mid G < v_1 < 1 < v_2 < G^{-1} < v_3 < \cdots < v_{s-1} \,\} \tag{4.1}$$

for some gap parameter $G \in (0, 1)$.

A generic set of $v_i \in V_s$ induces weight function values ω_j with $j = 1, 2, \ldots, s$. Following this parameterization, Table 1 is rewritten as Table 2. The parameters v_1, v_2 enclose 1 but do not necessarily match the endpoints of the gap $[G, G^{-1}]$. The parameters v_3, v_4 reflect some more intervals of the weight function. The remaining parameters $v_5, \ldots, v_7$ denote nonnegative weights. The weight for the interval $[0, v_1)$ is fixed to 1, as weight functions are invariant under scaling within SLiSe. In this notation, the $\omega_{\gamma\text{-SLiSe}}$ weight function from Table 1 translates into the vector $(0.95, 1.05, 1.4, 5, 0.01, 10, 20) \in V_5$. As we are going to illustrate in the next section, this parameterization scheme allows for a systematic improvement of weight functions, alongside a choice of weights and interval endings.

4.2. Minimization of parameterized weight functions. In order to compare the influence of distinct weight function values $\omega_j \in V_s$ on the minimization of WCR, we introduce a new objective function:

Table 1
The $\omega_{\gamma\text{-SLiSe}}$ weight function.

$\lvert x\rvert \in$	$[0, 0.95)$	$[0.95, 1.05)$	$[1.05, 1.4)$	$[1.4, 5)$	$[5, \infty)$
$\omega_{\gamma\text{-SLiSe}}(x)$	1	0.01	10	20	0

Table 2
Parameterized weight function ω for $s = 5$.

$\lvert x\rvert \in$	$[0, v_1)$	$[v_1, v_2)$	$[v_2, v_3)$	$[v_3, v_4)$	$[v_4, \infty)$
$\omega_j(x)$	1	v_5	v_6	v_7	0

Algorithm 4.1 (local WCR minimization).

procedure ReduceFilterWCR(v, β, z, G)
  $G \leftarrow \sqrt{G}$ ▷ For shifting of filter
  $n \leftarrow$ Length(v)
  $w \leftarrow h_{\beta,z}(v)$
  while Res(h) $> 10^{-9}$ **do**
    for $i = 1, 2, \ldots, n$ **do** ▷ Coordinate descent
      $\hat{v} \leftarrow$ AdaptiveDifferentialEvolution($h_{\beta,z}(v(v_i))$)
      if $h_{\beta,z}(\hat{v}) < w$ **then**
        $v \leftarrow \hat{v}$, and $w \leftarrow h_{\beta,z}(\hat{v})$
      end if
    end for
    $v' \leftarrow$ Nelder–Mead($h_{\beta,z}(v(v_3, \ldots, v_n))$)
    $v' \leftarrow (|v'_1|, \ldots, |v'_n|) \in V_s$
    $r_{\beta',z'} \leftarrow$ SLiSe(β, z, ω') ▷ See Algorithm 3.1
    $w' \leftarrow h_{\beta',z'}(v')$ ▷ New WCR value
    Res(h) $\leftarrow |w - w'| \,/\, w$ ▷ Compute WCR residual
    $w \leftarrow w', v \leftarrow v', \quad \beta \leftarrow \beta', \quad z \leftarrow z'$
  end while
  $(\beta, z) \leftarrow (G^{-1}\,\beta, G^{-1}\,z)$ ▷ Shifting coefficients and poles
  return $r_{\beta,z}$
end procedure

$$h := h_{\beta,z}(v) := w_G(r_{\beta,z}(v)) \quad \text{for } v \in V_s, \tag{4.2}$$

where $r_{\beta,z}(v)$ is computed by the Algorithm 3.1. $h_{\beta,z}(v)$ is a functional of a given filter $r_{\beta,z}$, and it associates the vector of parameters v with the WCR of the corresponding filter. As such, h establishes a meaningful metric to quantify the performance of a weight function ω. The minimization

$$\operatorname*{argmin}_{v \in V_s} h_{\beta,z}(v) \tag{4.3}$$

facilitates a systematic search for better weight functions and, consequently, rational filters with smaller worst-case convergence. For a given $r_{\beta,z}$, this is a nonlinear, derivative-free minimization problem, depending on only $(2s-3)$ variables. However, for each optimization step involving changes of v, the rational filter $r_{\beta,z}(v)$ has to be computed again by executing a call to the Algorithm 3.1. The end result is a nested optimization problem (2.8) with, possibly, thousands of calls to Algorithm 3.1.

Since the WCR functional cannot be expressed as a continuous function of v, to solve (4.3), we resort to using the Nelder–Mead algorithm, a prominent local, derivative-free minimization scheme.[3] In our case, Nelder–Mead generates competitive solutions quickly but suffers from stagnation at nonoptimal points [24, 20]. To overcome stagnation, we follow Carl Kelley's suggestion of restarting Nelder–Mead at the current iterate with adjusted parameters [16]. An explicit such parameter choice exists only for the case of smooth functions, introduced as *oriented restart*.

[3]Local means that the algorithm starts at an existing point, at best, close to the sought after minimum.

Since our functional h is nonsmooth, we obtain a new parameter choice by perturbing the current iterate carefully through *coordinate descent*. Coordinate descent is a simple, local, derivative-free minimization method that performs subsequent line searches along the coordinate directions, given some starting point $v \in V_s$. Independently of having detected stagnation, we use coordinate descent systematically to obtain a new starting point for each NELDER–MEAD call.

The general scheme outlined above is described in Algorithm 4.1 and implemented using the Julia programming language.[4] Given a weight function $v \in V_s$ and a filter $r_{\beta,z}$, the algorithm chooses better weight functions from V_s iteratively. At each iteration of the **while** loop, the weights of the current filter are updated to reduce the WCR, and from these, a new filter is computed. When the residual of the h function $\mathrm{Res}(h)$ (i.e., the relative difference of two subsequent h values) falls below an established threshold tolerance (see `line 5` of Algorithm 4.1), the algorithm returns the SLiSe filter of the last iteration, which minimizes the WCR among the weight functions in the search space V_s. Each **while** loop iteration follows three consecutive steps: (i) coordinate descent, (ii) NELDER–MEAD, (iii) computation of new SLiSe filter and convergence check.

