4

Nonsymmetric Eigenvalue Problems

4.1. Introduction

We discuss canonical forms (in section 4.2), perturbation theory (in section 4.3), and algorithms for the eigenvalue problem for a single nonsymmetric matrix A (in section 4.4). Chapter 5 is devoted to the special case of real symmetric matrices $A = A^T$ (and the SVD). Section 4.5 discusses generalizations to eigenvalue problems involving more than one matrix, including motivating applications from the analysis of vibrating systems, the solution of linear differential equations, and computational geometry. Finally, section 4.6 summarizes all the canonical forms, algorithms, costs, applications, and available software in a list.

One can roughly divide the algorithms for the eigenproblem into two groups: *direct methods* and *iterative methods.* This chapter considers only direct methods, which are intended to compute all of the eigenvalues, and (optionally) eigenvectors. Direct methods are typically used on dense matrices and cost $O(n^3)$ operations to compute all eigenvalues and eigenvectors; this cost is relatively insensitive to the actual matrix entries.

The main direct method used in practice is *QR iteration with implicit shifts* (see section 4.4.8). It is interesting that after more than 30 years of dependable service, convergence failures of this algorithm have quite recently been observed, analyzed, and patched [25, 65]. But there is still no global convergence proof, even though the current algorithm is considered quite reliable. So the problem of devising an algorithm that is numerically stable and globally (and quickly!) convergent remains open. (Note that "direct" methods must still iterate, since finding eigenvalues is mathematically equivalent to finding zeros of polynomials, for which no noniterative methods can exist. We call a method *direct* if experience shows that it (nearly) never fails to converge in a fixed number of iterations.)

Iterative methods, which are discussed in Chapter 7, are usually applied to sparse matrices or matrices for which matrix-vector multiplication is the only convenient operation to perform. Iterative methods typically provide approximations only to a subset of the eigenvalues and eigenvectors and are usually run only long enough to get a few adequately accurate eigenvalues rather than a large number. Their convergence properties depend strongly on the matrix entries.

4.2. Canonical Forms

DEFINITION 4.1. *The polynomial* $p(\lambda) = \det(A - \lambda I)$ *is called the* characteristic polynomial *of* A. *The roots of* $p(\lambda) = 0$ *are the* eigenvalues *of* A.

Since the degree of the characteristic polynomial $p(\lambda)$ equals n, the dimension of A, it has n roots, so A has n eigenvalues.

DEFINITION 4.2. *A nonzero vector* x *satisfying* $Ax = \lambda x$ *is a* (right) eigenvector *for the eigenvalue* λ. *A nonzero vector* y *such that* $y^*A = \lambda y^*$ *is a* left eigenvector. *(Recall that* $y^* = (\bar{y})^T$ *is the* conjugate transpose *of* y.*)*

Most of our algorithms will involve transforming the matrix A into simpler, or *canonical* forms, from which it is easy to compute its eigenvalues and eigenvectors. These transformations are called *similarity transformations* (see below). The two most common canonical forms are called the *Jordan form* and *Schur form*. The Jordan form is useful theoretically but is very hard to compute in a numerically stable fashion, which is why our algorithms will aim to compute the Schur form instead.

To motivate Jordan and Schur forms, let us ask which matrices have the property that their eigenvalues are easy to compute. The easiest case would be a *diagonal matrix*, whose eigenvalues are simply its diagonal entries. Equally easy would be a *triangular matrix*, whose eigenvalues are also its diagonal entries. Below we will see that a matrix in Jordan or Schur form is triangular. But recall that a real matrix can have complex eigenvalues, since the roots of its characteristic polynomial may be real or complex. Therefore, there is not always a *real* triangular matrix with the same eigenvalues as a *real* general matrix, since a real triangular matrix can only have real eigenvalues. Therefore, we must either use complex numbers or look beyond real triangular matrices for our canonical forms for real matrices. It will turn out to be sufficient to consider *block triangular matrices*, i.e., matrices of the form

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1b} \\ & A_{22} & \cdots & A_{2b} \\ & & \ddots & \vdots \\ & & & A_{bb} \end{bmatrix}, \tag{4.1}$$

where each A_{ii} is square and all entries below the A_{ii} blocks are zero. One can easily show that the characteristic polynomial $\det(A - \lambda I)$ of A is the product

$\prod_{i=1}^{b} \det(A_{ii} - \lambda I)$ of the characteristic polynomials of the A_{ii} and therefore that the set $\lambda(A)$ of eigenvalues of A is the union $\cup_{i=1}^{b}\lambda(A_{ii})$ of the sets of eigenvalues of the diagonal blocks A_{ii} (see Question 4.1). The canonical forms that we compute will be block triangular and will proceed computationally by breaking up large diagonal blocks into smaller ones. If we start with a complex matrix A, the final diagonal blocks will be 1-by-1, so the ultimate canonical form will be triangular. If we start with a real matrix A, the ultimate canonical form will have 1-by-1 diagonal blocks (corresponding to real eigenvalues) and 2-by-2 diagonal blocks (corresponding to complex conjugate pairs of eigenvalues); such a block triangular matrix is called *quasi-triangular.*

It is also easy to find the eigenvectors of a (block) triangular matrix; see section 4.2.1.

DEFINITION 4.3. *Let S be any nonsingular matrix. Then A and $B = S^{-1}AS$ are called* similar *matrices, and S is a* similarity transformation.

PROPOSITION 4.1. *Let $B = S^{-1}AS$, so A and B are similar. Then A and B have the same eigenvalues, and x (or y) is a right (or left) eigenvector of A if and only if $S^{-1}x$ (or S^*y) is a right (or left) eigenvector of B.*

Proof. Using the fact that $\det(X \cdot Y) = \det(X) \cdot \det(Y)$ for any square matrices X and Y, we can write $\det(A - \lambda I) = \det(S^{-1}(A - \lambda I)S) = \det(B - \lambda I)$, so A and B have the same characteristic polynomials. $Ax = \lambda x$ holds if and only if $S^{-1}ASS^{-1}x = \lambda S^{-1}x$ or $B(S^{-1}x) = \lambda(S^{-1}x)$. Similarly, $y^*A = \lambda y^*$ if and only if $y^*SS^{-1}AS = \lambda y^*S$ or $(S^*y)^*B = \lambda(S^*y)^*$. □

THEOREM 4.1. Jordan canonical form. *Given A, there exists a nonsingular S such that $S^{-1}AS = J$, where J is in* Jordan canonical form. *This means that J is block diagonal, with $J = \mathrm{diag}(J_{n_1}(\lambda_1), J_{n_2}(\lambda_2), \ldots, J_{n_k}(\lambda_k))$ and*

$$J_{n_i}(\lambda_i) = \begin{bmatrix} \lambda_i & 1 & & 0 \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ 0 & & & \lambda_i \end{bmatrix}^{n_i \times n_i} .$$

J is unique, up to permutations of its diagonal blocks.

For a proof of this theorem, see a book on linear algebra such as [110] or [139].

Each $J_m(\lambda)$ is called a *Jordan block* with eigenvalue λ of *algebraic multiplicity m.* If some $n_i = 1$, and λ_i is an eigenvalue of only that one Jordan block, then λ_i is called a *simple eigenvalue.* If all $n_i = 1$, so that J is diagonal, A is called *diagonalizable*; otherwise it is called *defective.* An n-by-n defective matrix does *not* have n eigenvectors, as described in more detail in the next

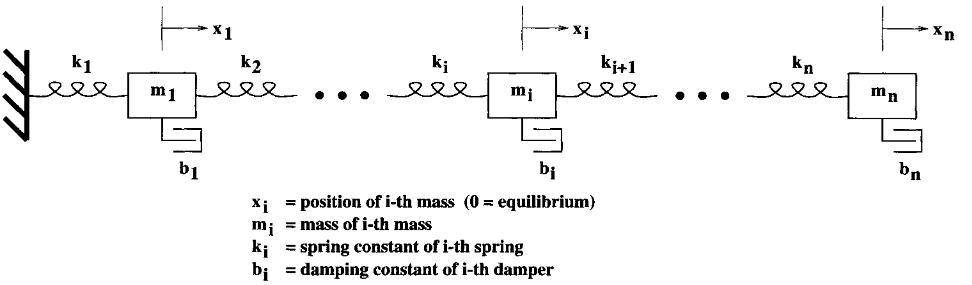
x1
k1
m1
k2
b1
xi
ki
mi
ki+1
bi
xn
kn
mn
bn
xi = position of i-th mass (0 = equilibrium)
mi = mass of i-th mass
ki = spring constant of i-th spring
bi = damping constant of i-th damper

Fig. 4.1. *Damped, vibrating mass-spring system.*

proposition. Although defective matrices are "rare" in a certain well-defined sense, the fact that some matrices do not have n eigenvectors is a fundamental fact confronting anyone designing algorithms to compute eigenvectors and eigenvalues. In section 4.3, we will see some of the difficulties that such matrices cause. Symmetric matrices, discussed in Chapter 5, are never defective.

PROPOSITION 4.2. *A Jordan block has one right eigenvector, $e_1 = [1, 0, \ldots, 0]^T$, and one left eigenvector, $e_n = [0, \ldots, 0, 1]^T$. Therefore, a matrix has n eigenvectors matching its n eigenvalues if and only if it is diagonalizable. In this case, $S^{-1}AS = \text{diag}(\lambda_i)$. This is equivalent to $AS = S\ \text{diag}(\lambda_i)$, so the ith column of S is a right eigenvector for λ_i. It is also equivalent to $S^{-1}A = \text{diag}(\lambda_i)S^{-1}$, so the conjugate transpose of the ith row of S^{-1} is a left eigenvector for λ_i. If all n eigenvalues of a matrix A are distinct, then A is diagonalizable.*

Proof. Let $J = J_m(\lambda)$ for ease of notation. It is easy to see $Je_1 = \lambda e_1$ and $e_n^T J = \lambda e_n^T$, so e_1 and e_n are right and left eigenvectors of J, respectively. To see that J has only one right eigenvector (up to scalar multiples), note that any eigenvector x must satisfy $(J - \lambda I)x = 0$, so x is in the null space of

$$J - \lambda I = \begin{bmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & 0 \end{bmatrix}.$$

But the null space of $J - \lambda I$ is clearly $\text{span}(e_1)$, so there is just one eigenvector. If all eigenvalues of A are distinct, then all its Jordan blocks must be 1-by-1, so $J = \text{diag}(\lambda_1, \ldots, \lambda_n)$ is diagonal. □

EXAMPLE 4.1. We illustrate the concepts of eigenvalue and eigenvector with a problem of *mechanical vibrations*. We will see a defective matrix arise in a natural physical context. Consider the damped mass spring system in Figure 4.1, which we will use to illustrate a variety of eigenvalue problems.

Newton's law $F = ma$ applied to this system yields

$$\begin{aligned} m_i\ddot{x}_i(t) &= k_i(x_{i-1}(t) - x_i(t)) \\ &\qquad \text{force on mass } i \text{ from spring } i \\ &\quad + k_{i+1}(x_{i+1}(t) - x_i(t)) \\ &\qquad \text{force on mass } i \text{ from spring } i+1 \\ &\quad - b_i\dot{x}_i(t) \\ &\qquad \text{force on mass } i \text{ from damper } i \end{aligned} \tag{4.2}$$

or

$$M\ddot{x}(t) = -B\dot{x}(t) - Kx(t), \tag{4.3}$$

where $M = \operatorname{diag}(m_1, \ldots, m_n)$, $B = \operatorname{diag}(b_1, \ldots, b_n)$, and

$$K = \begin{bmatrix} k_1 + k_2 & -k_2 & & & \\ -k_2 & k_2 + k_3 & -k_3 & & \\ & \ddots & \ddots & \ddots & \\ & & -k_{n-1} & k_{n-1} + k_n & -k_n \\ & & & -k_n & k_n \end{bmatrix}.$$

We assume that all the masses m_i are positive. M is called the *mass matrix*, B is the *damping matrix*, and K is the *stiffness matrix*.

Electrical engineers analyzing linear circuits arrive at an analogous equation by applying Kirchoff's and related laws instead of Newton's law. In this case x represents branch currents, M represent inductances, B represents resistances, and K represents admittances (reciprocal capacitances).

We will use a standard trick to change this second-order differential equation to a first-order differential equation, changing variables to

$$y(t) = \begin{bmatrix} \dot{x}(t) \\ x(t) \end{bmatrix}.$$

This yields

$$\begin{aligned} \dot{y}(t) &= \begin{bmatrix} \ddot{x}(t) \\ \dot{x}(t) \end{bmatrix} = \begin{bmatrix} -M^{-1}B\dot{x}(t) - M^{-1}Kx(t) \\ \dot{x}(t) \end{bmatrix} \\ &= \begin{bmatrix} -M^{-1}B & -M^{-1}K \\ I & 0 \end{bmatrix} \cdot \begin{bmatrix} \dot{x}(t) \\ x(t) \end{bmatrix} \\ &= \begin{bmatrix} -M^{-1}B & -M^{-1}K \\ I & 0 \end{bmatrix} \cdot y(t) \equiv Ay(t). \end{aligned} \tag{4.4}$$

To solve $\dot{y}(t) = Ay(t)$, we assume that $y(0)$ is given (i.e., the initial positions $x(0)$ and velocities $\dot{x}(0)$ are given).

One way to write down the solution of this differential equation is $y(t) = e^{At}y(0)$, where e^{At} is the matrix exponential. We will give another more elementary solution in the special case where A is diagonalizable; this will be

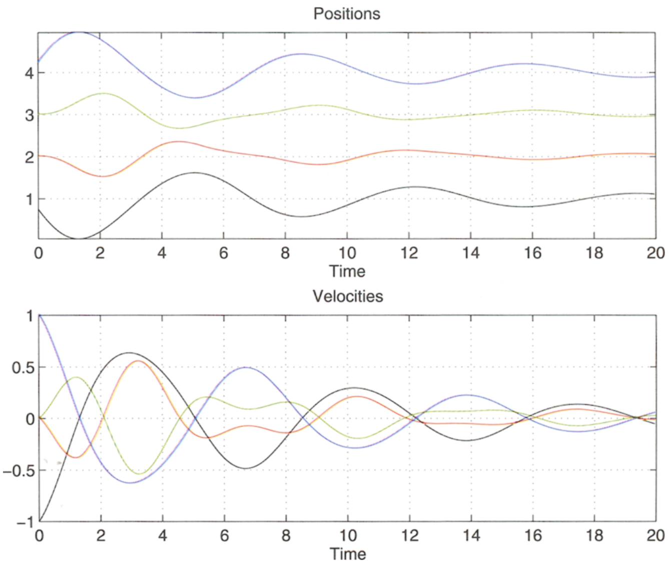

Fig. 4.2. *Positions and velocities of a mass-spring system with four masses* $m_1 = m_4 = 2$ *and* $m_2 = m_3 = 1$. *The spring constants are all* $k_i = 1$. *The damping constants are all* $b_i = .4$. *The initial displacements are* $x_1(0) = -.25$, $x_2(0) = x_3(0) = 0$, *and* $x_4(0) = .25$. *The initial velocities are* $v_1(0) = -1$, $v_2(0) = v_3(0) = 0$, *and* $v_4(0) = 1$. *The equilibrium positions are* 1, 2, 3, *and* 4. *The software for solving and plotting an arbitrary mass-spring system is HOMEPAGE/Matlab/massspring.m.*

true for almost all choices of m_i, k_i, and b_i. We will return to consider other situations later. (The general problem of computing matrix functions such as e^{At} is discussed further in section 4.5.1 and Question 4.4.)

