frontiers in Applied Mathematics and Statistics

ORIGINAL RESEARCH
published: 15 February 2019
doi: 10.3389/fams.2019.00005

Non-linear Least-Squares Optimization of Rational Filters for the Solution of Interior Hermitian Eigenvalue Problems

Jan Winkelmann[1] *and Edoardo Di Napoli*[2]*

[1] *Aachen Institute for Advanced Study in Computational Engineering Science, RWTH Aachen University, Aachen, Germany,*
[2] *Jülich Supercomputing Centre, Institute for Advanced Simulation, Forschungszentrum Jülich GmbH, Julich, Germany*

Rational filter functions can be used to improve convergence of contour-based eigensolvers, a popular family of algorithms for the solution of the interior eigenvalue problem. We present a framework for the optimization of rational filters based on a non-convex weighted Least-Squares scheme. When used in combination with a contour based eigensolvers library, our filters out-perform existing ones on a large and representative set of benchmark problems. This work provides a detailed description of: (1) a set up of the optimization process that exploits symmetries of the filter function for Hermitian eigenproblems, (2) a formulation of the gradient descent and Levenberg-Marquardt algorithms that exploits the symmetries, (3) a method to select the starting position for the optimization algorithms that reliably produces effective filters, (4) a constrained optimization scheme that produces filter functions with specific properties that may be beneficial to the performance of the eigensolver that employs them.

Keywords: Hermitian eigenproblem, rational approximation, spectral projector, contour based eigensolver, subspace iteration

OPEN ACCESS

Edited by:
Sorin-Mihai Grad,
Technische Universität Chemnitz, Germany

Reviewed by:
Vladimir Shikhman,
Technische Universität Chemnitz, Germany
Akira Imakura,
University of Tsukuba, Japan

***Correspondence:**
Edoardo Di Napoli
e.di.napoli@fz-juelich.de

Specialty section:
This article was submitted to Optimization, a section of the journal Frontiers in Applied Mathematics and Statistics

Received: *28 August 2018*
Accepted: *18 January 2019*
Published: *15 February 2019*

Citation:
Winkelmann J and Di Napoli E (2019) Non-linear Least-Squares Optimization of Rational Filters for the Solution of Interior Hermitian Eigenvalue Problems. Front. Appl. Math. Stat. 5:5. doi: 10.3389/fams.2019.00005

1. INTRODUCTION

The last fifteen years have witnessed a proliferation of papers on contour based methods for the solution of the Hermitian interior eigenvalue problem [1–4]. Contour based eigensolvers can be described as a family of methods based on the Cauchy's residue Theorem. The name "contour based" stems from an integration of the matrix resolvent $(M - zI)^{-1}$ along a contour in the complex plane encircling an interval $[a, b]$ within the spectrum of M. These contour integrals are calculated via numerical quadrature which exchange the direct solution of the eigenproblem for that of multiple independent linear systems. A key insight to the use of numerical quadrature is the reinterpretation of the contour integration of the matrix resolvent as a matrix-valued rational filter that maps eigenvalues inside and outside $[a, b]$ to one and zero respectively. From this point of view, the mathematical formalism of contour based methods can be considered closely related to the richer mathematical field of rational function approximation.

In this paper we conduct a detailed investigation of how a carefully crafted rational function can improve the efficacy of contour based eigensolvers. To this end, we look at the the numerical quadrature of a complex-valued resolvent $(t - z)^{-1}$ as an approximation of an ideal filter represented by the standard indicator function. Then, we cast the problem of finding an efficient filter as one of finding a continuous rational function that approximates the discontinuous indicator

function so as to improve the effectiveness of the eigensolver. Our method is based on a non-linear Least-Squares optimization process, whose outcome is the simultaneous selection of poles and coefficients characterizing a series of rational filters we termed **S**ymmetric non-**L**inear Opt**i**mized Least-**S**quar**e**s (SLiSe). We show that, given a large systematic selection of distinct intervals per problem M, our SLiSe filters have a statistically better rate of convergence than any other state-of-the-art rational filter. Moreover, our approach allows for constrained-optimization, which is a step toward problem-specific filters that are flexible enough to be capable of exploiting available information on the spectral structure of a given problem. Besides better convergence rates, such an approach can result in an eigensolver with better performance and less parallel workload imbalance. We illustrate the mathematical background, the construction of the optimization framework, and provide a software implementation of it in Julia [5] that generates ready-to-use rational filters.

1.1. Related Work

Two well-known contour based eigensolvers are the SS family of solvers due to Sakurai and Sugiura [2], and FEAST library due to Polizzi [1]. Early work by Sakurai et al. resulted in the SS-Hankel method (SS-H), a non-iterative method based on complex moment matrices. A different approach is taken by Polizzi's FEAST, which is an iterative contour based solver. Underlying FEAST is a Rayleigh-Ritz procedure where the contour integral is used to improve convergence. Later work by Sakurai et al. resulted in a Rayleigh-Ritz type method (SS-R) [3, 6] and various block methods [7]. Beyn proposed a similar method in [8]. A recent comparison of common contour based solvers is available in Imakura et al. [9].

The correspondence between contour methods and rational filters was discussed for FEAST in Tang and Polizzi [4], for SS-H in Ikegami et al. [3], and for SS-R in Ikegami and Sakurai [6]. Early on, Murakami proposed the use of classic rational filters from signal processing [10–14] in the "filter diagonalization method"; for a discussion of this work we refer to [15]. For FEAST the Elliptical filters were proposed in Güttel et al. [16] to improve load balancing and (in some cases) convergence behavior. The rational function point-of-view also enables the development of specialized algorithms: for instance, for real-symmetric matrices it is possible to avoid complex-arithmetic entirely [15].

More recently the filter itself has been treated as a parameter that can be designed via optimization methods. Barel [17] proposed a non-linear Least-Squares approach for non-Hermitian filters within the SS-H framework, while Xi and Saad [18] described linear Least-Squares optimized filters for the Hermitian FEAST solver. Since Barel's approach is geared toward non-Hermitian eigenproblems, it is not possible to make use of the conjugate or reflection symmetry. The resulting degrees of freedom make for a more difficult optimization problem. By requiring a parameter space search, his approach results in filters that are not as robust as our SLiSe. Barel's Least-Squares optimization is based on the discrete ℓ_2 norm, not a function approximation approach. Even in the case of Hermitian problems, optimizing the squared distances in only a sample of points must be done with great care. Choosing too small a number of sample points can have undesirable effects: it is possible to obtain poles near the real axis which are not detected by the sample points. On the other hand having a large number of sample points is very expensive. Finally, Barel's approach does not support constraints, as we present in section 4.4.

Xi and Saad present a FEAST related approach for linear Least-Squares optimized filters focusing on the Hermitian eigenproblem [18]. In this work, not the poles but only the coefficients of the rational function are optimized. The result is a robust process that is much easier to solve, at the cost of being less expressive. An optimization approach that does not optimize poles is limited by the initial choice of them. As we have shown in this work, the process of optimizing a SLiSe filter significantly moves the poles inside the complex plane. On the opposite, fixing the poles imposes a constraint on the optimization that is even larger than constraining just their imaginary part as illustrated in sections 4.4, 5. In practice, when optimizing solely the coefficients of the rational approximation, the optimization algorithm behaves very differently. For example, positive penalty parameters work much better in Xi and Saad's approach, because real poles cannot occur.

1.2. Contributions

This manuscript contains the following original contributions:

- Our optimization framework is based on a weighted Least-Squares function computed using the L_2 norm. All the quantities appearing in our optimization are computed using exact formulas for definite integrals as opposed to numerical quadrature. This is a very distinct approach than optimizing the rational function only at a number of sample points via the ℓ_2 norm as it is done, for instance, in [17]. The residual level function, its gradient, and the partial Hessian are all based on definite L_2 integrals. These quantities are then utilized with two distinct optimization methods: the traditional steepest descent and the more effective Levenberg-Marquardt. Flexibility is the landmark of our framework: we present penalty parameters and box constraints to obtain filters that, for example, are better suited for Krylov-based linear system solvers.
- We make explicit the intrinsic symmetries (conjugation and parity) of the rational function approximating the indicator function. As a consequence, when computing the quantities that enter in the optimization process, such as the gradient and Hessian, we keep track of fewer degrees of freedom. Since each process is iterative and the same numerical quantities have to be computed hundreds of thousands of times, reducing the sheer number of them results in at least four times faster numerical executions.
- Optimizing the rational function's poles and coefficients requires the solution of a non-convex problem for which we resort to a non-linear Least-Squares formulation. This is quite a harder problem than optimizing only for the Least-Squares coefficients as it is done, for instance, in [18]. Common pitfalls on non-convex optimizations are asymmetric rational filters, or failure to converge to a local minumum. In order to yield

consistent results, we provide a rationale for the selection of initial parameters that consistently yield filters obtained from a robust optimization process.
– We built and implemented an optimization package, written in the Julia language, which is publicly available and easy to use. As it stands the package could be easily integrated in existing eigensolver libraries, such as FEAST, to provide customized rational filters computed on the fly. In order to showcase the promise of our implementation, we also provide ready-to-use SLiSe filters. We compare such filters with existing ones using a large representative problem set. These SLiSe filters can be used as drop-in replacements for the current state-of-the-art spectrum-slicing eigensolver libraries. For instance, in FEAST users can employ the so-called expert routines as described in section 2.2.

The rest of this paper is structured as follows. To showcase the potential of our optimization method, section 2 discusses examples of SLiSe filters. Section 3 considers the formulation of the non-linear Least-Squares problem. We discuss the use of symmetries of the filters, as well as the residual level function and its gradients, which also make use of these symmetries. Section 4 illustrates the full framework for the generation of SLiSe filters with and without constraints. In section 5, we provide additional examples of optimized filters, and we conclude in section 6.

2. SPECTRAL PROJECTION USING SLISE FILTERS

In this section we briefly introduce the basics of subspace iteration with spectral projection. Then we provide an illustration of the effectiveness of our optimization framework by presenting SLiSe filters that are meant as replacements for existing state-of-the-art rational filters. The mathematical setup and methods by which these filters were obtained are the topic of later sections. The filters used in this section, and the optimization parameters used to obtain them, are available in **Appendix C**.

In section 2.2, we discuss a replacement for the Gauss filter. An improvement for this filter is particularly useful, since the Gauss filter and its variations are the default choice for many contour based solvers [1, 19]. Next, we illustrate the replacement for the Elliptic filter that is used for eigenproblems where an endpoint of the search interval coincides with, or is near, a large spectral cluster. Such a scenario can occur when multiple adjacent search intervals are selected. For the sake of brevity we limit the examples to rational filters with a total of 16 poles and relative coefficients. Naturally, the optimization methods presented in this work are applicable to filters with an arbitrary number of poles.

2.1. Subspace Iteration Method Accelerated via Spectral Projection

Given a Hermitian matrix $M = M^\dagger \in \mathbb{C}^{N\times N}$ and a proper interval $[a, b]$, with $[a, b] \cap [\lambda_1, \lambda_N] \neq \emptyset$, we are interested in finding the $\mathfrak{m}$ eigenpairs (λ, v) inside $[a, b]$ resolving the secular equation[1]:

$$Mv = \lambda v.$$

An efficient subspace iteration method requires a projection procedure that identifies a search subspace approximating the invariant eigenspace corresponding to the eigenvalues lying in the $[a, b]$ interval. In the course of this paper we focus solely on methods that achieve such projection through a rational function $f(M)$ of the matrix M. Such functions are also known as rational filters although other filter forms, such as polynomial filters, do exist. A rational filter of degree $(n-1, n)$ can be expressed as a matrix function in partial fraction decomposition

$$f(M) \doteq \sum_{i=1}^{n} \alpha_i (M - Iz_i)^{-1} \tag{1}$$

where $z_i \in \mathbb{C}\backslash\mathbb{R}$ and $\alpha_i \in \mathbb{C}$ are all distinct and are chosen in a way such that the eigenvalues of M inside $[a, b]$ are mapped to roughly one, and the eigenvalues outside are mapped to roughly zero. As a result, the filter suppresses eigenvalues outside of $[a, b]$, which improves convergence of the subspace iteration for the desired eigenvalues and eigenvectors. As a scalar function of $t \in \mathbb{R}$, $f(t)$ can be seen as a rational approximation to the indicator function $\chi_{[a,b]}(t)$, with unit value inside $[a, b]$ and zero everywhere else. Filters are usually generated for a search interval of $[-1, 1]$, and then mapped to $[a, b]$ via an appropriate transformation on the α_i's and z_i's. Without loss of generality we always consider the search interval to be $[-1, 1]$. Forming $f(M)$ from Equation (1) explicitly involves the calculation of the matrix inverse. Using $f(M)$ within a subspace iteration procedure requires only $X := f(M)Y$, which can be rewritten as n independent linear system solves:

$$(M - Iz_i)X = \alpha_i Y \qquad 1 \leq i \leq n. \tag{2}$$

If we assume that $f(M)$ outputs a good approximation to the search subspace, a common form of subspace iteration is based on the use of the Rayleigh-Ritz (RR) projection at each iteration. Provided that the approximating subspace has a dimension $\mathfrak{p}$ equal or larger than $\mathfrak{m}$, such a projection reduces the size of the eigenproblem to be solved, and is guaranteed to output eigenpairs of the reduced problem that are in a one-to-one correspondence with the eigenpairs inside the search interval.

