HYBRID ITERATIVE REFINED METHOD FOR COMPUTING A FEW EXTREME EIGENPAIRS OF A SYMMETRIC MATRIX *

JAMES BAGLAMA†, TOM BELLA ‡, AND JENNIFER PICUCCI §

Abstract. We developed a hybrid restarted Lanczos method that combines thick–restarting with Ritz vectors with iteratively refined Ritz vectors to compute a few of the extreme eigenvalues and associated eigenvectors of a large sparse symmetric matrix A. The iterative refined Ritz vectors use a scheme, where we replace the approximate eigenvalue in the original refined scheme with the latest computed refined Ritz value until convergence. The thick–restarting schemes have shown to be superior to most other schemes, particularly restarted schemes of linear combinations. However, the simple thick–restarting Lanczos scheme is not available when using refined or iterative refined Ritz vectors. Instead, we use a hybrid restarted scheme that switches between thick–restarted with Ritz vectors and restarting with a judiciously chosen linear combination of iterative refined Ritz vectors. We provide some theoretical results and several computed examples.

Key words. restarted Lanczos method, eigenvalue computation, singular value, refined Ritz, thick–restarted.

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

1. Introduction.

Large sparse symmetric eigenvalue problems

$$Ax = \lambda x \qquad A \in \mathbb{R}^{n\times n} \tag{1.1}$$

are some of the most important and profoundly studied areas in numerical linear algebra. Although these problems are numerically attractive, they can exhibit computational challenges for even the best modern routines, e.g. clustering of eigenvalues, matrix sizes (memory constraints), and orthogonality. The importance of these problems and computational challenges have spurred a considerable amount of research, e.g.[1, 2, 6, 7, 13, 21, 22, 27, 28, 29] and references within. The goal of this paper is twofold: the development of a new, simple algorithm that uses as little storage as possible to compute a few of the extreme eigenpairs of a large sparse symmetric matrix A, and provide some insightful results on the (iterative) refined Ritz scheme. Since A is so large, we assume its factorization is not feasible and only the evaluation of matrix–vector products with the matrix A is available.

The motivation for the paper starts with the pioneering algorithm of Sorensen [21] for symmetric matrices called the Implicitly Restarted Lanczos (IRL) method (non–symmetric case is referred to as Implicitly Restarted Arnoldi (IRA) method). The IRL method is a restarted Krylov subspace method that implicitly modifies the starting vector p_1 with an accelerating polynomial. The accelerating polynomial is determined by a specific selection of zeros of the polynomial, also called shifts. There are several choices for shifts; the IRL method in [21] uses Ritz values as shifts. The choice of shifts is crucial for performance of the IRL method and there have been investigations into other choices, e.g. Leja points [1] and harmonic Ritz [15]. In 1997, Jia [6] introduced

*Submitted to the editors on June 10, 2020. Revision submitted on November 11, 2020. Accepted on January 27, 2021.

†Department of Mathematics, University of Rhode Island, Kingston, RI 02881. E-mail: jbaglama@uri.edu.

‡Department of Mathematics, University of Rhode Island, Kingston, RI 02881. E-mail: tombella@uri.edu.

§Department of Mathematics, University of Rhode Island, Kingston, RI 02881 E-mail: jenniferpicucci@uri.edu and U.S. Army Engineer Research and Development Center, Vicksburg, Mississippi. Email: Jennifer.r.picucci@usace.army.mil

the concept of refined Ritz vectors. The idea is for a given approximate eigenvalue μ of A, to minimize $\|Az - \mu z\|$ for a unit vector z from the current working Krylov subspace. Refined Ritz vectors often provide better eigenvector approximations than the Ritz vectors, see analysis [9, 10] for details. Since refined Ritz will be the basis of our new method, a thorough discussion is presented in section 4. In [7], Jia applied the refined concept in combination with the IRA procedure, referred to as Implicitly Restarted Refined Arnoldi method (IRRA). Morgan [14] showed the equivalence of the IRA method with Ritz values as shifts with augmenting the Krylov subspace by certain Ritz vectors. Wu and Simon [27] exploited this idea in the symmetric case, and by using the property that all of the Ritz vectors are multiples of same residual vector, they created a simple augmented method, called the "Thick–Restarted Lanczos method." The method is mathematically equivalent to the IRL method and avoids the need for the implicitly shifted QR algorithm. Similarly, Stewart [25] presents the Krylov–Schur method for the non–symmetric case and showed its equivalence to, and numerical superiority over, the IRA method.

The key feature needed with the thick–restarting Lanczos method with Ritz vectors is that the resulting space remains a Krylov subspace, which is possible since the Ritz vectors are all multiples of a single residual vector, [14]. The simple thick–restarting scheme by Wu and Simon [27] with refined Ritz vectors in place of Ritz vectors is not possible, because refined Ritz vectors are not all multiples of a single residual vector, see Theorem 4.3 in section 4. Furthermore, as discussed in [9, 10, 17] on the relationship between refined Ritz and Ritz vectors they are not parallel unless refined Ritz vectors equal eigenvectors. We present a similar result for the context here, see Theorems 4.2 and 5.2.

To improve approximations to desired eigenpairs and decrease the refined Ritz residuals norm, we introduced an iterative scheme, where we replace the approximate eigenvalue in the refined scheme with the latest computed refined Ritz value until convergence, section 5. This process has the added benefit of eliminating part of the refined Ritz residuals. The resulting residual vector has convenient qualities and a "smaller" norm. However, like refined Ritz vectors, the iterative refined Ritz vectors are not all multiples of a single residual vector, see section 5.

Morgan showed in [14] that the IRA method developed by Sorensen can be implemented by using a starting vector with a cleverly chosen linear combination of the desired Ritz vectors. We use a similar linear combination when restarting with the iterative refined Ritz vectors. The idea is to inherit similar, beneficial, restarting properties.

However, when using only refined Ritz or iterative refined Ritz vectors for restarting we observed examples of stagnation, erratic convergence, or very slow convergence. Slow convergence is exacerbated with restarted methods when using a low dimensional subspace and/or clustered eigenvalues, see Examples 5.2 and 5.3 in section 5. Thick–restarting with Ritz vectors also exhibits very slow convergence when using a low dimensional subspace. Therefore, we implemented a hybrid method section 6 that depends on certain criteria for switching from thick–restarting with Ritz vectors to restarting with a linear combination of iterative refined Ritz vectors. We observed through numerical experiments that although switching from thick–restarted Ritz to a linear combination of iterative refined results in a temporary undesirable spike in the norms of the residuals, this often relieves the stagnation/slow convergence, resulting in an overall faster convergence. This has the added benefit of being able to use a small Krylov subspace, see Figure 5.1.

Hybrid methods have been used previously for combining other forms of eigen-

vector approximations. For example, a comparable method is a block hybrid method that was proposed in [11] in which thick–restarting is performed by "modified" Ritz vectors, computed over a block Krylov subspace. Their block hybrid algorithm uses a power method with refined Ritz vectors "stitched" together with thick–restarting with modified Ritz vectors. Although absent with their code, we do provide an example in section 7 on the symmetric matrix experiment presented in their paper.

It should be noted that the Jacobi–Davidson method and extensions thereof are competitive for the computation of a few eigenvalues and associated eigenvectors, particularly when a known preconditioner is available; see [5, 20, 22, 23, 28] for descriptions of such methods. We do provide some comparisons in section 7.

Throughout this paper $\|\cdot\|$ denotes the Euclidean vector norm or the associated induced matrix norm. When useful and for ease of presentation we will utilize MATLAB notation.

The subsequent parts of the paper are organized as follows. We begin with a background of restarted Lanczos method section 2, thick–restarting with Ritz vectors in section 3, and refined Ritz pairs in section 4. Some theoretical results and relationships on refined Ritz pairs are included in section 4. In section 5, we describe our new strategy for iterative refined Ritz vectors and provide several theoretical results, motivation, and relationships. Our new hybrid method for computing the extreme eigenpairs is presented in section 6. Numerical examples are presented in section 7 and conclusions follow in section 8.

2. Lanczos Method. For this discussion, we describe a Lanczos method that is modeled after the algorithms presented in [16, 27]. Given a unit vector p_1, we define the Krylov subspace

$$\mathbb{K}_m(A, p_1) = \text{span}\{p_1, Ap_1, A^2p_1, \ldots, A^{m-1}p_1\}. \tag{2.1}$$

Application of m steps of the MLan(0) Algorithm 2.1 given below with the initial starting vector p_1 applied to the symmetric matrix A generates a sequence of m orthogonal vectors $p_j \in \mathbb{R}^n$ that form an orthonormal basis for the Krylov subspace (2.1). This algorithmic process forms the well–known Lanczos decomposition,

$$AP_m = P_m T_m + f e_m^T, \tag{2.2}$$

where the Lanczos vectors

$$P_m = [p_1, p_2, \ldots, p_m] \in \mathbb{R}^{n\times m} \tag{2.3}$$

satisfy $P_m^T P_m = I_m$, and $f \in \mathbb{R}^n$, referred to as the residual vector, satisfies $P_m^T f = 0$. The matrix I_m denotes the identity matrix of order m and the vector $e_m \in \mathbb{R}^m$ consists of the last column of I_m. The matrix T_m is tridiagonal of the form

$$T_m = \begin{bmatrix} \alpha_1 & \beta_1 & & & 0 \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & & \ddots & \\ & & \ddots & \ddots & \beta_{m-1} \\ 0 & & & \beta_{m-1} & \alpha_m \end{bmatrix} \in \mathbb{R}^{m\times m}. \tag{2.4}$$

For brevity we give a more general version of MLan(ℓ) that will be used in section 3.

Algorithm 2.1 MLan(ℓ)

1: **Input:** $A \in \mathbb{R}^{n\times n}$: symmetric matrix,
$p_1, \ldots, p_{\ell+1} \in \mathbb{R}^{n\times(\ell+1)}$: orthonormal vector(s) (3.2),
$\bar{T}_{\ell+1} \in \mathbb{R}^{\ell+1\times\ell+1}$: if $\ell > 0$ input matrix (3.5),
m : maximum number of Lanczos vectors.
2: **Output:** $P_m = [p_1, p_2, \ldots, p_m] \in \mathbb{R}^{n\times m}$: orthogonal matrix (2.3) or (3.7),
$T_m \in \mathbb{R}^{m\times m}$: if $\ell = 0$ tridiagonal matrix (2.4), if $\ell > 0$ matrix (3.8),
$f \in \mathbb{R}^n$: residual vector.
3: **for** $j = \ell + 1 : m$ **do**
4: $f := Ap_j$
5: **if** $j > \ell + 1$ **then** $f := f - p_{j-1}\beta_j$
6: **if** $j = \ell + 1$ and $\ell > 0$ **then** $f := f - P_\ell \bar{T}(\ell+1, 1:\ell)^T$
7: $\alpha_j := f^T p_j$ and $f := f - p_j\alpha_j$
8: Reorthogonalization: $f := f - P_j(P_j^T f)$
9: **if** $j < m$ **then** $\beta_j := \|f\|$ and $p_{j+1} := f/\beta_j$

For our discussion, we will assume that we can perform $m - \ell$ steps of Algorithm 2.1 to generate the Lanczos tridiagonal decomposition (2.2) or, later, (3.6). In practice, near-breakdowns can occur, i.e. $\beta_j \approx 0$ for some j (step 9 Algorithm 2.1), see strategies such as those described in [2, 12] to continue the Lanczos process. Since m in our discussion is of a modest size and the desired number of eigenpairs k is small, a near-breakdown without convergence is rare and therefore, we will assume for the following discussion, T_m is unreduced. The reorthogonalization step introduced in step 8 of Algorithm 2.1 is a simple strategy to help maintain orthonormality among independent vectors for modest values $m \ll n$. More robust reorthogonalization strategies, e.g. selective or partial are described in the literature, see e.g. [16, 19, 27].

