SIAM J. SCI. COMPUT.
Vol. 37, No. 3, pp. A1365–A1387

COMPUTING EIGENVALUES OF REAL SYMMETRIC MATRICES WITH RATIONAL FILTERS IN REAL ARITHMETIC*

ANTHONY P. AUSTIN† AND LLOYD N. TREFETHEN†

Abstract. Powerful algorithms have recently been proposed for computing eigenvalues of large matrices by methods related to contour integrals; best known are the works of Sakurai and coauthors and Polizzi and coauthors. Even if the matrices are real symmetric, most such methods rely on complex arithmetic, leading to expensive linear systems to solve. An appealing technique for overcoming this starts from the observation that certain discretized contour integrals are equivalent to rational interpolation problems, for which there is no need to leave the real axis. Investigation shows that using rational interpolation per se suffers from instability; however, related techniques involving real rational filters can be very effective. This article presents a technique of this kind that is related to previous work published in Japanese by Murakami.

Key words. real symmetric matrix, eigenvalues, rational interpolation, rational filter, Rayleigh–Ritz, contour integral, spectral projection, Sakurai–Sugiura, FEAST

AMS subject classifications. 15A18, 41A20, 65F15

DOI. 10.1137/140984129

1. Introduction.

Let $A \in \mathbb{C}^{N\times N}$ be a large square matrix, and consider the problem of computing the eigenvalues of A that lie within a given region of the complex plane. Some of the most successful techniques for attacking this problem are based on projecting the matrix A onto an approximately invariant subspace associated with the eigenvalues of interest and computing the eigenvalues of the projection. Of these, perhaps the best known are the Krylov subspace techniques such as the implicitly restarted Arnoldi method [39], which is implemented in the widely used software package ARPACK [20].

Recently, a new class of algorithms has been proposed which derive their projections from complex contour integrals. Though early traces of these ideas can be found in the works of Goedecker on linear-scaling methods for electronic structure calculation [11, 12], the best-known algorithms of this type are the Sakurai–Sugiura (SS) method [37] and the FEAST algorithm, due to Polizzi [32]. A major computational advantage offered by these algorithms is that they are very easily parallelizable.

Some of the most important eigenvalue problems involve matrices A that are real and symmetric. For large such problems, it is natural to want to take advantage of the parallelism offered by contour integral methods; however, by their very nature, these methods require complex arithmetic even though the sought-after eigenvalues are real. Aside from the fact that the need to use complex arithmetic to solve a real problem is conceptually jarring, this means that methods based on contour integrals suffer roughly a factor of two penalty in both time and storage costs, compared with approaches that rely only on real arithmetic.

*Submitted to the journal's Methods and Algorithms for Scientific Computing section August 28, 2014; accepted for publication (in revised form) March 2, 2015; published electronically June 2, 2015. This work was supported by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013)/ERC grant agreement 291068. The views expressed in this article are not those of the ERC or the European Commission, and the European Union is not liable for any use that may be made of the information contained here.

http://www.siam.org/journals/sisc/37-3/98412.html

†University of Oxford, Mathematical Institute, Oxford OX2 6GG, UK (austin@maths.ox.ac.uk, trefethen@maths.ox.ac.uk).

In this paper, we present a technique that addresses this deficiency. Our approach is motivated by the connection between the SS method and rational interpolation established in [1] and can be succinctly described as a projection method which uses a rational filter with only real poles. Projection methods based on rational filters have been examined in the Japanese literature by Murakami [22, 23], whose work we discuss in some detail later on.

The remainder of this paper is organized as follows. In section 2, we review the connection between the SS method and rational interpolation, introduce the idea of using the latter to find eigenvalues, and show how it yields a method with SS-like parallelism that uses only real arithmetic. Unfortunately, as we will see in section 3, methods based directly on rational interpolation are numerically unstable, but in section 4, we will show how they can be reformulated to avoid this difficulty by using a Rayleigh–Ritz procedure and rational filters. Section 5 contains information pertinent to developing practical realizations of our method. In section 6, we give a summary of the proposed algorithm, and finally, in section 7, we illustrate the performance of our method on a numerical example.

While our main application is to real symmetric matrices A, much of our discussion does not depend directly on this structure. Accordingly, we will assume most of the time that A is arbitrary and specialize to the real symmetric or Hermitian case when appropriate.

2. Finding poles of the resolvent. The methods we consider are rooted in the fact that the eigenvalues of a matrix A are the poles of its resolvent $(A - zI)^{-1}$. Given vectors $u, v \in \mathbb{C}^N$, we consider the function

$$f(z) = u^*(A - zI)^{-1}v, \tag{2.1}$$

a "scalarized" version of $(A - zI)^{-1}$. (If A is Hermitian, it is common to take $u = v$.) If A is diagonalizable (an assumption we will make throughout the paper), we can write A in an eigenvalue decomposition as

$$A = \sum_{h=1}^{N'} \lambda_h P_h,$$

where $\lambda_1, \ldots, \lambda_{N'}$ are the distinct eigenvalues of A and $P_1, \ldots, P_{N'}$ are the corresponding spectral projectors. Then, f takes the form

$$f(z) = \sum_{h=1}^{N'} \frac{u^* P_h v}{\lambda_h - z}.$$

Thus, f is a rational function, and if v and u are generic in the sense that v (respectively, u) is not orthogonal to any of the right (respectively, left) eigenspaces of A, then f will have a simple pole at each of the points λ_h. Our goal will be to compute the poles of this function that lie within a given region of interest.

2.1. The Sakurai–Sugiura method. The original method proposed by Sakurai and Sugiura in [37] computes the poles of f within a region $\Omega \subset \mathbb{C}$ bounded by a simple, closed, piecewise smooth curve γ by applying the derivative-free variant of the pole-finding algorithm of Kravanja and Van Barel [1, 18, 19]. If A has $s \leq N'$

distinct eigenvalues within Ω (a number which, for the time being, we will assume is known) this algorithm works by computing the moment integrals

$$\mu_j = \frac{1}{2\pi i}\int_\gamma z^j f(z)\,dz, \qquad j = 0, \ldots, 2s-1, \tag{2.2}$$

and using them to form the $s \times s$ Hankel matrices

$$H_s = \begin{bmatrix} \mu_0 & \mu_1 & \cdots & \mu_{s-1} \\ \mu_1 & \mu_2 & \cdots & \mu_s \\ \vdots & \vdots & & \vdots \\ \mu_{s-1} & \mu_s & \cdots & \mu_{2s-2} \end{bmatrix}, \qquad H_s^< = \begin{bmatrix} \mu_1 & \mu_2 & \cdots & \mu_s \\ \mu_2 & \mu_3 & \cdots & \mu_{s+1} \\ \vdots & \vdots & & \vdots \\ \mu_s & \mu_{s+1} & \cdots & \mu_{2s-1} \end{bmatrix}.$$

The poles of f within γ are then given by the eigenvalues of the generalized eigenvalue problem for the pencil $H_s^< - \lambda H_s$.

Because it makes use of these Hankel matrices, we will refer to this method as the SS-H method.

In practice, the integrals (2.2) cannot be computed exactly and must be approximated using a quadrature rule. If this rule is defined by nodes $z_0, \ldots, z_{K-1} \in \mathbb{C}$ and corresponding weights $w_0, \ldots, w_{K-1} \in \mathbb{C}$, then we obtain the approximation

$$\mu_j \approx \sum_{k=0}^{K-1} w_k z_k^j u^*(A - z_k I)^{-1} v. \tag{2.3}$$

The dominant contribution to the computational cost of applying this method comes from solving the K linear systems involving shifts of A at each quadrature node required to compute (2.3). If A is large, these linear solves are potentially expensive; however, as each system is independent of the others, they can be solved in parallel.

2.2. Rational interpolation. An alternative method for computing the poles of f is to form a rational approximation to f and compute the poles of the approximation. The simplest type of rational approximation is a rational interpolant. Given K points $z_0, \ldots, z_{K-1} \in \mathbb{C}$, we seek polynomials p and q of appropriately chosen maximum degrees m and n such that

$$\frac{p(z_k)}{q(z_k)} = f(z_k), \qquad k = 0, \ldots, K-1.$$

In practice, it is easier to multiply through by the denominator and work with the linearized conditions

$$p(z_k) = f(z_k)q(z_k), \qquad k = 0, \ldots, K-1. \tag{2.4}$$

There are K interpolation conditions in total, and since these conditions only determine p and q up to a common constant factor, we must additionally impose a normalization condition. Balancing these $K+1$ constraints with the $m+n+2$ degrees of freedom in p and q, we see that we must choose m and n such that $m+n+1 = K$. Obviously, the degree of q should be at least as large as the number of eigenvalues sought. As we are assuming for now that this number is known, we will take n to be exactly equal to it. In practice, n will need to be selected based on an estimate of this number, and we note in particular that it may be advantageous to make n larger,

even if the number is known exactly; see section 5.2. Once we have computed the interpolant, finding the poles is simply a matter of finding the roots of q, which can be accomplished by solving an eigenvalue problem, the structure of which depends on the basis chosen to represent q.

In terms of computational structure, rational interpolation is very similar to the SS-H method. The dominant cost involved comes from solving the K linear systems required to evaluate f at each of the interpolation nodes, and, just as in the SS-H method, these systems can be solved in parallel. Also as in the SS-H method, obtaining the approximations to the eigenvalues of A boils down to solving a small, dense eigenvalue problem.

