Electronic Transactions on Numerical Analysis.
Volume 28, pp. 95-113, 2007.

ISSN 1068-9613.

ETNA
Kent State University
etna@mcs.kent.edu

A REFINED UNSYMMETRIC LANCZOS EIGENSOLVER FOR COMPUTING ACCURATE EIGENTRIPLETS OF A REAL UNSYMMETRIC MATRIX*

JEAN CHRISTOPHE TREMBLAY† AND TUCKER CARRINGTON JR.†

Dedicated to Gene Golub on the occasion of his 75th birthday

Abstract. For most unsymmetric matrices it is difficult to compute many accurate eigenvalues using the primitive form of the unsymmetric Lanczos algorithm (ULA). In this paper we propose a modification of the ULA. It is related to ideas used in [J. Chem. Phys. 122 (2005), 244107 (11 pages)] to compute resonance lifetimes. Using the refined ULA we suggest, the calculation of accurate extremal and interior eigenvalues is feasible. The refinement is simple: approximate right and left eigenvectors computed using the ULA are used to form a small projected matrix whose eigenvalues and eigenvectors are easily computed. There is no re-biorthogonalization of the Lanczos vectors and no need to store large numbers of vectors in memory. The method can therefore be used to compute eigenvalues of very large matrices. The idea is tested on several matrices.

Key words. Eigenproblem, unsymmetric matrices, Lanczos algorithm

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

1. Introduction. Scientists and engineers frequently need to compute many eigenvalues of large real matrices, i.e. to solve

$$\mathbf{G}\mathbf{X}_{(R)} = \mathbf{X}_{(R)}\mathbf{\Lambda}\,, \tag{1.1}$$

where $\mathbf{X}_{(R)}$ is a matrix of right eigenvectors, $\mathbf{x}_{(R)}$, and $\mathbf{\Lambda}$ is a diagonal matrix whose diagonal elements are eigenvalues; see for example [1, 6, 8, 12, 32]. They prefer eigensolvers that require little memory and are simple to implement. Of course, an eigensolver that is CPU efficient is also advantageous. For symmetric matrices the Lanczos procedure without re-orthogonalization is often the best choice, see [7, 9, 10, 14, 27, 28, 29, 31]. Using this procedure it is possible to compute eigenvalues by storing only two vectors. Eigenvectors can be obtained either by re-generating the Lanczos vectors or by storing them on disk. The method is simple and easy to use. For unsymmetric matrices, on the other hand, the known eigensolvers have important deficiencies. For the purpose of computing many eigentriplets (eigenvalues, right eigenvectors $\mathbf{x}_{(R)}$, and left eigenvectors $\mathbf{x}_{(L)}$), the best unsymmetric matrix eigensolvers are based either on the Arnoldi algorithm or on the unsymmetric Lanczos algorithm. In this paper we assume that the matrix $\mathbf{G}$ always has a complete set of eigenvectors.

The Arnoldi algorithm is an orthogonal Krylov projection method [17, 36]. It has the important disadvantage that it requires storing many vectors in memory. Vectors must be stored in memory because every Arnoldi vector is orthogonalized against *all* of the previous computed Arnoldi vectors. It would be very costly to orthogonalize by reading previous Arnoldi vectors from disk because the orthogonalization is done at each iteration. The CPU cost of the orthogonalization also increases with the number of iterations. These problems are to some extent mitigated by using the implicitly restarted Arnoldi (IRA) technique [33]. It is implemented in the ARPACK code which is widely used and appreciated [23, 42].

The unsymmetric Lanczos algorithm (ULA) is an oblique projection method that uses two separate Krylov spaces, for $\mathbf{G}$ and $\mathbf{G}^*$, where $*$ denotes the conjugate transpose [5, 17].

*Received December 22, 2006. Accepted for publication July 19, 2007. Recommended by D. B. Szyld. This work has been supported by the Natural Sciences and Engineering Research Council of Canada and the Fonds québécois de la recherche sur la nature et les technologies.

†Département de chimie, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal (Québec) H3C 3J7, Canada ({jc.tremblay,tucker.carrington}@umontreal.ca).

The ULA has the advantage that its use requires the storage of only four vectors, but the important disadvantage that in its primitive form it does not always yield accurate eigenvalues. The formal memory advantage of the ULA is, of course, useless if one is not able to compute accurate eigentriplets.

Due in part to the inaccuracy of the ULA, the most popular approach for solving the unsymmetric eigenproblem is the Implicitly Restarted Arnoldi (IRA) idea and the ARPACK program [33, 42]. A cleverly designed restarting algorithm contracts a Krylov subspace with p vectors to a Krylov subspace with $k < p$ vectors. The contraction can be done to facilitate the convergence of eigenvalues whose magnitude, or real part, or imaginary part, is largest or smallest [23, 42]. The algorithm is repeatedly restarted until the desired eigenvalues have converged. ARPACK is robust and stable. The memory cost of the method is determined by the need to store p Arnoldi vectors in memory to perform the orthogonalization. Increasing p increases both the memory cost and the CPU cost of the orthogonalization, but decreases the number of necessary restarts. ARPACK has been widely applied and is considered a method of choice [21, 22, 40]. For extremal eigenvalues its most important disadvantage is the need to store p vectors in memory. If the size of the matrix $\mathbf{G}$ is large, one requires a lot of memory in order to use ARPACK. In addition, to reliably obtain interior eigenvalues with ARPACK one must either use a large p (which is usually impractical) or a spectral transformation. The use of ARPACK for interior eigenvalues is discussed in [2, 19]. Using a spectral transformaton is costly, especially if the matrix is so large that it cannot be factorized.

To compute (interior or extremal) eigenvalues of a very large matrix with ARPACK one requires a prohibitive amount of memory and it is therefore tempting to endeavor to improve the ULA. Modifying the ULA, without losing its memory advantage, so that accurate eigenvalues and eigenvectors can be computed, would drastically reduce the memory cost of eigencomputations with very large matrices. The ULA (see section 2) uses two sets of biorthogonal vectors. Several authors have proposed modifications of the ULA. Conceptually, the simplest way to use the ULA to determine accurate eigenvalues is to re-biorthogonalize the two sets of Lanczos vectors. However, it is difficult to achieve biorthogonality that is good enough to ensure accurate eigenvalues. In addition, re-biorthogonalization significantly increases the CPU cost and negates the anticipated memory advantage of the ULA. The look-ahead modification is designed to deal with problems introduced by small normalizing factors [16, 30]. It does improve the accuracy of eigenvalues computed with the ULA and can be used with any of the other modifications described below. Day [13] has proposed a scheme for maintaining semi-biorthogonality. He shows that using this approach it is possible to compute eigenvalues accurate to “from full to half relative precision” but, to make the method work, all Lanczos vectors must either be stored in memory or read from disk at each re-biorthogonalization step. This significantly increases the memory cost or the calculation time. In [11], Cullum and Willoughby show that if the coefficients of the recursion relations are chosen so that the tridiagonal matrix is symmetric (but complex) accurate eigenvalues can be computed for some test problems. This is a low-memory method and it is simple to use but requires complex arithmetic. Using complex Lanczos vectors increases the memory and CPU costs by about a factor of two.

In this paper we show that it is possible to use Lanczos vectors to compute accurate eigenvalues and eigenvectors without storing vectors spanning Krylov space(s) in memory (we do no re-biorthogonalization of the Lanczos vectors). It is also possible to compute interior eigenvalues. To compute eigenvalues of a real matrix, we evaluate matrix-vector products in real arithmetic. The new idea is simple: we use approximate right and left eigenvectors obtained from the ULA to build a projected matrix which we diagonalize using a standard method. In effect we are using the ULA to compute vectors that have large components along

ETNA
Kent State University
etna@mcs.kent.edu

the eigenvector directions. The eigenvectors obtained from the ULA are not accurate, but they can be used to compute accurate eigenvalues and eigenvectors. We call this approach the refined unsymmetric Lanczos eigensolver (RULE). Although the word refined occurs in the name, the method is not directly related to that of [20]. It is built upon ideas introduced in [37].

The RULE's principle advantage is that it enables one to compute many eigenvalues of a large matrix without storing more than about a dozen vectors in memory. The number of eigenvalues (and eigenvectors) one can compute, from one tridiagonal matrix, is much larger than the number of vectors that one must store in memory. The RULE is a good alternative to ARPACK if the matrix for which eigenvalues are desired is so large that it is not possible to store many vectors in memory, or if one wishes interior eigenvalues. It makes it possible to extract accurate eigenvalues from large Krylov spaces. Although our numerical experiments (see section 4) show that the RULE works well for many difficult problems it is not backward stable. Nevertheless, it provides a means of computing eigentriplets of matrices for which the memory ARPACK requires exceeds that of the computer.

