56

Unsymmetric Matrix Eigenvalue Techniques

David S. Watkins
Washington State University

The definitions and basic properties of eigenvalues and eigenvectors are given in Section 4.4. A natural generalization is presented here in Section 56.1. Algorithms for computation of eigenvalues, eigenvectors, and their generalizations will be discussed in Sections 56.2 and 56.3.

If a large fraction of a matrix's entries are zeros, the matrix is called **sparse**. A matrix that is not sparse is called **dense**. Dense matrix techniques are methods that store the matrix in the conventional way, as an array, and operate on the array elements. Any matrix that is not too big to fit into a computer's main memory can be handled by dense matrix techniques, regardless of whether the matrix is dense or not. However, since the time to compute the eigenvalues of an $n \times n$ matrix by dense matrix techniques is proportional to n^3, the user may have to wait a while for the results if n is very large. Dense matrix techniques do not exploit the zeros in a matrix and tend to destroy them. With modern computers, dense matrix techniques can be applied to matrices of dimension up to 2000 or more. If a matrix is very large and sparse, and only a portion of the spectrum is needed, sparse matrix techniques (Section 56.3) are preferred.

The usual approach is to preprocess the matrix into Hessenberg form and then to effect a similarity transformation to triangular form: $T = S^{-1}AS$ by an iterative method. This yields the eigenvalues of A as the main-diagonal entries of T. For $k = 1, \ldots, n-1$, the first k columns of S span an invariant subspace. The eigenvectors of an upper-triangular matrix are easily computed by back substitution, and the eigenvectors of A can be deduced from the eigenvectors of T [GV13, Chap. 7.6], [Wat07, Chap. 4.8], [Wat10, Chap. 5.7]. If a matrix A is very large and sparse, only a partial similarity transformation is possible because a complete similarity transformation would require too much memory and take too long to compute.

56.1 The Generalized Eigenvalue Problem

Many matrix eigenvalue problems are most naturally viewed as generalized eigenvalue problems.

Definitions:

Given $A \in \mathbb{C}^{n\times n}$ and $B \in \mathbb{C}^{n\times n}$, the nonzero vector $\mathbf{v} \in \mathbb{C}^n$ is called an **eigenvector** of the pair

(A, B) if there are scalars μ, $\nu \in \mathbb{C}$, not both zero, such that

$$\nu A\mathbf{v} = \mu B\mathbf{v}.$$

Then, the scalar $\lambda = \mu/\nu$ is called the **eigenvalue** of (A, B) associated with the eigenvector $\mathbf{v}$. If $\nu = 0$, then the eigenvalue is ∞ by convention.

The expression $A - \lambda B$, with indeterminate λ, is called a **matrix pencil**. Whether we refer to the pencil $A - \lambda B$ or the pair (A, B), we are speaking of the same object. The pencil (or the pair (A, B)) is called **singular** if $A - \lambda B$ is singular for all $\lambda \in \mathbb{C}$. The pencil is **regular** if there exists a $\lambda \in \mathbb{C}$ such that $A - \lambda B$ is nonsingular. We will restrict our attention to regular pencils.

The **characteristic polynomial** of the pencil $A - \lambda B$ is $\det(\lambda B - A)$, and the **characteristic equation** is $\det(\lambda B - A) = 0$.

Two pairs (A, B) and (C, D) are **strictly equivalent** if there exist nonsingular matrices S_1 and S_2 such that $C - \lambda D = S_1(A - \lambda B)S_2$ for all $\lambda \in \mathbb{C}$. If S_1 and S_2 can be taken to be unitary, then the pairs are **strictly unitarily equivalent**.

A pair (A, B) is called **upper triangular** if both A and B are upper triangular.

Facts:

The following facts are discussed in [GV13, Chap. 7.7], [Wat07, Chap. 6.1], and [Wat10, Chap. 7.4].

1. When $B = I$, the generalized eigenvalue problem for the pair (A, B) reduces to the standard eigenvalue problem for the matrix A.
2. λ is an eigenvalue of (A, B) if and only if $A - \lambda B$ is singular.
3. λ is an eigenvalue of (A, B) if and only if $\ker(\lambda B - A) \neq \{0\}$.
4. The eigenvalues of (A, B) are exactly the solutions of the characteristic equation $\det(\lambda B - A) = 0$.
5. The characteristic polynomial $\det(\lambda B - A)$ is a polynomial in λ of degree $\leq n$.
6. The pair (A, B) (or the pencil $A - \lambda B$) is singular if and only if $\det(\lambda B - A) = 0$ for all λ.
7. If the pair (A, B) is regular, then $\det(\lambda B - A)$ is a nonzero polynomial of degree $k \leq n$. (A, B) has k finite eigenvalues.
8. The degree of $\det(\lambda B - A)$ is exactly n if and only if B is nonsingular.
9. If B is nonsingular, then the eigenvalues of (A, B) are exactly the eigenvalues of the matrices AB^{-1} and $B^{-1}A$.
10. If $\lambda \neq 0$, then λ is an eigenvalue of (A, B) if and only if λ^{-1} is an eigenvalue of (B, A).
11. Zero is an eigenvalue of (A, B) if and only if A is a singular matrix.
12. Infinity is an eigenvalue of (A, B) if and only if B is a singular matrix.
13. Two pairs that are strictly equivalent have the same eigenvalues.
14. If $C - \lambda D = S_1(A - \lambda B)S_2$, then $\mathbf{v}$ is an eigenvector of (A, B) if and only if $S_2^{-1}\mathbf{v}$ is an eigenvector of (C, D).
15. (Schur's Theorem) Every $A \in \mathbb{C}^{n\times n}$ is unitarily similar to an upper-triangular matrix S.
16. (Generalized Schur Theorem) Every pair (A, B) is strictly unitarily equivalent to an upper triangular pair (S, T).
17. The characteristic polynomial of an upper triangular pair (S, T) is $\prod_{k=1}^{n}(\lambda t_{kk} - s_{kk})$. The eigenvalues of (S, T) are $\lambda_k = s_{kk}/t_{kk}$, $k = 1, \ldots, n$. If $t_{kk} = 0$ and $s_{kk} \neq 0$, then $\lambda_k = \infty$. If $t_{kk} = 0$ and $s_{kk} = 0$ for some k, the pair (S, T) is singular.