Coordinate descent. This is performed in `lines 6–11`, improving the coordinates of the parameter vector $v \in V_s$ through a separate minimization problem for each variable v_i and $i \leq 2s-3$,

$$\text{(4.4a)} \qquad \operatorname*{argmin}_{c \in I_i} h_{\beta,z}(v(c)), \quad \text{where } v(c) := (v_1, \dots v_{i-1}, c, v_{i+1} \dots, v_{2s-3}) \in V_s,$$

while restricting the search space to a neighborhood I_i of v_i. For instance, for $s \geq 5$ the intervals I_i used are

$$\text{(4.4b)} \qquad I_i := \begin{cases} [G, \quad 1] & \text{if } i = 1, \\ [1, \quad G^{-1}] & \text{if } i = 2 \\ [G^{-1}, \, v_{i+1}] & \text{if } i = 3, \\ [v_{i-1}, \, v_{i+1}] & \text{if } 4 \leq i < s-1, \\ [\, v_{i-1}, 3\, v_i] & \text{if } i = s-1, \\ [0.1\, v_i, 10\, v_i] & \text{if } s \leq i \leq 2s-3. \end{cases}$$

We implement the coordinate descent minimization through the global, derivative-free minimization scheme ADAPTIVEDIFFERENTIALEVOLUTION from the Julia library BlackBoxOptim.jl [6]. Global minimization algorithms aim to find the global minimizer within a region I_i instead of converging to a local minimizer starting from a given point. If the value of WCR has decreased, the solution of the coordinate descent minimization $\hat{v}_i$ is used instead of v_i for the successive steps of the **while** loop iteration. This step is executed to prevent stagnation in the execution of the NELDER–MEAD minimization.

Nelder–Mead. In `line 12`, NELDER–MEAD is applied to a slightly modified version of the minimization problem formulated in (4.3): We keep v_1, v_2 fixed at the values obtained from the coordinate descent step. This choice is motivated by the high sensitivity of the functional h to changes in these variables, being close to the endpoints of the gap $[G, G^{-1}]$. In practice, NELDER–MEAD is used to solve the minimization problem

$$\text{(4.5a)} \qquad \operatorname*{argmin}_{b_1,\dots,b_r \in \mathbb{R}} h_{\beta,z}(v(b_1, \dots, b_r)),$$

[4]The code is freely available at https://github.com/SimLabQuantumMaterials/SLiSeFilters.jl.

where

$$v(b_1, \ldots, b_r) := (v_1, v_2, b_1, \ldots, b_r) \quad \text{and} \quad r = 2s - 5. \tag{4.5b}$$

Because the classical Nelder–Mead algorithm performs unconstrained minimization only, we also have to ensure that the minimizer v' lies within the admissible intervals defined by V_s. A straightforward approach would be to use a modified objective function within Nelder–Mead, defined as

$$\hat{h}(v) := \begin{cases} h(v) & \text{if } v \in V_s, \\ \infty & \text{otherwise} \end{cases} \tag{4.6}$$

to penalize invalid weight functions $v \notin V_s$. We verified that this approach works, but our experiments have shown that it slows down the convergence to the minimum. Based on our tests, we observed that, in practice, most of the violations $v \notin V_s$ stem from the selection of slightly negative weights by Nelder–Mead. This is likely caused by rounding errors, which may lead to a slight decrease of the WCR value for very small but negative ω_j. Every other violation of constraints seems to cause the opposite of a reduction in WCR value and is thus not chosen by Nelder–Mead. To overcome this problem, we adopted a very simple solution: We map the resulting minimizer of a Nelder–Mead into V_s explicitly by taking the absolute values $(|v_1^{(k+1)}|, \ldots, |v_{2s-3}^{(k+1)}|)$ instead of a possible negative (invalid) iterate $v^{(k+1)}$ (see `line 13`). While this approach only avoids violations of constraints caused by the choice of negative weights, we did not experience other violations of constraints in the minimizers returned by Nelder–Mead. For the sake of completeness, we have experimented with other derivative-free minimization algorithms (those from the minimization algorithm collection NLOpt [15], including a constrained version of Nelder–Mead), none of which led to a more competitive reduction of WCR value in the same standard setups.

SLiSe filter and convergence. As already mentioned, the SLiSe procedure is called multiple times within both the Nelder–Mead and the AdaptiveDifferentialEvolution procedures. The last call of SLiSe is executed so as to calculate the residual of the WCR functional and check for convergence. If convergence is reached, the algorithm returns values for the WCR, the poles z, and the coefficients β. The latter are rescaled, as explained in the following subsection 4.3, by introducing a linear scaling transformation to improve the behavior of the resulting filters at the interval $[-1, 1]$ boundaries.

4.3. Scaling the filter. Compared to previous filters such as Gauss–Legendre or Zolotarev, our new SLiSe filters obtained from the minimization problem (4.3) reduce the WCR by up to multiple orders of magnitude. So far, we assumed to know the appropriate gap parameter G in advance and minimized our filters accordingly. This assumption may be too optimistic for some eigenproblems: For a given G, we assume that no eigenvalues lie within the interval $[-G^{-1}, -G] \cup [G, G^{-1}]$. Since, in practice, this assumption may be violated, a rational filter with small function value within $[G, 1]$ may lead a slower convergence rate than the WCR or no convergence at all. An illustration of this problem is given in Figure 4, and its caption. Eigenvalues within $[1, G^{-1}]$ are less problematic because they are not sought after by the eigensolver (for an in-depth discussion, see section 5.3). While maintaining a competitive WCR value, the issue described above can be overcome by solving the slightly modified minimization problem

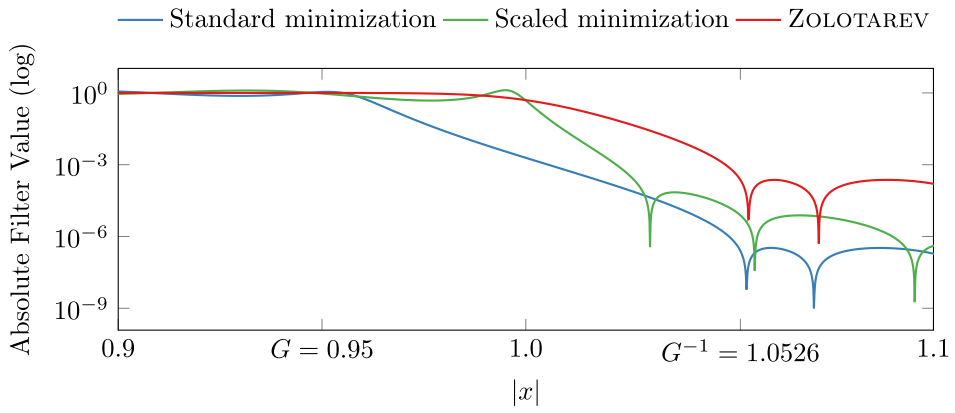


FIG. 4. *Logarithmic plot of* 16*-pole rational filters for gap parameter* $G = 0.95$*. The standard filter is a solution of the original minimization problem as of* (4.3)*, while the scaled filter solves the new minimization problem in* (4.7)*. Even though the scaling causes the WCR to increase from* 4.95e−6 *to* 1.04e−5 *, this is still below the WCR of the* Zolotarev *filter (*2.32e−4*). On the other hand, near* 1*, the standard filter attains a function value of about* 10^{-3}*. Hence, an eigenvalue inside the gap* $[G, 1]$ *could roughly halve the actual convergence rate for the standard filter. The other two filters are not affected by this problem.*

