SIAM J. SCI. COMPUT.
Vol. 46, No. 6, pp. A3552–A3573

A RATIONAL FILTERING ALGORITHM FOR SEQUENCES OF SHIFTED SYMMETRIC LINEAR SYSTEMS WITH APPLICATIONS TO FREQUENCY RESPONSE ANALYSIS*

ANTHONY P. AUSTIN†, LIOR HORESH‡, AND VASSILIS KALANTZIS‡

Abstract. We present a numerical algorithm for the solution of a large number of shifted linear systems for which the system pencil is symmetric and definite and the shifts lie inside a given real interval. Extending an earlier method due to Meerbergen and Bai [*SIAM J. Matrix Anal. Appl.,* 31 (2010), pp. 1642–1662], the algorithm uses a rational filter with poles at Chebyshev points to compute and deflate the components of the solution in the direction of eigenvectors of the system pencil corresponding to eigenvalues within the interval. It then solves the deflated systems for the remaining components using a Krylov subspace method with a preconditioner constructed by interpolating the factorizations at the filter poles. The algorithm parallelizes naturally. We demonstrate its effectiveness using matrix pencils from both model and real-world problems and discuss applications to frequency response analysis.

Key words. shifted linear systems, rational filtering, deflation, Chebyshev interpolation, frequency response analysis

MSC codes. 65F08, 65F15

DOI. 10.1137/23M1578474

1. Introduction.

We consider the problem of solving a sequence of shifted linear systems,

$$(A-\omega_j M)x_j = f_j, \qquad j=1,\dots,m, \tag{1.1}$$

where the $n\times n$ matrices A and M are real symmetric, M is positive definite, and the shifts $\omega_1,\dots,\omega_m$ belong to a given interval $[a,b]$ in $\mathbb{R}$. This problem arises in the frequency response analysis of linear dynamical systems, in which A and M are, respectively, the stiffness and mass matrices of a finite element model of such a system, and the ω_j are the squares of the frequencies at which the response is sought.

In principle, solving (1.1) is straightforward: factor $A-\omega_j M$ in an LU decomposition for each j and then recover x_j via a pair of triangular solves. This becomes expensive when n and m are large. One standard approach to reducing the cost is to approximate x_j by projecting (1.1) onto the subspace spanned by the eigenvectors of the pencil (A,M) corresponding to the eigenvalues of (A,M) that lie inside some interval $[c,d]$ that contains $[a,b]$. This is known as *modal superposition* [7, Chapter 12], and for it to be effective, all the x_j must lie nearly in the computed eigenspace. Rarely is it clear how to select the bounds c and d to ensure this; moreover, even if a suitable choice can be found, the number of needed eigenvectors can be very large.

*Submitted to the journal's Numerical Algorithms for Scientific Computing section June 12, 2023; accepted for publication (in revised form) August 14, 2024; published electronically November 12, 2024. The views expressed in this document are those of the authors and do not reflect the official policy or position of the Department of Defense, the Department of the Navy, or the U. S. Government.

https://doi.org/10.1137/23M1578474

Funding: The work of the second and third authors was supported by the Mathematical Sciences Council of IBM Research through its Exploratory Science Initiative.

†Department of Applied Mathematics, Naval Postgraduate School, Monterey, CA 93940 USA (anthony.austin@nps.edu).

‡IBM Research, Thomas J. Watson Research Center, Yorktown, Heights, NY 10598 USA (lhoresh@us.ibm.com, vkal@ibm.com).

In [29], Meerbergen and Bai address these difficulties by combining modal superposition with a Krylov subspace method. They approximate the eigenvectors of (A, M) corresponding to eigenvalues inside $[a, b]$ using what amounts to a version of the shift-and-invert Lanczos method. They then project (1.1) onto these approximate eigenvectors and solve for the corresponding components of x_j. To pick up the remaining components, they deflate the approximate eigendirections from (1.1) and apply Lanczos to the deflated system, using $(A - \sigma M)^{-1}$ for some shift $\sigma \in [a, b]$ as a preconditioner. By using the same shift for both the preconditioner and shift-and-invert Lanczos, Meerbergen and Bai are able to solve (1.1) using only a single factorization: that of $A - \sigma M$.

The key to the success of the Meerbergen and Bai algorithm is that it limits the interval over which the eigenvalue problem for (A, M) must be solved to $[a, b]$ itself instead of to a wider interval that contains it. This is effective; however, the algorithm's single-shift approach encounters difficulties when $[a, b]$ is deep in the interior of the spectrum of (A, M) and contains many eigenvalues. It is difficult to approximate eigenvectors corresponding to a large number of interior eigenvalues with only a single shift.

To remedy this, we propose a multishift extension of the Meerbergen and Bai algorithm that uses *rational filtering* to solve the eigenvalue problem for (A, M) over $[a, b]$. Rational filtering can be viewed as a version of the shift-and-invert technique (or inverse iteration) that uses multiple shifts, but in contrast to other multishift methods [12, 26, 49], the shifts are selected up front as the poles of a rational function—the *filter*—instead of on the fly. Because the shifted systems at each pole can be factored and solved independently, the resulting algorithms are embarrassingly parallel. Accordingly, rational filtering methods have proven popular as large-scale, parallel eigensolvers and have been implemented in several high-performance eigensolver software packages [11, 15, 19, 24, 25, 35]. These methods are particularly useful for interior eigenvalue problems and have been studied extensively in the recent literature; for further details, see [2, 13, 21, 22, 34, 40, 41, 45, 47] and the references therein.

We employ the reciprocal Chebyshev polynomial filter from [2, 30], the poles of which are Chebyshev points. As we discuss below, there are several reasons this filter is a good choice, but the most important is that it enables reuse of the factorizations incurred when computing the eigenvectors to precondition the solves of the deflated systems. Viewing the poles of the filter as interpolation points, we construct a polynomial interpolant through the factorizations and use this as a preconditioner. Results from Chebyshev interpolation theory guarantee the efficacy of this approach as the number of poles increases [46]; we use this theory to bound the M-norm condition numbers of the preconditioned systems. As our experiments show, one can obtain good results even with relatively few filter poles. Moreover, through the use of a Lagrange representation for the interpolant, the preconditioner retains the embarrassing parallelism of the filter. The result is a method for solving (1.1) that, given sufficient parallel resources, has an up-front cost in time of a single factorization—just like the Meerbergen and Bai algorithm—while still performing far fewer factorizations than would be required to solve (1.1) the straightforward way.

2. Splitting via modal superposition. We begin the description of our algorithm with a discussion of the splitting of (1.1) via modal superposition, which will also serve to fix some notation that we will use throughout the article. To reduce clutter, we suppress the subscript j and describe our method for a single equation,

$$(A - \omega M)x = f, \tag{2.1}$$

with $\omega \in [a, b]$.

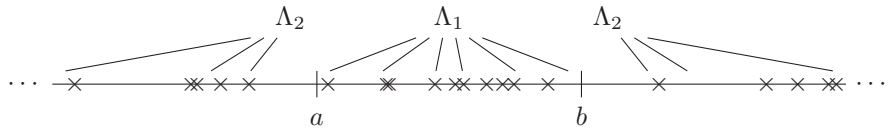

FIG. 2.1. *Schematic illustration of the splitting of the eigenvalues of* (A, M) *(crosses) according to whether they lie in* $[a, b]$*. Eigenvalues within* $[a, b]$ *appear on the diagonal of* Λ_1*; those outside* $[a, b]$*, on the diagonal of* Λ_2*.*

Solving the generalized eigenvalue problem for (A, M) leads to a decomposition,

$$AV = MV\Lambda, \tag{2.2}$$

where the columns of V are the eigenvectors of (A, M) and Λ is a diagonal matrix with the corresponding eigenvalues. Since A and M are real symmetric with M positive definite, the eigenvalues are real, and the eigenvectors are M-orthonormal:

$$V^T MV = I, \tag{2.3}$$

where I denotes the identity matrix. We order the eigenvalues so that those lying in $[a, b]$ appear first on the diagonal of Λ and partition V and Λ as

$$V = \begin{bmatrix} V_1 & V_2 \end{bmatrix}, \qquad \Lambda = \begin{bmatrix} \Lambda_1 & \\ & \Lambda_2 \end{bmatrix},$$

where the pairs (V_1, Λ_1) and (V_2, Λ_2) correspond to eigenvalues inside and outside $[a, b]$, respectively. Figure 2.1 provides an illustration of the partitioning of Λ.

Since the eigenvectors form a basis for $\mathbb{R}^n$, we can split the solution x to (2.1) into its components along the directions in V_1 and V_2 as

$$x = V_1 x_1 + V_2 x_2. \tag{2.4}$$

The relations (2.2) and (2.3) decouple (2.1) into separate equations for x_1 and x_2:

$$V_1^T (A - \omega M) V_1 x_1 = V_1^T f, \qquad V_2^T (A - \omega M) V_2 x_2 = V_2^T f. \tag{2.5}$$

Our aim is to solve these equations and then reconstitute x via (2.4).

3. Solving for x_1: Rational filtering. Rational filtering is a technique for computing the eigenvalues and corresponding eigenvectors of a (regular) matrix pencil that lie within a given region of the complex plane. In our case, the pencil is (A, M), and the region is the interval $[a, b]$ in $\mathbb{R}$. The key idea is to use a rational transformation of (A, M) to access the spectral projector associated with the eigenvalues in $[a, b]$. This is then combined with a standard eigenvalue iteration to set up a Rayleigh–Ritz projection for the target eigenpairs.