Examples:

1. Let $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ and $B = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}$. Then the characteristic equation of the pair (A, B) is $\lambda^2 + \lambda - 2 = 0$, and the eigenvalues are 1 and -2.
2. Since the pencil
$$\begin{bmatrix} 2 & 5 \\ 0 & 7 \end{bmatrix} - \lambda \begin{bmatrix} 5 & 1 \\ 0 & 3 \end{bmatrix}$$
is upper triangular, its characteristic polynomial is $(5\lambda - 2)(3\lambda - 7)$, and its eigenvalues are $2/5$ and $7/3$.
3. The pencil
$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$
has characteristic equation $\lambda = 0$. It is a regular pencil with eigenvalues 0 and ∞.

56.2 Dense Matrix Techniques

The steps that are usually followed for solving the unsymmetric eigenvalue problem are preprocessing, eigenvalue computation with Francis's implicitly shifted QR algorithm, and eigenvector computation. The characteristic equation, which is important in theory, plays no role in practical eigenvalue computations.

The most widely used public domain software for this problem is from LAPACK [ABB99] and Chapter 93. Versions in FORTRAN and C are available. The most popular proprietary software is MATLAB, which uses computational routines from LAPACK. Several of LAPACK's computational routines will be mentioned in this section. LAPACK also has a number of driver routines that call the computational routines to perform the most common tasks, thereby making the user's job easier. A very easy way to use LAPACK routines is to use MATLAB.

This section presents algorithms for the reader's edification. However, the reader is strongly advised to use well-tested software written by experts whenever possible, rather than writing his or her own code. The actual software is very complex and addresses details that cannot be discussed here.

Definitions:

A matrix $A \in \mathbb{C}^{n \times n}$ is called **upper Hessenberg** if $a_{ij} = 0$ whenever $i > j + 1$. This means that every entry below the first subdiagonal of A is zero. An upper Hessenberg matrix is called **unreduced upper Hessenberg** if $a_{j+1,j} \neq 0$ for $j = 1, \ldots, n - 1$.

A matrix $A \in \mathbb{R}^{n \times n}$ is called **quasi-triangular** if it is block upper triangular with 1×1 and 2×2 blocks along the main diagonal.

Facts:

The following facts are proved in [Dem97], [GV13], [Kre05], [Wat07], or [Wat10].

1. Preprocessing is a two-step process involving balancing the matrix and transforming by unitary similarity to upper Hessenberg form.
2. The first step, which is optional, is to balance the matrix. The balancing operation begins by performing a permutation similarity transformation that exposes any obvious eigenvalues. The remaining submatrix is irreducible. It then performs a diagonal similarity transformation $D^{-1}AD$ that attempts to make the norms of the ith row

and ith column as nearly equal as possible, $i = 1, \ldots, n$. This has the effect of reducing the overall norm of the matrix and in diminishing the effects of roundoff errors [Osb60]. The scaling factors in D are taken to be powers of the base of floating point arithmetic (usually 2). No roundoff errors are caused by this transformation.

3. All modern balancing routines, including the code GEBAL in LAPACK, are derived from the code in Parlett and Reinsch [PR69]. See also [Kre05].

Algorithm 1: Balancing an Irreducible Matrix. An irreducible matrix $A \in \mathbb{C}^{n\times n}$ is input. On output, A has been overwritten by $D^{-1}AD$, where D is diagonal.

$b \leftarrow$ base of floating point arithmetic (usually 2)
$D \leftarrow I_n$
done $\leftarrow 0$
while done $= 0$
  done $\leftarrow 1$
  for $j = 1 : n$
    $c \leftarrow \sum_{i\neq j} |a_{ij}|, \quad r \leftarrow \sum_{k\neq j} |a_{jk}|$
    $s \leftarrow c + r, \quad f \leftarrow 1$
    while $b\,c < r$
      $c \leftarrow b\,c, \quad r \leftarrow r/b, \quad f \leftarrow b\,f$
    while $b\,r < c$
      $c \leftarrow c/b, \quad r \leftarrow b\,r, \quad f \leftarrow f/b$
    if $c + r < 0.95\,s$
      done $\leftarrow 0, \quad d_{jj} \leftarrow f\,d_{jj}$
      $A_{1:n,j} \leftarrow f\,A_{1:n,j}, \quad A_{j,1:n} \leftarrow (1/f)\,A_{j,1:n}$
end