2. The standard unsymmetric Lanczos algorithm. The unsymmetric Lanczos algorithm is an iterative process for generating two sets of biorthogonal vectors, $\{\mathbf{v}_j\}_{j=1}^m$ and $\{\mathbf{w}_j\}_{j=1}^m$, which span two Krylov subspaces [5, 17],

$$\begin{aligned} K^{(m)}\{\mathbf{G},\mathbf{v}_1\} &= \mathrm{span}\,\{\mathbf{v}_1,\mathbf{G}\mathbf{v}_1,\ldots,\mathbf{G}^{\mathbf{m-1}}\mathbf{v}_1\}\,, \quad \text{and} \\ K^{(m)}\{\mathbf{G}^*,\mathbf{w}_1\} &= \mathrm{span}\,\{\mathbf{w}_1,\mathbf{G}^*\mathbf{w}_1,\ldots,(\mathbf{G}^*)^{\mathbf{m-1}}\mathbf{w}_1\}\,. \end{aligned}$$

The two sets of Lanczos vectors may be obtained from the two three-term recurrence relations

$$\begin{aligned} \rho_{k+1}\mathbf{v}_{k+1} &= \mathbf{r}_{k+1} = \mathbf{G}\mathbf{v}_k - \alpha_k\mathbf{v}_k - \gamma_k\mathbf{v}_{k-1}\,, \\ \xi_{k+1}^*\mathbf{w}_{k+1} &= \mathbf{s}_{k+1} = \mathbf{G}^*\mathbf{w}_k - \alpha_k^*\mathbf{w}_k - \frac{\rho_k^*\delta_k}{\delta_{k-1}}\mathbf{w}_{k-1}\,, \end{aligned}$$

where $\gamma_k = \xi_k\delta_k/\delta_{k-1}$, $\delta_k = \mathbf{w}_k{}^*\mathbf{v}_k$, $\alpha_k = \mathbf{w}_k{}^*\mathbf{G}\mathbf{v}_k/\delta_k$, and ρ_k and ξ_k normalize $\mathbf{r}_k$ and $\mathbf{s}_k$, respectively. The procedure is started by choosing two random starting vectors ($\mathbf{v}_1$ and $\mathbf{w}_1$) and by setting $\gamma_1 = \rho_1 = 0$. In matrix form these relations are written

$$\begin{aligned} \mathbf{G}\mathbf{V}_m &= \mathbf{V}_m\mathbf{T}_m + \rho_{m+1}\mathbf{v}_{m+1}\mathbf{e}_m^*\,, \\ \mathbf{G}^*\mathbf{W}_m &= \mathbf{W}_m\mathbf{\Delta}_m^{-1}\mathbf{T}_m^*\mathbf{\Delta}_m + \xi_{m+1}^*\mathbf{w}_{m+1}\mathbf{e}_m^*\,, \end{aligned}$$

where $\mathbf{V}_m$ and $\mathbf{W}_m$ are matrices whose columns are the first m vectors $\mathbf{v}_k$ and $\mathbf{w}_k$, respectively, and $\mathbf{\Delta}_m = \mathrm{diag}(\delta_1,\delta_2,\ldots,\delta_m)$. The scalars α_k, ρ_k, and γ_k are elements of the tridiagonal matrix $\mathbf{T}_m = \mathbf{\Delta}_m^{-1}\mathbf{W}_m^*\mathbf{G}\mathbf{V}_m$,

$$\mathbf{T}_m = \begin{pmatrix} \alpha_1 & \gamma_2 & & & \\ \rho_2 & \alpha_2 & \gamma_3 & & \\ & \rho_3 & \alpha_3 & \ddots & \\ & & \ddots & \ddots & \gamma_m \\ & & & \rho_m & \alpha_m \end{pmatrix}.$$

Eigenvalues of the matrix $\mathbf{T}_m$ are computed by solving

$$\mathbf{T}_m\mathbf{Z} = \mathbf{Z}\mathbf{\Theta}_m\,.$$

Some of the eigenvalues of $\mathbf{T}_m$ approximate eigenvalues of $\mathbf{G}$. There is some flexibility in the definition of ρ_k and ξ_k. We choose to normalize so that $\delta_k = \mathbf{w}_k{}^*\mathbf{v}_k = 1$ and

$\mathbf{\Delta}_m^{-1}\mathbf{T}_m^*\mathbf{\Delta}_m = \mathbf{T}_m^*$. This choice will be assumed for the remainder of the paper. In pseudocode the algorithm is:

Algorithm 1. Unsymmetric Lanczos

Choose $\mathbf{v}_1$ and $\mathbf{w}_1$ such that $\mathbf{w}_1^*\mathbf{v}_1 = 1$;
$\mathbf{r} = \mathbf{G}\mathbf{v}_1$;
$\mathbf{s} = \mathbf{G}^*\mathbf{w}_1$;
for $k = 1, 2, \ldots, m$
  $\alpha_k = \mathbf{w}_k^*\mathbf{r}$;
  $\mathbf{r} = \mathbf{r} - \alpha_k\mathbf{v}_k$;
  $\mathbf{s} = \mathbf{s} - \alpha_k^*\mathbf{w}_k$;
  $\mathbf{r} = \mathbf{r} - (\mathbf{r}^*\mathbf{w}_k)\,\mathbf{v}_k$;
  $\mathbf{s} = \mathbf{s} - (\mathbf{s}^*\mathbf{v}_k)\,\mathbf{w}_k$;
  if ($\|\mathbf{r}\| = 0$ or $\|\mathbf{s}\| = 0$), stop;
  $\tilde{\delta}_k = \mathbf{r}^*\mathbf{s}$;
  if ($\tilde{\delta}_k = 0$), stop;
  $\rho_{k+1} = |\tilde{\delta}_k|^{1/2}$;
  $\gamma_{k+1} = \tilde{\delta}_k^{\,*}/\rho_{k+1}$;
  $\mathbf{v}_{k+1} = \mathbf{r}/\rho_{k+1}$;
  $\mathbf{w}_{k+1} = \mathbf{s}/\gamma_{k+1}^*$;
  $\mathbf{r} = \mathbf{G}\mathbf{v}_{k+1} - \gamma_{k+1}\mathbf{v}_k$;
  $\mathbf{s} = \mathbf{G}^*\mathbf{w}_{k+1} - \rho_{k+1}^*\mathbf{w}_k$;
end.

Breakdown or near breakdown can occur when $\tilde{\delta}_k \simeq 0$. However, in our experience, it never occurred and we shall not discuss it further. Breakdown can be dealt with using look-ahead methods [16, 30].

As the number of iterations increases, roundoff errors will ineluctably lead to loss of biorthogonality of the Lanczos vectors. When an eigenvalue converges, the corresponding eigenvectors will contaminate successive Lanczos vectors and near copies of converged eigenvalues will be generated. Increasing the size of the Krylov subspace will increase the number of near copies. For the Hermitian case this phenomenon was first explained by Paige [27, 28, 29]. In that case, if nothing is done to minimize the loss of orthogonality, multiple copies of converged eigenvalues appear. Cullum and Willoughby denoted unconverged eigenvalues of the matrix $\mathbf{T}_m$ "spurious" eigenvalues and developed a practical procedure for identifying the spurious eigenvalues of a matrix $\mathbf{T}_m$ that is Hermitian [9, 10]. Copies of converged eigenvalues are equal to machine precision and cause no problem. For non-Hermitian matrices the "copies" are not nearly as close and it is harder to distinguish between spurious and good eigenvalues. To a large degree this problem can be overcome by choosing the coefficients in the recursion relations so that the tridiagonal matrix is complex symmetric [11], but to minimize CPU and memory costs we wish to use real arithmetic. In real arithmetic and using the ULA: (1) in general, it is not possible to find a Krylov subspace size m for which an eigenvalue of the matrix $\mathbf{T}_m$ accurately approximates an eigenvalue of the matrix $\mathbf{G}$; (2) for each nearly converged eigenvalue of the matrix $\mathbf{G}$ there is a cluster of closely spaced eigenvalues of $\mathbf{T}_m$, but the norm of differences between eigenvalues in the cluster may be so large that it is difficult to identify the cluster and impossible to determine an accurate value for the eigenvalue; (3) the width of the clusters increases with m; and (4) it is not possible to use the procedure Cullum and Willoughby developed to remove spurious eigenvalues from clumps of eigenvalues. The inadequacy of the ULA is well known [3, 4, 5, 39].

ETNA
Kent State University
etna@mcs.kent.edu

Consider the following example. We use Matlab 7 commands to create a random matrix of size $N = 300$

$$\begin{aligned} &\text{RAND('SEED',0);} \\ &\mathbf{G}_{ext} = \text{RAND}(N) \ . \end{aligned} \tag{2.1}$$

Eigenvalues computed with Matlab's EIG routine are shown in Figure 2.1.

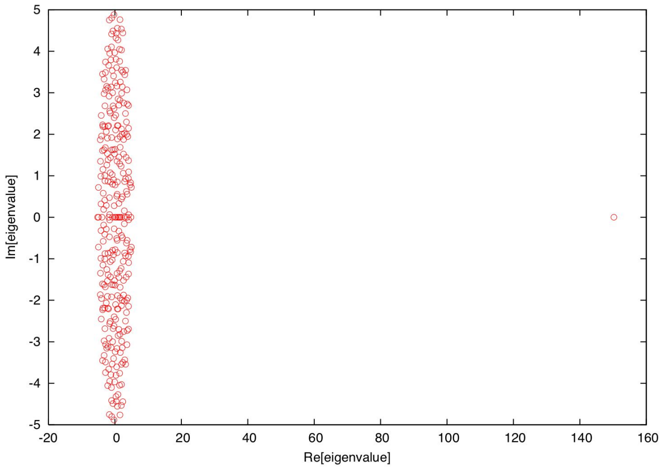

FIG. 2.1. *Distribution of eigenvalues of the matrix G_{ext}.*

One eigenvalue is well isolated. It is the eigenvalue of largest magnitude and is real. Its condition number,

$$c_\lambda = \frac{1}{|\mathbf{x}_{(L)}{}^* \mathbf{x}_{(R)}|} \, ,$$

is 1.001. The ULA with $m = 300$ produces many approximate copies of this eigenvalue, see Table 2.1. Even for this isolated well-conditioned eigenvalue, the imperfect copies makes it hard to extract accurate eigenvalues from a large tridiagonal matrix.