One of the most well-known solvers for the interior eigenvalue problem based on a rational resolvent followed by the RR projection is the FEAST library [1]. In FEAST the filter function is derived via n-point quadrature realization of the contour integral:

$$\frac{1}{2\pi i}\int_\Gamma \frac{\mathrm{d}t}{M - It} \approx \frac{1}{2\pi i}\sum_{i=1}^{n} w_i (M - Ix_i)^{-1} = f(M) \tag{3}$$

where Γ is a contour in the complex plane enclosing the search interval $[a, b]$. Quadrature along Γ yields integration nodes

[1] For notational convenience in the rest of the paper we make use of the $M^\dagger$ instead of the conventional M^H to indicate Hermitian conjugation.

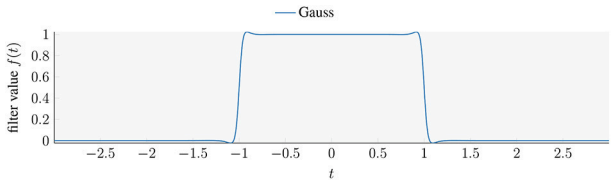

FIGURE 1 | Plot of the Gauss filter obtained by 16-points Gauss-Legendre quadrature of Equation (3), assuming $t \in \mathbb{R}$.

$x_i \in \mathbb{C}$ and weights $w_i \in \mathbb{C}$. While any quadrature rule can filter the eigenvalues to some extent, FEAST's defaults to the Gauss-Legendre quadrature rules. From Equation (3) follows that numerical integration of the contour integral is functionally equivalent to the rational filter formulation of Equation (1). Accordingly, we can interpret FEAST's filter function as a rational filter function with $x_i = z_i$ and $-\frac{i}{2\pi} w_i = \alpha_i$, which we call "Gauss filter." As we are considering Hermitian eigenproblems, we can plot the corresponding scalar function $f(t)$. **Figure 1** plots the Gauss filter obtained by a 16-points quadrature on a circle-shaped contour that circumscribes $[-1, 1]$ symmetrically with respect to the real axis. The resulting function is obviously even and real-valued over the entire real axis.

Given an ordering of the eigenvalues such that

$$|f(\lambda_1)| \geq |f(\lambda_2)| \geq \ldots \geq |f(\lambda_{\mathfrak{m}})| \geq \ldots \geq |f(\lambda_{\mathfrak{p}})| \geq \ldots \geq |f(\lambda_{\mathfrak{p}+1})| \geq \ldots \geq |f(\lambda_n)|,$$

the convergence ratio of the FEAST algorithm for a chosen filter f is given by [4, Theorem 4.4]

$$\tau = \frac{|f(\lambda_{\mathfrak{p}+1})|}{|f(\lambda_{\mathfrak{m}})|}. \tag{4}$$

For most filters it is the case that $f(1) = f(-1) = \frac{1}{2}$, in which case we can extend the condition to: $|f(\lambda_{\mathfrak{m}})| \geq \ldots \geq \frac{1}{2} \geq \ldots \geq |f(\lambda_{\mathfrak{p}})|$.

2.2. γ-SLiSe: A Replacement Candidate for the Gauss Filter

In order to have a fair comparison between an existing filter and our candidate as filter replacement, we need to establish a testing environment. Since our focus is the use of rational filters within subspace iteration methods, we use FEAST as our test environment for comparing filters. A simple filter comparison could use the number of FEAST iterations required to converge the entire subspace for a given matrix M and interval $[a, b]$, which we refer as *benchmark problem*. Obviously, the filter that requires fewer iterations is the better one. While rather basic, we show that this simple criterion already provides useful insights. Later in the paper we introduce a more sophisticated evaluation criterion based on the convergence ratio of the benchmark problem.

Once the comparison criterion has been selected, a natural way to decide which filter is superior is to select many, representative benchmark problems and see which filter performs better. Comparing filters on only a few, hand-selected, problems can introduce a strong bias. In **Appendix B**, we propose a method to obtain many benchmark problems which we will use throughout this section. The resulting comparison is based on a large number of these benchmark problems and provides a good statistical measure of our filters' quality. To construct a benchmark set, we fix the matrix M, construct a large number of distinct search intervals $[a, b]$, compute the exact number of eigenvalues $\mathfrak{m}$ each interval contains, and then set $\mathfrak{p}$ to be a multiple of this number. The benchmark set used throughout this section was obtained by selecting 2116 search intervals, each containing between 5 and 20% of the spectrum of the Hermitian "Si2" matrix of the University of Florida matrix collection [20]. The Gauss and γ-SLiSe filters were used as filtering method in FEAST's version 3.0 through the `scsrevx` routine, a driver that uses a sparse direct solver for the linear system solves[2]. For a problem with $\mathfrak{m}$ eigenpairs inside the search interval we select a size of the subspace iteration of $\mathfrak{p} = 1.5\mathfrak{m}$, the value that FEAST recommends for the Gauss filter.

Figure 2A shows a histogram of the iterations required for the 2,116 benchmark problems by the γ-SLiSe and the Gauss filter. Independently of which filter is used the vast majority of problems of the set requires either 3 or 4 FEAST iterations. Fast convergence of FEAST's subspace iteration is a known feature for the Gauss filter when $\mathfrak{p}$ is chosen large enough. When FEAST uses the γ-SLiSe filter, most benchmark problems require 3 iterations. In contrast, when the Gauss filter is used, a larger number of problems require 4 iterations. Summing the iterations for all of the benchmark problems the γ-SLiSe filter requires 6,774 iterations, while the Gauss filter requires 7,119 iterations. Since every iteration requires 8 linear system solves and additional overhead, such as the calculation of the residuals, saving even a single iteration is a substantial performance improvement.

The increased performance of the γ-SLiSe filter comes with some drawbacks. On the one hand, the Gauss filter is more versatile than the γ-SLiSe. For instance, outside of the selected interval, the Gauss filter decays very quickly to low values (see **Figure 3B**), which yields better convergence when increasing the number of vectors in the subspace iterations $\mathfrak{p}$ to a larger multiple of $\mathfrak{m}$. The γ-SLiSe does not have this property, and so does worse when a very large spectral gap is present. On the other hand, SLiSe filters are more flexible, so it is always possible to generate a different γ-SLiSe filter that performs well also for large spectral gaps.

2.2.1. Convergence Ratio as a Means of Comparison

While **Figure 2A** shows some promise as a tool for comparing filters, the approach has a number of problems. Subspace iteration counts provide a very coarse look at the performance of a filter, underlined by the fact that most benchmark problems require a very similar number of iterations for both filters. Furthermore, the evaluation requires the use of a linear system solver, and thus the outcome is dependent on the solver's algorithm and the parameters used (such as the desired accuracy,

[2]Feast was compiled with the Intel Compiler v16.0.2 and executed with a single thread; the target residual was adjusted to 10^{-13}.

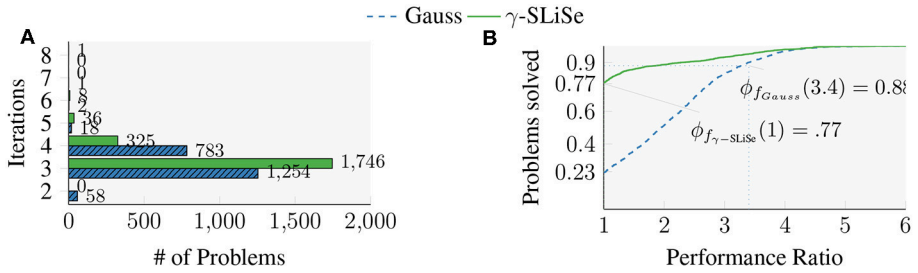

FIGURE 2 | Comparisons of the Gauss filter with our replacement candidate, γ-SLiSe, with $\mathfrak{p} = 1.5\mathfrak{m}$ on a test set of 2116 benchmark problems obtained from the "Si2" matrix. Both filters have 16 poles and are plotted in **Figure 3**. **(A)** Number of FEAST iterations required by γ-SLiSe and the Gauss filter. **(B)** Performance profiles of convergence rates for γ- SLiSe and Gauss filters.

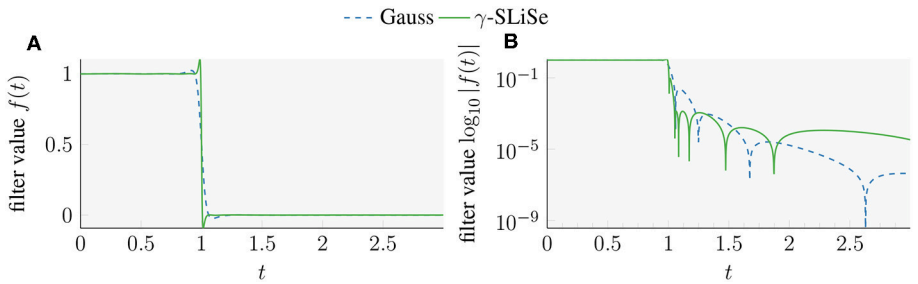

FIGURE 3 | Plots of the Gauss filter and our replacement candidate, γ-SLiSe. Both filters have 16 poles and coefficients. **(A)** Linear-scale plot. **(B)** Semi-log plot.

the type of solver, etc.). A reliable comparison should be based on a more fine-grained metric which does not depend on a specific solver, its specific parameters, and the computational kernels used. One such metric is the ratio of convergence τ defined in Equation (4). For small to medium sized problems, we can obtain all eigenvalues of the matrix and calculate τ analytically. Since τ is a function of only the eigenvalues, it does not depend on a specific solver, and it is cheaper to compute. For testing purposes, we computed all eigenvalues up front, and then calculated τ for every benchmark problem.

Figure 2B shows the convergence ratios for the same benchmark set as in **Figure 2A** in a form called *performance profiles* [21]. For a filter f, and a given point x on the abscissa, the corresponding value $\phi_f(x)$ of the graph indicates that for $100 \cdot \phi_f(x)$ percent of the benchmark problems the filter f is at most a factor of x worse than the fastest of all methods in question. For example, $\phi_{f_{Gauss}}(3.4) \approx 0.88$ indicates that for 88% of the benchmark problems the Gauss filter yields a ratio that is at worst a factor of 3.4 from the best. So the performance profiles not only report how often a filter performs best, but also how badly the filter performs when it is not the best. We will use performance profiles multiple times in this section to compare the filters. From the value of $\phi_{f_{\gamma\text{-SLiSe}}}(1)$ in **Figure 2B** we can infer that γ-SLiSe achieves the best convergence ratio for 77% of the benchmark problems, while the Gauss filter does best on the remaining 23% of the problems. Moreover, the Gauss filter has a convergence ratio that is worse than 3.4 times that of the γ-SLiSe filter for 12% of the problems. In conclusion, the SLiSe filter performs better than the Gauss filter in both metrics, iterations counts and convergence ratio. However, the performance profile plot of τ gives a much more detailed comparison.

A rough insight on the better convergence ratio of the γ-SLiSe filter can be extracted from **Figure 3** where **Figure 3A** plots the positive part of the abscissa of both filter functions, and **Figure 3B** shows the absolute value of the same filter functions as a log plot. Given that we used the FEAST recommended value of $\mathfrak{p} = 1.5\mathfrak{m}$, and making the likely assumption that most of the benchmark problems have a roughly equidistant spacing of the eigenvalues around the canonical search interval $[-1, 1]$, implies that $\lambda_{\mathfrak{p}+1} \approx \pm 1.25$. Due to a smaller absolute value between 1.1 and 1.6, the γ-SLiSe filter out-performs the state-of-the-art Gauss filter as long as $\lambda_{\mathfrak{p}+1} \in [1.1, 1.6]$. If either end of the search interval is near a large spectral cluster then $\lambda_{\mathfrak{p}+1} \ll 1.1$, and neither of the two filters will do particularly well. In such a case Elliptic filters are used.