4. In most cases, balancing will have little effect on the outcome of the computation, but sometimes it results in greatly improved accuracy [BDD00, Chap. 7.2].
5. The second preprocessing step is to transform the matrix to upper Hessenberg form. This is accomplished by a sequence of $n-2$ steps. On the jth step, zeros are introduced into the jth column.
6. For every $\mathbf{x} \in \mathbb{C}^n$ there is a unitary matrix U such that $U\mathbf{x} = \alpha\mathbf{e}_1$, for some scalar $\alpha \in \mathbb{C}$, where $\mathbf{e}_1$ is the vector having a 1 in the first position and zeros elsewhere. U can be chosen to be a rank-one modification of the identity matrix: $U = I + \mathbf{u}\mathbf{v}^*$. See Section 51.5 for a discussion of Householder and Givens matrices.
7.

Algorithm 2: Unitary Similarity Transformation to Upper Hessenberg Form. A general matrix $A \in \mathbb{C}^{n\times n}$ is input. On output, A has been overwritten by an upper Hessenberg matrix Q^*AQ. The unitary transforming matrix Q has also been generated.

$Q \leftarrow I_n$
for $j = 1 : n - 2$
  Let $\mathbf{x} = A_{j+1:n,j} \in \mathbb{C}^{n-j}$.
  Build unitary $U \in \mathbb{C}^{n-j\times n-j}$ such that $U^*\mathbf{x} = \gamma\mathbf{e}_1$.
  $A_{j+1:n,j:n} \leftarrow U^*A_{j+1:n,j:n}$
  $A_{1:n,j+1:n} \leftarrow A_{1:n,j+1:n}U$
  $Q_{1:n,j+1:n} \leftarrow Q_{1:n,j+1:n}U$
end

8. The cost of the reduction to Hessenberg form is proportional to n^3 for large n; that is, it is $O(n^3)$.
9. Once the matrix is in upper Hessenberg form, if any of the subdiagonal entries $a_{j+1,j}$ is zero, the matrix is block upper triangular with a $j \times j$ block and an $n-j \times n-j$ block, and the eigenvalue problem decouples to two independent problems of smaller size. Thus, we always work with unreduced upper Hessenberg matrices.
10. In practice we set an entry $a_{j+1,j}$ to zero whenever

$$|a_{j+1,j}| < \epsilon(|a_{jj}| + |a_{j+1,j+1}|),$$

where ϵ is the computer's unit roundoff.
11. If $T \in \mathbb{C}^{n\times n}$ is upper triangular and nonsingular, then T^{-1} is upper triangular. If $H \in \mathbb{C}^{n\times n}$ is upper Hessenberg, then TH, HT, and THT^{-1} are upper Hessenberg.
12. The standard method for computing the eigenvalues of a Hessenberg matrix is Francis's implicitly shifted QR algorithm [Fra61], [Wat11], an iterative method that produces a sequence of unitarily similar matrices that converges to upper triangular form.
13. The most basic (unshifted) version of the QR algorithm starts with $A_0 = A$, an unreduced upper Hessenberg matrix, and generates a sequence (A_m) as follows: Given A_{m-1}, a decomposition $A_{m-1} = Q_m R_m$, where Q_m is unitary and R_m is upper triangular, is computed. Then the factors are multiplied back together in reverse order to yield $A_m = R_m Q_m$. Equivalently, $A_m = Q_m^* A_{m-1} Q_m$.
14. Upper Hessenberg form is preserved by iterations of the QR algorithm.
15. The QR algorithm can also be applied to non-Hessenberg matrices, but the operations are much more economical in the Hessenberg case.
16. The basic QR algorithm converges slowly, so shifts of origin are used to accelerate convergence:

$$A_{m-1} - \mu_m I = Q_m R_m, \quad R_m Q_m + \mu_m I = Q_m^* A_{m-1} Q_m = A_m,$$

where $\mu_m \in \mathbb{C}$ is a shift chosen to approximate an eigenvalue.
17. Often it is convenient to take several steps at once:

Algorithm 3: Explicit QR iteration of degree k.

Choose k shifts $\mu_1, \ldots \mu_k$.
Let $p(A) = (A - \mu_1 I)(A - \mu_2 I) \cdots (A - \mu_k I)$.
Compute a QR decomposition $p(A) = QR$.
$A \leftarrow Q^* A Q$

18. A QR iteration of degree k is equivalent to k iterations of degree 1 with shifts $\mu_1, \ldots,$ μ_k applied in succession in any order [Wat07]. Upper Hessenberg form is preserved. In practice, k is never taken very big; typical values are 1, 2, 4, and 6.
19. One important application of multiple steps is to complex shifts applied to real matrices. Complex arithmetic is avoided by taking $k = 2$ and shifts related by $\mu_2 = \overline{\mu}_1$.
20. The usual choice of k shifts is the set of eigenvalues of the lower right hand $k \times k$ submatrix of the current iterate. With this choice of shifts at each iteration, the entry $a_{n-k+1,n-k}$ typically converges to zero quadratically [WE91], isolating a $k \times k$ submatrix after only a few iterations. However, convergence is not guaranteed, and failures do occasionally occur. No shifting strategy that guarantees convergence in all cases is known. For discussions of shifting strategies and convergence see [Wat07], [Wat10], or [WE91].

