LEAST-SQUARES RATIONAL FILTERS FOR THE SOLUTION OF INTERIOR EIGENVALUE PROBLEMS *

YUANZHE XI † AND YOUSEF SAAD†

Abstract. This paper presents a method for computing partial spectra of Hermitian matrices, based on a subspace iteration method combined with rational filtering. While the general approach is the same as that of the FEAST package, the emphasis of this paper is on the selection of the filter. Specifically, the goal is to show the appeal of a least-squares viewpoint for designing filters. In particular, the paper shows how to define filters with repeated poles and demonstrates that this option can be attractive when iterative methods are used.

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

1. Introduction. One of the most challenging computational problems today is related to the study of excited states in solid-state physics. While Density Functional Theory (DFT) deals with ground states, the study of excited states involves transitions and invariably leads to much more complex computations. The common issue in such calculations is that one has to compute a large number of occupied as well as unoccupied states. This is illustrated by the Time Dependent Density Functional Theory (TDDFT) methodology to solve the problem. The so-called Casida approach [1], converts the problem into the computation of all eigenvalues and eigenvectors of a 'coupling' matrix K. Building this coupling matrix requires a large number of both occupied and unoccupied states [14]. Each pair of occupied / unoccupied states yields one column of K and to compute each column requires solving a Poisson equation on the physical domain. This is an intense calculation but one that is highly parallelizable. Larger examples would lead to hundreds of thousands or millions of occupied / unoccupied pairs. There are two main computational bottlenecks with these calculations. One is the computation of the states and the other is the diagonalization of the coupling matrix. A similar type of calculations, a rather more complex one, is the GW approach, see, e.g., [7, 16].

An ideal solution to deal with the challenges just outlined is a divide-and-conquer approach whereby the desired spectrum is extracted by 'slices' or 'windows' for example of a few hundreds or thousands each. An important property that is exploited in an approach of this type is the fact that eigenvectors of separate slices need not be orthogonalized against each other. It was pointed out to us [11] that for extremely large matrices some reorthogonalization may still be necessary as eigenvalues are often very close to each other. In this situation it is likely that reorthogonalization will only involve nearby slices.

This deceivingly simple looking idea runs against a few challenges. First, we need to deal with eigenpairs at the interfaces of the slices to prevent duplicate or missing eigenvalues. Second, we need to decide on how to partition the desired spectrum, i.e., we need to select the slices. Finally, we need to find ways to compute the eigenpairs in each slice.

Let us consider the last point. Any spectrum slicing technique will require solving interior eigenvalue problems. This means that we will need to have a way of computing eigenvalues and eigenvectors well inside the spectrum. The standard way in which this is

*Both authors were 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 under grant CCF-1505970.

†Address: Department of Computer Science & Engineering, University of Minnesota, Twin Cities. Work supported by DOE. {yxi,saad} @cs.umn.edu

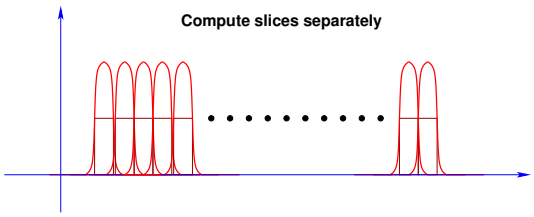

FIG. 1.1. *An illustration of spectrum slicing. Note that eigenvectors of separate slices need not be orthogonalized against each other. It may be only necessary to orthogonalize against a few or all eigenvectors of neighboring slices.*

done is via a shift-and-invert strategy combined with a standard projection method such as Lanczos or subspace iteration. The main steps of this technique are as follows:

Shift-invert Lanczos

1) Select a shift (or sequence of shifts) σ
2) Factor $A - \sigma I$: $A - \sigma I = LDL^T$
3) Apply the Lanczos algorithm to $(A - \sigma I)^{-1}$

The solves required with the matrix $A - \sigma I$ in step 3, are carried out using the factorization obtained in step 2. When the factorization is computable in a reasonable time, this approach can be quite effective and was used in industrial packages such as NASTRAN [9]. Its biggest limitation is the factorization which becomes prohibitive for large 3-D 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 filters. Here step 2 (factorization) of Shift-invert Lanczos is omitted and step 3 is replaced by an operation of the form $p(A)v$ where $p(t)$ is some polynomial designed to enhance the wanted eigenvalues and dampen the unwanted ones. Several existing codes employ polynomial filtering and this can be an excellent approach in the situation where the matrix-vector product operation is inexpensive. However, there is a hidden catch which can make the method very unappealing in some cases. Consider a spectrum such as the one pictured in Figure 1.2. Using polynomial filtering would be very ineffective for a case such as this one. This is because the interval containing all eigenvalues will be very large and we would need a very high degree polynomial to achieve the goal of separating wanted eigenvalues from unwanted ones. Ideally, one should take advantage of the separation of the stretched eigenvalues but a code based on such a strategy will be complex and possibly unreliable due to the variety of possible situations that can arise. 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. In particular, polynomial filtering is almost ruled out when dealing with generalized eigenvalue problems since a linear system must be solved with one of the two matrices in the pair, at each step.

In electronic structure calculations based on real space discretizations, polynomial filtering is attractive because the problems just mentioned do not arise. On the other hand,

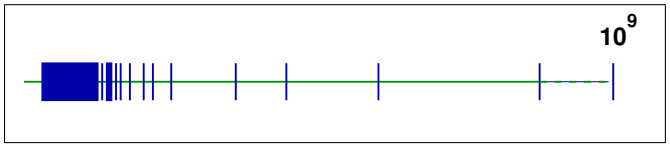

FIG. 1.2. *A spectrum with a long stretch on one end.*

rational filtering can offer competitive alternatives and will be studied in detail in this paper. 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 mid-point rule and Gaussian quadratures, and propose a mechanism to analyze their quality.