In fact, it turns out that the SS-H method and rational interpolation are mathematically equivalent (in exact arithmetic) in the most basic case where (1) the eigenvalues sought are those in the unit disc, (2) the contour integrals in the SS-H method are discretized using the trapezoidal rule in the roots of unity, and (3) those same roots are used for the interpolation nodes when constructing the rational interpolant. This was shown in [1], where the observation takes the form of a theorem asserting the equivalence of rational interpolation and the derivative-free Kravanja–Van Barel method. This equivalence, combined with the success enjoyed by contour integral methods in practice, is one of the main reasons we have been motivated to consider approaches to eigenvalue computation based on rational interpolation.

2.3. Rational interpolation on a real interval. Though the SS-H method and rational interpolation are very similar, as just observed, they provide different perspectives on how to approach eigenvalue computation. In particular, rational interpolation schemes are not constrained by the need to worry about contours, regions, or quadrature rules; there are only interpolation nodes. This simpler structure naturally leads one to consider schemes that are not so easily conceived in a framework based strictly on contour integrals.

Suppose that A is a Hermitian matrix, so that the eigenvalues of A are all real, and suppose that we seek the eigenvalues that lie in a given interval $I \subset \mathbb{R}$. For concreteness, let us suppose that $I = [-1, 1]$. Applying a contour integral method requires the selection of a contour in the complex plane that encloses I. The unit circle is an obvious choice and is often used, as are long, narrow ellipses that enclose I. Both of these choices of contour can be used as the basis for a successful algorithm.

If, in addition, A is real, however, this method has an unfortunate defect. As any reasonable choice of quadrature rule for approximating the contour integrals will have nodes that do not lie on the real axis, we must use complex arithmetic to solve a real eigenvalue problem. Complex operations take roughly twice as much work to perform as real operations, and storing complex matrices takes twice as much memory as storing real ones.

Rational interpolation affords us a way out of this problem. Unlike in a contour integral framework, we are not forced by the demands of a quadrature rule to take some evaluation points off the real axis. Instead, we are free to take the interpolation nodes all to lie in I. If we do this, a natural candidate for a good set of nodes is a set of Chebyshev points in I [42]. In this paper, we will work with the K *Chebyshev points of the first kind*, defined by

$$x_k = \cos\left(\frac{(2k+1)\pi}{2K}\right), \qquad k = 0, \ldots, K-1. \tag{2.5}$$

Later on, we will also make use of the *Chebyshev polynomials of the first kind*, denoted by T_j, which are defined by $T_0(x) = 1$, $T_1(x) = x$, and, for $j \geq 2$, by the recurrence

$$T_{j+1}(x) = 2xT_j(x) - T_{j-1}(x).$$

The zeros of T_K are exactly the first-kind Chebyshev points x_k just defined.

Thus, we arrive at the following simple algorithm: find p and q that satisfy (2.4) with $z_k = x_k$ and then compute the roots of q. Since all the points x_k are real, only linear systems involving real shifts are solved to evaluate the scalarized resolvent f. If A is real, only real arithmetic is employed.

All that remains is to decide how to solve the linearized rational interpolation problem (2.4). Rational interpolation has a propensity, well-known to practitioners, for rounding errors to cause spurious pole-zero pairs (sometimes called *Froissart doublets*) to appear when the interpolant is computed in the obvious way [42]. To combat this, we use the algorithm presented in [13], which combines the earlier algorithm of [29] with a regularization technique based on the SVD to detect and remove these spurious pairs. This algorithm is implemented in the MATLAB code `ratinterp` within the freely available Chebfun software package [6].

3. Instability of rational interpolation for finding eigenvalues. Unfortunately, the simple algorithm just described suffers from numerical instability. The difficulty is that the eigenvalues of A in $[-1, 1]$ may be distributed in such a manner that the polynomial rootfinding problem set up by the interpolation process is poorly conditioned. This problem can occur even if the number of eigenvalues of A in $[-1, 1]$ is small and the eigenvalue problem itself is well-conditioned (which it always is if A is Hermitian), as we now illustrate.[1]

Suppose that A is a 12×12 diagonal matrix with diagonal entries -10, 10, and $0, 0.1, 0.2, \ldots, 0.9$ and that we wish to compute the 10 eigenvalues of A inside $[-1, 1]$. Using the `ratinterp` code just mentioned, we can implement the algorithm of the previous section in just a few lines of MATLAB and Chebfun as follows:

```
A = diag([0:0.1:0.9, -10, 10]); I = eye(12); v = randn(12, 1);
K = 32; xk = chebpts(K, 1); fk = zeros(K, 1);
for k = 1:K, fk(k) = v'*((A - xk(k)*I) \ v); end
[p, q, r, mu, nu, pol] = ratinterp(fk, K - 11, 10, K, 'type1', 0);
```

The first line of this code creates the matrix A and generates the random vector v that we will use to scalarize the resolvent. The second line uses the Chebfun code `chebpts` to create a vector of $K = 32$ Chebyshev points (2.5) in $[-1, 1]$. The third line computes the values of the scalarized resolvent function at each interpolation point, storing the results in a vector `fk`. Finally, the fourth line computes a linearized rational interpolant to these values with a denominator degree of 10. The `'type1'` argument to `ratinterp` specifies that we are working with data from a first-kind Chebyshev grid, and the final `0` argument disables the aforementioned SVD-based robustness techniques, which are not necessary for this demonstration. The poles that form our computed approximations to the eigenvalues are stored in the output variable `pol`.

Approximations to the eigenvalues produced by a typical run of this code are displayed in Table 1. The eigenvalues at 0 and 0.9 are computed to only about

[1] This example was suggested by Grady Wright.

TABLE 1
Eigenvalues and absolute errors illustrating instability of eigenvalue computation via rational interpolation for the test problem considered in the text.

Exact	Computed	Error
0.0	0.00000240	2.40×10^{-6}
0.1	0.09992181	7.82×10^{-5}
0.2	0.20039592	3.96×10^{-4}
0.3	0.27060498	2.94×10^{-2}
0.4	0.40129619	1.30×10^{-3}
0.5	0.48977909	1.02×10^{-2}
0.6	0.59984376	1.56×10^{-4}
0.7	0.70009663	9.66×10^{-5}
0.8	0.79959982	4.00×10^{-4}
0.9	0.89999995	4.51×10^{-8}

five and seven digits of accuracy, respectively, and the accuracy is even worse for the eigenvalues in the middle of the spectrum. Increasing the value of K does not improve the accuracy, nor does enabling the SVD-based robustness techniques in `ratinterp`. The computation has been spoiled by rounding error.

The reason for this behavior becomes apparent when we look at the denominator polynomial of the rational interpolant, which is depicted in Figure 1. Observe how the polynomial is relatively large on the left half of the interval, while on the right half, where the spectrum of A is concentrated, it is nearly zero. (In fact, the values on the right half fluctuate near 10^{-8}.) This large scaling of the polynomial over the entire interval relative to its size on the half of the interval containing the spectrum causes its roots to be badly conditioned, resulting in the poor approximations to the eigenvalues we see in the table. In the polynomial rootfinding literature, this is sometimes referred to as a "dynamic range" problem [3].

The example just given is admittedly manufactured; however, it is easy to imagine that similar situations could arise in actual applications. Unfortunately, there seems to be little that can be done to prevent this without some a priori knowledge of the spectrum of A. A better solution is to avoid polynomial rootfinding altogether. Thus, we now modify the approach based on rational interpolation by turning to a method based on Rayleigh–Ritz procedures and rational filters.

4. Rayleigh–Ritz reformulation using rational filters.

4.1. The SS-RR method. Sakurai and coworkers noticed similar instabilities to those just described in the SS-H algorithm of [37] and devised an alternative version

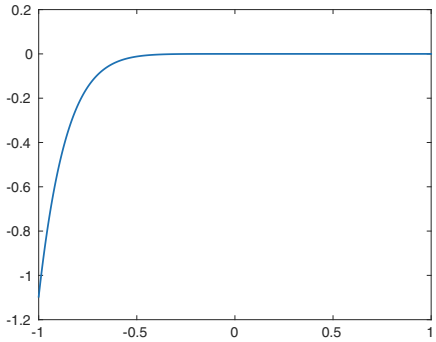

FIG. 1. *Plot of the denominator of the rational interpolant whose roots were computed to produce the eigenvalue approximations in Table* 1.

of their method based on a Rayleigh–Ritz procedure to correct this [15, 38]. When applied to Hermitian matrices, their algorithm works as follows. Given a vector v that is generic in the sense defined in section 2, the method computes a basis $\{v_0, \ldots, v_{s-1}\}$ for the invariant subspace corresponding to the eigenvalues of A within γ by computing the projections of the vectors $A^j v$ onto this eigenspace via the integrals

$$v_j = \frac{1}{2\pi i} \int_\gamma z^j (A - zI)^{-1} v \, dz, \qquad j = 0, \ldots, s-1. \tag{4.1}$$

The vectors v_j are then orthonormalized, and the resulting vectors are gathered as columns into a matrix Q. The desired eigenvalues and eigenvectors are then obtained by solving the $s \times s$ eigenvalue problem for the matrix $Q^* A Q$.