21. After each iteration, all of the subdiagonal entries should be checked to see if any of them can be set to zero. The objective is to break the big problem into many small problems in as few iterations as possible. Once a submatrix of size 1×1 has been isolated, an eigenvalue has been found. The eigenvalues of a 2×2 submatrix can be found by careful use of the quadratic formula. Complex conjugate eigenvalues of real matrices are extracted in pairs.
22. The explicit QR iteration shown above is expensive and never used in practice. Instead, the iteration is performed implicitly.

Algorithm 4: Francis Implicit QR iteration of degree k (chasing the bulge).

Choose k shifts $\mu_1, \ldots \mu_k$.
$\mathbf{x} \leftarrow \mathbf{e}_1$ % first column of identity matrix
for $j = 1 : k$
  $\left[\mathbf{x} \leftarrow (A - \mu_k I)\mathbf{x}\right.$
end % $\mathbf{x}$ is the first column of $p(A)$.
$\hat{\mathbf{x}} \leftarrow \mathbf{x}_{1:k+1}$ % $\mathbf{x}_{k+2:n} = 0$
Let $U \in \mathbb{C}^{k+1 \times k+1}$ be unitary with $U^*\mathbf{x} = \alpha \mathbf{e}_1$
$A_{1:k+1,1:n} \leftarrow U^* A_{1:k+1,1:n}$
$A_{1:n,1:k+1} \leftarrow A_{1:n,1:k+1} U$
Return A to upper Hessenberg form as in Algorithm 2 (Fact 7).

23. The initial transformation in the implicit QR iteration disturbs the upper Hessenberg form of A, making a bulge in the upper left-hand corner. The size of the bulge is equal to k. In the case $k = 2$, the pattern of nonzeros is

$$\begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ & & & * & * & * \\ & & & & * & * \end{bmatrix}.$$

The subsequent reduction to Hessenberg form chases the bulge down through the matrix and off the bottom. The equivalence of the explicit and implicit QR iterations is demonstrated in [GV13, Chap. 7.5] and [Wat07, Chap. 4.5]. For this result it is crucial that the matrix is unreduced upper Hessenberg
24. For a fixed small value of k, the implicit QR iteration requires only $O(n^2)$ work. Typically only a small number of iterations, independent of n, are needed per eigenvalue found; the total number of iterations is $O(n)$. Thus, the implicit QR algorithm is considered to be an $O(n^3)$ process.
25. The main unsymmetric QR routine in LAPACK is HSEQR, which chases bulges of degree $k = 2$. For efficient cache use, many bulges are chased in tight succession, and the transforming matrices are aggregated [BBM02a]. The technique of aggressive early deflation [BBM02b] is used to decrease the total number of iterations. For processing small submatrices, HSEQR calls LAHQR, a standard implicitly shifted QR code with $k = 2$.
26. If eigenvectors are wanted, the aggregate similarity transformation matrix S, the product of all transformations from start to finish, must be accumulated. $T = S^{-1}AS$, where A is the original matrix and T is the final upper triangular matrix. In the real case, T will not quite be upper triangular. It is **quasi-triangular** with a $2 \times$

2 block along the main diagonal for each complex conjugate pair of eigenvalues. This complicates the descriptions of the algorithms but does not cause any practical problems.

27. The eigenvectors of T are computed by back substitution [Wat10, Chap. 5.7]. For each eigenvector $\mathbf{x}$ of T, $S\mathbf{x}$ is an eigenvector of A. The total additional cost of the eigenvector computation is $O(n^3)$. In LAPACK these tasks are performed by the routines HSEQR and TREVC.
28. Invariant subspaces can also be computed. The eigenvalues of A are $\lambda_1 = t_{11}, \ldots, \lambda_n = t_{nn}$. If $\lambda_1, \ldots, \lambda_k$ are disjoint from $\lambda_{k+1}, \ldots, \lambda_n$, then because T is upper triangular, the first k columns of S span the invariant subspace associated with $\{\lambda_1, \ldots, \lambda_k\}$.
29. If an invariant subspace associated with k eigenvalues that are not at the top of T is wanted, then those k eigenvalues must be moved to the top by a sequence of swapping operations. Each operation is a unitary similarity transformation that reverses the positions of two adjacent main-diagonal entries of T. The transformations are applied to S as well. Once the desired eigenvalues have been moved to the top, the first k columns of the transformed S span the desired invariant subspace. For details see [BD93] and [GV13, Chap. 7.6]. In LAPACK these tasks are performed by the routines TREXC and TRSEN.
30. An important difference between the symmetric and unsymmetric eigenvalue problems is that in the unsymmetric case, the eigenvalues can be ill conditioned. That is, a small perturbation in the entries of A can cause a large change in the eigenvalues. Suppose λ is an eigenvalue of A of algebraic multiplicity 1, and let E be a perturbation that is small in the sense that $\|E\|_2 \ll \|A\|_2$. Then $A + E$ has an eigenvalue $\lambda + \delta$ near λ. A condition number for λ is the smallest number κ such that

$$|\delta| \leq \kappa \|E\|_2$$

for all small perturbations E. If $\mathbf{x}$ and $\mathbf{y}$ are eigenvectors of A and A^T, respectively, associated with λ, then [Wat10, Chap. 7.1]

$$\kappa \approx \frac{\|\mathbf{x}\|_2 \, \|\mathbf{y}\|_2}{|\mathbf{y}^T \mathbf{x}|}.$$

