SIAM J. SCI. COMPUT.
Vol. 38, No. 5, pp. A3020–A3045

COMPUTING PARTIAL SPECTRA WITH LEAST-SQUARES RATIONAL FILTERS*

YUANZHE XI† AND YOUSEF SAAD†

Abstract. We present a method for computing partial spectra of Hermitian matrices, based on a combination of subspace iteration with rational filtering. In contrast with classical rational filters derived from Cauchy integrals or from uniform approximations to a step function, we adopt a least-squares (LS) viewpoint for designing filters. One of the goals of the proposed approach is to build a filter that will lead to linear systems that are easier to solve by iterative methods. Among the main advantages of the proposed LS filters is their flexibility. Thus, we can place poles in more general locations than with the formulations mentioned above, and we can also repeat these poles a few times for better efficiency. This leads to a smaller number of required poles than in existing methods. As a consequence, factorization costs are reduced when direct solvers are used and the scheme is also beneficial for iterative solvers. The paper discusses iterative schemes to solve the linear systems resulting from the filtered subspace iteration that take advantage of the nature and structure of the proposed LS filters. The efficiency and robustness of the resulting method is illustrated with a few model problems as well as several Hamiltonian matrices from electronic structure calculations.

Key words. rational filters, polynomial filtering, subspace iteration, Cauchy integral formula, Krylov subspace methods, electronic structure

AMS subject classifications. 15A18, 65F10, 65F15, 65F50

DOI. 10.1137/16M1061965

1. Introduction.

This paper discusses a method to compute all eigenvalues inside a given interval $[a,\ b]$, along with their associated eigenvectors, of a large sparse Hermitian matrix A. This problem has important applications in computational physics, e.g., in electronic structure calculations. A common way in which these computations are often carried out is via a shift-and-invert strategy where a shift σ is selected inside $[a,\ b]$ and then Lanczos or subspace iteration is applied to $(A-\sigma I)^{-1}$. When the factorization is computable in a reasonable time, this approach can be quite effective and has been used in industrial packages such as NASTRAN [16]. Its biggest limitation is the factorization, which becomes prohibitive for large three-dimensional (3D) problems. If factoring A is too expensive, then one immediate thought would be to resort to iterative solvers. However, the systems that will be encountered are usually highly indefinite and sometimes even nearly singular because real shifts may be close to eigenvalues of A. Thus, iterative solvers will likely encounter difficulties.

A common alternative to standard shift-and-invert that has been studied extensively in the past is to use polynomial filtering [9, 34]. Here, the basic idea is to replace $(A-\sigma I)^{-1}$ with $p(A)$ where $p(t)$ is some polynomial designed to enhance the

*Submitted to the journal's Methods and Algorithms for Scientific Computing section February 17, 2016; accepted for publication (in revised form) July 25, 2016; published electronically September 27, 2016.

http://www.siam.org/journals/sisc/38-5/M106196.html

Funding: This work was supported by the "High-Performance Computing, and the Scientific Discovery through Advanced Computing (SciDAC) program" funded by U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research, and Basic Energy Sciences DE-SC0008877. The theoretical part of the contribution by the second author was supported by NSF grant CCF-1505970.

†Department of Computer Science & Engineering, University of Minnesota, Twin Cities, Minneapolis, MN 55455 (yxi@cs.umn.edu, saad@cs.umn.edu).

eigenvalues inside $[a,\ b]$ and dampen those outside this interval. However, there is a hidden catch which can make the method very unappealing in some cases. Consider a spectrum that is stretched on the high end, such as, for example, when $\lambda_i = (1-i/n)^3$, for $i = 1, 2, \ldots, n$ and n is large. Using polynomial filtering would be very ineffective if we want to compute eigenvalues somewhere in the middle or the lower end of the spectrum. This is because, relatively speaking, these eigenvalues will be very close to each other and we would need a very high degree polynomial to achieve the goal of separating wanted eigenvalues from unwanted ones. Another issue with polynomial filtering techniques is that they rely heavily on inexpensive matrix-vector (matvec) operations. The number of matvecs performed to reach convergence can be quite large, and the overall cost of the procedure may be unacceptably high when these are expensive. Moreover, polynomial filtering is rarely used to solve generalized eigenvalue problems since a linear system must be solved with one of the two matrices in the pair, at each step.

Rational filtering techniques offer competitive alternatives to polynomial filtering. Sakurai and his co-authors proposed a contour integral based eigensolver, called the Sakurai–Sugiura (SS) method [12, 27], to convert the original problem into a smaller Hankel matrix eigenvalue problem. Later, they stabilized their original method using a Rayleigh–Ritz procedure and this led to an improved technique known as CIRR [13, 21, 28]. Another well-known algorithm, developed by Polizzi and co-authors [14, 15, 23] and called FEAST, exploits a form of filtered subspace iteration [32]. Asakura and co-authors [1] and later Beyn [5] extended the idea of contour integrals to nonlinear eigenvalue problems and this was further generalized by Van Barel and co-authors [3, 4].

The use of rational filters in eigenvalue calculations requires solving a sequence of shifted linear systems. While considerable effort has been devoted to the design of accurate filters, relatively little attention has been paid to the influence of the resulting filters on the solution of these systems, a particularly important issue when iterative solvers are used. One of the goals of this paper is to address this issue. The main contributions of this paper are as follows:

1. Traditional rational filters are selected primarily as a means to approximate the Cauchy integral representation of the eigenprojector via a quadrature rule. We review the most common of these, e.g., those derived from the midpoint rule and Gaussian quadratures and propose a mechanism to analyze their quality based on the derivative of the filter at the boundaries.

2. A known limitation of the Cauchy-based rational filters is that their poles tend to accumulate near the real axis making the resulting linear systems harder to solve, especially by iterative methods. To reach a balance between the filter quality and the effect of the resulting poles on the solution of linear systems encountered when applying the filter, we introduce a least-squares (LS) rational approximation approach to derive filters. If selected properly, these LS rational filters can outperform those standard Cauchy filters in big part because they yield a better separation of the wanted part of the spectrum from the unwanted part. Since the poles can be arbitrarily selected, this strategy allows one to place poles far away from the real axis and even to repeat poles a few times, a feature that is beneficial for both iterative and direct solvers.

3. We design procedures to fully take advantage of the nature and structure of the filter when iterative methods are employed. In particular, real arithmetic is maximized when the original matrix is real and symmetric, and the same Krylov subspace is recycled to solve the related systems that arise in the course of applying the filter. In addition, we resort to polynomial filters to preprocess the right-hand

sides for difficult interior eigenvalue problems.

The plan of this paper is as follows. Section 2 reviews standard rational filters encountered in eigenvalue computations and provides a criterion for evaluating their quality. Section 3 introduces a LS viewpoint for deriving rational filters and points out various appealing features of this approach when iterative solvers are used. Section 4 discusses techniques to efficiently solve the linear systems that arise in the application of the LS rational filters. Numerical examples are provided in section 5 and the paper ends with concluding remarks in section 6.

2. Rational filters for linear eigenvalue problems. In this section we consider rational filters and discuss the traditional way in which they have been designed as well as a mechanism for analyzing their quality. We can write general rational filters in the form

$$\phi(z) = \sum_j \frac{\alpha_j}{\sigma_j - z}, \tag{2.1}$$

so that each step of the projection procedure will require computing vectors like $\sum_j \alpha_j(\sigma_j I - A)^{-1}v$. As can be seen, applying the filter to a vector v requires solving the linear systems $(\sigma_j I - A)w_j = v$. One may argue that since we now need to solve linear systems there is no real advantage of this approach versus a standard shift-and-invert approach; see, e.g., [26]. The big difference between the two is that the shifts will now be selected to be complex, i.e., away from the real axis, and this will render iterative solves manageable. In fact we will see in section 3.1 that the shifts σ_j can be selected far from the real axis.

2.1. Rational filters from the Cauchy integral formula. The traditional way to define rational filters for solving eigenvalue problems is to exploit the Cauchy integral representation of spectral projectors. Given a circle Γ enclosing a desired part of the spectrum, the eigenprojector associated with the eigenvalues inside the circle is given by

$$P = \frac{1}{2i\pi}\int_\Gamma (sI - A)^{-1} ds, \tag{2.2}$$

where the integration is performed counterclockwise. As it is stated this formula is not practically usable. If a numerical integration scheme is used, then P will be approximated by a certain linear operator which is put in the form

$$\tilde{P} = \sum_{k=1}^{2p} \alpha_k(\sigma_k I - A)^{-1}. \tag{2.3}$$

This rational function of A is then used instead of A in a Krylov subspace or subspace iteration algorithm.

A few details of this approach are now discussed, starting with the use of quadrature formulas in (2.2). Introducing a quadrature formula requires a little change of variables to convert the problem. If the eigenvalues of interest are in the interval $[\gamma - d, \gamma + d]$, we can shift and scale the original matrix A into $A' = (A - \gamma I)/d$ which has the effect of mapping those wanted eigenvalues into the interval $[-1, 1]$. Thus, Γ in (2.2) can be assumed to be a unit circle centered at the origin in what follows. Going back to the original Cauchy integral formula for a certain function h, we exploit the change of variables $s = e^{i\pi t}$ to get

$$h(z) = \frac{1}{2\pi i}\int_\Gamma \frac{h(s)}{s - z} ds = \frac{1}{2}\int_{-1}^{1} \frac{h(e^{i\pi t})}{e^{i\pi t} - z} e^{i\pi t} dt. \tag{2.4}$$

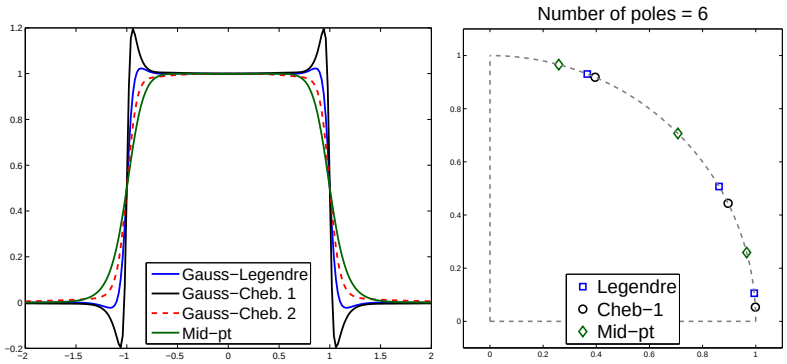

FIG. 1. *Standard midpoint rule, the Gauss–Chebyshev rules of the first and second kind, and the Gauss–Legendre rule. Left: filters. Right: poles.*

If $h(z)$ is real when z is real and $h(e^{i\pi t}) = h(e^{-i\pi t})$, then the above integral becomes

$$h(z) = \Re e \int_0^1 \frac{h(e^{i\pi t})}{e^{i\pi t} - z} e^{i\pi t} dt. \tag{2.5}$$

When $h(z) = 1$, the Cauchy integral formula representation of $h(z)$ will vanish for z outside Γ and behave like a step function when z is real.