Because of its use of a Rayleigh–Ritz procedure, we will refer to this method as the SS-RR method.

4.2. Contour integrals and filter functions. The key mechanism underlying the SS-H and SS-RR methods—and, indeed, all contour integral methods—is that the contour integrals (2.2) and (4.1) compute a projection of a vector onto the eigenspace of interest, since

$$P = -\frac{1}{2\pi i} \int_\gamma (A - zI)^{-1} \, dz \tag{4.2}$$

is the spectral projector associated with the eigenspace corresponding to the eigenvalues of A contained within γ [17]. When we discretize (4.2) using a quadrature rule defined by K distinct nodes $z_0, \ldots, z_{K-1}$ and corresponding weights $w_0, \ldots, w_{K-1}$, we obtain an approximate projector

$$\widehat{P} = \sum_{k=0}^{K-1} w_k (A - z_k I)^{-1}. \tag{4.3}$$

Written another way, we have $\widehat{P} = H(A)$, where H is the rational function

$$H(z) = \sum_{k=0}^{K-1} \frac{w_k}{z - z_k}. \tag{4.4}$$

We call H the *filter function* associated with the method because it describes how (4.3) acts to filter out undesired eigenvectors while retaining the rest. The points z_k are the *poles* of the filter, and the w_k are the corresponding *residues*. If λ is an eigenvalue of A for which $H(\lambda)$ is small, then when $\widehat{P}$ acts on a vector v, it will reduce the components of v in the directions of eigenvectors of A corresponding to λ.

Contour integral methods have been discussed from the viewpoint of rational filters in several places in the literature. Sakurai and coworkers do this for the SS-H method in [16] and for SS-RR in [15]. Tang and Polizzi do the same for FEAST in [40].

4.3. Filters derived from rational interpolation. We will now show that the process of rational interpolation described in section 2.2 for our scalarized resolvent function also acts to filter the spectrum of A by a rational function. This is clear in the case where the result from [1] mentioned at the end of section 2.2 is applicable,

for under those conditions, rational interpolation is equivalent to the SS-H method based on discretized contour integrals, and we have just shown that all discretized contour integrals have a rational filter behind them. The easiest way to see that this is true for other choices of the interpolation nodes is to extend the result from [1] to a more general setting.

Specifically, let $f : \mathbb{C} \to \mathbb{C} \cup \{\infty\}$ be a meromorphic function, let $z_0, \ldots, z_{K-1}$ be K distinct points in $\mathbb{C}$ that are not poles of f, and let $w_0, \ldots, w_{K-1} \in \mathbb{C}$ be nonzero. Let $n \geq 1$, and consider the following two computational procedures:

- Procedure (K):
  1. Compute the quantities
  $$\mu_j = \sum_{k=0}^{K-1} w_k z_k^j f(z_k), \qquad j = 0, \ldots, 2n-1.$$
  2. Form the Hankel matrices
  $$H_n = \begin{bmatrix} \mu_0 & \mu_1 & \cdots & \mu_{n-1} \\ \mu_1 & \mu_2 & \cdots & \mu_n \\ \vdots & \vdots & & \vdots \\ \mu_{n-1} & \mu_n & \cdots & \mu_{2n-2} \end{bmatrix}, \qquad H_n^< = \begin{bmatrix} \mu_1 & \mu_2 & \cdots & \mu_n \\ \mu_2 & \mu_3 & \cdots & \mu_{n+1} \\ \vdots & \vdots & & \vdots \\ \mu_n & \mu_{n+1} & \cdots & \mu_{2n-1} \end{bmatrix}.$$
  3. Compute the eigenvalues, counted according to multiplicity, of the matrix pencil $H_n^< - \lambda H_n$.
- Procedure (R):
  1. Compute the $J \leq K-1$ zeros $\eta_0, \ldots, \eta_{J-1}$ of the rational function
  $$H(z) = \sum_{k=0}^{K-1} \frac{w_k}{z - z_k}.$$
  2. Compute a linearized rational interpolant with maximum denominator degree n and maximum numerator degree $m = K - n - 1$ in the points $z_0, \ldots, z_{K-1}$ to the function
  $$g(z) = \left(\prod_{j=0}^{J-1} (z - \eta_j) \right) f(z).$$
  3. Calculate the zeros, counted according to multiplicity, of the denominator polynomial of the interpolant computed in step 2.

Procedure (K) is essentially a statement of the derivative-free Kravanja–Van Barel method after discretization, applied to compute n poles. In its original formulation, z_k and w_k are, respectively, the nodes and weights of some quadrature rule, but there is nothing in the statement of the result that constrains them to be chosen in this way. Procedure (K) gives a way to apply the derivative-free Kravanja–Van Barel method—and, hence, the SS-H method—with *any* rational filter of the form we are considering. In Procedure (R), we compute the roots of the denominator polynomial of a linearized rational interpolant to a modified version of f that incorporates the zeros of the filter. The result we now prove asserts that these two methods are equivalent under one additional assumption.

THEOREM 4.1. *The matrix H_n of Procedure* (K) *is nonsingular if and only if the denominator polynomial computed in Procedure* (R) *has degree exactly n. If these*

equivalent conditions hold, then Procedure (K) *and Procedure* (R) *yield identical results in exact arithmetic in step* 3*: the eigenvalues computed in Procedure* (K) *are the same as the roots computed in Procedure* (R).

The nonsingularity requirement precludes degenerate situations in which the pencil $H_n^< - \lambda H_n$ of Procedure (K) has infinite eigenvalues or the denominator of the interpolant in Procedure (R) has fewer than n roots.

Before proving this theorem, we first recall some definitions from the theory of barycentric representations of polynomial interpolants [2, 42]. Suppose that $\xi_0, \ldots, \xi_{K-1}$ are K distinct points in $\mathbb{C}$. The *node polynomial* for these points is $\ell(\xi) = (\xi - \xi_0)\cdots(\xi - \xi_{K-1})$. The *barycentric weights* $\nu_0, \ldots, \nu_{K-1}$ are defined by

$$\nu_k = \frac{1}{\ell'(\xi_k)} = \frac{1}{\prod_{j=0, j\neq k}^{K-1}(\xi_k - \xi_j)}. \tag{4.5}$$

Any polynomial p of degree at most $K-1$ can be expressed in terms of its values at the points ξ_k using the *barycentric formula (of the first kind)*:

$$p(\xi) = \ell(\xi)\sum_{k=0}^{K-1}\frac{\nu_k p(\xi_k)}{\xi - \xi_k}. \tag{4.6}$$

Proof of Theorem 4.1. Let $\ell(z) = (z - z_0)\cdots(z - z_{K-1})$ be the node polynomial for the points $z_0, \ldots, z_K$ and $\nu_0, \ldots, \nu_{K-1}$ be the corresponding barycentric weights. Consider the Hankel matrices

$$\widehat{H}_n = \begin{bmatrix} h_0 & h_1 & \cdots & h_{n-1} \\ h_1 & h_2 & \cdots & h_n \\ \vdots & \vdots & & \vdots \\ h_{n-1} & h_n & \cdots & h_{2n-2} \end{bmatrix}, \qquad \widehat{H}_n^< = \begin{bmatrix} h_1 & h_2 & \cdots & h_n \\ h_2 & h_3 & \cdots & h_{n-1} \\ \vdots & \vdots & & \vdots \\ h_n & h_{n-1} & \cdots & h_{2n-1} \end{bmatrix},$$

where

$$h_j = \sum_{k=0}^{K-1}\nu_k z_k^j g(z_k), \qquad j = 0, \ldots, n-1.$$

One can show [8, 10], [19, Theorems 1.2.2 and 2.3.4] that the denominator polynomial computed in Procedure (R) has degree exactly n if and only if $\widehat{H}_n$ is nonsingular and that if this is so, the roots of this polynomial are given by solving the generalized eigenvalue problem for the pencil $\widehat{H}_n^< - \lambda\widehat{H}_n$.

Since H has poles at exactly the points z_k and J zeros, we have $H(z) = p(z)/\ell(z)$ for some polynomial p of degree J. Using (4.6) to represent p in terms of its values at the points z_k and dividing through by $\ell(z)$, we find that

$$H(z) = \sum_{k=0}^{K-1}\frac{\nu_k p(z_k)}{z - z_k}.$$

It follows from the definition of H and the uniqueness of partial fraction representations that $w_k = \nu_k p(z_k)$.