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 pre-process the right-hand sides for difficult interior eigenvalue problems.

Section 2 reviews standard rational filters encountered in eigenvalue computations and provides a criterion for evaluating their quality. Section 3 introduces a least-squares 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 least-squares rational filters. Numerical examples are provided in Section 5 and the paper ends with concluding remarks in Section 6.

2. Rational filters for 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 vs. a standard Shift-and-invert approach, see, e.g., [21]. 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 (poles of the rational function) can be selected in an ad-hoc fashion 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 representations of spectral projectors. This is represented in the work by Sakurai and co-workers [22, 23] and by Polizzi in the context of the FEAST package [18]. Given a circle Γ enclosing a desired part of the spectrum (slice), 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 counter-clockwise. 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. To compute $\tilde{P}v$ we need to solve linear systems with the matrices $\sigma_k I - A$.

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 the following discussion. 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:

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

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

$$h(z) = \Re e \int_0^1 \frac{h(e^{i\pi t})}{e^{i\pi t} - z} e^{i\pi t} dt. \tag{2.4}$$

We are specifically interested in the case when $h(t)$ is the step function whose value is one in the interval $[-1,\ 1]$ and zero elsewhere.

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

$$h(z) \approx \Re e \sum_{k=1}^{p} w_k \frac{h(e^{i\pi t_k})}{e^{i\pi t_k} - z} e^{i\pi t_k} \equiv \Re e \sum_{k=1}^{p} w_k \frac{h(e^{i\theta_k})}{e^{i\theta_k} - z} e^{i\theta_k} \equiv \Re e \sum_{k=1}^{p} w_k \frac{h(\sigma_k)\sigma_k}{\sigma_k - z}, \tag{2.5}$$

where we have set for convenience $\theta_k \equiv \pi t_k$ and $\sigma_k \equiv e^{i\theta_k}$.

The simplest quadrature formula used is the mid-point rule where

$$\left\{ \begin{array}{ll} \theta_k & = \frac{(2k-1)\pi}{2p} \\ w_k & = \frac{1}{p} \end{array} \right. \qquad k = 1, \cdots, p. \tag{2.6}$$

Note that we have p nodes (poles) in the upper half circle and $2p$ points altogether if we count those on the lower half plane. Only the poles with positive imaginary parts are actually used in the calculations. Accordingly, we will refer to p as the number of poles keeping in mind that, implicitly, the formulas are based on a total of $2p$ poles. It is best to avoid the trapezoidal rule because it leads to a rational function that has the poles -1, 1, so $\phi(t)$ is not defined at these two points of the real axis.

Among the Gaussian quadrature rules, the Gauss-Legendre quadrature rule is quite popular in this context [18]. It is also possible to use the Gauss-Chebyshev quadrature formula (first kind),

$$\int_{-1}^{1} \frac{g(t)}{\sqrt{1-t^2}} dt \approx \frac{\pi}{p} \sum_{k=1}^{p} g(\cos(t_k)), \quad t_k = \frac{(2k-1)\pi}{2p}, \tag{2.7}$$

by applying it to the function $f(t) = h(e^{i\pi t})e^{i\pi t}/(e^{i\pi t} - z)$ in (2.4) as follows:

$$\int_0^1 f(t)dt = \frac{1}{2}\int_{-1}^{1} f((s+1)/2)ds = \frac{1}{2}\int_{-1}^{1} \frac{\sqrt{1-s^2}f((s+1)/2)}{\sqrt{1-s^2}} ds.$$

This will yield the following values for θ_k, w_k to be used in (2.5):

$$\left\{ \begin{array}{ll} \theta_k & = \frac{\pi}{2}\left(1 + \cos\left(\frac{(2k-1)\pi}{2p}\right)\right) \\ w_k & = \frac{\pi}{2p}\sin\left(\frac{(2k-1)\pi}{2p}\right) \end{array} \right. \qquad k = 1, \cdots, p. \tag{2.8}$$

Similarly, we can derive a Chebyshev quadrature rule of the second kind in which the weight $w(t) = (1-t^2)^{1/2}$ replaces $w(t) = (1-t^2)^{-1/2}$ of (2.7). Details are omitted. For the purpose of comparing the poles of the resulting schemes we show in Figure 2.1 the filters obtained from using the standard Mid-point 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.

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 we would be satisfied with some solution to the problem: "Find a rational filter with p poles, that will lead to the fastest convergence of the filtered subspace iteration."

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*

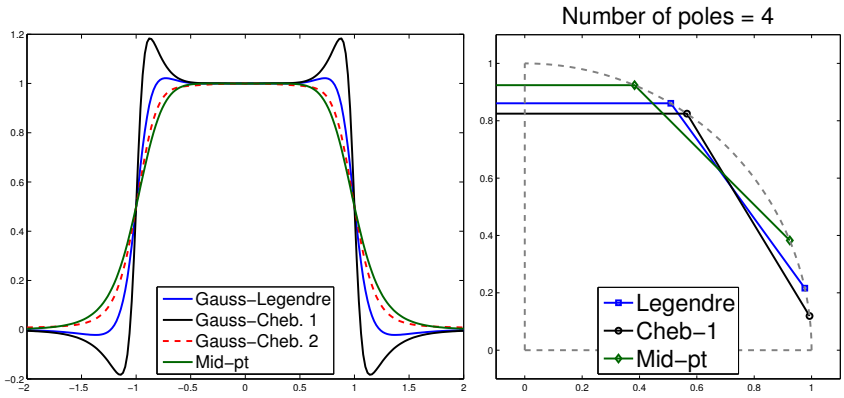

FIG. 2.1. *Standard Mid-point rule, the Gauss-Chebyshev rules of the first and second kind, and the Gauss-Legendre rule. Scaling used so* $\phi(-1) = \frac{1}{2}$. *Left: filters, right: poles.*

Figures 2.1 and 2.2 show 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.