Figure 2.2 shows the evolution of the error associated with the isolated eigenvalue. The error is defined as the the norm of the difference between the exact (computed with Matlab's EIG) eigenvalue and the average of the near copies in a cluster generated by the ULA. Two eigenvalues θ_j and θ_k are deemed to be in a cluster if

$$|\theta_k - \theta_j| \leq \eta \max(|\theta_k|, |\theta_j|) \, , \tag{2.2}$$

where η is a user defined tolerance [4]. For the matrix $\mathbf{G}_{ext}$, with the best possible value of m, the eigenvalue accuracy is quite good (14 significant digits). The best possible value of m can be determined by using the procedure outlined by Cullum and Willoughby for the complex symmetric Lanczos algorithm [10]. The error associated with the average of the eigenvalues of the cluster eigenvalues for $m = 300$ is quite large. Nevertheless with some effort (i.e., by finding the right m) the ULA can be used to compute an isolated extremal eigenvalue.

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 2.1
Selected eigenvalues of the 300 × 300 tridiagonal matrix that correspond to the isolated eigenvalue and an interior eigenvalue of the matrix $\mathbf{G}_{ext}$. The eigenvalues of matrix $\mathbf{G}_{ext}$ obtained with Matlab's EIG *are* $\lambda_1 = 150.234556208130$ *and* $\lambda_2 = 3.609257224171 + 3.069961788177i$.

Approximate eigenvalue	Error
150.234556208117	1.2989E-11
150.234556208121	8.9813E-12
150.234556208126	3.9790E-12
150.234556208129	9.9476E-13
150.234556208129	9.9476E-13
150.234556209596	1.4660E-09
150.234556220968	1.2838E-08
150.234556241745	3.3615E-08
150.234556268333	6.0203E-08
150.234556296133	8.8003E-08
150.234556320339	1.1221E-07
150.234556336769	1.2864E-07
$3.609257227317 + 3.069961798318i$	1.0617E-08
$3.609257246703 + 3.069961858171i$	7.3531E-08
$3.609256887624 + 3.069962297111i$	6.1014E-07
$3.609254779439 + 3.069965061303i$	4.0854E-06

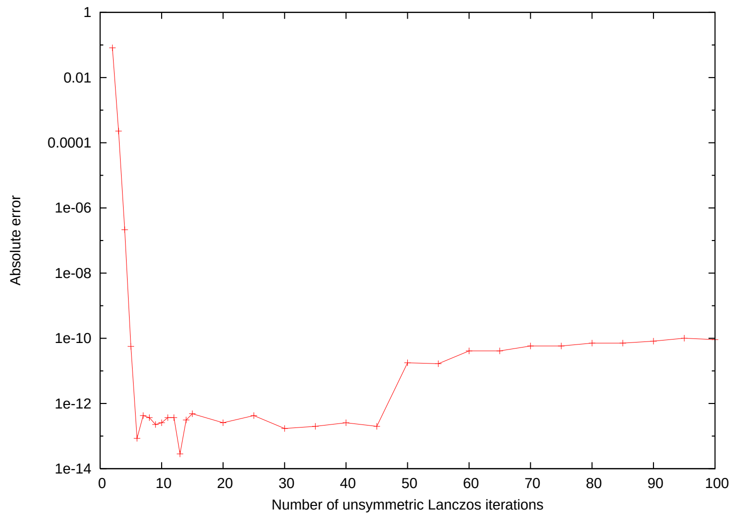

FIG. 2.2. *Error in an extremal extremal, isolated eigenvalue of the matrix $\mathbf{G}_{ext}$ computed with the ULA.*

Unfortunately, it is much harder to compute an eigenvalue that is not isolated (cf. Table 2.1). For interior eigenvalues all of the copies are further than 10^{-8} from the exact eigenvalues and it is impossible to find a tridiagonal matrix for which eigenvalues are very close to the desired exact eigenvalues of the matrix $\mathbf{G}_{ext}$. We find that using a simple Gram-Schmidt re-biorthogonalization is not good enough to get accurate interior eigenvalues (although it does reduce the number of matrix-vector products required to achieve an error of $\sim 10^{-8}$). We shall now show that approximate right and left eigenvectors, obtained from the ULA with a single value of m, *can* be used to obtain many accurate eigenvalues.

ETNA
Kent State University
etna@mcs.kent.edu

3. Refined unsymmetric Lanczos eigensolver. As briefly explained at the end of the introduction we use approximate right and left eigenvectors obtained from the ULA to determine accurate eigenvalues and eigenvectors of a general matrix $\mathbf{G}$. To do this we must store Lanczos vectors on disk (they are not accessed during the iteration and we do not re-biorthogonalize them) and calculate eigenvalues in groups. Because one typically needs only a small fraction of the eigenvalues of a huge matrix a small number of groups will be required. Although the RULE is designed for computing only a small fraction of the total number of eigenvalues, it may easily be used to calculate hundreds of eigenvalues. If a smaller number of eigenvalues is desired a Jacobi-Davidson algorithm would be a good alternative [15, 18].

We use the ULA with $\mathbf{v}_1 = \mathbf{w}_1 =$ a "random" vector to compute an unsymmetric tridiagonal matrix $\mathbf{T}_m$ for a large value of m. We transform the matrix $\mathbf{T}_m$ to obtain a complex symmetric tridiagonal matrix with the same eigenvalues,

$$\hat{\mathbf{T}}_m = \mathbf{\Omega}^{-\frac{1}{2}}\mathbf{T}_m\mathbf{\Omega}^{\frac{1}{2}} \tag{3.1}$$
$$= \begin{pmatrix} \alpha_1 & (\rho_2\gamma_2)^{\frac{1}{2}} & & & \\ (\rho_2\gamma_2)^{\frac{1}{2}} & \alpha_2 & (\rho_3\gamma_3)^{\frac{1}{2}} & & \\ & (\rho_3\gamma_3)^{\frac{1}{2}} & \alpha_3 & \ddots & \\ & & \ddots & \ddots & (\rho_m\gamma_m)^{\frac{1}{2}} \\ & & & (\rho_m\gamma_m)^{\frac{1}{2}} & \alpha_m \end{pmatrix},$$

where $\mathbf{\Omega} = \text{diag}(1, \frac{\rho_2\omega_1}{\gamma_2}, \frac{\rho_3\omega_2}{\gamma_3}, \ldots, \frac{\rho_m\omega_{m-1}}{\gamma_m})$ and ω_j is the j'th diagonal element of $\mathbf{\Omega}$. The matrix $\mathbf{\Omega}$ is obtained recursively. In a FORTRAN implementation of the RULE, eigenvalues of the symmetric tridiagonal matrix $\hat{\mathbf{T}}_m$ are computed using the CMTQL1 routine Cullum and Willoughby employ in their complex symmetric and unsymmetric Lanczos procedures [10]. We prefer to use the symmetric tridiagonal matrix $\hat{\mathbf{T}}_m$ and CMTQL1 because CMTQL1 exploits the symmetry to reduce both the CPU cost and the memory cost of calculating the eigenvalues. Our Matlab RULE routine simply uses EIG to diagonalize the matrix $\hat{\mathbf{T}}_m$. In a region of the spectrum in which we wish eigenvalues of the matrix $\mathbf{G}$ we form clusters of eigenvalues of the matrix $\hat{\mathbf{T}}_m$. To do so we build a list of the eigenvalues θ_i, $i = 1, \ldots, m$. For one eigenvalue in the list, θ_k, we compute $|\theta_k - \theta_j|$ for all other eigenvalues in the list. Eigenvalues θ_j in the same cluster, see (2.2), are removed from the list before making the cluster associated with another eigenvalue. Clusters with more than one eigenvalue contain near copies introduced by round-off error. If there are no near copies a cluster will contain only one eigenvalue; many clusters do only have one eigenvalue. A procedure for identifying spurious eigenvalues in the symmetric case was developed by Cullum and Willoughby [9, 10]. We use similar ideas to identify spurious eigenvalues for the unsymmetric problem. A complex symmetric tridiagonal matrix $\mathbf{T}_2$ is formed by removing the first column and the first row of matrix $\hat{\mathbf{T}}_m$. The matrix $\mathbf{T}_2$ is then diagonalized to obtain its eigenvalues, $\tilde{\theta}_i$. Isolated eigenvalues (i.e. one-eigenvalue clusters) of matrix $\hat{\mathbf{T}}_m$ that differ from isolated eigenvalues of matrix $\mathbf{T}_2$ by less than $\eta \max(|\tilde{\theta}_i|, |\theta_j|)$ are identified as spurious. Doing the Cullum and Willoughby spurious test enables us to avoid calculating approximate eigenvectors that correspond to the eigenvalues of matrix $\mathbf{T}_m$ that are not close to any of the eigenvalues of matrix $\mathbf{G}$, without excluding vectors whose eigenvalues are close to those of the matrix $\mathbf{G}$. For each non-spurious cluster we compute an average eigenvalue. These average eigenvalues are used as shifts with inverse iteration [41, pp. 619–633], to determine approximate right and left eigenvectors of the matrix $\mathbf{T}_m$ (one pair for each cluster). The inverse iteration algorithm we use is:

ETNA
Kent State University
etna@mcs.kent.edu

Algorithm 2. Two-sided inverse iteration

Choose random starting vectors $\mathbf{z}_r^j$ and $\mathbf{z}_l^j$;
Factorize $\mathbf{LU} = (\hat{\lambda}_j \mathbf{I} - \mathbf{T}_m)$;
for $i = 1, 2, \ldots$
  Solve $\mathbf{LU}\tilde{\mathbf{z}} = \mathbf{z}_r^j$;
  $\mathbf{z}_r^j = \tilde{\mathbf{z}}/\|\tilde{\mathbf{z}}\|$;
  Solve $\mathbf{U}^*\mathbf{L}^*\tilde{\mathbf{z}} = \mathbf{z}_l^j$;
  $\mathbf{z}_l^j = \tilde{\mathbf{z}}/\|\tilde{\mathbf{z}}\|$;
  Evaluate $\theta_j^{(i)} = \frac{\mathbf{z}_l^{j\,*}\mathbf{T}_m \mathbf{z}_r^j}{\mathbf{z}_l^{j\,*}\mathbf{z}_r^j}$;
  Exit if converged;
end.