When A is diagonalizable, we can write $A = S\Lambda S^{-1}$, where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$. Then $\dot{y}(t) = Ay(t)$ is equivalent to $\dot{y}(t) = S\Lambda S^{-1}y(t)$ or $S^{-1}\dot{y}(t) = \Lambda S^{-1}y(t)$ or $\dot{z}(t) = \Lambda z(t)$, where $z(t) \equiv S^{-1}y(t)$. This diagonal system of differential equations $\dot{z}_i(t) = \lambda_i z_i(t)$ has solutions $z_i(t) = e^{\lambda_i t}z_i(0)$, so $y(t) = S\text{diag}(e^{\lambda_1 t}, \ldots, e^{\lambda_n t})S^{-1}y(0) = Se^{\Lambda t}S^{-1}y(0)$. A sample numerical solution for four masses and springs is shown in Figure 4.2.

To see the physical significance of the nondiagonalizability of A for a mass-spring system, consider the case of a single mass, spring, and damper, whose differential equation we can simplify to $m\ddot{x}(t) = -b\dot{x}(t) - kx(t)$, and so $A = \left[\begin{array}{cc} -b/m & -k/m \\ 1 & 0 \end{array} \right]$. The two eigenvalues of A are $\lambda_{\pm} = \frac{b}{2m}(-1 \pm (1 - \frac{4km}{b^2})^{1/2})$. When $\frac{4km}{b^2} < 1$, the system is *overdamped*, and there are two negative real eigenvalues, whose mean value is $-\frac{b}{2m}$. In this case the solution eventually decays monotonically to zero. When $\frac{4km}{b^2} > 1$, the system is *underdamped*, and there are two complex conjugate eigenvalues with real part $-\frac{b}{2m}$. In this case the solution oscillates while decaying to zero. In both cases the system is diagonalizable since the eigenvalues are distinct. When $\frac{4km}{b^2} = 1$, the system is *critically damped*, there are two real eigenvalues equal to $-\frac{b}{2m}$, and A has a single 2-by-2 Jordan block with this eigenvalue. In other words, the nondiagonalizable matrices form the "boundary" between two physical behaviors: oscillation and monotonic decay.

When A is diagonalizable but S is an ill-conditioned matrix, so that S^{-1} is difficult to evaluate accurately, the explicit solution $y(t) = Se^{\Lambda t}S^{-1}y(0)$ will be quite inaccurate and useless numerically. We will use this mechanical system as a running example because it illustrates so many eigenproblems. ◇

To continue our discussion of canonical forms, it is convenient to define the following generalization of an eigenvector.

DEFINITION 4.4. *An* invariant subspace *of A is a subspace $\mathbf{X}$ of $\mathbb{R}^n$, with the property that $x \in \mathbf{X}$ implies that $Ax \in \mathbf{X}$. We also write this as $A\mathbf{X} \subseteq \mathbf{X}$.*

The simplest, one-dimensional invariant subspace is the set span(x) of all scalar multiples of an eigenvector x. Here is the analogous way to build an invariant subspace of larger dimension. Let $X = [x_1, \ldots, x_m]$, where $x_1, \ldots, x_m$ are any set of independent eigenvectors with eigenvalues $\lambda_1, \ldots, \lambda_m$. Then $\mathbf{X} = \text{span}(X)$ is an invariant subspace since $x \in \mathbf{X}$ implies $x = \sum_{i=1}^m \alpha_i x_i$ for some scalars α_i, so $Ax = \sum_{i=1}^m \alpha_i Ax_i = \sum_{i=1}^m \alpha_i \lambda_i x_i \in \mathbf{X}$. $A\mathbf{X}$ will equal $\mathbf{X}$ unless some eigenvalue λ_i equals zero. The next proposition generalizes this.

PROPOSITION 4.3. *Let A be n-by-n, let $X = [x_1, \ldots, x_m]$ be any n-by-m matrix with independent columns, and let $\mathbf{X} = \text{span}(X)$, the m-dimensional space spanned by the columns of X. Then $\mathbf{X}$ is an invariant subspace if and only if there is an m-by-m matrix B such that $AX = XB$. In this case the m eigenvalues of B are also eigenvalues of A. (When $m = 1$, $X = [x_1]$ is an eigenvector and B is an eigenvalue.)*

Proof. Assume first that $\mathbf{X}$ is invariant. Then each Ax_i is also in $\mathbf{X}$, so each Ax_i must be a linear combination of a basis of $\mathbf{X}$, say, $Ax_i = \sum_{j=1}^m x_j b_{ji}$. This last equation is equivalent to $AX = XB$. Conversely, $AX = XB$ means that each Ax_i is a linear combination of columns of X, so $\mathbf{X}$ is invariant.

Now assume $AX = XB$. Choose any n-by-$(n-m)$ matrix $\tilde{X}$ such that $\hat{X} = [X, \tilde{X}]$ is nonsingular. Then A and $\hat{X}^{-1}A\hat{X}$ are similar and so have the same eigenvalues. Write $\hat{X}^{-1} = \left[\begin{smallmatrix} Y^{m\times n} \\ \tilde{Y}^{(n-m)\times n} \end{smallmatrix}\right]$, so $\hat{X}^{-1}\hat{X} = I$ implies $YX = I$ and $\tilde{Y}X = 0$. Then $\hat{X}^{-1}A\hat{X} = \left[\begin{smallmatrix} Y \\ \tilde{Y} \end{smallmatrix}\right] \cdot [AX, A\tilde{X}] = \left[\begin{smallmatrix} YAX & YA\tilde{X} \\ \tilde{Y}AX & \tilde{Y}A\tilde{X} \end{smallmatrix}\right] = \left[\begin{smallmatrix} YXB & YA\tilde{X} \\ \tilde{Y}XB & \tilde{Y}A\tilde{X} \end{smallmatrix}\right] = \left[\begin{smallmatrix} B & YA\tilde{X} \\ 0 & \tilde{Y}A\tilde{X} \end{smallmatrix}\right]$. Thus by Question 4.1 the eigenvalues of A are the union of the eigenvalues of B and the eigenvalues of $\tilde{Y}A\tilde{X}$. □

For example, write the Jordan canonical form $S^{-1}AS = J = \text{diag}(J_{n_i}(\lambda_i))$ as $AS = SJ$, where $S = [S_1, S_2, \ldots, S_k]$ and S_i has n_i columns (the same as $J_{n_i}(\lambda_i)$; see Theorem 4.1 for notation). Then $AS = SJ$ implies $AS_i = S_i J_{n_i}(\lambda_i)$, i.e., $\text{span}(S_i)$ is an invariant subspace.

The Jordan form tells everything that we might want to know about a matrix and its eigenvalues, eigenvectors, and invariant subspaces. There are also explicit formulas based on the Jordan form to compute e^A or any other function of a matrix (see section 4.5.1). But it is bad to compute the Jordan form for two numerical reasons:

First reason: It is a discontinuous function of A, so any rounding error can change it completely.

EXAMPLE 4.2. Let

$$J_n(0) = \begin{bmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & 0 \end{bmatrix},$$

which is in Jordan form. For arbitrarily small ϵ, adding $i \cdot \epsilon$ to the (i, i) entry changes the eigenvalues to the n distinct values $i \cdot \epsilon$, and so the Jordan form changes from $J_n(0)$ to $\text{diag}(\epsilon, 2\epsilon, \ldots, n\epsilon)$. ◇

Second reason: It cannot be computed stably in general. In other words, when we have finished computing S and J, we cannot guarantee that $S^{-1}(A + \delta A)S = J$ for some small δA.

EXAMPLE 4.3. Suppose $S^{-1}AS = J$ exactly, where S is very ill-conditioned. ($\kappa(S) = \|S\| \cdot \|S^{-1}\|$ is very large.) Suppose that we are extremely lucky and manage to compute S *exactly* and J with just a tiny error δJ with $\|\delta J\| = O(\varepsilon)\|A\|$. How big is the backward error? In other words, how big must δA be so that $S^{-1}(A + \delta A)S = J + \delta J$? We get $\delta A = S\delta J S^{-1}$, and all that we can conclude is that $\|\delta A\| \leq \|S\| \cdot \|\delta J\| \cdot \|S^{-1}\| = O(\varepsilon)\kappa(S)\|A\|$. Thus $\|\delta A\|$ may be much larger than $\varepsilon\|A\|$, which prevents backward stability. ◇

So instead of computing $S^{-1}AS = J$, where S can be an arbitrarily ill-conditioned matrix, we will restrict S to be orthogonal (so $\kappa_2(S) = 1$) to guarantee stability. We cannot get a canonical form as simple as the Jordan form any more, but we do get something almost as good.

THEOREM 4.2. Schur canonical form. *Given A, there exists a unitary matrix Q and an upper triangular matrix T such that $Q^*AQ = T$. The eigenvalues of A are the diagonal entries of T.*

Proof. We use induction on n. It is obviously true if A is 1 by 1. Now let λ be any eigenvalue and u a corresponding eigenvector normalized so $\|u\|_2 = 1$. Choose $\tilde{U}$ so $U = [u, \tilde{U}]$ is a square unitary matrix. (Note that λ and u may be complex even if A is real.) Then

$$U^* \cdot A \cdot U = \begin{bmatrix} u^* \\ \tilde{U}^* \end{bmatrix} \cdot A \cdot [u, \tilde{U}] = \begin{bmatrix} u^*Au & u^*A\tilde{U} \\ \tilde{U}^*Au & \tilde{U}^*A\tilde{U} \end{bmatrix}.$$

Now as in the proof of Proposition 4.3, we can write $u^*Au = \lambda u^*u = \lambda$, and $\tilde{U}^*Au = \lambda\tilde{U}^*u = 0$ so $U^*AU \equiv [\begin{smallmatrix} \lambda & \tilde{a}_{12} \\ 0 & \tilde{A}_{22} \end{smallmatrix}]$. By induction, there is a unitary P, so $P^*A_{22}P = \tilde{T}$ is upper triangular. Then

$$U^*AU = \begin{bmatrix} \lambda & \tilde{a}_{12} \\ 0 & P\tilde{T}P^* \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & P \end{bmatrix} \begin{bmatrix} \lambda & \tilde{a}_{12}P \\ 0 & \tilde{T} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & P^* \end{bmatrix},$$

so

$$\begin{bmatrix} 1 & 0 \\ 0 & P^* \end{bmatrix} U^*AU \begin{bmatrix} 1 & 0 \\ 0 & P \end{bmatrix} = \begin{bmatrix} \lambda & \tilde{a}_{12}P \\ 0 & \tilde{T} \end{bmatrix} = T$$

is upper triangular and $Q = U[\begin{smallmatrix} 1 & 0 \\ 0 & P \end{smallmatrix}]$ is unitary as desired. □

Notice that the Schur form is not unique, because the eigenvalues may appear on the diagonal of T in any order.

This introduces complex numbers even when A is real. When A is real, we prefer a canonical form that uses only real numbers, because it will be cheaper to compute. As mentioned at the beginning of this section, this means that we will have to sacrifice a *triangular* canonical form and settle for a *block-triangular* canonical form.

THEOREM 4.3. Real Schur canonical form. *If A is real, there exists a real orthogonal matrix V such that $V^TAV = T$ is quasi–upper triangular. This means that T is block upper triangular with 1-by-1 and 2-by-2 blocks on the diagonal. Its eigenvalues are the eigenvalues of its diagonal blocks. The 1-by-1 blocks correspond to real eigenvalues, and the 2-by-2 blocks to complex conjugate pairs of eigenvalues.*

Proof. We use induction as before. Let λ be an eigenvalue. If λ is real, it has a real eigenvector u and we proceed as in the last theorem. If λ is complex, let u be a (necessarily) complex eigenvector, so $Au = \lambda u$. Since $\overline{Au} = A\overline{u} = \overline{\lambda}\overline{u}$, $\overline{\lambda}$ and $\overline{u}$ are also an eigenvalue/eigenvector pair. Let $u_R = \frac{1}{2}u + \frac{1}{2}\overline{u}$ be the real part of u and $u_I = \frac{1}{2i}u - \frac{1}{2i}\overline{u}$ be the imaginary part. Then $\text{span}\{u_R, u_I\} = \text{span}\{u, \overline{u}\}$ is a two-dimensional invariant subspace. Let $\tilde{U} = [u_R, u_I]$ and $\tilde{U} = QR$ be its QR decomposition. Thus $\text{span}\{Q\} = \text{span}\{u_R, u_I\}$ is invariant. Choose $\tilde{Q}$ so that $U = [Q, \tilde{Q}]$ is real and orthogonal, and compute

$$U^T \cdot A \cdot U = \begin{bmatrix} Q^T \\ \tilde{Q}^T \end{bmatrix} \cdot A \cdot [Q, \tilde{Q}] = \begin{bmatrix} Q^TAQ & Q^TA\tilde{Q} \\ \tilde{Q}^TAQ & \tilde{Q}^TA\tilde{Q} \end{bmatrix}.$$

Since Q spans an invariant subspace, there is a 2-by-2 matrix B such that $AQ = QB$. Now as in the proof of Proposition 4.3, we can write $Q^TAQ = Q^TQB = B$ and $\tilde{Q}^TAQ = \tilde{Q}^TQB = 0$, so $U^TAU = [\begin{smallmatrix} B & Q^TA\tilde{Q} \\ 0 & \tilde{Q}^TA\tilde{Q} \end{smallmatrix}]$. Now apply induction to $\tilde{Q}^TA\tilde{Q}$. □

4.2.1. Computing Eigenvectors from the Schur Form

Let $Q^*AQ = T$ be the Schur form. Then if $Tx = \lambda x$, we have $AQx = QTx = \lambda Qx$, so Qx is an eigenvector of A. So to find eigenvectors of A, it suffices to find eigenvectors of T.

Suppose that $\lambda = t_{ii}$ has multiplicity 1 (i.e., it is simple). Write $(T-\lambda I)x = 0$ as

$$\begin{aligned} 0 &= \begin{bmatrix} T_{11}-\lambda I & T_{12} & T_{13} \\ 0 & 0 & T_{23} \\ 0 & 0 & T_{33}-\lambda I \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \\ &= \begin{bmatrix} (T_{11}-\lambda I)x_1 + T_{12}x_2 + T_{13}x_3 \\ T_{23}x_3 \\ (T_{33}-\lambda I)x_3 \end{bmatrix}, \end{aligned}$$

where T_{11} is $(i-1)$-by-$(i-1)$, $T_{22} = \lambda$ is 1-by-1, T_{33} is $(n-i)$-by-$(n-i)$, and x is partitioned conformably. Since λ is simple, both $T_{11} - \lambda I$ and $T_{33} - \lambda I$ are nonsingular, so $(T_{33} - \lambda I)x_3 = 0$ implies $x_3 = 0$. Therefore $(T_{11} - \lambda I)x_1 = -T_{12}x_2$. Choosing (arbitrarily) $x_2 = 1$ means $x_1 = -(T_{11} - \lambda I)^{-1}T_{12}$, so

$$x = \begin{bmatrix} (\lambda I - T_{11})^{-1}T_{12} \\ 1 \\ 0 \end{bmatrix}.$$

In other words, we just need to solve a triangular system for x_1. To find a *real* eigenvector from real Schur form, we get a quasi-triangular system to solve. Computing complex eigenvectors from real Schur form using only real arithmetic also just involves equation solving but is a little trickier. See subroutine `strevc` in LAPACK for details.