Let $\{\theta_j, y_j\}_{j=1}^m$ be the m eigenpairs of the tridiagonal matrix T_m with the desired k eigenpairs appearing as the leading entries, i.e. $\{\theta_j, y_j\}_{j=1}^k$. Define $x_j := P_m y_j$. Then θ_j and x_j are commonly referred to as a Ritz value and a Ritz vector of A, respectively, or simply a Ritz pair. It follows from (2.2) that

$$Ax_j - \theta_j x_j = AP_m y_j - \theta_j P_m y_j = (P_m T_m - \theta_j P_m)\, y_j + f e_m^T y_j = f e_m^T y_j, \tag{2.5}$$

and therefore the norms of the residual errors for the Ritz pairs $\{\theta_j, x_j\}$ satisfy

$$\|Ax_j - \theta_j x_j\| = \beta_m |e_m^T y_j| \tag{2.6}$$

where $\beta_m = \|f\|$. For numerical experiments, we use a stopping criterion

$$\beta_m |e_m^T y_j| \le \epsilon \|A\| \tag{2.7}$$

where ϵ is a user specified tolerance and $\|A\|$ is approximated by the eigenvalue of largest magnitude over all iterations thus far of the computed matrices T_m. For a given number k of desired eigenpairs we have convergence when the norm (2.6) for all $j = 1, \ldots, k$ is less than $\epsilon\|A\|$.

3. Thick–Restarted Lanczos with Ritz vectors. We next briefly describe the method of thick–restarting with Ritz vectors as outlined in [27] which is needed in subsequent sections. For a thorough description, we refer the reader to [24, 25, 27, 28] and the references within.

From (2.5), we have for $j = 1, \ldots, k$

$$Ax_j = \theta_j x_j + p_{m+1}\bar{\beta}_j \tag{3.1}$$

where $p_{m+1} = f/\beta_m$ and $\bar{\beta}_j = \beta_m e_m^T y_j$. Define

$$\bar{P}_k := [x_1, \ldots, x_k] \quad \text{and} \quad \bar{P}_{k+1} := [\bar{P}_k, p_{m+1}]. \tag{3.2}$$

Then we have

$$A\bar{P}_k = \bar{P}_{k+1}\bar{T}_{k+1,k} \tag{3.3}$$

where

$$\bar{T}_{k+1,k} = \begin{bmatrix} \theta_1 & & & 0 \\ & \theta_2 & & \\ & & \ddots & \\ 0 & & & \theta_k \\ \bar{\beta}_1 & \bar{\beta}_2 & \ldots & \bar{\beta}_k \end{bmatrix}. \tag{3.4}$$

The factorization (3.3) can easily be extended to m vectors via MLan(k) Algorithm 2.1 by noticing that

$$\bar{P}_{k+1}^T A \bar{P}_{k+1} = \bar{T}_{k+1} = \begin{bmatrix} \theta_1 & & & 0 & \bar{\beta}_1 \\ & \theta_2 & & & \bar{\beta}_2 \\ & & \ddots & & \vdots \\ 0 & & & \theta_k & \bar{\beta}_k \\ \bar{\beta}_1 & \bar{\beta}_2 & \ldots & \bar{\beta}_k & \alpha_{k+1} \end{bmatrix} \tag{3.5}$$

where the last diagonal element α_{k+1} is computed in step 7 in the MLan(k) Algorithm 2.1. Step 6 in MLan(k) Algorithm 2.1 is used to ensure the orthogonalization of the newly computed Lanczos vector against $\bar{P}_k$ (3.2), c.f. Algorithm 3 in [27]. After $m-k$ steps of MLan(k) Algorithm 2.1 we have,

$$A\bar{P}_m = \bar{P}_m \bar{T}_m + \bar{f} e_m^T, \tag{3.6}$$

where

$$\bar{P}_m = [\bar{P}_{k+1}, p_{k+2}, \ldots, p_m] \in \mathbb{R}^{n\times m}, \tag{3.7}$$

satisfies $\bar{P}_m^T \bar{P}_m = I_m$, and $\bar{f} \in \mathbb{R}^n$ satisfies $\bar{P}_m^T \bar{f} = 0$ and

$$\bar{T}_m = \begin{bmatrix} \bar{T}_{k+1} & \beta_{k+1} & & & 0 \\ \beta_{k+1} & \alpha_{k+2} & \beta_{k+2} & & \\ & & & \ddots & \\ & & \ddots & \ddots & \beta_{m-1} \\ 0 & & & \beta_{m-1} & \alpha_m \end{bmatrix} \in \mathbb{R}^{m\times m}. \tag{3.8}$$

Algorithm 3.1 Thick–Restarted Lanczos

1: **Input:** $A \in \mathbb{R}^{n\times n}$: symmetric matrix,
$p_1 \in \mathbb{R}^n$: orthonormal vector,
k : number of desired eigenpairs of A,
m : maximum number of Lanczos vectors.
2: **Output:** (3.6) or $\{\theta_j, x_j\}_{j=1}^k$ approximate eigenpairs of A.
3: Compute factorization (2.2) by MLan(0) Algorithm 2.1
4: Compute the k desired eigenpairs $\{\theta_j, y_j\}_{j=1}^k$ of T_m (2.4) or $\bar{T}_m$ (3.8)
5: Check convergence (2.6)
6: Set up (3.3) and (3.5) and call MLan(k) Algorithm 2.1 to get factorization (3.6)
7: Goto 4

4. Refined Ritz vectors. We briefly describe and provide some results on refined Ritz vectors and values. For a thorough description, we refer the reader to [6, 7, 9, 10]. The refined Ritz vector z_j for an approximate desired eigenvalue μ_j of A is computed by

$$\min_{\substack{z_j\in\mathbb{K}_j(A,p_1)\\ \|z_j\|=1}} \|Az_j - \mu_j z_j\|. \tag{4.1}$$

Since $z_j \in \mathbb{K}_j(A, p_1)$, we can represent it in terms of the Lanczos vectors (2.3), i.e. $z_j = P_m w_j$ for some unit vector w_j. Given $\beta_m = \|f\|$, define

$$P_{m+1} := [P_m, p_{m+1}] \in \mathbb{R}^{n\times(m+1)}, \quad T_{m+1,m} := \begin{bmatrix} T_m \\ \beta_m e_m^T \end{bmatrix} \in \mathbb{R}^{(m+1)\times m}, \tag{4.2}$$

where $p_{m+1} := f/\beta_m$. Let $I_{m+1,m}$ denote the first m columns of the identity matrix of order $m+1$. For each μ_j compute the smallest singular value $\sigma_{\downarrow j}$ and associated unit singular vectors of $(T_{m+1,m} - \mu_j I_{m+1,m})$ i.e.

$$(T_{m+1,m} - \mu_j I_{m+1,m})v_j = \sigma_{\downarrow j} u_j \tag{4.3}$$

$$(T_{m+1,m} - \mu_j I_{m+1,m})^T u_j = \sigma_{\downarrow j} v_j. \tag{4.4}$$

We refer to the following unit vectors $v_j \in \mathbb{R}^m$ as the right singular vector and $u_j \in \mathbb{R}^{m+1}$ as the left singular vector associated with the smallest singular value $\sigma_{\downarrow j}$. Therefore,

$$\begin{aligned} \min_{\substack{z_j\in\mathbb{K}_j(A,p_1)\\ \|z_j\|=1}} \|Az_j - \mu_j z_j\| &= \min_{\|w_j\|=1} \|AP_m w_j - \mu_j P_m w_j\| \\ &= \min_{\|w_j\|=1} \|(P_{m+1}(T_{m+1,m} - \mu_j I_{m+1,m})w_j\| \\ &= \|(T_{m+1,m} - \mu_j I_{m+1,m})v_j\| = \sigma_{\downarrow j}. \end{aligned} \tag{4.5}$$

Then the refined Ritz vector z_j for μ_j is defined as $z_j := P_m v_j$. The approximate eigenvalue can be selected as the Ritz value θ_j or as the Rayleigh quotient $\rho_j = z_j^T A z_j = v_j^T T_m v_j$. Jia [7] suggested using ρ_j in place of θ_j as it may be more accurate. Setting $\mu_j = \theta_j$ and using (2.6), we have from [7, 9] that

$$\|Az_j - \rho_j z_j\| \le \|Az_j - \theta_j z_j\| \le \|Ax_j - \theta_j x_j\| = \beta_m |e_m^T y_j|. \tag{4.6}$$

If $\|Ax_j - \theta_j x_j\| \neq 0$, then $\|Az_j - \theta_j z_j\| < \|Ax_j - \theta_j x_j\|$. The following discussion establishes the left inequality for μ_j not necessarily $\mu_j = \theta_j$ and shows when a strict inequality exists. This sets the foundation for the subsequent section on iterative refined Ritz.

Using $\{\rho_j, z_j\}$ as the approximation, we have the following. Equate the first m rows of (4.3) and the last row to obtain

$$(T_m - \mu_j I_m)v_j = \sigma_{\downarrow j} u_{j(1:m)} \tag{4.7}$$

$$\beta_m e_m^T v_j = \sigma_{\downarrow j} u_{j(m+1)} \tag{4.8}$$

and left multiply (4.7) by v_j^T to obtain

$$\mu_j - \rho_j = -\sigma_{\downarrow j} v_j^T u_{j(1:m)}. \tag{4.9}$$

Then using the relationships (4.7)–(4.9) we have,

$$\begin{aligned} Az_j = AP_m v_j &= P_m T_m v_j + f e_m^T v_j \\ &= \rho_j z_j + \sigma_{\downarrow j} P_{m+1}(u_j - ([v_j; 0]^T u_j)[v_j; 0]) \end{aligned} \tag{4.10}$$

where $[v_j; 0] \in \mathbb{R}^{m+1}$. Using (4.10) we have,

$$\begin{aligned} \|Az_j - \rho_j z_j\| &= \sigma_{\downarrow j} \|P_{m+1}(u_j - ([v_j; 0]^T u_j)[v_j; 0])\| \\ &= \sigma_{\downarrow j} \|u_j - ([v_j; 0]^T u_j)[v_j; 0]\| \\ &\leq \sigma_{\downarrow j}. \end{aligned} \tag{4.11}$$

The inequality in (4.11) comes from using the orthogonal projection, as summarized in Lemma A.1 in the appendix. A strict inequality $\|Az_j - \rho_j z_j\| < \sigma_{\downarrow j}$ exists if $\mu_j \neq \rho_j$ then $[v_j; 0]^T u_j \neq 0$ in (4.9), also see Lemma A.1.

Notice from (4.9) and (4.10) we have

$$\begin{aligned} Az_j = AP_m v_j &= (\rho_j - \sigma_{\downarrow j} v_j^T u_{j(1:m)}) z_j + \sigma_{\downarrow j} P_{m+1} u_j \\ &= \mu_j z_j + \sigma_{\downarrow j} P_{m+1} u_j \end{aligned} \tag{4.12}$$

and $\|Az_j - \mu_j z_j\| = \sigma_{\downarrow j} \|P_{m+1} u_j\| = \sigma_{\downarrow j}$. Therefore, we have

$$\|Az_j - \rho_j z_j\| \leq \|Az_j - \mu_j z_j\| \tag{4.13}$$

and if $\mu_j \neq \rho_j$ then $\|Az_j - \rho_j z_j\| < \|Az_j - \mu_j z_j\|$. The relationship and convergence properties of the refined Ritz vector z_j and Ritz vector x_j are described in detail in [9, 10, 17]. In particular, they show, when $\sigma_{\downarrow j} = 0$, the vectors z_j and x_j are parallel to an eigenvector of A and $\rho_j = \theta_j$ is an exact eigenvalue of A. It is remarked in [17] that for a special, non–symmetric case, that z_j and x_j can be parallel, even though $\sigma_{\downarrow j} \neq 0$. However, this is not the case in the context here for the symmetric problem, as Theorem 4.2 below demonstrates. First, Corollary 4.1 is established.

COROLLARY 4.1. *Let T_m (2.4) be unreduced and $\beta_m \neq 0$. Given the singular value relationships (4.3) and (4.4) we have*
i.) $\sigma_{\downarrow j} \neq 0$ and
ii.) $u_j \neq \pm e_{m+1}$.

Proof. i.) To show that $\sigma_{\downarrow j} \neq 0$, we argue via contradiction. Let $\sigma_{\downarrow j} = 0$, then from (4.7) and (4.8) we have,

$$(T_m - \mu_j I_m)v_j = 0 \tag{4.14}$$

$$\beta_m e_m^T v_j = 0. \tag{4.15}$$

Given $\beta_m \neq 0$, we have from (4.15) $e_m^T v_j = 0$ and since T_m is unreduced, $e_m^T v_j \neq 0$, c.f. [16, Theorem 7.9.3].
ii.) To show that $u_j \neq \pm e_{m+1}$, we also argue via contradiction. Let $u_j = \pm e_{m+1}$. Then from (4.3) and (4.4) we have,

$$(T_m - \mu_j I_m) v_j = 0 \tag{4.16}$$

$$\pm \beta_m e_m = \sigma_{\downarrow j} v_j. \tag{4.17}$$

Given $\beta_m \neq 0$, we have from (4.17) that $v_j = \pm e_m$. Then from (4.16), we have $\beta_{m-1} = 0$. Contradiction to T_m being unreduced. ∎

THEOREM 4.2. *Let T_m (2.4) be unreduced and $\beta_m \neq 0$. Given the singular value relationships (4.3) and (4.4) with $\mu_j = \theta_j$. Then $z_j \neq \pm x_j$.*

Proof. To show that $z_j \neq \pm x_j$, we argue via contradiction. Let $x_j = \pm z_j$ then $y_j = \pm v_j$. From (4.7) and Corollary 4.1 we have

$$0 = (T_m - \theta_j I_m) y_j = \pm (T_m - \theta_j I_m) v_j = \pm \sigma_{\downarrow j} u_{j(1:m)} \neq 0. \tag{4.18}$$

∎

Therefore, the refined Ritz and Ritz vectors do not coincide until $\sigma_{\downarrow j} = 0$.