$$\operatorname*{argmin}_{\beta\in\mathbb{C}^m, z\in(\mathbb{C}\setminus\mathbb{R})^m} \frac{\max_{x\in[-\infty,-G^{-1}]\cup[G^{-1},\infty]} |r_{\beta,z}(x)|}{\min_{x\in[-1,1]} |r_{\beta,z}(x)|} \tag{4.7}$$

instead of minimizing w_G as in (2.7). For any solution (β, z) of this modified problem, the rational filter $r_{\beta,z}$ offers a larger function value inside the entire search interval $[-1, 1]$ than outside in $[-\infty, -G^{-1}] \cup [G^{-1}, \infty]$. When our filters are used in iterative eigensolvers based on spectral projection, this behavior ensures reliable and fast convergence.

Instead of modifying the WCR minimization procedure, we solve for the modified WCR in (4.7) by introducing a linear scaling transformation $u(x) := \sqrt{G}\, x$. We have

$$\operatorname*{argmin}_{\beta\in\mathbb{C}^m, z\in(\mathbb{H}^+R)^m} w_{\sqrt{G}}(r_{\beta,z}) = \operatorname*{argmin}_{\beta\in\mathbb{C}^m, z\in(\mathbb{H}^+R)^m} \frac{\max_{x\in[-\infty,-G^{-1}]\cup[G^{-1},\infty]} |r_{\beta,z}(u(x))|}{\min_{x\in[-1,1]} |r_{\beta,z}(u(x))|}.$$

Since the function composition $r_{\beta,z} \circ u$ is a rational filter itself, if $r_{\beta,z}$ solves (4.3), then $r_{\beta,z} \circ u$ solves (4.7). Additionally, it follows from the definition of rational filters in (1.2) that $r_{\beta,z} \circ u = r_{\sqrt{G^{-1}}\beta, \sqrt{G^{-1}}z}$ which characterizes the parameters of the resulting filter. This scaling of rational filters through linear transformation is incorporated in Algorithm 4.1 in `lines 2` and `19`.

In our implementation of Algorithm 4.1, we used as initial parameters the weight functions $\omega_{\gamma\text{-SLiSe}}$ used in [40] opportunely rescaled. For instance, for $s = 5$, we selected

$$v^{(0)} = (\sqrt{G}, \sqrt{G^{-1}}, 1.4, 5, .01, 10, 20)$$

as the initial parameters characterizing the weight function, for some gap parameter $G \in (0, 1)$. Accordingly, we chose both Zolotarev and Gauss–Legendre filters as initial conditions for WCR minimization. We obtained a number of new spectral filters, to which we refer as WiSe. In the following section, we provide a number of experimental tests illustrating the performance of these WiSe filters.

5. Experiments. In this section, we compare the Zolotarev and generalized Gauss–Legendre filters to the WiSe filters from the previous section in two different scenarios. First, in order to inspect the worst-case performance, we compute the WCR values for different gap parameters G and poles per quadrant m. Second, we compare WiSe filters computed using box-constrained minimization for different lower bounds; these filters play an important role when iterative linear system solves are used to compute the spectral resolvent. Third, we use the filters in the FEAST package and assess their performance on two eigenproblems used in past literature [32, 40, 11].

We provide the Julia library SLiSeFilters.jl to obtain WiSe filters and the generalized Gauss–Legendre filters. For Zolotarev filters, we use the RKToolbox, co-developed by Stefan Güttel. We will not consider other prominent examples of rational filters, notably trapezoid or former SLiSe filters, as they are not as competitive. Trapezoid filters offer a strictly monotonous decay in function value for $|x| > 1$, which is not as sharp as for Gauss–Legendre and Zolotarev filters, leading to significantly larger WCR values [11]. As for SLiSe filters, Figure 1 provides clear evidence that WiSe filters are superior when it comes to number of iterations to convergence.

5.1. Comparison of WCR values. In Table 3, we list WCR values for Gauss–Legendre, Zolotarev, and WiSe filters for different gap parameters G and poles per quadrant m. As m increases, Zolotarev filters feature a reliable, gradual decrease in WCR; by construction, they do not offer a decay in function value as $|x| \to \infty$, unlike Gauss–Legendre and WiSe. Hence, Gauss–Legendre and WiSe filters lead to quicker convergence for some FEAST instances, even if their WCR is larger. Gauss–Legendre filters show the largest WCRs and are not competitive with regards to worst-case performance. WiSe exhibits a significant reduction of WCR compared to previous filters, especially for the default choice of $m = 4$ within the FEAST eigensolver. Yet, the improvement over Zolotarev filters diminishes as m increases, because the dimension of the underlying minimization increases with m. Large numbers of poles per quadrant $m > 7$ correspond to high-degree rational functions and are not taken into consideration. It is important to notice that even when one has to set G to 0.9998, in the presence of eigenvalues clusters near the interval

Table 3

WCR values for different filters, gap parameters, and numbers of poles (smaller is better; row-wise minimum in bold).

Gap G	Poles per Quadrant m	Gauss–Legendre	Zolotarev	WiSe
0.95	3	$2.33e{-2}$	$2.24e{-3}$	$\mathbf{2.45e{-4}}$
	4	$1.37e{-3}$	$2.32e{-4}$	$\mathbf{1.04e{-5}}$
	5	$5.58e{-4}$	$2.41e{-5}$	$\mathbf{9.82e{-7}}$
	6	$4.80e{-5}$	$2.50e{-6}$	$\mathbf{7.53e{-7}}$
	7	$7.36e{-6}$	$\mathbf{2.59e{-7}}$	$2.73e{-7}$
0.98	3	$3.45e{-2}$	$7.46e{-3}$	$\mathbf{1.30e{-3}}$
	4	$2.43e{-2}$	$1.15e{-3}$	$\mathbf{1.63e{-4}}$
	5	$1.94e{-3}$	$1.77e{-4}$	$\mathbf{1.47e{-5}}$
	6	$1.16e{-3}$	$2.74e{-5}$	$\mathbf{1.56e{-5}}$
	7	$1.01e{-4}$	$\mathbf{4.24e{-6}}$	$9.52e{-6}$
0.998	3	$6.64e{-1}$	$4.23e{-2}$	$\mathbf{2.74e{-2}}$
	4	$4.69e{-1}$	$1.12e{-2}$	$\mathbf{3.53e{-3}}$
	5	$8.29e{-2}$	$3.04e{-3}$	$\mathbf{7.33e{-4}}$
	6	$4.45e{-2}$	$8.27e{-4}$	$\mathbf{9.81e{-5}}$
	7	$3.15e{-2}$	$2.26e{-4}$	$\mathbf{8.94e{-6}}$
0.9998	3	$9.51e{-1}$	$\mathbf{1.11e{-1}}$	$1.44e{-1}$
	4	$8.97e{-1}$	$\mathbf{3.85e{-2}}$	$4.16e{-2}$
	5	$8.22e{-1}$	$1.39e{-2}$	$\mathbf{1.04e{-2}}$
	6	$7.35e{-1}$	$5.09e{-3}$	$\mathbf{1.85e{-3}}$
	7	$6.41e{-1}$	$1.87e{-3}$	$\mathbf{5.19e{-4}}$

boundaries, WiSe still manages to produce satisfactory WCR values at the cost of higher m.