As the zeros of p are exactly the zeros of H, we can factor p to obtain $p(z) = \alpha(z - \eta_0)\cdots(z - \eta_{J-1})$ for some nonzero constant α. As $g(z) = (p(z)/\alpha)f(z)$, it follows that $h_j = \mu_j/\alpha$ for each j. Thus, $\widehat{H}_n = (1/\alpha)H_n$, so $\widehat{H}_n$ is nonsingular if and

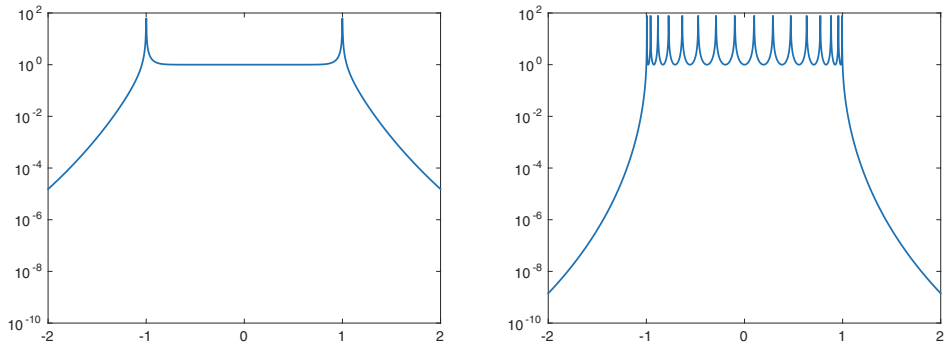

FIG. 2. *Magnitudes on* $[-2,2]$ *of filters derived from rational interpolation in the* K*th roots of unity (left) and* K *first-kind Chebyshev points on* $[-1,1]$ *(right) for* $K = 16$.

only if H_n is, establishing the equivalence of the conditions stated at the end of the theorem. Moreover, $\widehat{H}_n^< - \lambda \widehat{H}_n = (1/\alpha)(H_n^< - \lambda H_n)$, and so Procedure (R) reduces to the same generalized eigenvalue problem as Procedure (K), establishing the claim that the two yield identical results in exact arithmetic. ∎

The preceding discussion casts Theorem 4.1 as a way to associate a rational interpolation problem with the use of a given rational filter, but we can also use it "in reverse" to determine the rational filter that underlies a given rational interpolation problem. In particular, we see that finding eigenvalues of A by computing the poles of a rational interpolant to $f(z) = u^*(A - zI)^{-1}v$, unmodified, in distinct points, is equivalent to applying the SS-H method with a rational filter that has poles at those same points *and that has no zeros*.

Thus, if the interpolation nodes are $z_0, \ldots, z_{K-1}$, it follows that the rational filter implicitly applied by rational interpolation is $H(z) = \alpha/\ell(z)$, where $\ell(z) = (z - z_0)\cdots(z - z_{K-1})$ is the node polynomial for the interpolation points and α is a nonzero constant. Expressing r in pole-residue form, we have

$$H(z) = \alpha \sum_{k=0}^{K-1} \frac{\nu_k}{z - z_k}, \tag{4.7}$$

where $\nu_0, \ldots, \nu_{K-1}$ are the barycentric weights corresponding to the interpolation nodes z_k. This can be seen, e.g., by taking p to be the constant polynomial $p(z) = \alpha$ in (4.6). Rational filters of this form have the property that they achieve the maximum possible asymptotic decay rate as $|z| \to \infty$ among all rational filters (4.4) with poles at the same points, an immediate consequence of the fact that the numerator polynomial is a constant. When computing eigenvalues, this can be a desirable property, as it ensures that the corresponding approximate spectral projector strongly attenuates components of unwanted eigenvectors that are far from the region of interest.

As an example, if the points z_k are the Kth roots of unity, then the corresponding node polynomial is $\ell(z) = z^K - 1$, and the resulting filter is $H(z) = 1/(z^K - 1)$, up to an arbitrary scaling factor. For the K first-kind Chebyshev points in $[-1,1]$, the node polynomial is $\ell(z) = T_K(z)/2^K$, where T_K is the Kth degree Chebyshev polynomial of the first kind, defined in section 2.3. Rescaling to eliminate the 2^K factor, the filter is $H(z) = 1/T_K(z)$.

Graphs of the absolute values of these filters on $[-2,2]$ for $K = 16$ are shown in Figure 2. Note that while both of these filters ultimately decay as $O(z^{-K})$ as $|z| \to \infty$,

the filter associated with the Chebyshev points decays much more rapidly immediately outside of $[-1, 1]$. This means that approximate projectors based on it will do a much better job than the filter based on roots of unity at suppressing components in the direction of eigenvectors corresponding to eigenvalues that lie outside but close to $[-1, 1]$. This advantage is another reason to consider using methods based on this filter instead of methods derived from discretized contour integrals.

4.4. Rayleigh–Ritz for the Chebyshev interpolation filter. Now that we have determined the filters that underlie methods based on rational interpolation, we can remove the instabilities observed in section 3 by replacing rational interpolation with a Rayleigh–Ritz procedure based on the same filter. We proceed exactly as in the SS-RR method described in section 4.1, but instead of discretizing (4.1) to project v onto the eigenspace of interest, we use the filter (4.7).

For the first-kind Chebyshev grid on $[-1, 1]$ of length K defined by (2.5), it can be shown by direct calculation that the barycentric weight ν_k corresponding to the point x_k is $2^K T_{K-1}(x_k)/K$. After rescaling to eliminate the 2^K factor, the filter for rational interpolation on this grid may be written in pole-residue form as follows:

$$H(z) = \frac{1}{K} \sum_{k=0}^{K-1} \frac{T_{K-1}(x_k)}{z - x_k}. \tag{4.8}$$

Thus, in the notation of section 4.1, to calculate the vectors v_j that form the basis for the subspace we use when applying the Rayleigh–Ritz procedure, we compute

$$v_j = \frac{1}{K} \sum_{k=0}^{K-1} T_{K-1}(x_k) x_k^j (A - x_k I)^{-1} v, \qquad j = 0, \ldots, s-1. \tag{4.9}$$

Note that the computation of these vectors requires the solution of exactly the same linear systems as the algorithm based directly on rational interpolation presented previously. This reformulation therefore does not require any significant additional work compared to the original method.

To show that this allows us to get around the instabilities described in section 3, we rerun the same example from that section using the new procedure. This can be accomplished with the following MATLAB code:

```
A = diag([0:0.1:0.9, -10, 10]); I = eye(12); v = randn(12, 1);
K = 32; xk = chebpts(K, 1); wk = cos((K - 1)*acos(xk))/K;
V = zeros(12, 10); Y = zeros(12, K); e = ones(12, 1);
for k = 1:K, Y(:, k) = (A - xk(k)*I) \ v; end
for j = 0:1:9, V(:, j + 1) = sum((e*(wk.*xk.^j).').*Y, 2); end
[Q, R] = qr(V, 0); D = sort(eig(Q'*A*Q));
```

The first three lines simply set up the problem and initialize a few variables for storing results. The fourth line solves the systems at each of the Chebyshev points, storing the results in `Y`, and the fifth line implements (4.9) to compute the basis, storing the results in `V`. In the last line, we form the projected eigenvalue problem as described in section 4.1.

The results of running this code with the same random vector v used in the demonstration of section 3 are shown in Table 2. All 10 eigenvalues have been computed to full precision.

TABLE 2

Results of applying the reformulated method based on a Rayleigh–Ritz procedure to the test problem of section 3. All eigenvalues are computed to full accuracy. No instabilities are observed.

Computed eigenvalue	Error
0.000000000000000	1.76×10^{-16}
0.100000000000000	$0.00 \times 10^{+00}$
0.200000000000000	1.67×10^{-16}
0.300000000000000	2.78×10^{-16}
0.400000000000000	5.55×10^{-17}
0.500000000000000	2.78×10^{-16}
0.600000000000000	4.44×10^{-16}
0.699999999999999	1.11×10^{-15}
0.800000000000000	1.11×10^{-16}
0.900000000000002	2.11×10^{-15}

4.5. Contour integral derivation of the Chebyshev filter. While we arrived at the filter for rational interpolation in Chebyshev points via the equivalence established in Theorem 4.1, it is worth observing that it can also be obtained as a limit of filters derived from discretized contour integrals taken over certain ellipses that enclose the interval $[-1, 1]$. Before proceeding, we pause to outline an argument that shows this is the case.

Let D_r be the open disc in $\mathbb{C}$ with center 0 and radius $r > 1$. The ellipses we consider are *Bernstein ellipses* [42] that have the points ± 1 as their foci. If $r > 1$, then the Bernstein ellipse E_r is the image $J(D_r)$ of D_r under the Joukowski map $J(z) = (z + z^{-1})/2$. As $r \to 1$ from above, these ellipses collapse to the interval $[-1, 1]$.

If $\psi_0, \ldots, \psi_{K-1}$ are any K points that are equally spaced on the unit circle, then by transforming the integral (4.2), taken over ∂E_r, into one over ∂D_r via a change of variables using J and discretizing the result using the trapezoidal rule in these points, we obtain

$$\frac{1}{2\pi i} \int_{\partial E_r} (A - zI)^{-1}\, dz \approx \frac{1}{K} \sum_{k=0}^{K-1} (r\psi_k)\big(A - J(r\psi_k)I\big)^{-1} J'(r\psi_k).$$

Using the fact that $J'(z) = (z - z^{-1})/(2z)$ and letting $r \to 1$, one can show via straightforward computation that the filter obtained is

$$H(z) = \frac{1}{K} \sum_{k=0}^{K-1} \frac{i \operatorname{Im} \psi_k}{z - \operatorname{Re} \psi_k}.$$

Let $\psi_k = x_k + iT_{K-1}(x_k)$, $0 \le k \le K - 1$. Another computation shows that $\psi_k^K = i$ for each k, so these points are the Kth roots of i and hence are equally spaced on the unit circle. Our filter then becomes

$$H(z) = \frac{i}{K} \sum_{k=0}^{K-1} \frac{T_{K-1}(x_k)}{z - x_k},$$

which is the same as (4.8), apart from a factor of i.