Define

$$r_j := u_j - ([v_j; 0]^T u_j)[v_j; 0] \tag{4.19}$$

and notice that the refined Ritz vector and residual vectors satisfy $z_j^T (P_{m+1} r_j) = 0$. This is a requirement for restarting. Then from (4.10) we have similar to (3.1) for $j = 1, \ldots, k$,

$$A z_j = \rho_j z_j + \sigma_{\downarrow j} P_{m+1} r_j. \tag{4.20}$$

However, the setup of the thick–restarted method as described in section 3 cannot be used with $\{\rho_j, z_j\}$ in this context, since the residual vectors $\sigma_{\downarrow j} P_{m+1} r_j$ are not multiples of each other for different refined Ritz pairs $\{\rho_j, z_j\}$. This is shown in Theorem 4.3. Below is a needed relationship before establishing the result. Notice from (4.3), (4.9), and (4.19) that

$$\begin{bmatrix} T_m - \rho_j I_m \\ \beta_m e_m^T \end{bmatrix} v_j = \sigma_{\downarrow j} r_j \tag{4.21}$$

and if $\{\rho_j, v_j\}$ is not an eigenpair of T_m then $\sigma_{\downarrow j}(u_{j(1:m)} - (v_j^T u_{j(1:m)}) v_j) \neq 0$.

THEOREM 4.3. *Given refined Ritz pairs $\{\rho_{j_1}, z_{j_1}\}$ and $\{\rho_{j_2}, z_{j_2}\}$ with $\rho_{j_1} \neq \rho_{j_2}$ satisfying (4.20) and $\{\rho_{j_1}, v_{j_1}\}$ and $\{\rho_{j_2}, v_{j_2}\}$ not eigenpairs of T_m, we have $\sigma_{\downarrow j_1} P_{m+1} r_{j_1} \neq \gamma \sigma_{\downarrow j_2} P_{m+1} r_{j_2}$ for any scalar γ.*

Proof. To show that $\sigma_{\downarrow j_1} P_{m+1} r_{j_1} \neq \gamma \sigma_{\downarrow j_2} P_{m+1} r_{j_2}$, we argue via contradiction. Let $\sigma_{\downarrow j_1} P_{m+1} r_{j_1} = \gamma \sigma_{\downarrow j_2} P_{m+1} r_{j_2}$ where $\gamma \neq 0$. Then from multiplying (4.20) with $j = j_2$ from the left by $z_{j_1}^T$ and then by $z_{j_2}^T$ when $j = j_1$ gives the relationships $v_{j_1}^T T_m v_{j_2} - \rho_{j_2} v_{j_1}^T v_{j_2} = 0$ and $v_{j_2}^T T_m v_{j_1} - \rho_{j_1} v_{j_2}^T v_{j_1} = 0$. Since $\rho_{j_1} \neq \rho_{j_2}$ this implies $v_{j_2}^T T_m v_{j_1} = 0$ and $v_{j_2}^T v_{j_1} = 0$. From (4.21) we have $(T_m - \rho_{j_1} I_m) v_{j_1} = \gamma (T_m - \rho_{j_2} I_m) v_{j_2} \neq 0$ and $\beta_m e_m^T v_{j_1} = \gamma \beta_m e_m^T v_{j_2}$. Therefore,

$$\begin{aligned} 0 < v_{j_1}^T (T_m - \rho_{j_1} I_m)(T_m - \rho_{j_1} I_m) v_{j_1} &= \gamma v_{j_1}^T (T_m - \rho_{j_1} I_m)(T_m - \rho_{j_2} I_m) v_{j_2} \\ &= \gamma v_{j_1}^T T_m^2 v_{j_2}. \end{aligned}$$

For v_{j_1} from (4.3) and (4.4) we have $((T_m-\mu_{j_1}I_m)(T_m-\mu_{j_1}I_m)+\beta_m^2 e_m e_m^T)v_{j_1} = \sigma_{\downarrow j_1}^2 v_{j_1}$ and after left multiplying by $v_{j_2}^T$ we get $\gamma v_{j_1}^T T_m^2 v_{j_2} = -\beta_m^2\gamma^2(e_m^T v_{j_2})^2 < 0$. ∎

A restarted technique such as setting the starting vector p_1 in Algorithm 2.1 as a linear combination of k desired approximate refined Ritz vectors can be used, see [6]. We propose a different linear combination with constants chosen in a similar fashion outlined in [14]. Before presenting our restarted scheme, we will first focus on adjusting the refined Ritz vectors to reduce the residual norm. We also include some results and motivational remarks.

5. Iterative Refined Ritz vectors. Considering the refined Ritz pair $\{\rho_j, z_j\}$ may provide a better approximation by having a "smaller" norm (4.6) when used together, we propose iteratively refining the approximation. That is, set $\mu_j = \rho_j$ in (4.1) and re–compute the refined Ritz vectors as stated in (4.5) with the updated μ_j. This process can be repeated and creates an iterative scheme with a sequence of refined Ritz pairs, $\{\rho_j^{(i)}, z_j^{(i)}\}$ for $i = 1, 2, \ldots$. The process terminates with a refined Ritz pair $\{\hat{\rho}_j, \hat{z}_j\}$ that has favorable properties. We will refer to $\{\hat{\rho}_j, \hat{z}_j\}$, as the iterative refined Ritz value and vector respectively, and collectively as the iterative refined Ritz pair. Algorithm 5.1 outlines the computational process which we follow up with remarks. To show convergence, and hence termination of the iterative scheme, we establish via Theorem 5.1 that the nonnegative sequence $\sigma_{\downarrow j}^{(i)}$ computed from this process is bounded, decreasing, and hence converges. Let $\hat{\sigma}_{\downarrow j}$ be the value to which the sequence $\sigma_{\downarrow j}^{(i)}$ converges.

THEOREM 5.1. *Let $\mu_j = \rho_j^{(i-1)}$ with $\rho_j^{(0)} = \theta_j$, $z_j^{(i)} = P_m v_j^{(i)}$, and $\rho_j^{(i)} = z_j^{(i)^T} A z_j^{(i)}$ for $i = 1, 2, \ldots$. Then the computed smallest singular values $\sigma_{\downarrow j}^{(i)}$ for $i = 1, 2, \ldots$ from the equations (4.3) and (4.4) to solve (4.1) is a nonnegative bounded decreasing sequence and hence converges.*

Proof. We compute

$$
\begin{aligned}
0 \le \sigma_{\downarrow j}^{(i+1)} = \sigma_{\downarrow j}^{(i+1)} \|u_j^{(i+1)}\| &= \|\sigma_{\downarrow j}^{(i+1)} u_j^{(i+1)}\| \\
&= \|(T_{m+1,m} - \rho_j^{(i)} I_{m+1,m}) v_j^{(i+1)}\| \\
&\le \|(T_{m+1,m} - \rho_j^{(i)} I_{m+1,m}) v_j^{(i)}\|.
\end{aligned} \tag{5.1}
$$

We have the inequality in (5.1) since $v_j^{(i+1)}$ satisfies the minimization property in (4.5) when $\mu_j = \rho_j^{(i)}$. Note that $v_j^{(i)}$ satisfies the minimization property in (4.5) when $\mu_j = \rho_j^{(i-1)}$, therefore we have

$$
\begin{aligned}
(T_{m+1,m} - \rho_j^{(i-1)} I_{m+1,m}) v_j^{(i)} &= \sigma_{\downarrow j}^{(i)}\ u_j^{(i)} \\
T_{m+1,m} v_j^{(i)} &= \sigma_{\downarrow j}^{(i)}\ u_j^{(i)} + \rho_j^{(i-1)} I_{m+1,m} v_j^{(i)}.
\end{aligned} \tag{5.2}
$$

From (4.9) we have

$$
\rho_j^{(i)} - \rho_j^{(i-1)} = \sigma_{\downarrow j}^{(i)}\ v_j^{(i)^T} u_{j(1:m)^{(i)}}. \tag{5.3}
$$

Plugging (5.2) and (5.3) into (5.1) and continuing (5.1) we have

$$
\begin{aligned}
0 \leq \sigma_{\downarrow_j}^{(i+1)} \leq & \|T_{m+1,m} v_j^{(i)} - \rho_j^{(i)} I_{m+1,m} v_j^{(i)}\| \\
= & \sigma_{\downarrow_j}^{(i)} \ \|u_j^{(i)} - ([v_j;0]^{(i)^T} u_j^{(i)})[v_j;0]^{(i)}\| \\
\leq & \sigma_{\downarrow_j}^{(i)} \ .
\end{aligned} \tag{5.4}
$$

The last inequality in (5.4) comes from using Lemma A.1. A strict inequality exists if $([v_j;0]^{(i)^T} u_j^{(i)}) \neq 0$, see Lemma A.1. ∎

Notice from (4.11) and (5.4) we have

$$
\begin{aligned}
\|A z_j^{(i+1)} - \rho_j^{(i+1)} z_j^{(i+1)}\| \leq & \ \sigma_{\downarrow_j}^{(i+1)} \\
\leq & \ \sigma_{\downarrow_j}^{(i)} \ \|u_j^{(i)} - ([v_j;0]^{(i)^T} u_j^{(i)})[v_j;0]^{(i)}\| \\
= & \ \|A z_j^{(i)} - \rho_j^{(i)} z_j^{(i)}\|
\end{aligned} \tag{5.5}
$$

which implies $\{\rho_j^{(i+1)}, z_j^{(i+1)}\}$ can be a better approximation than $\{\rho_j^{(i)}, z_j^{(i)}\}$. If $\sigma_{\downarrow_j}^{(i)} = 0$ for some i we have from (5.5) that $\{\rho_j^{(i)}, z_j^{(i)}\}$ is an exact eigenpair of A. We assume for the remainder of the section that $0 < \hat{\sigma}_{\downarrow_j} \leq \sigma_{\downarrow_j}^{(i)}$.

Another point of view of the equations (4.3) and (4.4) is as an eigenvalue problem. In this context, define $H(\mu_j) := (T_m - \mu_j I_m)(T_m - \mu_j I_m) + \beta_m^2 e_m e_m^T$ where $\mu_j = \rho_j = v_j^T T_m v_j$ is a function of the unit vector v_j. Therefore, we have

$$
H(v_j) v_j = \sigma_{\downarrow_j}^2 v_j \tag{5.6}
$$

and we are searching for the smallest eigenvalue $\sigma_{\downarrow_j}^2$ and associated unit eigenvector v_j. The vector u_j can be obtained by

$$
u_j = 1/\sigma_{\downarrow_j} (T_{m+1,m} - \mu_j I_{m+1,m}) v_j. \tag{5.7}
$$

Equation (5.6) is referred to as an eigenvector–dependent nonlinear eigenvalue problem (NEPv) and the most commonly used routine for solving (5.6) is the simple self-consistent field (SCF) iteration process, see [3] and reference within.