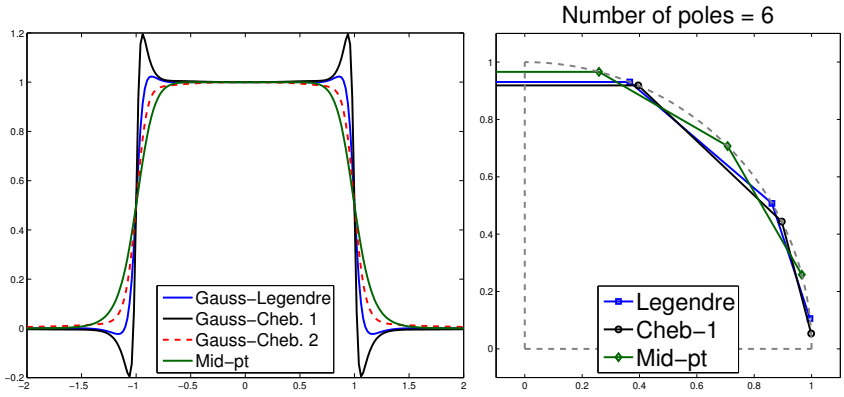

FIG. 2.2. *Same plots as in Figure 2.1 where the half number of poles is now 6.*

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 S. Güttel et al. 2014 who resort to uniform approximation. The optimal distribution of the poles found in [6] 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.

In this paper we do not address the optimal location of the poles when iterative solvers are in use. This is the subject of ongoing work.

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 2.1 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, $k = 1, \cdots, n$, 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 $\mu_1, \cdots, \mu_m$ transform eigenvalues $\lambda_i, i = 1, \cdots, m$ 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 larger than or equal to $1/2$ and the others less than $1/2$ as was done in the previous section, i.e.,

$$\mu_1 \geq \mu_2 \geq \cdots \mu_m \geq \frac{1}{2} > \mu_{m+1} \geq \cdots \tag{2.9}$$

This is achieved by dividing the original filter ϕ by $2\phi(1)$. Note that the threshold $1/2$ is selected 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:

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

$$\begin{cases} \text{If} \quad \lambda_i \in [-1,\ 1] \quad \text{then} \quad \phi(\lambda_i) \geq \mu_m \geq \frac{1}{2} = \phi(1) \\ \text{If} \quad \lambda_i \notin [-1,\ 1] \quad \text{then} \quad \phi(\lambda_i) \leq \mu_{m+1} < \frac{1}{2} = \phi(1) \end{cases} . \tag{2.10}$$

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

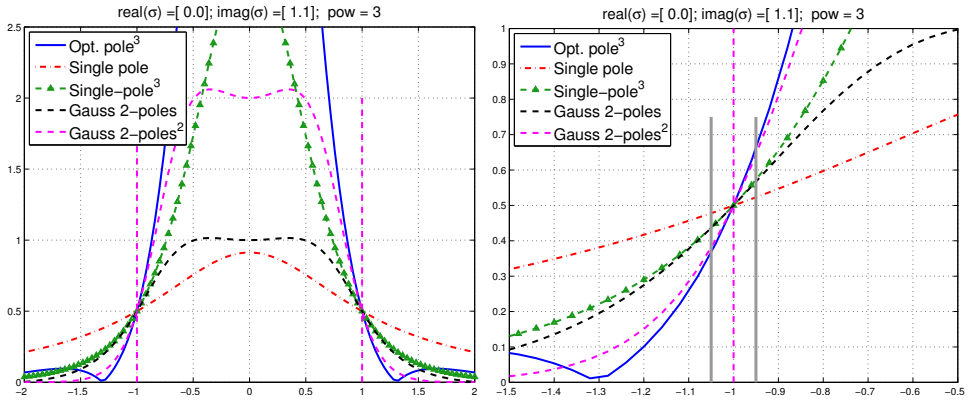

FIG. 2.3. *Various filters (left) and a zoom at a critical point (right).*

Figure 2.3 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 λ_is inside the interval $[-1, 1]$ is given by $|\mu_k/\mu_{m+1}|$. Then, clearly 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. In all this discussion, we have not made any reference to the other eigenvalues μ_k, those not located in the vicinity of 1/2. Based on this argument, the simplest and best 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 based on the interval $[-1,\ 1]$, that satisfies the requirement (2.10). Then we will call* 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:

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

Note that the sum is over all $2p$ poles so the 1/2 factor is necessary to get the actual real part needed in (2.5). 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 conjugation of the poles. 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*

$$\phi'(-1) = \frac{1}{2}\sum_{k=1}^{p} \frac{w_k}{1+\cos\theta_k} = \frac{1}{4}\sum_{k=1}^{p} \frac{w_k}{\cos^2\frac{\theta_k}{2}}. \tag{2.11}$$

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

[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.

common for schemes based on Gaussian quadrature. In the simplest case of the mid-point rule, the derivative is explicitly known.

