Electronic Transactions on Numerical Analysis.
Volume 45, pp. 16–32, 2016.
Published by Kent State University.
ISSN 1068–9613.

ETNA
Kent State University
http://etna.math.kent.edu

NEW FILTERING STRATEGIES FOR IMPLICITLY RESTARTED LANCZOS ITERATION*

ALEX BREUER†

Abstract. Implicitly-restarted Lanczos iteration methods are among the most powerful methods for finding a small number of eigenpairs of Hermitian matrices. These methods enjoy strong convergence and have manageable memory requirements. Implicitly-restarted Lanczos methods naturally find approximations to both the largest and smallest eigenpairs of the input matrix, but stagnation of Ritz values may lead to slow convergence, especially when one only wants eigenvalues from one end of the spectrum and memory is constrained. We develop a filtering strategy that breaks Ritz value stagnation for these one-sided eigenvalue problems. We demonstrate the reduction in matrix-vector product costs and note that this new variant has marked advantages when memory is constrained and when matrix-vector products are expensive.

Key words. Implicitly-restarted Lanczos, polynomial filtering, Hermitian eigenvalue problems, Krylov subspaces, Chebyshev filtering

AMS subject classifications. 65F15, 15A18, 90C99

1. Introduction.

Consider a Hermitian matrix A with the spectral decomposition

$$A = U\Lambda U^*,$$

and the problem of finding the k eigenpairs (λ_i, u_i) with either the largest or the smallest λ_i and

$$Au_i = \lambda_i u_i.$$

We call an eigenpair (λ_i, u_i) *leading* if λ_i is among the largest eigenvalues and *trailing* if it is among the smallest. Implicitly-restarted Lanczos methods [7, 15, 21, 34, 37, 41] are powerful methods for finding a few eigenpairs of a Hermitian A. All of these methods project A into a Krylov subspace to solve the eigenproblem.

DEFINITION 1.1. *Given a square matrix A, a vector $u^{(0)}$, and a natural number m, the mth* Krylov subspace *of A and $u^{(0)}$ is given by*

$$\mathcal{K}_m(A, u^{(0)}) = \operatorname{span}\{u^{(0)}, Au^{(0)}, A^2u^{(0)}, \ldots, A^{m-1}u^{(0)}\}.$$

In the absence of linear dependence of the $A^iu^{(0)}$, the dimension of the mth Krylov subspace is equal to m. In practice it is almost always the case that $\dim(\mathcal{K}_m(A, u^{(0)})) = m$.

All implicitly-restarted Lanczos methods use a Krylov subspace of fixed dimension m, iteratively getting start guesses that are better and better approximations of the wanted eigenvectors. It turns out that these start vectors can be generated implicitly; that is, one does not need to throw out all of the basis vectors for $\mathcal{K}_m(A, u^{(0)})$ when restarting with a new start vector $\tilde{u}^{(0)}$ to get $\mathcal{K}_m(A, \tilde{u}^{(0)})$. One can transform some of the basis vectors for $\mathcal{K}_m(A, u^{(0)})$ into basis vectors for $\mathcal{K}_m(A, \tilde{u}^{(0)})$. For the implicitly-restart Lanczos methods based on [15], the new start vector is $\tilde{u}^{(0)} := f(A)u^{(0)}$, where $f(x) = \prod_{j=1}^{d}(x - s_j)$ is a degree d polynomial whose roots are chosen to filter $u^{(0)}$ to make the cosine $\cos\vartheta(u_i, f(A)u^{(0)})$ larger for wanted eigenpairs (u_i, λ_i). Good choices of the roots s_j determines how fast the method converges.

*Received September 3, 2014. Accepted December 22, 2015. Published online on March 11, 2016. Recommended by M. Hochstenbach.

†Computational and Information Sciences Directorate, Army Research Laboratory, Aberdeen Proving Ground, MD 21005, USA (alexander.m.breuer.civ@mail.mil).

ETNA
Kent State University
http://etna.math.kent.edu

Much work has been done on how to find good shifts s_j. The original strategies in [15] suggest using either a set of Ritz values—eigenvalue approximations from the current Krylov subspace projection of A—that represents unwanted eigenvalues, or Leja points over an interval known to contain unwanted eigenvalues. Fast Leja points [5] have been shown to be almost as good as true Leja points, but are easier to compute. Using a set of Ritz values called *exact shifts* has been shown to be effective. Thick-Restart Lanczos [37] adds acceleration heuristics that choose a subset of unwanted Ritz values to use at each restart. The choice of m is known to be problem-dependent, but $m \geq 2k$ is a reasonable choice [23]. However, when $m \ll 2k$, exact shifts often give much slower convergence than when $m \geq 2k$.

Our contribution is a method that identifies and corrects the slow convergence of exact shift methods when m is small. We notice that stagnation of the Ritz values that are used to restart iteration causes the poor convergence of exact shift-restarted Lanczos when m is too small. The Ritz values that are used to restart the Lanczos algorithm keep assuming nearly the same values over and over again. We characterize properties of polynomial filters that can break that stagnation and develop a strategy to apply them when convergence may be suffering. From that, we develop a hybrid method that uses exact shifts when possible, and applies polynomial filters to break Ritz value stagnation when necessary.

2. Stagnation of Ritz values in exact shift Lanczos methods.

Ritz value stagnation is problematic for implicitly-restarted Lanczos methods when we want one end of the spectrum and when we use exact shifts. The exact shift strategy uses a subset of the unwanted Ritz values $\lambda_m^{(m)} \leq \lambda_{m-1}^{(m)} \leq \cdots \leq \lambda_{k+1}^{(m)}$ (when the leading k eigenvalues are wanted). When these Ritz values do not change much from restart to restart, the filtering polynomial may be inefficient. Rather than abandon exact shifts and use something else, we want to find a way to use exact shifts and recover from stagnation.

For wanted Ritz values, it has been shown [2,12] that convergence is guaranteed; we do not have to worry about wanted Ritz values stagnating. Instead, we do have to worry about stagnation of the *unwanted* Ritz values. Stagnation of unwanted Ritz values is problematic, as these directly determine the restarting of the Lanczos iteration. This is not a new idea; in fact, Jia noted as much in explaining the success of Leja points for small subspace sizes [21, pp. 200–201]. However, the way in which we address stagnation is unique.

To discuss more about stagnation, we first consider how the restart polynomial $f(x)$ affects convergence. This shows both how stagnation causes slow convergence and how stagnation can be broken. We then produce a quantitative definition of stagnation. With that, we can further modify [30, Theorem 2] to show that stagnation of the most extreme Ritz values is unavoidable. We then present an example that demonstrates Ritz value stagnation and shows how breaking that stagnation produces Ritz pairs with smaller residuals and reduced error.

2.1. Convergence of implicitly-restarted Lanczos.

Convergence of restarted Lanczos methods depends on both the length of the working subspace m and the filter polynomial $f^{(r)}(x)$, which is the product of the restart polynomials for restarts 1 through r. Clearly, when one wants the eigenvalue λ_i and can set $f^{(r)}(\lambda_j) = 0$ for all unwanted eigenvalues $j \neq i$, $f^{(r)}(A)u^{(0)} = u_i$. This also holds for a set of wanted eigenvalues $S \subset \{\lambda_j\}_{i=1}^n$; if $f^{(r)}(\lambda_j) = 0$ for all $\lambda_j \notin S$, then the set of Ritz values of $\mathcal{K}_{|S|}(A, u^{(0)})$ will be equal to S. Lehoucq and Sorensen noted that the ratio $f^{(r)}(\lambda_i)/f^{(r)}(\lambda_{i+1})$ gives the asymptotic effect of filtering in [10, p. 72]. We formalize their observation by adapting one of the classic results in [30].

COROLLARY 2.1 (Corollary of [30, Theorem 2]). *Assume that $\lambda_1 > \lambda_2 > \cdots > \lambda_N$. Let λ_i be an eigenvalue of A with associated eigenvector u_i such that $\langle u_i, u^{(0)} \rangle \neq 0$. Write $\lambda_i^{(m)}$ as the ith Ritz value of A restricted to $\mathcal{K}_m(A, f(A)u^{(0)})$. Pick some function $f^{(r)}(x)$,*

ETNA
Kent State University
http://etna.math.kent.edu

and let $\bar{f} = \max_{i<j\leq N} \left\{ f^{(r)}(\lambda_j)^2 \right\}$ *. Let*

$$K_i^{(n)} = \begin{cases} \prod_{j=1}^{i-1} \frac{\lambda_j^{(m)} - \lambda_N}{\lambda_j^{(m)} - \lambda_i} & \text{if } i \neq 1 \\ 1 & \text{otherwise} \end{cases} \qquad \text{and } \gamma_i = 1 + 2\frac{\lambda_i - \lambda_{i+1}}{\lambda_{i+1} - \lambda_N}.$$

We have

$$0 \leq \lambda_i - \lambda_i^{(m)} \leq (\lambda_i - \lambda_N) \left(\frac{\bar{f}}{f^{(r)}(\lambda_i)} \cdot \frac{K_i^{(n)} \tan \vartheta(u_i, u^{(0)})}{T_{m-i}(\gamma_i)} \right)^2.$$

Proof. We apply [30, Theorem 2] to the approximation of A in $\mathcal{K}_m(A, f^{(r)}(A)u^{(0)})$. Our point of departure is [30, Equation (2.11)]. Noting that the eigenbasis expansion of $f^{(r)}(A)u^{(0)} = \sum_{i=1}^N \alpha_i u_i = \sum_{i=1}^N c_i f^{(r)}(\lambda_i) u_i$ immediately leads to $\alpha_i = f^{(r)}(\lambda_i)c_i$, where $u^{(0)} = \sum_{i=1}^N c_i u_i$ is the expansion of $u^{(0)}$ in the eigenbasis of A. Applying this substitution to [30, Equation (2.11)] gives the desired result. ∎

REMARK 2.2. It is well-known that picking a filter such that $f^{(r)}(x)^2$ is relatively small for x in an interval $[a, b]$ that is known to contain only unwanted eigenvalues reduces the error. However, it requires a to be rather close to λ_N and b rather close to λ_{i+1} for a leading eigenvalue problem. Otherwise, the maximum of $f^{(r)}(x)^2$ over $[\lambda_N, \lambda_{i+1}]$ will occur near the endpoints: $\bar{f}$ will likely be equal to $f^{(r)}(\lambda_{i+1})$ or $f^{(r)}(\lambda_N)$.