Now any quadrature formula $\int_0^1 g(t)dt \approx \sum_{k=1}^p w_k g(t_k)$ can be used and it will lead to

$$h(z) \approx \Re e \sum_{k=1}^{p} w_k \frac{h(e^{i\pi t_k})}{e^{i\pi t_k} - z} e^{i\pi t_k} \equiv \Re e \sum_{k=1}^{p} w_k \frac{h(e^{i\theta_k})}{e^{i\theta_k} - z} e^{i\theta_k} \equiv \Re e \sum_{k=1}^{p} w_k \frac{h(\sigma_k)\sigma_k}{\sigma_k - z}, \tag{2.6}$$

where we have set for convenience $\theta_k \equiv \pi t_k$ and $\sigma_k \equiv e^{i\theta_k}$. Note that we have p nodes (poles) in the upper half circle and $2p$ nodes altogether if we count those on the lower half circle. Only the nodes with positive imaginary parts are actually used in the calculations. Accordingly, we will refer to p as the number of nodes keeping in mind that, implicitly, the formulas are based on a total of $2p$ nodes. The weights and nodes for several popular quadrature schemes are provided in Appendix A.

For the purpose of comparing the poles of the resulting schemes we show in Figure 1 the filters obtained from using the standard midpoint rule, the Gauss–Chebyshev rules of the first and second kind, and the Gauss–Legendre rule. The left subfigure shows the filters, and the right subfigure shows the corresponding poles located on the main quadrant of the complex plane (there is a 4-fold symmetry) except that the poles for the Chebyshev type-2 rational filters are omitted for better clarity.

Figure 1 shows that the filters with poles closer to the real axis have sharper drops at the boundary points -1 and 1 and those poles tend to concentrate nearer the real axis as the number of poles increases.

One may ask whether or not better filters can be designed. The answer to this question is not as simple as it appears at first. Indeed, if by "better" we mean a procedure that will lead to a faster procedure overall, then several parameters are involved some of which are difficult to analyze. For simplicity we could consider just the subspace iteration as the projection method. For example, if iterative solvers are to be used, then the location of the poles, preferably farther away from the real line, is crucial. This is in complete contrast with direct solvers that are oblivious to the

location of the poles. If direct solvers are used and we are using a filter with p poles, all that matters is that these poles will yield convergence in the smallest number of (filtered) subspace iterations.

One way to address the above question is to adopt an approximation theory viewpoint. Assuming the interval of interest is $[-1, 1]$, what is desired is a filter function $\phi(z)$ such that *when z is restricted to the real axis, then $\phi(z)$ is the best approximation to the step function $h(z)$ which has value one in $[-1, 1]$ and zero elsewhere.* For subspace iteration to converge the fastest we will need the sharpest possible decrease from the plateau of 1 inside the interval $[-1, 1]$ to zero outside the interval. An approach of this type is represented in the work by Güttel et al. who resort to uniform approximation. The optimal distribution of the poles found in [10] is ideal when direct solvers are employed. On the other hand, it was observed that the poles tended to concentrate near the real axis, rendering iterative solution techniques slow or ineffective. This discrepancy is illustrated in section 5 by numerical examples where the filters leading to faster convergence for subspace iteration when direct solvers are used are found to yield slower convergence with iterative solvers. A similar issue was also addressed in [21] where authors pointed out that the iteration number of iterative solvers increases as the location of the poles approach the real axis and suggested to place the poles along two horizontal lines in order to force the poles to be at a fixed distance from the real line.

2.2. Filter quality. This section addresses the following question: How can we evaluate the quality of a given rational filter to be used for eigenvalue calculations? To simplify the theory, we will assume that a subspace iteration-type approach is employed. Notice that in the left subfigure of Figure 1 the Chebyshev type 1 filter does not do a particularly good job at approximating the step function, relative to the other filters shown. However, our primary goal is to have a filter that yields large values inside $[-1, 1]$ and small ones outside, *with a sharp drop across the boundaries.* How well the step function is approximated is unimportant for the convergence. With this in mind, the Chebyshev filter may actually be preferred to the other ones.

It is convenient to restate the problem in terms of the reference interval $[-1, 1]$. The eigenvalues of interest, those wanted, are inside the interval $[-1, 1]$ and the others are outside. Let μ_k, $k = 1, \ldots, n$, be the eigenvalues of the filtered matrix $B = \phi(A)$, sorted in decreasing order of their magnitude:

$$\mu_1 \geq \mu_2 \geq \cdots \mu_m > \mu_{m+1} \geq \cdots ,$$

where $\mu_1, \ldots, \mu_m$ transform eigenvalues $\lambda_i, i = 1, \ldots, m$ inside the interval $[-1, 1]$ and the others transform eigenvalues $\lambda_i \notin [-1,\ 1]$.

In order to compare filters, we need to rescale them so that all those transformed wanted eigenvalues ($\mu_i's$) will be greater than or equal to $1/2$ and the others less than $1/2$, i.e.,

$$\mu_1 \geq \mu_2 \geq \cdots \mu_m \geq \frac{1}{2} > \mu_{m+1} \geq \cdots . \tag{2.7}$$

This is achieved by dividing the original filter ϕ by $2\phi(1)$. Note that the threshold $1/2$ is selected only because it occurs naturally in the Cauchy integral rational filters and that this scaling has no effect on the behavior of the resulting subspace iteration procedure. With this, a minimal requirement for the filter is the following:

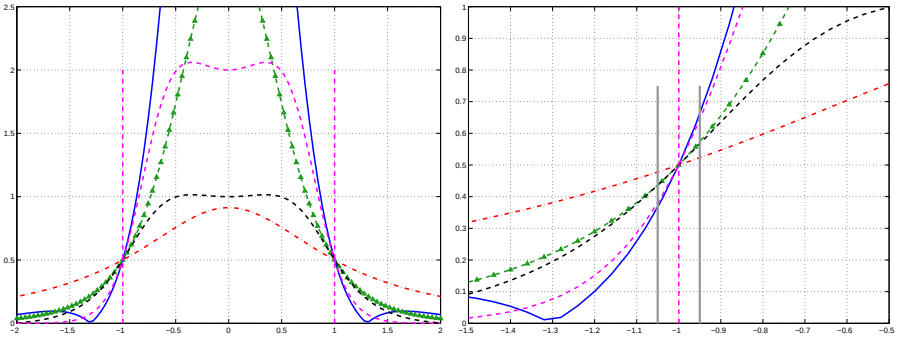

FIG. 2. *Various filters (left) and a zoom at a critical point (right).*

The two eigenvalues μ_m and μ_{m+1} are such that

$$\text{(2.8)} \qquad \begin{cases} \text{If} \quad \lambda_i \in [-1,\ 1], & \text{then} \quad \phi(\lambda_i) \geq \mu_m \geq \frac{1}{2} = \phi(1), \\ \text{If} \quad \lambda_i \notin [-1,\ 1], & \text{then} \quad |\phi(\lambda_i)| \leq \mu_{m+1} < \frac{1}{2} = \phi(1). \end{cases}$$

This requirement must be verified at the outset once the filter is obtained. Specifically, the filter should be such that $|\phi(\lambda_i)| < \frac{1}{2}$ for $t \notin [-1, 1]$ and $\phi(t) \geq \frac{1}{2}$ for $t \in [-1, 1]$.

Figure 2 shows various rational filters to compute eigenvalues in the interval $[-1, 1]$. The specific ways in which these filters are obtained are unimportant for the following discussion. An immediate question one may ask is: "Assume subspace iteration is used with the above filters, which one is likely to yield the fastest convergence?"

As is well known, when subspace iteration is employed, the convergence factor for each eigenvalue μ_k among those corresponding to the λ_i's inside the interval $[-1, 1]$ is given by $|\mu_k/\mu_{m+1}|$. Therefore, the slowest converging eigenvalues are those close to the threshold $1/2$. A better filter is one that achieves a better separation between these eigenvalues and those unwanted eigenvalues that are immediately below $1/2$. Thus, to handle a general case, it is best to have a filter whose derivatives at 1 and –1 are large in magnitude, with the assumption that the critical behavior of subspace iteration is governed by eigenvalues near these two boundaries.[1] In general, the larger the derivative the better the separation achieved for those eigenvalues μ_m, μ_{m+1} closest to the threshold $1/2$. Throughout this discussion, we have not made any reference to the other eigenvalues μ_k, those not located in the vicinity of $1/2$, because they are not critical to the convergence of subspace iteration. Based on this argument, the simplest indicator of performance of a filter is the magnitude of its derivative at -1 (or 1). Hence the following definition.

DEFINITION 2.1. *Let $\phi(z)$ be an even filter function that satisfies the requirement* (2.8). *Then we will call the* separation factor *of ϕ its derivative at $z = -1$.*

We now examine this measure for the Cauchy integral based filters for illustration. For any rational filter based on quadrature the derivative of ϕ at -1 takes the form

$$\phi'(z) = \frac{1}{2} \sum_{k=1}^{2p} \frac{\alpha_k}{(z - \sigma_k)^2} \rightarrow \phi'(-1) = \frac{1}{2} \sum_{k=1}^{2p} \frac{\alpha_k}{(1 + \sigma_k)^2}.$$

[1] It is conceivable that $\mu_m = \phi(\lambda_m)$ where λ_m is an eigenvalue of A that is far away from -1 or 1, in which case the derivative at -1, or 1 may not provide a good measure of the convergence speed. This may occur for very poor filters and is excluded from consideration.

Note that the sum is over all $2p$ poles and that the factor $1/2$ comes from the same factor in (2.4). Consider the situation when $\sigma_k = e^{i\theta_k}$ and $\alpha_k = w_k e^{i\theta_k}$ as is the case when Cauchy integrals are used along with quadrature. Then

$$\begin{aligned}\sum_{k=1}^{2p} \frac{\alpha_k}{(1+\sigma_k)^2} &= \sum_{k=1}^{2p} \frac{w_k e^{i\theta_k}}{(1+e^{i\theta_k})^2} = \sum_{k=1}^{2p} \frac{w_k}{e^{-i\theta_k}(1+e^{i\theta_k})^2} \\ &= \sum_{k=1}^{2p} \frac{w_k}{(e^{i\theta_k/2}+e^{-i\theta_k/2})^2} = \sum_{k=1}^{2p} \frac{w_k}{2+e^{i\theta_k}+e^{-i\theta_k}} \\ &= \frac{1}{2}\sum_{k=1}^{2p} \frac{w_k}{1+\cos\theta_k} \ = \sum_{k=1}^{p} \frac{w_k}{1+\cos\theta_k}.\end{aligned}$$

The last equality comes from the fact that the poles are conjugate symmetric. This leads to the result stated in the following lemma.

LEMMA 2.2. *The separation factor for a Cauchy integral based filter with poles located at angles θ_k and weights w_k is given by*

$$\phi'(-1) = \frac{1}{2}\sum_{k=1}^{p} \frac{w_k}{1+\cos\theta_k} = \frac{1}{4}\sum_{k=1}^{p} \frac{w_k}{\cos^2\frac{\theta_k}{2}}. \tag{2.9}$$

When one of the angles θ_k is equal to π as is the case for the trapezoidal rule, the derivative is infinite as expected. In situations when there is a pole close to -1 then the corresponding θ_k is close to π and the derivative becomes large (unless w_k is small). This situation is common for schemes based on Gaussian quadrature. In the simplest case of the midpoint rule, the derivative is explicitly known.

