Chapter 11. Eigensystems

11.0 Introduction

An $N \times N$ matrix $\mathbf{A}$ is said to have an *eigenvector* $\mathbf{x}$ and corresponding *eigenvalue* λ if

$$\mathbf{A} \cdot \mathbf{x} = \lambda \mathbf{x} \tag{11.0.1}$$

Obviously any multiple of an eigenvector $\mathbf{x}$ will also be an eigenvector, but we won't consider such multiples as being distinct eigenvectors. (The zero vector is not considered to be an eigenvector at all.) Evidently (11.0.1) can hold only if

$$\det |\mathbf{A} - \lambda \mathbf{1}| = 0 \tag{11.0.2}$$

which, if expanded out, is an Nth degree polynomial in λ whose roots are the eigenvalues. This proves that there are always N (not necessarily distinct) eigenvalues. Equal eigenvalues coming from multiple roots are called *degenerate*. Root-searching in the characteristic equation (11.0.2) is usually a very poor computational method for finding eigenvalues. We will learn much better ways in this chapter, as well as efficient ways for finding corresponding eigenvectors.

The above two equations also prove that every one of the N eigenvalues has a (not necessarily distinct) corresponding eigenvector: If λ is set to an eigenvalue, then the matrix $\mathbf{A} - \lambda \mathbf{1}$ is singular, and we know that every singular matrix has at least one nonzero vector in its nullspace (see §2.6 on singular value decomposition).

If you add $\tau \mathbf{x}$ to both sides of (11.0.1), you will easily see that the eigenvalues of any matrix can be changed or *shifted* by an additive constant τ by adding to the matrix that constant times the identity matrix. The eigenvectors are unchanged by this shift. Shifting, as we will see, is an important part of many algorithms for computing eigenvalues. We see also that there is no special significance to a zero eigenvalue. Any eigenvalue can be shifted to zero, or any zero eigenvalue can be shifted away from zero.

Definitions and Basic Facts

A matrix is called *symmetric* if it is equal to its transpose,

$$\mathbf{A} = \mathbf{A}^T \qquad \text{or} \qquad a_{ij} = a_{ji} \tag{11.0.3}$$

It is called *Hermitian* or *self-adjoint* if it equals the complex-conjugate of its transpose (its *Hermitian conjugate*, denoted by "†")

$$\mathbf{A} = \mathbf{A}^\dagger \qquad \text{or} \qquad a_{ij} = a_{ji}{}^* \tag{11.0.4}$$

It is termed *orthogonal* if its transpose equals its inverse,

$$\mathbf{A}^T \cdot \mathbf{A} = \mathbf{A} \cdot \mathbf{A}^T = \mathbf{1} \tag{11.0.5}$$

and *unitary* if its Hermitian conjugate equals its inverse. Finally, a matrix is called *normal* if it *commutes* with its Hermitian conjugate,

$$\mathbf{A} \cdot \mathbf{A}^\dagger = \mathbf{A}^\dagger \cdot \mathbf{A} \tag{11.0.6}$$

For real matrices, Hermitian means the same as symmetric, unitary means the same as orthogonal, and *both* of these distinct classes are normal.

The reason that "Hermitian" is an important concept has to do with eigenvalues. The eigenvalues of a Hermitian matrix are all real. In particular, the eigenvalues of a real symmetric matrix are all real. Contrariwise, the eigenvalues of a real nonsymmetric matrix may include real values, but may also include pairs of complex conjugate values; and the eigenvalues of a complex matrix that is not Hermitian will in general be complex.

The reason that "normal" is an important concept has to do with the eigenvectors. The eigenvectors of a normal matrix with nondegenerate (i.e., distinct) eigenvalues are complete and orthogonal, spanning the N-dimensional vector space. For a normal matrix with degenerate eigenvalues, we have the additional freedom of replacing the eigenvectors corresponding to a degenerate eigenvalue by linear combinations of themselves. Using this freedom, we can always perform Gram-Schmidt orthogonalization (consult any linear algebra text) and *find* a set of eigenvectors that are complete and orthogonal, just as in the nondegenerate case. The matrix whose columns are an orthonormal set of eigenvectors is evidently unitary. A special case is that the matrix of eigenvectors of a real, symmetric matrix is orthogonal, since the eigenvectors of that matrix are all real.

When a matrix is not normal, as typified by any random, nonsymmetric, real matrix, then in general we cannot find *any* orthonormal set of eigenvectors, nor even any pairs of eigenvectors that are orthogonal (except perhaps by rare chance). While the N non-orthonormal eigenvectors will "usually" span the N-dimensional vector space, they do not always do so; that is, the eigenvectors are not always complete. Such a matrix is said to be *defective*.