2.3. η-SLiSe: A Replacement Candidate for the Elliptic Filter

The Elliptic filter, also called Cauer or Zolotarev filter, has been proposed for use in the context of the interior eigenvalue problem in a number of publications [10–14, 16][3]. This filter is used specifically when large spectral clusters are at or near the endpoints of a search interval. In these cases the Gauss

[3]Technically the Elliptic filter is a class of filters depending on a number of parameters. We consider the specific filter discussed in Güttel et al. [16].

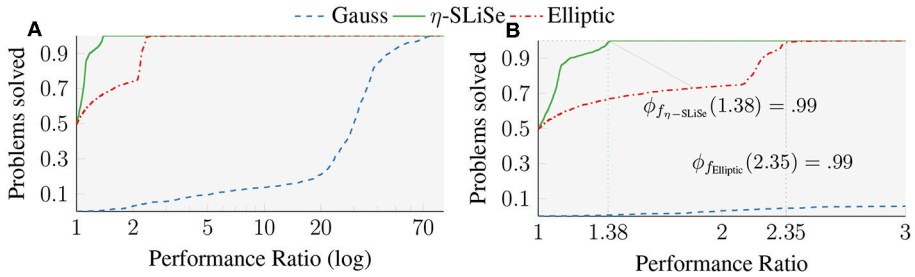

FIGURE 4 | Semi-log performance profile of the convergence ratio for the Gauss, η-SLiSe, and Elliptic filters. η-SLiSe is meant to replace the Elliptic filter. All filters have 16 poles and $\mathfrak{p} = \mathfrak{m}$. **(A)** Semi-log performance profile of the convergence rate. **(B)** The same performance profile zoomed into [1, 3] with a linear axis.

filter exhibits very slow convergence unless $\mathfrak{p}$ is chosen much larger than 1.5$\mathfrak{m}$. We propose η-SLiSe as a replacement for the Elliptic filter currently used in contour-based subspace iteration methods.

The Elliptic filter is an optimal filter, in the sense that the filter function is the best uniform rational approximation of the indicator function. For a subspace size of $\mathfrak{p} = \mathfrak{m}$, such a filter is worst-case optimal with respect to the convergence ratio of Equation (4). In simple terms, the Elliptic filter works better than the Gauss filter in the presence of spectral clusters close to the interval boundary because it drops from 1 to 0 more quickly. The η-SLiSe filter trades off slightly larger absolute values inside $[1, 1.01]$ for smaller absolute values in $[1.01, \infty]$ ($[-1.01, 1]$ and $[-\infty, 1.01]$, respectively).

Figure 4 shows a performance profile of the convergence ratio for the Gauss, η-SLiSe, and Elliptic filters; **Figure 4A** is a semi-log plot of the performance profile, while **Figure 4B** is a linear-scale plot, zoomed into the interval $[1, 3]$. The comparison between the two filters needs to be adjusted to represent the specific use case of spectral clusters near the endpoints. We use again the benchmark set from the "Si2" matrix, however, instead of target subspace size of $\mathfrak{p} = 1.5\mathfrak{m}$, we choose $\mathfrak{p} = \mathfrak{m}$. Thus, the eigenvalue corresponding to the largest absolute filter value outside the search interval determines convergence for all benchmark problems. For large spectral clusters $|f(\lambda_{\mathfrak{p}+1})|$ is likely to be close to $|f(\lambda_{\mathfrak{m}})|$, so this metric serves as an estimator of the filter's behavior with spectral clusters near the endpoints of the search interval.

The figure indicates that, for our representative set of benchmark problems, the η-SLiSe filter displays better convergence rates than the Elliptic filter. The Elliptic filter achieves the best convergence ratio for half of the benchmark problems, and η-SLiSe does it for the other half. Further, the figure indicates that for 99% of the problems, the Elliptic filter has a convergence ratio at most 2.35 times worse than the best against a factor of 1.38 of the η-SLiSe [$\phi_{f_{\text{Elliptic}}}(2.35) = 0.99$, and $\phi_{f_{\eta-\text{SLiSe}}}(1.38) = 0.99$]. Simply put, η-SLiSe performs better than the Elliptic filter for half of the problems, and for the other half it performs very similar to the Elliptical filter. Conversely, the Elliptical filter exhibits convergence rates for some problems that are twice as bad as the convergence rate of η-SLiSe. The Gauss filter exhibits the worst rates of convergence by far, but this is to be expected as the Gauss filter is not meant for this use-case.

3. RATIONAL FILTERS FOR HERMITIAN OPERATORS: A NON-LINEAR LEAST-SQUARES FORMULATION

Rational filters can be written as a sum of simple rational functions ϕ_i

$$f(t, z, \alpha) = \sum_{i=1}^{n} \phi_i \equiv \sum_{i=1}^{n} \frac{\alpha_i}{(t - z_i)} \tag{5}$$

with $z_i, \alpha_i \in \mathbb{C}$. We restrict ourselves to the case where the z_i and the α_i are pair-wise distinct, and the z_i have non-zero real and imaginary parts. Our initial goal is to set the stage for the formulation of a reliable method dictating the choice for the poles z_i and the coefficients α_i such that $f(t)$ constitutes an effective and flexible approximation to the ideal filter $h(t)$. In the Hermitian case such an ideal filter corresponds to the indicator function defined as

$$h(t) = \chi_{[-1,1]} = \begin{cases} 1 & \text{if } -1 \leq t \leq 1 \\ 0 & \text{otherwise} \end{cases}. \tag{6}$$

As $h(t)$ is discontinuous we have to specify in what sense a given filter $f(t, z, \alpha)$ is close to the target function $h(t)$. To this end, we focus on the squared difference of $h(t) - f(t, z, \alpha)$ with respect to the L_2 norm which is equivalent to defining a Least-Squares error function, or *residual level function*

$$F(z, \alpha) = \|h - f(z, \alpha)\|_2^2. \tag{7}$$

An optimal filter, in this sense, is the minimizer of the residual level function

$$\min_{\alpha_i, z_i \; 1 \leq i \leq n} \int_{-\infty}^{\infty} \mathfrak{G}(t) \left| h(t) - \sum_{i=1}^{n} \frac{\alpha_i}{(t - z_i)} \right|^2 dt, \tag{8}$$

resulting in a weighted Least-Squares optimization problem with the weight function $\mathfrak{G}(t)$. In order to simplify the optimization

process we impose the following restrictions on the weight function: (1) $\mathfrak{G}(t)$ is piece-wise constant with bounded interval endpoints, (2) the number of piece-wise constant parts is finite, (3) the weight function is even: $\mathfrak{G}(t) = \mathfrak{G}(-t)$.

In section 3.1, we illustrate how to exploit the symmetries of the indicator function $h(t)$ in the Hermitian case. The rational function $f(t)$ approximating $h(t)$ can be constructed so as to explicitly satisfy these symmetries, which limits the degrees of freedom used in the optimization problem. Section 3.2 contains a matrix formulation of the Least-Squares residual level function, while its gradients are introduced in section 3.3.

3.1. Discrete Symmetries of the Rational Approximant

We wish to construct a rational function $f(t, z, \alpha)$ that explicitly preserves the symmetries of the target function $h(t)$. The target function is invariant under two symmetry transformations, namely *complex conjugation*, and *reflection*. The first symmetry seems trivial since h is real-valued but f in general is not. The second symmetry states that h is even. For the rational function f to be a good approximation of h it should satisfy both symmetries

$$\begin{array}{ll} 1.\ \text{Complex Conjugation (C):} & \overline{f}(t, z, \alpha) = f(t, z, \alpha) \\ 2.\ \text{Reflection or Parity (P):} & f(-t, z, \alpha) = f(t, z, \alpha). \end{array} \tag{9}$$

Requiring the function f to satisfy the C symmetry implies that half of the rational monomials ϕ_is are conjugate of each other. In other words, half of the poles z_is (and corresponding α_is) are in the upper-half of $\mathbb{C}$ (indicated as $\mathbb{H}^+$) with the other half in the lower-half ($\mathbb{H}^-$). Then, without loss of generality we can enumerate the ϕ_i such that the first $p = \frac{n}{2}$ have poles z_i in $\mathbb{H}^+$: $\frac{\alpha_r}{t - z_r} = \phi_r = \overline{\phi_{r+p}}$ with $z_r \in \mathbb{H}^+$ for $1 \leq r \leq p$. This conclusion enables us to rewrite the sum over monomials ϕ_i as a sum over half the index range $1 \leq j \leq p = \frac{n}{2}$

$$f(t, \nu, \beta) = \sum_{j=1}^{p} \left(\psi_j + \overline{\psi}_j \right) \equiv \sum_{j=1}^{p} \left[\frac{\beta_j}{(t - \nu_j)} + \frac{\overline{\beta}_j}{(t - \overline{\nu}_j)} \right], \tag{10}$$

where we relabeled the poles z_is, the coefficients α_is, and the monomials ϕ_is to make the symmetry explicit.

$$\begin{array}{llll} z_j = \nu_j & \alpha_j = \beta_j & \phi_j = \psi_j \\ z_{j+p} = \overline{\nu}_j & \alpha_{j+p} = \overline{\beta}_j & \phi_{j+p} = \overline{\psi}_j \end{array}. \tag{11}$$

Notice that satisfying the C symmetry forces the association of conjugate coefficients $\overline{\beta}_j$ with conjugate poles $\overline{\nu}_j$.

Imposing the P symmetry explicitly is a bit trickier. It can be visualized as a symmetry for the complex numbers ν_j, β_j and their conjugates, living in $\mathbb{C}$. Consider the poles ν_k in the right (R) upper-half of the complex plane $\mathbb{H}^{+R}$, that is those ν_k with $\Re e(\nu_k) > 0$ and $\Im m(\nu_k) > 0$. Let the number of ν_k in $\mathbb{H}^{+R}$ be q. Applying the P symmetry to the monomials ψ_k with poles ν_k in $\mathbb{H}^{+R}$ yields:

$$\psi_k(t, \nu, \beta) = \frac{\beta_k}{(t - \nu_k)} \longrightarrow -\frac{\beta_k}{(t + \nu_k)} = \psi_k(t, -\nu, -\beta).$$

Consequently the reflection operation maps a rational monomial with a pole in $\mathbb{H}^{+R}$ to a monomial with a pole in $\mathbb{H}^{-L}$. For f to be invariant under reflection, the q ψ_j monomials with poles in $\mathbb{H}^{+R}$ "must" map to the $\overline{\psi}_j$ monomials with poles in $\mathbb{H}^{-L}$. By complex conjugation, these same q ψ_j monomials map to $\overline{\psi}_j$ monomials in $\mathbb{H}^{-R}$, consequently $p = 2q$. Now we can enumerate the monomials such that the first half of ψ_j are in $\mathbb{H}^{+R}$, the second half of ψ_j are in the $\mathbb{H}^{+L}$, the first half of $\overline{\psi}_j$ are in $\mathbb{H}^{-R}$ and the last q $\overline{\psi}_j$ are in $\mathbb{H}^{-L}$. Invariance under reflection implies then that $\psi_k(t, -\nu, -\beta) = \overline{\psi}_{k+q}(t, \overline{\nu}, \overline{\beta})$ for $1 \leq k \leq q$. The same reasoning can be repeated for monomials ψ_k in $\mathbb{H}^{+L}$. Finally, we can express f by summing over a reduced range $1 \leq k \leq q$ as

$$\begin{aligned} f(t, w, \gamma) &= \sum_{k=1}^{q} \left(\chi_k + \overline{\chi}_k + \chi_k^{\top\!\!\top} + \overline{\chi}_k^{\top\!\!\top} \right) \\ &= \sum_{k=1}^{q} \left[\frac{\gamma_k}{t - w_k} + \frac{\overline{\gamma}_k}{t - \overline{w}_k} - \frac{\gamma_k}{t + w_k} - \frac{\overline{\gamma}_k}{t + \overline{w}_k} \right]. \end{aligned} \tag{12}$$

Once again, we have relabeled poles, coefficients, and monomials so as to make explicit the symmetry indicated by the $\top\!\!\top$ symbol

$$\begin{array}{llll} \nu_k = w_k & \beta_k = \gamma_k & \psi_k = \chi_k \\ \nu_{k+q} = \overline{w}_k^{\top\!\!\top} = -\overline{w}_k & \beta_{k+q} = \overline{\gamma}_k^{\top\!\!\top} = -\overline{\gamma}_k & \psi_{k+q} = \overline{\chi}_k^{\top\!\!\top} \\ \overline{\nu}_k = \overline{w}_k & \overline{\beta}_k = \overline{\gamma}_k & \overline{\psi}_k = \overline{\chi}_k \\ \overline{\nu}_{k+q} = w_k^{\top\!\!\top} = -w_k & \overline{\beta}_{k+q} = \gamma_k^{\top\!\!\top} = -\gamma_k & \overline{\psi}_{k+q} = \chi_k^{\top\!\!\top} \end{array}. \tag{13}$$

We would have reached the same result if we started to require invariance under the symmetries in reverse order[4].