REMARK 2.3. It is only necessary that the filter polynomial $f^{(r)}(x)$ has small magnitude in the neighborhoods of all unwanted eigenvalues. For example, if one wants the i leading eigenpairs, it may be far more efficient to construct $f^{(r)}$ based on $\min_{x\in\{\lambda_j\}_{j=i+1}^N} f(x)^2$ rather than $\min_{x\in[\lambda_N,\lambda_{i+1}]} f(x)^2$; that is, there may be subintervals of $[\lambda_N, \lambda_{i+1}]$ that are free of unwanted eigenvalues, and it does not matter if $f(x)^2$ is large in those sub-intervals. This observation justifies exact shifts as a reasonable choice.

REMARK 2.4. These bounds also show that simply choosing a polynomial that satisfies $f(\lambda_{i+1})^2 \gg f(\lambda_j)$ for most $i < j$ and with $f(\lambda_i)^2/f(\lambda_{i+1})^2$ small can shrink the approximation error. This gives insight into which polynomials will be good choices for breaking Ritz value stagnation.

2.2. A quantitative measure of stagnation. We introduce a quantitative definition of Ritz value stagnation.

DEFINITION 2.5. *Suppose the restarted Lanczos algorithm has been restarted r times, and uses a working subspace of size m. Let τ be a nonnegative real number less than 1, let w be a natural number less than r, and let $\boldsymbol{\lambda}_{i:j}^{(m,n)} := [\lambda_i^{(m,n)}, \lambda_{i+1}^{(m,n)}, \ldots, \lambda_j^{(m,n)}]$ be a vector whose entries are Ritz values at restart $n \leq r$ of the Implicitly-Restarted Lanczos algorithm. Then, Ritz values i through j are said to be* stagnant over w iterations *to tolerance τ when*

$$\min_{\substack{r-w\leq n\leq r \\ r-w\leq k\leq r \\ k\neq n}} \sin\vartheta(\boldsymbol{\lambda}_{i:j}^{(m,n)}, \boldsymbol{\lambda}_{i:j}^{(m,k)}) = \min_{\substack{r-w\leq n\leq r \\ r-w\leq k\leq r \\ k\neq n}} \left\{ 1 - \frac{\left\langle \boldsymbol{\lambda}_{i:j}^{(m,n)}, \boldsymbol{\lambda}_{i:j}^{(m,k)} \right\rangle}{\left\| \boldsymbol{\lambda}_{i:j}^{(m,n)} \right\| \left\| \boldsymbol{\lambda}_{i:j}^{(m,k)} \right\|} \right\} \leq \tau.$$

This definition is convenient because it is invariant to scaling of the spectrum of A. Clearly, when Ritz values i through j have stagnated to some small tolerance τ, then we can be sure that those Ritz values do not change much from restart to restart.

One may notice that our definition of stagnation requires minimization of an angle between two sets of unwanted Ritz values over all sets of unwanted Ritz values up to w restarts before the current one. Ritz values typically do not have small $\sin\vartheta(\boldsymbol{\lambda}_{i:j}^{(m,r)}, \boldsymbol{\lambda}_{i:j}^{(m,r-1)})$ for neighboring

ETNA
Kent State University
http://etna.math.kent.edu

restarts; rather, the Ritz values $\boldsymbol{\lambda}_{i:j}^{(m,n)}$ will be close to one from a restart in recent history. We have observed that Ritz value stagnation is often periodic, with $\sin\vartheta(\boldsymbol{\lambda}_{i:j}^{(m,r-2)},\boldsymbol{\lambda}_{i:j}^{(m,r)})$ being far smaller than $\sin\vartheta(\boldsymbol{\lambda}_{i:j}^{(m,r-1)},\boldsymbol{\lambda}_{i:j}^{(m,r)})$.

2.3. The asymptotically unavoidable stagnation of extreme unwanted Ritz values. Extreme Ritz values will tend to stagnate, and this is a consequence of the asymptotic convergence properties of Ritz values to eigenvalues in Krylov subspaces. Our adaptations of the classic bounds in [30] show that extreme Ritz values will always have small error when the Krylov subspace is nontrivial and when arithmetic is finitely-precise. We present our adaptations of [30, Theorem 2]; these are intended to show that the error $(\lambda_i^{(m)}-\lambda_i)$ may be small even when the dimension m of the Krylov subspace is small. We then derive another result from Corollary 2.1.

COROLLARY 2.6. *Suppose $N>m$, and let all the assumptions of Corollary 2.1 hold. Pick a nonnegative integer i with $i<m$. Pick any natural number k with $k<N-i$. It follows that the error $(\lambda_{m-i}^{(m)}-\lambda_{N-i})$ (note that $\lambda_{m-i}^{(m)}\geq\lambda_{N-i}$) obeys*

$$0\leq\lambda_{m-i}^{(m)}-\lambda_{N-i}\leq(\lambda_k-\lambda_{N-i})+(\lambda_1-\lambda_{k-1})\left(\frac{\bar{f}}{f^{(r)}(\lambda_{k-1})}\cdot\frac{\bar{K}_{(i,k)}^{(m)}\tan\vartheta(u_i,u^{(0)})}{T_{m-i}(\gamma_{i,k})}\right)^2,$$

where

$$\gamma_{i,k}:=1+2\frac{\lambda_i-\lambda_{k-1}}{\lambda_{k-1}-\lambda_1}$$

and

$$\bar{K}_{(i,k)}^{(m)}:=\begin{cases}\prod_{j=1}^{i-1}\frac{\lambda_{m-j}^{(m)}-\lambda_1}{\lambda_{m-j}^{(m)}-\lambda_i} & \text{if } i<N\\ 1 & \text{otherwise}\end{cases}.$$

Proof. We pick an appropriate k, and split [30, Equation (2.11)] into

$$\begin{aligned}\lambda_{N-i}-\lambda_{m-i}^{(m)}\leq{}&(\lambda_k-\lambda_{N-i})\frac{\sum_{j=k}^{N-i-1}p(\lambda_j)^2f^{(r)}(\lambda_j)^2c_j^2}{\sum_{j=1}^{N}p(\lambda_j)^2f^{(r)}(\lambda_j)^2c_j^2}\\&+(\lambda_1-\lambda_{k-1})\frac{\sum_{j=1}^{k-1}p(\lambda_j)^2f^{(r)}(\lambda_j)^2c_j^2}{\sum_{j=1}^{N}p(\lambda_j)^2f^{(r)}(\lambda_j)^2c_j^2}\end{aligned}\tag{2.1}$$

for any polynomial of degree $m-i$. We substitute the shifted and scaled Chebyshev polynomial $T_{m-i}(1+2(x-\lambda_{k-1})/(\lambda_{k-1}-\lambda_1))$ for $p(x)$, and notice that the first partial sum in (2.1) is less than $(\lambda_k-\lambda_{N-i})$. Simplification of the second partial sum proceeds as in [30] and gives the desired result. ∎

The main idea is that the error $(\lambda_i^{(m)}-\lambda_i)$ may shrink much more with small m (n in the notation in [30]) than the bounds in [30, Theorem 2] might suggest. All that is required is that $\gamma_{i,k}$ is just a little larger than 1, and the Chebyshev polynomial will obtain a large value for $T_{n-i}(\gamma_{i,k})$ when i is small. Corollary 2.6 has important implications for Ritz value stagnation in finite arithmetic: for example, when (u_N,λ_N) is unwanted and $\cos\vartheta(u_N,u^{(0)})$ cannot be made smaller than machine precision, $\tan\vartheta(u_N,u^{(0)})$ is finite. In that case, it is possible that one may find a suitable k such that the bounds from Corollary 2.6 are tight enough to place $\lambda_m^{(m)}$ in $[\lambda_N,\lambda_1]$, perhaps even close to λ_N. Even for not-too-large m, say 10 or more, one

ETNA
Kent State University
http://etna.math.kent.edu

can place $\lambda_m^{(m)}$ into a relatively narrow interval for $\tan\vartheta(u_N, u^{(0)}) \approx \epsilon_{\text{machine}}$. We illustrate Corollary 2.6 with an example.