4.6. General rational filters and remarks on the literature. The use of a Rayleigh–Ritz procedure in conjunction with a rational filter for computing eigenvalues is not a new idea. For example, the shift-and-invert Arnoldi method [35], applied

with a given shift σ, can be thought of as filtering the spectrum with powers of the function $1/(z-\sigma)$. This observation was extended by Ruhe to filters generated by arbitrary rational functions with his introduction of rational Krylov methods [34] in the 1980s. Contour integral methods and the method we discuss here differ from traditional rational Krylov methods primarily in the mechanics of how they build the subspace. Unlike the rational Arnoldi algorithm, which applies each shift to the starting vector in succession, these methods apply the shifts simultaneously and then take linear combinations to form the result. Doing things the latter way makes the method easier to parallelize, but all the shifts must be chosen in advance. In contrast, a method that employs the former approach can choose the shifts adaptively [7, 44].

We arrived at the concept of a rational filter by considering how discretized contour integrals and rational interpolation act on the resolvent, but it is not necessary to proceed in this way. It is equally possible to begin directly with (4.4) and ask how to choose the poles z_k and residues w_k to construct an effective filter. Eigenvalue algorithms based on rational filters have been explored extensively from this viewpoint in the Japanese literature by Murakami [21, 22, 23, 24, 25, 26], who refers to them as filter diagonalization methods. The term "filter diagonalization" comes from a class of closely related algorithms introduced by Neuhauser in the 1990s for calculating eigenstates of quantum mechanical systems that lie in a given energy interval [27, 28, 41, 46].

In [22], Murakami considers the problem of computing all the eigenvalues within a given real interval of a matrix pencil $A-\lambda B$, where A and B are both real symmetric and B is positive-definite. After introducing the concept of filter diagonalization methods, he discusses the result mentioned in section 4.3 that one can obtain a filter $H(z)$ that decays as rapidly as possible as $|z| \to \infty$ by taking the residues to be the barycentric weights of the corresponding poles (though he does not appear to use exactly this language). Murakami actually proposes the use of the reciprocal Chebyshev filter that we have been considering, justifying it using the minimality properties of Chebyshev polynomials [42]. Ultimately, however, he sets it aside in favor of filters that have no poles on the real axis (and hence require complex arithmetic to implement), owing to the potential for numerical instabilities that can arise when one of the eigenvalues sought lies close to one of the filter poles. Murakami proposes a fix for these instabilities in [23], which we will discuss in section 5.1.

In [24, 25, 26], Murakami goes on to consider the use of rational filters based on the four "classic" filter types used in analog circuit design [30]: Butterworth, Chebyshev, inverse Chebyshev,[2]and elliptic. Each of these filter types satisfies a different optimality condition with respect to certain criteria, and hence each may be expected to perform particularly well in certain circumstances. All of them have poles located off the real axis. Of these filters, the elliptic (also called Cauer or Zolotarev) filter is especially noteworthy because of its ability to attain a sharper transition across the boundary of the search interval than the other types. The price one pays for using this filter is that it does not decay to zero at infinity. This makes it well-suited to problems for which there are unwanted eigenvalues that lie outside but close to the interval of interest. On the other hand, if the desired portion of the spectrum is fairly well-separated from the rest, one will typically achieve greater accuracy by using a filter that decays at infinity. The use of elliptic filters in conjunction with the FEAST algorithm has been considered in [45] and is explored further in the paper [14].

[2]In spite of their names, the "Chebyshev" and "inverse Chebyshev" filters are not the same as the filters based on the reciprocals of Chebyshev polynomials considered in this paper.

One can also consider filters derived from rational interpolation on an interval in points other than first-kind Chebyshev grids, i.e., filters which are reciprocals of polynomials other than Chebyshev polynomials. Recall that we based our choice of the first-kind Chebyshev points on their suitability for use as interpolation points, a property that stems from the fact that they cluster near the interval endpoints [42]. If one uses a Rayleigh–Ritz-based approach instead of one based on interpolation, one might imagine that this clustering property would be less important.

Nevertheless, we can still isolate two advantages to using the proposed filter based on the reciprocal Chebyshev polynomial. First, Chebyshev polynomials grow rapidly immediately outside of the interval $[-1, 1]$ compared to other polynomials of the same degree [33]. This means that filters based on their reciprocals will typically do a better job of suppressing unwanted eigenvalues outside but close to the interval of interest than will filters with the same number of poles that achieve the same asymptotic decay rate at infinity. We noted this advantage previously in section 4.3 when comparing the proposed filter to the one derived from rational interpolation in roots of unity (recall Figure 2).

Second, the residues for the reciprocal Chebyshev filter, i.e., the barycentric weights for the first-kind Chebyshev points, are roughly uniform in magnitude, and hence the terms in the pole-residue expansion (4.8) are weighted roughly equally. Other point distributions may give rise to barycentric weights that do not have this property; for instance, the weights for equispaced points vary by factors which grow exponentially as the number of points increases [2]. Filters with the maximum asymptotic decay rate derived from such points may thus excessively weight the contributions from linear systems solved at some of the poles relative to others, potentially reducing accuracy.

5. Some practical considerations. In the preceding sections, we have presented these methods in their simplest possible forms. In this section, we briefly discuss a few additional items which should be considered when realizing them in practice.

5.1. Dealing with eigenvalues near filter poles. The method we have proposed based on the use of the reciprocal Chebyshev polynomial filter draws its strength from its placement of the filter poles within the interval of interest. While we have shown that this can be advantageous, it also has a potential drawback. If it happens that one of the eigenvalues of A, say, λ, lies close to one of the filter poles, say, z_k, then $\|(A - z_k I)^{-1}\|$ will be large. Hence, the solutions to the linear systems at z_k will dominate those from the other poles, and the resulting filtered vectors will have large components in the direction of the eigenvectors of A corresponding to λ. If λ is sufficiently close to z_k, these components can overwhelm those in the directions of the other eigenvectors of A, degrading the accuracy of their computation and that of their corresponding eigenvalues, though λ itself will be computed highly accurately.

As an illustration, we consider the same problem from section 3 but with the eigenvalue of A at 0 shifted to lie at the point $\cos(31\pi/64) + 10^{-14}$. Furthermore, instead of taking A to be diagonal, we set $A = Q^* D Q$, where D is a diagonal matrix of the specified eigenvalues and Q is a randomly generated 12×12 orthogonal matrix. Since $\cos(31\pi/64) \approx 0.049$ belongs to the 32-point first-kind Chebyshev grid on $[-1, 1]$, A has an eigenvalue very close to one of the filter poles. The results of running the same code from section 4.4 with this new A are presented in the left half of Table 3. The eigenvalue near $\cos(31\pi/64)$ has been computed to full accuracy, while the other eigenvalues, especially those near the middle of the spectrum, have suffered badly.

TABLE 3

Eigenvalues and absolute errors for the example of section 5.1 *illustrating the handling of eigenvalues that fall extremely close to filter poles.*

Original filter		Dropped pole	
Eigenvalue	Error	Eigenvalue	Error
0.049067674327428	7.6×10^{-17}	0.049067674327428	1.7×10^{-16}
0.100005648969141	5.6×10^{-06}	0.100000000000000	2.4×10^{-16}
0.200031541585367	3.2×10^{-05}	0.200000000000000	8.3×10^{-17}
0.300016487697101	1.6×10^{-05}	0.300000000000000	3.9×10^{-16}
0.411431339360006	1.1×10^{-02}	0.400000000000000	5.6×10^{-17}
0.502930792938639	2.9×10^{-03}	0.500000000000000	2.2×10^{-16}
0.600000216771745	2.2×10^{-07}	0.599999999999998	2.4×10^{-15}
0.700033492323077	3.3×10^{-05}	0.699999999999999	1.4×10^{-15}
0.800000015779351	1.6×10^{-08}	0.800000000000000	1.1×10^{-16}
0.900000000238297	2.4×10^{-10}	0.899999999999999	1.1×10^{-15}

As mentioned in section 4.6, this phenomenon was noted by Murakami [22, 23], who refers to it as a "resonance problem." In [23], he presents two options for overcoming it: either drop the offending poles from the filter or shift them to lie somewhere else. The former is simpler and needs no extra linear solves but requires that one accept the use of a filter with a slower asymptotic decay at infinity than that with which one began the computation. The latter does not have this disadvantage, since it keeps the total number of poles the same, but it is more expensive, requiring the solution of additional linear systems at the new poles. Murakami discusses these solutions in the context of non-Hermitian eigenvalue problems. Nevertheless, it is clear that they are also applicable in the Hermitian (and, in particular, the real symmetric) case, though he does not appear to mention this explicitly.

Under either option, one will need to recompute the filter residues to ensure that the resulting filter has the desired behavior. If one is working with filters of the type (4.7) and wishes to maintain the property that the recomputed filter has the maximum possible asymptotic decay rate at infinity, this amounts to calculating the barycentric weights for the new grid. Murakami provides explicit formulae for doing this in [23], which he phrases as updates to the weights for the original grid.

Before one can move to address this problem, however, one must first determine whether it has occurred. In [23], Murakami suggests looking at the factor by which application of the resolvent at a given filter pole (via the solution of the corresponding linear system) magnifies the norm of the starting vector and declaring the pole problematic if this factor exceeds a predetermined threshold. One could also consider simply looking at the computed eigenvalues and seeing if any are close to the filter poles; however, doing this may be subtle, as it is not clear how close an eigenvalue must be to a pole to be considered "too close," and the induced error may vary greatly between eigenvalues, as the results in the left half of Table 3 demonstrate.