3.2. Residual Level Function

Let us expand Equation (7) by temporarily disregarding the discrete symmetries of the rational function and expressing f as in Equation (5)

$$\begin{aligned} F(z, \alpha) &= \| h - \sum_i \alpha_i \phi_i \|_2^2 \\ &= < h - \sum_i \alpha_i \phi_i, h - \sum_i \alpha_j \phi_j > \\ &= < h, h > - 2\Re e[\sum_i < \phi_i, h > \alpha_i] + \sum_i \sum_j \overline{\alpha}_i \alpha_j \\ &\quad < \phi_i, \phi_j > . \end{aligned}$$

The residual level function is expressible as

$$F(z, \alpha) = \alpha^\dagger G \alpha - 2\Re e[\eta^\dagger \alpha] + \| h \|^2 \tag{14}$$

where the $x^\dagger \equiv \overline{x}^\top$ indicates complex conjugation plus transposition (Hermitian conjugation) and

$$G_{ij} = < \phi_i, \phi_j >, \quad \eta_i = < \phi_i, h > .$$

[4]Physicists refer to the combination of these two discrete symmetries as CP invariance. A necessary condition for its existence is that the operators generating the transformations must commute.

The inner products $< \cdot, \cdot >$ are defined through a weight function $\mathfrak{G}(t)$

$$< \phi_i, \phi_j >= \int_{-\infty}^{+\infty} \mathfrak{G}(t)\overline{\phi_i}(t)\phi_j(t)dt = \int_{-\infty}^{+\infty} \frac{\mathfrak{G}(t)}{(t-\overline{z}_i)(t-z_j)}\, dt, \tag{15}$$

and

$$\eta_i =< \phi_i, h >= \int_{-\infty}^{+\infty} \frac{\mathfrak{G}(t)h(t)}{t-\overline{z}_i}\, dt \qquad \|h\|^2 = \int_{-\infty}^{+\infty} \mathfrak{G}(t)h(t)^2\, dt.$$

We could have started directly with the CP invariant formulation from Equation (12) of the rational function f. We preferred this approach in order to show how requiring each symmetry to be satisfied has direct consequences on the structure of the matrix G and vector η. Recall from Equation (11) how the z_is and α_is were mapped to ν_js and β_js so as to make explicit the C symmetry. Due to such map, the residual level function has the following block structure:

$$F(\nu, \beta) = \begin{pmatrix}\beta^\dagger & \beta^\top\end{pmatrix}\begin{pmatrix}A & B\\ \overline{B} & \overline{A}\end{pmatrix}\begin{pmatrix}\beta\\ \overline{\beta}\end{pmatrix} - 2\Re e\left[\begin{pmatrix}\zeta^\dagger & \zeta^\top\end{pmatrix}\begin{pmatrix}\beta\\ \overline{\beta}\end{pmatrix}\right] + \|h\|^2 \tag{16}$$

with

$$A_{i,j} = \int_{-\infty}^{+\infty} \frac{\mathfrak{G}(t)}{(t-\overline{\nu}_i)(t-\nu_j)}\, dt \quad \zeta_i = \int_{-\infty}^{+\infty} \frac{\mathfrak{G}(t)h(t)}{t-\overline{\nu}_i}\, dt$$
$$B_{\substack{i,j\\ i\neq j}} = \int_{-\infty}^{+\infty} \frac{\mathfrak{G}(t)}{(t-\overline{\nu}_i)(t-\overline{\nu}_j)}\, dt \quad B_{i,i} = \int_{-\infty}^{+\infty} \frac{\mathfrak{G}(t)}{(t-\overline{\nu}_i)^2}\, dt$$

for $i, j = 1, \ldots, p$. The matrix A is Hermitian while B is symmetric (complex), so that $\overline{B} = B^\dagger$. This equation can be reduced since it contains only half of the unknowns as the initial residual level function. For example $\beta^\top\overline{A}\,\overline{\beta} = \beta^\dagger A\beta$ and $\beta^\dagger B\overline{\beta} = \overline{(\beta^\top\overline{B}\beta)}$, so that

$$\begin{pmatrix}\beta^\dagger & \beta^\top\end{pmatrix}\begin{pmatrix}A & B\\ \overline{B} & \overline{A}\end{pmatrix}\begin{pmatrix}\beta\\ \overline{\beta}\end{pmatrix} = 2\left[\beta^\dagger A\beta + \Re e\left(\beta^\dagger B\overline{\beta}\right)\right]$$

and similarly $\zeta^\dagger\beta = \overline{\zeta^\top\overline{\beta}}$. With the simplifications above the residual level function reduces to

$$F(\nu, \beta) = \beta^\dagger A\beta + \Re e\left(\beta^\dagger B\overline{\beta}\right) - 2\Re e\left(\zeta^\dagger\beta\right) + \frac{1}{2}\|h\|^2. \tag{17}$$

Now we can require the latter equation to satisfy the reflection symmetry P. Equation (13) maps poles ν_j and coefficients β_j respectively to w_k and γ_k, so as to make the reflection symmetry explicit. The residual level function now takes on a new block form

$$F(w, \gamma) = \begin{pmatrix}\gamma^\dagger & (\gamma^{\overline{\top}})^\top\end{pmatrix}\begin{pmatrix}X & Y\\ Y^\dagger & \overline{X}^{\overline{\top}}\end{pmatrix}\begin{pmatrix}\gamma\\ \overline{\gamma}^{\overline{\top}}\end{pmatrix} + \Re e\left[\begin{pmatrix}\gamma^\dagger & (\gamma^{\overline{\top}})^\top\end{pmatrix}\begin{pmatrix}W & Z\\ Z^\dagger & \overline{W}^{\overline{\top}}\end{pmatrix}\begin{pmatrix}\overline{\gamma}\\ \gamma^{\overline{\top}}\end{pmatrix}\right]$$
$$- 2\Re e\left[\begin{pmatrix}\theta^\dagger & (\theta^{\overline{\top}})^\top\end{pmatrix}\begin{pmatrix}\gamma\\ \overline{\gamma}^{\overline{\top}}\end{pmatrix}\right] + \frac{1}{2}\|h\|^2 \tag{18}$$

with

$$W_{k,\ell} = \int_{-\infty}^{+\infty} \frac{\mathfrak{G}(t)}{(t-\overline{w}_k)(t-\overline{w}_\ell)}\, dt \quad \overline{W}^{\overline{\top}}_{k,\ell} = \int_{-\infty}^{+\infty} \frac{\mathfrak{G}(t)}{(t+w_k)(t+w_\ell)}\, dt$$
$$X_{k,\ell} = \int_{-\infty}^{+\infty} \frac{\mathfrak{G}(t)}{(t-\overline{w}_k)(t-w_\ell)}\, dt \quad \overline{X}^{\overline{\top}}_{k,\ell} = \int_{-\infty}^{+\infty} \frac{\mathfrak{G}(t)}{(t+w_k)(t+\overline{w}_\ell)}\, dt$$
$$Y_{k,\ell} = \int_{-\infty}^{+\infty} \frac{\mathfrak{G}(t)}{(t-\overline{w}_k)(t+\overline{w}_\ell)}\, dt \quad Z_{k,\ell} = \int_{-\infty}^{+\infty} \frac{\mathfrak{G}(t)}{(t-\overline{w}_k)(t+w_\ell)}\, dt$$
$$\theta_k = \int_{-\infty}^{+\infty} \frac{\mathfrak{G}(t)h(t)}{t-\overline{w}_k}\, dt \quad \overline{\theta}^{\overline{\top}}_k = \int_{-\infty}^{+\infty} \frac{\mathfrak{G}(t)h(t)}{t+w_k}\, dt$$

for $k, \ell = 1, \ldots, q$. These submatrix blocks can be computed analytically (see **Appendix A**), and preserve some of the properties of the matrix they are part of. For instance, $X^\dagger = X$ and $\overline{W} = W^\dagger$ while the P symmetry imposes new equalities $Y^\dagger = \overline{Y}^{\overline{\top}}$ and $Z^\dagger = \overline{Z}^{\overline{\top}}$. In addition to these, the P symmetry allows for additional equalities thanks to the symmetric integration boundaries of the inner product $\langle \cdot, \cdot \rangle$. In other words

$$\int_0^a g(t, w_k)\, dt = \int_{-a}^0 g^{\overline{\top}}(t, w_k)\, dt \qquad \text{for } g \text{ an even function of } t$$
$$\int_0^a g(t, w_k)\, dt = -\int_{-a}^0 g^{\overline{\top}}(t, w_k)\, dt \qquad \text{for } g \text{ an odd function of } t.$$

The direct implication of this observation is that, for instance, $X = X^{\overline{\top}}$, $W = W^{\overline{\top}}$.

If we expand the matrix expression for the residual level function and exploit all the symmetries the final expression for F becomes

$$F(w, \gamma) = \gamma^\dagger X\gamma + \left(\gamma^\dagger X\gamma\right)^\top - 2\,\Re e\left(\gamma^\dagger Y\overline{\gamma}\right) \tag{19}$$
$$+ \Re e\left[\gamma^\dagger W\overline{\gamma} + \left(\gamma^\dagger W\overline{\gamma}\right)^\dagger - 2\,\gamma^\dagger Z\gamma\right]$$
$$- 2\,\Re e\left(\theta^\dagger\gamma - (\theta^{\overline{\top}})^\top\overline{\gamma}\right) + \frac{1}{2}\|h\|^2.$$

Despite the apparent complexity of the expression above, it would have been quite more complex if we started to compute the residual level function directly from Equation (12); the quadratic term in γs alone would have accounted for 16 terms. Moreover the expression is a function of only γ_ks and their conjugates. Z and Y matrices appear only once, while X and W can be transformed after their computation.

3.3. Gradient of the Residual Level Function

Most optimization methods—including all the ones that we consider—require the gradient of the residual level function. We could compute the gradients analytically by using the expression for F derived in Equation (14). A simpler way is to compute ∇f out of the formulation in Equation (5) in conjunction with the formulation of F in terms on inner products $\langle \cdot, \cdot \rangle$

$$\begin{aligned}\nabla F &= \nabla\langle h-f, h-f\rangle = -\langle\nabla f, h-f\rangle - \langle h-f, \nabla f\rangle \\ &= 2\left[\langle f, \nabla f\rangle - \langle h, \nabla f\rangle\right]. \end{aligned} \tag{20}$$

Since $\overline{f} = f$ and the ∇ operator acts on the whole inner products, the quantities $\langle\nabla f, f\rangle$ and $\langle f, \nabla f\rangle$, while formally different, are actually the same. In addition, the gradient of a real function with respect to the conjugate of a complex variable is necessarily the conjugate of the gradient with respect to the variable itself: $\nabla_{\overline{z}} f = \frac{\partial f}{\partial \overline{z}} = \frac{\partial \overline{f}}{\partial \overline{z}} = \overline{\nabla_z f}$. Consequently we do not need to explicitly compute the gradient with respect to the conjugate of the poles and the coefficients. Equation (20) can be written making explicit use of the C-symmetric or the full CP-symmetric formulation of f, which is Equations (10) and (12) respectively.