PROPOSITION 2.3. *The separation factor for the Cauchy type rational filter based on the midpoint quadrature rule is equal to $\phi'(-1) = p/2$ where p is the number of poles.*

Proof. Based on (30) in [12], we know that the rational filter for the 2p-pole midpoint rule is

$$\phi(z) = \frac{1}{z^{2p}+1},$$

which has derivative $\phi'(z) = -2pz^{2p-1}/(z^{2p}+1)^2$. Thus, we obtain $\phi'(-1) = p/2$. ∎

Regarding the Chebyshev-based Gaussian quadrature rule, the following can be stated.

PROPOSITION 2.4. *Let $s = \frac{\pi}{2p}$. The separation factor for the Cauchy type rational filter based on the Chebyshev quadrature rule of the first kind satisfies the inequality*

$$\phi'(-1) \geq \frac{\frac{1}{2}s\sin s}{1-\sin(\frac{\pi}{2}\cos(s))} \approx \left(\frac{8p}{\pi^2}\right)^2. \tag{2.10}$$

Proof. Define $s_k = \frac{(2k-1)\pi}{2p}$, $k = 1, \ldots, p$. We start with the left part of expression (2.9),

$$\phi'(-1) = \frac{1}{2}\sum_{k=1}^{p} \frac{w_k}{1+\cos\theta_k} = \frac{1}{2}\sum_{k=1}^{p} \frac{s_1 \sin s_k}{1+\cos(\frac{\pi}{2}(1+\cos s_k))}. \tag{2.11}$$

Note that $\cos(\pi/2 + \pi\cos(s_k)/2) = -\sin(\pi\cos(s_k)/2)$. The smallest denominator is reached for $k = 1$. All terms in the sum (2.11) are positive and are in fact dominated by the first term that corresponds to $k = 1$. The lower bound is obtained by retaining this term only:

$$\phi'(-1) \geq \frac{\frac{1}{2}s_1 \sin s_1}{1 - \sin(\frac{\pi}{2}\cos s_1)},$$

and by denoting s_1 by s. Taylor series expansions allow one to obtain the approximation at the end of (2.10). ∎

Example. When $p = 8$ then $\phi'(-1)$ as calculated from (2.9) is 44.262. Its lower bound in (2.10) is 42.052 and the approximation of this term at the end of (2.10) is 42.049. This is a quite reasonable lower bound because the first term is much larger than the other ones in the sum. Had we taken two terms instead of one in the sum the lower bound would improve slightly to 43.618.

3. Least-squares rational filters. For better flexibility in the choice of the poles, as iterative solvers will be used, we will resort to a different approach from that of Cauchy integrals or uniform approximations. Consider a situation where the poles are selected in advance and we wish to get the best approximation to the step function in the *least-squares* sense. It turns out that this approach has many advantages, especially when iterative methods are considered. A similar approach was recently proposed by Van Barel in [4] where the rational filter functions are constructed for Hermitian or non-Hermitian problems, by solving nonlinear LS problems. The main difference between our approach is that we fix the location of the poles first and then compute the weights because we wish to put some constraints on the location of the poles. In contrast, the approach in [4] computes nodes and weights simultaneously by resorting to optimization tools [30]. Another difference is that we use exact L_2 integration instead of numerical quadrature when solving the least squares problem. Finally, our approach can accommodate repeated poles, leading to a reduction of the number of poles employed. For example, a filter with an excellent separation factor can be computed that uses only one pole repeated four times. This "repeated poles" feature will be discussed in detail in section 3.2.

Figure 3 shows an illustration. Just by looking at the sharpness of the curves around the interval boundaries, -1, and 1, one can guess that the LS filter (solid line) will perform better than the midpoint filter (dashed line) *based on the same poles* and about as well as the Cauchy–Gauss filter (dash-dotted line) using also three poles. Note here that the LS filters have been rescaled so as to have the same value 1/2 as the Cauchy filters at the points -1, 1. Moreover, the poles for the midpoint rule are advantageous relative to Gauss poles when iterative solvers are invoked for solving the linear systems and this will be confirmed by the numerical experiments; see section 5.

On the right figure, we plot the same two curves as on the left based on the rational functions from the Cauchy integral, but we change the poles for the LS approach to the three preset poles $[\pm 1 + 0.7i, 0.7i]$. These three poles are farther away from the real axis than those of the midpoint poles: $[(\pm\sqrt{3}+i)/2, i]$ or the Gaussian quadrature poles: $[\pm 0.9380 + 0.3467i, i]$. It is likely that linear systems obtained from these poles will be easier to solve by iterative methods. Yet, the separation factor appears to be better for the LS filter, indicating that in all likelihood the subspace iteration will converge somewhat faster than the Gauss method and certainly faster than the midpoint rule based method. The reason why LS rational filters usually yield larger

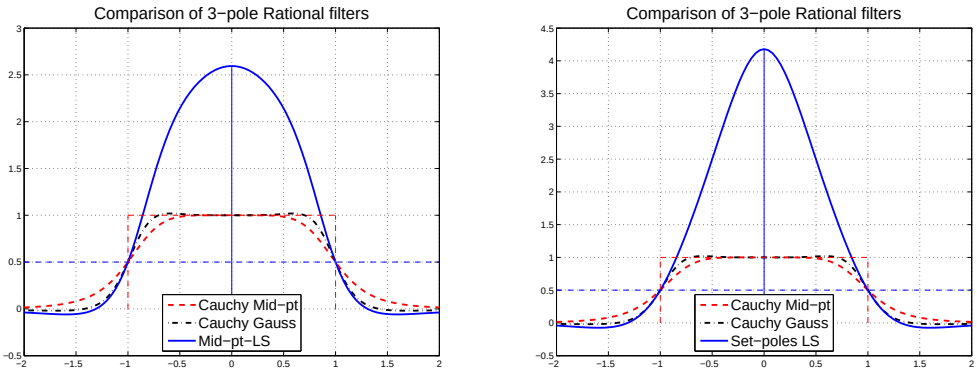

FIG. 3. *Left: Comparison of a standard rational function based on Cauchy integral using the midpoint rule with 3-poles (dashed line), a LS rational filter (solid line) using the same poles, and a standard 3-pole Cauchy integral using Gaussian quadrature (dash-dotted line). Right: The poles of the LS curve of the left figure are replaced by the preset values* $[-1+0.7i, 0.7i, 1+0.7i]$. *The other two curves are the same as on the left.*

separation factors is due to the special weight function used in the L_2 integration, which is discussed in the next section. This weight function relaxes the requirement to produce a good approximation to one in the interval $[-1, 1]$. This has the result of making the filter have larger values inside $[-1, 1]$, smaller values outside $[-1, 1]$, and sharper drops across the boundaries -1 and 1.

3.1. Obtaining least-squares rational filters. Consider the basis functions

$$\phi_j(z) = \frac{1}{z - \sigma_j}, \ j = 1, 2, \ldots, 2p,$$

where the poles $\sigma_1, \sigma_2, \ldots, \sigma_{2p}$ are all different from ± 1. We would like to find the function $\phi(z) = \sum_{j=1}^{2p} \alpha_j \phi_j(z)$ that minimizes

$$\|h - \phi\|_w^2 , \tag{3.1}$$

where h is the characteristic function which takes the value one for $t \in [-1,\ 1]$ and zero elsewhere, and the norm in (3.1) is associated with the standard L_2 inner product

$$\langle f, g \rangle = \int_{-\infty}^{\infty} w(t) f(t) \overline{g(t)} dt. \tag{3.2}$$

Here, the weight function $w(t)$ is taken to be of the form

$$w(t) = \begin{cases} 0 & \text{if} \quad |t| > a, \\ \beta & \text{if} \quad |t| \le 1, \\ 1 & \text{else}, \end{cases} \tag{3.3}$$

where a is typically set equal to ten or larger and β is a small scalar. In fact, the quality of LS filters depends on the value of β. This will be discussed at the end of this section.

General LS theory states that the linear combination $\phi = \sum \alpha_j \phi_j$ in (3.1) is optimal if and only if

$$\left\langle h - \sum_j \alpha_j \phi_j, \phi_i \right\rangle = 0, \quad i = 1, \ldots, 2p. \tag{3.4}$$

This means that the optimal α is the solution of the following linear system:

$$G\alpha = \eta, \tag{3.5}$$

where the entries of the matrix $G \in \mathbb{C}^{2p\times 2p}$ are $g_{ij} = \langle \phi_j, \phi_i \rangle$ and those of the vector $\eta \in \mathbb{C}^{2p}$ are $\eta_i = \langle h, \phi_i \rangle$.

Moreover, each g_{ij} and η_i can be computed analytically without resorting to numerical integration schemes. For example, simplifying the expression of g_{ij} we obtain

$$\begin{aligned} g_{ij} =< \phi_j, \phi_i > &= \int_{-\infty}^{+\infty} w(t)\phi_j(t)\overline{\phi_i(t)}dt = \int_{-\infty}^{+\infty} \frac{w(t)}{(t-\sigma_j)(t-\bar{\sigma}_i)}\, dt \\ &= \int_{-a}^{+a} \frac{1}{(t-\sigma_j)(t-\bar{\sigma}_i)}\, dt + (\beta - 1)\int_{-1}^{+1} \frac{1}{(t-\sigma_j)(t-\bar{\sigma}_i)}\, dt, \end{aligned}$$

and the two integrals involved in the above equation have the following closed form solution:

$$\int_c^d \frac{1}{(t-\sigma_j)(t-\bar{\sigma}_i)}\, dt = \begin{cases} \frac{1}{c-\sigma_j} - \frac{1}{d-\sigma_j} & \text{if} \quad \sigma_j = \bar{\sigma}_i, \\ \frac{1}{\sigma_j - \bar{\sigma}_i}\left(\log\frac{d-\sigma_j}{c-\sigma_j} - \log\frac{d-\bar{\sigma}_i}{c-\bar{\sigma}_i}\right) & \text{otherwise.} \end{cases} \tag{3.6}$$

Similarly,

$$\eta_i =< h, \phi_i >= \beta \int_{-1}^{+1} \frac{1}{t-\bar{\sigma}_i}\, dt = \beta \log\frac{\bar{\sigma}_i - 1}{\bar{\sigma}_i + 1}. \tag{3.7}$$

We now comment on the choice of β in (3.3). Figure 4 illustrates two LS rational filters using the same poles but different β's. The left subfigure shows that the LS filter with $\beta = 1$ approximates h better in $[-1, 1]$ than the one with $\beta = 0.01$. However, this is unimportant for the convergence of subspace iteration. In the right subfigure, we rescale both filters to make them take the same value $1/2$ at the points -1 and 1 in order to facilitate comparisons. Now it becomes clear that the LS filter with $\beta = 0.01$ does a better job at separating those wanted eigenvalues from unwanted ones. Figure 5 illustrates the influence of the important parameter β on the derivative of the related filter ϕ_β at -1. It suggests that reducing β leads to a much sharper drop across the boundaries at -1 and 1.