In more detail, we first select a rational function H, called the *filter,* of the form

$$H(z) = \sum_{k=0}^{K-1} \frac{w_k}{z - \zeta_k}, \tag{3.1}$$

which is large in magnitude on $[a, b]$ and small elsewhere. Then, when the matrix

$$H(M^{-1}A) = \sum_{k=0}^{K-1} w_k (A - \zeta_k M)^{-1} M \tag{3.2}$$

is applied to a vector, it will amplify (respectively, suppress) the components of that vector in the directions of the eigenvectors in V_1 (respectively, V_2), since

$$H(M^{-1}A) = VH(\Lambda)V^{-1} = \begin{bmatrix} V_1 & V_2 \end{bmatrix} \begin{bmatrix} H(\Lambda_1) & \\ & H(\Lambda_2) \end{bmatrix} \begin{bmatrix} V_1 & V_2 \end{bmatrix}^{-1}, \tag{3.3}$$

and $H(\Lambda_1)$ is much larger in magnitude than $H(\Lambda_2)$ by construction.

We can use H to build a subspace iteration, Arnoldi iteration, or other similar procedure for computing the eigenvalues in Λ_1 together with their corresponding eigenvectors. Algorithm 3.1 lists one option based on the randomized algorithm described in [21]. The idea is that if H is well chosen, then $H(M^{-1}A)$ is approximately low rank, since (3.3) and the relation $V^{-1} = V^T M$ imply

$$H(M^{-1}A) = V_1 H(\Lambda_1) V_1^T M + V_2 H(\Lambda_2) V_2^T M \approx V_1 H(\Lambda_1) V_1^T M$$

when $H(\Lambda_1)$ is much larger than $H(\Lambda_2)$. Hence, randomized methods [14, 27] should be effective for finding a basis for $\mathrm{Ran}\big(H(M^{-1}A)\big) \approx \mathrm{Ran}(V_1)$, the target eigenspace. With this basis in hand, the target eigenpairs can be computed using a standard Rayleigh–Ritz projection.

We emphasize that Algorithm 3.1 describes just one possibility of many. But even if we confine ourselves to its framework, there are several parameters—the initial subspace dimension d, the expansion dimension s, and the convergence tolerance ε—that must be selected appropriately for the method to be effective. While these details are important, treating them here would take us far afield from our objective of describing a method for (1.1), and so we shall say little about them, referring

Algorithm 3.1: Computation of V_1, Λ_1 via rational filtering.

Input : Pencil (A, M), search interval $[a, b]$
Output: Matrices V_1, Λ_1 of eigenvectors, eigenvalues of (A, M) in $[a, b]$

`/* Preliminary factorization of systems at filter poles. */`

Factor $A - \zeta_k M = L_k D_k L_k^T$ for $k = 0, \ldots, K-1$.

`/* Randomized algorithm for finding the range of` $H(M^{-1}A)$. `*/`

Pick initial dimension d, threshold ε. Generate random Gaussian $X \in \mathbb{R}^{n \times d}$.

while *not converged* **do**

  Compute $Y = H(M^{-1}A)X = \sum_{k=0}^{K-1} w_k L_k^{-T} D_k^{-1} L_k^{-1} X$.

  Write $Y = U\Sigma V^T$ in a reduced SVD with singular values $\sigma_1 \geq \cdots \geq \sigma_n$.

  if $\sigma_n/\sigma_1 < \varepsilon$ **then**

    Let $r = \min\{j : \sigma_j/\sigma_1 < \varepsilon\}$.

    Set $Y = U_{:,1:r}$ (first r columns of U).

    break

  Select expansion dimension s and generate random Gaussian $\tilde{X} \in \mathbb{R}^{n \times s}$.

  Set $X = \begin{bmatrix} U & \tilde{X} \end{bmatrix}$.

`/* Rayleigh--Ritz projection. */`

Solve the $r \times r$ eigenvalue problem $(Y^T A Y)w = \theta(Y^T M Y)w$. Let W be the matrix of eigenvectors and Θ the corresponding matrix of eigenvalues.

Discard eigenpairs (w, θ) for which θ lies outside $[a, b]$ or for which the residual $\|AYw - \theta MYw\|_2$ is large, leaving reduced matrices W_1 and Θ_1.

Set $V_1 = YW_1$ and $\Lambda_1 = \Theta_1$.

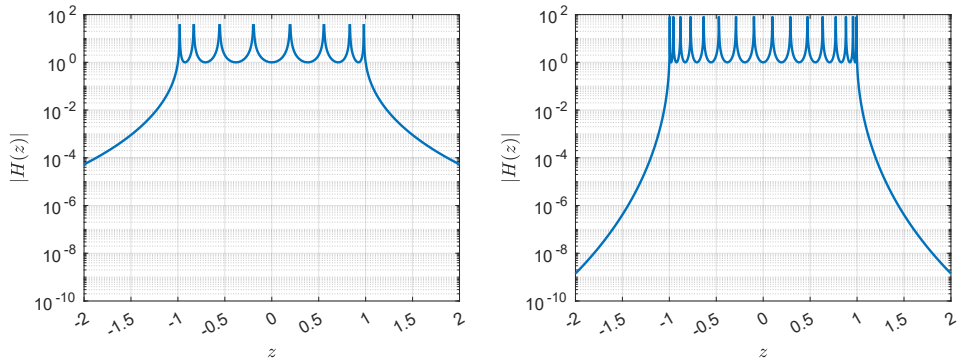

FIG. 3.1. *Magnitude for $z \in [-2,2]$ of the filter* (3.1) *with ζ_k and w_k defined by* (3.4) *and* (3.5). *Left: $K = 8$. Right: $K = 16$. The sharp spikes indicate the locations of the poles.*

the reader to [21] and the wider rational filtering literature for more information. Instead, we focus on the selection of the filter, as our choice for H will play an equally prominent role in the computation of x_2 in section 4 as it does here in the computation of x_1.

There are many methods for selecting H, including discretization of contour integrals [34, 40] and rational approximation of the indicator function for $[a,b]$ [13, 47]. Here, we use the reciprocal Chebyshev polynomial filter described in [2, 30]. This filter takes the poles ζ_k to be the K *Chebyshev points of the first kind* in $[a,b]$,

$$\zeta_k = \frac{a+b}{2} + \frac{2}{b-a}\cos\left(\frac{(2k+1)\pi}{2K}\right), \qquad k = 0,\ldots,K-1, \tag{3.4}$$

with the corresponding weights w_k given by

$$w_k = \frac{1}{K}\cos\left((K-1)\frac{(2k+1)\pi}{2K}\right), \qquad k = 0,\ldots,K-1. \tag{3.5}$$

Plots of $|H(z)|$ with this choice of the ζ_k and w_k for $K = 8, 16$ and $[a,b] = [-1,1]$ are shown in Figure 3.1. Observe that $|H(z)|$ decays rapidly as z gets further away from $[-1,1]$, signaling that H is a good filter for this interval. For the remainder of this article, when we refer to the filter H, we assume that the poles and weights are given by (3.4) and (3.5), respectively. The principal advantage of this filter over other commonly used filters, especially those derived from contour integration, is that because (A,M), the ζ_k, and the w_k are all real, the resulting algorithm uses only real arithmetic, roughly halving the computational and storage costs.

Each step of the main loop in Algorithm 3.1 requires the application of (3.2) to one or more vectors, which in turn requires the solution of K linear systems, generally with multiple right-hand sides. To solve these systems, we compute an LDL^T factorization of each of the matrices $A - \zeta_k M$. These factorizations—as well as the corresponding linear solves—can be carried out in parallel. The factorizations need only be performed once at the start of the algorithm; once constructed, they can be reused as many times as needed.

With V_1 and Λ_1 in hand, we can solve the first equation in (2.5) for x_1. We compute $V_1^T(A - \omega M)V_1 = \Lambda_1 - \omega I$, a diagonal matrix, and obtain

$$x_1 = (\Lambda_1 - \omega I)^{-1} V_1^T f.$$

This completes the first phase of our algorithm.

4. Solving for x_2: Deflated Krylov iteration. To compute the remaining components of x, we must solve the second equation in (2.5) for x_2. Because we do not compute V_2 or Λ_2, we cannot do this by solving a diagonal system as we did for x_1. Instead, we use the fact that the eigenvectors of (A, M) resolve the identity matrix into a sum of complementary M-orthogonal projectors,

$$I = V_1 V_1^T M + V_2 V_2^T M,$$

to write

$$V_2 = (V_2 V_2^T M) V_2 = (I - V_1 V_1^T M) V_2.$$

Thus, the equation for x_2 may be expressed in the form

$$(I - M V_1 V_1^T)(A - \omega M)(I - V_1 V_1^T M) V_2 x_2 = (I - M V_1 V_1^T) f. \tag{4.1}$$

We view (4.1) as an equation not for x_2 but for $V_2 x_2$, which is what we are really after. It is singular but consistent: it is just the system (2.1) with the V_1 components deflated.

We solve (4.1) using a deflated Krylov subspace method. Much is known about the behavior of Krylov subspace iterations in the presence of deflation; see, e.g., [4, 5, 8, 20, 32, 38, 42, 43] for details. All we require is an effective preconditioner to ensure the iteration converges quickly.

Consider the ideal preconditioner

$$P_*(\omega) = (I - V_1 V_1^T M)(A - \omega M)^{-1}(I - M V_1 V_1^T).$$

We call $P_*(\omega)$ "ideal" because it exactly inverts $A - \omega M$ over the invariant subspace spanned by the eigenvectors in V_2:

$$P_*(\omega)(I - M V_1 V_1^T)(A - \omega M)(I - V_1 V_1^T M) V_2 = V_2.$$

If we could apply $P_*(\omega)$ to a vector, we could solve (4.1) directly, but doing this requires us to factor $A - \omega M$, which we want to avoid.[1] Instead, we proceed as follows. When solving for x_1, we factored $A - \zeta_k M$ for each k. Thus, we have access to samples $P_*(\zeta_k)$ of P_* at the Chebyshev points (3.4). We use these samples to build a polynomial interpolant

$$P(\omega) = \sum_{k=0}^{K-1} \ell_k(\omega)(I - V_1 V_1^T M)(A - \zeta_k M)^{-1}(I - M V_1 V_1^T)$$

to $P_*(\omega)$, where

$$\ell_k(z) = \prod_{\substack{j=0 \\ j \neq k}}^{K-1} \frac{z - \zeta_j}{\zeta_k - \zeta_j} \tag{4.2}$$

is the kth Lagrange basis polynomial for interpolation in the ζ_k.