4.3. Perturbation Theory

In this section we will concentrate on understanding when eigenvalues are ill-conditioned and thus hard to compute accurately. In addition to providing error bounds for computed eigenvalues, we will also relate eigenvalue condition numbers to related quantities, including the distance to the nearest matrix with an *infinitely* ill-conditioned eigenvalue, and the condition number of the matrix of eigenvectors.

We begin our study by asking when eigenvalues have *infinite* condition numbers. This is the case for multiple eigenvalues, as the following example illustrates.

EXAMPLE 4.4. Let

$$A = \begin{bmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ \epsilon & & & 0 \end{bmatrix}$$

be an n-by-n matrix. Then A has characteristic polynomial $\lambda^n - \epsilon = 0$ so $\lambda = \sqrt[n]{\epsilon}$ (n possible values). The nth root of ϵ grows much faster than any multiple of ϵ for small ϵ. More formally, the condition number is infinite because $\frac{d\lambda}{d\epsilon} = \frac{\epsilon^{\frac{1}{n}-1}}{n} = \infty$ at $\epsilon = 0$ for $n \geq 2$. For example, take $n = 16$ and $\epsilon = 10^{-16}$. Then for each eigenvalue $|\lambda| = .1$. ◇

So we expect a large condition number if an eigenvalue is "close to multiple"; i.e., there is a small δA such that $A+\delta A$ has exactly a multiple eigenvalue. Having an infinite condition number does not mean that they cannot be computed with any correct digits, however.

PROPOSITION 4.4. *Eigenvalues of A are continuous functions of A, even if they are not differentiable.*

Proof. It suffices to prove the continuity of roots of polynomials, since the coefficients of the characteristic polynomial are continuous (in fact polynomial) functions of the matrix entries. We use the argument principle from complex analysis [2]: the number of roots of a polynomial p inside a simple closed curve γ is $\frac{1}{2\pi i}\oint_\gamma \frac{p'(z)}{p(z)}dz$. If p is changed just a little, $\frac{p'(z)}{p(z)}$ is changed just a little, so $\frac{1}{2\pi i}\oint_\gamma \frac{p'(z)}{p(z)}dz$ is changed just a little. But since it is an integer, it must be constant, so the number of roots inside the curve γ is constant. This means that the roots cannot pass outside the curve γ (no matter how small γ is, provided that we perturb p by little enough), so the roots must be continuous. □

In what follows, we will concentrate on computing the condition number of a simple eigenvalue. If λ is a simple eigenvalue of A and δA is small, then we can identify an eigenvalue $\lambda + \delta\lambda$ of $A + \delta A$ "corresponding to" λ: it is the closest one to λ. We can easily compute the condition number of a simple eigenvalue.

THEOREM 4.4. *Let λ be a simple eigenvalue of A with right eigenvector x and left eigenvector y, normalized so that $\|x\|_2 = \|y\|_2 = 1$. Let $\lambda + \delta\lambda$ be the corresponding eigenvalue of $A + \delta A$. Then*

$$\begin{aligned} \delta\lambda &= \frac{y^*\delta Ax}{y^*x} + O(\|\delta A\|^2) \text{ or} \\ |\delta\lambda| &\leq \frac{\|\delta A\|}{|y^*x|} + O(\|\delta A\|^2) = \sec\Theta(y,x)\|\delta A\| + O(\|\delta A\|^2), \end{aligned}$$

where $\Theta(y,x)$ is the acute angle between y and x. In other words, $\sec\Theta(y,x) = 1/|y^*x|$ *is the* condition number *of the eigenvalue λ.*

Proof. Subtract $Ax = \lambda x$ from $(A + \delta A)(x + \delta x) = (\lambda + \delta\lambda)(x + \delta x)$ to get

$$A\delta x + \delta Ax + \delta A\delta x = \lambda\delta x + \delta\lambda x + \delta\lambda\delta x.$$

Ignore the second-order terms (those with two "δ terms" as factors: $\delta A\delta x$ and $\delta\lambda\delta x$) and multiply by y^* to get $y^*A\delta x + y^*\delta Ax = y^*\lambda\delta x + y^*\delta\lambda x$.

Now $y^*A\delta x$ cancels $y^*\lambda\delta x$, so we can solve for $\delta\lambda = (y^*\delta Ax)/(y^*x)$ as desired. □

Note that a Jordan block has right and left eigenvectors e_1 and e_n, respectively, so the condition number of its eigenvalue is $1/|e_n^*e_1| = 1/0 = \infty$, which agrees with our earlier analysis.

At the other extreme, in the important special case of symmetric matrices, the condition number is 1, so the eigenvalues are always accurately determined by the data.

Corollary 4.1. *Let A be symmetric (or normal: $AA^* = A^*A$). Then $|\delta\lambda| \leq \|\delta A\| + O(\|\delta A\|^2)$.*

Proof. If A is symmetric or normal, then its eigenvectors are all orthogonal, i.e., $Q^*AQ = \Lambda$ with $QQ^* = I$. So the right eigenvectors x (columns of Q) and left eigenvectors y (conjugate transposes of the rows of Q^*) are identical, and $1/|y^*x| = 1$. □

To see a variety of numerical examples, run the Matlab code referred to in Question 4.14.

Later, in Theorem 5.1, we will prove that in fact $|\delta\lambda| \leq \|\delta A\|_2$ if $\delta A = \delta A^T$, no matter how large $\|\delta A\|_2$ is.

Theorem 4.4 is useful only for sufficiently small $\|\delta A\|$. We can remove the $O(\|\delta A\|^2)$ term and so get a simple theorem true for any size perturbation $\|\delta A\|$, at the cost of increasing the condition number by a factor of n.

Theorem 4.5. Bauer–Fike. *Let A have all simple eigenvalues (i.e., be diagonalizable). Call them λ_i, with right and left eigenvectors x_i and y_i, normalized so $\|x_i\|_2 = \|y_i\|_2 = 1$. Then the eigenvalues of $A + \delta A$ lie in disks B_i, where B_i has center λ_i and radius $n\frac{\|\delta A\|_2}{|y_i^* x_i|}$.*

Our proof will use Gershgorin's theorem (Theorem 2.9), which we repeat here.

Gershgorin's theorem. *Let B be an arbitrary matrix. Then the eigenvalues λ of B are located in the union of the n disks defined by $|\lambda - b_{ii}| \leq \sum_{j \neq i} |b_{ij}|$ for $i = 1$ to n.*

We will also need two simple lemmas.

Lemma 4.1. *Let $S = [x_1, \ldots, x_n]$, the nonsingular matrix of right eigenvectors. Then*

$$S^{-1} = \begin{bmatrix} y_1^*/y_1^*x_1 \\ y_2^*/y_2^*x_2 \\ \vdots \\ y_n^*/y_n^*x_n \end{bmatrix}.$$

Proof of Lemma. We know that $AS = S\Lambda$, where $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, since the columns x_i of S are eigenvectors. This is equivalent to $S^{-1}A = \Lambda S^{-1}$, so the rows of S^{-1} are conjugate transposes of the left eigenvectors y_i. So

$$S^{-1} = \begin{bmatrix} y_1^* \cdot c_1 \\ \vdots \\ y_n^* \cdot c_n \end{bmatrix}$$

for some constants c_i. But $I = S^{-1}S$, so $1 = (S^{-1}S)_{ii} = y_i^* x_i \cdot c_i$, and $c_i = \frac{1}{y_i^* x_i}$ as desired. □

Lemma 4.2. *If each column of (any matrix) S has two-norm equal to* 1, $\|S\|_2 \leq \sqrt{n}$. *Similarly, if each row of a matrix has two-norm equal to* 1, *its two-norm is at most* $\sqrt{n}$.

Proof of Lemma. $\|S\|_2 = \|S^T\|_2 = \max_{\|x\|_2=1} \|S^T x\|_2$. Each component of $S^T x$ is bounded by 1 by the Cauchy–Schwartz inequality, so $\|S^T x\|_2 \leq \left\|[1, \ldots, 1]^T\right\|_2 = \sqrt{n}$. □

Proof of the Bauer–Fike theorem. We will apply Gershgorin's theorem to $S^{-1}(A + \delta A)S = \Lambda + F$, where $\Lambda = S^{-1}AS = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ and $F = S^{-1}\delta AS$. The idea is to show that the eigenvalues of $A + \delta A$ lie in balls centered at the λ_i with the given radii. To do this, we take the disks containing the eigenvalues of $\Lambda + F$ that are defined by Gershgorin's theorem,

$$|\lambda - (\lambda_i + f_{ii})| \leq \sum_{j \neq i} |f_{ij}|,$$

and enlarge them slightly to get the disks

$$\begin{aligned} |\lambda - \lambda_i| &\leq \sum_j |f_{ij}| \\ &\leq n^{1/2} \cdot \left(\sum_j |f_{ij}|^2 \right)^{1/2} \quad \text{by Cauchy–Schwarz} \\ &= n^{1/2} \cdot \|F(i,:)\|_2. \end{aligned} \tag{4.5}$$

Now we need to bound the two-norm of the ith row $F(i,:)$ of $F = S^{-1}\delta AS$:

$$\begin{aligned} \|F(i,:)\|_2 &= \|(S^{-1}\delta AS)(i,:)\|_2 \\ &\leq \|(S^{-1})(i,:)\|_2 \cdot \|\delta A\|_2 \cdot \|S\|_2 \quad \text{by Lemma 1.7} \\ &\leq \frac{n^{1/2}}{|y_i^* x_i|} \cdot \|\delta A\|_2 \quad \text{by Lemmas 4.1 and 4.2.} \end{aligned}$$

Combined with equation (4.5), this proves the theorem. □

We do not want to leave the impression that multiple eigenvalues cannot be computed with any accuracy at all just because they have infinite condition numbers. Indeed, we expect to get a *fraction* of the digits correct rather than lose a fixed number of digits. To illustrate, consider the 2-by-2 matrix with a double eigenvalue at 1: $A = \left[\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix} \right]$. If we perturb the (2,1) entry (the most sensitive) from 0 to machine epsilon ε, the eigenvalues change from 1 to $1 \pm \sqrt{\varepsilon}$. In other words the computed eigenvalues agree with the true eigenvalue to half precision. More generally, with a triple root, we expect to get about one third of the digits correct, and so on for higher multiplicities. See also Question 1.20.

We now turn to a geometric property of the condition number shared by other problems. Recall the property of the condition number $\|A\| \cdot \|A^{-1}\|$ for matrix inversion: its reciprocal measured the distance to nearest singular matrix, i.e., matrix with an infinite condition number (see Theorem 2.1). An analogous fact is true about eigenvalues. Since multiple eigenvalues have infinite condition numbers, the set of matrices with multiple eigenvalues plays the same role for computing eigenvalues as the singular matrices did for matrix inversion, where being "close to singular" implied ill-conditioning.

THEOREM 4.6. *Let λ be a simple eigenvalue of A, with unit right and left eigenvectors x and y and condition number $c = 1/|y^*x|$. Then there is a δA such that $A + \delta A$ has a multiple eigenvalue at λ, and*

$$\frac{\|\delta A\|_2}{\|A\|_2} \le \frac{1}{\sqrt{c^2-1}}.$$

When $c \gg 1$, i.e., the eigenvalue is ill-conditioned, then the upper bound on the distance is $1/\sqrt{c^2-1} \approx 1/c$, the reciprocal of the condition number.

Proof. First we show that we can assume without loss of generality that A is upper triangular (in Schur form), with $a_{11} = \lambda$. This is because putting A in Schur form is equivalent to replacing A by $T = Q^*AQ$, where Q is unitary. If x and y are eigenvectors of A, then Q^*x and Q^*y are eigenvectors of T. Since $(Q^*y)^*(Q^*x) = y^*QQ^*x = y^*x$, changing to Schur form does not change the condition number of λ. (Another way to say this is that the condition number is the secant of the angle $\Theta(x,y)$ between x and y, and changing x to Q^*x and y to Q^*y just rotates x and y the same way without changing the angle between them.)

So without loss of generality we can assume that $A = \left[\begin{smallmatrix} \lambda & A_{12} \\ 0 & A_{22} \end{smallmatrix} \right]$. Then $x = e_1$ and y is parallel to $\tilde{y} = [1, A_{12}(\lambda I - A_{22})^{-1}]^*$, or $y = \tilde{y}/\|\tilde{y}\|_2$. Thus

$$c = \frac{1}{|y^*x|} = \frac{\|\tilde{y}\|_2}{|\tilde{y}^*x|} = \|\tilde{y}\|_2 = (1 + \|A_{12}(\lambda I - A_{22})^{-1}\|_2^2)^{1/2}$$

or

$$\begin{aligned} \sqrt{c^2-1} &= \|A_{12}(\lambda I - A_{22})^{-1}\|_2 \le \|A_{12}\|_2 \cdot \|(\lambda I - A_{22})^{-1}\|_2 \\ &\le \frac{\|A\|_2}{\sigma_{\min}(\lambda I - A_{22})}. \end{aligned}$$

By definition of the smallest singular value, there is a δA_{22} where $\|\delta A_{22}\|_2 = \sigma_{\min}(\lambda I - A_{22})$ such that $A_{22} + \delta A_{22} - \lambda I$ is singular; i.e., λ is an eigenvalue of $A_{22} + \delta A_{22}$. Thus $\left[\begin{smallmatrix} \lambda & A_{12} \\ 0 & A_{22}+\delta A_{22} \end{smallmatrix} \right]$ has a double eigenvalue at λ, where

$$\|\delta A_{22}\|_2 = \sigma_{\min}(\lambda I - A_{22}) \le \frac{\|A\|_2}{\sqrt{c^2-1}}$$

as desired. □

Finally, we relate the condition numbers of the eigenvalues to the smallest possible condition number $\|S\| \cdot \|S^{-1}\|$ of any similarity S that diagonalizes A: $S^{-1}AS = \Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$. The theorem says that if any eigenvalue has a large condition number, then S has to have an approximately equally large condition number. In other words, the condition numbers for finding the (worst) eigenvalue and for reducing the matrix to diagonal form are nearly the same.

THEOREM 4.7. *Let A be diagonalizable with eigenvalues λ_i and right and left eigenvectors x_i and y_i, respectively, normalized so that $\|x_i\|_2 = \|y_i\|_2 = 1$. Let us suppose that S satisfies $S^{-1}AS = \Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$. Then $\|S\|_2 \cdot \|S^{-1}\|_2 \ge \max_i 1/|y_i^* x_i|$. If we choose $S = [x_1, \ldots, x_n]$, then $\|S\|_2 \cdot \|S^{-1}\|_2 \le n \cdot \max_i 1/|y_i^* x_i|$; i.e., the condition number of S is within a factor of n of its smallest value.*

For a proof, see [69].