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

Proof. From (2.6), we have $w_k = 1/p$ and $\theta_k = (2k-1)\pi/(2p)$, for $k = 1, \cdots, p$. Then (2.11) becomes:

$$\phi'(-1) = \frac{1}{2p} \sum_{k=1}^{p} \frac{1}{1+\cos\theta_k} \quad \text{with} \quad \theta_k = \frac{(2k-1)\pi}{2p}, \quad k = 1, \cdots, p. \tag{2.12}$$

Because of the symmetry in the roots $-\cos\theta_k$ with respect to the origin, we can write the sum in (2.11) (ignoring the 1/(2p) factor for now) as

$$\sum_k \frac{1}{1+\cos\theta_k} = \sum_k \frac{1}{1-\cos\theta_k}. \tag{2.13}$$

Each term $\cos\theta_k$ is a root of T_p, the Chebyshev polynomial of the first kind of degree p. This polynomial can be written as $T_p(t) = 2^{p-1}\prod(t - \cos\theta_k)$. It is clear that the derivative of $\log(T_p(t))$ at $t_0 = 1$ is equal to the sum (2.13). On the other hand, recalling that the derivative of $T_p(t)$ is $pU_{p-1}(t)$ where U_p is the p-th degree Chebyshev polynomial of the second kind, we obtain

$$\frac{d}{dt}\log(T_p(t)) = \frac{T_p(t)'}{T_p(t)} = \frac{pU_{p-1}(t)}{T_p(t)}.$$

It is easy to show (induction with recurrence formula or the definition using the cosine change of variables) that $U_{p-1}(1) = p$. Also $T_p(1) = 1$. In the end the sum (2.13) is equal to p^2. Dividing this by the factor $2p$ shows that the sum (2.12) is equal to $p/2$ and this completes the proof. □

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:*

$$\phi'(-1) \geq \frac{\frac{1}{2}s\sin s}{1-\sin(\frac{\pi}{2}\cos(s))} \approx \left(\frac{8p}{\pi^2}\right)^2. \tag{2.14}$$

Proof. Define $s_k = \frac{(2k-1)\pi}{2p}$, $k = 1, \cdots, p$. We start with the left part of expression (2.11)

$$\phi'(-1) = \frac{1}{2}\sum_{k=1}^{p} \frac{w_k}{1+\cos\theta_k} = \frac{1}{2}\sum_{k=1}^{p} \frac{s_1\sin s_k}{1+\cos(\frac{\pi}{2}(1+\cos s_k))}. \tag{2.15}$$

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 (2.15) 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 to obtain the approximation at the end of (2.14). □

Example. When $p = 8$ then $\phi'(-1)$ as calculated from (2.11) is 44.262. Its lower bound in (2.14) is 42.052 and the approximation of this term at the end of (2.14) is 42.049. This is a quite reasonable lower bound because the first term is much larger than the other ones in the sum (2.12). Had we taken 2 terms instead of one in the sum the lower bound would improve slightly to 43.618.

3. Least-squares rational filters. For a 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 approximation. 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.

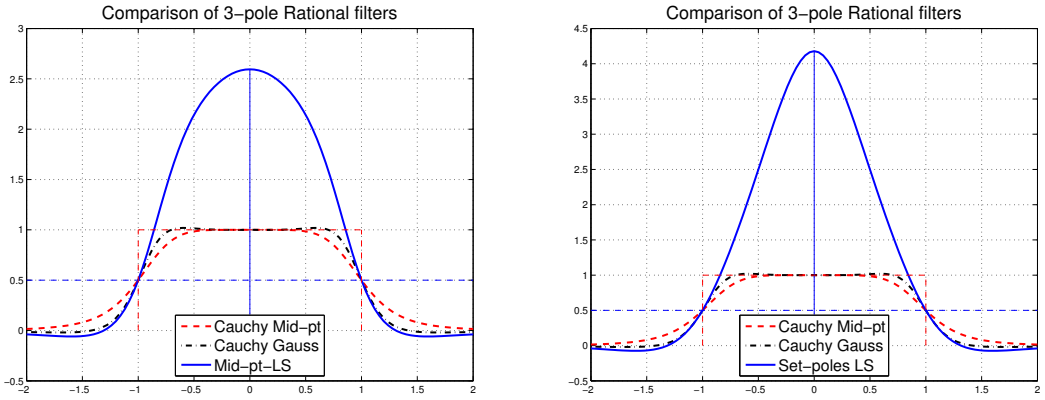

FIG. 3.1. *Left: Comparison of a standard rational function based on Cauchy integral using the mid-point rule with 3 poles (dashed red line), a least-squares rational filter (solid blue line) using the same poles, and a standard 3-pole Cauchy integral using Gaussian quadrature (black dash-dotted line). Right: the poles of the least-squares 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.*

Figure 3.1 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 Cauchy-mid-point filter (dashed red line) *based on the same poles* and about as well as the Cauchy-Gauss filter (black dash-dotted line) using also 3 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. The poles for the mid-point rule are likely to be advantageous relative to Gauss nodes if iterative solvers will be used.

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 least-squares 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 Cauchy-mid-point poles: $[(\pm\sqrt{3} + i)/2, i]$ or the Gaussian quadrature poles: $[\pm 0.9380 + 0.3467i, i]$. So, it is likely that linear systems to be solved with them by iterative methods will lead to faster convergence. Yet, the curves on the right also indicate that in all likelihood the subspace iteration will converge at a similar rate or somewhat faster than the Gauss method and certainly faster than the mid-point rule based method.