EXAMPLE 2.7. Let A be a $2,000 \times 2,000$ diagonal matrix with eigenvalues $0, 10^{-3}$, $2 \times 10^{-3}, \ldots, 1 - 10^{-3}, 1, 10, 10 + 10^{-3}, \ldots, 11$. Now we consider the eigenvalue approximations of A restricted to $\mathcal{K}_{12}(A, u^{(0)})$ for any $u^{(0)}$. If we set $k := 1,001$, then $\gamma_{2,000,k} = 15$. We have $T_{11}(\gamma_{2,000,k})^2 \approx 1.4 \times 10^{34}$, but $\tan\vartheta(\cos^{-1}\vartheta(\epsilon_{\text{machine}})) \approx 2.66 \times 10^{32}$ if machine epsilon $\epsilon_{\text{machine}} = 10^{-16}$. By Corollary 2.6, $\lambda_N - \lambda_m^{(m)} \leq 1.01$, and $\lambda_m^{(m)} \notin [\lambda_{1,000}, \lambda_1]$ for *any* start vector $u^{(0)}$, since $\cos\vartheta(u_N, u^{(0)}) \geq \epsilon_{\text{machine}}$.

Based on this observation, we would expect $\lambda_m^{(m)}$ not to vary much from restart to restart once $\cos\vartheta(u_N, u^{(0)})$ goes to $\epsilon_{\text{machine}}$. Thus, if we restart the Lanczos iteration by filtering with only a few of the smallest Ritz values of A, we would expect stagnation. We illustrate this in the following example.

EXAMPLE 2.8. Let A be the same matrix as in Example 2.7, and let $u^{(0)} := [1, 1, \ldots]$. We now consider the Ritz values used to restart the Lanczos iteration for this particular $u^{(0)}$ to observe Ritz value stagnation and motivate methods to break stagnation.

Based on the predictions of Corollary 2.6 and the observations in Example 2.7, we expect the smallest Ritz values to experience stagnation. To show this, we computed restarted Lanczos iterations with exact shifts, restarting using the smallest three Ritz values. The three smallest Ritz values from ordinary IRLan with exact shifts are shown in Figure 2.1; these are the roots of the restart polynomial. It is clear that these Ritz values have stagnated after only about 20 iterations, and that the prediction from Example 2.7 is correct: $\lambda_m^{(m)}$ is never greater than 1.

To break Ritz value stagnation, we use two factors of a degree-6 Chebyshev polynomial filter to restart at restarts 21 and 22. If $\underline{\lambda}_m^{(m)}$ is the smallest Ritz value encountered in the previous 20 restarts, then we set our stagnation-breaking filter to be

$$p(x) = T_6((2x - \underline{\lambda}_m^{(m)})/\underline{\lambda}_m^{(m)}) = f^{(21)}(x) f^{(22)}(x),$$

where $f^{(21)}(x) = \prod_{i=1}^{3}(x - t_i)$, and $f^{(21)}(x) = \prod_{i=4}^{6}(x - t_i)$, and $t_1, t_2, \ldots, t_6$ are the roots of $p(x)$. The Ritz values with this stagnation breaking are also shown in Figure 2.1. Note that both methods performed the same number of matrix-vector products and restarts.

This filtering clearly breaks the Ritz value stagnation. To exhibit the impact of breaking stagnation, we show the sum residual of wanted Ritz values:

$$\left\| AU_{1:9}^{(m)} - U_{1:9}^{(m)} \Lambda_{1:9}^{(m)} \right\|_F = \sqrt{\sum_{i=1}^{9} \left\| Au_i^{(m)} - u_i^{(m)} \lambda_i^{(m)} \right\|^2}$$

and the cosine $\cos\vartheta(u_{1,001:2,000}, f(A)u^{(0)}) = \sum_{i=1,001}^{2,000} \cos\vartheta(u_i, f(A)u^{(0)})$. These are shown in Figure 2.2. Note that breaking stagnation results in smaller residuals and better exclusion of unwanted eigenvectors whose eigenvalues are in $[0, 1]$.

The problem with extreme eigenvalues that Corollary 2.6 illustrates is caused by exactly the same mechanism that causes Lanczos vectors to lose orthogonality between iterations [26, 32, 33], but reorthogonalization methods cannot solve this problem unless we keep the unwanted eigenvectors. For unwanted eigenvectors, restarting discards any information which we would need to perform reorthogonalization.

We notice that when $m - k$ is not too small, the $\bar{K}_{(i,k)}^{(m)}$ term can become large for some Ritz values, and some restart roots are not bound to be in a small interval. However, when $m - k$ is small, Corollary 2.6 can be made sufficiently tight to guarantee that exact shifts would be nearly identical from iteration to iteration.

ETNA
Kent State University
http://etna.math.kent.edu

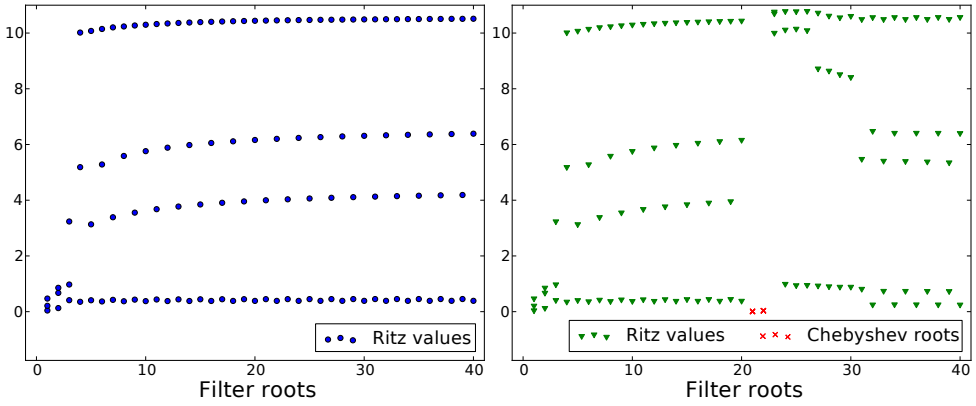

FIG. 2.1. *Smallest three Ritz values over restarts of Lanczos iteration to find the largest eigenvalues of the matrix in Example 2.8 when restarting using the three smallest unwanted Ritz values as exact shifts (left). These are also the roots of the exact shift restart polynomial $f^{(r)}(x)$ discussed in Section 2.1. The left-hand plot shows that the roots of the restart polynomial have stagnated after about 20 restarts. On the right are the smallest three unwanted Ritz values from restarted Lanczos iteration with the same subspace size and using exact shifts for restarts except for restarts 21 and 22. For those restarts, we set the restart polynomial to be the Chebyshev polynomial of the first kind of degree 6, scaled and shifted so that its roots were in $[0, \lambda_m^{(m)}]$, where $\lambda_m^{(m)}$ is the smallest Ritz value seen in all previous restarts. At restart 21, we restart with the first three roots of the Chebyshev polynomial, and at restart 22, we restart with the remaining three roots. The right-hand plot shows that applying the Chebyshev filter breaks Ritz value stagnation.*

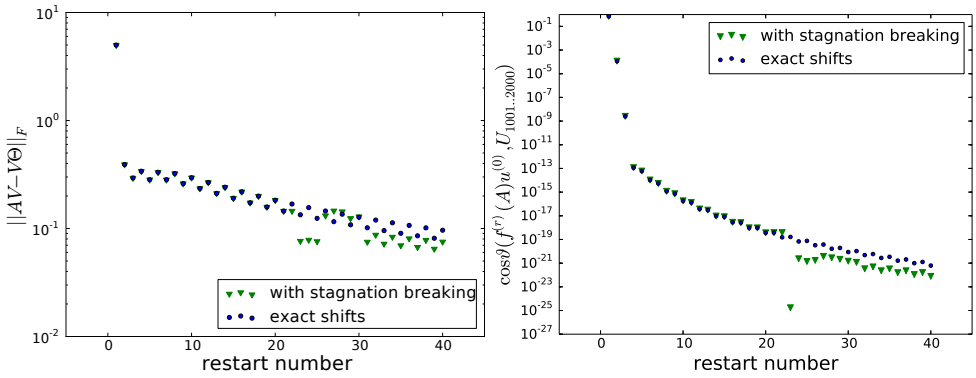

FIG. 2.2. *Residuals (left) and sum of cosines of the restart vector against the smallest 1,000 eigenvectors (right). Breaking Ritz value stagnation both reduces residuals and renders the restart vector more orthogonal to a set of unwanted eigenvectors. Note that the residual measured here is the square root of the sum of squares of residuals of all wanted eigenvectors.*

3. Choosing filtering intervals to break Ritz value stagnation. We have seen that stagnant Ritz values cause trouble for implicitly-restarted Lanczos methods when exact shifts are used and when the working subspace size m is not much larger than the number of wanted eigenpairs. Machine precision limits how small the bounds from Corollary 2.6 can get, so the smallest Ritz values can get stuck in some sense. However, one may break the stagnation by applying a filter polynomial as Example 2.8 showed. Though the idea of generating a filtering polynomial that minimizes over $[\lambda_N, \lambda_{i+1}]$ is a well-known trick, our contribution is that we construct our polynomials to filter different intervals than the shift selection strategies in [5–7, 15]. We focus on how we pick the lower bound a and upper bound b; our strategy for picking a and b is different from related methods. To filter over the interval $[a, b]$, we just use a shifted and scaled Chebyshev polynomial, though this is by no means the only possible choice.