The computation of iterative refined values is outlined in the Iterative Refined Algorithm 5.1. We assume $m \ll n$ and the computational time required per iteration for Algorithm 5.1 is negligible in comparison to the computational time required for a matrix–vector product with A when n is very large. There are several options for

Algorithm 5.1 Iterative Refined

1: **Input:** $T_{m+1,m} \in \mathbb{R}^{m+1 \times m}$ (4.2) and $\{\mu_j\}_{j=1}^k$
2: **Output:** $\{\hat{\rho}_j, \hat{v}_j, \hat{u}_j, \hat{\sigma}_{\downarrow_j}\}_{j=1}^k$
3: **for** $j = 1, 2, \ldots, k$ **do**
4: **for** $i = 1, 2, \ldots, maxit$ **do**
5: Compute $v_j^{(i)}, u_j^{(i)}$, and $\sigma_{\downarrow_j}^{(i)}$ using either (4.3) and (4.4) or (5.6) and (5.7)
6: Set $\rho_j^{(i)} := v_j^{(i)^T} T_m v_j^{(i)}$
7: Check convergence
8: Set $\mu_j := \rho_j^{(i)}$

step 7 in Algorithm 5.1 on checking convergence. For example, convergence can be checked by $|\rho_j^{(i)} - \rho_j^{(i-1)}|/|\rho_j^{(i)}| < eps$, where eps is machine epsilon. Some heuristics for stopping are provided in [3] and references within when using the SCF iteration to solve NEPv (5.6). It should be noted that stagnation can occur while using finite arithmetic and we propose including a check to exit when detected to avoid unnecessary iterations. As the restarted hybrid method presented in section 6 converges we notice via numerical experiments the number of iterations in Algorithm 5.1 reduce quickly to only a handful.

We see from Theorem 5.1 and the relationship $\sigma_{\downarrow j}^2 = v_j^T H(v_j) v_j$ that the output $\{\hat{\rho}_j, \hat{v}_j, \hat{u}_j, \hat{\sigma}_{\downarrow j}\}$ from Algorithm 5.1 satisfies

$$(T_{m+1,m} - \hat{\rho}_j I_{m+1,m})\hat{v}_j = \hat{\sigma}_{\downarrow j}\hat{u}_j \tag{5.8}$$

$$(T_{m+1,m} - \hat{\rho}_j I_{m+1,m})^T \hat{u}_j = \hat{\sigma}_{\downarrow j}\hat{v}_j \tag{5.9}$$

where $\hat{\rho}_j = \hat{v}_j^T T_m \hat{v}_j$. Equate the first m rows of (5.8) and left multiply by $\hat{v}_j^T$ to get

$$\hat{\sigma}_{\downarrow j}\hat{v}_j^T \hat{u}_{j(1:m)} = 0. \tag{5.10}$$

Using (4.19), (4.20), and (5.10) we have

$$A\hat{z}_j = \hat{\rho}_j \hat{z}_j + \hat{\sigma}_{\downarrow j} P_{m+1}\hat{u}_j, \tag{5.11}$$

where $\hat{z}_j = P_m \hat{v}_j$ and $\hat{z}_j^T P_{m+1}\hat{u}_j = 0$. Notice from (4.6), (5.5), and (5.11)

$$\hat{\sigma}_{\downarrow j} = \|A\hat{z}_j - \hat{\rho}_j \hat{z}_j\| \leq \|Az_j - \rho_j z_j\| \leq \|Az_j - \theta_j z_j\| \leq \|Ax_j - \theta_j x_j\|. \tag{5.12}$$

Notice that in floating point arithmetic we have $\hat{\sigma}_{\downarrow j}\hat{v}_j^T \hat{u}_{j(1:m)} \approx 0$ and should be included in equation (5.11) when used in computer codes, c.f. (4.20). It is not included in establishing subsequent results and equations, i.e. we assume (5.10) holds.

Although the results listed in [9, 10, 17] were developed for refined Ritz and Ritz pairs when $\mu_j = \theta_j$, we see from the relationships (5.5) and (5.12) that similar results apply for iterative refined Ritz. In particular, we have when $\hat{\sigma}_{\downarrow j} = 0$, vectors $\hat{z}_j$ and x_j are parallel to an eigenvector of A and $\hat{\rho}_j = \theta_j$ is an exact eigenvalue of A. Notice that Corollary 4.1 does not depend on $\mu_j = \theta_j$, however Theorem 4.2 depended on setting $\mu_j = \theta_j$, therefore we state the result in this context for $\hat{z}_j$.

THEOREM 5.2. *Let T_m (2.4) be unreduced and $\beta_m \neq 0$. Given the singular value relationships (5.8) and (5.9), then $\hat{z}_j \neq \pm x_j$.*

Proof. To show that $\hat{z}_j \neq \pm x_j$, we argue via contradiction. Let $x_j = \pm\hat{z}_j$ then $y_j = \pm\hat{v}_j$ and $\hat{\rho}_j = \hat{v}_j^T T_m \hat{v}_j = y_j^T T_m y_j = \theta_j$. From (5.8) and Corollary 4.1 we have

$$0 = (T_m - \theta_j I_m)y_j = \pm(T_m - \hat{\rho}_j I_m)\hat{v}_j = \pm\hat{\sigma}_{\downarrow j}\hat{u}_{j(1:m)} \neq 0 \tag{5.13}$$

∎

Therefore, the iterative refined Ritz and Ritz vectors do not coincide unless $\hat{\sigma}_{\downarrow j} = 0$. Theorem 4.3 also does not depend on $\mu_j = \theta_j$ and the result holds in this context. That is, the set–up of the thick–restarted method as described in section 3 cannot be used with $\{\hat{\rho}_j, \hat{z}_j\}$ in this context, since the residual vectors $\sigma_{\downarrow j} P_{m+1}\hat{r}_j$ are not multiples of each other for different iterative refined pairs $\{\hat{\rho}_j, \hat{z}_j\}$. The iterative refined Ritz pair have a "smaller" residual norm and possess similar properties to the refined Ritz pair. The following examples provide motivation on our restarting technique.

Example 5.1 Let A be the 4×4 symmetric matrix,

$$A = \begin{bmatrix} 9 & 1 & -2 & 1 \\ 1 & 8 & -3 & -2 \\ -2 & -3 & 7 & -1 \\ 1 & -2 & -1 & 6 \end{bmatrix}. \tag{5.14}$$

The eigenvalues of A are 12, 9, 6, 3. Using MLan(0) Algorithm 2.1 with $p_1 = \frac{1}{2}[1\ 1\ 1\ 1]^T$ and $m = 3$, gives a tridiagonal matrix T_3 with eigenvalues $\theta_1 = 11.7913$, $\theta_2 = 7.4755, \theta_3 = 3.0239$, and $\beta_3 = 1.8035$. We have for the largest eigenpair,

$$\hat{\sigma}_{\downarrow 1} = 0.831397 < \sigma_{\downarrow 1}\|r_1\| = 0.831400 < \beta_3|e_3^T y_1| = 0.885392 \tag{5.15}$$

where $\mu_1 = \theta_1$ in Algorithm 5.1. Equation (5.15) shows the residual norm with iterative refined pair is "smaller", with similar results for the other eigenpairs. In practice, the matrix A is very large and restarting is required. A restarted Lanczos method depends on many things for successful complementation, one of which is a "good" (re)starting vector. For a fair comparison, Table 5.1 displays the Ritz residual norms $\beta_3|e_3^T y_1|$ associated with $\{\theta_1, y_1\}$ for A where we set the (re)starting vector p_1 in MLan(0) Algorithm 2.1 on the next restart to be the computed iterative refined Ritz $P_3\hat{v}_1$, refined Ritz $P_3 v_1$, and Ritz vector $P_3 y_1$. Also, included in Table 5.1 is the sine of the angle for each vector $P_3\hat{v}_1$, $P_3 v_1$, and $P_3 y_1$ with the desired eigenvector x associated with the largest eigenvalue 12 of the matrix A. It should be noted that Krylov subspaces associated with each column in Table 5.1 are different, since they depend on the starting vector. However, we do see from Table 5.1 that using the iterative refined Ritz vector $P_3\hat{v}_1$ we are able to obtain a smaller Ritz residual norm on each restart and a starting vector "closer" to the desired eigenvector. Although the results for iterative refined Ritz norms and angles may appear to be only marginally smaller than refined Ritz norms, in a restarted scheme for large matrices this can be significant.

TABLE 5.1

Example 5.1. Displays the Ritz residual norms $\beta_3|e_3^T y_1|$ (2.6) associated with $\{\theta_1, y_1\}$ for different (re)starting vector p_1 in MLan(0) Algorithm 2.1 on the next restart. Also, displays the sine of the angle for each vector with the eigenvector x associated with the largest eigenvalue 12.

Restart	Iterative refined Ritz		Refined Ritz		Ritz	
	$\beta_3\lvert e_3^T y_1\rvert$	$\sin\angle(x, P_3\hat{v}_1)$	$\beta_3\lvert e_3^T y_1\rvert$	$\sin\angle(x, P_3 v_1)$	$\beta_3\lvert e_3^T y_1\rvert$	$\sin\angle(x, P_3 y_1)$
1	$2.4589 \cdot 10^{-2}$	$2.8770 \cdot 10^{-3}$	$2.4820 \cdot 10^{-2}$	$2.9060 \cdot 10^{-3}$	$6.6286 \cdot 10^{-2}$	$7.8691 \cdot 10^{-3}$
2	$7.0237 \cdot 10^{-4}$	$2.0977 \cdot 10^{-4}$	$7.1591 \cdot 10^{-4}$	$2.1353 \cdot 10^{-4}$	$2.1557 \cdot 10^{-3}$	$5.7290 \cdot 10^{-4}$
3	$1.8391 \cdot 10^{-5}$	$2.1518 \cdot 10^{-6}$	$1.8918 \cdot 10^{-5}$	$2.2148 \cdot 10^{-6}$	$1.5244 \cdot 10^{-4}$	$1.8096 \cdot 10^{-5}$
4	$5.2531 \cdot 10^{-7}$	$1.5699 \cdot 10^{-7}$	$5.4567 \cdot 10^{-7}$	$1.6187 \cdot 10^{-7}$	$4.9572 \cdot 10^{-6}$	$1.3167 \cdot 10^{-6}$

Example 5.1 is a good illustrative example that shows that we can get better results with iterative refined Ritz vectors. However, this example's features are ideal with well–separated eigenvalues of A and large T_m ($m = 3$ compared with $n = 4$) that yields a good approximation to all eigenpairs on the first iteration. In practice, A will be very large, $m \ll n$, and the corresponding generated Krylov subspace yielding a poor approximation to the desired eigenpairs. When m is kept very small we experienced very poor results, often with stagnation. This can be contributed in part to an overall poor approximation from the Krylov subspace and the computational process of refined or iterative refined Ritz values. The following example illustrates this undesirable scenario.

Example 5.2 Let A be the 3×3 symmetric matrix,

$$A = \begin{bmatrix} 0.0025 & 0.0485 & 0 \\ 0.0485 & 2.1509 & 2.3 \\ 0 & 2.3 & 0 \end{bmatrix}. \tag{5.16}$$

The eigenvalues of A are 3.6149, $2.4989 \cdot 10^{-3}$, and -1.4640. Using MLan(0) Algorithm 2.1 on matrix A with $p_1 = [1\ 0\ 0]^T$ and $m = 2$, yields a matrix $T_{3,2}$ that consist of the first 2 columns of A, with eigenvalues of $T_{3,2}(1\colon 2, 1\colon 2)$ as $\theta_1 = 2.1520$ and $\theta_2 = 0.0014$. We used Algorithm 5.1 with $\mu_1 = \theta_1 = 2.1520$ for computing refined Ritz and iterative refined Ritz pairs. Table 5.2a below displays the output of the refined Ritz ρ_1 and iterative refined $\hat{\rho}_1$ values and the sine of the angles between the eigenvector x associated with the largest eigenvalue 3.6149, and the eigenvectors y_1 and y_2, associated with θ_1 and θ_2, respectively. We see that refined Ritz pair $\{\rho_1, v_1\}$ and iterative refined Ritz pair $\{\hat{\rho}_1, \hat{v}_1\}$ are "closer" to $\{\theta_2, y_2\}$ than $\{\theta_1, y_1\}$ even though approximations set $\mu_1 = \theta_1$. This can cause a restarted method with refined Ritz or iterative refined Ritz to stagnate or converge slowly. See Table 5.2b where we set the (re)starting vector p_1 in MLan(0) Algorithm 2.1 on the next restart to be the computed iterative refined Ritz $P_2\hat{v}_1$, refined Ritz $P_2 v_1$, and Ritz vector $P_2 y_1$. The initial iterative refined Ritz pair is very close to the undesired value $\{\theta_2, y_2\}$ and the process cannot recover, causing stagnation. The initial refined Ritz pair is not as close, and does recover, however the overall convergence is a lot slower than using Ritz vector to restart. Also, the computed refined Ritz residual $\sigma_{\downarrow 1}\|r_1\|$ for the method that restarts with refined Ritz vector was only marginally better, e.g. on restart 4 with refined Ritz vector method we have $\sigma_{\downarrow 1}\|r_1\| = 5.3081 \cdot 10^{-5}$ (compared to $\beta_2|e_2^T y_1| = 5.3121 \cdot 10^{-5}$). In this example, notice that the initial Ritz vector is significantly closer to the desired eigenvector x and restarting with Ritz vector converges a lot faster.