For an overview of condition numbers for the eigenproblem, including eigenvectors, invariant subspaces, and the eigenvalues corresponding to an invariant subspace, see chapter 4 of the LAPACK manual [10], as well as [161, 237]. Algorithms for computing these condition numbers are available in subroutines `strsna` and `strsen` of LAPACK or by calling the driver routines `sgeevx` and `sgeesx`.

4.4. Algorithms for the Nonsymmetric Eigenproblem

We will build up to our ultimate algorithm, the shifted Hessenberg QR algorithm, by starting with simpler ones. For simplicity of exposition, we assume A is real.

Our first and simplest algorithm is the *power method* (section 4.4.1), which can find only the largest eigenvalue of A in absolute value and the corresponding eigenvector. To find the other eigenvalues and eigenvectors, we apply the power method to $(A - \sigma I)^{-1}$ for some *shift* σ, an algorithm called *inverse iteration* (section 4.4.2); note that the largest eigenvalue of $(A - \sigma I)^{-1}$ is $1/(\lambda_i - \sigma)$, where λ_i is the closest eigenvalue to σ, so we can choose which eigenvalues to find by choosing σ. Our next improvement to the power method lets us compute an entire invariant subspace at a time rather than just a single eigenvector;

we call this *orthogonal iteration* (section 4.4.3). Finally, we reorganize orthogonal iteration to make it convenient to apply to $(A-\sigma I)^{-1}$ instead of A; this is called QR iteration (section 4.4.4).

Mathematically speaking, QR iteration (with a shift σ) is our ultimate algorithm. But several problems remain to be solved to make it sufficiently fast and reliable for practical use (section 4.4.5). Section 4.4.6 discusses the first transformation designed to make QR iteration fast: reducing A from dense to *upper Hessenberg form* (nonzero only on and above the first subdiagonal). Subsequent sections describe how to implement QR iteration efficiently on upper Hessenberg matrices. (Section 4.4.7 shows how upper Hessenberg form simplifies in the cases of the symmetric eigenvalue problem and SVD.)

4.4.1. Power Method

ALGORITHM 4.1. *Power method: Given x_0, we iterate*

```
$i = 0$
repeat
    $y_{i+1} = Ax_i$
    $x_{i+1} = y_{i+1}/\|y_{i+1}\|_2$          (approximate eigenvector)
    $\tilde{\lambda}_{i+1} = x_{i+1}^T A x_{i+1}$          (approximate eigenvalue)
    $i = i+1$
until convergence
```

Let us first apply this algorithm in the very simple case when $A = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, with $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$. In this case the eigenvectors are just the columns e_i of the identity matrix. Note that x_i can also be written $x_i = A^i x_0/\|A^i x_0\|_2$, since the factors $1/\|y_{i+1}\|_2$ only scale x_{i+1} to be a unit vector and do not change its direction. Then we get

$$A^i x_0 \equiv A^i \begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_n \end{bmatrix} = \begin{bmatrix} \xi_1\lambda_1^i \\ \xi_2\lambda_2^i \\ \vdots \\ \xi_n\lambda_n^i \end{bmatrix} = \xi_1\lambda_1^i \begin{bmatrix} 1 \\ \frac{\xi_2}{\xi_1}\left(\frac{\lambda_2}{\lambda_1}\right)^i \\ \vdots \\ \frac{\xi_n}{\xi_1}\left(\frac{\lambda_n}{\lambda_1}\right)^i \end{bmatrix},$$

where we have assumed $\xi_1 \neq 0$. Since all the fractions λ_j/λ_1 are less than 1 in absolute value, $A^i x_0$ becomes more and more nearly parallel to e_1, so $x_i = A^i x_0/\|A^i x_0\|_2$ becomes closer and closer to $\pm e_1$, the eigenvector corresponding to the largest eigenvalue λ_1. The rate of convergence depends on how much smaller than 1 the ratios $|\lambda_2/\lambda_1| \geq \cdots \geq |\lambda_n/\lambda_1|$ are, the smaller the faster. Since x_i converges to $\pm e_1$, $\tilde{\lambda}_i = x_i^T A x_i$ converges to λ_1, the largest eigenvalue.

In showing that the power method converges, we have made several assumptions, most notably that A is diagonal. To analyze a more general case, we now assume that $A = S\Lambda S^{-1}$ is diagonalizable, with $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$

and the eigenvalues sorted so that $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$. Write $S = [s_1, \ldots, s_n]$, where the columns s_i are the corresponding eigenvectors and also satisfy $\|s_i\|_2 = 1$; in the last paragraph we had $S = I$. This lets us write $x_0 = S(S^{-1}x_0) \equiv S([\xi_1, \ldots, \xi_n]^T)$. Also, since $A = S\Lambda S^{-1}$, we can write

$$A^i = \underbrace{(S\Lambda S^{-1}) \cdots (S\Lambda S^{-1})}_{i \text{ times}} = S\Lambda^i S^{-1}$$

since all the $S^{-1} \cdot S$ pairs cancel. This finally lets us write

$$A^i x_0 = (S\Lambda^i S^{-1})S \begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_n \end{bmatrix} = S \begin{bmatrix} \xi_1 \lambda_1^i \\ \xi_2 \lambda_2^i \\ \vdots \\ \xi_n \lambda_n^i \end{bmatrix} = \xi_1 \lambda_1^i S \begin{bmatrix} 1 \\ \frac{\xi_2}{\xi_1}\left(\frac{\lambda_2}{\lambda_1}\right)^i \\ \vdots \\ \frac{\xi_n}{\xi_1}\left(\frac{\lambda_n}{\lambda_1}\right)^i \end{bmatrix}.$$

As before, the vector in brackets converges to e_1, so $A^i x_0$ gets closer and closer to a multiple of $Se_1 = s_1$, the eigenvector corresponding to λ_1. Therefore, $\tilde{\lambda}_i = x_i^T A x_i$ converges to $s_1^T A s_1 = s_1^T \lambda_1 s_1 = \lambda_1$.

A minor drawback of this method is the assumption that $\xi_1 \neq 0$, i.e., that x_0 is not the invariant subspace span$\{s_2, \ldots, s_n\}$; this is true with very high probability if x_0 is chosen at random. A major drawback is that it converges to the eigenvalue/eigenvector pair only for the eigenvalue of largest absolute magnitude, and its convergence rate depends on $|\lambda_2/\lambda_1|$, a quantity which may be close to 1 and thus cause very slow convergence. Indeed, if A is real and the largest eigenvalue is complex, there are two complex conjugate eigenvalues of largest absolute value $|\lambda_1| = |\lambda_2|$, and so the above analysis does not work at all. In the extreme case of an orthogonal matrix, *all* the eigenvalues have the same absolute value, namely, 1.

To plot the convergence of the power method, see HOMEPAGE/Matlab/powerplot.m.

4.4.2. Inverse Iteration

We will overcome the drawbacks of the power method just described by applying the power method to $(A - \sigma I)^{-1}$ instead of A, where σ is called a *shift*. This will let us converge to the eigenvalue closest to σ, rather than just λ_1. This method is called inverse iteration or the inverse power method.

ALGORITHM 4.2. *Inverse iteration: Given x_0, we iterate*

```
$i = 0$
repeat
    $y_{i+1} = (A - \sigma I)^{-1} x_i$
    $x_{i+1} = y_{i+1}/\|y_{i+1}\|_2$        (approximate eigenvector)
```

```
        $\tilde{\lambda}_{i+1} = x_{i+1}^T A x_{i+1}$        (approximate eigenvalue)
        $i = i + 1$
    until convergence
```

To analyze the convergence, note that $A = S\Lambda S^{-1}$ implies $A - \sigma I = S(\Lambda - \sigma I)S^{-1}$ and so $(A - \sigma I)^{-1} = S(\Lambda - \sigma I)^{-1}S^{-1}$. Thus $(A - \sigma I)^{-1}$ has the same eigenvectors s_i as A with corresponding eigenvalues $((\Lambda - \sigma I)^{-1})_{jj} = (\lambda_j - \sigma)^{-1}$. The same analysis as before tells us to expect x_i to converge to the eigenvector corresponding to the largest eigenvalue in absolute value. More specifically, assume that $|\lambda_k - \sigma|$ is smaller than all the other $|\lambda_i - \sigma|$ so that $(\lambda_k - \sigma)^{-1}$ is the largest eigenvalue in absolute value. Also, write $x_0 = S[\xi_1, \ldots, \xi_n]^T$ as before, and assume $\xi_k \neq 0$. Then

$$\begin{aligned}
(A - \sigma I)^{-i} x_0 &= (S(\Lambda - \sigma I)^{-i} S^{-1}) S \begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_n \end{bmatrix} = S \begin{bmatrix} \xi_1(\lambda_1 - \sigma)^{-i} \\ \vdots \\ \xi_n(\lambda_n - \sigma)^{-i} \end{bmatrix} \\
&= \xi_k (\lambda_k - \sigma)^{-i} S \begin{bmatrix} \frac{\xi_1}{\xi_k}\left(\frac{\lambda_k - \sigma}{\lambda_1 - \sigma}\right)^i \\ \vdots \\ 1 \\ \vdots \\ \frac{\xi_n}{\xi_k}\left(\frac{\lambda_k - \sigma}{\lambda_n - \sigma}\right)^i \end{bmatrix},
\end{aligned}$$

where the 1 is in entry k. Since all the fractions $(\lambda_k - \sigma)/(\lambda_i - \sigma)$ are less than one in absolute value, the vector in brackets approaches e_k, so $(A - \sigma I)^{-i} x_0$ gets closer and closer to a multiple of $Se_k = s_k$, the eigenvector corresponding to λ_k. As before, $\tilde{\lambda}_i = x_i^T A x_i$ also converges to λ_k.

The advantage of inverse iteration over the power method is the ability to converge to any desired eigenvalue (the one nearest the shift σ). By choosing σ very close to a desired eigenvalue, we can converge very quickly and thus not be as limited by the proximity of nearby eigenvalues as is the original power method. The method is particularly effective when we have a good approximation to an eigenvalue and want only its corresponding eigenvector (for example, see section 5.3.4). Later we will explain how to choose such a σ without knowing the eigenvalues, which is what we are trying to compute in the first place!

4.4.3. Orthogonal Iteration

Our next improvement will permit us to converge to a ($p > 1$)-dimensional invariant subspace, rather than one eigenvector at a time. It is called *orthogonal iteration* (and sometimes *subspace iteration* or *simultaneous iteration*).

ALGORITHM 4.3. *Orthogonal iteration: Let Z_0 be an $n \times p$ orthogonal matrix. Then we iterate*

```
$i = 0$
repeat
    $Y_{i+1} = AZ_i$
    Factor $Y_{i+1} = Z_{i+1}R_{i+1}$ using Algorithm 3.2 (QR decomposition)
                                      ($Z_{i+1}$ spans an approximate
                                      invariant subspace)
    $i = i + 1$
until convergence
```

Here is an informal analysis of this method. Assume $|\lambda_p| > |\lambda_{p+1}|$. If $p = 1$, this method and its analysis are identical to the power method. When $p > 1$, we write $\mathrm{span}\{Z_{i+1}\} = \mathrm{span}\{Y_{i+1}\} = \mathrm{span}\{AZ_i\}$, so $\mathrm{span}\{Z_i\} = \mathrm{span}\{A^i Z_0\} = \mathrm{span}\{S\Lambda^i S^{-1} Z_0\}$. Note that

$$\begin{aligned} S\Lambda^i S^{-1} Z_0 &= S \,\mathrm{diag}(\lambda_1^i, \ldots, \lambda_n^i) S^{-1} Z_0 \\ &= \lambda_p^i S \begin{bmatrix} (\lambda_1/\lambda_p)^i & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & (\lambda_n/\lambda_p)^i \end{bmatrix} S^{-1} Z_0. \end{aligned}$$

Since $|\frac{\lambda_j}{\lambda_p}| \geq 1$ if $j \leq p$, and $|\frac{\lambda_j}{\lambda_p}| < 1$ if $j > p$, we get

$$\begin{bmatrix} (\lambda_1/\lambda_p)^i & & \\ & \ddots & \\ & & (\lambda_n/\lambda_p)^i \end{bmatrix} S^{-1} Z_0 = \begin{bmatrix} V_i^{p \times p} \\ W_i^{(n-p) \times p} \end{bmatrix},$$

where W_i approaches zero like $(\lambda_{p+1}/\lambda_p)^i$, and V_i does not approach zero. Indeed, if V_0 has full rank (a generalization of the assumption in section 4.4.1 that $\xi_1 \neq 0$), then V_i will have full rank too. Write the matrix of eigenvectors $S = [s_1, \ldots, s_n] \equiv [S_p^{n \times p}, \hat{S}_p^{n \times (n-p)}]$, i.e., $S_p = [s_1, \ldots, s_p]$. Then $S\Lambda^i S^{-1} Z_0 = \lambda_p^i S[\begin{smallmatrix} V_i \\ W_i \end{smallmatrix}] = \lambda_p^i (S_p V_i + \hat{S}_p W_i)$. Thus

$$\mathrm{span}(Z_i) \;=\; \mathrm{span}(S\Lambda^i S^{-1} Z_0) = \mathrm{span}(S_p V_i + \hat{S}_p W_i) \Rightarrow \mathrm{span}(S_p X_i)$$

converges to $\mathrm{span}(S_p V_i) = \mathrm{span}(S_p)$, the invariant subspace spanned by the first p eigenvectors, as desired.

The use of the QR decomposition keeps the vectors spanning $\mathrm{span}\{A^i Z_0\}$ of full rank despite roundoff.

Note that if we follow only the first $\tilde{p} < p$ columns of Z_i through the iterations of the algorithm, they are *identical* to the columns that we would compute if we had started with only the first $\tilde{p}$ columns of Z_0 instead of p columns. In other words, orthogonal iteration is effectively running the algorithm for $\tilde{p} = 1, 2, \ldots, p$ all at the same time. So if *all* the eigenvalues have distinct absolute values, the same convergence analysis as before implies that the first $\tilde{p} \leq p$ columns of Z_i converge to $\text{span}\{s_1, \ldots, s_{\tilde{p}}\}$ for any $\tilde{p} \leq p$.

Thus, we can let $p = n$ and $Z_0 = I$ in the orthogonal iteration algorithm. The next theorem shows that under certain assumptions, we can use orthogonal iteration to compute the Schur form of A.