If $\kappa \gg 1$, λ is ill conditioned. If κ is not much bigger than 1, λ is well conditioned.
31. Condition numbers can also be defined for eigenvectors and invariant subspaces [GV13, Chap. 7.2], [Wat07, Chap. 2.7], [Wat10, Chap. 7.1]. Eigenvectors associated with a tight cluster of eigenvalues are always ill conditioned. A more meaningful object is the invariant subspace associated with all of the eigenvalues in the cluster. This space will usually be well conditioned, even though the eigenvectors are ill conditioned. The LAPACK routines TRSNA and TRSEN compute condition numbers for eigenvalues, eigenvectors, and invariant subspaces.
32. The invariant subspace associated with $\{\lambda_1, \ldots, \lambda_k\}$ will certainly be ill conditioned if any of the eigenvalues $\lambda_{k+1}, \ldots, \lambda_n$ are close to any of $\lambda_1, \ldots, \lambda_k$. A necessary (but not sufficient) condition for well conditioning is that $\lambda_1, \ldots, \lambda_k$ be well separated from $\lambda_{k+1}, \ldots, \lambda_n$. A related practical fact is that if two eigenvalues are very close together, it may not be possible to swap them stably by LAPACK's TREXC.
33. (Performance) A computer with an AMD Athlon dual-core processor running at 2.3 GHz with 2 GB main memory and 1 MB cache computed the complete eigensystem of a random 2000×2000 real matrix using MATLAB in 49 seconds. This included balancing, reduction to upper Hessenberg form, triangularization by the implicitly shifted QR algorithm, and back solving for the eigenvectors. All computed eigenpairs $(\lambda, \mathbf{v})$ satisfied $\|A\mathbf{v} - \lambda\mathbf{v}\|_1 < 10^{-15}\|A\|_1\|\mathbf{v}\|_1$.

34. (Performance, continued) A parallel version of the algorithm was used on 1024 cores of a parallel supercomputer to compute the Schur decomposition of a dense random matrix of dimension 10^5 in just under nine hours [GKK10].
35. (Generalized eigenvalue problem) The steps for solving the dense, unsymmetric, generalized eigenvalue problem $A\mathbf{v} = \lambda B\mathbf{v}$ are analogous to those for solving the standard problem. First (optionally) the pair (A, B) is balanced (by routine GGBAL in LAPACK). Then it is transformed by a strictly unitary equivalence to a condensed form in which A is upper Hessenberg and B is upper triangular. Then the QZ algorithm, a variant of Francis's algorithm, completes the reduction to triangular form. Details are given in [GV13, Chap. 7.7], [Wat07, Chap. 6], and [Wat10, Chap. 7.4]. In LAPACK the codes GGHRD and HGEQZ reduce the pair to Hessenberg-triangular form and perform the QZ iterations, respectively.
36. Once A has been reduced to triangular form, the eigenvalues are $\lambda_j = a_{jj}/b_{jj}$, $j = 1, \ldots, n$. The eigenvectors can be obtained by routines analogous to those used for the standard problem (LAPACK codes TGEVC and GGBAK), and condition numbers can be computed (LAPACK codes TGSNA and TGSEN).

Examples:

1. The matrix

$$A = \begin{bmatrix} -5.5849 \times 10^{-01} & -2.4075 \times 10^{+07} & -6.1644 \times 10^{+14} & 6.6275 \times 10^{+00} \\ -7.1724 \times 10^{-09} & -2.1248 \times 10^{+00} & -3.6183 \times 10^{+06} & 2.6435 \times 10^{-06} \\ -4.1508 \times 10^{-16} & -2.1647 \times 10^{-07} & 1.6229 \times 10^{-01} & -7.6315 \times 10^{-14} \\ 4.3648 \times 10^{-03} & 1.2614 \times 10^{+06} & -1.1986 \times 10^{+13} & -6.2002 \times 10^{-01} \end{bmatrix}$$

was balanced by Algorithm 1 (Fact 3) to produce

$$B = \begin{bmatrix} -0.5585 & -0.3587 & -1.0950 & 0.1036 \\ -0.4813 & -2.1248 & -0.4313 & 2.7719 \\ -0.2337 & -1.8158 & 0.1623 & -0.6713 \\ 0.2793 & 1.2029 & -1.3627 & -0.6200 \end{bmatrix}.$$

2. The matrix B of Example 1 was reduced to upper Hessenberg form by Algorithm 2 (Fact 7) to yield

$$H = \begin{bmatrix} -0.5585 & 0.7579 & 0.0908 & -0.8694 \\ 0.6036 & -3.2560 & -0.0825 & -1.8020 \\ 0 & 0.9777 & 1.2826 & -0.8298 \\ 0 & 0 & -1.5266 & -0.6091 \end{bmatrix}.$$

3. Algorithm 4 (Fact 22) was applied to the matrix H of Example 2 with $k = 1$ and shift $\mu_1 = h_{44} = -0.6091$ to produce

$$\begin{bmatrix} -3.1238 & -0.5257 & 1.0335 & 1.6798 \\ -1.3769 & 0.3051 & -1.5283 & 0.1296 \\ 0 & -1.4041 & 0.3261 & -1.0462 \\ 0 & 0 & -0.0473 & -0.6484 \end{bmatrix}.$$