3.1. Obtaining least-squares rational filters. In least-squares rational filters we are given a number $2p$ of poles $\sigma_1, \sigma_2, \cdots, \sigma_{2p}$ and their related basis functions

$$\phi_j(z) = \frac{1}{z - \sigma_j}$$

and would like to find the function $\phi(z) = \sum_{j=1}^{2p} \alpha_j \phi_j(z)$ that minimizes

$$\int_{-\infty}^{\infty} w(t)|h(t) - \phi(t)|^2 dt,$$

where $h(t)$ is the step function $\chi_{[-1,1]}$ and $w(t)$ is some weight function. Here $\chi_X(t)$ is the characteristic function which takes the value one for $t \in X$ and zero elsewhere.

In practice the weight function $w(t)$ is taken to be of the form

$$w(t) = \begin{cases} 0 & \text{if} \quad |t| > a \\ \beta & \text{if} \quad |t| \leq 1 \\ 1 & \text{else} \end{cases} \tag{3.1}$$

where a, which is normally infinity, is typically set equal to 10 or larger, and the scalar β is small, e.g., $\beta = 0.01$. Additional details on the computation of the coefficients α_j can be found in the Appendix.

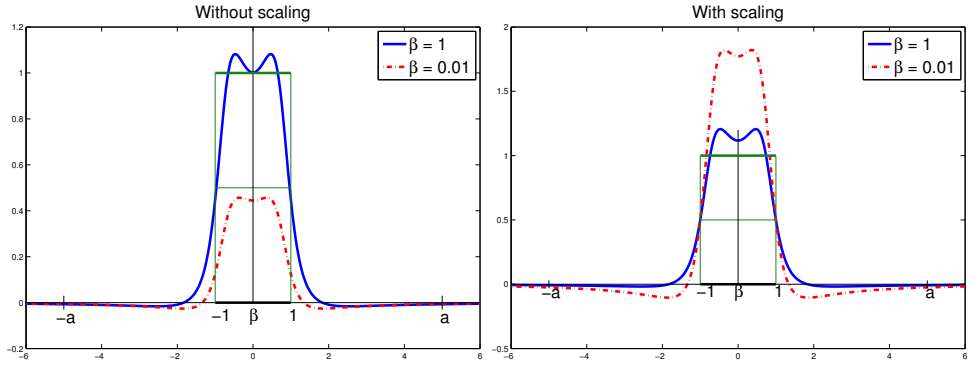

FIG. 3.2. *Comparison of two LS rational filters using the same poles located at* $[-0.75+0.5i, 0.75+0.5i]$ *but different* β*'s. Left: without scaling, right: scaling used so* $\phi(-1) = \frac{1}{2}$.

We now comment on the choice of β in (3.1). Figure 3.2 illustrates two least squares 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 3.3 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.

Another important advantage of least-squares 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, least-squares rational filters will also permit repeated poles (see next section), and this can be exploited by direct and iterative schemes alike.

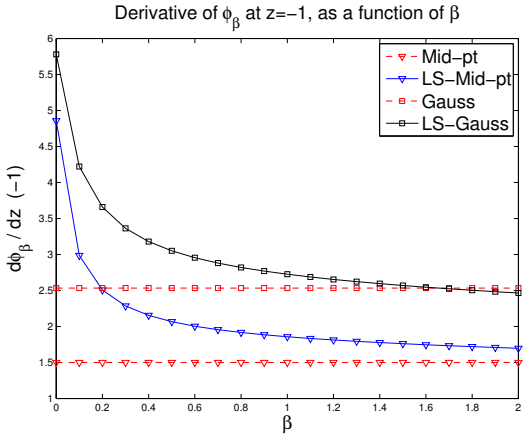

FIG. 3.3. *Comparison of separation factors associated with two rational filters based on Cauchy integral using the 3-pole mid-point rule and Gaussian quadrature rule, respectively, and least-squares rational filters using the same poles but different β's.*

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, for a single-pole function, repeating the pole k times would lead to a rational function 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

$$\phi(z) = \sum_{i=1}^{2p} \sum_{j=1}^{k_i} \frac{\alpha_{ij}}{(z-\sigma_i)^j}. \tag{3.2}$$

The hope is that repeating a pole would lead to more accurate filters, therefore faster convergence for subspace iteration, but that 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 pre-processing phase. When iterative solvers are employed, then there are some benefits in repeated poles as well. This will be discussed in the next section.

Figure 3.4(left) shows a comparison of least-squares repeated pole filters and Cauchy-type rational filters. The LS double pole filter (solid blue line) is obtained by doubling the pole located at $1.0i$, i.e., by repeating it twice. Based on the sharpness at the boundary points, we can conclude that this LS double pole filter will likely perform better than the Cauchy-Gauss filter based on the same pole (dashed red line) and about as well as the Cauchy-Gauss filter with two poles (dashed black line) if direct solvers are applied. However, this LS double pole filter is likely to 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 3.4 (Right) shows that the LS repeated pole filters become much sharper around the boundaries, -1 and 1, as the multiplicity of the pole increases.

Once the poles and their multiplicities are selected a filter of the general (3.2) is obtained. The coefficients α_{ij} are computed by a procedure that is similar to that of the single pole case.