We do not construct $P(\omega)$ explicitly; we require only the ability to apply $P(\omega)$ to a vector, and this can be done by applying each term of the sum to the vector individually and adding the results. Nor do we explicitly construct the terms, which

[1] Were we willing to factor $A - \omega M$, we would not need $P_*(\omega)$, as we could solve (2.1) directly.

can be applied using multiplications with M, V_1, and V_1^T and solves with $A - \zeta_k M$, the factors of which are already on hand. Since the terms are independent of one another, they can be applied in parallel.

If K is large enough, $P(\omega)$ will be a good approximation to $P_*(\omega)$ and hence a good preconditioner for (4.1). This is ensured by the following theorem, which asserts that the eigenvalues of the preconditioned matrix corresponding to eigenvectors in V_2 cluster in a disc centered at 1 with a radius that shrinks exponentially in K. In the statement of the theorem, $\Lambda_2(A, M)$ refers to the set of eigenvalues of (A, M) that do not belong to $[a, b]$, and

$$\operatorname{dist}(\lambda, [a, b]) = \min(|\lambda - a|, |\lambda - b|)$$

gives the distance from an eigenvalue $\lambda \in \Lambda_2(A, M)$ to $[a, b]$.

THEOREM 4.1. *The eigenvalues μ of the preconditioned matrix*

$$T(\omega) = P(\omega)(I - MV_1V_1^T)(A - \omega M)(I - V_1V_1^T M)$$

corresponding to eigenvectors in V_2 are real and satisfy

$$|\mu - 1| \leq C\rho^{-K}$$

uniformly for $\omega \in [a, b]$, where

$$\rho = (1 + 2\gamma) + \sqrt{(1 + 2\gamma)^2 - 1}, \qquad C = \frac{16\rho^3}{(\rho^2 - 1)(\rho - 1)},$$

and

$$\gamma = \min_{\lambda \in \Lambda_2(A,M)} \frac{\operatorname{dist}(\lambda, [a, b])}{b - a}.$$

The proof of Theorem 4.1 is a routine application of Chebyshev approximation theory. The projections $(I - V_1V_1^T M)$ and $(I - MV_1V_1^T)$ that appear in the definition of $P_*(\omega)$ remove the poles of $(A - \omega M)^{-1}$ at eigenvalues in $[a, b]$. Consequently, $P_*(\omega)$ is an analytic function of ω on $[a, b]$ [6, Chapter 2], [23, Chapter 2], and hence its Chebyshev interpolants on $[a, b]$ converge to it uniformly at an exponential rate [46, Chapter 8]. For further details, see Appendix A.

Theorem 4.1 shows that the efficacy of the preconditioner is governed by the number K of Chebyshev points and the relative separation of the eigenvalues in Λ_2 from $[a, b]$, given by γ. Figure 4.1 plots the convergence rate ρ and the bound $C\rho^{-K}$ as functions of γ. As K and γ increase, the bound shrinks, and the eigenvalues of $T(\omega)$ cluster more tightly around 1. Only when γ is very small—that is, when there are eigenvalues in Λ_2 that are very close to $[a, b]$—does one need a large value of K for the preconditioner to be effective. In this case, it may be better to compute these eigenvalues and their corresponding eigenvectors directly and incorporate them into Λ_1 and V_1. Such computations are common in rational filtering [17, 18, 39] and can be done during the first phase of our algorithm.

Note that the preconditioned matrix $T(\omega)$ is not symmetric unless $M = I$ and that the preconditioner $P(\omega)$, while symmetric, is not generally positive definite. Together, these facts would seem to necessitate the use of a nonsymmetric Krylov iteration, such as GMRES [37] instead of the symmetric MINRES [31] or conjugate gradient (CG)

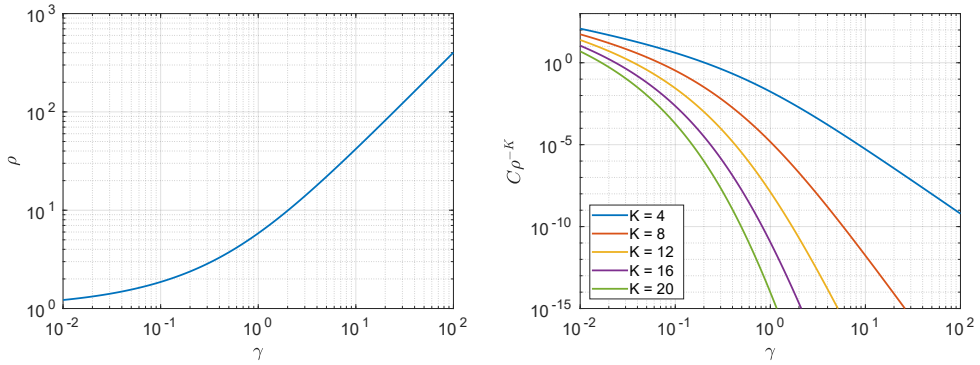

FIG. 4.1. *Illustration of the bound of Theorem* 4.1. *Left: plot of the convergence rate* ρ *as a function of* γ. *Right: plot of the eigenvalue bound* $C\rho^{-K}$ *as a function of* γ *for several values of* K.

[16] methods. As our experiments in section 6, which employ GMRES, will show, $P(\omega)$ is usually such an effective preconditioner that this is not a significant issue. Nevertheless, if one prefers to use a symmetric iteration, there are two methods for doing so. First, one can symmetrize the preconditioned system at the cost of computing a Cholesky factorization of M; details of this approach are given in Appendix B. Alternatively, one can observe that

$$T(\omega)^T M V_2 = M T(\omega) V_2.$$

Hence, $T(\omega)$ is self-adjoint in the M-inner product[2] over the subspace spanned by the eigenvectors in V_2, and so one can use a symmetric iteration, provided that one runs the iteration in this inner product.

The convergence rates of the symmetric iterations (either approach) can be estimated with the aid of the following corollary to Theorem 4.1.

COROLLARY 4.2. *For large enough* K, *the* M*-norm condition number* κ *of the preconditioned matrix* $T(\omega)$ *satisfies*

$$\kappa \leq \frac{1+\varepsilon}{1-\varepsilon},$$

where $\varepsilon = C\rho^{-K}$ *with* C *and* ρ *as in Theorem* 4.1.

Proof. Since $T(\omega)$ is self-adjoint in the M-inner product, its M-norm condition number is the ratio of the magnitude of its largest eigenvalue to that of its smallest. The result follows by taking K large enough to ensure $C\rho^{-K} < 1$ so that by Theorem 4.1, the eigenvalues μ of $T(\omega)$ obey $0 < 1-\varepsilon \leq \mu \leq 1+\varepsilon$. ∎

For instance, if K is large enough, then $T(\omega)$ is positive definite in the M-inner product. Thus, we can use CG in the M-inner product to solve the preconditioned system. The standard convergence analysis for CG [36, section 6.11.3] asserts that this iteration will converge at a rate governed by the ratio $(\sqrt{\kappa}-1)/(\sqrt{\kappa}+1)$, where κ is the M-norm condition number of $T(\omega)$. By providing a bound on κ, Corollary 4.2 provides a bound on this rate. Figure 4.2 plots these bounds as a function of γ for several values of K.

[2]The M-inner product on $\mathbb{R}^n$ is defined by $\langle x, y\rangle_M = y^T M x$.

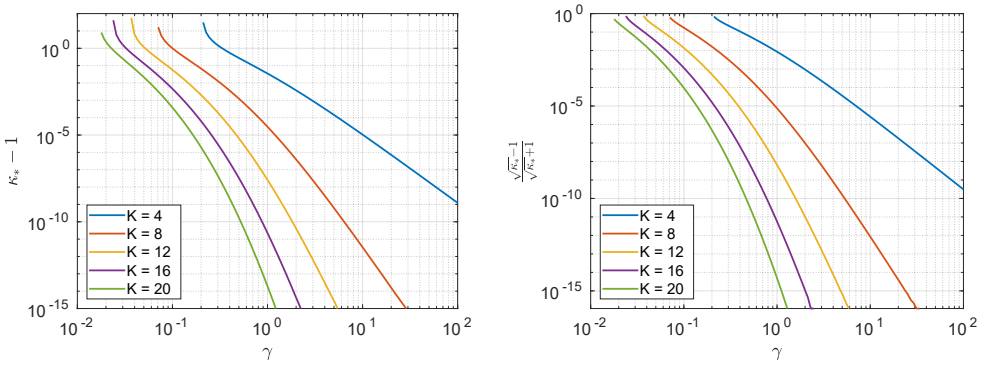

FIG. 4.2. *Illustration of the condition-number bound* $\kappa_* = (1+\varepsilon)/(1-\varepsilon)$ *of Corollary* 4.2 *(left) and the accompanying bound on the CG convergence rate (right) as functions of* γ *for several values of* K*. In the left panel, we plot* $\kappa_* - 1$ *instead of* κ_* *to show the detailed behavior of* κ_* *as it approaches* 1.