Left and Right Eigenvectors

While the eigenvectors of a non-normal matrix are not particularly orthogonal among themselves, they *do* have an orthogonality relation with a different set of vectors, which we must now define. Up to now our eigenvectors have been column vectors that are multiplied to the right of a matrix $\mathbf{A}$, as in (11.0.1). These, more explicitly, are termed *right eigenvectors*. We could also, however, try to find row vectors, which multiply $\mathbf{A}$ to the left and satisfy

$$\mathbf{x} \cdot \mathbf{A} = \lambda \mathbf{x} \tag{11.0.7}$$

These are called *left eigenvectors*. By taking the transpose of equation (11.0.7), we see that every left eigenvector is the transpose of a right eigenvector *of the transpose of* $\mathbf{A}$. Now by comparing to (11.0.2), and using the fact that the determinant of a matrix equals the determinant of its transpose, we also see that the left and right eigen*values* of $\mathbf{A}$ are identical.

If the matrix $\mathbf{A}$ is symmetric, then the left and right eigenvectors are just transposes of each other, that is, have the same numerical values as components. Likewise, if the matrix is self-adjoint, the left and right eigenvectors are Hermitian conjugates of each other. For the general nonnormal case, however, we have the following calculation: Let $\mathbf{X}_R$ be the matrix formed by columns from the right eigenvectors, and $\mathbf{X}_L$ be the matrix formed by rows from the left eigenvectors. Then (11.0.1) and (11.0.7) can be rewritten as

$$\mathbf{A} \cdot \mathbf{X}_R = \mathbf{X}_R \cdot \mathrm{diag}(\lambda_1 \dots \lambda_N) \qquad \mathbf{X}_L \cdot \mathbf{A} = \mathrm{diag}(\lambda_1 \dots \lambda_N) \cdot \mathbf{X}_L \tag{11.0.8}$$

Multiplying the first of these equations on the left by $\mathbf{X}_L$, the second on the right by $\mathbf{X}_R$, and subtracting the two, gives

$$(\mathbf{X}_L \cdot \mathbf{X}_R) \cdot \mathrm{diag}(\lambda_1 \dots \lambda_N) = \mathrm{diag}(\lambda_1 \dots \lambda_N) \cdot (\mathbf{X}_L \cdot \mathbf{X}_R) \tag{11.0.9}$$

This says that the matrix of dot products of the left and right eigenvectors commutes with the diagonal matrix of eigenvalues. But the only matrices that commute with a diagonal matrix *of distinct elements* are themselves diagonal. Thus, if the eigenvalues are nondegenerate, each left eigenvector is orthogonal to all right eigenvectors except its corresponding one, and vice versa. By choice of normalization, the dot products of corresponding left and right eigenvectors can always be made unity for any matrix with nondegenerate eigenvalues.

If some eigenvalues are degenerate, then either the left or the right eigenvectors corresponding to a degenerate eigenvalue must be linearly combined among themselves to achieve orthogonality with the right or left ones, respectively. This can always be done by a procedure akin to Gram-Schmidt orthogonalization. The normalization can then be adjusted to give unity for the nonzero dot products between corresponding left and right eigenvectors. If the dot product of corresponding left and right eigenvectors is zero at this stage, then you have a case where the eigenvectors are incomplete! Note that incomplete eigenvectors can occur only where there are degenerate eigenvalues, but do not always occur in such cases (in fact, never occur for the class of "normal" matrices). See [1] for a clear discussion.

In both the degenerate and nondegenerate cases, the final normalization to unity of all nonzero dot products produces the result: The matrix whose rows are left eigenvectors is the inverse matrix of the matrix whose columns are right eigenvectors, *if the inverse exists.*

Diagonalization of a Matrix

Multiplying the first equation in (11.0.8) by $\mathbf{X}_L$, and using the fact that $\mathbf{X}_L$ and $\mathbf{X}_R$ are matrix inverses, we get

$$\mathbf{X}_R^{-1} \cdot \mathbf{A} \cdot \mathbf{X}_R = \text{diag}(\lambda_1 \dots \lambda_N) \tag{11.0.10}$$