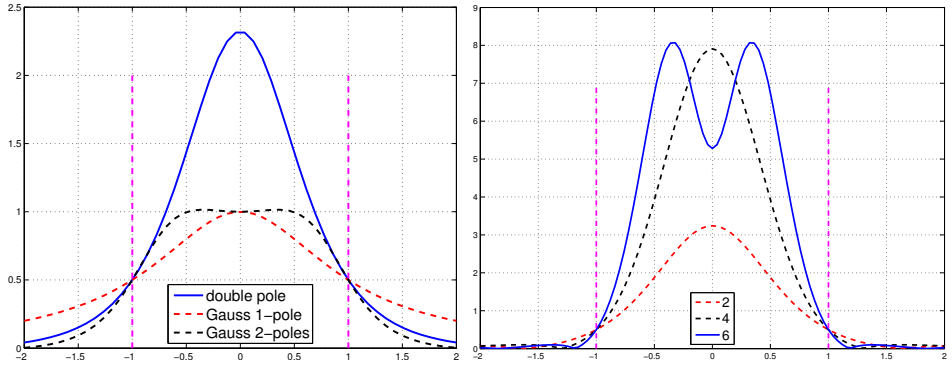

FIG. 3.4. *Left: Comparison of a least-squares repeated pole filter and two Cauchy-type filters using the Gaussian quadrature rule. The double pole filter is obtained from a single pole repeated twice. Right: Comparison of LS repeated pole filters obtained from repeating a pole located at* $1i$ *k times, where* $k = 2, 4, 6$.

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

4.1. Maximizing the use of real arithmetic. Apart from expensive factorization costs for large 3-D problems, direct solvers also suffer from the additional burden of complex arithmetic in the application of rational filters even when the original matrix A is real. However, iterative solvers, such as Krylov subspace methods, can bypass this drawback.

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

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

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 needs only to 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 (3.2), 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 repeated pole rational filter 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 [8, 13, 20, 24, 26]. In this paper, we improve the method proposed in [20] 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, $j = k, k-1, \cdots, 1$, to the linear systems $(A - \sigma I)x = b_j$, 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 [17] is used to compute an approximate solution x_k of the very first system $(A - \sigma I)x = b_k$. With (4.3), and assuming a zero initial guess, the approximate solution is of the form

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

where y_k is the mimimizer of

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

When $j = k-1, k-2, \cdots, 1$, 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 $(A-\sigma I)^{-1}b_j$, $j=1,\cdots,k-1$. 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:

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

where $\hat{z}_j$ denotes 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, $j=1,\cdots,k$, is solved through an $(m+1)\times m$ least squares 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 only need to 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 [2]. 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 [19]. Since our main task is to annihilate the component of b in the complementary eigenspace, we can pre-process the right-hand side by a polynomial filter [5] 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., [21], to deflate it from the search space. Locking essentially amounts to freezing the converged eigenvectors and continuing to iterate only on the non-converged 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 [15, 25] that this locking strategy is also quite useful in identifying eigenvalues that are highly clustered or of very high multiplicity.

Algorithm 1 subspace iteration with locking and rational filtering

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

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 $q_1, \cdots, q_j$. 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.

5. Numerical Experiments. In this section we present some experiments to illustrate the performance of the subspace iteration with least-squares 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 4 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 5.1(left) displays these filters. We then run the subspace iteration algorithm combined with these filters and plot the maximum residual at each iteration in Figure 5.1(right). During the iteration, all the shifted linear systems are solved by a direct method and the subspace dimension is fixed at 62 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.

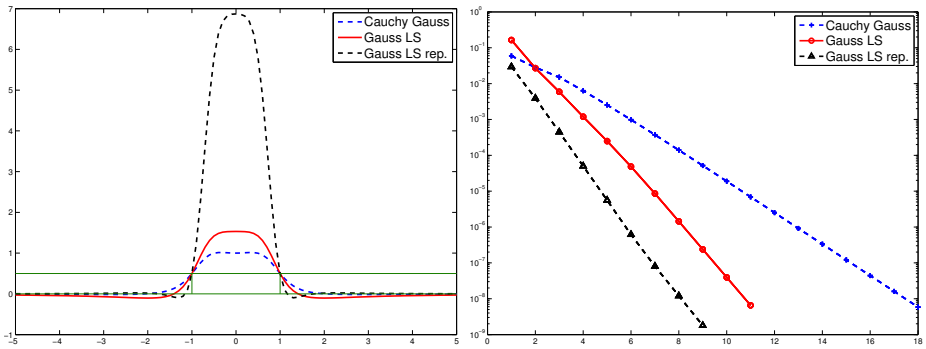

FIG. 5.1. *Comparison of Cauchy and least-squares rational filters to compute the smallest* 56 *eigenpairs for a 2D discrete Laplacian. Left: Filters; Right: Convergence history.*

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 [3] 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 mid-point quadrature rule.
- `Gauss-Cheby/LS-Mid-pt+GMRES(60)`: These 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 least-squares 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 5.1 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.

TABLE 5.1
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

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 5.2. 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 pre-processing 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 iterations 60 is reached.

TABLE 5.2
Computing 145 *eigenpairs inside* $[0.0, 0.2]$ *for the 3D discrete Laplacian example. The first five methods use a direct solver and the last six stop the iteration of the inner solves when either the relative residual norm is reduced by a factor of* 10^{-9} *or the maximum number of iterations* 60 *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

From Table 5.2, 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 3 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 mid-point rule converges faster than that based on the Gauss Chebyshev quadrature, which is in sharp contrast to 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 2.2) and this facilitates the inner iterative solves. Third, the LS repeated pole filters 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 5.1. 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 two optimal methods from each category in Table 5.2. As can be seen in Table 5.3, the performance of the `LS-repeated(6,150)` method is the best according to computational time.