ETNA
Kent State University
http://etna.math.kent.edu

3.1. Choosing a and b. Choosing the filtering interval endpoints is one of the key ingredients in our approach. Our interval strategy is based on some observations from Corollaries 2.1 and 2.6. These observations assume that the leading eigenvalues are wanted, and one may simply reverse the ordering of the Ritz- and eigenvalues and swap a and b to get the strategy when one wants the trailing eigenvalues.

1. We can see that exact shifts are an efficient choice because Ritz values will tend to "gravitate" towards clusters of Ritz values, but exact shifts will only be able to filter in $[\underline{\lambda}_m^{(m)}, \overline{\lambda}_{i+1}^{(m)}]$, where $\underline{\lambda}_m^{(m)}$ is the smallest Ritz value encountered and $\overline{\lambda}_{i+1}^{(m)}$ is the largest Ritz value encountered at index $i+1$;
2. stagnation of Ritz values will result in slow convergence;
3. stagnation can be broken with any polynomial that has a small magnitude for unwanted eigenvalues, a steep slope around the smallest wanted eigenvalue, and is large for wanted eigenvalues; and
4. exact shifts will not be able to filter eigenvalues smaller than $\underline{\lambda}_m^{(m)}$ effectively, where $\underline{\lambda}_m^{(m)}$ is the smallest $\lambda_m^{(m)}$ encountered.

We will, based on observation 1, let exact shifts filter in $[\underline{\lambda}_m^{(m)}, \overline{\lambda}_{i+1}^{(m)}]$—that is, we will not place stagnation breaking roots in $[\underline{\lambda}_m^{(m)}, \overline{\lambda}_{i+1}^{(m)}]$. We will break stagnation only when the Ritz values we use for restarting are stagnant. Our stagnation-breaking polynomial will be designed to filter out eigenvalues that we know exact shifts would fail to filter efficiently. Assuming an eigenproblem where the leading eigenvalues are wanted, we pick the lower endpoint a to be $\underline{\lambda}_m^{(m)} - \|r_{\underline{\lambda}_m^{(m)}}\|$, where $r_{\underline{\lambda}_m^{(m)}}$ is the Ritz residual for the Ritz value $\lambda_m^{(m)} = \underline{\lambda}_m^{(m)}$. We further pick the upper endpoint b to be $\underline{\lambda}_m^{(m)}$. For a trailing eigenvalue problem, we simply reverse the ordering of the eigenvalues and endpoints: a is the largest Ritz value encountered, and the width of the interval is the norm of the residual associated with the largest Ritz value encountered.

3.2. Comparison with other filtering intervals. Our interval choosing strategy may be contrasted with the filtering strategies in [5–7, 15] where exact shifts are not used. In all of these, $a \leq \lambda_m^{(m)}$ and $b \geq \lambda_{i+1}^{(m)}$ for a leading eigenvalue problem. The endpoints are updated at each restart such that $a \leq \lambda_m^{(m)}$ and $b \geq \lambda_{m-i}^{(m)}$ for some offset i. How to choose that offset is up to the user. It is interesting that choosing i to be small when m is close to k tends to work best [6, p. 15]; the upper endpoint b only gets close to λ_{k+1} when the restarts have filtered the start vector to be orthogonal to most other unwanted eigenvectors. Many restarts will have b far from λ_{k+1}, so they will be most biased to remove trailing eigenvectors. That behavior is just what we are trying to do when we break stagnation.

In [40], the authors propose the Chebyshev-Davidson method that uses interval filtering to accelerate convergence: the new search direction is given by $f(A)r$, where r is a residual vector, and $f(x)$ is a shifted and scaled Chebyshev polynomial. There are two advantages of applying filtering in a Krylov subspace that may save matrix vector products compared to filtering a less structured subspace as in the Davidson method. First, Lanczos methods do not need to explicitly compute the residual to determine convergence. Second, when one implicitly filters the start vector, one implicitly filters *all* the basis vectors of the subspace; d matrix-vector products filter m basis vectors. That is, if Q_m is a basis for $\mathcal{K}_m(A, u^{(0)})$, implicitly-restarting Lanczos iteration with a d-degree $f(x)$ yields $f(A)Q_{m-d}$ as a basis for $\mathcal{K}_{m-d}(A, f(A)u^{(0)})$, and then d Lanczos steps give $f(A)Q_m$ as a basis for $\mathcal{K}_m(A, f(A)u^{(0)})$. Krylov structure lets us filter m basis vectors with only d matvecs. In a Davidson algorithm, d matrix-vector products can only filter one basis vector.

It is notable that [40] uses a similar lower endpoint as our strategy in a Davidson algorithm but again chooses a larger upper endpoint. Our strategy chooses $b := \underline{\lambda}_m^{(m)}$ specifically to avoid choosing roots greater than $\underline{\lambda}_m^{(m)}$, where exact shifts may place roots. In some sense, our

ETNA
Kent State University
http://etna.math.kent.edu

strategy "trusts" exact shifts to do a good job and only filters in intervals that are disjoint from those that exact shifts can "know" about.

Algorithm 1 Implicitly-restarted Lanczos algorithm with exact shifts and stagnation breaking for a one-sided eigenvalue problem.

Require: Hermitian input matrix A, *a priori* chosen unit-length start vector $u^{(0)}$, stagnation breaking tolerance τ, stagnation window w, convergence residual tolerance t, working subspace size m, number of wanted eigenvalues k, stagnation breaking filter degree d

1: Generate an orthonormal basis for $\mathcal{K}_m(A, u^{(0)})$.
2: $r \leftarrow 0$
3: $f^{(0)}(x) = 1$
4: Get Ritz values $\lambda_1^{(m)}, \ldots, \lambda_m^{(m)}$ from restriction of A to $\mathcal{K}_m(A, u^{(0)})$, update $\underline{\lambda}_m^{(m)}$ and $\|r_{\underline{\lambda}_m^{(m)}}\|$.
5: **while** $\sum_{i=1}^{k} \|Au_i^{(m)} - u_i^{(m)}\lambda_i^{(m)}\|^2 > t^2$ **do**
6: **if** Ritz values $m : m-k$ have stagnated to τ over the last w restarts **then**
7: Set $s_1, s_2, \ldots, s_d$ to be the roots of the degree-d Chebyshev polynomial shifted and scaled to be in $[\underline{\lambda}_m^{(m)} - \|r_{\underline{\lambda}_m^{(m)}}\|, \underline{\lambda}_m^{(m)}]$, set $p \leftarrow d$.
8: **else**
9: Set p with $p \leq m-k$ by max-γ strategy in [37].
10: Set $s_1 \leftarrow \lambda_m^{(m)}, s_2 \leftarrow \lambda_{m-1}^{(m)} \ldots, s_p \leftarrow \lambda_{m-p}^{(m)}$.
11: **end if**
12: **for** $j \leftarrow 1, \ldots, p$ in steps of $m-k$ **do**
13: Extract $\mathcal{K}_k(A, (A - s_j I)f^{(r+m-k)}(A)u^{(0)})$ from $\mathcal{K}_m(A, f^{(r)}(A)u^{(0)})$ with $f^{(r+m-k)}(x) = (x - s_j)\cdots(x - s_{j+m-k})f^{(k)}(x)$.
14: Perform $m-k$ more Lanczos iterations to get $\mathcal{K}_m(A, f^{(r+m-k)}(A)u^{(0)})$.
15: $r \leftarrow r + m - k$
16: **end for**
17: Get Ritz values $\lambda_1^{(m)}, \ldots, \lambda_m^{(m)}$ from restriction of A to $\mathcal{K}_m(A, f^{(k)}(A)u^{(0)})$, update $\underline{\lambda}_m^{(m)}$ and $\|r_{\underline{\lambda}_m^{(m)}}\|$.
18: **end while**
19: Get Ritz vectors $u_1^{(m)}, \ldots, u_m^{(m)}$ and Ritz values $\lambda_1^{(m)}, \ldots, \lambda_m^{(m)}$ from restriction of A to $\mathcal{K}_m(A, f^{(k)}(A)u^{(0)})$ (Ritz values were already computed at line 17).
20: **return** wanted Ritz values and Ritz vectors.

3.3. Implicitly-restarted Lanczos with exact shifts and stagnation-breaking. We are now ready to present our shift strategy incorporated into the Implicitly-restarted Lanczos Algorithm (IRLan); it is presented in Algorithm 1. Our presentation of IRLan differs slightly from those in [10, 15]; we explain these in the following remarks.