This is a particular case of a *similarity transform* of the matrix $\mathbf{A}$,

$$\mathbf{A} \quad \rightarrow \quad \mathbf{Z}^{-1} \cdot \mathbf{A} \cdot \mathbf{Z} \tag{11.0.11}$$

for some transformation matrix $\mathbf{Z}$. Similarity transformations play a crucial role in the computation of eigenvalues, because they leave the eigenvalues of a matrix unchanged. This is easily seen from

$$\begin{aligned} \det\left|\mathbf{Z}^{-1} \cdot \mathbf{A} \cdot \mathbf{Z} - \lambda\mathbf{1}\right| &= \det\left|\mathbf{Z}^{-1} \cdot (\mathbf{A} - \lambda\mathbf{1}) \cdot \mathbf{Z}\right| \\ &= \det|\mathbf{Z}| \ \det|\mathbf{A} - \lambda\mathbf{1}| \ \det\left|\mathbf{Z}^{-1}\right| \\ &= \det|\mathbf{A} - \lambda\mathbf{1}| \end{aligned} \tag{11.0.12}$$

Equation (11.0.10) shows that any matrix with complete eigenvectors (which includes all normal matrices and "most" random nonnormal ones) can be diagonalized by a similarity transformation, that the columns of the transformation matrix that effects the diagonalization are the right eigenvectors, and that the rows of its inverse are the left eigenvectors.

For real, symmetric matrices, the eigenvectors are real and orthonormal, so the transformation matrix is orthogonal. The similarity transformation is then also an *orthogonal transformation* of the form

$$\mathbf{A} \quad \rightarrow \quad \mathbf{Z}^T \cdot \mathbf{A} \cdot \mathbf{Z} \tag{11.0.13}$$

While real nonsymmetric matrices can be diagonalized in their usual case of complete eigenvectors, the transformation matrix is not necessarily real. It turns out, however, that a real similarity transformation can "almost" do the job. It can reduce the matrix down to a form with little two-by-two blocks along the diagonal, all other elements zero. Each two-by-two block corresponds to a complex-conjugate pair of complex eigenvalues. We will see this idea exploited in some routines given later in the chapter.

The "grand strategy" of virtually all modern eigensystem routines is to nudge the matrix $\mathbf{A}$ towards diagonal form by a sequence of similarity transformations,

$$\begin{aligned} \mathbf{A} \quad &\rightarrow \quad \mathbf{P}_1^{-1} \cdot \mathbf{A} \cdot \mathbf{P}_1 \quad \rightarrow \quad \mathbf{P}_2^{-1} \cdot \mathbf{P}_1^{-1} \cdot \mathbf{A} \cdot \mathbf{P}_1 \cdot \mathbf{P}_2 \\ &\rightarrow \quad \mathbf{P}_3^{-1} \cdot \mathbf{P}_2^{-1} \cdot \mathbf{P}_1^{-1} \cdot \mathbf{A} \cdot \mathbf{P}_1 \cdot \mathbf{P}_2 \cdot \mathbf{P}_3 \quad \rightarrow \quad \text{etc.} \end{aligned} \tag{11.0.14}$$

If we get all the way to diagonal form, then the eigenvectors are the columns of the accumulated transformation

$$\mathbf{X}_R = \mathbf{P}_1 \cdot \mathbf{P}_2 \cdot \mathbf{P}_3 \cdot \ldots \tag{11.0.15}$$

Sometimes we do not want to go all the way to diagonal form. For example, if we are interested only in eigenvalues, not eigenvectors, it is enough to transform the matrix $\mathbf{A}$ to be triangular, with all elements below (or above) the diagonal zero. In this case the diagonal elements are already the eigenvalues, as you can see by mentally evaluating (11.0.2) using expansion by minors.

There are two rather different sets of techniques for implementing the grand strategy (11.0.14). It turns out that they work rather well in combination, so most modern eigensystem routines use both. The first set of techniques constructs individual $\mathbf{P}_i$'s as explicit "atomic" transformations designed to perform specific tasks, for example zeroing a particular off-diagonal element (Jacobi transformation, §11.1), or a whole particular row or column (Householder transformation, §11.2; elimination method, §11.5). In general, a finite sequence of these simple transformations cannot completely diagonalize a matrix. There are then two choices: either use the finite sequence of transformations to go most of the way (e.g., to some special form like *tridiagonal* or *Hessenberg*, see §11.2 and §11.5 below) and follow up with the second set of techniques about to be mentioned; or else iterate the finite sequence of simple transformations over and over until the deviation of the matrix from diagonal is negligibly small. This latter approach is conceptually simplest, so we will discuss it in the next section; however, for N greater than ~ 10, it is computationally inefficient by a roughly constant factor ~ 5.