TABLE 5.3
Computing 208 *eigenpairs inside* $[0.4, 0.5]$ *for the 3D discrete Laplacian. The first two 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	23	328.36	1524.30	1852.66
	5	7	489.99	915.18	1405.17
`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

The polynomial preconditioning technique proposed in Section 4.3 is now tested on this problem. The polynomial filter built for these tests is different from the one in [5]. It is obtained by approximating the Dirac-δ function so that the maximum value of the approximation inside the reference interval $[-1, 1]$ is 10 times larger than its values at 1 and -1. 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 5.4. 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 selected automatically by the algorithm. The two `LS-repeated` approaches in Table 5.4 first perform 4 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 5.4. 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 application.

TABLE 5.4

Computing 208 *eigenpairs inside* $[0.4, 0.5]$ *for the 3D 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

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 5.5.

TABLE 5.5

Computing 345 *eigenpairs inside* $[0.9, 1.0]$ *for the 3D 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

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 5.6 and the method in the fourth row, which performs the polynomial filtered subspace iteration in the first 5 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.

TABLE 5.6
Computing 345 *eigenpairs inside* $[0.9, 1.0]$ *for the 3D 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

5.2.3. A comparison with `ARPACK`. `ARPACK` [12] 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 5.1. 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 5.7, 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.

TABLE 5.7
Results for `ARPACK` *[12] to compute the eigenpairs of the 3D discrete Laplacian in Table 5.1.*

$[\mu, \xi]$	$[a, \xi]$	#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

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 iteration. Second, `ARPACK` will typically require much more storage. Indeed, for interior eigenvalue problems it necessitates to compute unneeded redundant eigenpairs as shown above, and it also requires the dimension of the Krylov subspace to be a few times larger than the number of computed eigenvalues, e.g., a factor 2 is recommended in the user manual, to guarantee convergence.

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 [10]. 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 5.8. The search intervals are selected to include the eigenvalues requested by the Time Dependent Density Functional Theory (TDDFT) application [5]. 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.

TABLE 5.8
Hamiltonians from the University of Florida Sparse Matrix Collection [4].

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	$[-1.2264, 32.7031]$	$[-0.65, -0.0096]$	250	0.0356
$Si_{41}Ge_{41}H_{72}$	185,639	15,011,265	$[-1.1214, 49.8185]$	$[-0.64, -0.00282]$	212	0.0220
$Si_{87}H_{76}$	240,369	10,661,631	$[-1.1964, 43.0746]$	$[-0.6600, -0.030]$	212	0.0264
$Ga_{41}As_{41}H_{72}$	268,096	18,488,476	$[-1.2502, 1300.93]$	$[-0.64, 0.0]$	201	0.0010

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 5.9. 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.

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

TABLE 5.9
Comparison results for the Hamiltonians in Table 5.8.

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

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 5.2, 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 6 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.

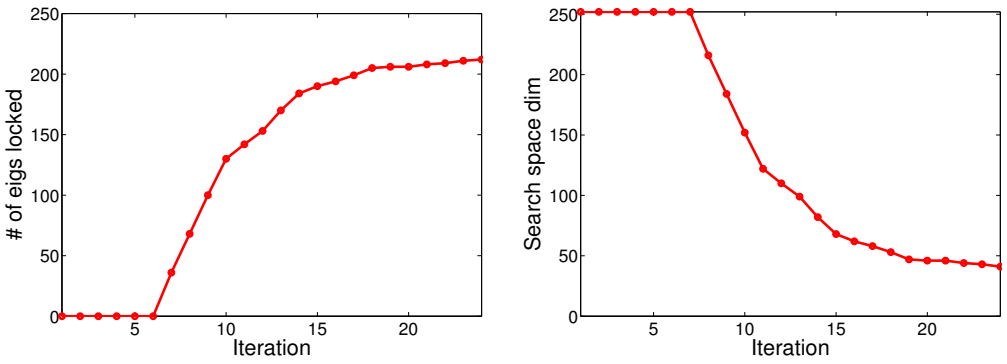

FIG. 5.2. *Convergence of the rational filtered subspace iteration for* $Si_{87}H_{76}$. *Left: Number of eigenpairs locked at each iteration. Right: Search space dimension at each iteration.*

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 least-squares 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.

A number of improvements can be made to the proposed scheme and these will be explored in our future work. For example, we plan to study the relation between the inner and outer stopping tolerances and to explore the benefits of replacing the standard single vector iterative methods in the inner solvers, by block Krylov subspace methods. Another broad avenue that is worth investigating is to explore the possibility of selecting optimal poles.

Appendix. Let h be the step function having value one in an interval $[-1,\ 1]$ and zero elsewhere. We are interested in the best approximation to this function by rational functions having poles $\sigma_1, \cdots, \sigma_{2p}$. In addition, we now consider all the poles, i.e., both those above and below the real axis and will assume that no pole is on the real axis and that the poles come in conjugate pairs. The original objective function to minimize is

$$\left\| h - \sum_i \alpha_i \phi_i \right\|_w^2, \quad \text{where} \quad \phi_i(t) = \frac{1}{t - \sigma_i}. \tag{6.1}$$

The norm in (6.1) is associated with the standard L-2 inner product

$$\langle f, g \rangle = \int_{-\infty}^{\infty} w(t) f(t) \overline{g(t)} dt. \tag{6.2}$$