Here, $\hat{\lambda}_j$ is an approximate eigenvalue or average eigenvalue obtained from the diagonalization of the matrix $\hat{\mathbf{T}}_m$, and $\mathbf{z}_r^j$ and $\mathbf{z}_l^j$ are the associated right and left eigenvectors of matrix $\mathbf{T}_m$. The Ritz value $\theta_j^{(i)}$ is used to monitor the convergence of the j^{th} pair of eigenvectors. Inverse iteration is terminated at $i = m_j$, if $\theta_j^{(i)}$ and $\theta_j^{(i-1)}$ differ by less than $10^{-13}|\hat{\lambda}_j|$. Typically five or fewer inverse iteration steps are required for each converged $\{\mathbf{z}_r^j, \mathbf{z}_l^j\}$ pair. There are alternatives to inverse iteration. Any method with which one can exploit the tridiagonal structure of the matrix $\mathbf{T}_m$ could be used, e.g., shift and invert Arnoldi could be used because the factorization of a tridiagonal matrix can be easily computed and stored.

After having computed, for each eigenvalue or average value, eigenvectors of the matrix $\mathbf{T}_m$ we determine approximate left and right eigenvectors of the matrix $\mathbf{G}$ by reading Lanczos vectors $\mathbf{v}_k$ and $\mathbf{w}_k$ from disk and combining them according to

$$\mathbf{r}_j = \sum_{k=1}^{m} \mathbf{z}_r^j(k)\mathbf{v}_k \,, \quad \mathbf{l}_j = \sum_{k=1}^{m} \mathbf{z}_l^j(k)\mathbf{w}_k \,, \tag{3.2}$$

where $\mathbf{r}_j$ and $\mathbf{l}_j$ are approximate jth right and left eigenvectors of matrix $\mathbf{G}$, and $\mathbf{z}_r^j(k)$ ($\mathbf{z}_l^j(k)$) is the kth component of $\mathbf{z}_r^j$ ($\mathbf{z}_l^j$). If η is small it can happen that several $\mathbf{r}_j$ and $\mathbf{l}_j$ vectors are nearly parallel. This occurs if several of the clusters contain eigenvalues of the matrix $\mathbf{T}_m$ associated with one exact eigenvalue of the matrix $\mathbf{G}$. To minimize the near linear dependance of the vectors $\mathbf{r}_j$ and $\mathbf{l}_j$ and to reduce the number of the $\{\mathbf{z}_r^j, \mathbf{z}_l^j\}$ pairs we retain only pairs for which Ritz values obtained from inverse iteration differ by more than η max$(|\theta_j^{m_j}|, |\theta_k^{m_k}|)$. The right eigenvectors $\mathbf{r}_j,\ j = 1, \ldots, k$, where k is the number of retained approximate eigenvectors, are the columns of a matrix we denote $\mathbf{R}_k$. The vectors $\mathbf{l}_j, j = 1, \ldots, k$, are the columns of a matrix we denote $\mathbf{L}_k$. One might also avoid nearly parallel vectors by increasing η, but if η is too large, eigenvalues of the matrix $\mathbf{T}_m$ associated with two or more exact eigenvalues will be lumped into a single cluster and one will miss eigenvalues. One could actually use the RULE with $\eta = 0$ (i.e. cluster width of zero). This would eliminate any possibility of missing closely-spaced eigenvalues, but it would increase the number of $\{\mathbf{z}_r^j, \mathbf{z}_l^j\}$ pairs to compute, and increase the number of vectors $\mathbf{r}_j$ and $\mathbf{l}_j$ to assemble and therefore make the calculation more costly.

To numerically solve (1.1) we replace

$$\mathbf{X}_{(R)} \simeq \hat{\mathbf{X}}_{(R)} = \mathbf{R}_k \mathbf{Y}_{(R)} \,. \tag{3.3}$$

This yields,

$$\mathbf{G}\mathbf{R}_k \mathbf{Y}_{(R)} = \mathbf{R}_k \mathbf{Y}_{(R)} \tilde{\mathbf{\Lambda}} \,,$$

ETNA
Kent State University
etna@mcs.kent.edu

where $\tilde{\mathbf{\Lambda}}$ is an approximation for $\mathbf{\Lambda}$ in (1.1). After multiplying both sides from the left by the conjugate transpose of $\mathbf{L}_k$, we obtain the generalized eigenvalue problem,

$$\mathbf{G}_k \mathbf{Y}_{(R)} = \mathbf{S}_k \mathbf{Y}_{(R)} \tilde{\mathbf{\Lambda}} \,, \tag{3.4}$$

where $\mathbf{G}_k = \mathbf{L}_k^* \mathbf{G} \mathbf{R}_k$ and $\mathbf{S}_k = \mathbf{L}_k^* \mathbf{R}_k$. The matrix $\mathbf{S}_k$ is an overlap matrix of two sets of vectors. This oblique Rayleigh-Ritz-type procedure works because the better the approximate eigenvector spaces, the better the eigenvalues obtained from the projected matrix [34, pp. 280–289]. For experiments we have done, the eigenvalues of (3.4) are accurate eigenvalues of $\mathbf{G}$ and accurate right eigenvectors can be obtained using (3.3). A matrix of approximate left eigenvectors, $\mathbf{X}_{(L)}$, is obtained from

$$\mathbf{X}_{(L)}^* \simeq \hat{\mathbf{X}}_{(L)}^* = \mathbf{Y}_{(L)}^* \mathbf{L}_k^* \,, \tag{3.5}$$

where the matrix $\mathbf{Y}_{(L)}$ is defined by

$$\mathbf{Y}_{(L)}^* \mathbf{G}_k = \tilde{\mathbf{\Lambda}} \mathbf{Y}_{(L)}^* \mathbf{S}_k \,,$$

and is computed simultaneously with the matrix of right eigenvectors $\mathbf{Y}_{(R)}$. If one wishes more eigenvalues it is necessary to compute additional approximate eigenvectors (from the same Lanczos vectors) and to form another matrix $\mathbf{G}_k$. Using the approximate eigenvectors to build the matrix $\mathbf{G}_k$ enables us to extract accurate eigenvalues from the Krylov spaces, but it does not change the subspaces themselves. The convergence rate of the RULE will be determined by the Krylov subspaces of the ULA which are to some extent affected by roundoff error.

The new method can be summarized as follows:

Algorithm 3. Refined Unsymmetric Lanczos Eigensolver (RULE)

1. Compute m steps of the Unsymmetric Lanczos Algorithm 1;
2. Form the matrix $\hat{\mathbf{T}}_m$ in (3.1) and compute its eigenvalues;
3. Form clusters of eigenvalues of $\hat{\mathbf{T}}_m$ and for each cluster compute an average;
4. Identify clusters with only one eigenvalue that are spurious;
5. Select shifts for non-spurious eigenvalue clusters in the spectral region of interest;
6. For each shift compute right and left eigenvectors of $\mathbf{T}_m$ using inverse iteration Algorithm 2;
7. Keep only eigenvectors of $\mathbf{T}_m$ with distinct Ritz values;
8. Assemble approximate left and right eigenvectors of matrix $\mathbf{G}$ according to (3.2);
9. Project matrix $\mathbf{G}$ onto these vectors according to (3.4);
10. Solve the complex generalized eigenvalue problem using the QZ algorithm [5].

We choose a value of m and generate Lanczos vectors. If the associated matrix $\hat{\mathbf{T}}_m$ has eigenvalues in the spectral region of interest, the matrix $\mathbf{G}_k$ is formed and (3.4) is solved. If the matrix $\hat{\mathbf{T}}_m$ does not have eigenvalues in the spectral region of interest or if the eigenvalues computed with the RULE are not converged, we increase m and calculate more Lanczos vectors. Increasing m cannot ruin the approximate eigenvectors so there is no need to carefully search for the best m. To obviate the need to store a large number of approximate eigenvectors in memory, eigenvalues are calculated in groups by repeating steps 5 to 10 of Algorithm 3. To compute eigenvalues with the largest/smallest, real/imaginary part/norm we use approximate eigenvectors whose corresponding eigenvalues are those with the largest/smallest, real/imaginary part/norm. To calculate interior eigenvalues close to a target value we use approximate eigenvectors associated with the eigenvalues close to the target.

ETNA
Kent State University
etna@mcs.kent.edu

4. Numerical experiments. Many numerical experiments were done using both Matlab and FORTRAN programs. With Matlab we have computed extremal eigenvalues of "random" and Riemann matrices generated using Matlab routines and of Tolosa matrices from the Matrix Market [43]. Also using Matlab we have computed interior eigenvalues of PDE matrices from the Matrix Market. Eigentriplets of the AIRFOIL matrix, available from Matrix Market, and a matrix we constructed to determine energies and lifetimes of metastable states of the HCO radical were computed using a FORTRAN implementation of the RULE. In all cases η was chosen to be the square root of the machine precision.