The second set of techniques, called *factorization methods*, is more subtle. Suppose that the matrix $\mathbf{A}$ can be factored into a left factor $\mathbf{F}_L$ and a right factor $\mathbf{F}_R$. Then

$$\mathbf{A} = \mathbf{F}_L \cdot \mathbf{F}_R \qquad \text{or equivalently} \qquad \mathbf{F}_L^{-1} \cdot \mathbf{A} = \mathbf{F}_R \tag{11.0.16}$$

If we now multiply back together the factors in the reverse order, and use the second equation in (11.0.16) we get

$$\mathbf{F}_R \cdot \mathbf{F}_L = \mathbf{F}_L^{-1} \cdot \mathbf{A} \cdot \mathbf{F}_L \tag{11.0.17}$$

which we recognize as having effected a similarity transformation on $\mathbf{A}$ with the transformation matrix being $\mathbf{F}_L$! In §11.3 and §11.6 we will discuss the *QR method* which exploits this idea.

Factorization methods also do not converge exactly in a finite number of transformations. But the better ones do converge rapidly and reliably, and, when following an appropriate initial reduction by simple similarity transformations, they are the methods of choice.

“Eigenpackages of Canned Eigenroutines”

You have probably gathered by now that the solution of eigensystems is a fairly complicated business. It is. It is one of the few subjects covered in this book for which we do *not* recommend that you avoid canned routines. On the contrary, the purpose of this chapter is precisely to give you some appreciation of what is going on inside such canned routines, so that you can make intelligent choices about using them, and intelligent diagnoses when something goes wrong.

You will find that almost all canned routines in use nowadays trace their ancestry back to routines published in Wilkinson and Reinsch’s *Handbook for Automatic Computation, Vol. II, Linear Algebra* [2]. This excellent reference, containing papers by a number of authors, is the Bible of the field. A public-domain implementation of the *Handbook* routines in FORTRAN is the EISPACK set of programs [3]. The routines in this chapter are translations of either the *Handbook* or EISPACK routines, so understanding these will take you a lot of the way towards understanding those canonical packages.

IMSL [4] and NAG [5] each provide proprietary implementations, in FORTRAN, of what are essentially the Handbook routines.

A good “eigenpackage” will provide separate routines, or separate paths through sequences of routines, for the following desired calculations:

- all eigenvalues and no eigenvectors
- all eigenvalues and some corresponding eigenvectors
- all eigenvalues and all corresponding eigenvectors

The purpose of these distinctions is to save compute time and storage; it is wasteful to calculate eigenvectors that you don’t need. Often one is interested only in the eigenvectors corresponding to the largest few eigenvalues, or largest few in magnitude, or few that are negative. The method usually used to calculate “some” eigenvectors is typically more efficient than calculating all eigenvectors if you desire fewer than about a quarter of the eigenvectors.

A good eigenpackage also provides separate paths for each of the above calculations for each of the following special forms of the matrix:

- real, symmetric, tridiagonal
- real, symmetric, banded (only a small number of sub- and superdiagonals are nonzero)
- real, symmetric
- real, nonsymmetric
- complex, Hermitian
- complex, non-Hermitian

Again, the purpose of these distinctions is to save time and storage by using the *least* general routine that will serve in any particular application.

In this chapter, as a bare introduction, we give good routines for the following paths:

- all eigenvalues and eigenvectors of a real, symmetric, tridiagonal matrix (§11.3)
- all eigenvalues and eigenvectors of a real, symmetric, matrix (§11.1–§11.3)
- all eigenvalues and eigenvectors of a complex, Hermitian matrix (§11.4)
- all eigenvalues and no eigenvectors of a real, nonsymmetric matrix

(§11.5–§11.6)

We also discuss, in §11.7, how to obtain some eigenvectors of nonsymmetric matrices by the method of inverse iteration.

Generalized and Nonlinear Eigenvalue Problems

Many eigenpackages also deal with the so-called *generalized eigenproblem,* [6]

$$\mathbf{A} \cdot \mathbf{x} = \lambda \mathbf{B} \cdot \mathbf{x} \tag{11.0.18}$$

where **A** and **B** are both matrices. Most such problems, where **B** is nonsingular, can be handled by the equivalent

$$(\mathbf{B}^{-1} \cdot \mathbf{A}) \cdot \mathbf{x} = \lambda \mathbf{x} \tag{11.0.19}$$

Often **A** and **B** are symmetric and **B** is positive definite. The matrix $\mathbf{B}^{-1} \cdot \mathbf{A}$ in (11.0.19) is not symmetric, but we can recover a symmetric eigenvalue problem by using the Cholesky decomposition $\mathbf{B} = \mathbf{L} \cdot \mathbf{L}^T$ of §2.9. Multiplying equation (11.0.18) by $\mathbf{L}^{-1}$, we get

$$\mathbf{C} \cdot (\mathbf{L}^T \cdot \mathbf{x}) = \lambda (\mathbf{L}^T \cdot \mathbf{x}) \tag{11.0.20}$$

where

$$\mathbf{C} = \mathbf{L}^{-1} \cdot \mathbf{A} \cdot (\mathbf{L}^{-1})^T \tag{11.0.21}$$