C symmetry – Let us first derive f with respect to ν_k and β_k

$$\frac{\partial f}{\partial \nu_k} = \frac{\beta_k}{(t-\nu_k)^2} \quad ; \quad \frac{\partial f}{\partial \beta_k} = \frac{1}{(t-\nu_k)}.$$

The expression above is the k^{th}-component of the gradients $\nabla_\nu f$ and $\nabla_\beta f$ respectively. Plugging them in Equation (20) and separating terms, we arrive at the matrix expressions

$$\nabla_\nu F = 2\left[\beta^\dagger \nabla A + \beta^\top \nabla\overline{B} - \nabla\zeta^\dagger\right] I_\beta \tag{21}$$

$$\nabla_\beta F = 2\left[\beta^\dagger A + \beta^\top \overline{B} - \zeta^\dagger\right] \tag{22}$$

where A, $\overline{B}$, and ζ are the same quantities defined in section 3.2, while remaining matrices are defined for $i, j = 1, \ldots, p$:

$$\nabla A_{i,j} = \int_{-\infty}^{+\infty} \frac{\mathfrak{G}(t)}{(t-\overline{\nu}_i)(t-\nu_j)^2}\, dt \qquad \nabla\zeta_i = \int_{-\infty}^{+\infty} \frac{\mathfrak{G}(t)h(t)}{(t-\overline{\nu}_i)^2}\, dt$$

$$\nabla\overline{B}_{i,j} = \int_{-\infty}^{+\infty} \frac{\mathfrak{G}(t)}{(t-\nu_i)(t-\nu_j)^2}\, dt \qquad I_\beta = \mathrm{diag}\,(\beta)$$

CP symmetry – Analogously to what is done in the C-symmetric case, we first compute the derivatives of f with respect to the poles and the coefficients

$$\begin{aligned}\frac{\partial f}{\partial w_k} &= \gamma_k\left[\frac{1}{(t-w_k)^2} + \frac{1}{(t+w_k)^2}\right]; \\ \frac{\partial f}{\partial \gamma_k} &= \frac{1}{(t-w_k)} - \frac{1}{(t+w_k)}.\end{aligned}$$

After entry-wise substitution of the above components of $\nabla_w F$ and $\nabla_\gamma F$, some tedious rearrangement, and using the parity with respect to the integration limits, we arrive at the following matrix equations

$$\nabla_w F = 4\left[\gamma^\dagger\left(\nabla X - \nabla Z\right) + \gamma^\top\left(\nabla\overline{W} - \nabla\overline{Y}\right) - \nabla\theta^\dagger\right] I_\gamma \tag{23}$$

$$\nabla_\gamma F = 4\left[\gamma^\dagger\left(X - Z-\right) + \gamma^\top\left(\overline{W} - \overline{Y}-\right) - \theta^\dagger\right]. \tag{24}$$

The matrices not previously introduced are defined as follows. For $k, \ell = 1, \ldots, q$:

$$\begin{aligned}\nabla X_{k,\ell} &= \int_{-\infty}^{+\infty} \frac{\mathfrak{G}(t)}{(t-\overline{w}_k)(t-w_\ell)^2}\, dt \\ \nabla Z_{k,\ell} &= -\int_{-\infty}^{+\infty} \frac{\mathfrak{G}(t)}{(t-\overline{w}_k)(t+w_\ell)^2}\, dt \\ \nabla\overline{W}_{k,\ell} &= \int_{-\infty}^{+\infty} \frac{\mathfrak{G}(t)}{(t-w_k)(t-w_\ell)^2}\, dt \\ \nabla\overline{Y}_{k,\ell} &= -\int_{-\infty}^{+\infty} \frac{\mathfrak{G}(t)}{(t-w_k)(t+w_\ell)^2}\, dt \\ \nabla\overline{\theta}_k &= \int_{-\infty}^{+\infty} \frac{\mathfrak{G}(t)h(t)}{(t-w_k)^2}\, dt \qquad I_\gamma = \mathrm{diag}\,(\gamma)\end{aligned}$$

Notice that, in both C-symmetric and CP-symmetric cases, the gradients are row vectors. While this is an arbitrary choice, it is a quite natural one to make. Moreover we decided to maintain the overall multiplicative factor in front of them; such a factor was scaled out of F when writing this in a symmetry transparent form. Once again this is an arbitrary choice since re-scaling the gradient of F does not influence the minimization process.

4. OPTIMIZING RATIONAL FILTERS FOR HERMITIAN OPERATORS

Our goal is to minimize the residual level function $F(x)$ of Equation (19) where we defined $x = (w, \gamma)$ for sake of conciseness. There are a number of possible approaches to the minimization problem. Starting at a given position $x^{(0)}$, *descent methods* minimize $F(x)$ by iteratively refining this position, such that $F(x^{(k+1)}) \leq F(x^{(k)})$. Using the CP symmetries the minimization problem can be stated as

$$\min_x\ F(x) = \min_x\ ||h - f(x)||_2^2, \tag{25}$$

with the initial choice of parameters, or *starting position*, of the form $x^{(0)} = [w_1^{(0)}, ..., w_q^{(0)}, \gamma_1^{(0)}, ..., \gamma_q^{(0)}]$.

In section 4.1, we use the the simple gradient-descent method as a first example of minimization of Equation (25). Section 4.2 illustrates the more sophisticated Levenberg-Marquardt method, which is more appropriate for non-convex Least-Squares problems. Depending on the starting position, descent methods, including the Levenberg-Marquardt method, may produce results that are not globally optimal. Section 4.3 deals with the choice of the starting position in a way that mitigates this problem. Section 4.4 delves into constrained optimization where it is advantageous to optimize more than just the squared differences between the filter and the indicator function $h(t)$. Section 4.5 concludes with a brief introduction of the software package specifically developed in the Julia programming language for this research project.

4.1. Optimization via Gradient Descent

In this section we remark, through the most basic descent method—*gradient descent*—on a number of issues of the

TABLE 1 | Poles, coefficients, and residual levels of the filter functions show in **Figure 5**.

	Starting position 1	Starting position 2	Starting position 3
w_1	$+0.997360 + 0.044537i$	$+0.721876 + 0.550662i$	$+0.742834 + 0.246735i$
w_2	$+0.721876 + 0.550662i$	$+0.997360 + 0.044537i$	$+0.744336 + 0.212182i$
γ_1	$-0.024019 - 0.002516i$	$-0.135117 - 0.148326i$	$-0.791429 - 0.331455i$
γ_2	$-0.135117 - 0.148326i$	$-0.024019 - 0.002516i$	$+0.639334 + 0.196414i$
$F([w\ \gamma])$	0.005846	0.005846	0.033640

optimization process. A gradient descent step drives down the residual level function, at the current position, along a direction colinear with its negative gradient. Given some mild assumptions, one eventually arrives at a local minimum. For a given starting position $x^{(0)}$ the update step of the gradient descent method is

$$x^{(k+1)} = x^{(k)} + s \cdot \Delta x^{(k)} = x^{(k)} - s \cdot \nabla_x F(x)\big|_{x=x^{(k)}}\ ,\ s \geq 0. \quad (26)$$

There are a number of ways to select the *step-length* s in Equation (26). While it is possible to set s as constant, we can improve the convergence of the method by approximating $s^{(k)} = \text{argmin}_{1\geq u\geq 0} F(x^{(k)} + u \cdot \Delta x^{(k)})$ at each iteration. To this end, we use a backtracking line search [22]: s is initialized at each step with $s = 1$ and then s is halved until $F(x + \Delta x) < F(x) + \frac{s}{2}\nabla_x F(x)\Delta x$.

Remark 1 (Slow convergence): Gradient descent is a fairly slow method, with linear convergence behavior at best. Even when equipped with a line-search the number of iterations required to reach a minimum is often in the millions.

In **Table 1**, we present three filters obtained using the gradient descent method. The real and imaginary part of the poles w and coefficients γ of the starting positions were chosen uniformly random between $[0, 1]$[5]. For each filter function $q = 2$, and so the filters have a total of 8 poles and coefficients. Starting Position 1 and 2 end in the same filter. Starting Position 3 generates a different filter with a residual level that is one order of magnitude higher than the others. The three filters are plotted on semi-log scale and on a normal scale on the right hand side and the left hand side of **Figure 5**, respectively. As the semi-log plot shows, this last filter will likely perform significantly worse than the first two.

Remark 2 (Dependence on starting position): The filter obtained from the optimization depends on the starting position $x^{(0)}$. Different starting positions can result in very different filters with different residual levels. Our experiments indicate that when increasing q the optimization finds more and more local minima, and their residual levels compared to the best known solutions get worse.

[5]The Least-Squares weights used to obtain these results are 1.0 inside $[-1000, 1000]$ and 0.0 everywhere else.

4.2. Improving Convergence Speed

We address Remark 1 by introducing the Levenberg-Marquardt method [23, 24] (LM), which is a solver for non-linear Least-Squares problems with a better convergence rate than gradient descent. LM is a hybrid of the gradient descent and the Gauss-Newton method (a detailed description of both methods can be found in Madsen et al. [25]). We present the Gauss-Newton method first, then introduce the LM method. We give a formulation based on the inner product presented in Equation (15).

For the sake of clarity, we re-write the residual level function in terms of $\xi([\alpha\ z], t) = h(t) - f(t, z, \alpha)$

$$\begin{aligned} F(x) &= ||\xi(x)||_2^2 = \langle \xi(x), \xi(x)\rangle = \int_{-\infty}^{\infty} \mathfrak{G}(t)|\xi(x,t)|^2 dt \\ &= \int_{-\infty}^{\infty} \mathfrak{G}(t)|h(t) - f(t,z,\alpha)|^2 dt, \end{aligned}$$

where we indicate the collection of parameters $[z\ \alpha]$ with x. The basis of the Gauss-Newton method is a linear approximation of ξ

$$\xi(x + \Delta x) \approx \xi(x) + \nabla\xi(x) \cdot \Delta x \doteq \xi + \sum_i \frac{\partial \xi}{\partial x_i}\Delta x_i \quad (27)$$

which, in the following, we refer to with the shortcut notation $\xi + \nabla\xi \cdot \Delta x$. The Levenberg-Marquardt method aims at minimizing a linear approximation of the residual level function by Gauss-Newton iterates. By using the linear approximation of ξ, such a requirement can be formulated as the minimization of the following function with respect to Δx

$$\begin{aligned} F(x + \Delta x) = ||\xi(x + \Delta x)||_2^2 &\approx \langle \xi, \xi\rangle + 2\langle \xi, \nabla\xi \cdot \Delta x\rangle \\ &\quad + \langle \nabla\xi \cdot \Delta x, \nabla\xi \cdot \Delta x\rangle \\ &= F(x) + \nabla F(x) \cdot \Delta x + (\Delta x)^\dagger \cdot \langle \nabla\xi, \nabla\xi\rangle \cdot \Delta x. \end{aligned}$$

Notice that the condition $\overline{\xi}(x + \Delta x) = \xi(x + \Delta x)$ implies that $(\Delta x)^\dagger \cdot \overline{\nabla\xi} = \nabla\xi \cdot \Delta x$ with the consequence that $\langle \nabla\xi \cdot \Delta x, \xi\rangle = \langle \xi, \nabla\xi \cdot \Delta x\rangle$. Taking the partial derivative of $F(x + \Delta x)$ with respect to Δx and equating it to zero one gets

$$\partial_{\Delta x} F(x + \Delta x) = 2\left[\langle \xi, \nabla\xi\rangle + \langle \nabla\xi, \nabla\xi\rangle \cdot \Delta x\right] = 0$$

Using the formulation above, the Gauss-Newton method iterates over k in the following way:

$$\begin{aligned} &1.\ \text{Set:} && H := \langle \nabla\xi(x^{(k)}), \nabla\xi(x^{(k)})\rangle \\ &2.\ \text{Solve:} && H \cdot \Delta x_{GN}^{(k)} = \langle \xi(x^{(k)}), \nabla\xi(x^{(k)})\rangle = -\frac{1}{2}\nabla F(x^{(k)}) \\ &3.\ \text{Update:} && x^{(k+1)} = x^{(k)} + s \cdot \Delta x_{GN}^{(k)}. \end{aligned}$$

Here H acts as a linear approximation of the Hessian of F. Consequently, if H is well-conditioned, the Gauss-Newton method can have quadratic convergence. However, there are still scenarios where Gauss-Newton steps provide only a small improvement of the residual. In those cases it is beneficial to temporarily switch to gradient descent, even though it only has

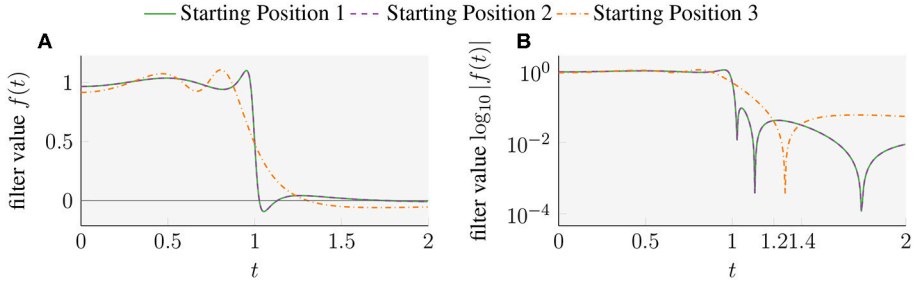

FIGURE 5 | (A) Three SLiSe filters obtained from gradient descent with different starting positions. **(B)** Semi-log plot with absolute value of the filter functions. The poles and residuals of the functions are listed in **Table 1**.

linear convergence at best. The Levenberg-Marquardt method employs a *dampening parameter* $\mu \geq 0$ to switch between the Gauss-Newton and gradient descent methods. By adding a dampening term, the second step is substituted by

$$2.\ \text{Solve:} \quad (H + \mu I)\Delta x_{LM}^{(k)} = -\frac{1}{2}\nabla F(x^{(k)}), \tag{28}$$

where the gradient of F is intended over both the poles and coefficients. Such an addition is equivalent to a constrained Gauss-Newton with the parameter μ working as a Lagrange multiplier.