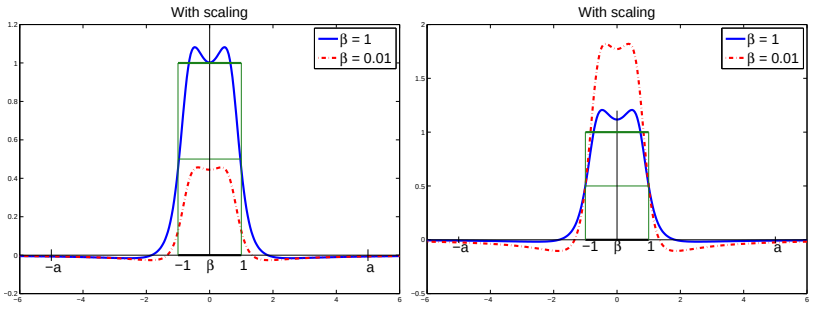

FIG. 4. *Comparison of two LS rational filters using the same poles located at* $[-0.75+0.5i, 0.75+0.5i]$ *but different* β*'s. Left: Without scaling. Right: Scaling used so* $\phi(-1) = \frac{1}{2}$.

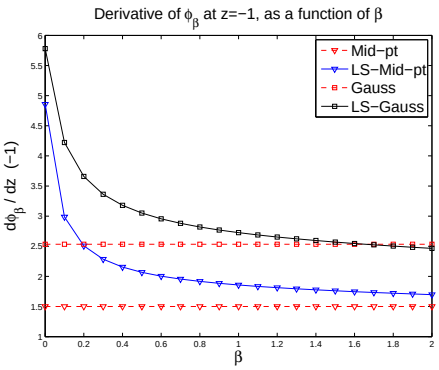

FIG. 5. *Comparison of separation factors associated with two rational filters based on Cauchy integral using the* 3*-pole midpoint rule and Gaussian quadrature rule, respectively, and LS rational filters using the same poles but different* β*'s.*

Another important advantage of LS filters is that their poles can be selected far away from the real axis, and this will typically result in a faster convergence of the iterative schemes employed to solve the related linear systems. Finally, LS rational filters will also permit repeated poles (see next section), and this can be exploited by direct and iterative schemes alike.

3.2. Repeated poles. A natural question which does not seem to have been raised before is whether or not we can select poles that are repeated a few times. For example, a rational function with a single pole repeated k times is of the form

$$r(z) = \frac{\alpha_1}{z-\sigma} + \frac{\alpha_2}{(z-\sigma)^2} + \cdots + \frac{\alpha_k}{(z-\sigma)^k} \ .$$

Generally we are interested in functions of the form

$$\phi(z) = \sum_{j=1}^{2p} \sum_{k=1}^{k_j} \frac{\alpha_{jk}}{(z-\sigma_j)^k}. \tag{3.8}$$

The hope is that repeating a pole would lead to filters with sharper drops at the boundaries, therefore; faster convergence for subspace iteration, but the cost of the solves will only be marginally increased. For example, if we use direct solvers, computing $(A-\sigma I)^{-1}v$ is dominated by the factorization, a price that is paid once for each pole in a preprocessing phase. When iterative solvers are employed, then there are some benefits in repeating poles as well. This will be discussed in the next section.

Figure 6 (left) shows a comparison of LS filters with repeated poles and Cauchy-type rational filters. The LS double pole filter is obtained by doubling the pole located at $1.0i$, i.e., by repeating. Based on the sharpness at the boundary points, we can expect that this LS filter will likely outperform the Cauchy–Gauss filter that is based on the same pole and perform about as well as the Cauchy–Gauss filter with two poles if direct solvers are applied. However, this LS double pole filter may be superior to both Cauchy–Gauss filters if iterative solvers are in use. This is because its pole is located farther away from the real axis than the two poles of the Gauss-2-poles filter. Figure 6 (right) shows that the LS filters with repeated poles become much sharper at the boundaries, -1 and 1, as the multiplicity of the pole increases.

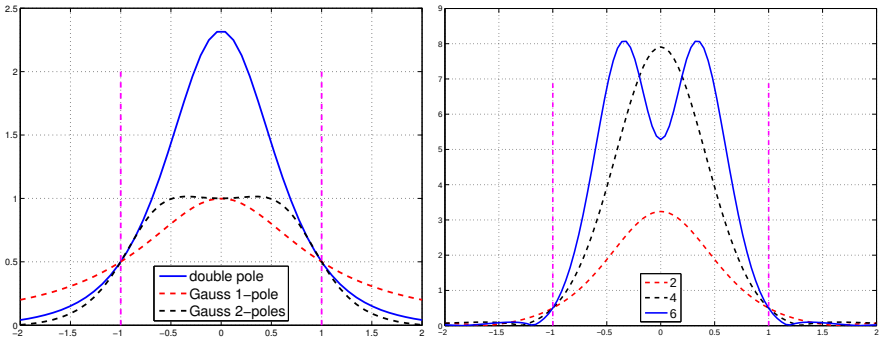

FIG. 6. *Left: Comparison of a LS filter with repeated poles and two Cauchy-type filters using the Gaussian quadrature rule. The double pole filter is obtained from a single pole repeated twice. Right: Comparison of LS filters with repeated poles obtained from repeating a pole located at* $1i$ k *times, where* $k = 2, 4, 6$.

Once the poles and their multiplicities are selected a filter of the general form (3.8) is obtained. The matrix G and right-hand side η are computed by a procedure similar to that of the single pole case. Below, we summarize the main differences with this case.

The entries of G are now equal to the inner products $<(z-\sigma_j)^{-k_1}, (z-\sigma_i)^{-k_2}>$. To compute these analytically, we obtain a partial fraction expansion of the integrand

$$\frac{1}{(z-\sigma_j)^{k_1}(z-\bar{\sigma}_i)^{k_2}} = \sum_{i=1}^{k_1} \frac{\theta_i}{(z-\sigma_j)^i} + \sum_{j=1}^{k_2} \frac{\gamma_j}{(z-\bar{\sigma}_i)^j}. \tag{3.9}$$

Now the integral of each term of the right-hand side of (3.9) has a closed-form expression. The coefficients θ_i and γ_j are obtained by simply equating both sides of (3.9). Details are omitted. The right-hand side η is calculated based on the following formula:

$$< h, \frac{1}{(z-\sigma_j)^k} >= \begin{cases} \beta \log \frac{\bar{\sigma}_j - 1}{\bar{\sigma}_j + 1} & \text{if} \quad k = 1, \\ \frac{\beta}{1-k} \left((1-\bar{\sigma}_j)^{1-k} - (-1-\bar{\sigma}_j)^{1-k} \right) & \text{otherwise.} \end{cases} \tag{3.10}$$

Because the multiplicity of the poles is relatively small, numerical stability is not an issue with the above calculations.

In the case of a repeated single pole, it is relatively easy to dynamically select the location of the pole to improve efficiency. Recalling that the interval of reference is $[-1,\ 1]$, we place the pole along the y-axis so as to obtain a symmetric filter. Assume that iterative methods are used for solving the linear systems. By moving the imaginary part upward and downward it is possible to optimize the procedure. A key observation is that the linear systems become very easy to solve by iterative methods toward the end of the subspace iteration procedure, because the right-hand sides become close to eigenvectors. Then, we can loosen the demand on the iterative solver when the subspace shows signs of convergence. This is achieved by making the filter sharper at the boundaries by moving the pole downward to get a better separation factor. In contrast, at the beginning of the process, a filter with sharper drops at the boundaries will cause the iterative solver to be quite slow, so it is best to place the pole higher until some accuracy is reached for the approximate eigenvectors.

In other words we can select the filter to balance the convergence between the linear system solver and the subspace iteration.

One reason we suspect that filters with repeated poles have not received attention in the literature thus far is that the ability to solve the linear systems at different poles in parallel has long been viewed as a key advantage of these methods. For problems large enough that the use of iterative solvers is mandatory, this advantage is blunted by the potential for severe load balancing problems due to differing convergence rates of the solvers at different poles. The apparent loss in parallelism that comes with using a single-pole filter can be overcome in this setting by taking advantage of parallelism from other sources, such as parallel matrix-vector products and parallel preconditioners. One also has the other two sources of parallelism available to all rational filtering methods: the parallelism that comes from having multiple right-hand sides and that from the ability to subdivide a target interval with many eigenvalues into smaller subintervals that can be processed simultaneously. Another advantage to using single-pole filters that applies even when direct solvers are used is that filters with fewer poles use less memory, so we can solve larger problems with the same computing resources.

4. Iterative inner solves. The primary goal of this section is to show how iterative solvers can be effectively exploited when LS repeated pole rational filters are employed.

4.1. Maximizing the use of real arithmetic. Apart from expensive factorization costs for large 3D problems, direct solvers also suffer from the additional burden of complex arithmetic in the application of rational filters when the poles are complex even though the original matrix A is real. This issue was recently bypassed in [2] by using rational filters with real poles only.

In this section, we will show that iterative solvers, such as Krylov subspace methods, can bypass this drawback even when the poles are complex. This property was also exploited in [21] for standard rational filters.

Given an initial guess x_0 and its residual $r_0 = b - Ax_0$, the basic idea of Krylov subspace methods to solve the system

$$(A - \sigma I)x = b$$

is to seek an approximate solution x_m from an affine subspace $x_0 + \mathcal{K}_m(A - \sigma I, r_0)$ by imposing certain orthogonality conditions, where

$$\mathcal{K}_m(A - \sigma I, r_0) = \text{span}\{r_0, (A - \sigma I)r_0, \ldots, (A - \sigma I)^{m-1} r_0\}.$$

A few common methods of this type start by constructing an orthogonal basis of $\mathcal{K}_m(A - \sigma I, r_0)$. As is well known, Krylov subspaces are shift-invariant, that is,

$$\mathcal{K}_m(A - \sigma I, r_0) = \mathcal{K}_m(A, r_0) \tag{4.1}$$

for any scalar $\sigma \in \mathbb{C}$. Therefore, this basis can be computed by applying the *Lanczos method* on the pair (A, r_0) rather than $(A - \sigma I, r_0)$. An immediate advantage of this approach is that the basis construction process can be carried out in real arithmetic when both A and r_0 are real. Recall that the right-hand side b in our problem is a real Ritz vector when the original matrix A is real and symmetric. Therefore, one way to keep r_0 real is to simply choose the initial guess x_0 as a zero vector so that $r_0 = b$. There are other advantages of this choice of x_0 as well, which will be discussed in the next section.

Assume the Lanczos method is started with $v_1 = b/\|b\|_2$ and does not break down in the first m steps, then it will produce the factorization

$$AV_m = V_m T_m + t_{m+1,m} v_{m+1} e_m^T, \tag{4.2}$$

where the columns of $V_m \in \mathbb{R}^{n,m}$ form an orthonormal basis of $\mathcal{K}_m(A, b)$ and $T_m \in \mathbb{R}^{m,m}$ is a tridiagonal matrix when $A \in \mathbb{R}^{n,n}$ is symmetric. Based on (4.2), the following relation holds for $A - \sigma I$:

$$(A - \sigma I)V_m = V_m(T_m - \sigma I_m) + t_{m+1,m} v_{m+1} e_m^T. \tag{4.3}$$

Note in passing that the above equality can also be written as

$$(A - \sigma I)V_m = V_{m+1}(T_{m+1,m} - \sigma I_{m+1,m}), \tag{4.4}$$

where $I_{m+1,m}$ represents the leading $(m+1) \times m$ block of an identity matrix of size $m+1$ and $T_{m+1,m}$ is the matrix obtained by adding a zero row to T_m and then replacing the entry in location $(m+1, m)$ of the resulting matrix by $t_{m+1,m}$.