4.1. Random matrix generated with (2.1). The random matrices of different size, obtained using (2.1), all had one isolated, extremal eigenvalue. In section 2 it is mentioned that it is possible to use the ULA to obtain a good approximation for the isolated eigenvalue if and only if a tridiagonal matrix of the optimal size is used. The smallest error that can be achieved, by choosing the best possible value of m, with the ULA is of the order of 10^{-13}. For other (non isolated) eigenvalues, even with an optimal m, the ULA does poorly. Interior eigenvalue errors are of the order of 10^{-8}. Both ARPACK and our implementation of the RULE yield extremal and non-extremal eigenvalues with absolute errors less than 10^{-13} and there is no need to select an optimal value of m.

4.2. Riemann matrices. We have computed eigenvalues of different Riemann matrices, the largest of which is of size $N = 5000$. The matrices are associated with the Riemann hypothesis and can be generated with Matlab using the function

$$\mathbf{G}_{rie} = \text{GALLERY}(\text{'RIEMANN'}, N)\,.$$

For more details, the reader is referred to the Matrix Computation Toolbox available on the website of Professor N. Higham [44]. The complex eigenvalue distribution for the matrix is shown in Figure 4.1.

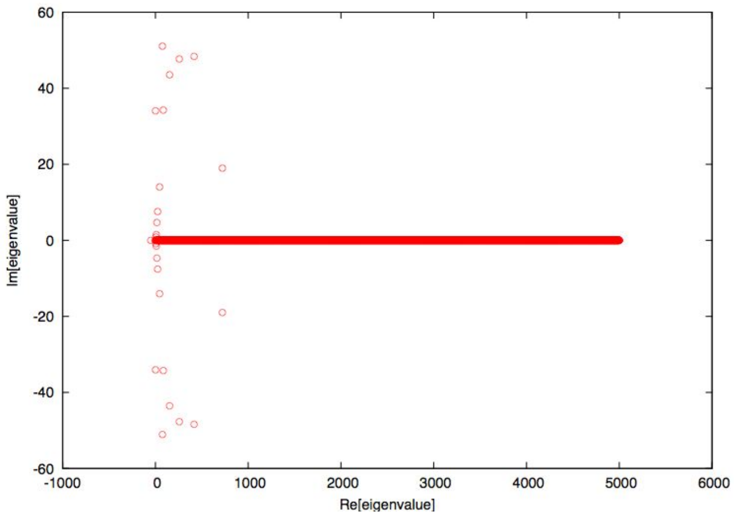

FIG. 4.1. *Distribution of eigenvalues of the RIEMANN5000 matrix.*

We decided to focus on eigenvalues with large imaginary parts since they are located near the convex hull of the spectrum but do not have the largest norms. Due to the fact that the desired eigenvalues with large imaginary parts are close to the convex hull the ULA should work

ETNA
Kent State University
etna@mcs.kent.edu

fairly well [5]. The convergence behavior of a typical eigenvalue computed with the ULA is similar to that shown in Figure 2.2. Although the ULA is expected to work well, the minimum error is greater than 10^{-9} for all eigenvalues reported in Table 4.1. The error, defined as the norm of the difference between the average of the nearly degenerate copies and the eigenvalue computed with Matlab's EIG, increases as m is increased further. The 12 eigenvalues with the largest absolute imaginary parts computed with the RULE are presented in Table 4.1. Note that the refinement decreases the eigenvalue error by many orders of magnitude. Some errors are reduced by 4 orders of magnitude.

TABLE 4.1
Error in selected eigenvalues of the RIEMANN5000 matrix.

Eigenvalue	ULA*	RULE**	ARPACK†	$\lvert\mathbf{x}^*_{(L)}\mathbf{x}_{(R)}\rvert$
$2.024454 - 34.08310i$	3.3E-9	4.5E-11	7.4E-11	5.9558E-03
$2.024454 + 34.08310i$	3.3E-9	4.2E-11	7.4E-11	5.9558E-03
$84.80854 - 34.24698i$	9.4E-8	1.3E-10	9.0E-11	9.2899E-03
$84.80854 + 34.24698i$	9.4E-8	1.5E-10	9.0E-11	9.2899E-03
$76.12058 - 51.07108i$	1.5E-8	3.0E-10	2.3E-10	5.7480E-03
$76.12058 + 51.07108i$	1.5E-8	2.9E-10	2.3E-10	5.7480E-03
$152.9928 - 43.53188i$	4.3E-8	4.5E-11	2.4E-11	4.1720E-02
$152.9928 + 43.53188i$	4.3E-8	4.6E-11	2.4E-11	4.1720E-02
$257.0954 - 47.71716i$	6.4E-7	3.4E-11	3.0E-11	8.9580E-02
$257.0954 + 47.71716i$	6.4E-7	3.2E-11	3.0E-11	8.9580E-02
$417.5244 - 48.37068i$	4.5E-8	3.3E-11	2.3E-11	1.5945E-01
$417.5244 + 48.37068i$	4.5E-8	3.5E-11	2.3E-11	1.5945E-01

* 475 Lanczos iterations; 950 matrix-vector products
** 475 Lanczos iterations; 962 matrix-vector products
† ARPACK parameters: k = 12, p=150, 7 restarts, 1116 matrix-vector products

In the fourth column of Table 4.1 we report results obtained with ARPACK. Eigenvalues computed with ARPACK are also accurate. For some matrices eigenvalues computed with the RULE will be more accurate than those computed with ARPACK, for others the reverse will be true. We do not claim that eigenvalues computed with the RULE will always be more accurate. Nonetheless the RULE has the important advantage that it requires much less memory. Is this advantage obtained by sacrificing efficiency (i.e. increased CPU cost)?

The efficiency of the RULE can be assessed on the basis of the number of matrix-vector products required to determine accurate eigenvalues (doing the matrix-vector products is the most time-consuming part of the calculation). Although the RULE requires matrix-vector products with both $\mathbf{G}^*$ and $\mathbf{G}$ and ARPACK needs only matrix-vector products with $\mathbf{G}$ the number of RULE matrix-vector products (962) required is similar to the number of required ARPACK matrix-vector products (1116). Over and above those required to do the 475 Lanczos iterations the RULE requires only 12 matrix-vector products. The additional 12 are necessary for building the matrix $\mathbf{G}_k$. The cost of an ARPACK calculation includes not only the cost of doing the matrix-vector products but also the cost of the orthogonalization. We have chosen the ARPACK parameter p to approximately minimize the number of matrix-vector products. With this choice the number of ARPACK and RULE matrix-vector products is similar but the memory cost of the RULE is much less. It would have been possible to choose a smaller p to reduce the memory cost of ARPACK, but this would increase the number of restarts and the number of matrix-vector products (increasing p decreases the CPU cost but increases the memory cost). These results show that, for this problem, the RULE is not only

ETNA
Kent State University
etna@mcs.kent.edu

accurate but also efficient in terms of CPU time.

4.3. TOLOSA matrices. The TOLOSA matrices were used in the stability analysis of an airplane in flight. The eigenvalues of interest are the three with the largest imaginary parts. For more detail we refer the reader to the Matrix Market [43]. The desired eigenvalues are located on the convex hull of the spectrum (cf. Figure 4.2).

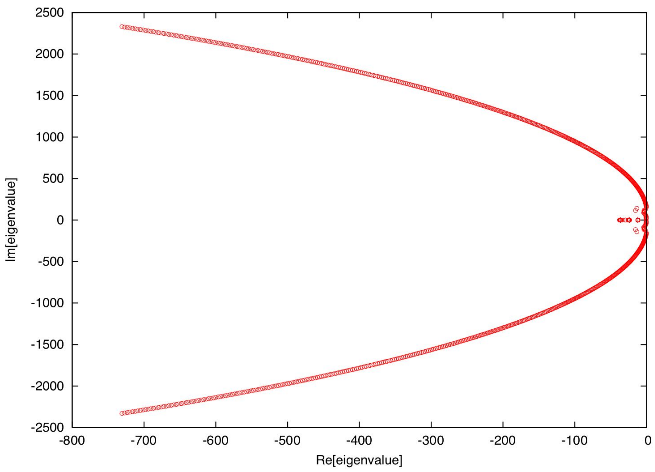

FIG. 4.2. *Distribution of eigenvalues of the TOLS2000 matrix.*

Matrices of size 90, 340, 1090 and 2000 were studied. Using both the RULE and ARPACK it is easy to converge many eigenvalues of the three smaller matrices. We therefore focus on the largest matrix. Table 4.2 shows eigenvalue errors. The accuracy of the eigenvalues computed with the ULA for TOLS2000 is poor.

TABLE 4.2
Error in extremal eigenvalues of the TOLS2000 matrix.