5. Summary of algorithm. The procedure we have just described for solving (1.1) is summarized in Algorithm 5.1. We emphasize that this algorithm exposes natural K-way parallelism during the most expensive operations: the factorizations in step 1 can all be carried out in parallel, as can the solves with these factorizations required at each step of the iterations in steps 2 and 5. One can also consider parallelizing the loop in step 3, treating the systems independently, or batching them, solving several simultaneously, but we do not pursue these possibilities here.

A crude complexity analysis suggests that when the number m of systems to be solved is very large, Algorithm 5.1 will outperform the naive approach of factoring and solving each system individually when

$$T_{\text{fact}} \gtrsim (s_K K - 1) T_{\text{subst}}, \tag{5.1}$$

where T_{fact} is the amount of time required to factor one of the shifted systems, T_{subst} is the amount of time required to perform triangular substitutions with the factors, and s_K is the average number of Krylov subspace iterations required to solve each system in step 5 of Algorithm 5.1. This estimate applies whether one is working in serial or K-way in parallel; details are given in Appendix C.

In general, we expect s_K to decrease with increasing K, as a larger value of K yields a more effective preconditioner; however, as it is $s_K K$ that matters, not s_K alone, there is a trade-off. This trade-off is illustrated in the graphs of Figure 5.1, which plot $s_K K$ versus γ for several values of K using the estimated CG convergence rates from the right pane of Figure 4.2 for two different CG convergence tolerances. For instance, the plot in the left pane shows that with $\gamma = 10^{-1}$ and a CG convergence tolerance of 10^{-6}, T_{fact} must be about 50 times larger than T_{subst} for Algorithm 5.1 to beat the naive approach with $K = 16$, while for $K = 20$, the factor shrinks to approximately 40. These figures are based on upper bounds and imprecise estimates; we encourage the reader not to read into them too deeply.

Finally, we note that one can reduce s_K without increasing K via continuation, using solutions found at earlier values of j as initial guesses for the Krylov methods at later values. We use this technique in our numerical experiments but do not explore it in greater depth here.

Algorithm 5.1. Solution of multiple shifted linear systems.

Input : Pencil (A, M), interval $[a, b]$, shifts $\omega_1, \ldots, \omega_m$ in $[a, b]$, right-hand sides $f_1, \ldots, f_m$

Output: Solutions x_j to $(A - \omega_j M)x_j = f_j$ for $j = 1, \ldots, m$

1 Factor $A - \zeta_k M = L_k D_k L_k^T$ for $k = 0, \ldots, K-1$.

2 Use the factorizations to compute V_1, Λ_1 via Algorithm 3.1 or other rational filtering method using the filter defined by (3.1), (3.4), and (3.5).

3 **for** $j = 1, \ldots, m$ **do**

4 Compute $V_1 x_{j,1} = V_1(\Lambda_1 - \omega_j I)^{-1} V_1^T f_j$.

5 Compute $V_2 x_{j,2}$ by solving

$$P(\omega_j)(I - MV_1V_1^T)(A - \omega_j M)(I - V_1V_1^T M)V_2 x_{j,2} = P(\omega_j)(I - MV_1V_1^T)f_j$$

using a Krylov subspace iteration. Use the factorizations computed in step 1 to apply $P(\omega_j)$ at each step of the iteration.

6 Set $x_j = V_1 x_{j,1} + V_2 x_{j,2}$.

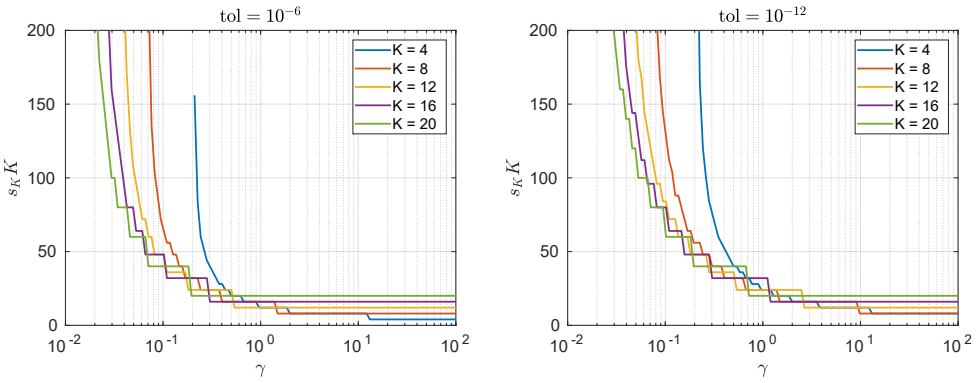

FIG. 5.1. *Plots of $s_K K$ versus γ for several values of K with s_K estimated using the CG convergence rates from the right pane of Figure 4.2 for CG convergence tolerances of 10^{-6} (left) and 10^{-12} (right). The stair-step nature of the curves is due to the fact that s_K and K are integers.*

6. Numerical results. We now provide several numerical examples that illustrate the effectiveness of the proposed method. As mentioned in the introduction, the principal application requiring the solution of (1.1) is frequency response analysis, and all test problems we consider come from this application. All computations were carried out using MATLAB R2022b on a GNU/Linux workstation with dual 24-core AMD EPYC 7402 processors clocked at 2.8 GHz and 1 TB of main memory.

For the rational filtering computations, we use a variant of Algorithm 3.1. We set the initial subspace dimension d in step 2 to 125% of the number of eigenvalues in $[a, b]$, which we compute by factoring the system pencil at a and b and using Sylvester's law of inertia [33, Theorem 3.3.1].[3] We set the expansion dimension s in step 10 to 10% of the current subspace size and the convergence tolerance ε in step 6 to 10^{-12}; the latter can be relaxed to larger values at the expense of accuracy. The criterion for the residual check in step 13 is

[3]Note that as neither a nor b is a pole for our filter, these factorizations are "wasted" in the sense that they are not reused for any purpose following the inertia count. This is easily remedied by switching to a filter with poles at the *Chebyshev points of the second kind* on $[a, b]$, which include the interval's endpoints [46].

$$\|AYw - \theta MYw\|_2 \leq \varepsilon |\lambda_{\max}(A, M)| \|Yw\|_2, \tag{6.1}$$

where $|\lambda_{\max}(A, M)|$ is the largest-magnitude eigenvalue of (A, M). We estimate $\lambda_{\max}(A, M)$ crudely using the Krylov–Schur method implemented in the MATLAB `eigs` function [44] with the tolerance set to 10^{-2}. Approximate eigenpairs (Yw, θ) that do not satisfy (6.1) are deemed not to have converged. If the total number of converged eigenpairs corresponding to eigenvalues within $[a, b]$ does not match the inertia count, we deem Algorithm 3.1 not to have converged and continue iterating.

For the deflated Krylov solves, we use unrestarted GMRES via the MATLAB `gmres` function with the convergence tolerance set to 10^{-8}. The preconditioner is as described in section 4. To further reduce the iteration counts, we employ the continuation strategy described at the end of section 5.

6.1. Laplacian test problems. In our first set of tests, we apply our method to solve (1.1) for some pencils (A, M) generated by finite element discretizations of the Laplacian on a rectangle of dimensions $1 \times 2^{1/4}$ in 2D and a cuboid of dimensions $1 \times 2^{1/4} \times 3^{1/4}$ in 3D[4] with natural (Neumann) boundary conditions. We generated the meshes and matrices using the MATLAB Partial Differential Equations Toolbox, which uses P1 Lagrange elements. For each pencil, we first select an interval $[a, b]$ containing its 185 algebraically smallest eigenvalues and then select a set of $m = 100$ equally spaced shifts ω_j in $[a, b]$. For the right-hand sides, we select a vector f of the appropriate dimension with entries drawn at random from a standard normal distribution, scale f to have unit norm, and then set $f_j = f$ for all j. In terms of the application to frequency response analysis, these are low-frequency response problems for either a vibrating rectangular membrane (in 2D) or cuboidal acoustic cavity (in 3D). Table 6.1 summarizes these test problems.

The results of applying our method to these problems with several values of K are displayed in Table 6.2. The table reports the number of iterations required for the rational filtering phase of the algorithm to converge, the dimension n_{defl} of the deflation space, and the value of γ as defined in Theorem 4.1.[5] Here, $n_{\text{defl}} = 185$ because we use only the eigenvectors corresponding to eigenvalues within $[a, b]$ to

TABLE 6.1

Parameters for the test problems generated by finite element discretizations of the Laplacian on rectangular (2D) *and cuboid* (3D) *domains. The interval* $[a, b]$ *contains the algebraically smallest* 185 *eigenvalues of* (A, M).

Problem	Dimension	a	b
fem2D-50K	49,769	-1.00×10^{-1}	1.80×10^{3}
fem2D-100K	99,998	-1.00×10^{-1}	1.80×10^{3}
fem2D-200K	199,935	-1.00×10^{-1}	1.80×10^{3}
fem2D-500K	499,583	-1.00×10^{-1}	1.80×10^{3}
fem3D-50K	49,739	-1.00×10^{-1}	2.93×10^{2}
fem3D-100K	100,293	-1.00×10^{-1}	2.93×10^{2}
fem3D-200K	199,588	-1.00×10^{-1}	2.93×10^{2}
fem3D-500K	501,322	-1.00×10^{-1}	2.93×10^{2}

[4]These dimensions were selected to yield problems for which all eigenvalues are simple. This was done solely to facilitate comparison between the 2D and 3D cases, as it simplifies the task of finding intervals $[a, b]$ containing a given number of eigenvalues. Problems with eigenvalues of nonunit multiplicity generally present no issues for our method; indeed, this is one of its strengths.

[5]We compute γ by applying Algorithm 3.1 on an interval containing but slightly larger than $[a, b]$ to find the eigenvalue of (A, M) nearest to $[a, b]$ whose eigenvector is not used in the deflation space. We do this only to facilitate the discussion in this article; it is not necessary in practice.