THEOREM 4.8. *Consider running orthogonal iteration on matrix A with $p = n$ and $Z_0 = I$. If all the eigenvalues of A have distinct absolute values and if all the principal submatrices $S(1:j, 1:j)$ have full rank, then $A_i \equiv Z_i^T A Z_i$ converges to the Schur form of A, i.e., an upper triangular matrix with the eigenvalues on the diagonal. The eigenvalues will appear in decreasing order of absolute value.*

Sketch of Proof. The assumption about nonsingularity of $S(1:j, 1:j)$ for all j implies that X_0 is nonsingular, as required by the earlier analysis. Geometrically, this means that no vector in the invariant subspace $\text{span}\{s_i, \ldots, s_j\}$ is orthogonal to $\text{span}\{e_i, \ldots, e_j\}$, the space spanned by the first j columns of $Z_0 I$. First note that Z_i is a square orthogonal matrix, so A and $A_i = Z_i^T A Z_i$ are similar. Write $Z_i = [Z_{1i}, Z_{2i}]$, where Z_{1i} has p columns, so

$$A_i = Z_i^T A Z_i = \begin{bmatrix} Z_{1i}^T A Z_{1i} & Z_{1i}^T A Z_{2i} \\ Z_{2i}^T A Z_{1i} & Z_{2i}^T A Z_{2i} \end{bmatrix}.$$

Since $\text{span}\{Z_{1i}\}$ converges to an invariant subspace of A, $\text{span}\{A Z_{1i}\}$ converges to the same subspace, so $Z_{2i}^T A Z_{1i}$ converges to $Z_{2i}^T Z_{1i} = 0$. Since this is true for all $p < n$, every subdiagonal entry of A_i converges to zero, so A_i converges to upper triangular form, i.e., Schur form. □

In fact, this proof shows that the submatrix $Z_{2i}^T A Z_{1i} = A_i(p+1:n, 1:p)$ should converge to zero like $|\lambda_{p+1}/\lambda_p|^i$. Thus, λ_p should appear as the (p,p) entry of A_i and converge like $\max(|\lambda_{p+1}/\lambda_p|^i, |\lambda_p/\lambda_{p-1}|^i)$.

EXAMPLE 4.5. The convergence behavior of orthogonal iteration is illustrated by the following numerical experiment, where we took $\Lambda = \text{diag}(1, 2, 6, 30)$ and a random S (with condition number about 20), formed $A = S \cdot \Lambda \cdot S^{-1}$, and ran orthogonal iteration on A with $p = 4$ for 19 iterations. Figures 4.3 and 4.4 show the convergence of the algorithm. Figure 4.3 plots the actual errors $|A_i(p,p) - \lambda_p|$ in the computed eigenvalues as solid lines and the approximations $\max(|\lambda_{p+1}/\lambda_p|^i, |\lambda_p/\lambda_{p-1}|^i)$ as dotted lines. Since the graphs are (essentially) straight lines with the same slope on a semilog scale, this means that they are both graphs of functions of the form $y = c \cdot r^i$, where c and r are constants and r (the slope) is the same for both, as we predicted above.

Similarly, Figure 4.4 plots the actual values $\|A_i(p+1:n,1:p)\|_2$ as solid lines and the approximations $|\lambda_{p+1}/\lambda_p|^i$ as dotted lines; they also match well. Here are A_0 and A_{19} for comparison:

$$\begin{aligned} A = A_0 &= \begin{bmatrix} 3.5488 & 15.593 & 8.5775 & -4.0123 \\ 2.3595 & 24.526 & 14.596 & -5.8157 \\ 8.9953\cdot 10^{-2} & 27.599 & 21.483 & -5.8415 \\ 1.9227 & 55.667 & 39.717 & -10.558 \end{bmatrix}, \\ A_{19} &= \begin{bmatrix} 30.000 & -32.557 & -70.844 & 14.984 \\ 6.7607\cdot 10^{-13} & 6.0000 & 1.8143 & -.55754 \\ 1.5452\cdot 10^{-23} & 1.1086\cdot 10^{-9} & 2.0000 & -.25894 \\ 7.3360\cdot 10^{-29} & 3.3769\cdot 10^{-15} & 4.9533\cdot 10^{-6} & 1.0000 \end{bmatrix}. \end{aligned}$$

See HOMEPAGE/Matlab/qriter.m for Matlab software to run this and similar examples. ◇

EXAMPLE 4.6. To see why the assumption in Theorem 4.8 about nonsingularity of $S(1:j,1:j)$ is necessary, suppose that A is diagonal with the eigenvalues *not* in decreasing order on the diagonal. Then orthogonal iteration yields $Z_i = \mathrm{diag}(\pm 1)$ (a diagonal matrix with diagonal entries ± 1) and $A_i = A$ for all i, so the eigenvalues do not move into decreasing order. To see why the assumption that the eigenvalues have distinct absolute values is necessary, suppose that A is orthogonal, so all its eigenvalues have absolute value 1. Again, the algorithm leaves A_i essentially unchanged. (The rows and columns may be multiplied by -1.)

4.4.4. QR Iteration

Our next goal is to reorganize orthogonal iteration to incorporate shifting and inverting, as in section 4.4.2. This will make it more efficient and eliminate the assumption that eigenvalues differ in magnitude, which was needed in Theorem 4.8 to prove convergence.

ALGORITHM 4.4. *QR iteration: Given A_0, we iterate*

```
$i = 0$
repeat
    Factor $A_i = Q_i R_i$        (the QR decomposition)
    $A_{i+1} = R_i Q_i$
    $i = i + 1$
until convergence
```

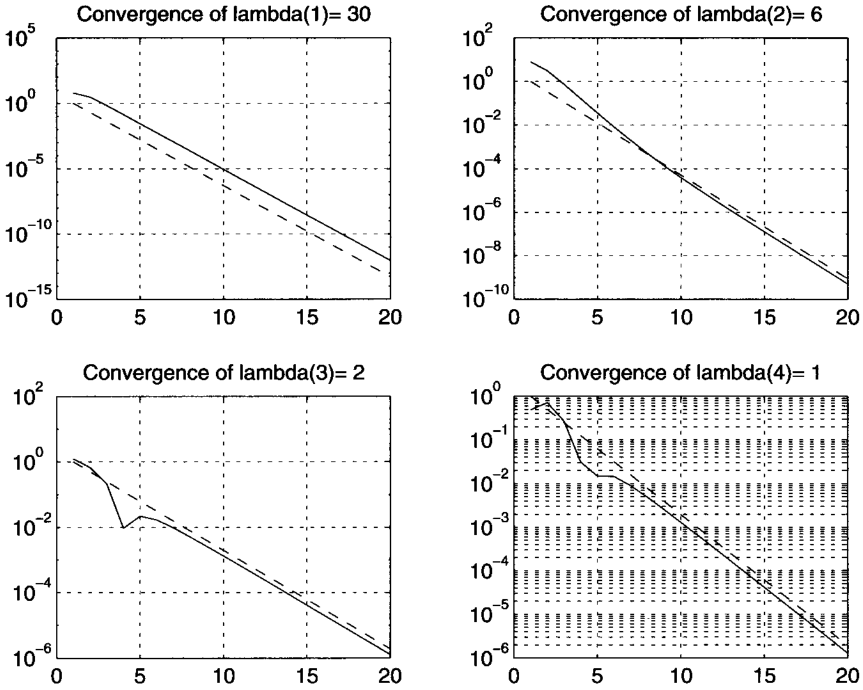

Fig. 4.3. *Convergence of diagonal entries during orthogonal iteration.*

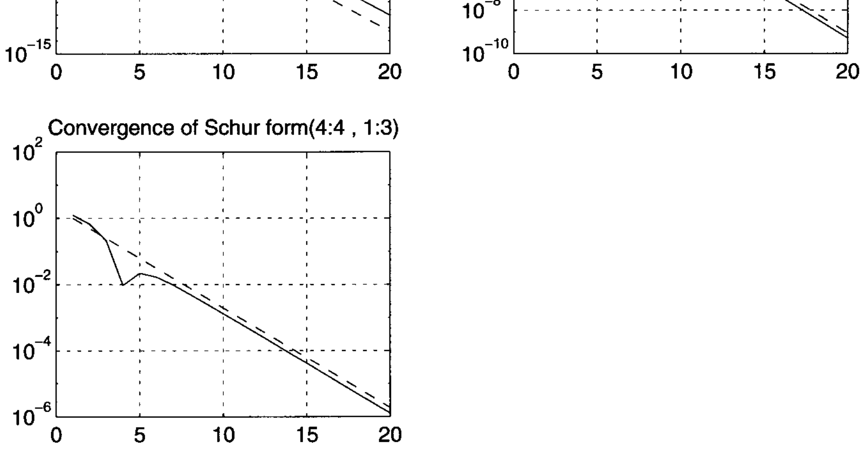

Fig. 4.4. *Convergence to Schur form during orthogonal iteration.*

Since $A_{i+1} = R_iQ_i = Q_i^T(Q_iR_i)Q_i = Q_i^TA_iQ_i$, A_{i+1} and A_i are orthogonally similar.

We claim that the A_i computed by QR iteration is identical to the matrix $Z_i^TAZ_i$ implicitly computed by orthogonal iteration.

LEMMA 4.3. *Let A_i be the matrix computed by Algorithm 4.4. Then $A_i = Z_i^TAZ_i$, where Z_i is the matrix computed from orthogonal iteration (Algorithm 4.3) starting with $Z_0 = I$. Thus A_i converges to Schur form if all the eigenvalues have different absolute values.*

Proof. We use induction. Assume $A_i = Z_i^TAZ_i$. From Algorithm 4.3, we can write $AZ_i = Z_{i+1}R_{i+1}$, where Z_{i+1} is orthogonal and R_{i+1} is upper triangular. Then $Z_i^TAZ_i = Z_i^T(Z_{i+1}R_{i+1})$ is the product of an orthogonal matrix $Q = Z_i^TZ_{i+1}$ and an upper triangular matrix $R = R_{i+1} = Z_{i+1}^TAZ_i$; this must be the QR decomposition $A_i = QR$, since the QR decomposition is unique (except for possibly multiplying each column of Q and row of R by -1). Then

$$Z_{i+1}^TAZ_{i+1} = (Z_{i+1}^TAZ_i)(Z_i^TZ_{i+1}) = R_{i+1}(Z_i^TZ_{i+1}) = RQ.$$

This is precisely how the QR iteration maps A_i to A_{i+1}, so $Z_{i+1}^TAZ_{i+1} = A_{i+1}$ as desired. □

To see a variety of numerical examples illustrating the convergence of QR iteration, run the Matlab code referred to in Question 4.15.

From earlier analysis, we know that the convergence rate depends on the ratios of eigenvalues. To speed convergence, we use shifting and inverting.

ALGORITHM 4.5. *QR iteration with a shift: Given A_0, we iterate*

```
$i = 0$
repeat
    Choose a shift $\sigma_i$ near an eigenvalue of $A$
    Factor $A_i - \sigma_i I = Q_iR_i$   (QR decomposition)
    $A_{i+1} = R_iQ_i + \sigma_i I$
    $i = i + 1$
until convergence
```

LEMMA 4.4. *A_i and A_{i+1} are orthogonally similar.*

Proof. $A_{i+1} = R_iQ_i + \sigma_iI = Q_i^TQ_iR_iQ_i + \sigma_iQ_i^TQ_i = Q_i^T(Q_iR_i + \sigma_iI)Q_i = Q_i^TA_iQ_i$. □

If R_i is nonsingular, we may also write

$$\begin{aligned} A_{i+1} &= R_iQ_i + \sigma_iI = R_iQ_iR_iR_i^{-1} + \sigma_iR_iR_i^{-1} = R_i(Q_iR_i + \sigma_iI)R_i^{-1} \\ &= R_iA_iR_i^{-1}. \end{aligned}$$

If σ_i is an *exact* eigenvalue of A_i, then we claim that QR iteration converges in one step: since σ_i is an eigenvalue, $A_i - \sigma_i I$ is singular, so R_i is singular, and so some diagonal entry of R_i must be zero. Suppose $R_i(n,n) = 0$. This implies that the last row of $R_i Q_i$ is 0, so the last row of $A_{i+1} = R_i Q_i + \sigma_i I$ equals $\sigma_i e_n^T$, where e_n is the nth column of the n-by-n identity matrix. In other words, the last row of A_{i+1} is zero except for the eigenvalue σ_i appearing in the (n,n) entry. This means that the algorithm has *converged*, because A_{i+1} is block upper triangular, with a trailing 1-by-1 block σ_i; the leading $(n-1)$-by-$(n-1)$ block A' is a new, smaller eigenproblem to which QR iteration can be solved without ever modifying σ_i again: $A_{i+1} = \left[\begin{smallmatrix} A' & a \\ 0 & \sigma_i \end{smallmatrix}\right]$.

When σ_i is not an exact eigenvalue, then we will accept $A_{i+1}(n,n)$ as having converged when the lower left block $A_{i+1}(n, 1:n-1)$ is small enough. Recall from our earlier analysis that we expect $A_{i+1}(n, 1:n-1)$ to shrink by a factor $|\lambda_k - \sigma_i| / \min_{j \neq k} |\lambda_j - \sigma_i|$, where $|\lambda_k - \sigma_i| = \min_j |\lambda_j - \sigma_i|$. So if σ_i is a very good approximation to eigenvalue λ_k, we expect fast convergence.

Here is another way to see why convergence should be fast, by recognizing that QR iteration is implicitly doing inverse iteration. When σ_i is an exact eigenvalue, the last column $\underline{q}_i$ of Q_i will be a left eigenvector of A_i for eigenvalue σ_i, since $\underline{q}_i^* A_i = \underline{q}_i^*(Q_i R_i + \sigma_i I) = e_n^T R_i + \sigma_i \underline{q}_i^* = \sigma_i \underline{q}_i^*$. When σ_i is close to an eigenvalue, we expect $\underline{q}_i$ to be close to an eigenvector for the following reason: $\underline{q}_i$ is parallel to $((A_i - \sigma_i I)^*)^{-1} e_n$ (we explain why below). In other words $\underline{q}_i$ is the same as would be obtained from inverse iteration on $(A_i - \sigma_i I)^*$ (and so we expect it to be close to a left eigenvector).

Here is the proof that $\underline{q}_i$ is parallel to $((A_i - \sigma_i I)^*)^{-1} e_n$. $A_i - \sigma_i I = Q_i R_i$ implies $(A_i - \sigma_i I) R_i^{-1} = Q_i$. Inverting and taking the conjugate transpose of both sides leave the right-hand side Q_i unchanged and change the left-hand side to $((A_i - \sigma_i I)^*)^{-1} R_i^*$, whose last column is $((A_i - \sigma_i I)^*)^{-1} \cdot [0, \ldots, 0, R_i(n,n)]^T$, which is proportional to the last column of $((A_i - \sigma_i I)^*)^{-1}$.

How do we choose σ_i to be an accurate approximate eigenvalue, when we are trying to compute eigenvalues in the first place? We will say more about this later, but for now note that near convergence to a *real* eigenvalue $A_i(n,n)$ is close to that eigenvalue, so $\sigma_i = A_i(n,n)$ is a good choice of shift. In fact, it yields local *quadratic convergence*, which means that the number of correct digits *doubles* at every step. We explain why quadratic convergence occurs as follows: Suppose at step i that $\|A_i(n, 1:n-1)\| / \|A\| \equiv \eta \ll 1$. If we were to set $A_i(n, 1:n-1)$ to exactly 0, we would make A_i block upper triangular and so perturb a true eigenvalue λ_k to make it equal to $A_i(n,n)$. If this eigenvalue is far from the other eigenvalues, it will not be ill-conditioned, so this perturbation will be $O(\eta \|A\|)$. In other words, $|\lambda_k - A_i(n,n)| = O(\eta \|A\|)$. On the next iteration, if we choose $\sigma_i = A_i(n,n)$, we expect $A_{i+1}(n, 1:n-1)$ to shrink by a factor $|\lambda_k - \sigma_i| / \min_{j \neq k} |\lambda_j - \sigma_i| = O(\eta)$, implying that $\|A_{i+1}(n, 1:n-1)\| = O(\eta^2 \|A\|)$, or $\|A_{i+1}(n, 1:n-1)\| / \|A\| = O(\eta^2)$. Decreasing the error this way from η to $O(\eta^2)$ is quadratic convergence.