Eigenvalue	ULA*	RULE**	ARPACK†	$\lvert\mathbf{x}^*_{(L)}\mathbf{x}_{(R)}\rvert$
$-723.2940 - 2319.859i$	5.4E-05	6.3E-07	2.5E-09	7.8574E-04
$-723.2940 + 2319.859i$	5.4E-05	6.3E-07	2.5E-09	7.8574E-04
$-726.9866 - 2324.992i$	4.8E-06	8.8E-11	1.4E-07	7.8360E-04
$-726.9866 + 2324.992i$	4.8E-06	8.5E-11	1.4E-07	7.8360E-04
$-730.6886 - 2330.120i$	6.7E-07	1.2E-11	4.6E-08	7.8148E-04
$-730.6886 + 2330.120i$	6.7E-07	1.2E-11	4.6E-08	7.8148E-04

* 170 Lanczos iterations; 340 matrix-vector products
** 170 Lanczos iterations; 346 matrix-vector products
† ARPACK parameters: k = 6, p=300, 2 restarts, 888 matrix-vector products

The convergence behavior of the ULA for the eigenvalue of TOLS2000 with the third largest imaginary part is presented in the top panel of Figure 4.3. The associated cluster width is shown in the lower panel. As was also true for the RIEMANN matrices, the width of the cluster increases significantly when the first copy appears. As more copies accumulate the

ETNA
Kent State University
etna@mcs.kent.edu

width of the cluster of nearly degenerate copies increases further. The ULA is indeed poor, for example, the absolute error is never less than 10^{-5} for the eigenvalue shown in Figure 4.3. In contrast, eigenvalues obtained with the RULE are quite accurate (and about as accurate as those computed with ARPACK).

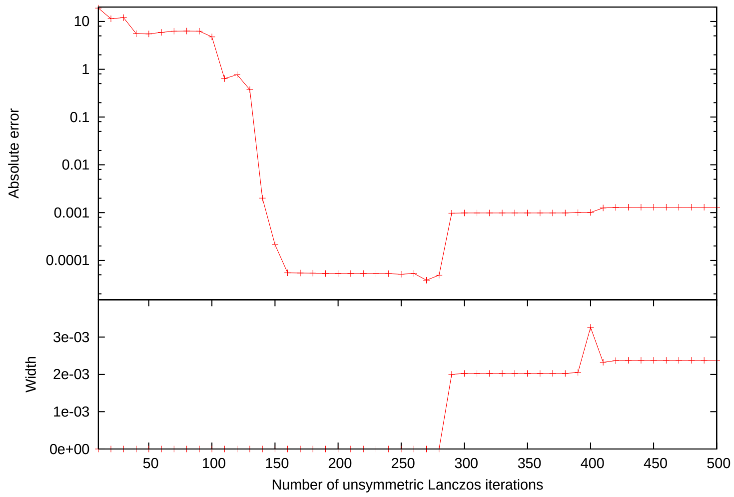

FIG. 4.3. *Convergence and cluster width of an extremal eigenvalue of TOLS2000 obtained with the ULA.*

4.4. PDE matrices. In this subsection we show that it is possible to use the RULE to compute interior eigenvalues. Without using a spectral transformation one can do this by using the SM option of ARPACK, but this requires storing many vectors in memory. To test the computation of interior eigenvalues with the RULE we have used matrices obtained by discretizing a 2-D elliptic differential equation with a five-point central finite difference scheme. On Matrix Market the matrix we use is called PDE2961 [43]. In Figure 4.4 we show the spectrum of the matrix of size $N = 2961$. The approximate eigenvectors we use are those whose corresponding eigenvalues lie in a rectangular box of width 0.5 and height $0.1i$ centered at a target value of $8.3 + 0.35i$. Again, the convergence behaviour of an eigenvalue computed with the ULA is similar to that depicted in Figure 2.2. Errors for eigenvalues close to the target are reported in Table 4.3. It is clear that the ULA is poor and that refinement improves the eigenvalues significantly. Although only eigenvalues close to the target are shown, all extremal eigenvalues are *also* well converged. An advantage of the RULE is that one does not need tridiagonal matrices of different size for different eigenvalues. All of the eigenvalues are extracted from one large tridiagonal matrix. This makes the method simple to use.

It is possible to compute the eigenvalues in Table 4.3 by applying ARPACK to $(\mathbf{G} - \sigma\mathbf{I})$, choosing σ to be in the centre of the desired spectral region, and using the SM option to extract eigenvalues with the smallest magnitude. Increasing p and therefore increasing the number of vectors stored in memory decreases the number of required matrix-vector products. We find that if $p = 100$ and $k = 50$ the number of vectors is not too large and 1385 matrix-vector products is enough to converge the eigenvalues in Table 4.3. Note also that because σ is complex, the cost of each ARPACK matrix-vector product, the cost of the ARPACK re-orthogonalisation, and the memory required to store the Arnoldi vectors are approximately doubled.

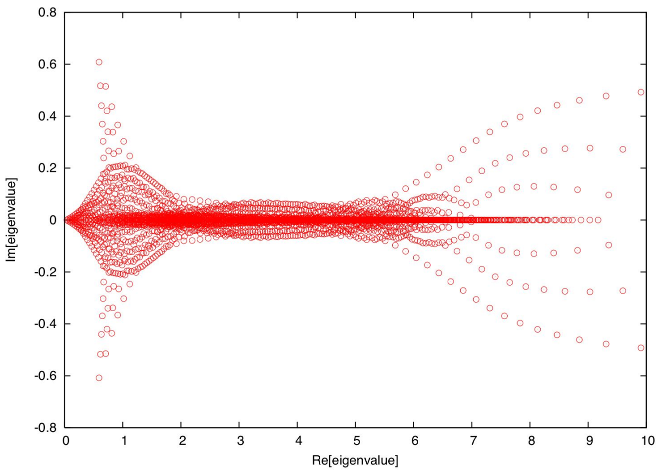

FIG. 4.4. *Distribution of eigenvalues of the PDE2961 matrix.*

TABLE 4.3
Interior eigenvalues of the PDE2961 matrix. The chosen target is $8.3 + 0.35i$. *The error is defined as the norm of the difference between computed and exact values.*

Refined eigenvalue	ULA*	RULE**	$\lvert \mathbf{x}^*_{(L)} \mathbf{x}_{(R)} \rvert$
$7.831661 + 0.3970848i$	7.4E-09	2.4E-12	3.0064E-02
$7.928410 + 0.2564949i$	3.3E-07	3.0E-11	2.0611E-03
$8.130354 + 0.4211668i$	1.0E-08	6.3E-14	5.3381E-02
$8.240396 + 0.2678090i$	3.8E-08	3.0E-13	3.8846E-03
$8.465128 + 0.4423992i$	2.7E-09	1.4E-14	8.1620E-02
$8.600357 + 0.2746980i$	9.4E-08	1.1E-13	6.3615E-03

* 450 Lanczos iterations; 900 matrix-vector products
** 450 Lanczos iterations; 906 matrix-vector products

4.5. AIRFOIL matrix. The AIRFOIL matrix of matrix market is a good test as the matrix is large (23 560), but small enough that benchmark eigenvalues can be computed with a direct method. The matrix is obtained by modeling flow over airfoils with a finite difference representation of the Navier-Stokes equations. More details can be found on the Matrix Market [43]. We focus on the eigenvalues of largest magnitude for which both ARPACK and the RULE should work well. The spectral distribution is depicted in Figure 4.5. As can be seen from Table 4.4 accurate eigenvalues are indeed obtained using both the RULE and ARPACK with a fairly small number of iterations. The error is defined as the absolute difference between an eigenvalue computed with Householder/QR [5] and either the RULE or ARPACK. The residuals are right eigenvector residuals defined by [4],

$$\text{residual} \;=\; \frac{||\mathbf{G}\mathbf{x}_j - \tilde{\lambda}_j \mathbf{x}_j||_2}{||\mathbf{G}||_1}\,. \tag{4.1}$$

ETNA
Kent State University
etna@mcs.kent.edu

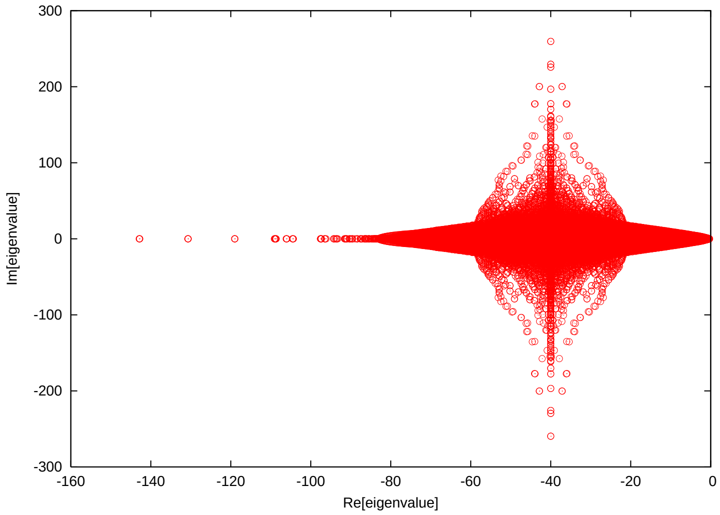

FIG. 4.5. *Distribution of eigenvalues of the AF23560 matrix.*

The matrix-vector product count slightly favours ARPACK in this case. The ARPACK results were obtained with a residual tolerance parameter equal to the machine precision.