In practice, the dampening parameter μ is re-adjusted after every iteration. For a small μ, the solution to Equation (28) is similar to the Gauss-Newton update step $\Delta x_{LM} \simeq \Delta x_{GN}$. On the other hand, when μ is large then $\Delta x_{LM} \simeq -\frac{1}{\mu}\nabla F(x)$, which is similar to the gradient descent update step. Often, instead of $H + \mu I$ one uses $H + \mu \cdot \text{diag}(H)$. Generally, when a LM-step reduces the residual by a large amount, then μ is decreased. When a step does not decrease the residual, or does not reduce the residual by enough, μ is increased. So for large μ the resulting update Δx_{LM} becomes increasingly similar to the gradient descent update.

Since the formulations of the LM method given so far is independent from the symmetry of f, it is valid for any formulation of the rational filter. Let us look now more in detail at H. This is a matrix whose entries are $H_{i,j} = \langle \nabla_{x_i} f, \nabla_{x_j} f \rangle$ which, in the case of CP symmetry, can be represented in block form as

$$H = \begin{pmatrix} H_2 & -\overline{H}_1 & \overline{H}_1 & -H_2 \\ -H_1 & H_2^\dagger & -H_2^\dagger & H_1 \\ H_1 & -H_2^\dagger & H_2^\dagger & -H_1 \\ -H_2 & \overline{H}_1 & -\overline{H}_1 & H_2 \end{pmatrix}. \tag{29}$$

Only two of the matrix blocks composing H are independent and are made of the following sub-blocks

$$\begin{aligned} H_1 &= \begin{pmatrix} \langle \nabla_w f, \nabla_w f \rangle & \langle \nabla_w f, \nabla_\gamma f \rangle \\ \langle \nabla_\gamma f, \nabla_w f \rangle & \langle \nabla_\gamma f, \nabla_\gamma f \rangle \end{pmatrix} \\ &= 2\begin{pmatrix} I_\gamma \left[\nabla\nabla\overline{W} - \nabla\nabla\overline{Y}\right] I_\gamma & I_\gamma \left[\nabla\overline{W} - \nabla\overline{Y}\right]^\top \\ \left[\nabla\overline{W} - \nabla\overline{Y}\right] I_\gamma & \overline{W} + \overline{Y} \end{pmatrix} \end{aligned} \tag{30}$$

$$\begin{aligned} H_2 &= \begin{pmatrix} \langle \nabla_{\overline{w}} f, \nabla_w f \rangle & \langle \nabla_{\overline{w}} f, \nabla_\gamma f \rangle \\ \langle \nabla_{\overline{\gamma}} f, \nabla_w f \rangle & \langle \nabla_{\overline{\gamma}} f, \nabla_\gamma f \rangle \end{pmatrix} \\ &= 2\begin{pmatrix} I_\gamma^\dagger \left[\overline{\nabla}\nabla X - \overline{\nabla}\nabla Z\right] I_\gamma & I_\gamma^\dagger \left[\nabla X - \nabla Z\right]^\dagger \\ \left[\nabla X - \nabla Z\right] I_\gamma & X - Z \end{pmatrix} \end{aligned} \tag{31}$$

where the only additional matrices that need to be defined are

$$\begin{aligned} \nabla\nabla\overline{W}_{i,j} &= \int_{-\infty}^{+\infty} \frac{\mathfrak{G}(t)}{(t - w_i)^2(t - w_j)^2}\, dt \\ \nabla\nabla\overline{Y}_{i,j} &= -\int_{-\infty}^{+\infty} \frac{\mathfrak{G}(t)}{(t - w_i)^2(t + w_j)^2}\, dt \\ \overline{\nabla}\nabla X_{i,j} &= \int_{-\infty}^{+\infty} \frac{\mathfrak{G}(t)}{(t - \overline{w}_i)^2(t - w_j)^2}\, dt \\ \overline{\nabla}\nabla Z_{i,j} &= -\int_{-\infty}^{+\infty} \frac{\mathfrak{G}(t)}{(t - \overline{w}_i)^2(t + w_j)^2}\, dt. \end{aligned}$$

The factor of 2 in front of the block matrices H_1 and H_2 comes from the P symmetry in combination with the evenness of the integral boundaries (i.e. $X = X^{\mathbb{T}}$, $\nabla X = \nabla X^{\mathbb{T}}$, etc.). Notice that block rows 3 and 4 of Equation (29) are exactly equivalent, up to a sign, to block rows 2 and 1 respectively. Similarly, block columns 3 and 4 are proportional to block columns 2 and 1. This is expected due to the CP symmetry. In addition, H_1 is complex symmetric ($H_1^\dagger = \overline{H}_1$), and H_2 is complex Hermitian ($H_2^\dagger = H_2$) which, implicitly, verifies that H is complex Hermitian itself.

Going back to the Equation (28), one can also write the vector Δx_{LM} in block notation,

$$(\Delta x_{LM})^\top = [(\Delta y)^\top \ -(\Delta y)^\dagger \quad (\Delta y)^\dagger \ -(\Delta y)^\top]$$

with $(\Delta y)^\top = [\Delta w^\top \Delta \gamma^\top]$. Writing Equation (28) in terms of Δy with $\mu \cdot \text{diag}(H)$ in place of μI one obtains 4 separate equations. Thanks to the symmetries of H, it is possible to reduce them to a single equation solving for Δy. There are several different ways to achieve this result. Starting from the third equation, one can extract $(H_1 \Delta y)$ as a function of $H_2^\dagger$, $\Delta\overline{y}$, and $\nabla_{\overline{y}} F$, then take the conjugate of it, and substitute it in the fourth equation. The final result is a linear system whose dimension is one fourth of the original size of H

$$\left[i\,\Im m\,(H_2) - \mu\ \text{diag}\,(H_2)\right]\Delta y = \frac{1}{2}\nabla_y F. \tag{32}$$

The Levemberg-Marquardt method is considerably faster than gradient descent. Let us illustrate their performance difference with a short example. The optimization process for γ-SLiSe carried on with Levemberg-Marquardt (see section 2.2) requires $8.2 \cdot 10^5$ iterations and about 711 s on an Intel i7-5600U CPU. On the other hand, gradient descent needs $1.0 \cdot 10^7$ iterations and about 3.5 h for the same optimization on the same hardware. Comparing the performance of both approaches using only number of iterations can be misleading. Each optimization method performs a very different amount of work per iteration: LM assembles and solves a linear system, while gradient descent performs a line-search and thus evaluates the residual level function many times. Nevetheless, by providing iteration counts and timings there can be no doubt that the LM method is significantly faster than gradient descent. Such result is obtained thanks to the improved convergence of the LM method together with the reduced size of the equation to be solved.

4.3. Systematic Choice of Starting Position

In general, random choices for w and γ do not produce good filters. In **Figure 5** we have seen an example of multiple local minima of the residual level function F with $q = 2$. To realize why the starting positions are crucial for the optimization method let us consider what happens when increasing the number of poles and coefficients q in one quadrant of $\mathbb{C}$. [6] Our extensive numerical experiments indicate that higher q values not only increase the number of local minima in F, but have the effect of worsening the residual levels of most of these additional local minima w.r.t. the best known local minimum. In one of our numerical experiments, using a random starting position with $q = 8$ yields a filter with a residual of 5.4×10^{-4}, while with $q = 4$ we achieved a residual level of 2.4×10^{-4}. Choosing $q = 8$ requires 16 linear system solves for each filter application, whereas $q = 4$ requires only 8 solves. To offset the additional cost of the linear system solves we want a filter that has a lower residual level, thus likely requiring fewer subspace iterations. Without a good starting position we obtain a filter with $q = 8$ that requires *more* linear system solves and *more* iterations. What is more, the optimization for a filter with $q = 8$ is significantly more expensive than for $q = 4$. Simply put, random starting positions tend to yield worse and worse filters for larger q.

We can reinterpret the problem of finding good starting positions for the poles and coefficients as finding a filter that serves as a good initial guess for an optimized filter. To this purpose we could use contour filters expressed through a numerical implementation of Cauchy's Residue Theorem (see section 2.1), namely the Gauss and Trapezoidal filters, or more specialized ones, like the Elliptic filter, that have already been proposed as substitutes for the contour solvers [16].

Figure 6 shows the residual levels of different filters for different q. The residual levels of the Gauss and Elliptic filters are indicated with blue and red triangles respectively. Conversely, shown with blue and red circles are the residual level of the filters that are obtained when the initial position of the SLiSe optimization is started with the corresponding Gauss and Elliptic filter poles and coefficients. The SLiSe filters have significantly lower residual levels than the Gauss and Elliptic filters. Using poles and coefficients from Gauss and Elliptic filters as starting positions for a $q < 10$ results in the same unconstrained SLiSe filter, although the Gauss and Elliptic are quite different. While we cannot claim optimality for the SLiSe filters so obtained, but we have extensively explored the space of initial conditions and found no evidence of better filters obtained starting from different positions. In our experience existing filters, and the Elliptic filter in particular, make for excellent starting positions practically eliminating the problem of finding an appropriate starting position.

[6] For some spectrum slicing methods, such as the DD-PP projection [26] which does not iterate the subspace, it is essential to be able to choose large q.

4.4. Constrained Optimization

In this section we discuss two filter properties that can be achieved by implementing constraints on the optimization. First, we introduce a penalty term on the gradient of the filter function, which results in filters of varying "steepness" and with varying amounts of "overshooting." Given our standard search interval, $[-1, 1]$, we call the derivative of the filter $f(t)$ at the endpoints of the interval $\nabla_t f(t)|_{t=1} = -\nabla_t f(t)|_{t=-1}$ the *steepness* of the filter. The steepness—or separation factor—of a filter has previously been suggested as a quality measure in the context of optimizing exclusively the coefficients of the rational function [18]. A steep filter is considered desirable because it dampens eigenvalues outside the search interval even when the eigenvalues are close to the interval endpoints and, in doing so, improves convergence. When optimizing the poles of the rational functions together with its coefficients the steepness alone does not serve as a good measure of quality anymore. It is possible for a filter to be steep, and yet not to dampen eigenvalues outside of the search interval sufficiently well. The lack of dampening is due to an effect called "overshooting," which is a known behavior when approximating discontinuous functions with continuous ones. Usually a very steep filter tends to overshoot at the endpoint of $[-1, 1]$, and conversely filters that do not overshoot are not particularly steep. By adding a *penalty term* to the residual level function and tuning the value of the parameter c, we disadvantage those features that make for ineffective filters

$$\min_{w_i,\gamma_i \; 1\le i\le q} ||h - f(w,\gamma)||_2^2 + \frac{c}{2} \cdot \left[\nabla_t f(t)|_{t=1} - \nabla_t f(t)|_{t=-1}\right]. \tag{33}$$

Note that since $\nabla_t f(1)$ is supposed to be negative, a positive c will make for a steeper filter when minimizing the residual.

A different kind of constraint, namely a box constraints on the imaginary part of the poles, may be advantageous when Krylov solvers are used for the linear system solves as they appear in Equation (2). In such cases, the condition number of the shifted matrix $M - w_i I$ may become large when $|\Im m(w_i)|$ is small, which can occur for large degrees of the Elliptic and SLiSe filters. One possible way to circumvent this problem is to ensure that the absolute value of the imaginary part for each pole is large enough. For SLiSe filters this translates into a box constraint d

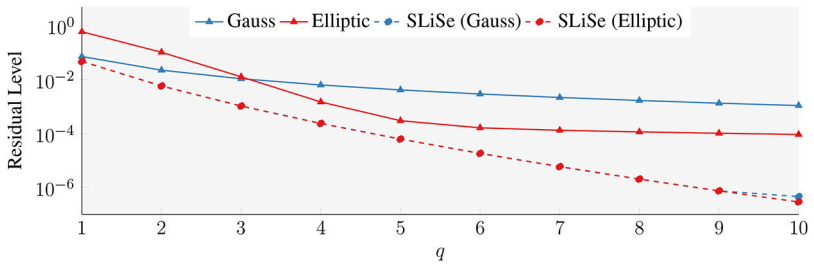

FIGURE 6 | Residual levels of Gauss and Elliptic filters, as well as, SLiSe filters obtained from Gauss and Elliptic filters as starting positions.