TABLE 5.2
Example 5.2.

Iterative Refined Ritz $\hat{\rho}_1 = 0.0017$			Refined Ritz $\rho_1 = 0.0655$			Ritz
$\sin\angle(x, P_2\hat{v}_1)$	$\sin\angle(y_1, \hat{v}_1)$	$\sin\angle(y_2, \hat{v}_1)$	$\sin\angle(x, P_2 v_1)$	$\sin\angle(y_1, v_1)$	$\sin\angle(y_2, v_1)$	$\sin\angle(x, P_2 y_1)$
1.000	0.0120	0.9999	0.9904	0.1727	0.9850	0.5368

(a) Display results when using Algorithm 5.1 with $\mu_1 = \theta_1 = 2.1520$ applied to $T_{3,2}$. Also, displays the sine of the angle for each vector with the eigenvector x associated with the largest eigenvalue 3.6149.

Restart	Iterative refined Ritz		Refined Ritz		Ritz	
	$\beta_2\lvert e_2^T y_1\rvert$	$\sin\angle(x, P_2\hat{v}_1)$	$\beta_2\lvert e_2^T y_1\rvert$	$\sin\angle(x, P_2 v_1)$	$\beta_2\lvert e_2^T y_1\rvert$	$\sin\angle(x, P_2 y_1)$
1	$1.2276 \cdot 10^{0}$	$9.9998 \cdot 10^{-1}$	$9.0575 \cdot 10^{-1}$	$1.8281 \cdot 10^{-1}$	$2.7847 \cdot 10^{-2}$	$7.7072 \cdot 10^{-3}$
2	$2.2985 \cdot 10^{0}$	$1.0000 \cdot 10^{0}$	$3.5993 \cdot 10^{-2}$	$9.9344 \cdot 10^{-3}$	$3.3735 \cdot 10^{-4}$	$6.6449 \cdot 10^{-5}$
3	$1.2275 \cdot 10^{0}$	$1.0000 \cdot 10^{0}$	$1.3828 \cdot 10^{-3}$	$2.7442 \cdot 10^{-4}$	$2.9083 \cdot 10^{-6}$	$8.0490 \cdot 10^{-7}$
4	$2.2983 \cdot 10^{0}$	$1.0000 \cdot 10^{0}$	$5.3121 \cdot 10^{-5}$	$1.4661 \cdot 10^{-5}$	$3.5229 \cdot 10^{-8}$	$6.9394 \cdot 10^{-9}$

(b) Displays the Ritz residual norms $\beta_2|e_2^T y_1|$ (2.6) associated with $\{\theta_1, y_1\}$ for different (re)starting vector p_1 in MLan(0) Algorithm 2.1 on the next restart. Also, displays the sine of the angle for each vector with the eigenvector x associated with the largest eigenvalue 3.6149.

Although the example 5.2 is contrived, it illustrates what can happen. The next example further illustrates the problem.

Example 5.3 Let $A = \text{diag}(1:500)$ be a 500×500 diagonal matrix. We are searching for the largest eigenpair. We set $m = 2$ and the (re)starting vector p_1 in MLan(0) Algorithm 2.1 on the next restart to be the computed Ritz vector $P_2 y_1$, refined Ritz $P_2 v_1$, or iterative refined Ritz $P_2 \hat{v}_1$. We ran the example 100 times with a different beginning random vector p_1. The results are presented in Figure 5.1a, Figure 5.1b, and Figure 5.1c and show that restarting with iterative refined Ritz $P_2 \hat{v}_1$ or refined Ritz $P_2 v_1$ caused stagnation, or erratic behavior and slow convergence. Restarting with Ritz vector, Figure 5.1a was not erratic, but convergence was slow.

Example 5.1 showed that when Krylov subspace was a fairly "good" subspace that restarting with iterative refined Ritz vector provided the better results. However, examples 5.2 and 5.3 demonstrated the pitfalls of using a conventional restarting scheme with just (iterative) refined Ritz vectors when the Krylov subspace was a "poor" subspace. Although examples 5.2 and 5.3 may show that the iterative refined Ritz vectors to be problematic, the iterative refined Ritz vectors highlight, more so than refined Ritz vectors, when they should not be used to restart. Using this information and that restarting with Ritz vectors convergence was not erratic (although slow), we developed a hybrid method that switches, depending on some parameters, between thick–restarting with Ritz vectors and restarting with iterative refined Ritz vectors and show this combination overcomes the stagnation and erratic behavior producing a faster overall converging method.

6. Hybrid Methods. The hybrid method developed here uses thick–restarted as the main routine and, under certain conditions, switches to restarting with iterative refined Ritz vectors. The iterative refined Ritz vectors from a "good" Krylov subspace can be better approximations, but a trade off is a restarted scheme that loses the benefits of thick–restarted which are crucial and traced back to the IRL method. Using only the thick–restarted with a small m value converges very slowly or not at all. We have found that when the Krylov subspace is "good" that switching to restarting with iterative refined Ritz vectors, even for a few iterations, results in a faster overall convergence.

For $k = 1$ (single eigenpair) restating equations (3.1) and (5.11) for the Ritz pair $\{x_1, \theta_1\}$ and the iterative refined Ritz pair $\{\hat{z}_1, \hat{\rho}_1\}$

$$Ax_1 = [x_1, p_2] \begin{bmatrix} \theta_1 \\ \bar{\beta}_1 \end{bmatrix} \quad \text{where } p_2 = f/\beta_m \tag{6.1}$$

$$A\hat{z}_1 = [\hat{z}_1, p_2] \begin{bmatrix} \hat{\rho}_1 \\ \hat{\sigma}_{\downarrow 1} \end{bmatrix} \quad \text{where } p_2 = P_{m+1}\hat{u}_1. \tag{6.2}$$

Depending on certain parameters for switching described in subsection 6.1, we can restart by calling the MLan(1) Algorithm 2.1 with starting vector p_2. This is slightly different than a restarted Lanczos method, by using relationships in equations (6.1) and (6.2) to avoid a matrix–vector product with A on each restart.

The hybrid method for finding $k > 1$ eigenpairs has added challenges. The thick–restarted algorithm is set up to compute $k \geq 1$ eigenpairs, however the iterative refined Ritz does not fit this structure. Therefore, we implement a standard restart technique with a starting vector p_1 constructed as a linear combination of k iterative refined Ritz vectors

$$p_1 = \sum_{j=1}^{k} c_j \hat{z}_j. \tag{6.3}$$

This was the set up for the refined Ritz algorithm [6, Algorithm 1] where the combination of the refined Ritz vectors for the starting vectors p_1 were constructed based on coefficients c_j chosen as described in Saad [18]. In our development, we chose the coefficients c_j in a similar way to the description outlined by Morgan [14]. In simplest terms, for Ritz vectors x_j, constants c_j are chosen for a starting vector $p_1 = c_1x_1 + \ldots + c_kx_k$ to eliminate the coefficients $\beta_m e_m^T y_j$ multiplying the common residual vector p_{m+1}. That is, the choice of coefficients removes p_{m+1} when the starting vector p_1 is multiplied by A in the next iteration to build out the Krylov subspace. The coefficients can be determined by solving a certain homogenous $k-1 \times k$ linear system. It was then proven by Morgan with these specially chosen coefficients c_j that the span$\{p_1, Ap_1, \ldots, A^{k-1}p_1\}$ = span$\{x_1, x_2, \ldots, x_k\}$ which is the same subspace resulting from implementing Sorensen's IRA method [21]. We refer the reader to [14] for specific details and theoretical results. Building on this idea, we can, in a similar way, remove some of the coefficients associated with p_{m+1} for iterative refined vectors. The rationale on implementing this technique is to have a restarted method that may inherit similar convergence benefits of the IRL method. We did observe fast convergence, even with removing only some of the coefficients associated with p_{m+1}.

From equations (4.8), (5.11), and (6.3) we have

$$Ap_1 = \sum_{j=1}^{k} c_j(\hat{\rho}_j P_m \hat{v}_j + \beta_m e_m^T \hat{v}_j p_{m+1} + \hat{\sigma}_{\downarrow j} P_m \hat{u}_{j(1:m)}). \tag{6.4}$$

Therefore, we select coefficients c_j such that $\beta_m e_m^T \hat{v}_j c_j = 0$ which removes p_{m+1} from (6.4). If we ignore $\hat{\sigma}_{\downarrow j} P_m \hat{u}_{j(1:m)}$ in (6.4) and all future occurrences, as we multiple (6.4) by A we obtain a similar $k-1 \times k$ homogenous system linear system to the one presented in [14, Section 3] where i^{th} row is represented as,

$$\beta_m \begin{bmatrix} \hat{\rho}_1^{i-1} e_m^T \hat{v}_1 & \hat{\rho}_2^{i-1} e_m^T \hat{v}_2 & \ldots & \hat{\rho}_k^{i-1} e_m^T \hat{v}_k \end{bmatrix}. \tag{6.5}$$

If we leave $\hat{\sigma}_{\downarrow j} P_m \hat{u}_{j(1:m)}$ in (6.4) and continue with p_{m+1} removed in (6.4), we have,

$$A^2 p_1 = \sum_{j=1}^{k} c_j(\hat{\rho}_j^2 P_m \hat{v}_j + \hat{\rho}_j \hat{\sigma}_{\downarrow j} P_m \hat{u}_{j(1:m)} + \beta_m \hat{\rho}_j e_m^T \hat{v}_j p_{m+1} + \hat{\sigma}_{\downarrow j}\ AP_m \hat{u}_{j(1:m)}) \tag{6.6}$$

where

$$\hat{\sigma}_{\downarrow j}\ AP_m \hat{u}_{j(1:m)} = \hat{\sigma}_{\downarrow j} P_m T_m \hat{u}_{j(1:m)} + \beta_m (e_m^T T_m \hat{v}_j - \hat{\rho}_j e_m^T \hat{v}_j) p_{m+1}. \tag{6.7}$$

Collecting terms multiplying p_{m+1} in (6.6) and (6.7) we have,

$$(\beta_m \hat{\rho}_j e_m^T \hat{v}_j + \beta_m (e_m^T T_m \hat{v}_j - \hat{\rho}_j e_m^T \hat{v}_j)) c_j = \beta_m e_m^T T_m \hat{v}_j c_j. \tag{6.8}$$

We therefore select coefficients c_j such that $\beta_m e_m^T T_m \hat{v}_j c_j = 0$ which removes p_{m+1} from (6.6). If we ignore $\hat{\sigma}_{\downarrow j} P_m T_m \hat{u}_{j(1:m)}$ in (6.7) and all future occurrences, as we multiply (6.4) by A we obtain the following $k-1 \times k$ homogenous system linear system where coefficient matrix has the same first row as in (6.5) and is represented as,

$$\beta_m \begin{bmatrix} e_m^T \hat{v}_1 & e_m^T \hat{v}_2 & \ldots & e_m^T \hat{v}_k \\ \hat{\rho}_1^{i-2} e_m^T T_m \hat{v}_1 & \hat{\rho}_2^{i-2} e_m^T T_m \hat{v}_2 & \ldots & \hat{\rho}_k^{i-2} e_m^T T_m \hat{v}_k \end{bmatrix} \quad i > 1. \tag{6.9}$$

Notice from (5.8) that as $\hat{\sigma}_{\downarrow j}$ approaches zero, we expect $T_m \hat{v}_j$ to approach $\hat{\rho}_j \hat{v}_j$ and (6.9) would become like the homogenous system (6.5). If we leave $\hat{\sigma}_{\downarrow j} P_m T_m \hat{u}_{j(1:m)}$ in

(6.7) and continue to remove p_{m+1} we get the $k-1 \times k$ homogenous linear system where coefficient matrix has the same first two rows as in (6.9) and is represented as,

$$\beta_m \begin{bmatrix} e_m^T \hat{v}_1 & e_m^T \hat{v}_2 & \dots & e_m^T \hat{v}_k \\ e_m^T T_m \hat{v}_1 & e_m^T T_m \hat{v}_2 & \dots & e_m^T T_m \hat{v}_k \\ \hat{\rho}_1^{i-2} e_m^T T_m \hat{v}_1 + s_1 & \hat{\rho}_2^{i-2} e_m^T T_m \hat{v}_2 + s_2 & \dots & \hat{\rho}_k^{i-2} e_m^T T_m \hat{v}_k + s_k \end{bmatrix} \quad i > 2 \tag{6.10}$$

where

$$s_j = \hat{\sigma}_{1_j} \sum_{\ell=3}^{i} \hat{\rho}_j^{i-\ell} e_m^T T_m^{\ell-2} \hat{u}_{j(1:m)} \qquad 1 \le j \le k. \tag{6.11}$$

Likewise, as $\hat{\sigma}_{1_j}$ approaches zero, we expect $T_m \hat{v}_j$ to approach $\hat{\rho}_j \hat{v}_j$ and s_j to approach zero, hence (6.10) would also become like the homogenous system (6.5). The matrices become more complicated and ill–conditioned as we include more terms for eliminating the coefficients multiplying p_{m+1}. We do assume that k is small and have observed similar results with all three systems, but more consistent results over a wide range of problems when using (6.9) or (6.10). We solved the $k-1 \times k$ homogenous linear system by finding the null space vector using the singular value decomposition. When a column becomes numerically zero, indicating an iterative refined Ritz vector has converged, we remove that column and the last row of the matrix and compute the null space vector of the reduced matrix. We then replace the corresponding coefficient with the norm of the iterative refined Ritz residual vector before creating the linear combination for restart.