Hence, the Lanczos factorization (4.3), or (4.4), is available for free for any σ, real or complex, from that of A in (4.2). In the next section, we show that complex arithmetic need only be invoked for the projected problem involving the smaller matrix $T_{m+1,m} - \sigma I_{m+1,m}$. In addition, we will also see that the same V_m can be recycled for solving various shifted linear systems.

4.2. Recycling Krylov subspaces. With the Krylov subspace information available in (4.3), we can solve the shifted linear systems encountered when applying rational filters as in (3.8), in a simple way. Consider first the case where there is only one pole located at σ with multiplicity $k > 1$. Under this assumption, the application of the rational filter with repeated poles on a vector b amounts to computing

$$x \equiv \sum_{j=1}^{k} \alpha_j (A - \sigma I)^{-j} b. \tag{4.5}$$

Using Krylov subspace methods to solve matrix equation problems such as the one in (4.5) has been extensively studied in the literature [11, 19, 25, 29, 33]. In this paper, we improve the method proposed in [25] by preserving the minimal residual property over the subspace $\mathcal{K}_m(A - \sigma I, b)$ while computing an approximate solution of (4.5).

First, a Horner-like scheme is used to expand the right-hand side of (4.5) as follows:

$$\begin{aligned} x &= (A - \sigma I)^{-1}(\alpha_1 I + (A - \sigma I)^{-1}(\alpha_2 I + \cdots (\alpha_{k-1} I + \alpha_k (A - \sigma I)^{-1}) \cdots))b \\ &= (A - \sigma I)^{-1}(\alpha_1 b + (A - \sigma I)^{-1}(\alpha_2 b + \cdots (\alpha_{k-1} b + \alpha_k (A - \sigma I)^{-1} b) \cdots)). \end{aligned}$$

From the above expansion, it is easy to see that x can be computed from a sequence of approximate solutions x_j, $j = k, k-1, \ldots, 1$, to the linear systems $(A - \sigma I)x_j = b_j$, where

$$b_j = \alpha_j b + x_{j+1} \quad (\text{when } j = k \text{ set: } x_{j+1} \equiv 0). \tag{4.6}$$

The final x_j, i.e., x_1, will be the desired approximation to x in (4.5).

MINRES [22] is used to compute an approximate solution x_k of the very first system $(A - \sigma I)x = b_k$. With (4.3), and using a zero initial guess, the approximate solution is

$$x_k = V_m y_k, \tag{4.7}$$

where y_k is the minimizer of

$$\min_{y\in\mathbb{C}^m} \|b_k - (A-\sigma I)V_m y\|_2. \tag{4.8}$$

When $j = k-1, k-2, \ldots, 1$, the approximate solution is taken of the form (4.7) with k replaced by j, where similarly y_j minimizes (4.8) with b_k replaced by b_j.

Notice that we use the same Krylov subspace $\mathcal{K}_m(A-\sigma I, b)$, rather than $\mathcal{K}_m(A-\sigma I, b_j)$, to compute approximations to $k-1$ vectors $(A-\sigma I)^{-1}b_j$, $j = 1, \ldots, k-1$. The rationale behind this approach is that as b gets close to an eigenvector corresponding to the eigenvalue λ in the outer subspace iteration, the solution of the linear system $(A-\sigma I)x = b$ can be approximated by $b/(\lambda-\sigma)$. As a result, b_j points nearly in the same direction as b which leads to $\mathcal{K}_m(A-\sigma I, b_j) \approx \mathcal{K}_m(A-\sigma I, b)$. By recycling $\mathcal{K}_m(A-\sigma I, b)$ k times, we are able to reduce the cost of constructing k orthogonal bases for each $\mathcal{K}_m(A-\sigma I, b_j)$ to only one for $\mathcal{K}_m(A-\sigma I, b)$. In addition, since all b_js except b are complex vectors, building such a basis for $\mathcal{K}_m(A-\sigma I, b)$ is less than half as costly as building one for each $\mathcal{K}_m(A-\sigma I, b_j)$.

Another consequence of recycling the subspace $\mathcal{K}_m(A-\sigma I, b)$ is that the b_j's are all in the range of V_m:

$$b_j = \alpha_j b + x_{j+1} = \alpha_j \|b\|_2 v_1 + V_m y_{j+1} = V_m(\alpha_j \|b\|_2 e_1 + y_{j+1}) \equiv V_m z_j.$$

This property simplifies the computation of y_j because, exploiting (4.4), we obtain

$$\begin{aligned}
\|b_j - (A-\sigma I)V_m y\|_2 &= \|V_m z_j - (A-\sigma I)V_m y\|_2 \\
&= \left\| V_{m+1} \begin{pmatrix} z_j \\ 0 \end{pmatrix} - V_{m+1}(T_{m+1,m} - \sigma I_{m+1,m})y \right\|_2 \\
&= \|\hat{z}_j - (T_{m+1,m} - \sigma I_{m+1,m})y\|_2,
\end{aligned}$$

where $\hat{z}_j$ is the vector of length $m+1$ obtained by appending a zero component to the end of z_j.

Hence, all the complex operations in the computation of an approximate solution to (4.5) are performed in the projected space of V_{m+1}. For example, each y_j, $j = 1, \ldots, k$, is solved through an $(m+1)\times m$ LS problem with a different right-hand side $\hat{z}_j$ but the same coefficient matrix $T_{m+1,m} - \sigma I_{m+1,m}$. To form z_j, we need only add $\alpha_j\|b\|_2$ to the first entry of y_{j+1}, which is readily available from the computation at step $j+1$. Moreover, the real part of x_1, which is actually needed in the outer iteration calculations, can be obtained by multiplying V_m with the real part of y_1. The intermediate vectors x_j and b_j in (4.6) are not needed explicitly during the whole solution process.

The above discussion can be easily generalized to the case where the rational filter has more than one pole. Details are omitted.

4.3. Preconditioning issues. When the right-hand side b gets close to the eigenvector solution, the subspace iteration process acts as an implicit preconditioner for the shifted linear system making it increasingly easier to solve iteratively.

However, slow convergence is often observed in the first few outer iterations due to inaccurate inner solves. When the eigenvalues of interest are deep inside the spectrum, the outer convergence often stagnates. Under these circumstances, some form of preconditioning is mandatory. ILU-type preconditioners are one possible choice but our experience is that their performance is far from satisfactory for highly indefinite problems [6]. Another issue is that ILU-preconditioners cannot take advantage of

the relation between the eigenvectors and the right-hand sides. Consider the right-preconditioned system:

$$(A - \sigma I)M^{-1}u = b, \quad u = Mx,$$

where M is an ILU-type preconditioner for $A-\sigma I$. When b approaches one eigenvector of A, this b is no longer a good approximation to an eigenvector of $(A - \sigma I)M^{-1}$. Meanwhile, the selection of a criterion to stop the preconditioning in the subspace iteration itself is quite challenging, especially when we have to solve multiple right-hand sides for each shifted system in one outer iteration.

Another choice that overcomes these issues is polynomial-type preconditioners [24]. Since our main task is to annihilate the component of b in the complementary eigenspace, we can preprocess the right-hand side by a polynomial filter [9] at the beginning of subspace iteration. As soon as the Ritz vectors gain two digits of accuracy, we immediately switch to rational filters in the subsequent outer iterations.

4.4. Locking. As shown in the previous sections, LS rational filters generally do not approximate the step function as well as Cauchy-type rational filters. This has the effect of yielding different rates of convergence of each approximate eigenvector in the subspace iteration. Once an eigenvector has converged, it is wasteful to continue processing it in the subsequent iterations. We can apply *locking* (see, e.g., [26]) to deflate it from the search space. Locking essentially amounts to freezing the converged eigenvectors and continuing to iterate only on the nonconverged ones. However, in order to maintain the orthogonality among the eigenvectors frozen at different iterations, we still have to perform subsequent orthogonalizations against the frozen vectors. This also helps with the convergence of inner iterative solves. See Algorithm 1 for a brief description of the filtered subspace iteration combined with locking for computing the n_{ev} eigenvalues inside a given interval $[a, b]$. It has been shown in [20, 31] that this locking strategy is also quite useful in identifying eigenvalues that are highly clustered or of very high multiplicity.

In Algorithm 1, a subspace with a basis X is first initialized. In the eigenvalue loop, $\phi(A)$ in step (a) represents the filtered matrix of A where ϕ is either a rational or a polynomial filter. Step (b) then orthonormalizes $\phi(A)X$ with respect to the frozen eigenvectors $q_1, \ldots, q_j$. Steps (c-d) perform the Rayleigh Ritz projection procedure on the original matrix A. Finally, step (e) checks the convergence of computed eigenvalues and appends the newly converged eigenvectors to Q. The dimension of the search space for the next iteration is also reduced accordingly.

Algorithm 1. Subspace iteration with locking and rational filtering.

1. **Start**: Choose an initial system of vectors $X = [x_1, \ldots, x_m]$ and set $j = 0$.
2. **Eigenvalue loop**: While $j \leq n_{ev}$ do:
   (a) Compute $\hat{Z} = [q_1, \ldots, q_j, \phi(A)X]$. ▷ ϕ: a rational/polynomial filter
   (b) Orthonormalize the column vectors of $\hat{Z}$ into Z (first j columns will be invariant) such that $Z = [q_1, \ldots, q_j, Z_{m-j}]$.
   (c) Compute $B = Z_{m-j}^* A Z_{m-j}$.
   (d) Compute the eigenvectors $Y = [y_{j+1}, \ldots, y_m]$ of B associated with the eigenvalues $\lambda_{j+1}, \ldots, \lambda_m$ and form $X = Z_{m-j}[y_{j+1}, \ldots, y_m]$.
   (e) Test the eigenvalues $\lambda_j, \ldots, \lambda_m$ for convergence. Let i_{conv} be the number of newly converged eigenvalues. Remove i_{conv} corresponding converged eigenvectors from X and append them to $Q = [q_1, \ldots, q_j]$. Set $j = j + i_{conv}$.

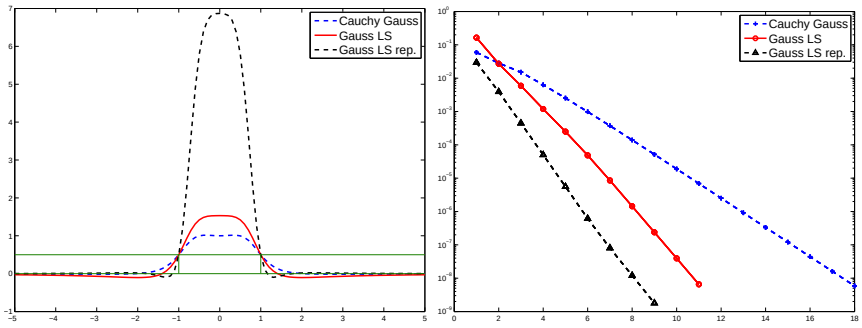

FIG. 7. *Comparison of Cauchy and least-squares rational filters to compute the smallest* 56 *eigenpairs for a two-dimensional* 2*D discrete Laplacian. Left: Filters. Right: The maximum residual for the computed eigenpairs inside the target interval* $[0.0, 0.2]$ *at each iteration.*

5. Numerical experiments. In this section we present some experiments to illustrate the performance of the subspace iteration with LS rational filters.