Nevertheless, we note that the distance which qualifies as "too close" may be even smaller than one may expect at first, thanks to the Rayleigh quotient effect: an $O(\varepsilon)$ error in an approximation to an eigenvector induces an $O(\varepsilon^2)$ error in the Rayleigh-quotient estimate of the corresponding eigenvalue [43]. If an eigenvalue is a relative distance α from a pole, then since the terms in the partial fraction expansion for the filter function are inverse linear, we might anticipate induced errors in the nonresonant eigenvectors on the order of $\varepsilon_m \alpha^{-1}$, where ε_m is the machine epsilon, yielding errors in the nonresonant eigenvalues on the order of $\varepsilon_m^2 \alpha^{-2}$. Thus, the errors

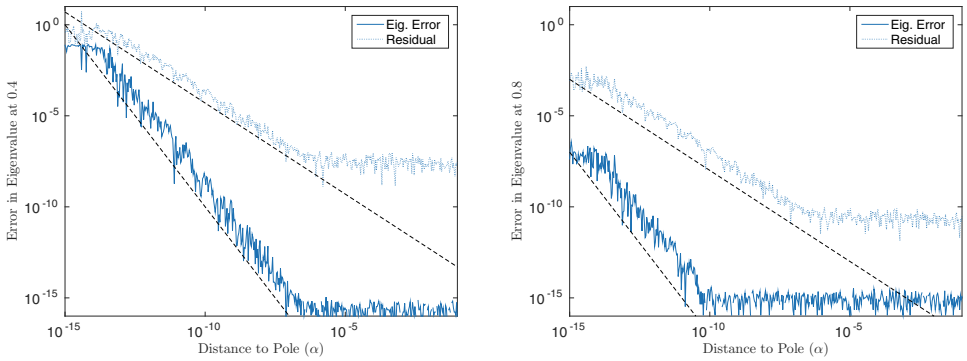

FIG. 3. *Behavior of residuals and errors in the eigenvalues at* 0.4 *(left) and* 0.8 *(right) as the eigenvalue near* $\cos(31\pi/64)$ *in the example considered in section* 5.1 *gets close to that point.*

in the eigenvalues due to the resonance effect fall off rapidly as the distance α increases. This suggests that if one seeks to compute only eigenvalues and not eigenvectors, then this problem may be less cause for concern than it may seem initially.

These facts are illustrated in the plots of Figure 3, which we produced by varying the distance α between the eigenvalue near $\cos(31\pi/64)$ and that point in the numerical example just discussed. The solid lines show the errors in the approximations to the eigenvalues at 0.4 (left) and 0.8 (right), and the dotted lines show the associated residuals, a measure of the error in the corresponding eigenvectors. The dashed lines in both plots illustrate the decay rates of $O(\alpha^{-1})$ for the residuals and $O(\alpha^{-2})$ for the eigenvalue errors. The error in the eigenvalue at 0.4 reaches the level of machine precision for α larger than around 10^{-7}, while for 0.8, this occurs for α as small as 10^{-10}. The residuals at these values are relatively large; however, if one is not concerned with approximating the eigenvectors, this is not a problem.

Returning to our original numerical example and noting that the solution to the linear system at the pole at $\cos(31\pi/64)$ has a norm larger than that of the starting vector by a factor on the order of 10^{13}, we conclude that a resonance problem has occurred and decide to correct it by simply dropping this pole from the filter. Recalculating the barycentric weights using (4.5) and applying the new filter, we obtain the results in the right half of Table 3. All eigenvalues have now been obtained to full precision.

5.2. Determining the subspace size. In our descriptions above, we have made the assumption that the number s of eigenvalues within the region of interest is known in advance and taken the dimension d of the subspace used for the Rayleigh–Ritz procedure to be equal to this number. In practice, s will have to be calculated or estimated in some way. For modest-size Hermitian eigenvalue problems, this can be accomplished using the "spectrum slicing" technique based on Sylvester's law of inertia and the LDL^* decomposition [31]. For larger problems, stochastic techniques have been developed that use contour integrals to estimate the trace of the spectral projector onto the region of interest [5, 9]. It is not immediately obvious how to extend these latter techniques to work with arbitrary rational filters because they rely on the filter taking the same (or approximately the same) value at every eigenvalue in the search region. Further investigation is needed.

Actually, all that is required is that $d \geq s$, and it is often advantageous to take d to be larger than s even when s is known exactly. This is especially helpful if the

spectrum of A is not well-separated so that there are eigenvalues close to but outside the search region that may not be adequately suppressed by the filter. Increasing d has the effect of incorporating eigenvectors corresponding to such eigenvalues within the search subspace so that they are computed instead of ignored. This results in a larger projected eigenvalue problem and the need to solve additional linear systems if block methods are used (see the next subsection). Nevertheless, the cost is typically far less than would be required to solve the problem by adding more filter poles to achieve the desired level of suppression. In [32], Polizzi suggests choosing $d \geq 1.5s$ as a rule of thumb.

While taking d to be too small yields poor results due to the influence of unwanted eigenvectors, if d is too large, one will typically find that some of the computed eigenpairs are spurious and need to be discarded. One way to deal with these is to check the residuals of the computed eigenpairs and eliminate those which are large. This is essentially what is done in the version 2.1 release of FEAST [40].

An alternative technique, proposed by Sakurai and coworkers [15, 16], uses the SVD to pare down the search subspace prior to solving the projected eigenvalue problem. The basis vectors for the subspace computed by applying the rational filter are gathered as columns into a matrix. One then computes the reduced SVD of this matrix and replaces the original basis vectors with the left singular vectors, omitting those corresponding to negligible singular values. This is very similar both in spirit and in execution to the techniques employed by `ratinterp` for eliminating spurious pole-zero pairs from rational interpolants briefly mentioned in section 2.3. In [36], Sakurai and coworkers propose further that this technique can be used to help detect when one's initial choice of the subspace dimension is too small: if none of the singular values are negligible, a larger basis is probably needed.

5.3. Block methods. If implemented exactly as described above, these methods cannot detect if an eigenvalue is derogatory, i.e., if it has multiplicity greater than one. For the SS-H and rational interpolation methods, this follows from the fact that the scalarized resolvent (2.1) has only a simple pole at each of the eigenvalues of A, even if some of those eigenvalues have non-unit multiplicity. From the perspective of SS-RR, this occurs because the subspace is generated from the projected powers of A applied to a single initial vector v.

This problem can be addressed by using multiple starting vectors to build the subspace. Sakurai and coworkers introduced this technique for their algorithms in [15] and [16] under the name of the "block Sakurai–Sugiura method," while Polizzi used it from the outset in FEAST [32]. Doing this requires one to solve additional linear systems at each filter pole, but since systems corresponding to different starting vectors are independent, they can be solved in parallel.

Aside from being able to detect higher-multiplicity eigenvalues, an additional benefit to using multiple starting vectors is that it allows one to use fewer projected powers of A when building the subspace [16]. This is useful because higher powers weaken the filter. For instance, when using (4.9) to apply the reciprocal Chebyshev polynomial filter we have been considering, the vector v_j is computed by filtering v with

$$H_j(z) = \frac{1}{K} \sum_{k=0}^{K-1} \frac{x_k^j T_{K-1}(x_k)}{z - x_k} = \frac{z^j}{T_K(z)}, \qquad 0 \leq j \leq K-1.$$

The higher the power j, the more slowly the filter decays as $|z| \to \infty$.

Allowing higher powers does, however, have the advantage of requiring fewer linear solves, so a balance must be struck. To date, there is no consensus about how this is best accomplished. In [36], Sakurai and coworkers propose the heuristic that the number of starting vectors be chosen so that the number of powers (including the zero power) employed to build the subspace is at most $K/4$. Polizzi, on the other hand, does not use higher powers at all in FEAST [32].

5.4. Outer iteration. Finally, if the accuracy of the computed eigenpairs is not satisfactory, it can be improved by using a simple iterative procedure: just take the computed eigenvectors (or some linear combination(s) thereof) as starting vectors and repeat the process, filtering them to generate a new subspace and applying the Rayleigh–Ritz procedure again to get a new set of eigenpairs. This idea was proposed by Polizzi in [32] for FEAST, and it amounts to applying powers of the underlying filter or, equivalently, subspace iteration [40]. The disadvantage to doing this is that an additional set of linear solves is required for each iteration. Since the same filter is used each time, however, one can mitigate this cost by computing the LU factors of the resolvent at each filter pole during the first pass and then reusing them on subsequent passes.