Notice that when restarting with the linear combination of iterative refined Ritz vectors, a single matrix–vector product can be saved per iteration by utilizing the relationship (6.4) before restarting. Setting $p_1 = p_1/\|p_1\|$ and setting $\tilde{f}$ to be right side of the equality in (6.4) multiplied by $1/\|p_1\|$ we have $Ap_1 = \tilde{f}$, $\tilde{\alpha}_1 = p_1^T \tilde{f}$ and $\tilde{f} = \tilde{f} - p_1 \tilde{\alpha}_1$, $\tilde{\beta}_1 = \|\tilde{f}\|$ and

$$Ap_1 = [p_1, p_2] \begin{bmatrix} \tilde{\alpha}_1 \\ \tilde{\beta}_1 \end{bmatrix} \tag{6.12}$$

where $p_2 = \tilde{f}/\tilde{\beta}_1$. The MLan(1) Algorithm 2.1 can be continued with p_2. Algorithm 6.1 outlines the hybrid method.

6.1. Hybrid Thick–Restarted and Iterative Refined Ritz Algorithm. Hybrid Thick–Restarted and Iterative Refined Ritz Algorithm 6.1 presents the main algorithm of the paper. Algorithm 6.1 starts with the efficient thick–restarted routine. We provide parameters as to when restarting with iterative refined Ritz vectors can be used. The parameters were chosen from numerous experiments on a variety of problems. A careful balance is needed, since the iterative refined Ritz vectors can give a better approximation when the Krylov subspace is "good", but thick–restarted is a more efficient restarting scheme, but often has slower convergence. Also, as illustrated in Example 5.2 in section 5, the iterative refined Ritz pair may be "closer" to a different Ritz pair of T_m than the originally sought after Ritz pair. Therefore, since m is kept small relative to k we suggest using thick–restarted for the beginning iterations to build a more accurate approximation subspace, i.e. until $\max_{1 \le j \le k} |\bar{\beta}_j| \le \epsilon^{0.1} \|A\|$ where ϵ is a user input tolerance for overall convergence and $\bar{\beta}_j$ is from (3.1). We then check the angle via inner product between $\hat{z}_j$ and x_j, i.e. between $\hat{v}_j$ and y_j. If the angle is acceptable we use iterative refined Ritz vector(s) to restart. Numerous experiments

suggest using $\min_{1\le j\le k} |y_j^T \hat{v}_j| > 0.9$. However, when $k > 1$ these conditions alone do not always prevent missing eigenvalues. One solution is to also require the input value μ_j into Iterative Refined Algorithm 5.1 to be the best approximation eigenvalue of A over all computed θ_j's values thus far and to reject using restarting with iterative refined Ritz vectors if the current computed $\hat{\rho}_j$ are not "better" than the past iteration's best approximation. For example, during a current iteration (iter) of Algorithm 6.1, if searching for the k largest in magnitude eigenpairs, we require in step 5 for the call to Algorithm 5.1 that

$$\mu_j = \max_{1 \le \mathrm{i} \le \mathrm{iter}} |\theta_j^{(i)}| \quad \text{for} \quad 1 \le j \le k \tag{6.13}$$

and for step 7

$$|\hat{\rho}_j^{(\mathrm{iter})}| \ge \max_{1 \le \mathrm{i} \le \mathrm{iter\text{-}1}} |\theta_j^{(i)}| \quad \text{for} \quad 1 \le j \le k. \tag{6.14}$$

Similar requirements are made for other desired extreme eigenvalue locations. When $k = 1$ we found that using (6.13) was a needed requirement for the best results, but encountered poor convergence results when enforcing (6.14) with $m = 2$, see Examples 7.4 and 7.5 in section 7. The following example illustrates the methods presented.

Algorithm 6.1 Hybrid Thick–Restarted and Iterative Refined Ritz

1: **Input:** $A \in \mathbb{R}^{n\times n}$: symmetric matrix,
$p_1 \in \mathbb{R}^n$: orthonormal vector,
ϵ: usr specified tolerance,
δ_1: usr specified tolerance on when switching can start
(recommended: $\delta_1 := \epsilon^{0.1}$),
δ_2: usr specified tolerance on when vectors are "close"
(recommended: $\delta_2 := 0.9$),
k: number of desired eigenpairs of A,
m: maximum number of Lanczos vectors.
2: **Output:** k approximate desired eigenpair(s) of A.
3: Compute factorization (2.2) by MLan(0) Algorithm 2.1
4: Compute the k desired eigenpair $\{\theta_j, y_j\}_{j=1}^k$ of T_m (2.4) or $\bar{T}_m$ (3.8)
5: Compute $\{\hat{\rho}_j, \hat{v}_j, \hat{u}_j, \hat{\sigma}_j\}_{j=1}^k$ by Iterative Refined Algorithm 5.1 with e.g. μ_j (6.13)
6: Check convergence (2.7) or (7.1)
7: **if** all $\hat{\rho}_j$ converged in Algorithm 5.1 and satisfy e.g. (6.14) **then**
8: **if** $\max_{1\le j\le k} |\bar{\beta}_j| \le \delta_1 \|A\|$ and $\min_{1\le j\le k} |x_j^T \hat{z}_j| > \delta_2$ **then**
9: **if** $k > 1$ **then** compute c_j from (6.5), (6.9) or (6.10)
10: Restart with iterative refined (6.2) or (6.12) via MLan(1) Algorithm 2.1
11: **else**
12: Restart with Ritz (6.1) or (3.3) via MLan(k) Algorithm 2.1
13: Goto 4

Example 6.1 Let A be the 256,000 × 256,000 matrix *Lin* from the SuiteSparse Matrix Collection [4]. The largest in magnitude eigenvalue is 1063.63 and the next three largest in magnitude are 1063.32, 1063.31, and 1062.98. We set $m = 15$ and $k = 1$ and $k = 4$. We compared Thick–Restarted Ritz Algorithm 3.1 with Algorithm 6.1. For $k = 1$ we also included the restarted refined Ritz as described in [6, Algorithm

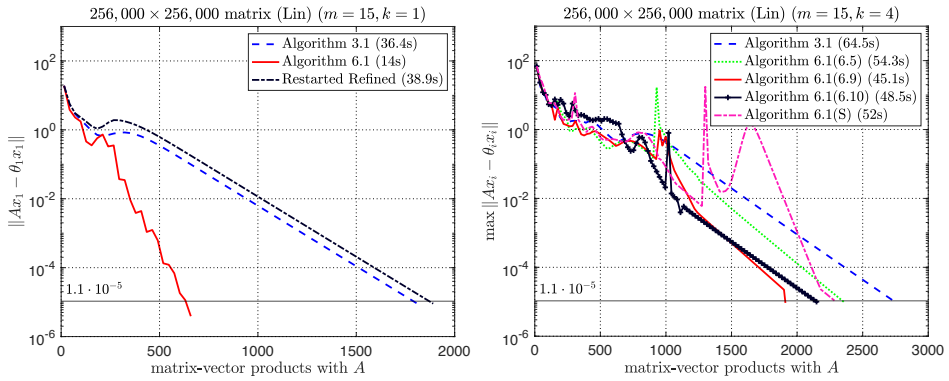

FIG. 6.1. *Example 6.1. Matrix A is the* $256,000 \times 256,000$ *matrix Lin from the SuiteSparse Matrix Collection [4]. Algorithm 6.1 converges faster than the other restarted methods.*

1]. We did not include restarted refined Ritz for $k = 4$ as the method did not converge within 3000 matrix-vector products. We used the same starting vector for each routine, a normalized random vector and a stopping criteria $10^{-8}\|A\|$. For fair comparisons, we recorded the Ritz norm residual $\|Ax_1 - \theta_1 x_1\|$ for $k = 1$ and $\max \|Ax_i - \theta_i x_i\|$ for $k = 4$. For $k = 4$ we compared results using Algorithm 3.1 with Algorithm 6.1 using (6.5), (6.9), and (6.10). We also computed the coefficients in (6.3) using Saad's method [18] indicated by Algorithm 6.1 (S) in the legend. The graphs in Figure 6.1 show that Algorithm 6.1 with (6.9) and (6.10) outperforms the other routines. Numerical comparison with this matrix with other routines is given in Example 7.3 in section 7.

Example 6.2 Revisiting Example 5.3 where $A = \text{diag}(1{:}500)$ a 500×500 diagonal matrix and $m = 2$. Figures 5.1a, 5.1b, and 5.1c showed for 100 restarts we had stagnation, or erratic behavior and no convergence within 500 matrix-vector products using Ritz vector $P_2 y_1$, refined Ritz $P_2 v_1$ or iterative refined Ritz $P_2 \hat{v}_1$ as the restarting vector in MLan(0) Algorithm 2.1. Figure 5.1e, Algorithm 6.1, with 100 random restarts always converged within tolerance $10^{-8}\|A\|$ with no more than 170 matrix-vector products and typical convergence between 100 and 150 matrix-vector products. We also modified Algorithm 6.1 to call Algorithm 5.1 to compute refined Ritz vectors. All other parameters, in Algorithm 6.1 remained the same. Figure 5.1d displays the results and shows using refined Ritz vectors in place of iterative refined Ritz vectors performed poorly.

7. Numerical Examples. This section presents some numerical examples that illustrate the performance of Algorithm 6.1. For ease of comparisons we implemented Algorithm 6.1 in a MATLAB code called `trreigs`[1]. We compare our method to the publicly available MATLAB code `irbleigs`[1][1], the MATLAB interfaced code `primme_eigs`[22][2], and MATLAB's built–in function `eigs`. We refer the reader to the citations and noted websites for full details and descriptions of parameters. There are numerous selections and varieties of combinations of parameters for each code. Some choices and combinations yield faster convergence than others. We cannot provide examples with all possible combinations. We used either the default values for the parameters or parameter choices that represent the fairest comparison with respect to Lanczos basis size and similarity with respect to the foundational Lanczos method for

[1]Code available at: http://www.math.uri.edu/~jbaglama
[2]Code available at: https://github.com/primme/primme

`trreigs`. All examples and methods used a common unit length vector with random entries that are normally distributed entries with zero mean.

The parameters for `trreigs` are based on Algorithm 6.1. For all examples we set $\delta_1 := \epsilon^{0.1}$, $\delta_2 := 0.9$, and maximum iterations 100 for Algorithm 5.1. The number of eigenpairs k, maximum size of the Lanczos basis m, tolerance for convergence ϵ, and location of eigenvalues are set depending on the example. In addition to checking the Ritz residual norm (2.7) for termination, the code `trreigs` also checks

$$(7.1)\quad \|AP_m\hat{q}_j - \hat{\rho}_j P_m\hat{q}_j\| = \sqrt{(T_m\hat{q}_j - \hat{\rho}_j\hat{q}_j)^T(T_m\hat{q}_j - \hat{\rho}_j\hat{q}_j) + \beta_m^2(e_m^T\hat{q}_j)^2} \le \epsilon\|A\|$$

where $\hat{q}_j$ is the j^{th} column of the QR factorization of $[\hat{v}_1, \ldots, \hat{v}_k]$. We use the QR factorization to ensure that the eigenvectors are orthogonal. Furthermore, to help avoid the pitfalls of Example 5.2, we only check (7.1) when $|x_j^T\hat{z}_j| > \delta_2$. The technique of including additional vectors ($> k$) can greatly accelerate the convergence in restarted methods, like thick–restarting with Ritz vectors. There are many strategies for determining the number of restart vectors, see e.g. [24, 26]. A comparison of heuristic techniques is given in [26]. We implemented a simple but often effective strategy when the hybrid scheme uses thick–restarted, we restart with

$$(7.2)\qquad k = \max(\text{floor}(n_c + (m - n_c)/2), k)$$

vectors where n_c is the number of converged desired approximate eigenvectors. However, using more than k vectors in the restarting scheme for iterative refined part was found to be counterproductive, often not satisfying the criteria in Algorithm 6.1 for switching. Since $m \ll n$ the code `trreigs` uses full reorthogonalization as outlined in step 8 of Algorithm 2.1. This is is a simple strategy, but can increase overall computational times.