5.1. An experiment with MATLAB. We begin with a MATLAB experiment to illustrate LS filters and the convergence of the corresponding filtered subspace iteration. The test matrix results from a discretized 2D Laplacian on a 73×53 grid leading to a problem of size $n = 3869$. We compute the smallest 56 eigenvalues inside the interval $[0.0, 0.2]$ with three different filters on this matrix. The first is a Cauchy filter with four poles from the Gauss–Chebyshev quadrature rule of the first kind ($p = 2$ on each $1/2$ plane) and the second is a LS filter with the same poles. The third is a LS filter using double poles generated by repeating each of the previous Gauss–Chebyshev poles twice. Figure 7(left) displays these filters. We then run the subspace iteration algorithm combined with these filters and plot the maximum residual for the computed eigenpairs inside the target interval at each iteration in Figure 7(right). During the iteration, all the shifted linear systems are solved by a direct method and the subspace dimension is fixed at 61 which is 5 more than the number of the eigenvalues sought. As can be expected from the filter curves, the LS rational filtering approach converges much faster than the Cauchy-based one.

5.2. Experiments with a 3D Laplacian. Next, we show a number of comparisons with a few larger problems. The algorithms discussed in this paper have been implemented in C and the following experiments were performed in sequential mode on a Linux machine with Intel Core i7-4770 processor and 16G memory. The code was compiled with the gcc compiler using the -O2 optimization level. The convergence tolerance for the residual norm in the subspace iteration is set at 10^{-8} and the subspace dimension is equal to the number of eigenvalues inside the search interval plus 20. We choose UMFPACK [7] as the default direct solver in the experiments.

The following notation is used to denote different combinations of the filter and solver used in the subspace iteration:

- `Gauss-Cheby`: The filter selects both poles and weights based on the Gauss–Chebyshev quadrature rule of the first kind and the solver used is UMFPACK.
- `LS-Gauss-Cheby`: This method only differs from `Gauss-Cheby` in that the weights are computed from the LS approach.
- `Mid-pt` and `LS-Mid-pt`: Both are defined in a similar way as the previous ones except that the poles are selected based on the midpoint quadrature rule.

TABLE 1
Experiment setting for the 3D discrete Laplacian example.

Interval $[\eta, \xi]$	#eigs	$\frac{\xi-a}{b-a}$
$[0.0, 0.2]$	145	0.0157
$[0.4, 0.5]$	208	0.0407
$[0.9, 1.0]$	345	0.0825

- `Gauss-Cheby/LS-Mid-pt+GMRES(60)`: These methods differ from `Gauss-Cheby` and `LS-Mid-pt` in that UMFPACK is replaced by GMRES(60) without restarting to solve the shifted linear systems.
- `LS-Gauss-repeated(k)`: The LS filter is generated by repeating each pole k times where the poles are selected based on the Gauss–Chebyshev quadrature rule of the first kind. UMFPACK is called to solve the shifted linear systems.
- `LS-repeated(k,m)`: The LS rational filter is generated by repeating one pole k times. The real part of this pole equals zero and the imaginary part is selected to maximize the separation factor of the filter. The shifted linear system is solved by the recycling method proposed in section 4.2 with a maximum Krylov subspace dimension m.

Consider the discretization of the Laplacian $-\Delta$ subject to the homogeneous Dirichlet boundary conditions over the unit cube $[0, 1]^3$. Here, we fix one matrix discretized from a $50 \times 50 \times 50$ grid such that the resulting matrix has size $125,000$. The range of the spectrum $[a, b]$ for this matrix is $[0.01138, 11.98862]$.

The search interval $[\eta, \xi]$, the number of eigenvalues inside $[\eta, \xi]$ and the ratio $\frac{\xi-a}{b-a}$ are listed in Table 1 for this example. Here $\frac{\xi-a}{b-a}$ is meant to reflect the degree of difficulty for the iterative inner solves. That is, the smaller $|\frac{\xi-a}{b-a} - \frac{1}{2}|$ is, the more indefinite the shifted matrix is.

5.2.1. Extreme eigenvalue problems. We first apply 11 combinations of the filter and solver to compute all the desired eigenpairs inside the interval $[0.0, 0.2]$. The methods are divided into two groups: *Direct* and *Iterative*, as indicated in the first column of Table 2. The number of poles p in the upper half plane is shown in the table along with the number of outer iterations and the total CPU time. For the five methods in the Direct group, the CPU time is further divided into the factorization time for UMFPACK to factor p shifted linear systems in the preprocessing phase and the subsequent subspace iteration time. The inner iteration of the Krylov subspace methods used in the Iterative group is stopped when either the relative residual norm is reduced by a factor of 10^{-9} or the maximum number of 60 iterations is reached.

From Table 2, we can draw the following conclusions. First, for the methods in the Direct group, only the three LS methods (`LS-Gauss-Cheby`, `LS-Mid-pt` and `LS-Gauss-repeated(2)` with three poles) converge in less than 1000 seconds. On the other hand, as the number of poles grows, such as when it reaches 5 in this test, the LS approaches have comparable performance with that of the `Gauss-Cheby` approach but they are still much better than the `Mid-pt` approach. Nevertheless, the number of outer iterations for each method all reduces accordingly in this case. Second, in the Iterative group, the two methods using GMRES(60) are much slower than those based on the recycling strategies proposed in section 4.2, and even slower than those in the Direct group. In spite of this, a comparison between these two indicates that the method using the poles based on the midpoint rule converges faster than that based on the Gauss–Chebyshev quadrature, which is in sharp contrast to

TABLE 2

Computing 145 *eigenvalues inside* $[0.0, 0.2]$ *and their associated eigenvectors for the* 3*D discrete Laplacian example. The first five methods use a direct solver and the last six stop the iteration of the inner solvers when either the relative residual norm is reduced by a factor of* 10^{-9} *or the maximum number of* 60 *iterations is reached.*

	Method	#poles	#iter.	CPU time (sec.)		
				Fact.	Iter.	Total
Direct	`Gauss-Cheby`	3	17	294.32	832.97	1127.29
		5	5	489.85	593.36	1073.21
	`Mid-pt`	3	32	293.89	1297.92	1591.81
		5	18	490.01	1376.42	1866.43
	`LS-Gauss-Cheby`	3	12	295.12	555.16	850.28
		5	9	489.85	659.69	1149.54
	`LS-Mid-pt`	3	12	293.98	511.49	805.47
		5	10	489.04	704.31	1193.35
	`LS-Gauss-repeated(2)`	3	8	295.15	692.84	987.99
Iterative	`Gauss-Cheby+GMRES(60)`	3	13	0	2822.44	2822.44
	`LS-Mid-pt+GMRES(60)`	3	12	0	2564.05	2564.05
	`LS-repeated(3,60)`	1	17	0	217.22	217.22
	`LS-repeated(6,60)`	1	10	0	170.61	170.61
	`LS-repeated(9,60)`	1	9	0	163.93	163.93
	`LS-repeated(12,60)`	1	10	0	176.09	176.09

the situation in the Direct group. This discrepancy is consistent with our analysis in section 2.1 indicating that the location of the poles becomes crucial to the overall performance when iterative solvers are used. Although the `Mid-pt` filter has a smaller separation factor than the `Gauss-Cheby` filter, its poles are located farther away from the real axis (see Figure 1) and this facilitates the inner iterative solves. Third, the LS filters with repeated poles used in the last four methods are generated by repeating one pole 3, 6, 9, and 12 times. A higher multiplicity leads to a larger separation factor but results in fewer digits of accuracy after successive solves based on recycling the same subspace. A compromise between these two factors is to select a moderate multiplicity which is fixed to 6 in the following experiments. In the end, when we compare the performances of these two groups, the `LS-repeated` approach appears to be most efficient for this test.

5.2.2. Interior eigenvalue problems. In the next tests we compute all the eigenpairs in the second interval $[0.4, 0.5]$ shown in Table 1. Since the target interval is deeper inside the spectrum than $[0.0, 0.2]$, the resulting shifted linear systems are more indefinite. As a result, we reduce the stopping tolerance for the inner iterative solves to 10^{-10} and also use a larger Krylov subspace. For simplicity we choose some optimal methods from each category in Table 2. As can be seen in Table 3, the performance of the `LS-repeated(6,150)` method is the best according to computational time.

The polynomial preconditioning technique proposed in section 4.3 is now tested on this problem. The polynomial filter built for these tests is obtained by approximating the Dirac-δ function so that the maximum value of the approximation inside the reference interval $[-1, 1]$ is ten times larger than its values at 1 and -1, which can be achieved by gradually increasing the degree of the polynomial used. The number of matvecs performed in the application of the filter, the number of outer iterations (#iter.), and the iteration time are reported separately for the Chebyshev polynomial (`Cheby-Poly`) filter and LS rational filter in Table 4. The total computational time in the last column is equal to the sum of the time spent by these two filters. We also list the degree of the polynomial used in the `Cheby-Poly` filter, which is

TABLE 3

Computing 208 *eigenvalues inside* $[0.4, 0.5]$ *and their associated eigenvectors for the* 3*D discrete Laplacian. The first three methods use a direct solver and the last two methods stop the iteration of the inner solvers when either the relative residual norm is reduced by a factor of* 10^{-10} *or the maximum number of iterations is reached.*

Method	#poles	#iter.	CPU time (sec.)		
			Fact.	Iter.	Total
`LS-Gauss-Cheby`	3	23	328.36	1524.30	1852.66
	5	7	489.99	915.18	1405.17
`LS-Mid-pt`	3	13	330.49	893.48	1223.97
`LS-Gauss-repeated(2)`	3	11	328.34	1116.77	1445.11
`LS-repeated(6,100)`	1	32	0	1098.77	1098.77
`LS-repeated(6,150)`	1	19	0	840.06	840.06

TABLE 4

Computing 208 *eigenvalues inside* $[0.4, 0.5]$ *and their associated eigenvectors for the* 3*D discrete Laplacian example. The first two methods use the* `LS-repeated` *approach combined with the polynomial preconditioning technique while the last two use the polynomial filtered subspace iteration. The inner iteration in the first two methods stops when either the relative residual norm is reduced by a factor of* 10^{-10} *or the maximum number of iterations is reached.*

Method	Cheby–Poly. filter				Rational filter			Total time
	Degree	#mv.	#iter.	Time	#mv.	#iter.	Time	
`LS-repeated(6,100)`	197	180576	4	304.09	138350	19	407.10	711.19
`LS-repeated(6,150)`	197	180576	4	305.65	131216	12	309.15	614.80
`Cheby-Poly`	197	1428758	28	814.02	0	0	0	814.02
`Cheby-Poly`	100	2838648	208	2302.23	0	0	0	2302.23

selected automatically by the algorithm. The two `LS-repeated` approaches in Table 4 first perform four polynomial filtered subspace iterations to reduce the maximum residual norm below 10^{-2} and then switch to the rational filter for the subsequent iterations. The total computational time now drops to 711.19 and 614.80 seconds for the `LS-repeated(6,100)` and `LS-repeated(6,150)` methods, respectively. Thus, the preconditioning technique saves roughly 25% computational time over the methods based on rational filtering only. We also test the polynomial filtered subspace iteration methods, denoted as `Cheby-Poly`, and report the computational time in the last two rows in Table 4. When compared to the second approach in this table, the `Cheby-Poly` method with a Chebyshev polynomial of degree 197 requires 25% more computational time. If a smaller degree, e.g., 100, is specified by hand, the computational cost in one application of the filter diminishes, but the overall computational time increases dramatically because of the inferior quality of the resulting filter. Therefore, the polynomial preconditioning technique indeed improves the performance of the filtered subspace iteration compared with the methods relying only on either a rational or polynomial filter. However, polynomial filtering requires many more matvecs than the rational filter. Thus if the test matrix is not as sparse as this discrete Laplacian example, the rational filtering approach could become far more effective than polynomial filtering. This will be illustrated in the next section through other matrices coming from realistic applications.