6. Summary of proposed algorithm. Taking into account some of the considerations discussed in the previous section, a more practical version of the algorithm we have been discussing based on the reciprocal Chebyshev polynomial filter might proceed as follows:

1. Fix the matrix A and the search interval $[a, b]$. Choose the number K of filter poles and the maximum number M of powers of A that will be used to build the search subspace. Let $x_0, \ldots, x_{K-1}$ be the filter poles given by (2.5), rescaled to lie in $[a, b]$, and let $\nu_0, \ldots, \nu_{K-1}$ be the corresponding barycentric weights.
2. Compute or estimate the number s of eigenvalues of A in $[a, b]$, e.g., using Sylvester's law of inertia.
3. Decide the minimum dimension $d_{\min} \geq s$ of the search subspace and calculate the number L of starting vectors as $L = \lceil d_{\min}/M \rceil$. The search subspace dimension is then $d = ML$.
4. Generate L random starting vectors, and gather them as columns into an $N \times L$ matrix V.
5. Factor $(A - x_k I) = L_k U_k$ at each filter pole x_k.
6. For each k, solve $L_k U_k W_k = V$ for W_k.
7. For each power $0 \leq j \leq M - 1$, calculate $R_j = \sum_{k=0}^{K-1} x_k^j \nu_k W_k$. Let $R = \begin{bmatrix} R_0 & \cdots & R_{M-1} \end{bmatrix}$.
8. Factor $R = XSY^*$ in an SVD. Let Q be the first r columns of X, where r is the number of singular values of R greater than a chosen tolerance.
9. Compute the eigenvalues λ_j and corresponding eigenvectors y_j of Q^*AQ. Discard those that do not lie within the search interval. The remaining λ_j are approximations to the desired eigenvalues of A, and the $v_j = Qy_j$ are the corresponding approximate eigenvectors.
10. If an eigenvalue λ_j is too close to a filter pole, adjust the filter using one of the strategies outlined in section 5.1 and return to step 7. Otherwise, proceed.
11. If greater accuracy (as measured, e.g., by the size of the residuals of the approximate eigenpairs) is desired, reassign the L columns of V to be suitably chosen linear combinations of the vectors v_j, and return to step 6.

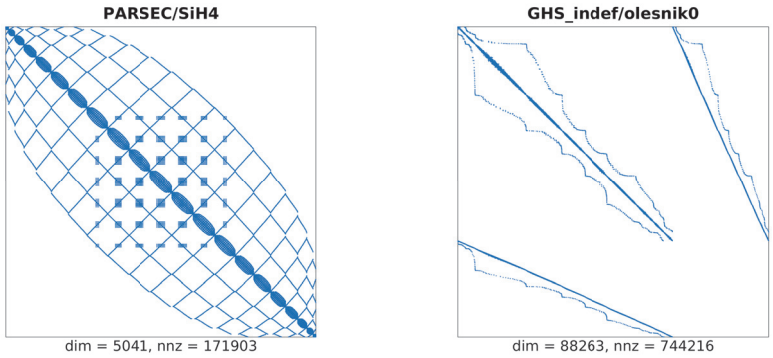

FIG. 4. *Sparsity patterns for the matrices of the test problems considered in section* 7.

7. Numerical examples. We close with a pair of examples illustrating the application of the proposed algorithm to larger problems. Our first test matrix A is the PARSEC/SiH4 matrix from the University of Florida Sparse Matrix Collection [4]. A is real symmetric with dimensions 5041×5041, and it has 171,903 nonzero entries. The sparsity pattern is shown in the left half of Figure 4.

We apply our method using the reciprocal Chebyshev polynomial filter with poles at 16 Chebyshev points of the first kind in $[-1, 1]$. Using the approach mentioned in section 5.2 based on Sylvester's law of inertia, we find that A has 35 eigenvalues in $[-1, 1]$. We take the dimension d of our search subspace to be 72, roughly twice this value. We use a block Sakurai–Sugiura-like approach (see section 5.3) using up to $16/4 = 4$ powers of A, following the rule of thumb from [36], and $72/4 = 18$ starting vectors. We do not employ any form of outer iteration (see section 5.4).

Our computations were carried out in MATLAB R2013a on 1 core of a machine with twin 8-core Intel Xeon processors, clocked at 2.7 GHz, and 256 GB of RAM. The results are shown in Tables 4 and 5.

Table 4 displays a selection of the eigenvalues in $[-1, 1]$ (specifically, the lowest 10) computed by each of three methods. The results for the remaining eigenvalues are similar. The leftmost column shows those computed by the method just described. For comparison, the middle column shows approximations to the same eigenvalues computed using the same procedure but using a contour integral method based on applying the midpoint rule in 32 points (i.e., the trapezoidal rule in 32 roots of unity, shifted along the unit circle by an angle of $\pi/32$), exploiting the symmetry to require only 16 linear solves. This is the same as the number of solves required by the reciprocal Chebyshev filter, but they require complex arithmetic. Finally, this problem is small enough that dense methods can solve it in a reasonable amount of time, and the results of using MATLAB's built-in `eig` function are shown in the rightmost column. Digits in each eigenvalue that were computed the same for all three methods are underlined. All methods agree to at least 10 digits on all eigenvalues.

Table 5 shows the amount of time taken by each method to produce the figures in Table 4. For the projection methods based on rational filters, this includes the time required to count the eigenvalues using Sylvester's law. We observe that the method which uses the filter based on the reciprocal Chebyshev polynomial is faster than that that uses the filter derived from the midpoint rule by approximately a factor of 2, reflecting the difference between real and complex arithmetic. Direct solution of the

TABLE 4
Selected eigenvalues for the PARSEC/SiH4 test problem of section 7. Figures in each row that are the same in all three columns are underlined.

Cheb. polynomial	Midpoint rule	Dense solver
−0.995566528834318	−0.995566528834321	−0.995566528834260
−0.625158654642699	−0.625158654642699	−0.625158654642678
−0.625158654640977	−0.625158654640979	−0.625158654640871
−0.625158654638943	−0.625158654638941	−0.625158654638861
0.034524054616914	0.034524054616922	0.034524054616893
0.034524054618691	0.034524054618692	0.034524054618906
0.034524054620245	0.034524054620246	0.034524054620210
0.045410512335108	0.045410512335109	0.045410512335284
0.176963580133752	0.176963580133772	0.176963580133777
0.176963580137865	0.176963580137871	0.176963580138021

TABLE 5
Computation times on a single processor for the PARSEC/SiH4 test problem of section 7. Use of multiple processors would allow a speedup of the first two figures by a factor of up to 16.

Cheb. polynomial	Midpoint rule	Dense solver
38 s	63 s	420 s

problem with `eig` is the slowest of the three, as expected for a problem of this size. In particular, the rational filter methods are much faster in spite of the fact that we employed only 1 core for the computations. Since the parameters for each have been chosen to require 16 independent linear solves, they can in theory be sped up by a factor of up to 16.

For our second example, we take A to be the GHS_indef/olesnik0 test matrix, also from the University of Florida Sparse Matrix Collection. This real symmetric matrix has dimensions 88,263 × 88,263 and has 744,216 nonzero entries. Its sparsity pattern is plotted in the right half of Figure 4.

We search for the eigenvalues of A in $[1.005, 1.010]$, again using the reciprocal Chebyshev polynomial filter with poles at 16 Chebyshev points of the first kind. Employing Sylvester's law, we find that there are 44 eigenvalues of A in this interval. We take the search subspace dimension to be 88 and use 22 starting vectors, again limiting the number of powers of A to 4. This time, we employ one step of outer iteration to refine the eigenpairs.

Ten of the computed eigenvalues and their 2-norm relative residuals (defined for an approximate eigenvalue λ and corresponding eigenvector v as $\|Av - \lambda v\|/\|Av\|$) are displayed in the first two columns of Table 6. As with the previous example, we have also computed the same eigenvalues using an equivalent contour integral method based on the midpoint rule that requires the same number of linear solves. Finally, we performed the computation a third time using MATLAB's `eigs` function, based on ARPACK [20], to search for 44 eigenvalues near 1.0075, the midpoint of the target interval. All three methods agree to at least 10 digits in the displayed eigenvalues. The maximum residual for all eigenvalues computed using the reciprocal Chebyshev polynomial filter, including those not displayed in the table, is 2.1×10^{-8}. For the midpoint rule, it is 2.2×10^{-7}. For `eigs`, it is 2.1×10^{-12}

Timings for each of the methods applied to this problem are given in Table 7. The values all include the time required to count the eigenvalues in the search interval using

TABLE 6

Selected eigenvalues and residuals for the GHS_indef/olesnik0 test problem of section 7. Figures in the eigenvalues that are the same for all three methods are underlined.

Cheb. polynomial		Midpoint rule		eigs	
Eigenvalue	Residual	Eigenvalue	Residual	Eigenvalue	Residual
1.005045284020284	8.4×10^{-09}	1.005045284020280	2.6×10^{-10}	1.005045284020283	5.4×10^{-13}
1.005103490546754	3.4×10^{-10}	1.005103490546747	4.7×10^{-10}	1.005103490546746	7.5×10^{-13}
1.005242127920682	8.1×10^{-10}	1.005242127920672	4.1×10^{-10}	1.005242127920671	4.7×10^{-13}
1.005379305986318	4.4×10^{-09}	1.005379305986315	3.5×10^{-10}	1.005379305986314	2.0×10^{-12}
1.005441288487731	5.0×10^{-09}	1.005441288487725	2.9×10^{-10}	1.005441288487726	5.3×10^{-13}
1.005502251201476	4.1×10^{-09}	1.005502251201470	4.4×10^{-10}	1.005502251201470	2.4×10^{-13}
1.005588537196233	2.1×10^{-10}	1.005588537196231	2.6×10^{-10}	1.005588537196233	3.4×10^{-13}
1.005691187808958	4.2×10^{-09}	1.005691187808951	3.3×10^{-10}	1.005691187808950	1.8×10^{-12}
1.005875698341051	1.3×10^{-09}	1.005875698341042	4.6×10^{-10}	1.005875698341042	1.6×10^{-12}
1.006254446406822	1.8×10^{-09}	1.006254446406818	5.4×10^{-10}	1.006254446406822	1.1×10^{-12}

TABLE 7

Computation times on a single processor for the GHS_indef/olesnik0 test problem of section 7. Again, the use of multiple processors would allow a speedup of the first two figures.