EXAMPLE 4.7. Here are some shifted QR iterations starting with the same 4-by-4 matrix A_0 as in Example 4.5, with shift $\sigma_i = A_i(4,4)$. The convergence is a bit erratic at first but eventually becomes quadratic near the end, with $\|A_i(4,1:3)\| \approx |A_i(4,3)|$ approximately squaring at each of the last three steps. Also, the number of correct digits in $A_i(4,4)$ doubles at the fourth through second-to-last steps.

$A_0(4,:) =$	$+1.9$	$+56.$	$+40.$	-10.558
$A_1(4,:) =$	$-.85$	-4.9	$+2.2\cdot10^{-2}$	-6.6068
$A_2(4,:) =$	$+.35$	$+.86$	$+.30$	0.74894
$A_3(4,:) =$	$-1.2\cdot10^{-2}$	$-.17$	$-.70$	1.4672
$A_4(4,:) =$	$-1.5\cdot10^{-4}$	$-1.8\cdot10^{-2}$	$-.38$	1.4045
$A_5(4,:) =$	$-3.0\cdot10^{-6}$	$-2.2\cdot10^{-3}$	$-.50$	1.1403
$A_6(4,:) =$	$-1.4\cdot10^{-8}$	$-6.3\cdot10^{-5}$	$-7.8\cdot10^{-2}$	1.0272
$A_7(4,:) =$	$-1.4\cdot10^{-11}$	$-3.6\cdot10^{-7}$	$-2.3\cdot10^{-3}$	0.99941
$A_8(4,:) =$	$+2.8\cdot10^{-16}$	$+4.2\cdot10^{-11}$	$+1.4\cdot10^{-6}$	0.9999996468853453
$A_9(4,:) =$	$-3.4\cdot10^{-24}$	$-3.0\cdot10^{-18}$	$-4.8\cdot10^{-13}$	0.9999999999998767
$A_{10}(4,:) =$	$+1.5\cdot10^{-38}$	$+7.4\cdot10^{-32}$	$+6.0\cdot10^{-26}$	1.000000000000001

By the time we reach A_{10}, the rest of the matrix has made a lot of progress toward convergence as well, so later eigenvalues will be computed very quickly, in one or two steps each:

$$A_{10} = \begin{bmatrix} 30.000 & -32.557 & -70.844 & 14.985 \\ 6.1548\cdot10^{-6} & 6.0000 & 1.8143 & -.55754 \\ 2.5531\cdot10^{-13} & 2.0120\cdot10^{-6} & 2.0000 & -.25894 \\ 1.4692\cdot10^{-38} & 7.4289\cdot10^{-32} & 6.0040\cdot10^{-26} & 1.0000 \end{bmatrix}. \quad \diamond$$

4.4.5. Making QR Iteration Practical

Here are some remaining problems we have to solve to make the algorithm more practical:

1. The iteration is too expensive. The QR decomposition costs $O(n^3)$ flops, so if we were lucky enough to do only one iteration per eigenvalue, the cost would be $O(n^4)$. But we seek an algorithm with a total cost of only $O(n^3)$.

2. How shall we choose σ_i to accelerate convergence to a complex eigenvalue? Choosing σ_i complex means all arithmetic has to be complex, increasing the cost by a factor of about 4 when A is real. We seek an algorithm that uses all real arithmetic if A is real and converges to real Schur form.

3. How do we recognize convergence?

The solutions to these problems, which we will describe in more detail later, are as follows:

1. We will initially reduce the matrix to *upper Hessenberg form*; this means that A is zero below the first subdiagonal (i.e., $a_{ij} = 0$ if $i > j + 1$) (see section 4.4.6). Then we will apply a step of QR iteration *implicitly*, i.e., without computing Q or multiplying by it explicitly (see section 4.4.8). This will reduce the cost of one QR iteration from $O(n^3)$ to $O(n^2)$ and the overall cost from $O(n^4)$ to $O(n^3)$ as desired

   When A is symmetric we will reduce it to tridiagonal form instead, reducing the cost of a single QR iteration further to $O(n)$. This is discussed in section 4.4.7 and Chapter 5.

2. Since complex eigenvalues of real matrices occur in complex conjugate pairs, we can shift by σ_i and $\bar{\sigma}_i$ simultaneously; it turns out that this will permit us to maintain real arithmetic (see section 4.4.8). If A is symmetric, all eigenvalues are real, and this is not an issue.

3. Convergence occurs when subdiagonal entries of A_i are "small enough." To help choose a practical threshold, we use the notion of backward stability: Since A_i is related to A by a similarity transformation by an orthogonal matrix, we expect A_i to have roundoff errors of size $O(\varepsilon\|A\|)$ in it anyway. Therefore, any subdiagonal entry of A_i smaller than $O(\varepsilon\|A\|)$ in magnitude may as well be zero, so we set it to zero.[16] When A is upper Hessenberg, setting $a_{p+1,p}$ to zero will make A into a block upper triangular matrix $A = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}$, where A_{11} is p-by-p and A_{11} and A_{22} are both Hessenberg. Then the eigenvalues of A_{11} and A_{22} may be found independently to get the eigenvalues of A. When all these diagonal blocks are 1-by-1 or 2-by-2, the algorithm has finished.

4.4.6. Hessenberg Reduction

Given a real matrix A, we seek an orthogonal Q so that QAQ^T is upper Hessenberg. The algorithm is a simple variation on the idea used for the QR decomposition.

EXAMPLE 4.8. We illustrate the general pattern of Hessenberg reduction with a 5-by-5 example. Each Q_i below is a 5-by-5 Householder reflection, chosen to zero out entries $i + 2$ through n in column i and leaving entries 1 through i unchanged.

[16] In practice, we use a slightly more stringent condition, replacing $\|A\|$ with the norm of a submatrix of A, to take into account matrices which may be "graded" with large entries in one place and small entries elsewhere. We can also set a subdiagonal entry to zero when the product $a_{p+1,p}a_{p+2,p+1}$ of two adjacent subdiagonal entries is small enough. See the LAPACK routine `slahqr` for details.

1. Choose Q_1 so

$$Q_1A = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ o & x & x & x & x \\ o & x & x & x & x \\ o & x & x & x & x \end{bmatrix} \quad \text{and } A_1 \equiv Q_1AQ_1^T = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ o & x & x & x & x \\ o & x & x & x & x \\ o & x & x & x & x \end{bmatrix}.$$

   Q_1 leaves the first row of Q_1A unchanged, and Q_1^T leaves the first column of $Q_1AQ_1^T$ unchanged, including the zeros.

2. Choose Q_2 so

$$Q_2A_1 = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ o & x & x & x & x \\ o & o & x & x & x \\ o & o & x & x & x \end{bmatrix} \quad \text{and } A_2 \equiv Q_2A_1Q_2^T = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ o & x & x & x & x \\ o & o & x & x & x \\ o & o & x & x & x \end{bmatrix}.$$

   Q_2 changes only the last three rows of A_1, and Q_2^T leaves the first two columns of $Q_2A_1Q_2^T$ unchanged, including the zeros.

3. Choose Q_3 so

$$Q_3A_2 = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ o & x & x & x & x \\ o & o & x & x & x \\ o & o & o & x & x \end{bmatrix} \quad \text{and } A_3 = Q_3A_2Q_3^T = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ o & x & x & x & x \\ o & o & x & x & x \\ o & o & o & x & x \end{bmatrix},$$

   which is upper Hessenberg. Altogether $A_3 = (Q_3Q_2Q_1) \cdot A(Q_3Q_2Q_1)^T \equiv QAQ^T$. ◇

The general algorithm for Hessenberg reduction is as follows.

ALGORITHM 4.6. *Reduction to upper Hessenberg form:*

```
if Q is desired, set $Q = I$
for $i = 1 : n - 2$
    $u_i = \mathrm{House}(A(i+1 : n, i))$
    $P_i = I - 2u_iu_i^T$        /* $Q_i = \mathrm{diag}(I^{i \times i}, P_i)$ */
    $A(i+1 : n, i : n) = P_i \cdot A(i+1 : n, i : n)$
    $A(1 : n, i+1 : n) = A(1 : n, i+1 : n) \cdot P_i$
    if Q is desired
        $Q(i+1 : n, i : n) = P_i \cdot Q(i+1 : n, i : n)$        /* $Q = Q_i \cdot Q$ */
    end if
end for
```

As with the QR decomposition, one does not form P_i explicitly but instead multiplies by $I - 2u_iu_i^T$ via matrix-vector operations. The u_i vectors can also be stored below the subdiagonal, similar to the QR decomposition. They can be applied using Level 3 BLAS, as described in Question 3.17. This algorithm is available as the Matlab command `hess` or the LAPACK routine `sgehrd`.

The number of floating point operations is easily counted to be $\frac{10}{3}n^3 + O(n^2)$, or $\frac{14}{3}n^3 + O(n^2)$ if the product $Q = Q_{n-1} \cdots Q_1$ is computed as well.

The advantage of Hessenberg form under QR iteration is that it costs only $6n^2 + O(n)$ flops per iteration instead of $O(n^3)$, and its form is preserved so that the matrix remains upper Hessenberg.

PROPOSITION 4.5. *Hessenberg form is preserved by QR iteration.*

Proof. It is easy to confirm that the QR decomposition of an upper Hessenberg matrix like $A_i - \sigma I$ yields an upper Hessenberg Q (since the jth column of Q is a linear combination of the leading j columns of $A_i - \sigma I$). Then it is easy to confirm that RQ remains upper Hessenberg and adding σI does not change this. □

DEFINITION 4.5. *An upper Hessenberg matrix H is* unreduced *if all subdiagonals are nonzero.*

It is easy to see that if H is reduced because $h_{i+1,i} = 0$, then its eigenvalues are those of its leading i-by-i Hessenberg submatrix and its trailing $(n-i)$-by-$(n-i)$ Hessenberg submatrix, so we need consider only unreduced matrices.

4.4.7. Tridiagonal and Bidiagonal Reduction

If A is symmetric, the Hessenberg reduction process leaves A symmetric at each step, so zeros are created in symmetric positions. This means we need work on only half the matrix, reducing the operation count to $\frac{4}{3}n^3 + O(n^2)$ or $\frac{8}{3}n^3 + O(n^2)$ to form $Q_{n-1} \ldots Q_1$ as well. We call this algorithm *tridiagonal reduction*. We will use this algorithm in Chapter 5. This routine is available as LAPACK routine `ssytrd`.

Looking ahead a bit to our discussion of computing the SVD in section 5.4, we recall from section 3.2.3 that the eigenvalues of the symmetric matrix A^TA are the squares of the singular values of A. Our eventual SVD algorithm will use this fact, so we would like to find a form for A which implies that A^TA is tridiagonal. We will choose A to be *upper bidiagonal*, or nonzero only on the diagonal and first superdiagonal. Thus, we want to compute orthogonal matrices Q and V such that QAV is bidiagonal. The algorithm, called *bidiagonal reduction*, is very similar to Hessenberg and tridiagonal reduction.

EXAMPLE 4.9. Here is a 4-by-4 example of bidiagonal reduction, which illustrates the general pattern:

1. Choose Q_1 so

$$Q_1A = \begin{bmatrix} x & x & x & x \\ o & x & x & x \\ o & x & x & x \\ o & x & x & x \end{bmatrix} \text{ and } V_1 \text{ so } A_1 \equiv Q_1AV_1 = \begin{bmatrix} x & x & o & o \\ o & x & x & x \\ o & x & x & x \\ o & x & x & x \end{bmatrix}.$$

   Q_1 is a Householder reflection, and V_1 is a Householder reflection that leaves the first column of Q_1A unchanged.

2. Choose Q_2 so

$$Q_2A_1 = \begin{bmatrix} x & x & o & o \\ o & x & x & x \\ o & o & x & x \\ o & o & x & x \end{bmatrix} \text{ and } V_2 \text{ so } A_2 \equiv Q_2A_1V_2 = \begin{bmatrix} x & x & o & o \\ o & x & x & o \\ o & o & x & x \\ o & o & x & x \end{bmatrix}.$$

   Q_2 is a Householder reflection that leaves the first row of A_1 unchanged. V_2 is a Householder reflection that leaves the first two columns of Q_2A_1 unchanged.

3. Choose Q_3 so

$$Q_3A_2 = \begin{bmatrix} x & x & o & o \\ o & x & x & o \\ o & o & x & x \\ o & o & o & x \end{bmatrix} \text{ and } V_3 = I \text{ so } A_3 = Q_3A_2.$$

   Q_3 is a Householder reflection that leaves the first two rows of A_2 unchanged. ◇

In general, if A is n-by-n, then we get orthogonal matrices $Q = Q_{n-1} \cdots Q_1$ and $V = V_1 \cdots V_{n-2}$ such that $QAV = A'$ is upper bidiagonal.

Note that $A'^TA' = V^TA^TQ^TQAV = V^TA^TAV$, so A'^TA' has the same eigenvalues as A^TA; i.e., A' has the same singular values as A.

The cost of this bidiagonal reduction is $\frac{8}{3}n^3 + O(n^2)$ flops, plus another $4n^3 + O(n^2)$ flops to compute Q and V. This routine is available as LAPACK routine `sgebrd`.

4.4.8. QR Iteration with Implicit Shifts

In this section we show how to implement QR iteration cheaply on an upper Hessenberg matrix. The implementation will be *implicit* in the sense that we do not explicitly compute the QR factorization of a matrix H but rather construct Q implicitly as a product of Givens rotations and other simple orthogonal matrices. The *implicit Q theorem* described below shows that this implicitly constructed Q is the Q we want. Then we show how to incorporate a single shift σ, which is necessary to accelerate convergence. To retain real arithmetic in the presence of complex eigenvalues, we then show how to do a *double* shift, i.e., combine two consecutive QR iterations with complex conjugate shifts σ and $\bar{\sigma}$; the result after this double shift is again real. Finally, we discuss strategies for choosing shifts σ and $\bar{\sigma}$ to provide reliable quadratic convergence. However, there have been recent discoveries of rare situation where convergence does not occur [25, 65], so finding a completely reliable and fast implementation of QR iteration remains an open problem.

Implicit Q Theorem

Our eventual implementation of QR iteration will depend on the following theorem.

THEOREM 4.9. Implicit Q theorem. *Suppose that $Q^TAQ = H$ is unreduced upper Hessenberg. Then columns 2 through n of Q are determined uniquely (up to signs) by the first column of Q.*

This theorem implies that to compute $A_{i+1} = Q_i^T A_i Q_i$ from A_i in the QR algorithm, we will need only to

1. compute the first column of Q_i (which is parallel to the first column of $A_i - \sigma_i I$ and so can be gotten just by normalizing this column vector).

2. choose other columns of Q_i so Q_i is orthogonal and A_{i+1} is unreduced Hessenberg.

Then by the implicit Q theorem, we know that we will have computed A_{i+1} correctly because Q_i is unique up to signs, which do not matter. (Signs do not matter because changing the signs of the columns of Q_i is the same as changing $A_i - \sigma_i I = Q_i R_i$ to $(Q_i S_i)(S_i R_i)$, where $S_i = \text{diag}(\pm 1, \ldots, \pm 1)$. Then $A_{i+1} = (S_i R_i)(Q_i S_i) + \sigma_i I = S_i(R_i Q_i + \sigma_i I)S_i$, which is an orthogonal similarity that just changes the signs of the columns and rows of A_{i+1}.)