The MATLAB code `irbleigs` is a block Lanczos method that uses the implicitly restarted formulas to apply Leja points as shifts. Given a Lanczos basis with m blocks, the method applies m Leja shifts via the implicit shift formulas until a single block of vectors is obtained and then restarts. Although the method utilizes implicit formulas for applying shifts to obtain a starting block, the overall structure can be considered an explicit restarted Lanczos method. For fair comparisons, `irbleigs` should be restricted to block size one, however that restriction often caused abnormally large number of matrix–vector products or no convergence. Therefore, in order to provide the fairest comparison with `irbleigs`, when appropriate, we recorded results with block size greater than one where the combination with the number of blocks is equal to the maximum size of the Lanczos basis m. Block size and number of blocks for `irbleigs` are reported in Table 7.1 as (block size, number of blocks). The common starting vector is used, where the rest of the starting block is filled in with random vectors. Besides the number k of desired eigenpairs and tolerance for convergence we used the default settings for all other parameters.

MATLAB's built–in function `eigs` used symmetric parameter true and Lanczos basis max size as m, and all default settings except the number k of desired eigenpairs, convergence tolerance, and common starting vector.

The MATLAB interfaced code `primme_eigs` uses the state–of–the–art high performance C99 library PRIMME for computing the eigenvalues and eigenvectors. This is an impressive, carefully designed code that includes numerous parameter settings, multiple routines/techniques, and options to include preconditioning. There are 15 choices for methods. For comparisons, we used the setting for method to be

"default_min_matvecs" (referred to as min_mv in Table 7.1) which is the best method for heavy matrix-vector products and performed better than the default "dynamic" for Examples 7.1 to 7.4. On some m choices for Example 7.5, "dynamic" performed better, therefore we reported the better results for Example 7.5 ("dynamic" is referred to as dyn in Table 7.1). We also included the method "jdqmr_etol" (referred to as jd_tol in Table 7.1). `primme_eigs` allows the user to input preconditioners to accelerate convergence. We did not apply any preconditioners for the reported examples. We set the parameters "isreal" and "isdouble" to be true and used k for desired eigenpairs and the common tolerance for convergence. Unless specified in the example, we used the default values for all other adjustable parameters. `primme_eigs` allows the user to include any number of initial guesses to the eigenvectors, however we only set a starting vector to the routine to be the common starting vector used for all routines in that example. `primme_eigs` allows the user to select the maximum size of the search subspace. We set the maximum size of the search subspace to be the common restrict value m for the other routines. It should be noted, the storage requirement and maximum size of the search space for `primme_eigs` are not always equivalent, see [22, Section 3.4.1].

All examples use the location of eigenvalues to be largest in magnitude. Example 7.1 also includes an example for smallest algebraic. In all examples, the matrix A was only accessed by call to a function with input x and output Ax. In the Table 7.1, the cpu times are in seconds recorded using MATLAB's tic-toc command. The row for error represents $\max \|Ax_i - \lambda_i x_i\|$ which was computed outside the routines with the outputted approximations. The references to (6.9) and (6.10) refer to the different matrices used to find the coefficients for `trreigs`. Similar to Example 6.1, using (6.5) reported inferior results and is not recorded. We finally remark that the performance of the methods in our comparisons also depends on the machine architecture, MATLAB coding style, and numerical implementation, (e.g. selective, partial or full reorthgonalization). The MATLAB code `trreigs` is only an illustration of Algorithm 6.1 and was not designed in the same fashion as the publicly/commercially developed codes. Nevertheless, the examples do show that Algorithm 6.1 can match or outperform the performance of the other methods. All numerical examples were performed on matrices from SuiteSparse Matrix Collection [4] and all computations were carried out using MATLAB version R2019b on an iMac with 3.7Ghz Intel Core i5 processor and 32GB (2667 MHz) of memory using operating system macOS Mojave. Machine epsilon is $\epsilon = 2.2 \cdot 10^{-16}$.

Example 7.1. We considered two matrices, 2,680 × 2,680 *dwt2680* and 2,233 × 2,233 *lshp2233* that were used as numerical examples in [8]. For *dwt2680* the author was seeking 5 dominant eigenvalues and for *lshp2233* the 5 smallest algebraic eigenvalues. Both examples used a stopping criteria of 10^{-6}. The examples in [8] compared several related methods, the implicitly restarted Arnoldi (IRA), the implicitly restarted refined Arnoldi (IRRA), and the implicitly restarted refined harmonic Arnoldi (IRRHA). As a point of reference the best computed result for *dwt2680* for the smallest used space $m = 20$, was for the IRRA with 244 mvp, [8, Table 7] and the best computed result for *lshp2233* for the smallest used space $m = 20$, was for the IRRHA with 1333 mvp, [8, Table 6]. Table 7.1a displays the results for *dwt2680* and Table 7.1b displays the results for *lshp2233*. For both matrices and all methods we used $k = 5$ and $\epsilon = 10^{-6}$. For *dwt2680* we display results for $m = 10, 20$ and for *lshp2233* for $m = 20$. The code `trreigs` displays the best results with respect to mvp for *lshp2233* and when $m = 10$ for *dwt2680* with comparable results when $m = 20$.

Example 7.2. We considered the 12,992 × 12,992 matrix *tuma2*. This was the

only symmetric matrix that was use as a numerical example in [11] for finding the 6 dominant eigenvalues with a stopping criteria of 10^{-10}. The example in [11] compared several related methods, thick–restarted block Arnoldi, modified thick–restarted block Arnoldi, a hybrid modified Ritz thick–restarted and refined block Arnoldi method, and the block Krylov-Schur algorithm [29]. As a point of reference the best computed result with respect to mvp for the smallest used space $m = 18$, was for the modified thick–restarted block Arnoldi with 1520 mvp, [11, Table 6]. Table 7.1c displays the results for $k = 5$, $m = 10, 18$ and $\epsilon = 10^{-10}$. The code `trreigs` displays competitive results.

Example 7.3 We considered the 256,000 × 256,000 matrix *Lin* that was used in Example 6.1 in subsection 6.1. We are searching for largest eigenvalue(s) and associated vector(s). We compared the codes with $k = 1, 4$, $m = 15$, and $\epsilon = 10^{-8}$. Table 7.1d displays the results. Notice that the results are significantly better than the recorded results in Fig. 6.1. This is due in part to using the strategy (7.2) for the thick–restarted scheme and incorporating stopping criteria (7.1). The code `trreigs` displays the best results with respect to mvp when compared to `eigs` and `irbleigs` for both $k = 1$ and $k = 4$.

Example 7.4 We considered the 1,062,400 × 1,062,400 matrix *nlpkkt80*. We are searching for largest eigenvalue and associated eigenvector while using the smallest search space. We set $\epsilon = 10^{-6}$. `eigs` did not record convergence within 6000 matrix-products until $m = 10$. For m equal to 3 for `primme_eigs` we set the parameters $maxPrevRetain = 1$ and $minRestartSize = 1$, otherwise they were set as the default values. The reported largest eigenvalue was 259.799. We include results for `trreigs` not requiring (6.14) restriction. This column is labeled 'FLT' under `trreigs`. Table 7.1e displays the results. The code `trreigs` with 'FLT' displays the best results with respect to mvp and smallest space $m = 2$.

Example 7.5 We considered the 214,005,017×214,005,017 matrix *kmer_V1r*. We are searching for largest eigenvalue and associated eigenvector while using the smallest possible search space. We set $\epsilon = 10^{-6}$ and used the smallest possible value for m for each routine to get convergence within 200 matrix–vector products. The Matlab code, `irbleigs` did not converge for $m = 3, 4, 5$ and jdqmr_etol did not converge for $m = 3, 4$ and therefore are not reported. For m equal to 3 for `primme_eigs` we set the parameters $maxPrevRetain = 1$ and $minRestartSize = 1$, otherwise they were set as the default values. We include results for `trreigs` not requiring (6.14) restriction. This column is labeled 'FLT' under `trreigs`. The reported largest eigenvalue was 6.50346. Table 7.1f displays the results. For the smallest space $m = 2$ the code `trreigs` with 'FLT' displays competitive results.

8. Conclusions. This paper presents a restarted hybrid method that combines thick–restarting with restarting with a linear combination of iterative refined Ritz vectors. The method does not require factorization of A, and can therefore be applied to very large problems. Numerical examples show the method to be competitive with other available codes with respect to matrix–vector products and storage required.

Appendix A. Lemma.

LEMMA A.1. *Given* $\|y\| = 1$ *and* $\|x\| = 1$ *then* $\|x - (x^T y)y\| \leq 1$. *Additionally, if* $x^T y \neq 0$ *then* $\|x - (x^T y)y\| < 1$.

Proof. Follows from $\|x-(x^T y)y\|^2 = 1-(x^T y)^2$ and $0 \leq (x^T y)^2 \leq \|x\|^2\|y\|^2 = 1$.□

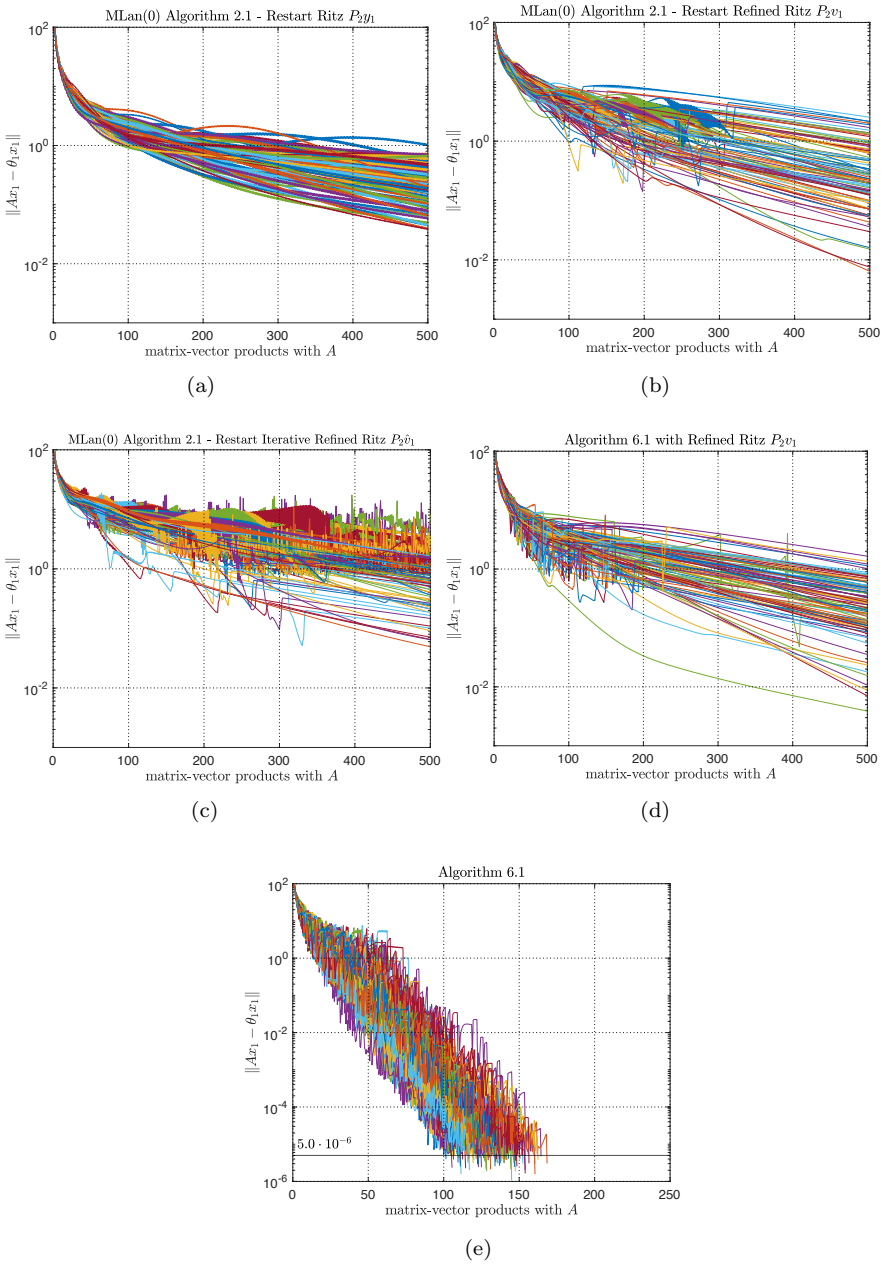

FIG. 5.1. *Example 5.3 and 6.2. Matrix* $A = \text{diag}(1:500)$ *is a* 500×500 *diagonal matrix. We are searching for the largest eigenpair and set* $m = 2$. *Figures (a), (b), and (c) use MLan(0) Algorithm 2.1 where the (re)starting vector* p_1 *on the next restart to be Ritz vector* P_2y_1, *refined Ritz* P_2v_1, *and iterative refined Ritz* $P_2\hat{v}_1$, *respectively. Figure (d) displays Algorithm 6.1 but with refined Ritz vectors in place of iterative refined Ritz vectors and figure (e) uses Algorithm 6.1 as presented. The example is done 100 times for each figure with a different beginning random vector. Each line represent a start with a random vector and then a restart using the stated vector. The process was terminated at 500 matrix-vector products or when* $\|Ax_1 - \theta_1 x_1\| \leq 10^{-8}\|A\|$. *Only Algorithm 6.1 as presented converged, figure (e).*