REMARK 3.1. In [10, 15], the polynomial $f(x) = \prod_{j=1}^{d}(x - s_j)$ is evaluated all at once; $\mathcal{K}_{m-d}(A, f^{(k+d)}(A)u^{(0)})$ is extracted from $\mathcal{K}_m(A, f^{(k)}(A)u^{(0)})$ before the Lanczos factorization is advanced. We interleave these steps in lines 12–16, which is equivalent. Interleaving the steps allows one to have $d \geq m$.

REMARK 3.2. Our presentation of IRLan has an explicit residual-based convergence criteria, but explicitly computing the sum of residuals is not necessary; rather, we test the sum of residuals of Ritz pairs using the Lanczos recurrence $AQ_m = Q_m T_{m,m} + r_m e_m^*$, where Q_m is an orthonormal basis for $\mathcal{K}_m(A, u^{(0)})$ and e_m is the mth standard basis vector.

REMARK 3.3. Since we do not need to explicitly compute residuals per Remark 3.2, we do not explicitly need Ritz vectors. Since the implicit restarts maintain the tridiagonality of

ETNA
Kent State University
http://etna.math.kent.edu

$T_{m,m}$, Ritz values can be computed with complexity $O(m^2)$ (e.g., [18]) rather than the $O(m^3)$ for a generic full eigensolve.

4. Numerical examples. We demonstrate the performance of IRLan with stagnation breaking on four eigenvalue problems.[1] All calculations were performed on a Dell Dimension workstation running Redhat Linux 6.5 with 16 Xeon X5650 cores and 12 GB of memory. We used Octave 4.0.0 [19]. LAPACK and BLAS were provided by ATLAS 3.10.2 [35].

In all cases, we constrain memory to show the advantage of stagnation breaking when the working subspace size is less than twice the number of desired eigenpairs. Since eigenvalue calculation depends on the initial guess—which we choose as a random vector of uniformly-distributed values—we perform multiple runs and present averaged results. When comparing across different eigenvalue computation methods, we use the same series of random start vectors.

We compared our method against filtering with Fast Leja points using the same strategy as in [5–9]: we choose Fast Leja points in $[\underline{\lambda}_m^{(m)}, \overline{\lambda}_{m-1}^{(m)}]$, where $\underline{\lambda}_m^{(m)}$ is the smallest Ritz value encountered, and $\overline{\lambda}_{m-1}^{(m)}$ is the largest Ritz value at index $m-1$ encountered. We also compared against the Chebyshev-Davidson method in [40], and Implicitly-restarted Lanczos with exact shifts (no stagnation-breaking). We used the MATLAB code from Y. Zhou's web site [39] to get our Chebyshev-Davidson results. We used our own fast Leja point generation code, which we found comparable to IRBLEIGS [4] with unweighted fast Leja points. For exact shifts, we selected shifts using the max-γ strategy from Thick Restart Lanczos [37]; we noticed that this produced equivalent matrix-vector product counts compared to the Fortran implementation of TRLan [36] accessed via the SLEPc4Py [1] bindings. The only difference between IRLan with exact shifts and IRLan with stagnation breaking is the stagnation breaking; all other shifts are the same.

We present four examples: the small diagonal example from Example 2.7 to illustrate the effect of the input parameters of Algorithm 1, another small example to allow comparison against results reported in [9], and two large examples on which timing results are meaningful.

For subspaces with m close to the number of wanted eigenvalues k, the Chebyshev-Davidson software resets m if it is too small; we generated results for m as small as was allowed. We note that all methods were implemented in Octave to allow for timing comparison in Section 4.3. Our results show that stagnation breaking requires fewer matrix-vector products than the reference methods; we can expect stagnation breaking to have advantages when matrix-vector products are the predominant computational cost.

4.1. The diagonal matrix from Example 2.7. The matrix from Example 2.7 happens to be a nice choice for demonstrating positive attributes of Algorithm 1. The matrix has a wide gap $[1, 10]$ in which there are no eigenvalues. Though Fast Leja points tend to work well when the size of the working subspace m is close to the number of wanted eigenvalues k, any Leja point that falls in $[1, 10]$ will be a poor choice. However, there is no way to know how to avoid placing Leja points in intervals free of eigenvalues without already knowing what those intervals are beforehand.

4.1.1. Matrix-vector product use. We first study the number of matrix-vector products needed to get convergence of the first 5 and the first 10 eigenpairs of the matrix from Example 2.7. Based on the stagnation of Ritz values in Example 2.8, we would expect stagnation-breaking to have a positive impact on IRLan with exact shifts. Also, we would expect Fast Leja points to encounter difficulty with this matrix, since it has a wide interval

[1] The author's MATLAB implementation of IRLan with all restart methods analyzed here is available on github: https://github.com/alexmbreuer/irlan.

ETNA
Kent State University
http://etna.math.kent.edu

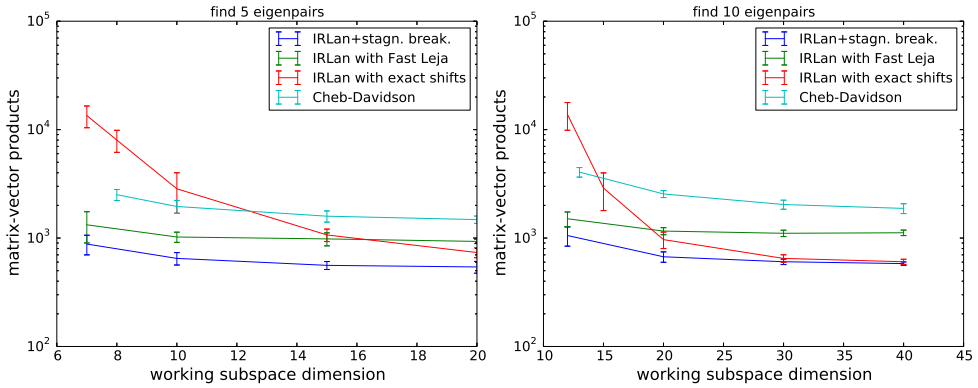

FIG. 4.1. *Matrix-vector products needed by ordinary IRLan with exact shifts and IRLan with stagnation breaking (IRLan + stag. break.) to compute the largest 5 and 10 eigenpairs of A from Example 2.7 given different working subspace dimensions. Matrix-vector products for IRLan with Fast Leja points and for Chebyshev-Davidson are also shown. Plots to find 5 eigenpairs are in the left-hand plot; plots to find 10 are on the right. These plots represent the averages over 10 runs with random start vectors; error bars show three standard deviations from the means.*

that is free of eigenvalues. Finally, we would not be surprised to see that Chebyshev-Davidson requires more matrix-vector products than restarted Lanczos methods; Chebyshev-Davidson can only refine one basis vector with each filter application, rather than implicitly filtering all basis vectors at the same time. Our expectations were not disappointed.

The number of matrix-vector products to find the largest 5 and largest 10 eigenpairs of the matrix from Example 2.7 is shown in Figure 4.1. We performed 10 runs of each method and present the average number of matrix-vector products. An important feature of the plots in Figure 4.1 is that the matrix-vector product use of IRLan with stagnation breaking grows more slowly as the working subspace size is restricted. This may be contrasted with standard IRLan with exact shifts, which requires almost an order of magnitude more matrix-vector products to find k eigenvectors when the size of the working subspace is only $k+2$.

4.1.2. The τ parameter in IRLan with stagnation breaking. We study the effect of the τ parameter that determines Ritz value stagnation (used on line 6 of Algorithm 1) on convergence of eigenvalues. We ran IRLan with stagnation breaking on A from Example 2.7 with $10^{-6} \leq \tau \leq 10^{-3}$ and $w := 4$; all other parameters were set to values used in the previous comparisons. We show the number of matrix-vector products versus working subspace size in Figure 4.2; the error bars in this figure represent one standard deviation from the mean of the 10 runs. There are differences between runs with differing values of τ, but all of the standard deviations overlap. One may conclude that setting τ to be between 10^{-6} and 10^{-3} should produce equivalent results to those reported in this paper. In general, τ should be set according to how quickly the unwanted, opposite end of the spectrum converges in the Krylov subspace: faster convergence of unwanted eigenvalues requires smaller τ, and slower convergence requires larger τ.