The numbers in bold in Table 3 show that, for almost all pairs (G, m), WiSe filters have the lowest WCR value. Based on Theorem 2.1, the best worst-case performance is offered by the filter with smallest WCR. As we will see in the following sections, this claim is confirmed by our numerical results.

5.2. Influencing the condition number of the resolvent matrices. In this section we present numerical evidence of how WiSe filters obtained through box-constrained minimization can improve the condition number of the resolvent matrices $(A - z_i I)$. As already mentioned in section 3.2, a large condition number for such matrices can dramatically deteriorate convergence speed when iterative methods are used to solve for (2.3). We have also shown how the condition number of a resolvent matrix is bounded from above, up to a constant factor, by the imaginary part of the poles z_i of the rational filter $r_{\beta,z}$. Because of this observation, we define the *worst-case condition number* as

$$C(r) := \max_{z_i \in \mathbb{H}^{+R}} \frac{1}{|\mathrm{Im}(z_i)|} \tag{5.1}$$

and use it to estimate the worst possible condition number of the resolvent matrices. Strictly speaking, $C(r)$ does not bound from above every $\kappa(A - z_i I)$, but it has the advantage that it does not depend on the specific matrix A and still provides a useful picture of how the condition number of the resolvent matrices is influenced by the placement of the filter's poles. Imposing a lower bound $\texttt{lb} \geq 0$ on the imaginary parts of the poles z_i, as we have done when using the L-BFGS-B algorithm to compute the WiSe filters (see section 3.2), limits the value of $C(r)$.

In our numerical test, we have computed $C(r)$ and WCR for two different box-constrained WiSe filters and several different values of `lb`. The initial values of β, z, and G in the minimization process for the two WiSe filters come from the coefficients and poles of the standard 16-pole Gauss–Legendre with $G = 0.95$ and Zolotarev with $G = 0.998$ filters, respectively. With these choices, the standard Gauss–Legendre filter has approximately $w_{0.95}(r) = 2.42 \cdot 10^{-2}$, $C(r) = 1.60 \cdot 10^{1}$, and $\min_i(\mathrm{Im}(z_i)) = 0.062$. Likewise, the standard Zolotarev filter has $w_{0.998}(r) = 1.12 \cdot 10^{-2}$, $C(r) = 4.55 \cdot 10^{2}$, and $\min_i(\mathrm{Im}(z_i)) = 0.0022$. The minimum values for the imaginary part of the poles for the corresponding unconstrained WiSe filters are $\min_i(\mathrm{Im}(z_i)) = 0.0046$ and $\min_i(\mathrm{Im}(z_i)) = 0.0007$, respectively.

In Figure 5, we plot the worst-case condition number $C(r)$ and the w_G for both box-constrained WiSe filters with respect to a range of lower bounds `lb` defined by the values of $\min_i(\mathrm{Im}(z_i))$ for the unconstrained WiSe and its corresponding standard filter (either Gauss–Legendre or Zolotarev). Since we observed that the minimization process shows some fluctuations in returning the w_G value for the box-constrained WiSe filter, we have repeated all minimizations 20 times for each different `lb` value. These fluctuations are likely the result of the nonlinear, randomized minimization process, which is highly sensitive to the selected parameters. In practice, these fluctuations of the WCR values do not represent a problem because the corresponding $C(r)$ values remain quite stable. Moreover, irrespective of the lower bound `lb`, all observed WCR values of the box-constrained WiSe filters lie below those of the corresponding Zolotarev and Gauss–Legendre filters for all 20 repetitions.

On the other hand, the $C(r)$ plots show a clear decrease of the worst-case condition number up to almost an order of magnitude as the value of `lb` increases. From the

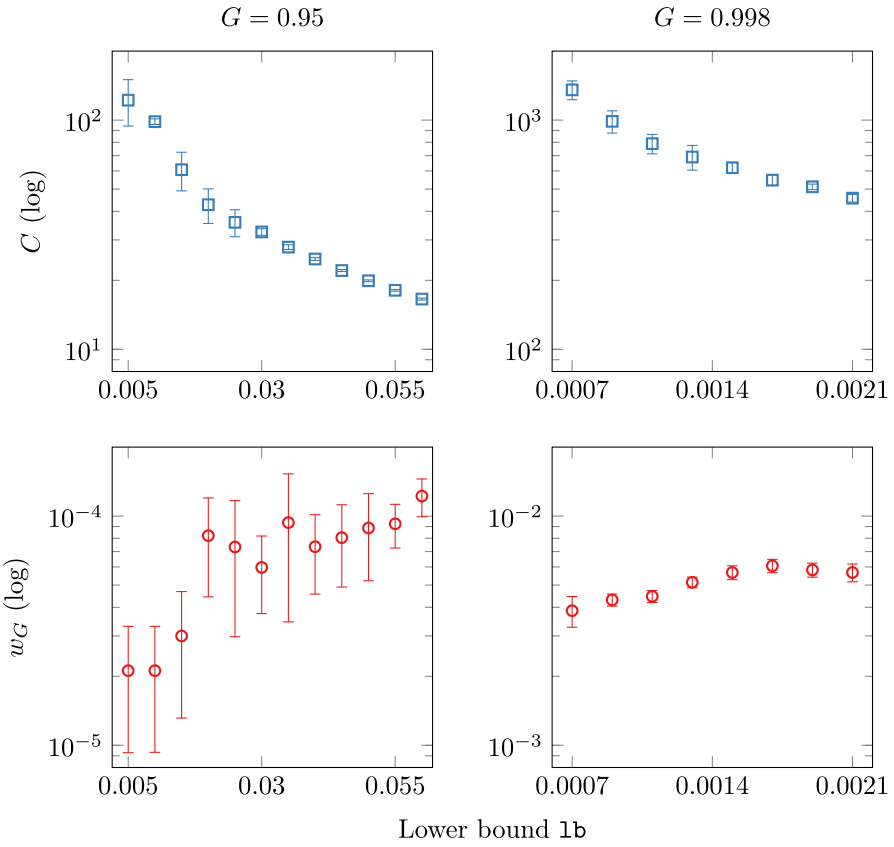


FIG. 5. *The effect of different lower bounds* `lb` *in WiSe on the worst-case condition number* C *and the WCR* w_G *of the resulting filters. Since WiSe is a randomized, nonlinear minimization method, the means and standard deviations from* 20 *repetitions are reported.*