TABLE 6.2
Results of applying the proposed method to the finite element test problems of Table 6.1 using only the eigenpairs within $[a, b]$ for deflation.

Problem	K	Filt. iters.	n_{defl}	γ	GMRES iters. Min.	GMRES iters. Max.	GMRES iters. Avg.	Residual Min.	Residual Max.
fem2D-50K	8	9	185	4.2×10^{-03}	1	15	10.0	7×10^{-10}	6×10^{-08}
	16	2	185	4.2×10^{-03}	1	8	6.1	3×10^{-11}	5×10^{-08}
	24	1	185	4.2×10^{-03}	1	6	4.5	4×10^{-11}	1×10^{-08}
fem2D-100K	8	9	185	4.2×10^{-03}	1	15	9.9	3×10^{-10}	3×10^{-08}
	16	2	185	4.2×10^{-03}	1	8	6.0	2×10^{-11}	7×10^{-08}
	24	1	185	4.2×10^{-03}	1	5	4.4	5×10^{-11}	2×10^{-08}
fem2D-200K	8	8	185	4.2×10^{-03}	1	15	9.8	6×10^{-10}	8×10^{-08}
	16	2	185	4.2×10^{-03}	1	8	6.0	3×10^{-11}	1×10^{-09}
	24	1	185	4.2×10^{-03}	1	6	4.5	7×10^{-11}	9×10^{-09}
fem2D-500K	8	8	185	4.2×10^{-03}	1	15	9.9	2×10^{-10}	2×10^{-08}
	16	2	185	4.2×10^{-03}	1	8	6.0	2×10^{-11}	7×10^{-09}
	24	1	185	4.2×10^{-03}	1	5	4.4	5×10^{-11}	2×10^{-08}
fem3D-50K	8	19	185	3.7×10^{-03}	1	18	11.6	7×10^{-11}	4×10^{-08}
	16	4	185	3.7×10^{-03}	1	11	7.1	5×10^{-11}	3×10^{-08}
	24	2	185	3.7×10^{-03}	1	7	5.1	4×10^{-12}	2×10^{-09}
fem3D-100K	8	18	185	2.8×10^{-03}	1	17	11.9	1×10^{-10}	8×10^{-09}
	16	3	185	2.8×10^{-03}	1	10	7.5	3×10^{-10}	7×10^{-08}
	24	2	185	2.8×10^{-03}	1	7	5.6	8×10^{-12}	8×10^{-10}
fem3D-200K	8	18	185	2.4×10^{-03}	1	17	12.0	6×10^{-11}	1×10^{-08}
	16	3	185	2.4×10^{-03}	1	10	7.6	2×10^{-10}	2×10^{-08}
	24	2	185	2.4×10^{-03}	1	9	5.8	1×10^{-11}	1×10^{-09}
fem3D-500K	8	17	185	2.2×10^{-03}	1	17	11.9	4×10^{-11}	8×10^{-09}
	16	3	185	2.2×10^{-03}	1	10	7.5	6×10^{-11}	6×10^{-09}
	24	2	185	2.2×10^{-03}	1	8	5.8	6×10^{-12}	8×10^{-10}

form the deflation space. The table also reports the minimum, maximum, and average number of GMRES iterations required to solve the deflated systems, and the minimum and maximum 2-norm residuals $\|f - (A - \omega_j M)x_j\|_2$ over all j.

Our method solves the systems successfully, attaining residuals on the order of 10^{-6} or less in all cases; this is more than sufficient accuracy for most frequency response applications. On the other hand, the values of γ are small: the eigenvalues of these pencils are densely packed at the low end of the spectrum, and as a result, each pencil has at least one eigenvalue outside but very near to $[a, b]$. This diminishes the effectiveness of the preconditioner: on average, the deflated systems require 4–10 iterations for the 2D problems and 6–12 for the 3D problems, depending on K. Increasing K does reduce the number of iterations required to reach convergence (for both phases of the computation), but this comes at a price. For the 2D problems, the product $s_K K$ ranges from about 80 to 110; for the 3D problems, the range is approximately 90 to 150. We conclude that for our method to be competitive with the naive approach, the time required to factor the systems must be around 100 times greater than the time required to solve with the factors. This is certainly not the case in 2D and is a tall order even in 3D for the sizes of the problems considered.

Fortunately, there is a way to improve things. During the rational filtering phase, it is common for some of the eigenvalues outside $[a, b]$ to converge in addition to those inside. In the experiments of Table 6.2, we discarded these and used only the eigenvectors corresponding to eigenvalues within $[a, b]$ to form the deflation space. If we

TABLE 6.3

Results of applying the proposed method to the finite element test problems of Table 6.1 *using all converged eigenpairs for deflation. The GMRES iteration counts have decreased significantly relative to the values reported in Table* 6.2 *at the expense of an increase in the residual norms.*

Problem	K	Filt. iters.	n_{defl}	γ	GMRES iters. Min.	GMRES iters. Max.	GMRES iters. Avg.	Residual Min.	Residual Max.
fem2D-50K	8	9	287	5.7×10^{-01}	1	2	1.9	3×10^{-07}	7×10^{-07}
	16	2	214	1.7×10^{-01}	1	2	1.7	6×10^{-08}	4×10^{-07}
	24	1	199	8.2×10^{-02}	1	1	1.0	8×10^{-08}	1×10^{-06}
fem2D-100K	8	9	335	8.5×10^{-01}	1	2	1.0	1×10^{-06}	3×10^{-06}
	16	2	219	2.1×10^{-01}	1	2	1.3	4×10^{-07}	3×10^{-06}
	24	1	201	1.0×10^{-01}	1	1	1.0	2×10^{-07}	3×10^{-06}
fem2D-200K	8	8	332	8.0×10^{-01}	1	2	1.1	2×10^{-06}	5×10^{-06}
	16	2	222	2.1×10^{-01}	1	1	1.0	2×10^{-06}	7×10^{-06}
	24	1	203	1.1×10^{-01}	1	1	1.0	1×10^{-06}	1×10^{-05}
fem2D-500K	8	8	352	9.5×10^{-01}	1	2	1.0	7×10^{-06}	1×10^{-05}
	16	2	230	2.5×10^{-01}	1	2	1.4	7×10^{-06}	3×10^{-05}
	24	1	209	1.3×10^{-01}	1	1	1.0	6×10^{-06}	5×10^{-05}
fem3D-50K	8	19	357	6.2×10^{-01}	1	2	1.7	1×10^{-07}	3×10^{-07}
	16	4	227	1.8×10^{-01}	1	2	1.4	7×10^{-08}	4×10^{-07}
	24	2	217	1.4×10^{-01}	1	1	1.0	5×10^{-08}	6×10^{-07}
fem3D-100K	8	18	375	6.7×10^{-01}	1	2	1.6	2×10^{-07}	6×10^{-07}
	16	4	239	2.1×10^{-01}	1	1	1.0	2×10^{-07}	1×10^{-06}
	24	2	217	1.4×10^{-01}	1	1	1.0	6×10^{-08}	5×10^{-07}
fem3D-200K	8	18	455	9.2×10^{-01}	1	1	1.0	6×10^{-07}	1×10^{-06}
	16	3	230	1.9×10^{-01}	1	2	1.1	3×10^{-07}	2×10^{-06}
	24	2	217	1.3×10^{-01}	1	1	1.0	8×10^{-08}	7×10^{-07}
fem3D-500K	8	17	544	$1.2 \times 10^{+00}$	1	1	1.0	2×10^{-06}	3×10^{-06}
	16	3	235	2.1×10^{-01}	1	2	1.0	1×10^{-06}	7×10^{-06}
	24	2	219	1.5×10^{-01}	1	1	1.0	4×10^{-07}	3×10^{-06}

deflate against the eigenvectors corresponding to these "extra" eigenvalues as well, we obtain the results shown in Table 6.3. Compared with the results of Table 6.2, the dimensions of the deflation spaces have increased significantly—more than twofold for some combinations of the parameters—but because we have computed more eigenvalues, the values of γ have increased greatly, and consequently our preconditioner has become more effective. Not one of the deflated systems requires more than 2 GMRES iterations to solve, and in many cases, the number is much closer to 1. The product $s_K K$ ranges roughly from 10 to 30. Our method is now competitive for the larger 2D problems and a clear winner for the larger 3D problems. The disadvantage to doing this is that the systems are solved less accurately: the residuals in Table 6.3 are noticeably larger than those in Table 6.2. This happens because the eigenvectors corresponding to eigenvalues outside $[a, b]$ are not computed to the same accuracy as those inside $[a, b]$ by the rational filtering method. Nevertheless, the residuals in Table 6.3 are still small enough for most frequency response applications.

6.2. SuiteSparse test problems. Next, we consider several problems involving pencils from real-world frequency response applications taken from the SuiteSparse matrix collection [9]. In addition to low-frequency problems involving these pencils, we consider mid-frequency problems as well to demonstrate the effectiveness of our method at solving such problems. The pencils and intervals $[a, b]$ are listed in Table 6.4. We construct the problems so that both the low- and mid-frequency

TABLE 6.4
Parameters for the test problems from the SuiteSparse collection. The symbol n_{eig} refers to the number of eigenvalues of the pencil in $[a, b]$.