4.1.3. The w parameter in IRLan with stagnation breaking. Along with the τ parameter, the w parameter determines how often IRLan with stagnation breaking performs stagnation-breaking restarts. We examine the difference in matrix-vector product costs when w is varied. We set $\tau := 5 \times 10^{-6}$ and $w := 3, 4, 5, 6$; all other parameters were set to values used in the previous comparisons. We ran on A from Example 2.7 and report the number of matrix-vector products in Figure 4.3. Again, we ran 10 runs with different start vectors and averaged the results. Like in the experiment with τ, it is evident that varying w does not

ETNA
Kent State University
http://etna.math.kent.edu

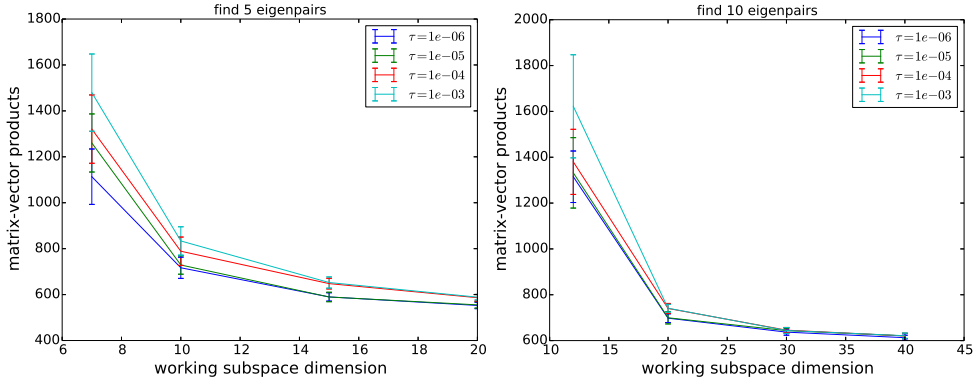

FIG. 4.2. *Matrix-vector products as a function of τ from line 6 of Algorithm 1 for finding 5 (left) and 10 (right) eigenvectors of A from Example 2.7. These results were averaged over 10 runs with random start vectors. The y-axis is linear, and the error bars represent one standard deviation from the means.*

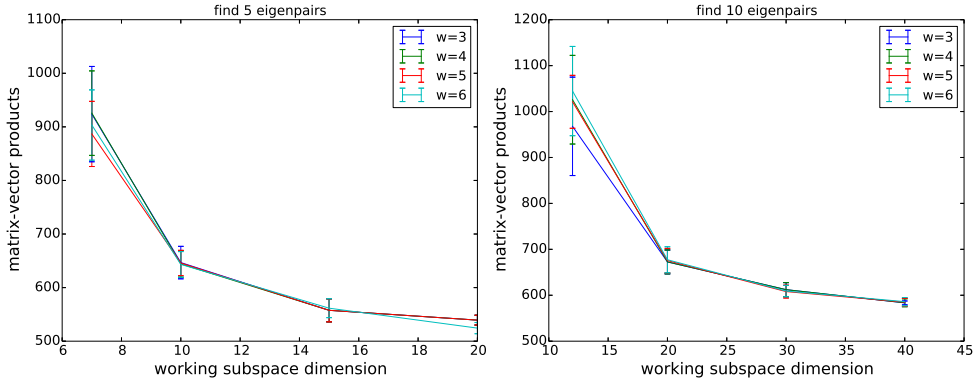

FIG. 4.3. *Matrix-vector products as a function of w from line 6 of Algorithm 1 for finding 5 (left) and 10 (right) eigenvectors of A from Example 2.7. These results were averaged over 10 runs with random start vectors. The y-axis is linear, and the error bars represent one standard deviation from the means.*

affect the matrix-vector product cost appreciably. A value of $3 \leq w \leq 6$ should be expected to reproduce the results in this paper.

4.2. Finding singular values and vectors. The eigenvalue problem is closely related to the singular value problem [17, Theorem 3.2] of finding $A = U\Sigma V^*$, where U and V are unitary and Σ is diagonal and positive semidefinite. A well-known trick [4, 20] is to take a (possibly rectangular) input matrix A and define

$$M = \begin{bmatrix} & A \\ A^* & \end{bmatrix}. \tag{4.1}$$

Each eigenvalue of M is then either a singular value of A or -1 times a singular value of A. Eigenvectors of M give both left and right singular vectors of A. When finding the largest singular values and their singular vectors, M might give slower convergence of singular value approximations to singular values [20], but does have the advantage that one can compute the matrix-vector products on A and A^* in parallel when evaluating a matrix-vector product on M. If A is sparse, this is equivalent to evaluating a compressed sparse column and a compressed sparse row product [10, Ch. 10].

ETNA
Kent State University
http://etna.math.kent.edu

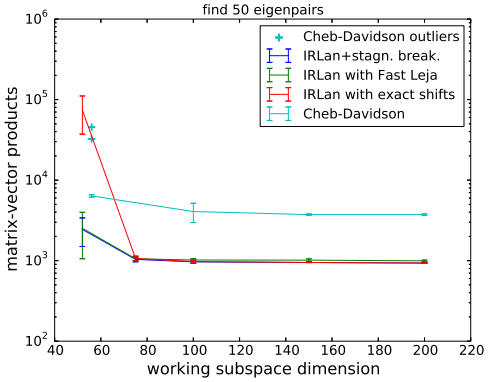

FIG. 4.4. *Matrix-vector products needed by IRLan with stagnation breaking, Fast Leja points, ordinary exact shifts and Chebyshev-Davidson to compute the 50 largest eigenvalues of M formed from well1850 via (4.1). These results were averaged over 10 runs with random start vectors, the error bars represent three standard deviations from the means. For two runs at 56 dimensions, Chebyshev-Davidson required substantially more iterations than for the others; we treat those two outliers separately.*

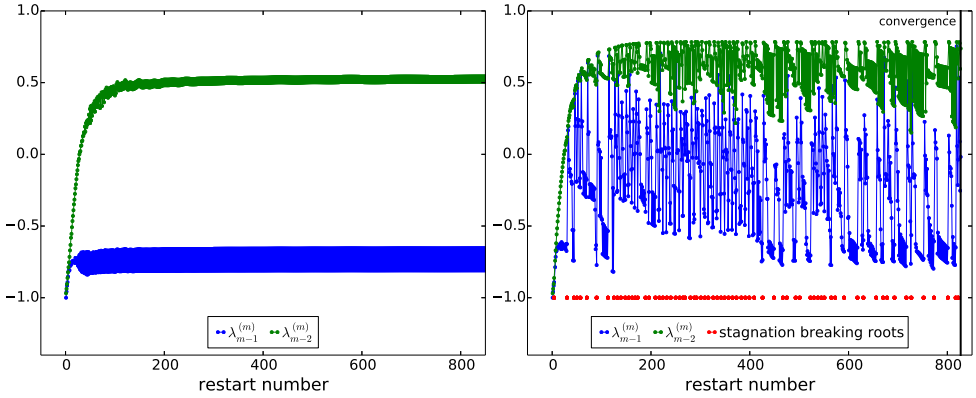

FIG. 4.5. *Smallest 2 Ritz values from an arbitrary run for finding 50 eigenvalues of the well1850 M matrix for IRLan with ordinary exact shifts (left) and with stagnation breaking (right). The stagnation breaking plot also shows the roots of the Chebyshev polynomials that break stagnation, marked in red. The left-hand plot only shows the roots of the first 850 restart polynomials, but the run took 34,517 restarts to obtain convergence. In contrast, IRLan with stagnation breaking converged after 827 restarts. Exact shifts exhibit stagnation that is effectively broken in the right-hand plot.*

We use the matrix well1850 from the University of Florida sparse matrix collection [16] (also in the Matrix Market collection [14]) and look for the 50 largest singular values. The number of matrix-vector products is shown in Figure 4.4.

To confirm our expectations from Corollary 2.6 on stagnation of Ritz values, we plot the smallest two Ritz values from an arbitrary one of the left-most runs of IRLan in Figure 4.5: finding 50 eigenvalues using a working subspace size of 52. These correspond to the unwanted Ritz values that define the restart polynomial. We also plotted the smallest Ritz values from the run of IRLan with stagnation breaking using the same start vector as the exact shift run and the same working subspace size. The figure shows that Ritz values from ordinary IRLan with exact shifts do indeed experience stagnation. It is also evident that stagnation breaking prevents the Ritz values from repeatedly assuming the same values over and over.

This matrix allows us to compare to the results reported in [9, Example 2]. Finding the

ETNA
Kent State University
http://etna.math.kent.edu

largest singular value to a convergence tolerance of 10^{-6} is reported to take 72 matrix-vector products with IRBLB and 82 matrix-vector products with IRBLA [8] when using a working subspace size of 5. We computed the largest singular value of M using the same subspace size with stagnation breaking IRLan with $\tau := 5 \times 10^{-6}$ and with a convergence tolerance of 10^{-6}, which required an average of 77.6 matrix-vector products over 10 runs. When we set $\tau := 10^{-3}$, IRLan with stagnation breaking required an average of 59.2 matrix vector products per iteration.

4.3. Two graph eigenproblems. Spectral graph analysis requires finding eigenvectors of matrices derived from the graph's adjacency matrix [13, 22, 25, 27, 29, 31, 38]. These eigenproblems are typically somewhat hard, and arbitrarily many eigenvectors may be required. Some important classes of graphs—scale-free networks [11]—may produce matrices with many nonzero entries, which leads to expensive matrix-vector products. When matrix vector products are expensive relative to other arithmetic operations, smaller matrix vector product usage correlates directly with shorter wall clock times. We examine two normalized graph Laplacian matrices [28]: one from a $10,000 \times 10,000$ synthetic matrix generated using the GRAPHGEN graph generator implementation in SNAP [24] (using the Albert-Barabási adjacency model [3]), and the other from University of Florida sparse matrix collection [16]. We obtain a leading eigenvalue problem from the normalized Laplacian by applying a shift. The normalized Laplacian is $L := I - D^{-1/2}AD^{-1/2}$, where A is the adjacency matrix of the graph and D is a diagonal matrix whose entries are vertex degree. All eigenvalues of L are in $[0, 2]$. We set our input matrix to be

$$A_{\text{input}} := 2I - L, \tag{4.2}$$

so all eigenvalues are still in $[0, 2]$ and the largest ones are wanted. Both matrices are large enough to provide meaningful wall clock measurements. For both matrices we measure wall clock time using the Octave/Matlab `tic` and `toc` commands. We used a convergence tolerance of 10^{-8} for both graphs. We used $w := 4$ and $\tau := 5 \times 10^{-6}$ for stagnation-breaking IRLan. We performed five runs with random start vectors, and present the average wall clock times and number of matrix-vector products.