WCR plots one can also observe that larger lower bounds seem to show less fluctuation, which indicates a reduced dependence on the minimization parameters. In turn, if the chosen lower bound `lb` for the L-BFGS-B minimization is closer to the $\min_i(\mathrm{Im}(z_i))$ of the starting filter, the nonlinear minimization process might converge more quickly. Overall, these numerical experiments confirm how WiSe filters outperform existing ones even in the case of box-constraints optimization and can be efficiently used in those cases where iterative methods, such as GMRES or CG, have to be used to solve for the linear systems corresponding to large and sparse resolvent matrices.

5.3. Experiments with FEAST. For our numerical experiments, we used FEAST in version 3.0, compiled with the Intel Compiler 17.0.0, and run on an Intel Core i7-6900K. We selected the default FEAST parameters with the exception of disabling[5] the repeated factorization of the underlying linear systems. This substantially reduces runtime but requires sufficient RAM. There are two reasons why we chose to use a direct method for the solution of the linear systems: first, because of its higher accuracy when compared to iterative methods, and second to have much more reliable and reproducible time-to-solution independently of the rational filter used.

[5]This behavior can be achieved through the FEAST parameter `fpm(10) = 1`. For details, consult the FEAST documentation [5].

A required argument of FEAST is an upper bound M_0 on the number of eigenvalues M in the search interval; M_0 indicates the size of the reduced eigenvalue problem that is solved in every FEAST iteration. As already mentioned in section 2, the convergence rate of FEAST substantially depends on this subspace size M_0. Let us denote with $I \supset [-1, 1]$ the interval centered around 0 that contains $M_0 > M$ eigenvalues; then, FEAST's convergence rate is proportional to

$$\frac{|r(\lambda_{\text{out}})|}{|r(\lambda_{\text{in}})|}, \tag{5.2}$$

where $|r(\lambda_{\text{in}})| = \min_{\lambda \in [-1,1] \cap \sigma(A)} |r(\lambda)|$ and $|r(\lambda_{\text{out}})| = \max_{\lambda \notin I \cap \sigma(A)} |r(\lambda)|$. A smaller subspace size M_0 yields faster FEAST iterations but decreases the convergence rate and thus increases the number of iterations. As a compromise, the original FEAST publication [32] suggested a subspace size of $M_0 = \lceil C \times M \rceil$ for $C = 1.5$. This provides reliable convergence within FEAST but not necessarily fastest convergence, as we see in the following. Because only estimates of the actual eigencounts are available in advance, we assess different scenarios by studying a number of eigencount multipliers C.

It is important to notice that in all our comparison we change only the rational filter and maintain all other parts of FEAST unchanged. In particular we let FEAST use the same default linear system solves to tackle (2.3) and direct eigensolver in the Rayleigh–Ritz step. Since the level of accuracy of FEAST is for all practical purposes determined by these two tasks [18, 19], we consider the accuracy of the determined eigenpairs across distinct filters comparable. In other words, since FEAST reaches convergence with all filters using the same procedures, the solutions are considered to be equally accurate.

5.3.1. Experiment I. In Figure 6, we compare the convergence of FEAST for different eigencount multipliers $C > 1$ on a specific interval of the carbon nanotube (CNT) eigenproblem. The matrices, corresponding to this generalized eigenproblem, represent the discretized Hamiltonian and overlap operators of a physical system studied in the context of a specific density functional theory method and have been

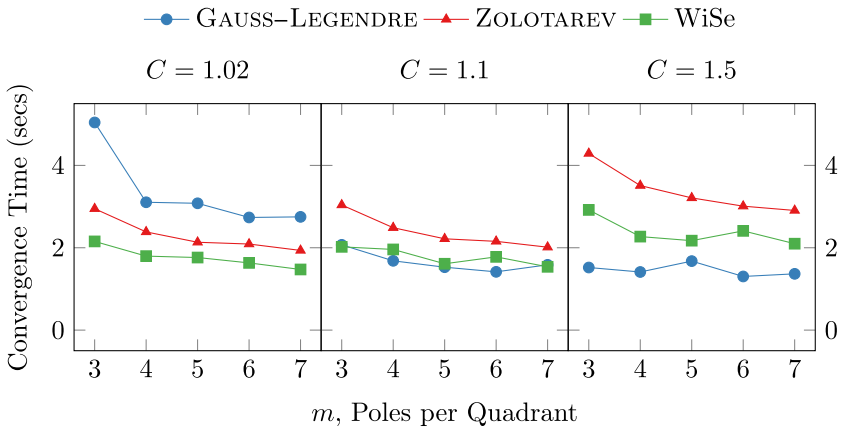


FIG. 6. *Time required to solve the CNT eigenproblem* [32, 11] *through FEAST for different eigencount multipliers C, numbers of poles m, and filters. Averages of* 10 *executions are reported. All the three filters are computed for a value of $G = 0.998$ and lead to eigenpairs with the same level of accuracy.*

used to analyze the worst-case performance of FEAST in previous publications [32, 11]. These sparse CNT matrices $A, B \in \mathbb{R}^{12450\times 12450}$, with 86 808 nonzero entries, define the interior eigenproblem $Ax = \lambda Bx$, where one is interested in obtaining the $M = 100$ eigenvalues in the interval $[-65.0, 4.96]$. The figure is divided into three quadrants, each corresponding to one of three increasing eigencount multipliers $C = 1.02, 1.1, 1.5$. On the y-axis of each quadrant, the convergence time (in seconds) of the FEAST eigensolver, equipped with three distinct filters, is plotted against increasing numbers of poles per quadrant m for each of these filters. The value of the parameter G is kept fixed for all filters across the entire figure.

The result of this experiment demonstrates that best worst-case convergence of the FEAST eigensolver correlates strongly with the WCR value of the filter that is used to project onto the active subspace. On the other hand, the size of the active subspace M_0 also contributes to the convergence time and can be a confounding factor in interpreting the numerical results. In order to minimize the influence of the latter on our interpretation of WCR, we consider first the quadrant for $C = 1.02$. From Table 3, we expect that best convergence time is achieved by FEAST when equipped with WiSe filters, followed by Zolotarev and Gauss–Legendre filters, respectively. This is indeed the case; the performance of the three filters is clearly separated by a gap, which reflects the differences in WCR between the filters for all m.