Type	Problem	Dimension	a	b	n_{eig}
Low	Kuu/Muu	7,102	$-9.29 \times 10^{+00}$	$1.67 \times 10^{+03}$	184
	crystk03/crystm03	24,696	-1.00×10^{-01}	7.15×10^{-01}	190
	qa8fk/qa8 fm	66,127	-1.00×10^{-01}	$3.29 \times 10^{+01}$	262
	windscreen	22,692	-1.00×10^{-01}	$9.13 \times 10^{+06}$	234
	bs01	127,224	$-2.96 \times 10^{+03}$	$5.23 \times 10^{+07}$	167
Mid	Kuu/Muu	7,102	$-7.50 \times 10^{+03}$	$9.50 \times 10^{+03}$	184
	crystk03/crystm03	24,696	$-1.00 \times 10^{+03}$	$1.25 \times 10^{+03}$	190
	qa8fk/qa8 fm	66,127	$-1.00 \times 10^{+03}$	$1.01 \times 10^{+03}$	262
	windscreen	22,692	$-1.00 \times 10^{+07}$	$5.00 \times 10^{+07}$	234
	bs01	127,224	$-1.00 \times 10^{+07}$	$6.00 \times 10^{+07}$	167

TABLE 6.5
Results for the low-frequency problems from the SuiteSparse collection.

Problem	K	Filt. iters.	n_{defl}	γ	GMRES iters. Min.	GMRES iters. Max.	GMRES iters. Avg.	Residual Min.	Residual Max.
Kuu/Muu	8	13	252	3.7×10^{-01}	1	2	2.0	2×10^{-08}	7×10^{-08}
	16	2	196	7.3×10^{-02}	1	2	2.0	1×10^{-08}	1×10^{-07}
	24	1	192	5.9×10^{-02}	1	2	1.9	1×10^{-08}	2×10^{-07}
crystk03/crystm03	8	9	280	5.9×10^{-01}	1	2	1.8	9×10^{-07}	2×10^{-06}
	16	2	217	1.8×10^{-01}	1	2	1.3	5×10^{-07}	3×10^{-06}
	24	1	207	1.2×10^{-01}	1	1	1.0	2×10^{-07}	2×10^{-06}
qa8fk/qa8 fm	8	15	299	1.3×10^{-01}	1	3	2.7	2×10^{-09}	2×10^{-06}
	16	3	279	7.5×10^{-02}	1	2	2.0	1×10^{-09}	2×10^{-07}
	24	1	271	2.4×10^{-02}	1	2	2.0	7×10^{-10}	8×10^{-07}
windscreen	8	2	240	4.4×10^{-02}	1	5	4.1	8×10^{-07}	3×10^{-04}
	16	2	260	2.4×10^{-01}	1	1	1.0	1×10^{-07}	8×10^{-05}
	24	1	235	1.4×10^{-02}	1	3	2.8	1×10^{-06}	6×10^{-04}
bs01	8	8	315	8.5×10^{-01}	1	2	1.1	2×10^{-05}	4×10^{-05}
	16	2	205	2.1×10^{-01}	1	2	1.4	9×10^{-06}	5×10^{-05}
	24	1	189	1.3×10^{-01}	1	1	1.0	6×10^{-06}	6×10^{-05}

problems for a given pencil have the same number of eigenvalues within their respective intervals, enabling us to study how the performance of our method varies when only the location of the eigenvalues changes, not their number. Note that the windscreen pencil was used as a test problem in the paper of Meerbergen and Bai, wherein the authors solve a low-frequency response problem involving it with (in our notation) $[a, b] = [0, 10^4]$, in which the pencil has 12 eigenvalues. Here, we consider an interval roughly 1000 times wider and which contains nearly 20 times as many eigenvalues, a much more challenging problem.

The results of applying our method to these problems are summarized in Tables 6.5 (for the low-frequency problems) and 6.6 (mid-frequency problems). The residuals attained are on par with those reported for the Laplacian discretizations in Table 6.3, with the exception of the bs01 problem, for which they are a bit larger. Aside from the low-frequency windscreen problem with $K = 8$, no deflated system required more than 3 GMRES iterations to solve. The iteration counts for the rational filtering phase of the algorithm are generally higher for the mid-frequency problems than for the low-frequency ones, reflecting the greater challenge posed by an inte-

TABLE 6.6
Results for the mid-frequency problems from the SuiteSparse collection.

Problem	K	Filt. iters.	n_{defl}	γ	GMRES iters.			Residual	
					Min.	Max.	Avg.	Min.	Max.
Kuu/Muu	8	20	249	1.3×10^{-01}	1	3	2.7	2×10^{-08}	1×10^{-06}
	16	5	200	3.9×10^{-02}	1	3	2.5	3×10^{-08}	5×10^{-06}
	24	2	196	4.4×10^{-02}	1	2	2.0	3×10^{-08}	2×10^{-06}
crystk03/crystm03	8	33	340	9.8×10^{-02}	1	3	2.5	9×10^{-08}	3×10^{-07}
	16	6	232	2.6×10^{-02}	1	3	2.0	1×10^{-07}	6×10^{-07}
	24	2	208	4.8×10^{-02}	1	2	1.9	3×10^{-08}	5×10^{-06}
qa8fk/qa8 fm	8	19	359	1.2×10^{-01}	1	3	2.8	4×10^{-08}	7×10^{-06}
	16	5	301	6.1×10^{-02}	1	2	2.0	2×10^{-08}	8×10^{-06}
	24	2	288	4.1×10^{-02}	1	2	2.0	1×10^{-08}	3×10^{-07}
windscreen	8	11	539	4.0×10^{-01}	1	2	2.0	7×10^{-08}	2×10^{-07}
	16	8	310	1.1×10^{-01}	1	2	2.0	6×10^{-08}	3×10^{-07}
	24	3	261	2.3×10^{-02}	1	3	2.2	7×10^{-08}	5×10^{-07}
bs01	8	11	411	$1.3 \times 10^{+00}$	1	2	1.0	3×10^{-05}	5×10^{-05}
	16	4	246	3.1×10^{-01}	1	1	1.0	2×10^{-05}	8×10^{-05}
	24	2	206	1.1×10^{-01}	1	2	1.1	3×10^{-05}	2×10^{-04}

rior eigenvalue problem. Nevertheless, the GMRES iteration counts and residuals for the two types of problems are comparable. Because the cost of the rational filtering phase is amortized over all the frequencies at which the response is sought, this means that solving a mid-frequency problem with our method is asymptotically no more expensive than solving a low-frequency problem as the number of frequencies becomes large.

6.3. Execution time. We conclude this section with an example discussing the performance of our method in terms of time to solution. We consider the same fem3D-500K problem described above, taking $K = 16$, but we increase the number of shifts at which we solve (1.1) from $m = 100$ to $m = 1000$. We run both our method and the naive approach, where the latter is implemented using `mldivide` (i.e., "backslash") in MATLAB, which uses MA57 [10] for these problems. We run all computations in serial using a single thread[6] and record the execution times for each method.

The results are displayed in Table 6.7, which also breaks down the cost of our method by computational phase.[7] While computing the factors at each of the poles and applying the rational filter is not cheap, it enables us to solve each system in about 2.5x less time than the naive way; moreover, with enough systems to solve, the cost of the rational filtering phase can be amortized away. In this case, with $m = 1000$, the result is a roughly 1.75x improvement in the time to solution of the overall problem compared with the naive approach.

[6]While both the naive method and our method are generally too expensive to consider running this way, doing this allows us to avoid the challenges of comparing parallel implementations, which can be difficult to develop in MATLAB. We do not believe this is an issue, as both methods are embarrassingly parallel through the same mechanism—the independence of linear systems at different shifts—and so we expect the ratios of their execution times in serial and in parallel to be comparable.

[7]The reader may notice that it takes a little over 4x the amount of time to factor a shifted system at one of the filter poles as it does to solve a shifted system using `mldivide`. The reason this occurs is that MATLAB stores the LDL^T factors in compressed-sparse-column (CSC) format, and there is a cost associated with converting the factors to this format from the internal format of the MA57 algorithm used to compute them. The `mldivide` function solves the systems using the MA57 format directly and does not incur this cost.

TABLE 6.7
Execution times for our method vs. those for the naive approach for the fem3D-500K problem with $m = 1000$. All times are in seconds.

Method	Phase	Time (s) Average	Total
Alg. 5.1	Count eigenvalues in $[a, b]$	—	6.80×10^2
	Compute factors at poles	7.00×10^2 (per pole)	1.16×10^4
	Rational filtering (Alg. 3.1)	4.67×10^3 (per iteration)	1.40×10^4
	Solve systems	6.88×10^1 (per system)	6.88×10^4
	TOTAL	—	$\mathbf{9.50 \times 10^4}$
Naive	Solve systems	1.68×10^2 (per system)	1.68×10^5
	TOTAL	—	$\mathbf{1.68 \times 10^5}$

7. Conclusion. We have presented an algorithm for solving a sequence of shifted linear systems for which the system pencil is symmetric and definite and the shifts are drawn from a given real interval. The algorithm, which can be viewed as a multishift version of an algorithm of Meerbergen and Bai, parallelizes naturally and, as our experiments show, can be significantly more effective than the naive approach of factoring and solving each system independently.

We expect our algorithm to be most useful for problems of this sort for which the interval containing the shifts lies deep in the interior of the spectrum of the system pencil and also contains many of the pencil's eigenvalues. An example would be computing the frequency response of a system over a mid-frequency range in which the system has a large number of natural frequencies. Problems of this type are difficult to handle with traditional methods based on modal superposition or automated multilevel substructuring [3].

One limitation of our algorithm as presented here is that it relies heavily on the fact that the system pencil has all real eigenvalues. In the context of frequency response analysis, this corresponds to an undamped system. It should be possible to extend our method to handle damped systems with complex eigenvalues by using a filter with poles at equispaced points on a circle in the complex plane [1, 40] instead of the reciprocal Chebyshev polynomial filter used here. We plan to investigate this possibility in future work.