Here, $w(t)$ is a weight function given by (3.1) where β is a positive parameter, and a is taken to be a large number, or just $a = +\infty$. Though not considered here, it is also possible to consider an inner product without the conjugations by exploiting symmetry of the poles with respect to the real axis. The general least-squares theory states that linear combination $\sum \alpha_j \phi_j$ of functions in (6.1) is optimal if and only if

$$\left\langle h - \sum_j \alpha_j \phi_j, \phi_i \right\rangle = 0 \quad i = 1, \cdots 2p. \tag{6.3}$$

Let G the $2p \times 2p$ matrix with entries $g_{ij} = \langle \phi_j, \phi_i \rangle = \overline{\langle \phi_i, \phi_j \rangle}$ and η the vector with entries $\eta_i = \langle h, \phi_i \rangle, i = 1, \cdots, 2p$. Then the optimal α is the solution of the system

$$G\alpha = \eta. \tag{6.4}$$

Let us evaluate a term g_{ij} when $i \neq j$.

$$< \phi_i, \phi_j >= \int_{-\infty}^{+\infty} w(t) \phi_i(t) \overline{\phi_j(t)} dt = \int_{-\infty}^{+\infty} \frac{w(t)}{(t - \sigma_i)(t - \bar{\sigma}_j)}\ dt. \tag{6.5}$$

Similarly,

$$\eta_j =< h, \phi_j >= \int_{-\infty}^{+\infty} \frac{w(t) h(t)}{t - \bar{\sigma}_j}\ dt. \tag{6.6}$$

Due to the nature of the poles, the matrix and right-hand side have a special structure. If we consider the first half of the poles $\sigma_1, \sigma_2, \cdots, \sigma_p$, and assume that the other half are their

conjugates, i.e., $\sigma_{p+k} = \bar{\sigma}_k$. Then clearly the matrix G and the right-hand side η in (5.5) have the following block structure:

$$G = \begin{pmatrix} A & B \\ \bar{B} & \bar{A} \end{pmatrix}, \qquad \eta = \begin{pmatrix} \eta_u \\ \bar{\eta}_u \end{pmatrix}. \tag{6.7}$$

In addition, the matrix A is Hermitian while B is symmetric (complex), so that $\bar{B} = B^H$. The matrix G is clearly Hermitian positive definite. It is easy to see that the solution to this system is of the form $\alpha = \begin{pmatrix} \alpha_u \\ \bar{\alpha}_u \end{pmatrix}$. This is expected due to the symmetry of the problem.

REFERENCES

[1] M. E. CASIDA, *Time-Dependent Density Functional Response Theory for Molecules*, WORLD SCIENTIFIC, 2011, ch. 5, pp. 155–192.

[2] E. Chow and Y. Saad, *Experimental study of ILU preconditioners for indefinite matrices*, J. Comput. Appl. Math, 86 (1997), pp. 387–414.

[3] T. Davis, *Algorithm 832: UMFPACK V4.3—an unsymmetric-pattern multifrontal method*, ACM Trans. Math. Softw., 30 (2004), pp. 196–199.

[4] T. Davis and Y. Hu, *The University of Florida Sparse Matrix Collection*, ACM Trans. Math. Softw., 38 (2011), pp. 1:1–1:25.

[5] 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.

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

[7] L. Hedin, *New method for calculating the one-particle green's function with application to the electron-gas problem*, Phys. Rev., 139 (1965), pp. A796–A823.

[8] L. Hoffnung, R. Li, and Q. Ye, *Krylov type subspace methods for matrix polynomials*, Linear Algebra Appl., 415 (2006), pp. 52 – 81. Special Issue on Large Scale Linear and Nonlinear Eigenvalue Problems.

[9] L. Komzsik and T. Rose, *Substructuring in MSC/NASTRAN for large scale parallel applications*, Computing Systems in Engineering, 2 (1991), pp. 167–173.

[10] Leeor Kronik, Adi Makmal, Murilo L. Tiago, M. M. G. Alemany, Manish Jain, Xiangyang Huang, Yousef Saad, and James 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), p. 1063 1079.

[11] B. Lang. Private communication.

[12] 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/`.

[13] 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.

[14] E. Lorin de la Grandmaison, S. B. Gowda, Y. Saad, M. L. Tiago, and J. R. Chelikowsky, *Efficient computation of the coupling matrix in time-dependent density functional theory*, Computer Physics Communications, 167 (2005), pp. 7–22.

[15] 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.

[16] A. S. McLeod, P. Doak, S. Sharifzadeh, and J. B. Neaton, *Berkeley GW*, Jan 2011.

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

[18] E. Polizzi, *Density-matrix-based algorithm for solving eigenvalue problems*, Phys. Rev. B, 79 (2009), p. 115112.

[19] 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.

[20] ———, *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., Boston, 1990, Birkhauser, pp. 401–410.

[21] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, rev. ed., SIAM, Philadelphia, 2011.

[22] T. Sakurai and H. Sugiura, *A projection method for generalized eigenvalue problems using numerical integration*, J. Comput. Appl. Math., 159 (2003), pp. 119 – 128. Japan-China Joint Seminar on

Numerical Mathematics; In Search for the Frontier of Computational and Applied Mathematics toward the 21st Century.

[23] T. Sakurai and H. Tadano, *CIRR: a Rayleigh-Ritz type method with contour integral for for generalized eigenvalue problems*, Hokkaido Mathematical Journal, 36 (2007), pp. 745–757.

[24] 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.

[25] 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.

[26] H. van der Vorst, *Iterative Krylov Methods for Large Linear Systems*, Cambridge University Press, Cambridge, 2003.