The influence of m on filter performance is less pronounced. This is not surprising. For instance, the Zolotarev filter enables FEAST to converge with a very slow decrease of convergence rate as the number of poles increases, for all considered C. The only consequence of increasing C is a growth in convergence time; the size of the subspace increases with C, and more operations with vectors must be performed by the linear system solver while the rate of convergence remains practically constant. This behavior is due to the *equioscillation* of Zolotarev filters: These filters are optimal in approximating the ideal filter in ∞-norm but do not decay away from the filtered interval. WiSe filters behave similarly: While they decay (moderately) away from the interval, their effectiveness is not determined by their value away from it, but rather their behavior very close to its boundary (see Figure 2). This is reflected by the very slight decrease in time-to-convergence as m increases, although the corresponding WCR value decreases substantially as shown in Table 3.

As the size of the active subspace increases, its influence on convergence time becomes ever more pronounced. This is because the difference between the convergence rate of FEAST in (5.2) may become increasingly larger than the WCR of the used filter as $C \gg 1.02$ [11]. In other words, larger active subspaces dilute the correlation between WCR of the filter and convergence rate of the eigensolver. This is clearly visible if one traverses the quadrants in Figure 6 from left to right and is best illustrated by the Gauss–Legendre filters. These suffer from slow convergence for small C but can compensate for their large WCR for large subspace sizes because they decay rapidly in function value away from the search interval. In other words, for Gauss–Legendre, the size of the active subspace is much more relevant than the WCR of the filter.

This simple analysis, based on a very specific interior eigenvalue problem, seems to suggest that WiSe filters should always be preferred over Zolotarev filters in those cases in which the WiSe WCR is smaller than Zolotarev (compare Table 3). When comparing the performance of WiSe and Gauss–Legendre filters, it seems that the size of the active subspace plays a major role in identifying the point at which one filter outperforms the other. In order to address this question, we examine the

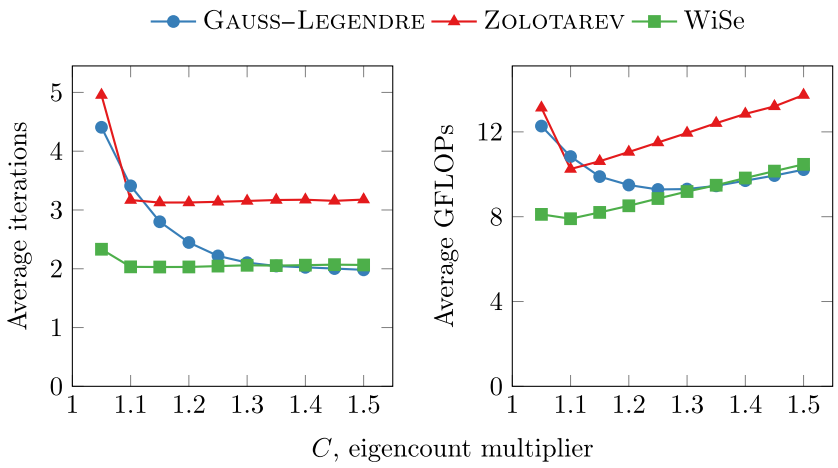


FIG. 7. *The average numbers of iterations and FLOPs (floating-point operations), required by FEAST to solve* 2117 *benchmark problems* [40]*, for different subspace sizes (by multiplying the actual eigencounts with a fixed scalar C) and filters with $G = 0.95$.*

convergence rate of FEAST in the next section, in terms of both number of FLOPs and number of subspace iterations, using a large set of representative eigenproblems.

5.3.2. Experiment II. We consider a moderate gap parameter of $G = 0.95$ and a set of 2117 interior eigenproblems. These eigenproblems were obtained from Si2, a sparse and symmetric matrix from the University of Florida Sparse Matrix Collection [3], by selecting 2117 different search intervals $[a, b]$[6] as described in [40, Appendix B]. Each search interval uniquely identifies an interior eigenproblem with its unique eigenvalue distribution and eigenvalue count. As such, it is quite general and statistically relevant, since it reflects the large variations that are possible in distributing and clustering eigenvalues inside, outside, and in the vicinity of the search interval ends.

We initially solved for each of the 2117 benchmark problems with a fixed value of $m = 4$ poles per quadrant, the default in the FEAST eigensolver. As in the previous section, we repeated this test for all three filters for increasing values of the eigencount multiplier C. The results of these tests are graphically reported in Figure 7, which plots the number of subspace iterations and the total number of FLOPs performed by FEAST. These results confirm the analysis of section 5.3.1. The Zolotarev and WiSe filters maintain a linear behavior as a function of increasing dimension of the active subspace as soon as $C \geq 1.1$. In other words, the WCR of these filters influences the convergence of the eigensolver only for active subspaces that closely match the true number of eigenvalues in the interval $[a, b]$. As soon as the size of the active subspace gets larger, the convergence of the eigensolver is dictated by (5.2). This interpretation is made even clearer when one looks at the linear increase in FLOPs for FEAST, equipped with these two filters: While the average number of iterations remains constant, the total number of FLOPs increases due to the linear increase in the total number of right-hand-side vectors Y for which (2.3) must be solved.

[6]The code to obtain such benchmark sets from arbitrary matrices is freely available at https://github.com/SimLabQuantumMaterials/SpectrumSlicingTestSuite.jl.

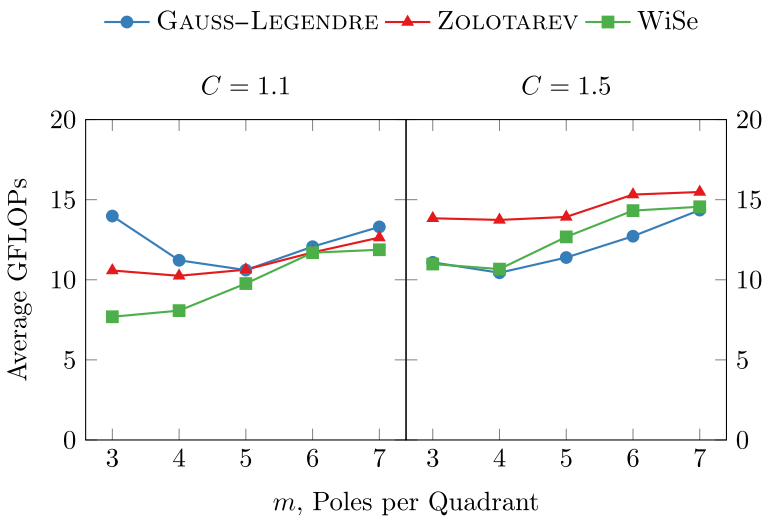


FIG. 8. *Comparison of average FLOPs, required to solve* 2117 *benchmark problems* [40]*, using* GAUSS–LEGENDRE*,* ZOLOTAREV*, and WiSe filters for different of poles numbers m per quadrant, while $C \in \{1.1, 1.5\}$ and $G = 0.95$.*