Appendix A. Proof of Theorem 4.1. The crux of the proof is the development of a bound on the error in the approximation of $1/(\lambda - \omega)$ for $\omega \in [a, b]$ by its polynomial interpolant (in ω) in the Chebyshev points (3.4) for each eigenvalue λ of (A, M) outside of $[a, b]$. We begin by expanding $1/(\lambda - \omega)$ on $[a, b]$ in a series of *Chebyshev polynomials of the first kind*; these are the polynomials T_k defined by $T_0(x) = 1$, $T_1(x) = x$, and for $k \geq 1$ by the recurrence

$$T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x).$$

The T_k are most useful for developing approximations to functions defined on $[-1, 1]$. Since our function is defined on $[a, b]$, we apply a linear change of variable and work instead with the polynomials

$$\widehat{T}_k(x) = T_k\left(\frac{2}{b-a}x - \frac{b+a}{b-a}\right).$$

The required expansion is given by the following proposition.

PROPOSITION A.1. *For $\lambda \notin [a,b]$,*

$$\frac{1}{\lambda-\omega} = \frac{4t}{(b-a)(1-t^2)}\left(1+2\sum_{k=1}^{\infty} t^k \widehat{T}_k(\omega)\right),$$

the series converging uniformly for $\omega \in [a,b]$, where

$$t = \widehat{\lambda} - \operatorname{sign}(\widehat{\lambda})\sqrt{\widehat{\lambda}^2-1}, \qquad \widehat{\lambda} = \frac{2}{b-a}\lambda - \frac{b+a}{b-a}.$$

Proof. See [46, Exercise 3.14] for the case where $[a,b]=[-1,1]$. The result follows from this basic case by changing variables as described above. ∎

Next, we need the following standard result, which bounds the error in a Chebyshev interpolant to a function in terms of the sizes of the coefficients of the function's Chebyshev expansion.

PROPOSITION A.2. *Let $f:[a,b]\to\mathbb{R}$ be continuous, and let p_K be the polynomial interpolant to f in the points* (3.4)*. If f admits the Chebyshev expansion*

$$f(\omega) = \sum_{k=0}^{\infty} c_k \widehat{T}_k(\omega)$$

on $[a,b]$, then

$$\|f-p_K\|_\infty \le 2\sum_{k=K}^{\infty} |c_k|,$$

where $\|\cdot\|_\infty$ denotes the supremum norm on $[a,b]$.

Proof. This is a consequence of the aliasing property of the $\widehat{T}_k$ over the points (3.4), the triangle inequality, and the fact that $\|\widehat{T}_k\|_\infty = 1$. See [28, section 6.3.1], [46, Chapter 4], and [48] for details. ∎

Combining Propositions A.1 and A.2 yields the required bound on the error in the interpolant to $1/(\lambda-\omega)$.

PROPOSITION A.3. *For $\lambda \notin [a,b]$,*

$$\left|\sum_{k=0}^{K-1} \ell_k(\omega)\frac{1}{\lambda-\zeta_k} - \frac{1}{\lambda-\omega}\right| \le \frac{16\rho^2}{(b-a)(\rho^2-1)(\rho-1)}\rho^{-K}$$

for all $\omega \in [a,b]$, where ζ_k and ℓ_k are as in (3.4) *and* (4.2)*, respectively, and*

$$\rho = (1+2\gamma)+\sqrt{(1+2\gamma)^2-1}, \qquad \gamma = \frac{\operatorname{dist}(\lambda,[a,b])}{b-a}.$$

Proof. For $k \ge 1$, the coefficient c_k of $\widehat{T}_k$ in the expansion of Proposition A.1 is

$$c_k = \frac{8t}{(b-a)(1-t^2)}t^k.$$

Noting that $|t|<1$ and applying Proposition A.2, we obtain

$$\left|\sum_{k=0}^{K-1} \ell_k(\omega)\frac{1}{\lambda-\zeta_k} - \frac{1}{\lambda-\omega}\right| \le \frac{8|t|}{(b-a)(1-t^2)}\frac{|t|^K}{1-|t|}.$$

Finally, from

$$|\widehat{\lambda}| = \frac{2}{b-a}\left|\lambda - \frac{a+b}{2}\right| = \frac{2}{b-a}\left[\operatorname{dist}(\lambda,[a,b]) + \frac{b-a}{2}\right] = 1 + 2\gamma,$$

it follows that $|t| = 1/\rho$, which yields the result. ∎

We can now prove Theorem 4.1.

Proof of Theorem 4.1. Let $\lambda_1, \ldots, \lambda_s$ denote the eigenvalues in $\Lambda_2(A, M)$. For each j, let ρ_j and γ_j be the values of ρ and γ from Proposition A.3 for $\lambda = \lambda_j$. Note that ρ_j is an increasing function of γ_j.

A straightforward calculation shows that

$$T(\omega)V_2 = V_2 \sum_{k=0}^{K-1} \ell_k(\omega)(\Lambda_2 - \zeta_k I)^{-1}(\Lambda_2 - \omega I).$$

Thus, the eigenvalues of $T(\omega)$ corresponding to the eigenvectors in V_2 are precisely the real numbers

$$\mu_j = \sum_{k=0}^{K-1} \ell_k(\omega)\frac{\lambda_j - \omega}{\lambda_j - \zeta_k}, \qquad j = 1, \ldots, s.$$

By Proposition A.3,

$$|\mu_j - 1| \le \frac{16\rho_j^2|\lambda_j - \omega|}{(b-a)(\rho_j^2 - 1)(\rho_j - 1)}\rho_j^{-K},$$

and since

$$|\lambda_j - \omega| \le \operatorname{dist}(\lambda_j,[a,b]) + (b-a) = (1+\gamma_j)(b-a) \le \rho_j(b-a),$$

we have

$$|\mu_j - 1| \le \frac{16\rho_j^3}{(\rho_j^2 - 1)(\rho_j - 1)}\rho_j^{-K}.$$

The theorem now follows by maximizing the right-hand side of this inequality over j, observing that it is a decreasing function of ρ_j and, hence, of γ_j. ∎

Appendix B. Symmetrization of the preconditioned system for x_2. In this appendix, we provide details of the symmetrization approach to solving the deflated system for x_2 mentioned briefly in section 4. The idea is to factor $M = LL^T$ in a Cholesky decomposition, multiply both sides of (2.1) by L^{-1} on the left, and change variables according to $y = L^T x$, yielding

$$(L^{-1}AL^{-T} - \omega I)y = L^{-1}f. \tag{B.1}$$

One can apply our method to this equivalent system and then recover x from y via a triangular solve with L^T. Since the "mass matrix" for (B.1) is the identity matrix, the projections that arise in the formation of the deflated system and our preconditioner will be orthogonal, leading to a symmetric preconditioned system.

In more detail, recall that $L^{-1}AL^{-T}$ and (A, M) have the same eigenvalues and that the matrix Q of eigenvectors of the former is related to that of the latter by $Q = L^T V$. Defining $Q_1 = L^T V_1$ and $Q_2 = L^T V_2$, the deflated system for x_2 is

$$(I - Q_1Q_1^T)(L^{-1}AL^{-T} - \omega I)(I - Q_1Q_1^T)Q_2x_2 = (I - Q_1Q_1^T)L^{-1}f, \tag{B.2}$$

and our preconditioner is

$$P(\omega) = \sum_{k=0}^{K-1} \ell_k(\omega)(I - Q_1 Q_1^T)(L^{-1}AL^{-T} - \zeta_k I)^{-1}(I - Q_1 Q_1^T). \tag{B.3}$$

One checks readily that the preconditioned matrix

$$T(\omega) = P(\omega)(I - Q_1 Q_1^T)(L^{-1}AL^{-T} - \omega I)(I - Q_1 Q_1^T)$$

satisfies

$$T(\omega)^T Q_2 = T(\omega) Q_2$$

and hence is symmetric over the subspace spanned by the eigenvectors in Q_2.

The only remaining issue to address is how to work with (B.2) and (B.3) without explicitly forming $L^{-1}AL^{-T}$, which may be expensive. We require only the ability to apply $L^{-1}AL^{-T}$ and $(L^{-1}AL^{-T} - \zeta_k I)^{-1}$ to vectors. The former can be done by multiplying by L^{-T}, A, and L^{-1} in succession, where the multiplications by L^{-T} and L^{-1} are accomplished via triangular solves. As for the latter, we have

$$(L^{-1}AL^{-T} - \zeta_k I)^{-1} = L(A - \zeta_k M)^{-1} L^T,$$

and so applying $(L^{-1}AL^{-T} - \zeta_k I)^{-1}$ to a vector can be done by multiplying by L^T, then $(A - \zeta_k M)^{-1}$, and finally L, where the multiplication by $(A - \zeta_k M)^{-1}$ is carried out using the factors of $A - \zeta_k M$ computed during the first phase of our algorithm.

Appendix C. Complexity analysis. Here, we give the details of the complexity analysis leading to the condition (5.1) discussed in section 5. We make the following assumptions:

- The time T_{fact} required to factor $A - \sigma M$ does not depend on σ.
- The time T_{subst} required to perform a solve with the factors of $A - \sigma M$ (a pair of triangular substitutions) also does not depend on σ.
- The cost of the factorizations and solves dominates that of all other operations in the algorithm so that the latter can be ignored.

Under the naive approach to (1.1), each system must be factored and solved individually at a cost of $T_{\text{fact}} + T_{\text{subst}}$ per system. As there are m systems, the total cost is $m(T_{\text{fact}} + T_{\text{subst}})$ if the systems are solved serially or $m(T_{\text{fact}} + T_{\text{subst}})/K$ if they are solved K-way in parallel.