TABLE 7.1
Numerical Examples section 7.

	trreigs				irbleigs		eigs		primme_eigs	
m	10		20		10	20	10	20	10	
	(6.9)	(6.10)	(6.9)	(6.10)	(1,10)	(1,20)			min_mv	jd_tol
mvp	98	106	94	129	140	188	104	86	102	173
cpu	0.10s	0.06s	0.04s	0.05s	0.16s	0.06s	0.05s	0.02s	0.04s	0.02s
err	$7.9\cdot10^{-6}$	$7.6\cdot10^{-6}$	$6.1\cdot10^{-6}$	$9.4\cdot10^{-7}$	$8.0\cdot10^{-6}$	$7.7\cdot10^{-6}$	$6.9\cdot10^{-6}$	$2.3\cdot10^{-6}$	$1.1\cdot10^{-5}$	$1.1\cdot10^{-5}$

(a) Example 7.1. 2,680 × 2,680 matrix *dwt2680* ($k = 5$). Largest in magnitude.

	trreigs		irbleigs	eigs	primme_eigs	
	(6.9)	(6.10)	(4,5)		min_mv	jd_tol
mvp	497	435	920	510	456	687
cpu	0.34s	0.27s	0.23s	0.08s	0.08s	0.05s
err	$2.8\cdot10^{-6}$	$2.5\cdot10^{-6}$	$3.2\cdot10^{-6}$	$8.8\cdot10^{-7}$	$6.8\cdot10^{-6}$	$4.1\cdot10^{-6}$

(b) Example 7.1. 2,233 × 2,233 matrix *lshp2233* ($k = 5$, $m = 20$). Smallest algebraic.

	trreigs				irbleigs		eigs		primme_eigs	
m	10		18		10	18	10	18	10	
	(6.9)	(6.10)	(6.9)	(6.10)	(2,5)	(3,6)			min_mv	jd_tol
mvp	647	498	264	264	488	561	713	240	408	586
cpu	0.56s	0.44s	0.26s	0.25s	0.34s	0.25s	0.29s	0.09s	0.21s	0.12
err	$2.6\cdot10^{-10}$	$2.2\cdot10^{-10}$	$4.3\cdot10^{-10}$	$4.3\cdot10^{-10}$	$1.9\cdot10^{-7}$	$1.2\cdot10^{-8}$	$4.4\cdot10^{-10}$	$4.8\cdot10^{-10}$	$4.5\cdot10^{-10}$	$3.9\cdot10^{-10}$

(c) Example 7.2. 12,992 × 12,992 matrix *tuma2* ($k = 6$). Largest in magnitude.

	trreigs			irbleigs		eigs		primme_eigs			
k	1	4		1	4	1	4	1		4	
		(6.9)	(6.10)	(3,5)	(3,5)			min_mv	jd_tol	min_mv	jd_tol
mvp	491	970	1129	1170	1953	839	1095	380	449	649	874
cpu	6.89s	13.85s	18.56s	8.73s	16.73s	4.05s	4.64s	3.62s	1.69s	6.70s	3.22s
err	$7.6\cdot10^{-6}$	$9.5\cdot10^{-6}$	$7.8\cdot10^{-6}$	$7.9\cdot10^{-6}$	$1.0\cdot10^{-5}$	$8.3\cdot10^{-6}$	$5.0\cdot10^{-6}$	$9.7\cdot10^{-6}$	$1.1\cdot10^{-5}$	$1.1\cdot10^{-5}$	$1.1\cdot10^{-5}$

(d) Example 7.3. 256,000 × 256,000 matrix *Lin* ($m = 15$). Largest in magnitude.

	trreigs		irbleigs	eigs	primme_eigs			
m	2	2	3	10	3		4	
		FLT	(1,3)		min_mv	jd_tol	min_mv	jd_tol
mvp	1098	692	1422	5800	2014	1507	742	1010
cpu	48.73s	30.62s	58.08s	227.66s	81.45s	49.22s	31.94s	32.97s
err	$2.4\cdot10^{-4}$	$2.6\cdot10^{-4}$	$1.8\cdot10^{-4}$	$2.6\cdot10^{-4}$	$2.5\cdot10^{-4}$	$2.2\cdot10^{-4}$	$2.6\cdot10^{-4}$	$2.3\cdot10^{-4}$

(e) Example 7.4. 1,062,400 × 1,062,400 matrix *nlpkkt80* ($k = 1$). Largest in magnitude.

	trreigs		eigs			primme_eigs			
m	2	2	3	4	5	3	4	5	
		FLT				min_mv	dyn	dyn	jd_tol
mvp	98	70	131	132	98	60	58	69	80
cpu	987.81s	650.78s	599.94s	654.22s	490.73s	370.15s	493.21s	2998.2s	2225.7s
err	$6.4\cdot10^{-6}$	$5.4\cdot10^{-6}$	$6.3\cdot10^{-6}$	$5.5\cdot10^{-6}$	$5.5\cdot10^{-6}$	$5.2\cdot10^{-6}$	$4.6\cdot10^{-6}$	$4.8\cdot10^{-6}$	$3.1\cdot10^{-6}$

(f) Example 7.5. 214,005,017 × 214,005,017 matrix *kmer_V1r* ($k = 1$). Largest in magnitude.

REFERENCES

[1] J. BAGLAMA, D. CALVETTI, AND L. REICHEL, *IRBL: An implicitly restarted block-Lanczos method for large-scale Hermitian eigenproblems*, SIAM Journal on Scientific Computing, 24 (2003), pp. 1650–1677.

[2] Z. BAI, J. DEMMEL, J. DONGARRA, A. RUHE, AND H. VAN DER VORST, *Templates for the solution of algebraic eigenvalue problems: a practical guide*, SIAM, 2000.

[3] Y. Cai, L.-H. Zhang, Z. Bai, and R.-C. Li, *On an eigenvector-dependent nonlinear eigenvalue problem*, SIAM Journal on Matrix Analysis and Applications, 39 (2018), pp. 1360–1382.
[4] T. A. Davis and Y. Hu, *The University of Florida sparse matrix collection*, ACM Transactions on Mathematical Software, 38 (2011), pp. 1–25.
[5] S. Feng and Z. Jia, *A refined Jacobi-Davidson method and its correction equation*, Computers & Mathematics with Applications, 49 (2005), pp. 417–427.
[6] Z. Jia, *Refined iterative algorithms based on Arnoldi's process for large unsymmetric eigenproblems*, Linear Algebra and its Applications, 259 (1997), pp. 1–23.
[7] Z. Jia, *Polynomial characterizations of the approximate eigenvectors by the refined Arnoldi method and an implicitly restarted refined Arnoldi algorithm*, Linear Algebra and its Applications, 287 (1999), pp. 191–214.
[8] Z. Jia, *The refined harmonic Arnoldi method and an implicitly restarted refined algorithm for computing interior eigenpairs of large matrices*, Applied Numerical Mathematics, 42 (2002), pp. 489–512.
[9] Z. Jia, *Some theoretical comparisons of refined Ritz vectors and Ritz vectors*, Science in China Series A: Mathematics, 47 (2004), pp. 222–233.
[10] Z. Jia and G. W. Stewart, *An analysis of the Rayleigh–Ritz method for approximating eigenspaces*, Mathematics of Computation, 70 (2001), pp. 637–647.
[11] W. Jiang and G. Wu, *A thick-restarted block Arnoldi algorithm with modified Ritz vectors for large eigenproblems*, Computers & Mathematics with Applications, 60 (2010), pp. 873–889.
[12] R. B. Lehoucq and D. C. Sorensen, *Deflation techniques for an implicitly restarted Arnoldi iteration*, SIAM Journal on Matrix Analysis and Applications, 17 (1996), pp. 789–821.
[13] R. Li, Y. Xi, E. Vecharynski, C. Yang, and Y. Saad, *A Thick-Restart Lanczos algorithm with polynomial filtering for Hermitian eigenvalue problems*, SIAM Journal on Scientific Computing, 38 (2016), pp. A2512–A2534.
[14] R. B. Morgan, *On restarting the Arnoldi method for large nonsymmetric eigenvalue problems*, Mathematics of Computation, 65 (1996), pp. 1213–1230.
[15] R. B. Morgan, *Implicitly restarted GMRES and Arnoldi methods for nonsymmetric systems of equations*, SIAM Journal on Matrix Analysis and Applications, 21 (2000), pp. 1112–1135.
[16] B. N. Parlett, *The Symmetric Eigenvalue Problem*, vol. 20, SIAM, 1998.
[17] M. Ravibabu and A. Singh, *On refined Ritz vectors and polynomial characterization*, Computers & Mathematics with Applications, 67 (2014), pp. 1057–1064.
[18] Y. Saad, *Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices*, Linear Algebra and its Applications, 34 (1980), pp. 269–295.
[19] H. D. Simon, *Analysis of the symmetric Lanczos algorithm with reorthogonalization methods*, Linear Algebra and its Applications, 61 (1984), pp. 101–131.
[20] G. L. Sleijpen and H. A. Van der Vorst, *A Jacobi–Davidson iteration method for linear eigenvalue problems*, SIAM Review, 42 (2000), pp. 267–293.
[21] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM Journal on Matrix Analysis and Applications, 13 (1992), pp. 357–385.
[22] A. Stathopoulos and J. R. McCombs, *PRIMME: preconditioned iterative multimethod eigensolver methods and software description*, ACM Transactions on Mathematical Software, 37 (2010), p. 21.
[23] A. Stathopoulos and Y. Saad, *Restarting techniques for the (Jacobi–) Davidson symmetric eigenvalue methods*, Electronic Transactions on Numerical Analysis, 7 (1998), pp. 163–181.
[24] A. Stathopoulos, Y. Saad, and K. Wu, *Dynamic thick restarting of the Davidson, and the implicitly restarted Arnoldi methods*, SIAM Journal on Scientific Computing, 19 (1998), pp. 227–245.
[25] G. W. Stewart, *A Krylov–Schur algorithm for large eigenproblems*, SIAM Journal on Matrix Analysis and Applications, 23 (2002), pp. 601–614.
[26] K. Wu and H. Simon, *Dynamic restarting schemes for eigenvalue problems*, tech. report, Lawrence Berkeley National Lab., CA (US), 1999.
[27] K. Wu and H. Simon, *Thick-restart Lanczos method for large symmetric eigenvalue problems*, SIAM Journal on Matrix Analysis and Applications, 22 (2000), pp. 602–616.
[28] L. Wu, F. Xue, and A. Stathopoulos, *TRPL+K: Thick-restart preconditioned Lanczos+ K method for large symmetric eigenvalue problems*, SIAM Journal on Scientific Computing, 41 (2019), pp. A1013–A1040.
[29] Y. Zhou and Y. Saad, *Block Krylov–Schur method for large symmetric eigenvalue problems*, Numerical Algorithms, 47 (2008), pp. 341–359.