The matrix **C** is symmetric and its eigenvalues are the same as those of the original problem (11.0.18); its eigenfunctions are $\mathbf{L}^T \cdot \mathbf{x}$. The efficient way to form **C** is first to solve the equation

$$\mathbf{Y} \cdot \mathbf{L}^T = \mathbf{A} \tag{11.0.22}$$

for the lower triangle of the matrix **Y**. Then solve

$$\mathbf{L} \cdot \mathbf{C} = \mathbf{Y} \tag{11.0.23}$$

for the lower triangle of the symmetric matrix **C**.

Another generalization of the standard eigenvalue problem is to problems nonlinear in the eigenvalue λ, for example,

$$(\mathbf{A}\lambda^2 + \mathbf{B}\lambda + \mathbf{C}) \cdot \mathbf{x} = 0 \tag{11.0.24}$$

This can be turned into a linear problem by introducing an additional unknown eigenvector **y** and solving the $2N \times 2N$ eigensystem,

$$\begin{pmatrix} 0 & \mathbf{1} \\ -\mathbf{A}^{-1} \cdot \mathbf{C} & -\mathbf{A}^{-1} \cdot \mathbf{B} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = \lambda \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \tag{11.0.25}$$

This technique generalizes to higher-order polynomials in λ. A polynomial of degree M produces a linear $MN \times MN$ eigensystem (see [7]).

CITED REFERENCES AND FURTHER READING:

Stoer, J., and Bulirsch, R. 1980, *Introduction to Numerical Analysis* (New York: Springer-Verlag), Chapter 6. [1]

Wilkinson, J.H., and Reinsch, C. 1971, *Linear Algebra*, vol. II of *Handbook for Automatic Computation* (New York: Springer-Verlag). [2]

Smith, B.T., et al. 1976, *Matrix Eigensystem Routines — EISPACK Guide*, 2nd ed., vol. 6 of Lecture Notes in Computer Science (New York: Springer-Verlag). [3]

IMSL Math/Library Users Manual (IMSL Inc., 2500 CityWest Boulevard, Houston TX 77042). [4]

NAG Fortran Library (Numerical Algorithms Group, 256 Banbury Road, Oxford OX27DE, U.K.), Chapter F02. [5]

Golub, G.H., and Van Loan, C.F. 1989, *Matrix Computations*, 2nd ed. (Baltimore: Johns Hopkins University Press), §7.7. [6]

Wilkinson, J.H. 1965, *The Algebraic Eigenvalue Problem* (New York: Oxford University Press). [7]

Acton, F.S. 1970, *Numerical Methods That Work*; 1990, corrected edition (Washington: Mathematical Association of America), Chapter 13.

Horn, R.A., and Johnson, C.R. 1985, *Matrix Analysis* (Cambridge: Cambridge University Press).

11.1 Jacobi Transformations of a Symmetric Matrix

The Jacobi method consists of a sequence of orthogonal similarity transformations of the form of equation (11.0.14). Each transformation (a *Jacobi rotation*) is just a plane rotation designed to annihilate one of the off-diagonal matrix elements. Successive transformations undo previously set zeros, but the off-diagonal elements nevertheless get smaller and smaller, until the matrix is diagonal to machine precision. Accumulating the product of the transformations as you go gives the matrix of eigenvectors, equation (11.0.15), while the elements of the final diagonal matrix are the eigenvalues.

The Jacobi method is absolutely foolproof for all real symmetric matrices. For matrices of order greater than about 10, say, the algorithm is slower, by a significant constant factor, than the QR method we shall give in §11.3. However, the Jacobi algorithm is much simpler than the more efficient methods. We thus recommend it for matrices of moderate order, where expense is not a major consideration.

The basic Jacobi rotation $\mathbf{P}_{pq}$ is a matrix of the form

$$\mathbf{P}_{pq} = \begin{bmatrix} 1 & & & & & & \\ & \cdots & & & & & \\ & & c & \cdots & s & & \\ & & \vdots & 1 & \vdots & & \\ & & -s & \cdots & c & & \\ & & & & & \cdots & \\ & & & & & & 1 \end{bmatrix} \tag{11.1.1}$$

Here all the diagonal elements are unity except for the two elements c in rows (and columns) p and q. All off-diagonal elements are zero except the two elements s and $-s$. The numbers c and s are the cosine and sine of a rotation angle ϕ, so $c^2 + s^2 = 1$.