Proof of the implicit Q theorem. Suppose that $Q^TAQ = H$ and $V^TAV = G$ are unreduced upper Hessenberg, Q and V are orthogonal, and the first columns of Q and V are equal. Let $(X)_i$ denote the ith column of X. We wish to show $(Q)_i = \pm(V)_i$ for all $i > 1$, or equivalently, that $W \equiv V^TQ = \text{diag}(\pm 1, \ldots, \pm 1)$.

Since $W = V^TQ$, we get $GW = GV^TQ = V^TAQ = V^TQH = WH$. Now $GW = WH$ implies $G(W)_i = (GW)_i = (WH)_i = \sum_{j=1}^{i+1} h_{ji}(W)_j$, so $h_{i+1,i}(W)_{i+1} = G(W)_i - \sum_{j=1}^{i} h_{ji}(W)_j$. Since $(W)_1 = [1, 0, \ldots, 0]^T$ and G is upper Hessenberg, we can use induction on i to show that $(W)_i$ is nonzero in entries 1 to i only; i.e., W is upper triangular. Since W is also orthogonal, W is diagonal $= \text{diag}(\pm 1, \ldots, \pm 1)$. □

Implicit Single Shift QR Algorithm

To see how to use the implicit Q theorem to compute A_1 from $A_0 = A$, we use a 5-by-5 example.

EXAMPLE 4.10. 1. Choose

$$Q_1^T = \begin{bmatrix} c_1 & s_1 & & & \\ -s_1 & c_1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \quad \text{so } A_1 \equiv Q_1^T A Q_1 = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ + & x & x & x & x \\ o & o & x & x & x \\ o & o & o & x & x \end{bmatrix}.$$

We discuss how to choose c_1 and s_1 below; for now they may be any Givens rotation. The + in position (3,1) is called a *bulge* and needs to be gotten rid of to restore Hessenberg form.

2. Choose

$$Q_2^T = \begin{bmatrix} 1 & & & & \\ & c_2 & s_2 & & \\ & -s_2 & c_2 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \quad \text{so } Q_2^T A_1 = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ o & x & x & x & x \\ o & o & x & x & x \\ o & o & o & x & x \end{bmatrix}$$

and

$$A_2 \equiv Q_2^T A_1 Q_2 = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ o & x & x & x & x \\ o & + & x & x & x \\ o & o & o & x & x \end{bmatrix}.$$

Thus the bulge has been "chased" from (3,1) to (4,2).

3. Choose

$$Q_3^T = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & c_3 & s_3 & \\ & & -s_3 & c_3 & \\ & & & & 1 \end{bmatrix} \quad \text{so } Q_3^T A_2 = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ o & x & x & x & x \\ o & o & x & x & x \\ o & o & o & x & x \end{bmatrix}$$

and

$$A_3 \equiv Q_3^T A_2 Q_3 = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ o & x & x & x & x \\ o & o & x & x & x \\ o & o & + & x & x \end{bmatrix}.$$

The bulge has been chased from (4,2) to (5,3).

4. Choose

$$Q_4^T = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & c_4 & s_4 \\ & & & -s_4 & c_4 \end{bmatrix} \text{ so } Q_4^T A_3 = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ o & x & x & x & x \\ o & o & x & x & x \\ o & o & o & x & x \end{bmatrix}$$

and

$$A_4 = Q_4^T A_3 Q_4 = \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ o & x & x & x & x \\ o & o & x & x & x \\ o & o & o & x & x \end{bmatrix},$$

so we are back to upper Hessenberg form.

Altogether $Q^T AQ$ is upper Hessenberg, where

$$Q = Q_1 Q_2 Q_3 Q_4 = \begin{bmatrix} c_1 & x & x & x & x \\ s_1 & x & x & x & x \\ & s_2 & x & x & x \\ & & s_3 & x & x \\ & & & s_4 & x \end{bmatrix},$$

so the first column of Q is $[c_1, s_1, 0, \ldots, 0]^T$, which by the implicit Q theorem has uniquely determined the other columns of Q (up to signs). We now choose the first column of Q to be proportional to the first column of $A - \sigma I$, $[a_{11} - \sigma, a_{21}, 0, \ldots, 0]^T$. This means Q is the same as in the QR decomposition of $A - \sigma I$, as desired. ◇

The cost of one implicit QR iteration for an n-by-n matrix is $6n^2 + O(n)$.

Implicit Double Shift QR Algorithm

This section describes how to maintain real arithmetic by shifting by σ and $\overline{\sigma}$ at the same time. This is essential for an efficient practical implementation but not for a mathematical understanding of the algorithm and may be skipped on a first reading.

The results of shifting by σ and $\bar{\sigma}$ in succession are

$$\begin{aligned} A_0 - \sigma I &= Q_1 R_1, \\ A_1 &= R_1 Q_1 + \sigma I \quad \text{so } A_1 = Q_1^T A_0 Q_1, \\ A_1 - \overline{\sigma} I &= Q_2 R_2, \\ A_2 &= R_2 Q_2 + \overline{\sigma} I \quad \text{so } A_2 = Q_2^T A_1 Q_2 = Q_2^T Q_1^T A_0 Q_1 Q_2. \end{aligned}$$

LEMMA 4.5. *We can choose Q_1 and Q_2 so*

(1) *Q_1Q_2 is real,*
(2) *A_2 is therefore real,*
(3) *the first column of Q_1Q_2 is easy to compute.*

Proof. Since $Q_2R_2 = A_1 - \overline{\sigma}I = R_1Q_1 + (\sigma - \overline{\sigma})I$, we get

$$\begin{aligned} Q_1Q_2R_2R_1 &= Q_1(R_1Q_1 + (\sigma - \bar{\sigma})I)R_1 \\ &= Q_1R_1Q_1R_1 + (\sigma - \overline{\sigma})Q_1R_1 \\ &= (A_0 - \sigma I)(A_0 - \sigma I) + (\sigma - \overline{\sigma})(A_0 - \sigma I) \\ &= A_0^2 - 2(\Re\sigma)A_0 + |\sigma|^2 I \equiv M. \end{aligned}$$

Thus $(Q_1Q_2)(R_2R_1)$ is the QR decomposition of the real matrix M, and therefore Q_1Q_2, as well as R_2R_1, can be chosen real. This means that $A_2 = (Q_1Q_2)^T A(Q_1Q_2)$ also is real.

The first column of Q_1Q_2 is proportional to the first column of $A_0^2 - 2\Re\sigma A_0 + |\sigma^2|I$, which is

$$\begin{bmatrix} a_{11}^2 + a_{12}a_{21} - 2(\Re\sigma)a_{11} + |\sigma|^2 \\ a_{21}(a_{11} + a_{22} - 2(\Re\sigma)) \\ a_{21}a_{32} \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \qquad \square$$

The rest of the columns of Q_1Q_2 are computed implicitly using the implicit Q theorem. The process is still called "bulge chasing," but now the bulge is 2-by-2 instead of 1-by-1.

EXAMPLE 4.11. Here is a 6-by-6 example of bulge chasing.

1. Choose $Q_1^T = [\begin{smallmatrix} \tilde{Q}_1^T & 0 \\ 0 & I \end{smallmatrix}]$, where the first column of $\tilde{Q}_1^T$ is given as above, so

$$Q_1^T A = \begin{bmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ + & x & x & x & x & x \\ o & o & x & x & x & x \\ o & o & o & x & x & x \\ o & o & o & o & x & x \end{bmatrix} \quad \text{and } A_1 \equiv Q_1^T A Q_1 = \begin{bmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ + & x & x & x & x & x \\ + & + & x & x & x & x \\ o & o & o & x & x & x \\ o & o & o & o & x & x \end{bmatrix}.$$

We see that there is a 2-by-2 bulge, indicted by plus signs.

2. Choose a Householder reflection Q_2^T, which affects only rows 2, 3, and 4 of $Q_2^T A_1$, zeroing out entries $(3,1)$ and $(4,1)$ of A_1 (this means that Q_2^T is the identity matrix outside rows and columns 2 through 4):

$$Q_2^T A_1 = \begin{bmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ o & x & x & x & x & x \\ o & + & x & x & x & x \\ o & o & o & x & x & x \\ o & o & o & o & x & x \end{bmatrix} \quad \text{and } A_2 \equiv Q_2^T A_1 Q_2 = \begin{bmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ o & x & x & x & x & x \\ o & + & x & x & x & x \\ o & + & + & x & x & x \\ o & o & o & o & x & x \end{bmatrix},$$

and the 2-by-2 bulge has been "chased" one column.

3. Choose a Householder reflection Q_3^T, which affects only rows 3, 4, and 5 of $Q_3^T A_2$, zeroing out entries $(4,2)$ and $(5,2)$ of A_2 (this means that Q_3^T is the identity outside rows and columns 3 through 5):

$$Q_3^T A_2 = \begin{bmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ o & x & x & x & x & x \\ o & o & x & x & x & x \\ o & o & + & x & x & x \\ o & o & o & o & x & x \end{bmatrix} \quad \text{and } A_3 \equiv Q_3^T A_2 Q_3 = \begin{bmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ o & x & x & x & x & x \\ o & o & x & x & x & x \\ o & o & + & x & x & x \\ o & o & + & + & x & x \end{bmatrix}.$$

4. Choose a Householder reflection Q_4^T, which affects only rows 4, 5, and 6 of $Q_4^T A_3$, zeroing out entries $(5,3)$ and $(6,3)$ of A_3 (this means that Q_4^T is the identity matrix outside rows and columns 4 through 6):

$$A_4 \equiv Q_4^T A_3 Q_4 = \begin{bmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ o & x & x & x & x & x \\ o & o & x & x & x & x \\ o & o & o & x & x & x \\ o & o & o & + & x & x \end{bmatrix}.$$

5. Choose

$$Q_5^T = \left[\begin{array}{cccc|cc} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ \hline & & & & c & s \\ & & & & -s & c \end{array}\right] \quad \text{so } A_5 = Q_5^T A_4 Q_5 = \begin{bmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ o & x & x & x & x & x \\ o & o & x & x & x & x \\ o & o & o & x & x & x \\ o & o & o & o & x & x \end{bmatrix}. \diamond$$

Choosing a Shift for the QR Algorithm

To completely specify one iteration of either single shift or double shift Hessenberg QR iteration, we need to choose the shift σ (and $\overline{\sigma}$). Recall from the end of section 4.4.4 that a reasonable choice of single shift, one that resulted in asymptotic quadratic convergence to a real eigenvalue, was $\sigma = a_{n,n}$, the bottom right entry of A_i. The generalization for double shifting is to use the *Francis shift*, which means that σ and $\overline{\sigma}$ are the eigenvalues of the bottom 2-by-2 corner of A_i: $\left[\begin{smallmatrix} a_{n-1,n-1} & a_{n-1,n} \\ a_{n,n-1} & a_{n,n} \end{smallmatrix} \right]$. This will let us converge to either two real eigenvalues in the bottom 2-by-2 corner or a single 2-by-2 block with complex conjugate eigenvalues. When we are close to convergence, we expect $a_{n-1,n-2}$ (and possibly $a_{n,n-1}$) to be small so that the eigenvalues of this 2-by-2 matrix are good approximations for eigenvalues of A. Indeed, one can show that this choice leads to quadratic convergence asymptotically. This means that once $a_{n-1,n-2}$ (and possibly $a_{n,n-1}$) is small enough, its magnitude will square at each step and quickly approach zero. In practice, this works so well that on average only two QR iterations per eigenvalue are needed for convergence for almost all matrices. This justifies calling QR iteration a "direct" method.

In practice, the QR iteration with the Francis shift can fail to converge (indeed, it leaves

$$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

unchanged). So the practical algorithm in use for decades had an "exceptional shift" every 10 shifts if convergence had not occurred. Still, tiny sets of matrices where that algorithm did not converge were discovered only recently [25, 65]; matrices in a small neighborhood of

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & h & 0 \\ 0 & -h & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

where h is a few thousand times machine epsilon, form such a set. So another "exceptional shift" was recently added to the algorithm to patch this case. But it is still an open problem to find a shift strategy that guarantees fast convergence for all matrices.

4.5. Other Nonsymmetric Eigenvalue Problems

4.5.1. Regular Matrix Pencils and Weierstrass Canonical Form

The standard eigenvalue problem asks for which scalars z the matrix $A - zI$ is singular; these scalars are the eigenvalues. This notion generalizes in several important ways.

DEFINITION 4.6. *$A - \lambda B$, where A and B are m-by-n matrices, is called a* matrix pencil, *or just a* pencil. *Here λ is an indeterminate, not a particular, numerical value.*

DEFINITION 4.7. *If A and B are square and* $\det(A - \lambda B)$ *is not identically zero, the pencil $A - \lambda B$ is called* regular. *Otherwise it is called* singular. *When $A - \lambda B$ is regular, $p(\lambda) \equiv \det(A - \lambda B)$ is called the* characteristic polynomial *of $A - \lambda B$ and the eigenvalues of $A - \lambda B$ are defined to be*

(1) *the roots of $p(\lambda) = 0$,*
(2) ∞ *(with multiplicity $n - \deg(p)$) if $\deg(p) < n$.*

EXAMPLE 4.12. Let

$$A - \lambda B = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 0 \end{bmatrix} - \lambda \begin{bmatrix} 2 & & \\ & 0 & \\ & & 1 \end{bmatrix}.$$

Then $p(\lambda) = \det(A - \lambda B) = (1 - 2\lambda) \cdot (1 - 0\lambda) \cdot (0 - \lambda) = (2\lambda - 1)\lambda$, so the eigenvalues are $\lambda = \frac{1}{2}, 0$ and ∞. ◇

Matrix pencils arise naturally in many mathematical models of physical systems; we give examples below. The next proposition relates the eigenvalues of a regular pencil $A - \lambda B$ to the eigenvalues of a single matrix.

PROPOSITION 4.6. *Let $A - \lambda B$ be regular. If B is nonsingular, all eigenvalues of $A - \lambda B$ are finite and the same as the eigenvalues of AB^{-1} or $B^{-1}A$. If B is singular, $A - \lambda B$ has eigenvalue ∞ with multiplicity $n -$* rank(B). *If A is nonsingular, the eigenvalues of $A - \lambda B$ are the same as the reciprocals of the eigenvalues of $A^{-1}B$ or BA^{-1}, where a zero eigenvalue of $A^{-1}B$ corresponds to an infinite eigenvalue of $A - \lambda B$.*

Proof. If B is nonsingular and λ' is an eigenvalue, then $0 = \det(A - \lambda' B) = \det(AB^{-1} - \lambda' I) = \det(B^{-1}A - \lambda' I)$, so λ' is also an eigenvalue of AB^{-1} and $B^{-1}A$. If B is singular, then take $p(\lambda) = \det(A - \lambda B)$, write the SVD of B as $B = U\Sigma V^T$, and substitute to get

$$p(\lambda) = \det(A - \lambda U \Sigma V^T) = \det(U(U^T AV - \lambda\Sigma)V^T) = \pm\det(U^T AV - \lambda\Sigma).$$

Since $\text{rank}(B) = \text{rank}(\Sigma)$, only rank$(B)$ λ's appear in $U^T AV - \lambda\Sigma$, so the degree of the polynomial $\det(U^T AV - \lambda\Sigma)$ is rank(B).