4.6. Lifetimes of states of HCO. The imaginary part of the complex energy of a resonance state is related to the lifetime of the state [35, 38]. The most popular approach for computing resonance states uses a complex absorbing potential (CAP) to enforce appropriate boundary conditions. The operator equation to be solved is,

$$\left(\hat{H} + u_k \hat{W}(r)\right) \psi_k = E_k \psi_k \,, \tag{4.2}$$

where $\hat{H}$ is a Hamiltonian operator, $\hat{W}(r)$ is a CAP depending on the dissociation coordinate r, and E_k is the complex resonance energy; u_k is a function of E_k. If u_k is chosen appropriately and the operators are represented in a discrete variable representation (DVR) basis [24], (4.2) may be written as a quadratic matrix eigenvalue problem. There is some freedom in the choice of u_k; different choices yield different quadratic eigenvalue problems. We choose $u_k = i\sqrt{(\sigma - E_k)}$ so that after linearizing the quadratic eigenvalue problem in the standard fashion [5], we obtain an unsymmetric eigenvalue problem,

$$\mathbf{G}\mathbf{x}_k = u_k \mathbf{x}_k \,, \tag{4.3}$$

where

$$\mathbf{G} = \begin{pmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{H} - \sigma\mathbf{I} & \mathbf{W} \end{pmatrix} \tag{4.4}$$

is a *real* matrix and $\mathbf{x}_k = \binom{\psi_k}{u_k \psi_k}$. To obtain accurate resonance lifetimes one usually premultiplies the CAP by a parameter δ_{cap} and computes eigenvalues for several values of δ_{cap} to find the optimal absorbing potential. We choose σ large enough to ensure that for many states the E_k computed from the eigenvalues of matrix $\mathbf{G}$ are close to the actual resonance energies. For more detail the reader is referred to [25, 26, 37].

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 4.4
Error in extremal eigenvalues of the AF23560 matrix.

Eigenvalue	ULA*	RULE**	ARPACK†	RULE residual	$\lvert \mathbf{x}^*_{(L)}\mathbf{x}_{(R)} \rvert$
$-37.18288 + 200.1360i$	1.1E-08	7.1E-13	2.3E-07	3.3E-08	0.21239
$-37.18288 - 200.1360i$	1.1E-08	4.3E-13	2.3E-07	3.5E-08	0.21239
$-37.14444 + 200.2190i$	3.8E-09	7.4E-13	4.8E-07	1.1E-07	0.21716
$-37.14444 - 200.2190i$	3.8E-09	5.8E-13	4.8E-07	9.1E-08	0.21716
$-42.81924 + 200.1356i$	7.5E-08	1.1E-12	1.9E-07	1.2E-07	0.21236
$-42.81924 - 200.1356i$	7.5E-08	7.0E-13	1.9E-07	7.0E-08	0.21236
$-42.85767 + 200.2186i$	9.0E-08	9.5E-13	3.5E-07	8.3E-08	0.21713
$-42.85767 - 200.2186i$	9.0E-08	5.5E-13	3.5E-07	8.9E-08	0.21713
$-40.00075 + 225.7328i$	1.2E-08	6.5E-13	8.6E-07	4.7E-08	0.26668
$-40.00075 - 225.7328i$	1.2E-08	3.7E-13	8.6E-07	2.2E-08	0.26668
$-40.00074 + 225.7714i$	3.4E-09	7.4E-13	8.4E-07	2.9E-08	0.26530
$-40.00074 - 225.7714i$	3.3E-09	1.1E-12	8.4E-07	2.8E-08	0.26530
$-40.00010 + 229.5187i$	1.1E-08	3.1E-13	6.2E-07	3.9E-08	0.22165
$-40.00010 - 229.5187i$	1.1E-08	1.1E-13	6.2E-07	3.3E-08	0.22165
$-40.00095 + 229.5291i$	1.2E-08	5.7E-13	8.0E-07	7.7E-08	0.22236
$-40.00095 - 229.5291i$	1.2E-08	1.5E-13	8.0E-07	2.9E-08	0.22236
$-40.00045 + 259.5435i$	1.2E-10	1.0E-12	4.8E-07	2.2E-07	0.41406
$-40.00045 - 259.5435i$	1.4E-10	7.6E-13	4.8E-07	8.7E-08	0.41406
$-40.00041 + 259.5534i$	9.0E-11	6.9E-13	1.6E-06	1.9E-07	0.41406
$-40.00041 - 259.5534i$	1.6E-10	4.1E-13	1.6E-06	6.2E-08	0.41406

* 125 Lanczos iterations; 250 matrix-vector products
** 125 Lanczos iterations; 270 matrix-vector products
† ARPACK parameters: k = 20, p=50, 6 restarts, 172 matrix-vector products

The physically interesting eigenvalues, i.e. the bound states and the resonances, are located on and close to the imaginary axis, respectively. Because they are not extremal eigenvalues and because their condition numbers are large, it is difficult to compute them accurately. The DVR basis necessary to converge all low-lying resonance states of the HCO molecule gives rise to a real unsymmetric matrix $\mathbf{G}$ of size 184000, which is far too big to be diagonalized with direct methods on the computers we use. Table 4.5 shows some eigenvalues for HCO in the region $\text{Im}[\lambda] \geq 55$ and $|\text{Re}[\lambda]| \leq 0.1$, associated with narrow resonances and bound states. Because the RULE is a two-sided method we obtain simultaneously right and left eigenvectors. This makes it straightforward to compute condition numbers (last column).

To test the quality of the RULE results, we compute right eigenvector residuals. The accuracy of the eigenvalues computed with the RULE is good. With 3000 Lanczos iterations it is possible to compute not only the eigenvalues in Table 4.5 but also many others. The eigenvalues reported in Table 4.5 can be computed with fewer than 3000 iterations. Increasing the number of iterations does not adversely affect the quality of the eigenvalues. In constrast, the accuracy of eigenvalues computed with the ULA deteriorates as the number of iterations is increased (cf. Figures 2.2, and 4.3). Figure 4.6 shows the convergence of the residual, computed with the RULE, for the eigenvalue $\tilde{\lambda} = -0.000000000001 + 70.00861i$. The condition number of this eigenvalue is $\sim 10^2$. Despite the large condition number, the residual is less than $\sim 10^{-9}$ with 2200 Lanczos iterations. If the number of Lanczos iterations is increased the residual oscillates about an average value of $\sim 10^{-8}$. Similar oscillations are observed for most of the residuals. The oscillations appear to be due to fluctuations in the quality of the basis vectors $\mathbf{r}_j, \mathbf{l}_j$ used to form the matrix $\mathbf{G}_k$. The quality of the $\mathbf{r}_j, \mathbf{l}_j$ depends on the size of the Krylov subspaces used to compute them. The fluctuations do not significantly affect

TABLE 4.5
Selected narrow resonance states of the HCO molecule. The residual error is defined as the norm of the residual vectors.

Refined eigenvalue	RULE residual	$\lvert \mathbf{x}^*_{(L)}\mathbf{x}_{(R)} \rvert$
$0.000000000001608 + 99.63317i$	4.3E-09	2.007E-02
$0.000000000423172 + 94.06122i$	4.3E-09	2.126E-02
$-0.000000000030880 + 89.78747i$	2.0E-07	2.227E-02
$0.000000000017839 + 88.24727i$	7.6E-08	2.266E-02
$0.000000000000180 + 86.54286i$	1.3E-08	2.311E-02
$0.000000000000180 + 86.54286i$	1.9E-09	2.392E-02
$0.000000000013164 + 82.15547i$	1.9E-07	2.434E-02
$-0.000000000000079 + 80.30376i$	1.6E-08	2.490E-02
$-0.000000000003208 + 78.87198i$	2.0E-08	2.535E-02
$0.000000000002248 + 77.02845i$	8.2E-09	2.596E-02
$-0.000000000000226 + 75.73480i$	2.5E-09	2.640E-02
$-0.000000000000173 + 75.02610i$	9.3E-09	2.665E-02
$-0.000000000046711 + 73.83727i$	3.4E-08	2.708E-02
$-0.000000000044467 + 73.27523i$	1.8E-08	2.729E-02
$0.000000000000506 + 71.77733i$	3.2E-09	2.786E-02
$-0.000000000001369 + 70.00861i$	2.3E-09	2.856E-02
$-0.001503206670272 + 69.62269i$	1.2E-08	2.872E-02
$-0.001616758817681 + 69.28458i$	1.1E-07	2.886E-02
$-0.001799287965362 + 68.97843i$	7.1E-07	2.899E-02
$0.000000192654410 + 68.91187i$	1.1E-06	2.902E-02
$-0.001958836517186 + 68.69687i$	4.1E-06	2.910E-02

Number of unsymmetric Lanczos iterations: 3000

the accuracy of the eigenvalues since to eleven significant digits they do not depend on the number of Lanczos iterations.

5. Conclusion. In a previous paper [37] we used the idea of projecting onto approximate eigenvectors, computed from the ULA, to determine lifetimes of metastable states by calculating many eigentriplets. In this paper we have generalized the approach and made it more robust. In [37] we did not project a matrix onto its own approximate eigenspaces, instead we projected one matrix onto approximate eigenspaces of another, related, matrix. Other important differences between the approach of the this paper and the method used in [37] are: (1) In [37] approximate eigenvectors were obtained by searching for the best value of m for each eigenvalue; we have found that this is not necessary. (2) In [37] we solved a special (factorizable) generalized eigenvalue problem and in this paper we solve the unsymmetric eigenproblem. (3) We have modified the way clusters are defined and approximate eigenvectors are retained to ensure that we do not miss eigenvalues.