As C grows, the rate of convergence of FEAST, equipped with the GAUSS–LEGENDRE filter, equals the one of WiSe—which is dictated by the WCR at smaller values of C. For subspace multipliers $C > 1.3$, the GAUSS–LEGENDRE starts competing, on average, with the WiSe. It must be noted that this comes at a cost: Using GAUSS–LEGENDRE for a relatively large active subspace, such as the default value of $C = 1.5$ suggested by FEAST, has on average a higher FLOP count than the WiSe filters for eigenvalue counts $C < 1.3$. This observation is fairly independent of the number of poles per quadrant used, as shown in Figure 8. GAUSS–LEGENDRE filters have a slight advantage with respect to FLOP count for large subspace sizes ($C = 1.5$), but they behave worse than the WiSe for any number of poles and small subspace sizes ($C = 1.1$). Due to the decay in value of the filter function, the WiSe filter even outperforms ZOLOTAREV for $m = 7$, despite its larger WCR. For larger m, overall FLOP count increases, while the difference in FLOP count across the filters shrinks.

In conclusion, FEAST, equipped with WiSe filters, offers a competitive advantage over the use of GAUSS–LEGENDRE and ZOLOTAREV filters. WiSe filters are quite stable with respect to the convergence rate of the eigensolver, irrespective of the active subspace or the degree of the filter function. Their use seems to almost always minimize the total FLOP count required by FEAST to reach convergence. In addition, their effectiveness for small eigencount multipliers suggests that WiSe filters should be preferred in all those cases where it is necessary to contain the subspace size, either because the RAM is limited or because the underlying spectrum distribution is unknown.

6. Conclusions. In this work, we show how we decreased time-to-convergence of the SLiSe optimization framework by using in it the minimization algorithm L-BFGS-B. When computing a box-constrained SLiSe filter, only hundreds of function evaluations are needed instead of millions. We exploit the improved performance by introducing a second optimization process for the numerical minimization of the WCR

of the SLiSe rational filters. The byproduct of such a minimization is the elimination of the dependence of the filter on the weight functions used in the nonlinear least-squares functional. The new WiSe filters outperform Gauss–Legendre and Zolotarev filters in terms of execution time, number of subspace iterations, and FLOP count necessary to reach convergence by the eigensolver.

Increasing the performance of the optimization of rational filters and eliminating their dependence on a number of adjustable parameters has an additional indirect and important impact on the eigensolver using the rational filters as spectral projectors. This class of solvers lend themselves to multiple levels of parallelism: At the highest level each interval $[a, b]$ can be split into subintervals $[a_j, b_j]$, each of which trivially constitutes a separate eigenproblem; at a mid-level the spectral solver requires the solution of a linear system for each pole z_i; at the lowest level each linear system has to be solved for multiple right-hand sides. While such a general scheme makes this class of eigensolver attractive, it complicates substantially the problem of balancing the computational load. One of the main contributions to the uncertainty of a well-balanced computation is the ability of the spectral filter to determine the number of subspace iterations needed to converge the full subspace corresponding to each $[a_j, b_j]$.

Our WiSe filters overcome this uncertainty by (1) decoupling the rate of convergence from the size of the active search subspace and (2) drastically reducing the dependence on the number of poles which can be safely set to a standard value (e.g., $m = 4$ in FEAST). The net result is that the spectral filter has the same effectiveness for any subinterval selected: For a given linear system solver the number of iterations required to reach convergence is minimized and independent of the eigenvalues distribution. Load balancing is then achieved by choosing subintervals with approximately the same eigenvalue count. Since obtaining a good estimate for the eigenvalue count and the eigenvalue distribution is a solved problem [4, 23], the result presented in this paper eliminates the influence of the spectral filter on load balancing for all practical purposes. The remaining challenge is balancing the load when solving for distinct linear system with multiple right-hand sides. This is the focus of further ongoing work.

Acknowledgments. We thank Jan Winkelmann for having provided support to the first author in developing the bulk of the work that contributed to this paper and Sebastian Achilles for useful discussions.

REFERENCES

[1] B. Beckermann and A. Townsend, *On the singular values of matrices with displacement structure*, SIAM J. Matrix Anal. Appl., 38 (2017), pp. 1227–1248, https://doi.org/10.1137/16M1096426.

[2] R. Byrd, P. Lu, J. Nocedal, and C. Zhu, *A limited memory algorithm for bound constrained optimization*, SIAM J. Sci. Comput., 16 (1995), pp. 1190–1208, https://doi.org/10.1137/0916069.

[3] T. A. Davis and Y. Hu, *The University of Florida sparse matrix collection*, ACM Trans. Math. Software, 38 (2011), pp. 1:1–1:25, https://doi.org/10.1145/2049662.2049663.

[4] E. Di Napoli, E. Polizzi, and Y. Saad, *Efficient estimation of eigenvalue counts in an interval*, Numer. Linear Algebra Appl., 23 (2016), pp. 674–692, https://doi.org/10.1002/nla.2048.

[5] E. Polizzi, J. Kestyn, B. Gavin, B. Spring, S. Guettel, J. Brenneck, and P. tang, *FEAST Eigenvalue Solver*, version 3.0, http://www.feast-solver.org/.

[6] R. Feldt, *Black-Box Optimization for Julia*, version 30, https://github.com/robertfeldt/BlackBoxOptim.jl.

[7] Y. Futamura and T. Sakurai, *z-Pares: Parallel Eigenvalue Solver*, http://zpares.cs.tsukuba.ac.jp/.