For Algorithm 5.1, we pay KT_{fact} in serial for the factorizations at the filter poles, plus $c_K K T_{\text{subst}}$ for the rational filtering computation, where c_K is the number of applications of the filter required for the rational filtering algorithm to converge. We then pay $s_K K T_{\text{subst}}$ to solve each deflated system, giving a total cost of $KT_{\text{fact}} + c_K K T_{\text{subst}} + m s_K K T_{\text{subst}}$ in serial. For K-way parallel operation, we divide the serial cost by K to obtain $T_{\text{fact}} + c_K T_{\text{subst}} + m s_K T_{\text{subst}}$.

Whether the costs used are those for execution in serial or execution in parallel, we find that the cost of the naive approach will exceed that of Algorithm 5.1 when

$$\frac{T_{\text{fact}}}{T_{\text{subst}}} > \frac{K c_K + m s_K K - m}{m - K}.$$

In the large-m limit, the expression on the right-hand side of this inequality tends to $s_K K - 1$, yielding (5.1).

Acknowledgments. We thank Zhaojun Bai for inviting us to speak about this work in a minisymposium at the SIAM CSE23 meeting in Amsterdam. We thank Zhaojun Bai and Karl Meerbergen for feedback and discussions. We thank Gabriel Ervin from MathWorks for providing us with insight into the subtleties of MATLAB's routines for solving sparse symmetric indefinite systems.

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, https://doi.org/10.1137/130931035.

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

[3] J. K. Bennighof and R. B. Lehoucq, *An automated multilevel substructuring method for eigenspace computation in linear elastodynamics*, SIAM J. Sci. Comput., 25 (2004), pp. 2084–2106, https://doi.org/10.1137/s1064827502400650.

[4] S. Birk and A. Frommer, *A deflated conjugate gradient method for multiple right hand sides and multiple shifts*, Numer. Algorithms, 67 (2014), pp. 507–529, https://doi.org/10.1007/s11075-013-9805-9.

[5] T. F. Chan and M. K. Ng, *Galerkin projection methods for solving multiple linear systems*, SIAM J. Sci. Comput., 21 (1999), pp. 836–850, https://doi.org/10.1137/s1064827598310227.

[6] F. Chatelin, *Eigenvalues of Matrices*, revised ed., SIAM, Philadelphia, 2012, https://doi.org/10.1137/1.9781611972467.

[7] R. W. Clough and J. Penzien, *Dynamics of Structures*, 3rd ed., Computers & Structures Inc., Berkeley, CA, 2003.

[8] D. Darnell, R. B. Morgan, and W. Wilcox, *Deflated GMRES for systems with multiple shifts and multiple right-hand sides*, Linear Algebra Appl., 429 (2008), pp. 2415–2434, https://doi.org/10.1016/j.laa.2008.04.019.

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

[10] I. S. Duff, *MA57—A code for the solution of sparse symmetric definite and indefinite systems*, ACM Trans. Math. Software, 30 (2004), pp. 118–144, https://doi.org/10.1145/992200.992202.

[11] Y. Futamura and T. Sakurai, *z-Pares Users' Guide, Release* 0.9.5, 2014.

[12] R. G. Grimes, J. G. Lewis, and H. D. Simon, *A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 228–272, https://doi.org/10.1137/s0895479888151111.

[13] 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, https://doi.org/10.1137/140980090.

[14] N. Halko, P. G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), pp. 217–288, https://doi.org/10.1137/090771806.

[15] V. Hernandez, J. E. Roman, and V. Vidal, *SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems*, ACM Trans. Math. Software, 31 (2005), pp. 351–362, https://doi.org/10.1145/1089014.1089019.

[16] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Stand., 49 (1952), pp. 409–436, https://doi.org/10.6028/jres.049.044.

[17] 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, https://doi.org/10.11650/twjm/1500405869.

[18] 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, https://doi.org/10.1016/j.cam.2009.09.029.

[19] Intel Corporation, *Extended eigensolver routines*, in Developer Reference for Intel oneAPI Math Kernel Library for C, 1st ed., 2023.

[20] V. Kalantzis, A. C. I. Malossi, C. Bekas, A. Curioni, E. Gallopoulos, and Y. Saad, *A scalable iterative dense linear system solver for multiple right-hand sides in data analytics*, Parallel Comput., 74 (2018), pp. 136–153, https://doi.org/10.1016/j.parco.2017.12.005.

[21] V. Kalantzis, Y. Xi, and L. Horesh, *Fast randomized non-Hermitian eigensolvers based on rational filtering and matrix partitioning*, SIAM J. Sci. Comput., 43 (2021), pp. S791–S815, https://doi.org/10.1137/20m1349217.
[22] V. Kalantzis, Y. Xi, and Y. Saad, *Beyond automated multilevel substructuring: Domain decomposition with rational filtering*, SIAM J. Sci. Comput., 40 (2018), pp. C477–C502, https://doi.org/10.1137/17m1154527.
[23] T. Kato, *Perturbation Theory for Linear Operators*, Springer-Verlag, New York, 1966.
[24] R. Li, Y. Xi, L. Erlandson, and Y. Saad, *The eigenvalues slicing library (EVSL): Algorithms, implementation, and software*, SIAM J. Sci. Comput., 41 (2019), pp. C393–C415, https://doi.org/10.1137/18m1170935.
[25] Y. Maeda, T. Sakurai, and J. E. Roman, *Contour Integral Spectrum Slicing Method in SLEPc*, SLEPc Technical Report STR-11, 2016.
[26] O. A. Marques, *BLZPACK: Description and User's Guide*, Technical Report TR/PA/95/30, CERFACS, 1995.
[27] P.-G. Martinsson, *Randomized methods for matrix computations*, in The Mathematics of Data, IAS/Park City Math. Ser. 25, M. W. Mahoney, J. C. Duchi, and A. C. Gilbert, eds., American Mathematical Society, Providence, RI, 2018, pp. 187–231.
[28] J. C. Mason and D. C. Handscomb, *Chebyshev Polynomials*, CRC Press, Boca Raton, FL, 2003.
[29] K. Meerbergen and Z. Bai, *The Lanczos method for parameterized symmetric linear systems with multiple right-hand sides*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 1642–1662, https://doi.org/10.1137/08073144x.
[30] H. Murakami, *A filter diagonalization method by the linear combination of resolvents*, IPSJ Trans. Adv. Comput. Syst., 49 (2008), pp. 66–87, http://id.nii.ac.jp/1001/00018206/.
[31] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629, https://doi.org/10.1137/0712047.
[32] M. L. Parks, E. de Sturler, G. Mackey, D. D. Johnson, and S. Maiti, *Recycling Krylov subspaces for sequences of linear systems*, SIAM J. Sci. Comput., 28 (2006), pp. 1651–1674, https://doi.org/10.1137/040607277.
[33] B. N. Parlett, *The Symmetric Eigenvalue Problem*, SIAM, Philadelphia, 1998, https://doi.org/10.1137/1.9781611971163.
[34] E. Polizzi, *Density-matrix-based algorithm for solving eigenvalue problems*, Phys. Rev. B, 79 (2009), pp. 11512: 1–6, https://doi.org/10.1103/PhysRevB.79.115112.
[35] E. Polizzi, *FEAST Eigenvalue Solver v4.0 User Guide*, arXiv:2002.04807 [cs.MS], 2020.
[36] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003, https://doi.org/10.1137/1.9780898718003.
[37] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869, https://doi.org/10.1137/0907058.
[38] Y. Saad, M. Yeung, J. Erhel, and F. Guyomarc'h, *A deflated version of the conjugate gradient algorithm*, SIAM J. Sci. Comput., 21 (2000), pp. 1909–1926, https://doi.org/10.1137/s1064829598339761.
[39] 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, https://doi.org/10.1260/1748-3018.7.3.249.
[40] T. Sakurai and H. Sugiura, *A projection method for generalized eigenvalue problems using numerical integration*, J. Comput. Appl. Math., 159 (2003), pp. 119–128, https://doi.org/10.1016/S0377-0427(03)00565-X.
[41] T. Sakurai and H. Tadano, *CIRR: A Rayleigh-Ritz type method with contour integral for generalized eigenvalue problems*, Hokkaido Math. J., 36 (2007), pp. 745–757, https://doi.org/10.14492/hokmj/1272848031.
[42] K. M. Soodhalter, E. de Sturler, and M. E. Kilmer, *A survey of subspace recycling iterative methods*, GAMM Mitteilungen, 43 (2020), e202000016, https://doi.org/10.1002/gamm.202000016.
[43] A. Stathopoulos and K. Orginos, *Computing and deflating eigenvalues while solving multiple right-hand side linear systems with an application to quantum chromodynamics*, SIAM J. Sci. Comput., 32 (2010), pp. 439–462, https://doi.org/10.1137/080725532.
[44] G. W. Stewart, *A Krylov–Schur algorithm for large eigenproblems*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 601–614, https://doi.org/10.1137/S0895479800371529.
[45] P. T. P. Tang and E. Polizzi, *FEAST as a subspace iteration eigensolver accelerated by approximate spectral projection*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 354–390, https://doi.org/10.1137/13090866X.

[46] L. N. Trefethen, *Approximation Theory and Approximation Practice*, Extended Edition, SIAM, Philadelphia, 2020, https://doi.org/10.1137/1.9781611975949.
[47] Y. Xi and Y. Saad, *Computing partial spectra with least-squares rational filters*, SIAM J. Sci. Comput., 38 (2016), pp. A3020–A3045, https://doi.org/10.1137/16M1061965.
[48] K. Xu, *The Chebyshev points of the first kind*, Appl. Numer. Math., 102 (2016), pp. 17–30, https://doi.org/10.1016/j.apnum.2015.12.002.
[49] H. Zhang, B. Smith, M. Sternberg, and P. Zapol, *SIPs: Shift-and-Invert parallel spectral transformations*, ACM Trans. Math. Software, 33 (2007), pp. 9:1–9:19, https://doi.org/10.1145/1236463.1236464.