A contrast between Lanczos solvers and Chebyshev-Davidson with respect to run times is apparent in this section. The results reported here contrast with those reported in [40], where Chebyshev-Davidson was reported to outperform TRLan. In the results reported in [40, Figure 6.2], one can see that run times decrease despite increasing matrix-vector product counts. This suggests that the savings in orthogonalization and restarting due to better filtering offset the extra filtering costs; relatively inexpensive matrix-vector products are a prerequisite. For many applications, one will have sufficiently inexpensive matrix-vector products, but when the input matrix is too dense, one may witness relatively expensive matrix-vector products. When matrix-vector products are sufficiently expensive, performing fewer matrix vector products will realize savings. The following examples demonstrate this.

4.3.1. A synthetic Albert-Barabási graph. We generated a $10,000 \times 10,000$ synthetic graph using the `graphgen` generator in SNAP [24], where each vertex brings on average 128 edges to the graph (the graphgen command was `graphgen -g:k -n:10000 -k:128`). The resulting normalized Laplacian matrix had $2,553,488$ nonzero elements. Additionally, the matrix had a large gap between the smallest and second-smallest eigenvalues: the smallest eigenvalue is 0, but the second-smallest is ≈ 0.876326; the largest eigenvalue of A_{input} from (4.2) is then 2, and the second-largest is ≈ 1.123674.

We ran IRLan with stagnation breaking using $w := 4$ and $\tau := 5 \times 10^{-6}$. We tried to get the fastest possible run times for Chebyshev-Davidson by using degrees of $2, 5$, and 15; larger degrees caused Chebyshev Davidson to fail. We note that $d := 15$ produced the best run times.

ETNA
Kent State University
http://etna.math.kent.edu

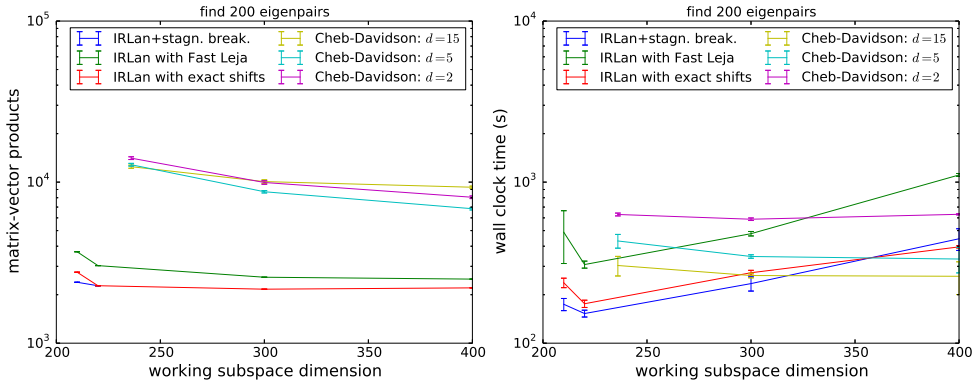

FIG. 4.6. *Matrix-vector products (left) and wall clock times (right) for IRLan with Leja Points, stagnation breaking, ordinary exact shifts and Chebyshev-Davidson to find the 200 largest eigenvalues of A_{input} for the graph Laplacian matrix for a 10,000 vertex synthetic graph. The Krylov subspace methods show a distinct advantage over Chebyshev-Davidson for matrix-vector products, but Chebyshev-Davidson has been shown to have smaller costs for orthogonalization and refinement. Nevertheless, matrix-vector products are sufficiently expensive to render Chebyshev-Davidson slower for small search spaces.*

The gap between the two smallest wanted eigenvalues apparently limits the polynomial degree that can be used in Chebyshev-Davidson; even simple power iteration is effective for this eigenvector. With a sufficiently-high degree Chebyshev filter polynomial $f(x)$, $\cos \vartheta(f(A)x, u_1) \approx 1$ for *any* x. That is, applying the filter polynomial to any vector gives a vector that is numerically indistinguishable from an eigenvector. Once this eigenvector is in the working subspace, the Chebyshev-Davidson code cannot expand its search subspace. At each subspace expansion step, the new vector $f(A)x \approx u_1$ is already in the subspace, so the code simply replaces the filtered vector $f(A)x$ with (another) random vector and continues trying to expand the search space [39, file bchdav.m, line 689]. An apparently infinite loop results. We saw this behavior for polynomial degrees greater than 20, so we report results obtained with degrees of 2, 5 and 15.

We computed the 200 leading eigenvectors of A_{input} using subspace dimensions ranging from 210 to 400. The matrix-vector products and wall clock times are shown in Figure 4.6. One can see that the smallest wall clock times are for stagnation breaking Lanczos, but Chebyshev-Davidson exhibits roughly monotonic run times. This is in contrast to the implicitly-restarted Lanczos methods, whose orthogonalization costs scale quadratically with subspace dimension. A known advantage of Chebyshev-Davidson is that it may have reduced orthogonalization costs [40], which likely accounts for the observed monotonicity. However, Chebyshev-Davidson has larger wall clock times for dimensions under 300; this is because matrix-vector products are simply too expensive to overcome any savings in orthogonalization costs.

4.3.2. The Citeseer co-author graph. The co-author graph constructed from Citeseer data [16, coPapersCiteseer] has a ratio of edges to vertices that, like the previous example, results in matrix-vector products that are expensive relative to other linear algebra operations. The matrix is $434,102 \times 434,102$ but has $320,507,542$ nonzero entries. The relatively expensive matrix-vector products dominate compute costs and render Lanczos methods cheaper than Chebyshev-Davidson. Again, we transform the normalized graph Laplacian into a leading eigenproblem per (4.2).

For this example, we computed $k := 40, 65, 90, 115$ with search spaces of $k + 10$ for Lanczos methods, and the smallest search spaces that Chebyshev-Davidson would allow. We ran IRLan with stagnation breaking using $w := 4$ and $\tau := 5 \times 10^{-6}$. The number of matrix-vector products and the wall clock times are shown in Figure 4.7.

ETNA
Kent State University
http://etna.math.kent.edu

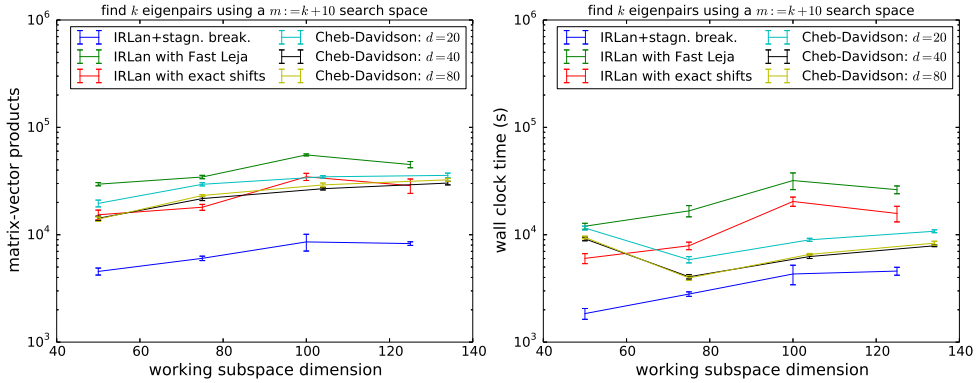

FIG. 4.7. *Matrix-vector products (left) and wall clock times (right) for IRLan with stagnation breaking, IRLan with exact shifts, IRLan with Fast Leja points and Chebyshev-Davidson to find the k largest eigenvalues of* A_{input} *for the graph Laplacian matrix for the coauthor graph from the University of Florida sparse matrix collection.*

As for the previous example, we tried to get the best run times out of Chebyshev-Davidson by using polynomial degrees of $d := 20, 40, 80$. As in [40], run times decreased as polynomial degree increased, but $d := 80$ is roughly equivalent to $d := 40$. A point of diminishing returns appears to have been reached. However, that $d := 20$ is slower than $d := 40$ suggests that degrees less than 20 will have slower run times.

Again, any savings Chebyshev-Davidson has in orthogonalization and refinement are outweighed by matrix-vector disadvantages. We notice that the relationship shown in [40]—that increases in the degree d lead to decreases in computational costs—is not realized here. In fact, one can see that the wall clock times for the $d := 80$ runs are slightly larger than those for $d := 40$. Per Amdahl's argument, when matrix-vector products are the principal and overwhelming compute cost, matrix-vector product counts will predict wall clock times.