Cheb. polynomial	Midpoint rule	`eigs`
42 s	61 s	11 s

Sylvester's law. As before, the method based on the reciprocal Chebyshev polynomial is faster than the one based on the midpoint rule, though the speedup is closer to a factor of 1.5 for this problem instead of 2. MATLAB's `eigs` is considerably faster than both; however, as in the previous example, since we are using only 1 core for our computations, we are not taking full advantage of the parallelism offered by the other methods. As before, these methods can be sped up by approximately a factor of 16, making their timings much more competitive.

Acknowledgments. We are grateful for many fruitful discussions on this subject with our colleagues at the University of Oxford, especially Hrothgar, Mohsin Javed, Hadrien Montanelli, Alex Townsend, and Kuan Xu, and with others, including Jared Aurentz, David Bindel, Stefan Güttel, Peter Kravanja, Karl Meerbergen, Yuji Nakatsukasa, Eric Polizzi, Tetsuya Sakurai, Marc Van Barel, and Grady Wright. We are especially grateful to Prof. Sakurai for inviting us to the very stimulating EPASA2014 meeting in Japan, where part of this work was originally presented. Finally, we thank Mark Embree, whose careful reading of a draft of this paper helped us improve it substantially, and the anonymous referees for providing several valuable comments.

REFERENCES

[1] A. P. AUSTIN, P. KRAVANJA, AND L. N. TREFETHEN, *Numerical algorithms based on analytic function values at roots of unity*, SIAM J. Numer. Anal., 52 (2014), pp. 1795–1821.

[2] J.-P. BERRUT AND L. N. TREFETHEN, *Barycentric Lagrange interpolation*, SIAM Rev., 46 (2004), pp. 501–517.

[3] J. P. BOYD, *Finding the zeros of a univariate equation: Proxy rootfinders, Chebyshev interpolation, and the companion matrix*, SIAM Rev., 55 (2013), pp. 375–396.

[4] T. A. DAVIS AND Y. HU, *The University of Florida Sparse Matrix Collection*, ACM Trans. Math. Software, 38 (2011), pp. 1:1–1:25.

[5] E. Di Napoli, E. Polizzi, and Y. Saad, *Efficient estimation of eigenvalue counts in an interval*, Numer. Linear Algebra Appl., submitted.
[6] T. A. Driscoll, N. Hale, and L. N. Trefethen, eds., *Chebfun Guide*, Pafnuty Publications, Oxford, UK, 2014.
[7] V. Druskin and V. Simoncini, *Adaptive rational Krylov subspaces for large-scale dynamical systems*, Systems Control Lett., 60 (2011), pp. 546–560.
[8] Ö. Eğecioğlu and Ç. K. Koç, *A fast algorithm for rational interpolation via orthogonal polynomials*, Math. Comp., 53 (1989), pp. 249–264.
[9] Y. Futamura, H. Tadano, and T. Sakurai, *Parallel stochastic estimation method of eigenvalue distribution*, JSIAM Lett., 2 (2010), pp. 127–130.
[10] L. Gemignani, *Rational interpolation via orthogonal polynomials*, Comput. Math. Appl., 26 (1993), pp. 27–34.
[11] S. Goedecker, *Low complexity algorithms for electronic structure calculations*, J. Comput. Phys., 118 (1995), pp. 261–268.
[12] S. Goedecker, *Linear scaling electronic structure methods*, Rev. Modern Phys., 71 (1999), pp. 1085–1123.
[13] P. Gonnet, R. Pachón, and L. N. Trefethen, *Robust rational interpolation and least-squares*, Electron. Trans. Numer. Anal., 38 (2011), pp. 146–167.
[14] S. Güttel, E. Polizzi, P. T. P. Tang, and G. Viaud, *Optimized quadrature rules and load balancing for the FEAST eigenvalue solver*, SIAM J. Sci. Comput., submitted.
[15] T. Ikegami and T. Sakurai, *Contour integral eigensolver for non-Hermitian systems: A Rayleigh-Ritz-type approach*, Taiwanese J. Math., 14 (2010), pp. 825–837.
[16] 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.
[17] T. Kato, *Perturbation Theory for Linear Operators*, Springer-Verlag, New York, 1966.
[18] P. Kravanja and M. Van Barel, *A derivative-free algorithm for computing zeros of analytic functions*, Computing, 63 (1999), pp. 69–91.
[19] P. Kravanja and M. Van Barel, *Computing the Zeros of Analytic Functions*, Springer, New York, 2000.
[20] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK User's Guide: Solution of Large Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, 1998.
[21] H. Murakami, *An Experiment of the Filter Diagonalization Method for the Banded Generalized Symmetric-Definite Eigenproblem*, IPSJ SIG Technical Report 59 (2007-HPC-110), Information Processing Society of Japan, 2007.
[22] H. Murakami, *A filter diagonalization method by the linear combination of resolvents*, IPSJ Trans. Adv. Comput. Syst., 49 (2008), pp. 66–87.
[23] H. Murakami, *The Filter Diagonalization Method for the Unsymmetric Matrix Eigenproblem*, IPSJ SIG Technical Report 43 (2008-HPC-115), Information Processing Society of Japan, 2008.
[24] H. Murakami, *Experiments of Filter Diagonalization Method for Real Symmetric Definite Generalized Eigenproblems by the Use of Elliptic Filters*, IPSJ SIG Technical Report 1 (2010-HPC-125), Information Processing Society of Japan, 2010.
[25] H. Murakami, *Filter designs for the symmetric eigenproblems to solve eigenpairs whose eigenvalues are in the specified interval*, IPSJ Trans. Adv. Comput. Syst., 3 (2010), pp. 1–21.
[26] H. Murakami, *Optimization of Bandpass Filters for Eigensolver*, IPSJ SIG Technical Report 3 (2010-HPC-124), Information Processing Society of Japan, 2010.
[27] D. Neuhauser, *Bound state eigenfunctions from wave packets: Time → energy resolution*, J. Chem. Phys., 93 (1990), pp. 2611–2616.
[28] D. Neuhauser, *Time-dependent reactive scattering in the presence of narrow resonances: Avoiding long propagation times*, J. Chem. Phys., 95 (1991), pp. 4927–4932.
[29] R. Pachón, P. Gonnet, and J. van Deun, *Fast and stable rational interpolation in roots of unity and Chebyshev points*, SIAM J. Numer. Anal., 50 (2012), pp. 1713–1734.
[30] T. W. Parks and C. S. Burrus, *Digital Filter Design*, John Wiley, New York, 1987.
[31] B. N. Parlett, *The Symmetric Eigenvalue Problem*, SIAM, Philadelphia, 1998.
[32] E. Polizzi, *Density-matrix-based algorithm for solving eigenvalue problems*, Phys. Rev. B, 79 (2009), pp. 11512:1–6.
[33] T. J. Rivlin, *The Chebyshev Polynomials*, John Wiley, New York, 1974.
[34] A. Ruhe, *Rational Krylov sequence methods for eigenvalue computation*, Linear Algebra Appl., 58 (1984), pp. 391–405.
[35] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, SIAM, Philadelphia, 2011.

[36] T. Sakurai, Y. Futamura, and H. Tadano, *Efficient parameter estimation and implementation of a contour integral-based eigensolver*, J. Algorithms Comput. Technol., 7 (2013), pp. 249–269.
[37] T. Sakurai and H. Sugiura, *A projection method for generalized eigenvalue problems using numerical integration*, J. Comput. Appl. Math., 159 (2003), pp. 119–128.
[38] 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.
[39] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.
[40] P. T. P. Tang and E. Polizzi, *FEAST as a subspace iteration eigensolver accelerated by approximate spectral projection*, SIAM J. Sci. Comput., (2014), pp. 354–390.
[41] S. Toledo and E. Rabani, *Very large electronic structure calculations using an out-of-core filter-diagonalization method*, J. Comput. Phys., 180 (2002), pp. 256–269.
[42] L. N. Trefethen, *Approximation Theory and Approximation Practice*, SIAM, Philadelphia, 2013.
[43] L. N. Trefethen and D. Bau, III, *Numerical Linear Algebra*, SIAM, Philadelphia, 1997.
[44] R. Van Beeumen, K. Meerbergen, and W. Michiels, *A rational Krylov method based on Hermite interpolation for nonlinear eigenvalue problems*, SIAM J. Sci. Comput., 35 (2013), pp. A327–A350.
[45] G. Viaud, *The FEAST Algorithm for Generalised Eigenvalue Problems*, M.Sc. thesis, University of Oxford, Oxford, UK, 2012.
[46] M. R. Wall and D. Neuhauser, *Extraction, through filter-diagonalization, of general quantum eigenvalues or classical normal mode frequencies from a small number of residues or a short-time segment of a signal.* I. *Theory and application to a quantum-dynamics model*, J. Chem. Phys., 102 (1995), pp. 8011–8022.