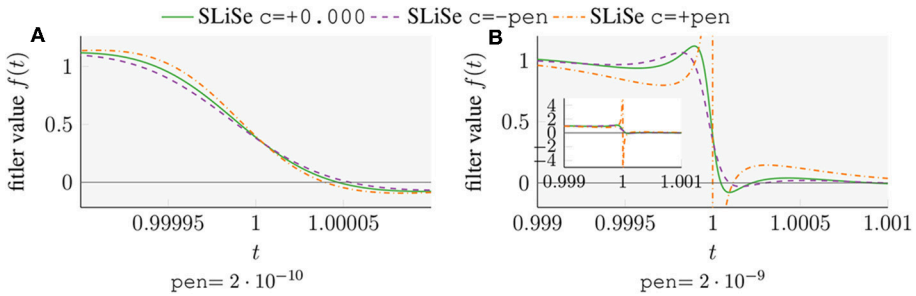

FIGURE 7 | Examples of filters obtained via penalty terms. **(A)** An appropriately sized penalty parameter yields filters of different steepness. **(B)** Too large a (positive) penalty parameter results in a pole on the real axis.

on the parameters:

$$\min_{w_i,\gamma_i\ 1\le i\le q} ||h(t) - f(t,w,\gamma)||_2^2 \quad \text{s.t.} \quad \forall i : |\Im m\{w_i\}| \ge d\,. \quad (34)$$

Both approaches to constrained optimization presented in this section require some tuning in the selection of the parameters. This lack of automatism is the exception in our work. Constrained optimization without the need for extensive parameter selection is an area of future work. In section 4.4.1, we discuss how to adjust the optimization to generate filters of varying steepness, while section 4.4.2 describes the box constraints approach.

4.4.1. Steepness and Overshooting via Penalty Term

From a visual inspection of **Figure 3A** it is evident how the Gauss filter starts to drop significantly earlier than the endpoint of the search interval at $t = 1$, and it is significantly less steep than the γ-SLiSe filter. Conversely, the latter maintains a value close to one inside the search interval closer to the endpoint, and it takes small values just outside the search interval. The increased steepness of the γ-SLiSe filter comes at the cost of overshooting: the SLiSe filter shoot up above 1 and drops below 0 just inside and outside the search interval respectively. The surge above 1 is not a problem, however the overshooting outside $[-1, 1]$ is an undesirable effect since it maps eigenvalues to relatively large negative—and thus large in magnitude—values outside the endpoints, hurting convergence.

While it is possible to increase the steepness of SLiSe filters via the penalty term, there is little benefit in even steeper filters with more overshooting. **Figure 7A** shows an illustrative example of filters obtained with an appropriately sized penalty parameter and number of poles per quadrant $q = 4$. **Table 2** shows the residuals without penalty term and the steepness of the filter $\nabla_t f(1)$ for the resulting filters. A negative penalty term tends to decrease steepness and overshooting while a positive c increases them. Since the penalty parameter is chosen significantly smaller than the residual, the change to the filter is not very pronounced. Choosing a larger penalty parameter may have negative consequences and requires great care, especially when c is chosen to be positive. **Figure 7B** shows a filter generated with a larger value for c, resulting in a very steep slope. Such filter has a pole on the real axis that causes significant overshooting which substantially hurts convergence. Additionally, a real pole affects the corresponding linear system, which can become seriously ill-conditioned causing failure or slow converge of the iterative solver. While a large and positive parameter is problematic because of the risk of a real pole, this is usually not a concern in practical applications. Often, it is more important

TABLE 2 | Residual levels and steepness of the filter functions shown in **Figure 7A**.

Penalty parameter c	Residual	Steepness
0	2.265829×10^{-5}	-1.211081×10^{4}
$-2 \cdot 10^{-10}$	2.279018×10^{-5}	-1.063877×10^{4}
$+2 \cdot 10^{-10}$	2.288353×10^{-5}	-1.429542×10^{4}

to limit the overshooting than it is to make the already steep SLiSe filters even steeper. **Figure 7B** shows that a negative penalty parameter c reduces overshooting. η-SLiSe was obtained using a negative penalty parameter, which we discuss further in section 5.

4.4.2. Large Imaginary Parts via Box Constraints

Implementing box constraints in our framework is straightforward: Instead of the old update step in Equation (26) we use gradient projection by updating as follows

$$x^{(k+1)} = \mathcal{P}(x^{(k)} - s \cdot \nabla_x F(x)\big|_{x=x^{(k)}}) \ , \ s \geq 0, \tag{35}$$

where $\mathcal{P}(x)$ projects x into the constraints. In practice we project onto the constrained value only when the constraint is violated. We implement this scheme by forcing the absolute value of the imaginary part of every pole to be equal or larger than some value `lb`. For a single pole w_l the projection is

$$\mathcal{P}(w_l) = \begin{cases} \Re e(w_l) + i \cdot \text{sgn}(\Im m(w_l)) \cdot \texttt{lb} & |\Im m(w_l)| < \texttt{lb} \\ \Re e(w_l) + i \cdot \Im m(w_l) & \text{otherwise} \ . \end{cases}$$

In the following, we compare filters with different box constraints. **Figure 8** shows four filters each with $q = 4$. Shown are the Elliptic filter, a SLiSe filter obtained via unconstrained optimization[7], and two additional SLiSe filters obtained with different `lb` constraints. For the Elliptic filter $\min_{1 \leq i \leq q} |\Im m(w_i)|$ is about 0.0022, while it is 0.001 for the unconstrained SLiSe filter. In the case of the constrained SLiSe filters, we consider "SLiSe (`lb=0.0022`)" filter, with a `lb` corresponding the the Elliptic filter. Additionally, we also examine "SLiSe (`lb=0.0016`)" generated with a constraint of 0.0016, about half way between the Elliptic and the unconstrained SLiSe filters. We calculate the condition number for each of the shifted linear systems that result from an application of the filter to each of the benchmark problems illustrated in section 2.

Figure 8 is a performance profile of the largest condition number for each of the benchmark problems. The Elliptic filter and the SLiSe (`lb=0.0022`) perform identically in the performance profile. Both filters have a value of one over the entire abscissa range, which indicates that the condition numbers are the same. The figure implies that the largest condition number is influenced solely by the smallest absolute imaginary value of the poles. Accordingly, the constrained filter with `lb` of 0.0016 performs worse, and the unconstrained SLiSe filter is the worst. This effect can be understood by realizing that the shifted matrices $M - Iw_l$ become nearly singular only if w_l is near an eigenvalue of M. An increase in the `lb` constraint has a large influence on the filter. As a result, caution is required when optimizing with box constraints.

4.5. The SliSeFilters Software Package

The optimization methodology described in section 4 is implemented in a software package written in the Julia programming language and can be accessed by the interested reader in a publicly available repository named SliSeFilters[8]. The code is fairly straithforward and aims at a fire-and-forget approach, where the target is to obtain the desired results without parameter space exploration. There is one set of parameters that have not been discussed but nonetheless can influence the outcome of the optimization procedure: the weight functions $\mathfrak{G}(t)$. While the automatic selections of $\mathfrak{G}(t)$ is the topic of a forthcoming publication, here we briefly remark on the adoption of three simple guidelines for the choice of the weight functions. Each guideline is a heuristic that can be easily implemented as part of the optimization algorithm, or checked manually.

Guideline 1 (Outside): Slowly taper off the weights in a large enough neighborhood outside the search interval. In order to avoid trial-and-error, it is advisable to detect increasing local maxima of the absolute value of the rational function during the optimization process and increase the weights where required.

Guideline 2 (Endpoints): The weight function should be selected so as to be symmetric around the endpoints of the search interval. Accordingly, during the optimization process, choose $\mathfrak{G}(\pm 1 - \epsilon) = \mathfrak{G}(\pm 1 + \epsilon)$ for $\epsilon \leq 0.2$. Additionally, if desired, scale the resulting filter to $f(\pm 1) = 0.5$.

Guideline 3 (Inside): The weights in the entire search interval should be chosen large enough to prevent large oscillations. During the optimization process, it is advisable to monitor the difference between minima and maxima of the filter inside $[-1, 1]$ and adapt weights accordingly.

5. A RICH VARIETY OF FILTERS: PRACTICE AND EXPERIENCE

In the previous section we discussed a number of techniques to influence the optimization procedure. Some of the filters resulting from this procedure are already presented in section 2. In this section we illustrate the effects of the constrained optimization described in section 4.4 and highlight its potential. First, we discuss η-SLiSe, where a penalty parameter is used to limit overshooting. Second, we present a filter that uses box constraints to achieve better condition numbers for the shifted matrices that arise from the filter application. In these examples the focus is on illustrating the power of the SLiSe filters by showing the improvement over the Elliptic filter. All the experimental tests are executed on the same benchmark set described and used in section 2. For experimental tests using benchmark sets generated from different eigenvalue problems, we refer the reader to the Jupyter notebook which is part of the freely available test suite[9].

5.1. Penalty Parameter: η-SLiSe

We have seen η-SLiSe in section 2.3, where we compare its convergence rate to the Elliptic and Gauss filters. η-SLiSe was obtained via a negative penalty parameter, to limit the

[7]The weights are available in **Appendix C** under "Box-SLiSe."
[8]https://github.com/SimLabQuantumMaterials/SLiSeFilters.jl
[9]https://github.com/SimLabQuantumMaterials/Spectrum SlicingTestSuite.jl.

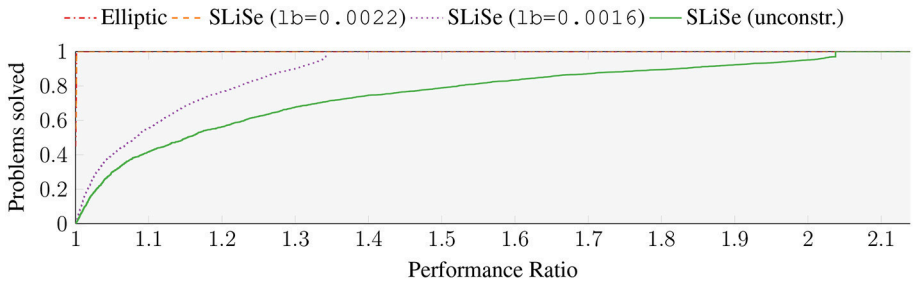

FIGURE 8 | Performance profile comparing the worst condition number for the linear systems arising from filter application to 2116 problems obtained from the "Si2" matrix. The performance ratio for the Elliptic and SLiSe (`lb=0.0022`) filters are one over the entire abscissa range.

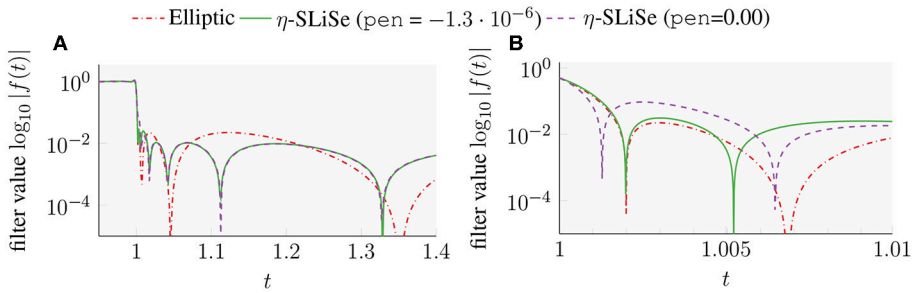

FIGURE 9 | Plots of the Elliptic filter, η-SLiSe, and the η-SLiSe filter without penalty term. All filters have 16 poles and coefficients. **(A)** Semi-log plot. **(B)** Semi-log plot. Abscissa limited to [1, 1.01].

overshooting of the filter. Without the penalty term the filter would perform significantly worse. **Figure 9** shows the Elliptic and η-SLiSe filter, and the unconstrained version of η-SLiSe. **Figure 9A** shows an abscissa range of $[0.95, 1.4]$. At this scale both SLiSe filters look identical and oscillate less than the Elliptic filter, which we would expect from the rate of convergence for η-SLiSe. **Figure 9B** shows the same filters for an abscissa range of $[1, 1.01]$, just outside the search interval. At this scale, the two SLiSe filters are very different. The SLiSe filter without penalty term is steeper and overshoots more than the Elliptic filter, which results in larger absolute function values inside $[1.001, 1.006]$. The penalty parameter for η-SLiSe was chosen large enough to make the filter about as steep as the Elliptic filter. As a result, η-SLiSe overshoots less than the unconstrained filter and oscillates at a magnitude only slightly larger than the Elliptic filter. Compared to the Elliptic filter, η-SLiSe trades-off slightly larger (absolute) function values near the end of the search interval for much smaller values farther away.

The value of the penalty parameter is chosen to be of large absolute value, as compared to the values in section 4.4.1. Such a large penalty parameter results in a significant reduction in the steepness and overshooting of the filter. The difference between the SLiSe filters with and without penalty term is large near the end of the search interval, but negligible in $[1.05, \infty]$. In "designing" such a filter we first chose the weights such that the filter has the desired behavior for most of the t axis, e.g., $[0, 1] \cup [1.05, \infty]$ and overshoots slightly more than desired. Then, we used a negative penalty term to lessen the overshooting; this step usually requires only very few iterations.