The process was repeated twice again (with $\mu_1 = h_{44}$) to yield

$$\begin{bmatrix} -3.1219 & 0.7193 & 1.2718 & -1.4630 \\ 0.8637 & 1.8018 & 0.0868 & -0.3916 \\ 0 & 0.6770 & -1.2385 & 1.1642 \\ 0 & 0 & -0.0036 & -0.5824 \end{bmatrix}$$

and

$$\begin{bmatrix} -3.0939 & -0.6040 & 1.3771 & 1.2656 \\ -0.8305 & 1.8532 & -0.3517 & 0.5050 \\ 0 & 0.2000 & -1.3114 & -1.3478 \\ 0 & 0 & 0.00003 & -0.5888 \end{bmatrix}.$$

The (4,4) entry is an eigenvalue of A correct to four decimal places.
This matrix happens to have a real eigenvalue. If it had not, Algorithm 4 could have been used with $k = 2$ to extract the complex eigenvalues in pairs.

4. For an example of an ill-conditioned eigenvalue (Fact 30) consider a matrix

$$A = \begin{bmatrix} 1 & t \\ 0 & 1+\epsilon \end{bmatrix},$$

where t is large or ϵ is small or both. Since A is upper triangular, its eigenvalues are 1 and $1+\epsilon$. Eigenvectors of A and A^T associated with the eigenvalue 1 are

$$\mathbf{x} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} 1 \\ -t/\epsilon \end{bmatrix},$$

respectively. Since $\|\mathbf{x}\|_2 = 1$, $\|\mathbf{y}\|_2 = \sqrt{1 + t^2/\epsilon^2}$, and $|\mathbf{y}^T\mathbf{x}| = 1$, the condition number of eigenvalue $\lambda = 1$ is $\kappa = \sqrt{1 + t^2/\epsilon^2} \approx t/\epsilon$. Thus if, for example, $t = 10^7$ and $\epsilon = 10^{-7}$, we have $\kappa \approx 10^{14}$.

5. This example illustrates Fact 31 on the ill conditioning of eigenvectors associated with a tight cluster of eigenvalues. Given a positive number ϵ that is as small as you please, the matrices

$$A_1 = \begin{bmatrix} 2+\epsilon & 0 & 0 \\ 0 & 2-\epsilon & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and

$$A_2 = \begin{bmatrix} 2 & \epsilon & 0 \\ \epsilon & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

both have eigenvalues 1, $2+\epsilon$, and $2-\epsilon$, and they are very close together: $\|A_1 - A_2\|_2 = \sqrt{2}\epsilon$. However, unit eigenvectors associated with clustered eigenvalues $2+\epsilon$ and $2-\epsilon$ for A_1 are

$$\mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix},$$

while unit eigenvectors for A_2 are

$$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \quad \text{and} \quad \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}.$$

Thus, the tiny perturbation of order ϵ from A_1 to A_2 changes the eigenvectors completely; the eigenvectors are ill conditioned. In contrast, the two-dimensional invariant subspace associated with the cluster $2+\epsilon$, $2-\epsilon$ is $Span(\mathbf{e}_1, \mathbf{e}_2)$ for both A_1 and A_2, and it is well conditioned.

56.3 Sparse Matrix Techniques

If the matrix A is large and sparse, and just a few eigenvalues are needed, sparse matrix techniques are appropriate. Some examples of common tasks are (1) find the few eigenvalues of largest modulus, (2) find the few eigenvalues with largest real part, and (3) find the few eigenvalues nearest some target value τ. The corresponding eigenvectors might also be wanted. These tasks are normally accomplished by computing the low-dimensional invariant subspace associated with the desired eigenvalues. Then the information about the eigenvalues and eigenvectors is extracted from the invariant subspace.

The most widely used method for the sparse unsymmetric eigenvalue problem is the implicitly restarted Arnoldi method, as implemented in ARPACK [LSY98], which is discussed in Chapter 94. MATLAB's sparse eigenvalue command "eigs" calls ARPACK. An important variant is the Krylov-Schur algorithm of Stewart [Ste01].

Definitions:

Given a subspace $\mathcal{S}$ of $\mathbb{C}^n$, a vector $\mathbf{v} \in \mathcal{S}$ is called a **Ritz vector** of A from $\mathcal{S}$ if there is a $\theta \in \mathbb{C}$ such that $A\mathbf{v} - \theta\mathbf{v} \perp \mathcal{S}$. The scalar θ is the **Ritz value** associated with $\mathcal{S}$. The pair $(\theta, \mathbf{v})$ is a **Ritz pair**.