[8] B. Gavin and E. Polizzi, *Krylov eigenvalue strategy using the FEAST algorithm with inexact system solves*, Numer. Linear Algebra Appl., 25 (2018), p. e2188, https://doi.org/10.1002/nla.2188.
[9] D. Göddeke, *Fast and Accurate Finite-Element Multigrid Solvers for PDE Simulations on GPU Clusters*, Ph.D. thesis, Technische Universität Dortmund, 2011.
[10] J. Gopalakrishnan, L. Grubišić, and J. Ovall, Spectral discretization errors in filtered subspace iteration, Math. Comp., 89 (2020), pp. 203–228.
[11] S. Güttel, E. Polizzi, P. Tang, and G. Viaud, *Zolotarev quadrature rules and load balancing for the FEAST eigensolver*, SIAM J. Sci. Comput., 37 (2015), pp. A2100–A2122, https://doi.org/10.1137/140980090.
[12] T. Ikegami and T. Sakurai, *Contour integral eigensolver for non-Hermitian systems: A Rayleigh-Ritz-type approach*, Taiwanese J. Math., 14 (2010), pp. 825–837.
[13] T. Ikegami, T. Sakurai, and U. Nagashima, *A filter diagonalization for generalized eigenvalue problems based on the Skaurai-Sugiuara projection method*, J. Comput. Appl. Math., 233 (2010), pp. 1927–1936.
[14] A. Imakura, L. Du, and T. Sakurai, *Error bounds of Rayleigh-Ritz type contour integral-based eigensolver for solving generalized eigenvalue problems*, Numer. Algorithms, 71 (2016), pp. 103–120, https://doi.org/10.1007/s11075-015-9987-4.
[15] S. G. Johnson, *The NLopt Nonlinear-Optimization Package*, http://ab-initio.mit.edu/nlopt.
[16] C. Kelley, *Detection and remediation of stagnation in the Nelder–Mead algorithm using a sufficient decrease condition*, SIAM Journal on Optimization, 10 (1999), pp. 43–55, https://doi.org/10.1137/S1052623497315203.
[17] J. Kestyn, V. Kalantzis, E. Polizzi, and Y. Saad, *Pfeast: A high performance sparse eigenvalue solver using distributed-memory linear solvers*, in SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2016, pp. 178–189.
[18] L. Krämer, E. Di Napoli, M. Galgon, B. Lang, and P. Bientinesi, *Dissecting the FEAST algorithm for generalized eigenproblems*, J. Comput. Appl. Math., 244 (2013), https://doi.org/10.1016/j.cam.2012.11.014.
[19] L. Krämer and B. Lang, *Concergence of integration-based methods for the solution of standard and generalized Hermitian eigenvalue problems*, Electron. Trans. Numer. Anal., 48 (2018), p. 183–201, https://doi.org/10.1553/etna_vol48s183.
[20] J. Lagarias, J. Reeds, M. Wright, and P. Wright, *Convergence properties of the Nelder–Mead simplex method in low dimensions*, SIAM J. Optim., 9 (1998), pp. 112–147, https://doi.org/10.1137/S1052623496303470.
[21] R. Li, Y. Xi, L. Erlandson, and Y. Saad, *The eigenvalues slicing library (EVSL): Algorithms, implementation, and software*, SIAM J. Sci. Comput., 41 (2019), C393–C415.
[22] Y. Li, G. Geng, and Q. Jiang, *A parallelized contour integral Rayleigh–Ritz method for computing critical eigenvalues of large-scale power systems*, IEEE Trans. Smart Grid, 9 (2018), pp. 3573–3581.
[23] L. Lin, Y. Saad, and C. Yang, *Approximating spetral densities of large matrices*, SIAM Rev., 58 (2016), pp. 34–65, https://doi.org/10.1137/130934283.
[24] K. McKinnon, *Convergence of the Nelder–Mead simplex method to a nonstationary point*, SIAM J. Optim., 9 (1998), pp. 148–158, https://doi.org/10.1137/S1052623496303482.
[25] J. L. Morales and J. Nocedal, *Remark on "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization"*, ACM Trans. Math. Software, 38 (2011), pp. 7:1–7:4, https://doi.org/10.1145/2049662.2049669.
[26] H. Murakami, *An Experiment of the Filter Diagonalization Method for the Banded Generalized Symmetric-Definite Eigenproblem*, Tech. Report 59 (2007-HPC-110), Tokyo Metropolitan University, 2007.
[27] H. Murakami, *A filter diagonalization method by the linear combination of resolvents*, IPSJ Trans. ACS, 49 (2008), pp. 66–87.
[28] H. Murakami, *The Filter Diagonalization Method for the Unsymmetric Matrix Eigenproblem*, Tech. Report 43 (2008-HPC-115), Tokyo Metropolitan University, 2008.
[29] H. Murakami, *Experiments of Filter Diagonalization Method for Real Symmetric Definite Generalized Eigenproblems by the use of Elliptic Filters*, Tech. Report 2010-HPC-125, Tokyo Metropolitan University, 2010.
[30] H. Murakami, *Optimization of Bandpass Filters for Eigensolver*, Tech. Report 2010-HPC-124, Tokyo Metropolitan University, 2010.
[31] J. Nocedal and S. Wright, *Numerical Optimization*, 2nd ed., Springer-Verlag, New York, 2006, https://doi.org/10.1007/978-0-387-40065-5.
[32] E. Polizzi, *Density-matrix-based algorithm for solving eigenvalue problems*, Phys. Rev. B, 79 (2009), pp. 115112–115117, https://doi.org/10.1103/PhysRevB.79.115112.

[33] 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, M. Sato, ed., Springer Singapore, Singapore, 2019, pp. 37–57, https://doi.org/10.1007/978-981-13-1924-2_3.

[34] T. Sakurai and H. Sugiura, *A projection method for generalized eigenvalue problems using numerical integration*, J. Comput. Appl. Math., 159 (2003), pp. 119–128, https://doi.org/10.1016/S0377-0427(03)00565-X.

[35] T. Sakurai and H. Tadano, *CIRR: A Rayleigh-Ritz type method with contour integral for generalized eigenvalue problems*, Hokkaido Math. J., 36 (2007), pp. 745–757, https://doi.org/10.14492/hokmj/1272848031.

[36] T. Sakurai and H. Tadano, *CIRR: A Rayleigh-Ritz type method with contour integral for generalized eigenvalue problems*, Hokkaido Math. J., 36 (2007), pp. 745–757, https://doi.org/10.14492/hokmj/1272848031.

[37] L. Sorber, M. V. Barel, and L. D. Lathauwer, *Unconstrained optimization of real functions in complex variables*, SIAM J. Optim., 22 (2012), pp. 879–898, https://doi.org/10.1137/110832124.

[38] P. T. P. Tang and E. Polizzi, *FEAST as a subspace iteration eigensolver accelerated by approximate spectral projection*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 354–390, https://doi.org/10.1137/13090866X.

[39] M. Van Barel, *Designing rational filter functions for solving eigenvalue problems by contour integration*, Linear Algebra Appl., 502 (2016), pp. 346–365.

[40] J. Winkelmann and E. Di Napoli, *Non-linear least-squares optimization of rational filters for the solution of interior Hermition eigenvalue problems*, Front Appl. Math. Stat., 5 (2019), 5.

[41] Y. Xi and Y. Saad, *Computing partial spectra with least-squares rational filters*, SIAM J. Sci. Comput., 38 (2016), pp. A3020–A3045.

[42] C. Zhu, R. Byrd, P. Lu, and J. Nocedal, *Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization*, ACM Trans. Math. Software, 23 (1997), pp. 550–560, https://doi.org/10.1145/279232.279236.