5.2. Box Constraint: κ-SLiSe

Figure 10A shows a performance profile of the Gauss, Elliptic, and ζ-SLiSe filter that compares the rates of convergence for these filters. It appears that ζ-SLiSe would be a better replacement for the Elliptic filter than η-SLiSe, as shown in **Figure 4**. However, ζ-SLiSe filter does not conform to Guideline 1. The filter has a single local extremum that has a significantly larger absolute value than both of its neighboring extrema. This extremum is still smaller than the equi-oscillation of the Elliptic filter, which is the reason why ζ-SLiSe performs better than the Elliptic filter.

There is another problem with ζ-SLiSe. The filter has a pole with an absolute imaginary value of $\min_{1 \le i \le q} |\Im m(w_i)| \approx 0.001$. Analogously to the filters discussed in section 4.4.2, this small imaginary part deteriorates the convergence of Krylov-based linear system solvers. To mitigate this problem we can use box constraints to obtain a filter where each pole has an absolute imaginary value of at least 0.0022, the same value of the Elliptic filter. The κ-SLiSe, obtained with a box constraint of `lb`(=0.0022) and the same weight function used to obtain ζ-SLiSe. κ-SLiSe violates Guideline 1, just as ζ-SLiSe does.

Figure 10B shows the performance profile for κ-SLiSe against the Elliptic filter but without the ζ-SLiSe. The box constraint results in two changes of κ-SLiSe as compared to the ζ-SLiSe. First, the box constraint contributes to lessen the overshooting of the filter. As a result κ-SLiSe attains a better rate of convergence than the Elliptic filter for 86% of the benchmark problems, as compared to the 70% of ζ-SLiSe. Second, κ-SLiSe has larger (as

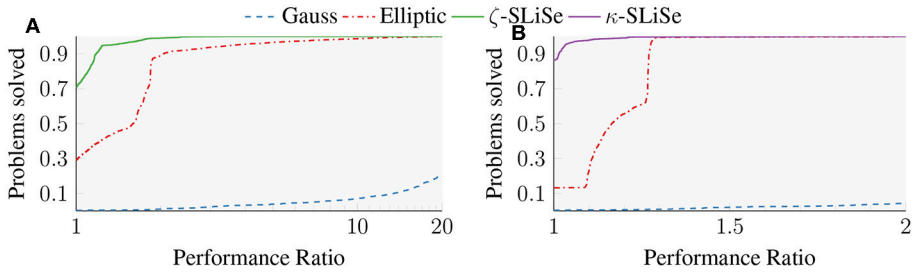

FIGURE 10 | Semi-log performance profile of the convergence rate for the Gauss, Elliptic, ζ-SLiSe, and κ-SLiSe filters. All filters have 16 poles and $\mathfrak{p} = \mathfrak{m}$. The last two filters violate Guideline 1. **(A)** Performance Profile of the Gauss, Elliptic, and ζ-SLiSe filters. **(B)** Performance Profile of the Gauss, Elliptic, and κ-SLiSe filters.

TABLE 3 | Overview of SLiSe-filters, the section in which each filter is introduced, and a short comment for each filter.

Filter	Section	Comment
γ-SLiSe	Section 2.2	Replacement for gauss filter
ζ-SLiSe	Section 5.1	Replacement for Elliptic filter; Violation of guideline 1
η-SLiSe	Section 2.3	ζ-SLiSe optimized with use of penalty parameter
κ-SLiSe	Section 5.2	ζ-SLiSe optimized with use of box constraints

compared to ζ-SLiSe) absolute function values for most of the t axis, which results in rates of convergence that are closer to the rates of the Elliptic filter. This behavior is reflected in **Figure 10B**, where above a performance ratio of 1.3 the two filters have exactly the same profile. κ-SLiSe illustrates that box constraints can be used to obtain SLiSe filters with good convergence rates. Nevertheless, larger box constraints usually tend to result in worse rates of convergence, as compared to an unconstrained filter. While κ-SLiSe is still marginally better than the Elliptic filter, it yields rates of convergence that are not nearly as good as ζ-SLiSe or η-SLiSe.

To close out this section, **Table 3** provides an overview of all filters we have presented in this work as well as their distinguishing features. γ-SLiSe, introduced in section 2.2, is our proposed replacement for the Gauss filter. As such is it well-suited as a default filter, when little is known about the spectral properties of the problem at hand. ζ-SLiSe is a filter that aims to replace the Elliptic filter, for small spectral gaps at the endpoints or for application where load-balacing is a priority. Earlier in this section we discussed ζ-SLiSe, a filter that violates Guideline 1: the optimization weights $\mathfrak{G}(t)$ taper off very quickly outside of the $[-1, 1]$ interval. η-SLiSe—introduced much earlier, in Sec. 2.3—improves on ζ-SLiSe by means of a penalty parameter that limits overshooting. Finally, we discussed κ-SLiSe, another variation of ζ-SLiSe for use with Krylov-based solver for the linear systems arising from the filter application.

6. CONCLUSIONS AND OUTLOOK

In this work we set up a weighted non-linear Least-Squares framework for the optimization of a class of rational filters. Our approach is non-linear in the sense that it optimizes both the pole placement and the coefficients of the rational filters, requiring the solution of a non-convex problem. Because of its non-convexity the iterative optimization process has to be carefully engineered so as to guarantee symmetry preservation and numerical stability. The implementation of such framework in a simple software package offers a rich set of solutions among which we provide with alternatives to the standard Gauss and Elliptic filters that are currently used in contour based eigensolvers.

First, we explicitly formulate the filters to be conjugation and parity invariant. Such a formulation stabilizes the target function and reduces the complexity of the optimization process by at least a factor of four. Because the iterative optimization depends on the initial value of its parameters, convergence to an optimal solution is not easily guaranteed. We show that a careful placement of the starting positions for the poles and the selection of an appropriate optimization algorithm lead to solutions with systematically smaller residuals as the degree of the filter is increased.

When used in combination with a spectrum slicing eigensolver, our optimized rational SLiSe filters can significantly improve the efficiency of contour based eigensolvers on large sets of spectral intervals for any given problem. In the specific instances of the standard Gauss and Elliptic filters, we show-case the flexibility of our approach by providing SLiSe replacements. This flexibility is achieved through the guided selection of Least-Squares weights and the use of constrained optimization. The optimization with box constraints addresses the issue of rational filters with poles very close to the real axis, which lead to almost singular matrix resolvents. Such constrained optimization decreases the condition number of the resolvents, which positively affects the solution of the corresponding linear system solves when Krylov based methods are employed.

Significant effort went into designing the optimization process and its implementation so as to be user-ready. The resulting software is written in the Julia programming language and is available in a public repository together with a test suite and further examples in addition to the one presented in this manuscript. Future work will focus on the automatic selection of the Least-Square weights as well as in more robust optimization approaches when it comes to constraints. The

ultimate goal is to provide filters that adapt to any given specific problem. Such filters would be generated on-the-fly and take advantage of spectral information, if cheaply available. Not only do problem-specific filters promise better convergence, but they can provide automatic load balancing between multiple contour "slices." The present work is meant as a significant step toward such a direction.

AUTHOR CONTRIBUTIONS

All authors listed have made a substantial, direct and intellectual contribution to the work, and approved it for publication.

ACKNOWLEDGMENTS

Financial support from the Jülich Aachen Research Alliance High Performance Computing and the Deutsche Forschungsgemeinschaft (DFG) through grant GSC 111 is gratefully acknowledged.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fams.2019.00005/full#supplementary-material

REFERENCES

1. Polizzi E. Density-matrix-based algorithm for solving eigenvalue problems. *Phys Rev B* (2009) **79**:115112. doi: 10.1103/PhysRevB.79.115112
2. Sakurai T, Sugiura H. A projection method for generalized eigenvalue problems using numerical integration. *J Comput Appl Math.* (2003) **159**:119–28. doi: 10.1016/S0377-0427(03)00565-X
3. Ikegami T, Sakurai T, Nagashima U. A filter diagonalization for generalized eigenvalue problems based on the Skaurai-Sugiuara projection method. *J Comput Appl Math.* (2010) **233**:1927–36. doi: 10.1016/j.cam.2009.09.029
4. Tang PTP, Polizzi E. FEAST as a subspace iteration eigensolver accelerated by approximate spectral projection. *SIAM J Matrix Anal Appl.* (2014) **35**:354–90. doi: 10.1137/13090866X
5. Bezanson J, Edelman A, Karpinski S, Shah VB. Julia: a fresh approach to numerical computing. *SIAM Rev.* (2017) **59**:65–98. doi: 10.1137/141000671
6. Ikegami T, Sakurai T. Contour integral eigensolver for non-hermitian systems: a Rayleigh-Ritz-type approach. *Taiwanese J Math.* (2010) **14**:825–37. doi: 10.11650/twjm/1500405869
7. Imakura A, Sakurai T. Block Krylov-type complex moment-based eigensolvers for solving generalized eigenvalue problems. *Numeric. Algor.* (2016) **75**:413–33. doi: 10.1007/s11075-016-0241-5
8. Beyn WJ. An integral method for solving nonlinear eigenvalue problems. *Linear Algebra Appl.* (2012) **436**:3839–63. doi: 10.1016/j.laa.2011.03.030
9. Imakura A, Du L, Sakurai T. Relationships among contour integral-based methods for solving generalized eigenvalue problems. *Japan J Indust Appl Math.* (2016) **33**:721–50. doi: 10.1007/s13160-016-0224-x
10. Murakami H. *Optimization of Bandpass Filters for Eigensolver*. Tokyo Metropolitan University (2010).
11. Murakami H. *Experiments of Filter Diagonalization Method for Real Symmetric Definite Generalized Eigenproblems by the Use of Elliptic Filters.* Tokyo Metropolitan University (2010).
12. Murakami H. *An Experiment of the Filter Diagonalization Method for the Banded Generalized Symmetric-Definite Eigenproblem.* Tokyo Metropolitan University (2007).
13. Murakami H. *The Filter Diagonalization Method for the Unsymmetric Matrix Eigenproblem.* Tokyo Metropolitan University (2008).
14. Murakami H. A filter diagonalization method by the linear combination of resolvents. *IPSJ Trans ACS* (2008) **49**:66–87.
15. Austin AP, Trefethen LN. Computing eigenvalues of real symmetric matrices with rational filters in real arithmetic. *SIAM J Sci Comput.* (2015) **37**:A1365–87. doi: 10.1137/140984129
16. Güttel S, Polizzi E, Tang PTP, Viaud G. Zolotarev quadrature rules and load balancing for the FEAST eigensolver. *SIAM J Sci Comput.* (2015) **37**:A2100–22. doi: 10.1137/140980090
17. Barel MV. Designing rational filter functions for solving eigenvalue problems by contour integration. *Linear Algebra Appl.* (2016) **502**:346–5. doi: 10.1016/j.laa.2015.05.029
18. Xi Y, Saad Y. Computing partial spectra with least-squares rational filters. *SIAM J Sci Comput.* (2016) **38**:A3020–45. doi: 10.1137/16M1061965
19. Saad Y, Li R, Xi Y, Vecharinsky E, Yang C. *EigenValues Slicing Library*. Available online at: http://www-users.cs.umn.edu/~saad/software/EVSL/ (Accessed November 05, 2016).
20. Davis TA, Hu Y. The University of Florida sparse matrix collection. *ACM Trans Math Softw.* (2011) **38**:1:1–1:25. doi: 10.1145/2049662.2049663
21. Dolan ED, Moré JJ. Benchmarking optimization software with performance profiles. *Math Program.* (2002) **91**:201–13. doi: 10.1007/s101070100263
22. Boyd S, Vandenberghe L. *Convex Optimization.* Cambridge: Cambridge University Press (2004).
23. Marquardt DW. An algorithm for least-squares estimation of nonlinear parameters. *J Soc Ind Appl Math.* (1963) **11**:431–41. doi: 10.1137/0111030
24. Levenberg K. A method for the solution of certain non-linear problems in least squares. *Q Appl Math.* (1944) **2**:164–8. doi: 10.1090/qam/10666
25. Madsen K, Nielsen HB, Tingleff O. *Methods for Non-Linear Least Squares Problems*. Technical University of Denmark (2004).
26. Kalantzis, V, Kestin, J, Polizzi E, Saad, Y. Domain decomposition approaches for accelerating contour integration eigenvalue solvers for symmetric eigenvalue problems. *Numer Linear Algebra Appl.* (2018) 25:e2154. doi: 10.1002/nla.2154

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.