Facts:

1. [Wat10, Chap. 6.1] Let $\mathbf{v}_1, \ldots, \mathbf{v}_m$ be a basis for a subspace $\mathcal{S}$ of $\mathbb{C}^n$, and let $V = [\mathbf{v}_1 \ \cdots \mathbf{v}_m]$. Then $\mathcal{S}$ is invariant under A if and only if there is a $B \in \mathbb{C}^{m\times m}$ such that $AV = VB$.
2. [Wat10, Chap. 6.1] If $AV = VB$, then the eigenvalues of B are eigenvalues of A. If $\mathbf{x}$ is an eigenvector of B associated with eigenvalue μ, then $V\mathbf{x}$ is an eigenvector of A associated with μ.
3. [Wat10, Chap. 6.6] Let $\mathbf{v}_1, \ldots, \mathbf{v}_m$ be an orthonormal basis of $\mathcal{S}$, $V = [\mathbf{v}_1 \ \cdots \mathbf{v}_m]$, and $B = V^*AV$. Then the Ritz values of A associated with $\mathcal{S}$ are exactly the eigenvalues of B. If $(\theta, \mathbf{x})$ is an eigenpair of B, then $(\theta, V\mathbf{x})$ is a Ritz pair of A, and conversely.
4. If A is very large and sparse, it is essential to store A in a sparse data structure, in which only the nonzero entries of A are stored. One simple structure stores two integers n and nnz, which represent the dimension of the matrix and the number of nonzeros in the matrix, respectively. The matrix entries are stored in an array ent of length nnz, and the row and column indices are stored in two integer arrays of length nnz called row and col, respectively. For example, if the nonzero entry a_{ij} is stored in $ent(m)$, then this is indicated by setting $row(m) = i$ and $col(m) = j$. The space needed to store a matrix in this data structure is proportional to nnz.
5. Many operations that are routinely applied to dense matrices are impossible if the matrix is stored sparsely. Similarity transformations are out of the question because they quickly turn the zeros to nonzeros, transforming the sparse matrix to a full matrix.
6. One operation that is always possible is to multiply the matrix by a vector. This requires one pass through the data structure, and the work is proportional to nnz.

> **Algorithm 5: Sparse Matrix-Vector Multiply.**
> Multiply A by $\mathbf{x}$ and store the result in $\mathbf{y}$.
>
> $\mathbf{y} \leftarrow 0$
> for $m = 1 : nnz$
> $\quad \mathbf{y}(row(m)) \leftarrow \mathbf{y}(row(m)) + ent(m) * \mathbf{x}(col(m))$
> end

7. Because the matrix-vector multiply is so easy, many sparse matrix methods access

the matrix A in only this way. At each step, A is multiplied by one or several vectors, and this is the only way A is used.

8. The following standard methodology is widely used. A starting vector $\mathbf{v}_1$ is chosen, and the algorithm adds one vector per step, so that after $j-1$ steps it has produced j orthonormal vectors $\mathbf{v}_1, \ldots, \mathbf{v}_j$. Let $V_j = [\mathbf{v}_1, \ldots, \mathbf{v}_j] \in \mathbb{C}^{n\times j}$, and let $\mathcal{S}_j = Span(V_j) = Span(\mathbf{v}_1, \ldots, \mathbf{v}_j)$. The jth step uses information from $\mathcal{S}_j$ to produce $\mathbf{v}_{j+1}$. The Ritz values of A associated with $\mathcal{S}_j$ are the eigenvalues of the $j \times j$ matrix $B_j = V_j^* A V_j$. The Ritz pair $(\theta, \mathbf{w})$ for which θ has the largest modulus is an estimate of the largest eigenvalue of A, and $\mathbf{x} = V_j\mathbf{w}$ is an estimate of the associated eigenvector. The residual $\mathbf{r}_j = A\mathbf{x} - \mathbf{x}\theta$ gives an indication of the quality of the approximate eigenpair.
9. Several methods use the residual $\mathbf{r}_j$ to help decide on the next basis vector $\mathbf{v}_{j+1}$. These methods typically use $\mathbf{r}_j$ to determine another vector $\mathbf{s}_j$, which is then orthonormalized against $\mathbf{v}_1, \ldots, \mathbf{v}_j$ to produce $\mathbf{v}_{j+1}$. The choice $\mathbf{s}_j = \mathbf{r}_j$ leads to a method that is equivalent to the Arnoldi process. However, Arnoldi's process should not be implemented this way in practice; see Chapter 57. The choice $\mathbf{s}_j = (D - \theta I)^{-1}\mathbf{r}_j$, where D is the diagonal part of A, gives Davidson's method. The Jacobi-Davidson methods have more elaborate ways of choosing $\mathbf{s}_j$. See [BDD00, Chap. 7.12] for details.
10. Periodic purging is employed to keep the dimension of the active subspace from becoming too large. Given m vectors, the purging process keeps the most promising k-dimensional subspace of $\mathcal{S}_m = Span(V_m)$ and discards the rest. Again, let $B_m = V_m^* A V_m$, and let $B_m = U_m T_m U_m^*$ be a unitary similarity transformation to upper triangular form. The Ritz values lie on the main diagonal of T_m and can be placed in any order. Place the k most promising Ritz values at the top. Let $\tilde{V}_m = V_m U_m$, and let $\tilde{V}_k$ denote the $n \times k$ submatrix of $\tilde{V}_m$ consisting of the first k columns. The columns of $\tilde{V}_k$ are the vectors that are kept.
11. After each purge, the algorithm can be continued from step k. Once the basis has been expanded back to m vectors, another purge can be carried out. After a number of cycles of expansion and purging, the invariant subspace associated with the desired eigenvalues will have been found.
12. When purging is carried out in connection with the Arnoldi process, it is called an implicit restart, and there are some extra details. See Chapter 57, [Ste01], and [Wat07, Chap. 9.3].
13. The Implicitly Restarted Arnoldi process is well suited for computing the eigenvalues on the periphery of the spectrum of A. Thus, it is good for computing the eigenvalues of maximum modulus or those of maximum or minimum real part.
14. For computing interior eigenvalues, the shift-and-invert strategy is often helpful. Suppose the eigenvalues nearest some target value τ are sought. The matrix $(A - \tau I)^{-1}$ has the same eigenvectors as A, but the eigenvalues are different. If $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of A, then $(\lambda_1 - \tau)^{-1} \ldots, (\lambda_n - \tau)^{-1}$ are the eigenvalues of $(A - \tau I)^{-1}$. The eigenvalues of $(A - \tau I)^{-1}$ of largest modulus correspond to the eigenvalues of A closest to τ. These can be computed by applying the implicitly restarted Arnoldi process to $(A - \tau I)^{-1}$. This is feasible whenever operations of the type $\mathbf{w} \leftarrow (A - \tau I)^{-1}\mathbf{x}$ can be performed efficiently. If a sparse decomposition $A - \tau I = PLU$ can be computed, as described in Chapters 51 and 53, then that decomposition can be used to perform the operation $\mathbf{w} \leftarrow (A - \tau I)^{-1}\mathbf{x}$ by back solves. If the LU factors take up too much space to fit into memory, this method cannot be used.
15. Another option for solving $(A - \tau I)\mathbf{w} = \mathbf{x}$ is to use an iterative method, as described in Chapter 54. However, this is very computationally intensive, as the systems must be solved to high accuracy if the eigenvalues are to be computed accurately.
16. The shift-and-invert strategy can also be applied to the generalized eigenvalue prob-