5. Conclusion. We have shown that restarted Lanczos methods that use exact shifts may encounter difficulty due to stagnation of the discarded Ritz values from iteration to iteration. This problem is particularly acute when the dimension of the subspace is restricted; that is, when one does not set m greater than 2 times the number of wanted eigenvalues k, but sets it instead close to k. We have developed a method to break Ritz value stagnation and accelerate convergence for values of m close to k, but we have also seen a benefit for larger values of m as well. We have demonstrated that this approach requires fewer matrix-vector products to achieve convergence of eigenpairs than restarts using only exact shifts or Fast Leja points. One may expect our method to produce smaller wall clock times, especially when matrix-vector product costs dominate compute times; this may be the case when the matrix has a great many nonzero entries. This method introduces new parameters to be defined by the user—w and τ—but we have seen that the method is not sensitive to w and provided advice for how to choose τ in Section 4.1.2. The method works competitively even out-of-the-box with $w := 4$ and $\tau := 5 \times 10^{-6}$. The advantage over other implicit restart methods is most pronounced when memory is restricted.

Acknowledgements. The author would like to thank the referees, whose comments have made this paper immeasurably better.

REFERENCES

[1] *SLEPc for Python*. https://bitbucket.org/slepc/slepc4py.

ETNA
Kent State University
http://etna.math.kent.edu

[2] K. AISHIMA, *Global convergence of the restarted Lanczos and Jacobi-Davidson methods for symmetric eigenvalue problems*, Numer. Math., 131 (2015), pp. 405–423.
[3] R. ALBERT AND A.-L. BARABÁSI, *Statistical mechanics of complex networks*, Rev. Modern Phys., 74 (2002), pp. 47–97.
[4] J. BAGLAMA, D. CALVETTI, AND L. REICHEL, *Algorithm 827: IRBLEIGS: a MATLAB program for computing a few eigenvalues and eigenvectors of a Hermitian matrix*, ACM Trans. Math. Software, 29 (2003), pp. 337–348. http://www.math.uri.edu/~jbaglama/#Software.
[5] ———, *Fast Leja points*, Electron. Trans. Numer. Anal., 7 (1998), pp. 124–140. http://etna.ricam.oeaw.ac.at/vol.7.1998/pp124-140.dir/pp124-140.pdf
[6] ———, *IRBL: an implicitly restarted block-Lanczos method for large-scale Hermitian eigenproblems*, SIAM J. Sci. Comput., 24 (2003), pp. 1650–1677.
[7] J. BAGLAMA, D. CALVETTI, L. REICHEL, AND A. RUTTAN, *Computation of a few small eigenvalues of a large matrix with application to liquid crystal modeling*, J. Comput. Phys., 146 (1998), pp. 203–226.
[8] J. BAGLAMA AND L. REICHEL, *Augmented implicitly restarted Lanczos bidiagonalization methods*, SIAM J. Sci. Comput., 27 (2005), pp. 19–42
[9] ———, *An implicitly restarted block Lanczos bidiagonalization method using Leja shifts*, BIT, 53 (2013), pp. 285–310.
[10] Z. BAI, J. DEMMEL, J. DONGARRA, A. RUHE, AND H. VAN DER VORST, *Templates for the Solution of Algebraic Eigenvalue Problems*, SIAM, Philadelphia, 2000.
[11] A.-L. BARABÁSI, A. RÉKA, AND H. JEONG, *Mean-field theory for scale-free random networks*, Phys. A, 272 (1999), pp. 173–187.
[12] C. A. BEATTIE, M. EMBREE, AND D. C. SORENSEN, *Convergence of polynomial restart Krylov methods for eigenvalue computations*, SIAM Rev., 47 (2005), pp. 492–515.
[13] M. BELKIN AND P. NIYOGI, *Laplacian eigenmaps for dimensionality reduction and data representation*, Neural Comput., 15 (2003), pp. 1373–1396.
[14] R. F. BOISVERT, R. POZO, K. A. REMINGTON, R. F. BARRETT, AND J. DONGARRA, *Matrix market: a web resource for test matrix collections.*, in Quality of Numerical Software, R. F. Boisvert, ed., IFIP Advances in Information and Communication Technology 1997, Springer, New York, 1997, pp. 125–137.
[15] D. CALVETTI, L. REICHEL, AND D. C. SORENSEN, *An implicitly restarted Lanczos method for large symmetric eigenvalue problems*, Electron. Trans. Numer. Anal., 2 (1994), pp. 1–21. http://etna.ricam.oeaw.ac.at/vol.2.1994/pp1-21.dir/pp1-21.pdf
[16] T. A. DAVIS AND Y. HU, *The University of Florida sparse matrix collection*, ACM Trans. Math. Software, 38 (2011), Art. 1, (25 pages).
[17] J. W. DEMMEL, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, 1997.
[18] I. S. DHILLON, *A New $O(n^2)$ Algorithm for the Symmetric Tridiagonal Eigenvalue/Eigenvector Problem*, Ph.D. Thesis, Computer Science Division, University of California, Berkeley, 1997.
[19] J. W. EATON, *GNU Octave Manual*, Network Theory Ltd., UK, 2002.
[20] G. H. GOLUB, F. T. LUK, AND M. L. OVERTON, *A block Lánczos method for computing the singular values of corresponding singular vectors of a matrix*, ACM Trans. Math. Software, 7 (1981), pp. 149–169.
[21] Z. JIA, *Polynomial characterizations of the approximate eigenvectors by the refined Arnoldi method and an implicitly restarted refined Arnoldi algorithm*, Linear Algebra Appl., 287 (1999), pp. 191–214.
[22] Y. KOREN, *On spectral graph drawing*, in Computing and combinatorics, T. Warnow and B. Zhu, eds., vol. 2697 of Lecture Notes in Comput. Sci., Springer, Berlin, 2003, pp. 496–508.
[23] R. LEHOUCQ, D. SORENSEN, AND C. YANG, *ARPACK User's Guide: Solution of Large-Scale Eigenvalue Problems With Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, 1998.
[24] J. LESKOVEC AND R. SOSIČ, *SNAP: A general purpose network analysis and graph mining library in C++*, June, 2014. http://snap.stanford.edu/snap.
[25] M. E. J. NEWMAN, *Finding community structure in networks using the eigenvectors of matrices*, Phys. Rev. E, 74 (2006), 036104, (19 pages).
[26] B. N. PARLETT AND D. S. SCOTT, *The Lanczos algorithm with selective orthogonalization*, Math. Comp., 33 (1979), pp. 217–238.
[27] A. POTHEN, H. D. SIMON, AND K.-P. LIOU, *Partitioning sparse matrices with eigenvectors of graphs*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 430–452.
[28] C. POZRIKIDIS, *An Introduction to Grids, Graphs, and Networks*, Oxford University Press, Oxford, 2014.
[29] H. QIU AND E. HANCOCK, *Clustering and embedding using commute times*, IEEE Trans. Pattern Anal. Mach. Intell., 29 (2007), pp. 1873–1890.
[30] Y. SAAD, *On the rates of convergence of the Lanczos and the block-Lanczos methods*, SIAM J. Numer. Anal., 17 (1980), pp. 687–706.
[31] M. SAERENS, F. FOUSS, L. YEN, AND P. DUPONT, *The principal components analysis of a graph and its relationships to spectral clustering*, in Machine Learning: ECML 2004, J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, eds., vol. 3201 of Lecture Notes in Comput. Sci., Springer, Berlin, 2004, pp. 371–383.

ETNA
Kent State University
http://etna.math.kent.edu

[32] H. D. SIMON, *Analysis of the symmetric Lanczos algorithm with reorthogonalization methods*, Linear Algebra Appl., 61 (1984), pp. 101–131.
[33] ———, *The Lanczos algorithm with partial reorthogonalization*, Math. Comp., 42 (1984), pp. 115–142.
[34] G. W. STEWART, *A Krylov-Schur algorithm for large eigenproblems*, SIAM J. Matrix Anal. Appl., 23 (2002), pp. 601–614.
[35] R. C. WHALEY AND A. PETITET, *Minimizing development and maintenance costs in supporting persistently optimized BLAS*, Softw. Pract. Exper., 35 (2005), pp. 101–121.
[36] K. WU AND H. SIMON, *TRLan software package*, March, 1999. http://crd-legacy.lbl.gov/~kewu/trlan.html.
[37] ———, *Thick-restart Lanczos method for large symmetric eigenvalue problems*, SIAM J. Matrix Anal. Appl., 22 (2000), pp. 602–616.
[38] L. YEN, D. VANVYVE, F. WOUTERS, F. FOUSS, M. VERLEYSEN, AND M. SAERENS, *Clustering using a random walk based distance measure*, in Proceedings of the 13th Symposium on Artificial Neural Networks, ESANN Electronic Proceedings, 2005, pp. 317–324.
[39] Y. ZHOU, *A block Chebyshev-Davidson with inner-outer restart for large eigenvalue problems*, J. Comput. Phys., 229 (2010), pp. 9188–9200. http://faculty.smu.edu/yzhou/code.htm.
[40] Y. ZHOU AND Y. SAAD, *A Chebyshev-Davidson algorithm for large symmetric eigenproblems*, SIAM J. Matrix Anal. Appl., 29 (2007), pp. 954–971
[41] ———, *Block Krylov-Schur method for large symmetric eigenvalue problems*, Numer. Algorithms, 47 (2008), pp. 341–359.