For the third interval $[0.9, 1.0]$, we test the same four combinations as for the previous interval. Since the shifted matrices are much more indefinite in this case, the methods based on iterative solvers and no preconditioning are even slower than the `LS-Gauss-repeated(2)` approach as shown in Table 5.

TABLE 5

Computing 345 *eigenvalues inside* $[0.9, 1.0]$ *and their associated eigenvectors for the* 3*D discrete Laplacian example. The first four methods use a direct solver and the last two methods stop the iteration of the inner solves when either the relative residual norm is reduced by a factor of* 10^{-10} *or the maximum number of iterations is reached.*

Method	#poles	#iter.	CPU time (sec.)		
			Fact.	Iter.	Total
`LS-Gauss-Cheby`	3	31	294.47	3177.46	3471.93
	5	12	489.93	2404.69	2894.62
`LS-Gauss-repeated(2)`	3	8	294.85	1816.70	2111.55
`LS-repeated(6,100)`	1	52	0	3037.36	3037.36
`LS-repeated(6,150)`	1	25	0	2235.55	2235.55

TABLE 6

Computing 345 *eigenvalues inside* $[0.9, 1.0]$ *and their associated eigenvectors for the* 3*D discrete Laplacian example. The first four methods use the* `LS-repeated` *approach combined with the polynomial preconditioning technique while the last one uses the polynomial filtered subspace iteration. The inner iteration in the first four methods stops when either the relative residual norm is reduced by a factor of* 10^{-10} *or the maximum number of iterations is reached.*

Method	Cheby–Poly. filter			Rational filter			Total time
	#mv.	#iter.	Time	#mv.	#iter.	Time	
`LS-repeated(6,100)`	414640	4	725.32	396425	37	1175.82	1901.14
`LS-repeated(6,150)`	414640	4	721.59	326206	18	851.57	1573.16
`LS-repeated(6,100)`	518300	5	904.83	274536	29	786.93	1691.76
`LS-repeated(6,150)`	518300	5	906.99	238723	18	622.58	1529.57
`Cheby-Poly`	2069305	20	1634.16	0	0	0	1634.16

With the polynomial preconditioning, we can improve the performance of the rational filtered subspace iteration. A Chebyshev polynomial of degree 284 is used for all the methods in Table 6 and the method in the fourth row, which performs the polynomial filtered subspace iteration in the first five iterations, is best with respect to the computational time for this test. However, the difference in efficiency between this preconditioned scheme and the polynomial filtered approach in the fifth row becomes less significant.

5.2.3. A comparison with `ARPACK`. `ARPACK` [18] is a de-facto general-purpose benchmark code for solving large eigenvalue problems and, in particular, a version of it is used in the `eigs` function of MATLAB. In this experiment we run `ARPACK` to compute all the eigenpairs inside the three intervals in Table 1. Since the original Fortran 77 legacy code is much slower than the MATLAB build-in function `eigs`, we choose to run `ARPACK` through `eigs` in the experiments. As extreme eigenvalues start to converge first in `ARPACK`, this means more eigenvalues need to be computed in order to get all the desired ones. In fact, we essentially have to compute all the eigenvalues inside the larger interval $[a, \xi]$ instead of $[\mu, \xi]$. We set the maximum number of Lanczos basis vectors as twice of the number of the eigenvalues in $[0, \xi]$ and the convergence tolerance 10^{-8} in the experiments. According to the computational results in Table 7, even though `ARPACK` takes slightly less computational time for the first interval than the optimal LS rational filtering approach, it spends much more time for the other two intervals.

We would like to comment on the major differences between `ARPACK` and the methods proposed in this paper. First, `ARPACK` implements the implicit restart Arnoldi method while the proposed methods of this paper are all based on subspace

TABLE 7
Results for ARPACK [18] *to compute the eigenpairs of the* 3*D discrete Laplacian in Table* 1.

$[\mu, \xi]$	$[a, \xi]$	#eigs in $[a, \xi]$	CPU time (sec.)
$[0.0, 0.2]$	$[0.0, 0.2]$	145	134.71
$[0.4, 0.5]$	$[0.0, 0.5]$	675	948.94
$[0.9, 1.0]$	$[0.0, 1.0]$	2112	6847.74

TABLE 8
Hamiltonians from the University of Florida Sparse Matrix Collection [8].

Matrix	n	nnz	$[a, b]$	$[\eta, \xi]$	#eig	$\frac{\xi-a}{b-a}$
$Ge_{87}H_{76}$	112,985	7,892,195	$[-1.2140, 32.7641]$	$[-0.645, -0.0053]$	212	0.0356
$Ge_{99}H_{100}$	112,985	8,451,295	$[-1.2264, 32.7031]$	$[-0.650, -0.0096]$	250	0.0356
$Si_{41}Ge_{41}H_{72}$	185,639	15,011,265	$[-1.1214, 49.8185]$	$[-0.640, -0.0028]$	218	0.0220
$Si_{87}H_{76}$	240,369	10,661,631	$[-1.1964, 43.0746]$	$[-0.660, -0.0300]$	213	0.0264
$Ga_{41}As_{41}H_{72}$	268,096	18,488,476	$[-1.2502, 1300.93]$	$[-0.640, -0.0000]$	201	0.0010

iteration. Second, ARPACK will typically require much more storage. Indeed, for interior eigenvalue problems it necessitates computing unneeded eigenpairs as shown above.

While it is clear that for extreme eigenvalue problems other methods, e.g., ones based on Krylov subspaces such as ARPACK, can be superior, we point out that we have not made any attempt to optimize our code for extreme eigenvalue problems. For example, if the sought eigenvalues are known to belong to a small interval $[a, \xi]$, it is possible to speed up the computation by using a rational filter constructed on a larger interval $[a - c, \xi]$ for some positive constant c, rather than $[a, \xi]$. In this case, the inner iterative solutions would converge much faster since the pole location can be selected even farther away from the real axis.

5.3. Matrices from electronic structure calculations. In this section we compute eigenpairs of five (Hamiltonian) matrices from electronic structure calculations generated from the PARSEC package [17]. The sizes n, numbers of nonzeros nnz, the ranges of the spectrum $[a, b]$, the search intervals $[\eta, \xi]$, as well as the numbers of eigenvalues inside this interval and the ratio $\frac{\xi-a}{b-a}$ are shown in Table 8. The search intervals are selected to include the eigenvalues requested by the Time Dependent Density Functional Theory (TDDFT) application [9]. Note that these matrices are discretized from 3D models and have fairly dense factors. In fact, the ratio of the number of nonzeros of the LU factors obtained from UMFPACK to the square of the matrix size is in the range $[19.277\%, 23.141\%]$ rendering any method using a direct solver to factor their complex shifted variants very ineffective. Thus, we will only consider the LS-repeated and Cheby-Poly methods in the experiments.

We report the maximum Krylov subspace dimension m used in the inner solve, the number of matvecs (#mv), the number of subspace iterations (#iter.) and the total computational time in Table 9. The degree of the polynomial used in the Cheby-Poly method is also listed. We choose the subspace dimension as the number of eigenvalues inside $[\eta, \xi]$ plus 40 and stop the inner solve in LS-repeated when either the relative residual norm is reduced by a factor of 10^{-10} or the number of the maximum iteration is reached. As the matrix size increases, we increase the Krylov subspace dimension m accordingly. If the method does not converge in 50 outer iterations, we mark them by an X in the table.

TABLE 9
Comparison results for the Hamiltonians in Table 8.

Matrix	`LS-repeated(6,m)`				`Cheby-Poly`			
	m	#mv.	#iter.	Time	Degree	#mv.	#iter.	Time
$Ge_{87}H_{76}$	60	177526	19	1781.67	73	444778	24	2234.77
$Ge_{99}H_{100}$	60	201331	21	2175.74	73	575456	27	2890.06
$Si_{41}Ge_{41}H_{72}$	70	209485	18	3746.55	91	507605	22	4887.92
$Si_{87}H_{76}$	70	216772	24	3334.60	84	574976	27	4016.31
$Ga_{41}As_{41}H_{72}$	100	3788431	27	8583.39	174	X	X	X

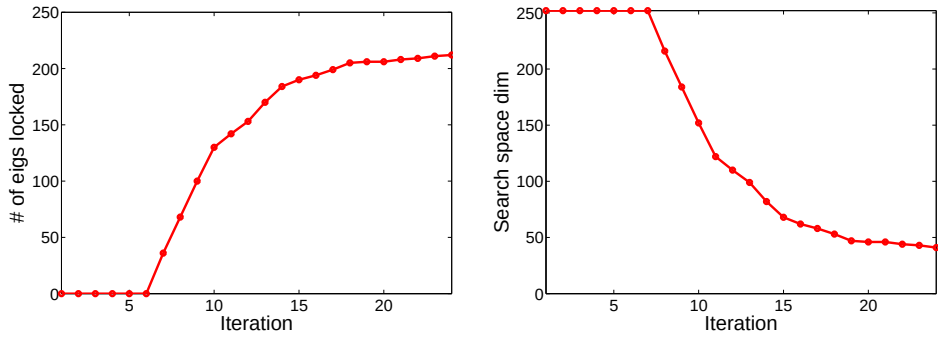

FIG. 8. *Convergence of the rational filtered subspace iteration for* $Si_{87}H_{76}$. *Left: Number of eigenvectors locked at each iteration. Right: Search space dimension at each iteration.*

Similar to the discrete Laplacian example, the rational filtering approach takes fewer matvecs and less computational time than the polynomial filtering approach for the five Hamiltonian problems. More specifically, `LS-repeated` saves roughly 40% matvecs and 25% computational time for the first four matrices. This performance gap becomes much wider for the last one. This is because the spectrum range $[a, b]$ of $Ga_{41}As_{41}H_{72}$ is about 40 times larger than that of the other four. Although the Chebyshev filter construction algorithm selects a relatively high degree of 174 to deal with this situation, the quality of the resulting filter does not compare well with that of the corresponding rational filter. Actually, the `Cheby-Poly` approach only locks 34 and 55 eigenpairs at the 37th and 50th iteration, respectively. The stretched spectrum of $Ga_{41}As_{41}H_{72}$ seems to cause less trouble to the solvers in `LS-repeated` by Krylov subspace methods.