lem $A\mathbf{v} = \lambda B\mathbf{v}$. The implicitly restarted Arnoldi process is applied to the operator $(A - \tau B)^{-1}B$ to find eigenvalues near τ.

17. If the matrix is too large for the shift-and-invert strategy, Jacobi-Davidson methods can be considered [BDD00, Chap. 7.12]. These also require the iterative solution of linear systems. In this family of methods, inaccurate solution of the linear systems may slow convergence of the algorithm, but it will not cause the eigenvalues to be computed inaccurately.
18. Arnoldi-based and Jacobi-Davidson algorithms are described in [BDD00]. A brief overview is given in [Wat10, Chap. 6]. Balancing of sparse matrices is discussed in [BDD00, Chap. 7.2].

References

[ABB99] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide, Third Edition.* SIAM, Philadelphia, 1999. www.netlib.org/lapack/lug/

[BD93] Z. Bai and J.W. Demmel. On swapping diagonal blocks in real Schur form. *Lin. Alg. Appl.*, 186: 73–95, 1993

[BDD00] 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, Philadelphia, 2000.

[BBM02a] K. Braman, R. Byers, and R. Mathias. The multi-shift QR algorithm part I: maintaining well focused shifts and level 3 performance. *SIAM J. Matrix Anal. Appl.*, 23: 929–947, 2002.

[BBM02b] K. Braman, R. Byers, and R. Mathias. The multi-shift QR algorithm part II: aggressive early deflation. *SIAM J. Matrix Anal. Appl.*, 23: 948–973, 2002.

[Dem97] J.W. Demmel. *Applied Numerical Linear Algebra* SIAM, Philadelphia, 1997.

[Fra61] J.G.F. Francis. The QR transformation, Part II. *Computer J.*, 4: 332–345, 1961

[GV13] G.H. Golub and C.F. Van Loan. *Matrix Computations*, 4th ed. The Johns Hopkins University Press, Baltimore, MD, 2013.

[GKK10] R. Granat, B. Kågström, and D. Kressner. A novel parallel QR algorithm for hybrid distributed memory HPC systems. *SIAM J. Sci. Stat. Comput.*, 32: 2345–2378, 2010.

[Kre05] D. Kressner. *Numerical Methods for General and Structured Eigenproblems* Springer, New York, 2005.

[LSY98] R.B. Lehoucq, D.C. Sorensen, and C. Yang. *ARPACK Users' Guide.* SIAM, Philadelphia, 1998.

[Osb60] E.E. Osborne. On pre-conditioning of matrices. *J. Assoc. Comput. Mach.*, 7: 338–345 1960.

[PR69] B.N. Parlett and C. Reinsch. Balancing a matrix for calculation of eigenvalues and eigenvectors. *Numer. Math.*, 13: 293–304, 1969. Also published as contribution II/11 in [WR71].

[Ste01] G.W. Stewart. A Krylov-Schur algorithm for large eigenproblems. *SIAM J. Matrix Anal. Appl.*, 23: 601–614, 2001.

[Wat07] D.S. Watkins. *The Matrix Eigenvalue Problem: GR and Krylov Subspace Methods.* SIAM, Philadelphia, 2007.

[Wat10] D.S. Watkins. *Fundamentals of Matrix Computations*, 3rd ed. John Wiley & Sons, New York, 2010.

[Wat11] D.S. Watkins. Francis's algorithm. *Amer. Math. Monthly*, 118: 387–403, 2011.

[WE91] D.S. Watkins and L. Elsner. Convergence of algorithms of decomposition type for the eigenvalue problem. *Lin. Alg. Appl.*, 143: 19–47, 1991.

[WR71] J.H. Wilkinson and C. Reinsch. *Handbook for Automatic Computation, Volume II, Linear Algebra.* Springer Verlag, New York, 1971.