In this paper we show that a simple refinement makes it possible to extract accurate eigenvalues from the Krylov subspaces obtained from the ULA. We call the procedure the RULE. Using several test matrices we have shown that the RULE yields accurate extremal and interior eigenvalues. For all the eigenvalues in the tables we have computed eigenvectors and residuals. To use the RULE there is no need to store a large number of vectors in memory and no need to re-biorthogonalize Lanczos vectors. This makes it possible to use the RULE with very large matrices to compute large numbers of eigenvalues. One of the key advantages of the RULE is its ability to make use of large Krylov subspaces. The CPU cost of the RULE

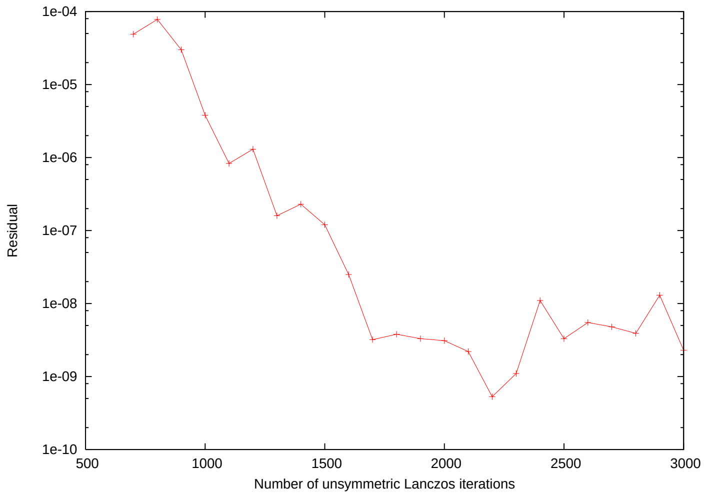

FIG. 4.6. *Residual of an eigenvalue associated with a resonance of HCO.*

is close to or less than that of ARPACK. To use the RULE one must store vectors on disk but reading them from disk does not take significant computer time because they are only used *after* the tridiagonal matrix has been generated. It is important to stress that the refinement we propose is very inexpensive. We typically compute eigenvalues in groups of about 20 or less. For each group of 20 the cost of the refinement is only 20 additional matrix-vector products.

Acknowledgments. We thank Rich Lehoucq, Chris Paige, Andreas Stathopoulos, Michiel Hochstenbach, Yvan Notay, and a referee who made suggestions for improving the manuscript.

REFERENCES

[1] P. ARBENZ, U. L. HETMANIUK, R. B. LEHOUCQ, AND R. S. TUMINARO, *A comparison of eigensolvers for large-scale 3D modal analysis using AMG-preconditioned iterative methods*, Internat. J. Numer. Methods Engrg., 64 (2005), pp. 204-236.

[2] J. BAGLAMA, D. CALVETTI, AND L. REICHEL, *Iterative methods for the computation of a few eigenvalues of a large symmetric matrix*, BIT, 36 (1996), pp. 400-421.

[3] Z. BAI, *Error analysis of the Lanczos algorithm for the nonsymmetric eigenvalue problem*, Math. Comp., 62 (1994), pp. 209-226.

[4] Z. BAI, D. DAY, AND Q. YE *ABLE: An adaptive block-Lanczos method for non-Hermitian eigenvalue problems*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 1060-1082.

[5] Z. BAI, J. DEMMEL, J. DONGARRA, A. RUHE, AND H. A. VAN DER VORST, eds., *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, SIAM, Philadelphia, 2000.

[6] J. G. L. BOOTEN AND H. A. VAN DER VORST, *Cracking large-scale eigenvalue computations, part I: algorithms*, Comput. Phys., 10 (1996), pp. 239-242.

[7] M. J. BRAMLEY AND T. CARRINGTON JR. *A general discrete variable method to calculate vibrational energy levels of three- and four-atom molecules*, J. Chem. Phys., 99, (1993), pp. 8519-8541.

[8] T. CARRINGTON JR., *Methods for calculating vibrational energy levels*, Can. J. Chem., 82 (2003), pp. 900-914.

[9] J. K. CULLUM AND R. A. WILLOUGHBY, *Computing eigenvalues of very large symmetric matrices-an implementation of a Lanczos algorithm with no reorthogonalization*, J. Comput. Phys., 44 (1981), pp. 329-358.

[10] ———, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations. Volume 1, Theory*, Birkhäuser, Boston, 1985.

[11] ———, *A practical procedure for computing eigenvalues of large sparse nonsymmetric matrices*, in Large Scale Eigenvalue Problems, J. K. Cullum and R. A. Willoughby, eds., North-Holland, Amsterdam, 1986.

[12] E. R. DAVIDSON, *The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real symmetric matrices*, J. Comput. Phys., 17 (1975), pp. 87-94.

[13] D. DAY, *An efficient implementation of the nonsymmetric Lanczos algorithm*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 566-589.
[14] U. ELSNER, V. MEHRMANN, F. MILDE, R. A. RÖMER, AND M. SCHREIBER, *The Anderson model of localization: a challenge for modern eigenvalue methods*, SIAM J. Sci. Comput., 20 (1999), pp. 2089-2102.
[15] D. R. FOKKEMA, G. L .G. SLEIJPEN, AND H. A. VAN DER VORST, *Jacobi-Davidson style QR and QZ algorithms for the partial reduction of matrix pencils*, SIAM J. Sci. Comput., 20 (1998), pp. 94-125.
[16] R. W. FREUND, M. GUTKNECHT, AND N. M. NACHTIGAL, *An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices*, SIAM J. Sci. Comput., 14 (1993), pp. 137-158.
[17] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, 2nd ed., Johns Hopkins University Press, Baltimore, 1989.
[18] M. E. HOCHSTENBACH AND G. L. G. SLEIJPEN, *Two-sided and alternating Jacobi–Davidson*, Linear Algebra Appl., 358 (2003), pp. 145–172.
[19] S. HUANG AND T. CARRINGTON JR., *Calculating interior eigenvalues and eigenvectors with an implicitly restarted and a filter diagonalization method*, Appl. Numer. Math., 37 (2001), pp. 307-317.
[20] Z. JIA, *Refined iterative algorithms based on Arnoldi's process for large unsymmetric eigenproblems*, Linear Algebra Appl., 259 (1997), pp. 1-23.
[21] R. B. LEHOUCQ, S. K. GRAY, D. H. ZHANG, AND J. C. LIGHT, *Vibrational eigenstates of four-atom molecules: a parallel strategy employing the implicitly restarted Lanczos method*, Comput. Physics Comm., 109 (1998), pp. 15-26.
[22] R. B. LEHOUCQ AND J. A. SCOTT, *An evaluation of software for computing eigenvalues of sparse nonsymmetric matrices*, Preprint MCS-P547-1195, Argonne National Laboratory.
[23] R. B. LEHOUCQ, D. C. SORENSEN, AND C. YANG, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, 1998.
[24] J. C. LIGHT AND T. CARRINGTON JR., *Discrete variable representations and their utilization* Adv. Chem. Phys., 114 (2000), pp. 263-310.
[25] V. A. MANDELSHTAM AND A. NEUMAIER, *Further generalization and numerical implementation of pseudotime Schrödinger equations for quantum scattering calculations*, J. Theor. Comput. Chem., 1 (2002), pp. 1-12.
[26] A. NEUMAIER AND V. A. MANDELSHTAM, *Pseudotime Schrödinger equation with absorbing potential for quantum scattering calculations*, Phys. Rev. Lett., 86, (2001), pp. 5031-5034.
[27] C. C. PAIGE, *The Computation of Eigenvalues and Eigenvectors of Very Large Sparse Matrices.*, PhD thesis, University of London, 1971.
[28] ———, *Computational variants of the Lanczos method for the eigenproblem*, J. Inst. Math. Appl., 10 (1972), pp. 373–381.
[29] ———, *Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix*, J. Inst. Math. Appl., 18 (1996), pp. 341-349.
[30] B. N. PARLETT, D. R. TAYLOR AND Z. A. LIU, *A look-ahead Lanczos algorithm for unsymmetric matrices*, Math. Comp., 44 (1985), pp. 105-124
[31] P.-N. ROY AND T. CARRINGTON JR., *An evaluation of methods designed to calculate energy levels in a selected range and application to a (one-dimensional) Morse oscillator and (three-dimensional) HCN/HNC*, J. Chem. Phys., 103 (1995), pp. 5600-5612.
[32] F. SARTORETTO, G. GAMBPMATO, AND G. PINI, *Accelerated simultaneous iterations for large finite element eigenvalues*, J. Comput. Phys., 81 (1989), pp. 53-69.
[33] D. C. SORENSEN, *Implicit application of polynomial filters in a $k-$step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357-385.
[34] G. W. STEWART, *Matrix Algorithms, Volume II: Eigensystems*, SIAM, Philadelphia, 2001.
[35] J. R. TAYLOR, *Scattering Theory*, Wiley, New York, 1972
[36] L. N. TREFETHEN AND D. BAU, III, *Numerical Linear Algebra*, SIAM, Philadelphia, 1997.
[37] J. C. TREMBLAY AND T. CARRINGTON JR., *Computing resonance energies, widths, and wave functions using a Lanczos method in real arithmetic*, J. Chem. Phys., 122 (2005), pp. 244107:1-11.
[38] D. G. TRUHLAR ed., *Resonances in Electron-Molecule Scattering, Van der Waals Complexes, and Reactive Chemical Dynamics*, American Chemical Society, Symposium Series No. 263, Washington, 1984.
[39] H. I. VAN DER VEEN AND K. VUIK, *Bi-Lanczos with partial orthogonalization*, Comput. & Structures, 56 (1995), pp.605-613.
[40] J. S. WARSA, T. A. WAREING, J. E. MOREL, J. M. MCGHEE, AND R. B. LEHOUCQ, *Krylov subspace iterations for deterministic $k-$eigenvalue calculations*, Nucl. Sci. Eng., 147 (2004), pp. 26-42.
[41] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Oxford University Press, New York, 1988.
[42] http://www.caam.rice.edu/software/ARPACK/
[43] http://math.nist.gov/MatrixMarket/
[44] http://www.ma.man.ac.uk/∼higham/mctoolbox/