In Figure 8, we use $Si_{87}H_{76}$ to illustrate a typical convergence of the rational filtered subspace iteration combined with the locking strategy. In the left subfigure, we plot the total number of eigenpairs locked at each iteration. We can see that no eigenpair reaches eight digits of accuracy in the first six iterations. Thereafter, eigenpairs start converging quickly and get deflated but this pattern flattens out beginning at the 18th iteration. In fact, the last few converging eigenpairs correspond to the eigenvalues near the boundaries of the search interval. Thus, a sharp drop of the rational filter across the boundaries is quite critical for a fast convergence. In the right subfigure, we plot the dimension of the search space at each iteration. It starts from 252 and eventually reduces to 41 leading to a lower computational cost per iteration as the iteration proceeds.

6. Conclusion. When it comes to designing rational filters for the eigenvalue problem, it is important to keep in mind the various parameters at play. In particular,

a good filter for a method based on a direct solver may not be the best when iterative methods are used instead. We have argued that it is not essential to build a filter that is an accurate approximation of the step function as is typically done via the Cauchy integral representation or uniform norm approximations. Instead, a LS approximation viewpoint was advocated that has several important advantages. Foremost among these is the flexibility to select poles away from the real line and to repeat these poles in an effort to reduce the overall computational cost. The numerical experiments have shown that this approach can indeed lead to superior performance when iterative methods are utilized for solving the related linear systems that arise when applying the filter, especially for the repeated pole filters.

A number of improvements can be made to the proposed scheme and these will be explored in our future work. For example, one issue worth exploring is the relation between the inner and outer stopping tolerances. Another is the potential advantage of replacing the standard single vector iterative methods in the inner solvers, with block Krylov subspace methods. Finally, another broad avenue is to explore the possibility of selecting optimal poles.

Appendix A. Classical quadrature rules. The simplest quadrature formula is the midpoint rule where the nodes x_k and weights, w_k are given by

$$\begin{cases} x_k = \frac{(2k-1)}{2p}, \\ w_k = \frac{1}{p}, \end{cases} \qquad k = 1, \ldots, p. \tag{A.1}$$

The nodes and weights for the Gauss–Chebyshev quadrature rule of the first kind are:

$$\begin{cases} x_k = \frac{1}{2}\left(1 + \cos\left(\frac{(2k-1)\pi}{2p}\right)\right), \\ w_k = \frac{\pi}{2p}\sin\left(\frac{(2k-1)\pi}{2p}\right), \end{cases} \qquad k = 1, \ldots, p, \tag{A.2}$$

while those associated with the Gauss–Chebyshev quadrature rule of the second kind are

$$\begin{cases} x_k = \frac{1}{2}\left(1 + \cos\left(\frac{k\pi}{p+1}\right)\right), \\ w_k = \frac{\pi}{2(p+1)}\sin\left(\frac{k\pi}{p+1}\right), \end{cases} \qquad k = 1, \ldots, p. \tag{A.3}$$

Among the most popular quadrature schemes in the context of contour integral methods for eigenvalue problems is the Gauss–Legendre rule whose nodes and weights are given by

$$\begin{cases} x_k = \frac{t_k+1}{2}, \\ w_k = \frac{1}{(1-t_k^2)[L_p{}'(t_k)]^2}, \end{cases} \qquad k = 1, \ldots, p, \tag{A.4}$$

where t_k is the kth root of the pth Legendre polynomial $L_p(x)$.

Acknowledgments. The authors would like to thank the anonymous referees for carefully reading the manuscript. We also thank the second referee for providing a short proof of Proposition 2.3.

REFERENCES

[1] J. Asakura, T. Sakurai, H. Tadano, T. Ikegami, and K. Kimura, *A numerical method for nonlinear eigenvalue problems using contour integrals*, JSIAM Lett., 1 (2009), pp. 52–55, doi:10.14495/jsiaml.1.52.

[2] A. P. Austin and L. N. Trefethen, *Computing eigenvalues of real symmetric matrices with rational filters in real arithmetic*, SIAM J. Sci. Comput., 37 (2015), pp. A1365–A1387, doi:10.1137/140984129.

[3] M. V. Barel, *Designing rational filter functions for solving eigenvalue problems by contour integration*, Linear Algebra Appl, 502 (2016), pp. 346–365, doi:10.1016/j.laa.2015.05.029.

[4] M. V. Barel and P. Kravanja, *Nonlinear eigenvalue problems and contour integrals*, J. Comput. Appl. Math., 292 (2016), pp. 526–540, doi:10.1016/j.cam.2015.07.012.

[5] W.-J. Beyn, *An integral method for solving nonlinear eigenvalue problems*, Linear Algebra Appl., 436 (2012), pp. 3839–3863, doi:10.1016/j.laa.2011.03.030.

[6] E. Chow and Y. Saad, *Experimental study of ILU preconditioners for indefinite matrices*, J. Comput. Appl. Math, 86 (1997), pp. 387–414, doi:10.1016/S0377-0427(97)00171-4.

[7] T. Davis, *Algorithm* 832*: UMFPACK V*4.3*—an unsymmetric-pattern multifrontal method*, ACM Trans. Math. Software, 30 (2004), pp. 196–199, doi:10.1145/992200.992206.

[8] T. Davis and Y. Hu, *The University of Florida Sparse Matrix Collection*, ACM Trans. Math. Software, 38 (2011), pp. 1:1–1:25, doi:10.1145/2049662.2049663.

[9] H. R. Fang and Y. Saad, *A filtered Lanczos procedure for extreme and interior eigenvalue problems*, SIAM J. Sci. Comput., 34 (2012), pp. A2220–A2246, doi:10.1137/110836535.

[10] S. Güttel, E. Polizzi, P. Tang, and G. Viaud, *Zolotarev quadrature rules and load balancing for the FEAST eigensolver*, SIAM J. Sci. Comput., 37 (2015), pp. A2100–A2122, doi:10.1137/140980090.

[11] L. Hoffnung, R. Li, and Q. Ye, *Krylov type subspace methods for matrix polynomials*, Linear Algebra Appl., 415 (2006), pp. 52–81, doi:10.1016/j.laa.2005.09.016.

[12] T. Ikegami, T. Sakurai, and U. Nagashima, *A filter diagonalization for generalized eigenvalue problems based on the Sakurai-Sugiura projection method*, J. Comput. Appl. Math., 233 (2010), pp. 1927–1936, doi:10.1016/j.cam.2009.09.029.

[13] A. Imakura, L. Du, and T. Sakurai, *Error bounds of Rayleigh-Ritz type contour integral-based eigensolver for solving generalized eigenvalue problems*, Numer. Algorithms, 71 (2016), pp. 103–120, doi:10.1007/s11075-015-9987-4.

[14] V. Kalantzis, J. Kestyn, E. Polizzi, and Y. Saad, *Domain decomposition approaches for accelerating contour integration eigenvalue solvers for symmetric eigenvalue problems*. preprint, 2016.

[15] J. Kestyn, E. Polizzi, and P. Tang, *FEAST eigensolver for non-Hermitian problems*, arXiv:1506.04463 [math.NA], 2015, http://arxiv.org/abs/1506.04463.

[16] L. Komzsik and T. Rose, *Substructuring in MSC/NASTRAN for large scale parallel applications*, Comput. Syst. Eng., 2 (1991), pp. 167–173, doi:10.1016/0956-0521(91)90017-Y.

[17] L. Kronik, A. Makmal, M. L. Tiago, M. M. G. Alemany, M. Jain, X. Huang, Y. Saad, and J. R. Chelikowsky, *PARSEC the pseudopotential algorithm for real-space electronic structure calculations: recent advances and novel applications to nano-structure*, Phys. Stat. Sol. (B), 243 (2006), pp. 1063–1079, doi:10.1002/pssb.200541463.

[18] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK USERS GUIDE: Solution of Large Scale Eigenvalue Problems by Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, 1998. Available at http://www.caam.rice.edu/software/ARPACK/.

[19] R. Li and Q. Ye, *A Krylov subspace method for quadratic matrix polynomials with application to constrained least squares problems*, SIAM J. Matrix Anal. Appl., 25 (2003), pp. 405–428, doi:10.1137/S0895479802409390.

[20] J. R. McCombs and A. Stathopoulos, *Iterative validation of eigensolvers: A scheme for improving the reliability of Hermitian eigenvalue solvers*, SIAM J. Sci. Comput., 28 (2006), pp. 2337–2358, doi:10.1137/050627617.

[21] H. Ohno, Y. Kuramashi, T. Sakurai, and H. Tadano, *A quadrature-based eigensolver with a Krylov subspace method for shifted linear systems for Hermitian eigenproblems in lattice QCD*, JSIAM Lett., 2 (2010), pp. 115–118, doi:10.14495/jsiaml.2.115.

[22] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629, doi:10.1137/0712047.

[23] E. Polizzi, *Density-matrix-based algorithm for solving eigenvalue problems*, Phys. Rev. B, 79 (2009), 115112, doi:10.1103/PhysRevB.79.115112.

[24] Y. Saad, *Iterative solution of indefinite symmetric linear systems by methods using orthogonal polynomials over two disjoint intervals*, SIAM J. Numer. Anal., 20 (1983), pp. 784–811 doi:10.1137/0720052.

[25] Y. Saad, *An overview of Krylov subspace methods with applications to control problems*, in Signal Processing, Scattering, Operator Theory, and Numerical Methods, Proceedings of the International Symposium MTNS-89, Vol III, M. A. Kaashoek, J. H. van Schuppen, and A. C. Ran, eds., Birkhauser, Boston, 1990, pp. 401–410.

[26] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, rev. ed., SIAM, Philadelphia, 2011, doi:10.1137/1.9781611970739.

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

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

[29] K. M. Soodhalter, D. B. Szyld, and F. Xue, *Krylov subspace recycling for sequences of shifted linear systems*, Appl. Numer. Math., 81 (2014), pp. 105–118, doi:10.1016/j.apnum.2014.02.006.

[30] L. Sorber, M. V. Barel, and L. D. Lathauwer, *Tensorlab v*2.0, January 2014, available online, www.tensorlab.net/.

[31] A. Stathopoulos and J. R. McCombs, *PRIMME: Preconditioned iterative multimethod eigensolver: Methods and software description*, ACM Trans. Math. Softw., 37 (2010), pp. 21:1–21:30, doi:10.1145/1731022.1731031.

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

[33] H. van der Vorst, *Iterative Krylov Methods for Large Linear Systems*, Cambridge University Press, Cambridge, 2003, doi:10.1017/CBO9780511615115.001.

[34] Y. Zhou, Y. Saad, M. L. Tiago, and J. R. Chelikowsky, *Parallel self-consistent-field calculations via Chebyshev-filtered subspace acceleration*, Phy. Rev. E, 74 (2006), 066704, doi:10.1103/PhysRevE.74.066704.