If A is nonsingular, $\det(A - \lambda B) = 0$ if and only if $\det(I - \lambda A^{-1}B) = 0$ or $\det(I - \lambda BA^{-1}) = 0$. This equality can hold only if $\lambda \neq 0$ and $1/\lambda$ is an eigenvalue of $A^{-1}B$ or BA^{-1}. □

DEFINITION 4.8. *Let λ' be a finite eigenvalue of the regular pencil $A - \lambda B$. Then $x \neq 0$ is a* right eigenvector *if $(A-\lambda' B)x = 0$, or equivalently $Ax = \lambda' Bx$. If $\lambda' = \infty$ is an eigenvalue and $Bx = 0$, then x is a right eigenvector. A left eigenvector of $A - \lambda B$ is a right eigenvector of $(A - \lambda B)^*$.*

EXAMPLE 4.13. Consider the pencil $A - \lambda B$ in Example 4.12. Since A and B are diagonal, the right and left eigenvectors are just the columns of the identity matrix. ◇

EXAMPLE 4.14. Consider the damped mass-spring system from Example 4.1. There are two matrix pencils that arise naturally from this problem. First, we can write the eigenvalue problem

$$Ax = \begin{bmatrix} -M^{-1}B & -M^{-1}K \\ I & 0 \end{bmatrix} x = \lambda x$$

as

$$\begin{bmatrix} -B & -K \\ I & 0 \end{bmatrix} x = \lambda \begin{bmatrix} M & 0 \\ 0 & I \end{bmatrix} x.$$

This may be a superior formulation if M is very ill-conditioned, so that $M^{-1}B$ and $M^{-1}K$ are hard to compute accurately.

Second, it is common to consider the case $B = 0$ (no damping), so the original differential equation is $M\ddot{x}(t) + Kx(t) = 0$. Seeking solutions of the form $x_i(t) = e^{\lambda_i t}x_i(0)$, we get $\lambda_i^2 e^{\lambda_i t} M x_i(0) + e^{\lambda_i t} K x_i(0) = 0$, or $\lambda_i^2 M x_i(0) + K x_i(0) = 0$. In other words, $-\lambda_i^2$ is an eigenvalue and $x_i(0)$ is a right eigenvector of the pencil $K - \lambda M$. Since we are assuming that M is nonsingular, these are also the eigenvalue and right eigenvector of $M^{-1}K$. ◇

Infinite eigenvalues also arise naturally in practice. For example, later in this section we will show how infinite eigenvalues correspond to *impulse response* in a system described by ordinary differential equations with linear constraints, or *differential-algebraic equations* [41]. See also Question 4.16 for an application of matrix pencils to computational geometry and computer graphics.

Recall that all of our theory and algorithms for the eigenvalue problem of a single matrix A depended on finding a similarity transformation $S^{-1}AS$ of A that is in "simpler" form than A. The next definition shows how to generalize the notion of similarity to matrix pencils. Then we show how the Jordan form and Schur form generalize to pencils.

DEFINITION 4.9. *Let P_L and P_R be nonsingular matrices. Then pencils $A-\lambda B$ and $P_L A P_R - \lambda P_L B P_R$ are called* equivalent.

PROPOSITION 4.7. *The equivalent regular pencils $A-\lambda B$ and $P_L A P_R - \lambda P_L B P_R$ have the same eigenvalues. The vector x is a right eigenvector of $A - \lambda B$ if* *and only if $P_R^{-1}x$ is a right eigenvector of $P_LAP_R - \lambda P_LBP_R$. The vector y is a left eigenvector of $A - \lambda B$ if and only if $(P_L^*)^{-1}y$ is a left eigenvector of $P_LAP_R - \lambda P_LBP_R$.*

Proof.

$\det(A - \lambda B) = 0$ if and only if $\det(P_L(A - \lambda B)P_R) = 0$.

$(A - \lambda B)x = 0$ if and only if $P_L(A - \lambda B)P_RP_R^{-1}x = 0$.

$(A - \lambda B)^*y = 0$ if and only if $P_R^*(A - \lambda B)^*P_L^*(P_L^*)^{-1}y = 0$. □

The following theorem generalizes the Jordan canonical form to regular matrix pencils.

THEOREM 4.10. Weierstrass canonical form. *Let $A - \lambda B$ be regular. Then there are nonsingular P_L and P_R such that*

$$P_L(A - \lambda B)P_R = \text{diag}(J_{n_1}(\lambda_1) - \lambda I_{n_1}, \ldots, J_{n_k}(\lambda_{n_k}) - \lambda I_{n_k}, N_{m_1}, \ldots, N_{m_r}),$$

where $J_{n_i}(\lambda_i)$ is an n_i-by-n_i Jordan block with eigenvalue λ_i,

$$J_{n_i}(\lambda_i) = \begin{bmatrix} \lambda_i & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{bmatrix},$$

and N_{m_i} is a "Jordan block for $\lambda = \infty$ with multiplicity m_i,"

$$N_{m_i} = \begin{bmatrix} 1 & \lambda & & \\ & 1 & \ddots & \\ & & \ddots & \lambda \\ & & & 1 \end{bmatrix} = I_{m_i} - \lambda J_{m_i}(0).$$

For a proof, see [110].

Application of Jordan and Weierstrass Forms to Differential Equations

Consider the linear differential equation $\dot{x}(t) = Ax(t) + f(t)$, $x(0) = x_0$. An explicit solution is given by $x(t) = e^{At}x_0 + \int_0^t e^{A(t-\tau)}f(\tau)d\tau$. If we know the Jordan form $A = SJS^{-1}$, we may change variables in the differential equation to $y(t) = S^{-1}x(t)$ to get $\dot{y}(t) = Jy(t) + S^{-1}f(t)$, with solution $y(t) = e^{Jt}y_0 + \int_0^t e^{J(t-\tau)}S^{-1}f(\tau)d\tau$. There is an explicit formula to compute e^{Jt} or any other function $f(J)$ of a matrix in Jordan form J. (We should not use this formula numerically! For the basis of a better algorithm, see Question 4.4.) Suppose that f is given by its Taylor series $f(z) = \sum_{i=0}^{\infty} \frac{f^{(i)}(0)z^i}{i!}$ and J is a single Jordan block $J = \lambda I + N$, where N has ones on the first superdiagonal and zeros elsewhere. Then

$$\begin{aligned} f(J) &= \sum_{i=0}^{\infty} \frac{f^{(i)}(0)(\lambda I + N)^i}{i!} \\ &= \sum_{i=0}^{\infty} \frac{f^{(i)}(0)}{i!} \sum_{j=0}^{i} \left(\begin{array}{c} i \\ j \end{array} \right) \lambda^{i-j} N^j \quad \text{by the binomial theorem} \\ &= \sum_{j=0}^{\infty} \sum_{i=j}^{\infty} \frac{f^{(i)}(0)}{i!} \left(\begin{array}{c} i \\ j \end{array} \right) \lambda^{i-j} N^j \quad \text{reversing the order of summation} \\ &= \sum_{j=0}^{n-1} N^j \sum_{i=j}^{\infty} \frac{f^{(i)}(0)}{i!} \left(\begin{array}{c} i \\ j \end{array} \right) \lambda^{i-j}, \end{aligned}$$

where in the last equality we used the fact that $N^j = 0$ for $j > n-1$. Note that N^j has ones on the jth superdiagonal and zeros elsewhere. Finally, note that $\sum_{i=j}^{\infty} \frac{f^{(i)}(0)}{i!} \left(\begin{array}{c} i \\ j \end{array} \right) \lambda^{i-j}$ is the Taylor expansion for $f^{(j)}(\lambda)/j!$. Thus

$$\begin{aligned} f(J) &= f\left(\left[\begin{array}{cccc} \lambda & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \lambda \end{array} \right]^{n \times n} \right) = \sum_{j=0}^{n-1} \frac{N^j f^{(j)}(\lambda)}{j!} \\ &= \left[\begin{array}{ccccc} f(\lambda) & f'(\lambda) & \frac{f''(\lambda)}{2!} & \cdots & \frac{f^{(n-1)}(\lambda)}{(n-1)!} \\ & \ddots & \ddots & \ddots & \vdots \\ & & \ddots & \ddots & \frac{f''(\lambda)}{2!} \\ & & & \ddots & f'(\lambda) \\ & & & & f(\lambda) \end{array} \right] \end{aligned} \tag{4.6}$$

so that $f(J)$ is upper triangular with $f^{(j)}(\lambda)/j!$ on the jth superdiagonal.

To solve the more general problem $B\dot{x} = Ax + f(t)$, $A - \lambda B$ regular, we use the Weierstrass form: let $P_L(A - \lambda B)P_R$ be in Weierstrass form, and rewrite the equation as $P_L B P_R P_R^{-1} \dot{x} = P_L A P_R P_R^{-1} x + P_L f(t)$. Let $P_R^{-1} x = y$ and $P_L f(t) = g(t)$. Now the problem has been decomposed into subproblems:

$$\left[\begin{array}{cccccc} I_{n_1} & & & & & \\ & \ddots & & & & \\ & & I_{n_k} & & & \\ & & & J_{m_1}(0) & & \\ & & & & \ddots & \\ & & & & & J_{m_r}(0) \end{array} \right] \dot{y}$$

$$= \begin{bmatrix} J_{n_1}(\lambda_1) & & & & & \\ & \ddots & & & & \\ & & J_{n_k}(\lambda_k) & & & \\ & & & I_{m_1} & & \\ & & & & \ddots & \\ & & & & & I_{m_r} \end{bmatrix} y + g.$$

Each subproblem $\dot{\tilde{y}} = J_{n_i}(\lambda_i)\tilde{y} + \tilde{g}(t) \equiv J\tilde{y} + \tilde{y}(t)$ is a standard linear ODE as above with solution

$$\tilde{y}(t) = \tilde{y}(0)e^{Jt} + \int_0^t e^{J(t-\tau)}\tilde{g}(\tau)d\tau$$

The solution of $J_m(0)\dot{\tilde{y}} = \tilde{y} + \tilde{g}(t)$ is gotten by back substitution starting from the last equation: write $J_m(0)\dot{\tilde{y}} = \tilde{y} + \tilde{g}(t)$ as

$$\begin{bmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & 0 \end{bmatrix} \begin{bmatrix} \dot{\tilde{y}}_1 \\ \vdots \\ \vdots \\ \dot{\tilde{y}}_m \end{bmatrix} = \begin{bmatrix} \tilde{y}_1 \\ \vdots \\ \vdots \\ \tilde{y}_m \end{bmatrix} + \begin{bmatrix} \tilde{g}_1 \\ \vdots \\ \vdots \\ \tilde{g}_m \end{bmatrix}.$$

The mth (last) equation says $0 = \tilde{y}_m + \tilde{g}_m$ or $\tilde{y}_m = -\tilde{g}_m$. The ith equation says $\dot{\tilde{y}}_{i+1} = \tilde{y}_i + \tilde{g}_i$, so $\tilde{y}_i = \dot{\tilde{y}}_{i+1} - \tilde{g}_i$ and thus

$$\tilde{y}_i = \sum_{k=i}^{m} -\frac{d^{k-i}}{dt^{k-i}}\tilde{g}_k(t).$$

Therefore the solution depends on derivatives of $\tilde{g}$, *not* an integral of $\tilde{g}$ as in the usual ODE. Thus a continuous $\tilde{g}$ which is not differentiable can cause a discontinuity in the solution; this is sometimes called an *impulse response* and occurs only if there are infinite eigenvalues. Furthermore, to have a continuous solution $\tilde{y}$ must satisfy certain consistency conditions at $t = 0$:

$$y_i(0) = \sum_{k=m}^{i} -\frac{d^{k-i}}{dt^{k-i}}g_k(0).$$

Numerical methods, based on time-stepping, for solving such *differential algebraic equations*, or ODEs with algebraic constraints, are described in [41].

Generalized Schur Form for Regular Pencils

Just as we cannot compute the Jordan form stably, we cannot compute its generalization by Weierstrass stably. Instead, we compute the generalized Schur form.

THEOREM 4.11. Generalized Schur form. *Let $A - \lambda B$ be regular. Then there exist unitary Q_L and Q_R so that $Q_L A Q_R = T_A$ and $Q_L B Q_R = T_B$ are both upper triangular. The eigenvalues of $A - \lambda B$ are then $T_{A_{ii}}/T_{B_{ii}}$, the ratios of the diagonal entries of T_A and T_B.*

Proof. The proof is very much like that for the usual Schur form. Let λ' be an eigenvalue and x be a unit right eigenvector: $\|x\|_2 = 1$. Since $Ax - \lambda' Bx = 0$, both Ax and Bx are multiples of the same unit vector y (even if one of Ax or Bx is zero). Now let $X = [x, \tilde{X}]$ and $Y = [y, \tilde{Y}]$ be unitary matrices with first columns x and y, respectively. Then $Y^*AX = \begin{bmatrix} \tilde{a}_{11} & \tilde{a}_{12} \\ 0 & \tilde{A}_{22} \end{bmatrix}$ and $Y^*BX = \begin{bmatrix} \tilde{b}_{11} & \tilde{b}_{12} \\ 0 & \tilde{B}_{22} \end{bmatrix}$ by construction. Apply this process inductively to $\tilde{A}_{22} - \lambda \tilde{B}_{22}$. □

If A and B are real, there is a generalized real Schur form too: real orthogonal Q_L and Q_R, where $Q_L A Q_R$ is quasi–upper triangular and $Q_L B Q_R$ is upper triangular.

The QR algorithm and all its refinements generalize to compute the generalized (real) Schur form; it is called the QZ algorithm and available in LAPACK subroutine `sgges`. In Matlab one uses the command `eig(A,B)`.

Definite Pencils

A simpler special case that often arises in practice is the pencil $A - \lambda B$, where $A = A^T$, $B = B^T$, and B is positive definite. Such pencils are called *definite pencils.*

THEOREM 4.12. *Let $A = A^T$, and let $B = B^T$ be positive definite. Then there is a real nonsingular matrix X so that $X^T A X = \mathrm{diag}(\alpha_1, \ldots, \alpha_n)$ and $X^T B X = \mathrm{diag}(\beta_1, \ldots, \beta_n)$. In particular, all the eigenvalues α_i/β_i are real and finite.*

Proof. The proof that we give is actually the algorithm used to solve the problem:

(1) Let $LL^T = B$ be the Cholesky decomposition.
(2) Let $H = L^{-1} A L^{-T}$; note that H is symmetric.
(3) Let $H = Q \Lambda Q^T$, with Q orthogonal, Λ real and diagonal.

Then $X = L^{-T} Q$ satisfies $X^T A X = Q^T L^{-1} A L^{-T} Q = \Lambda$ and $X^T B X = Q^T L^{-1} B L^{-T} Q = I$. □

Note that the theorem is also true if $\alpha A + \beta B$ is positive definite for some scalars α and β.

Software for this problem is available as LAPACK routine `ssygv`.

EXAMPLE 4.15. Consider the pencil $K - \lambda M$ from Example 4.14. This is a definite pencil since the stiffness matrix K is symmetric and the mass matrix

M is symmetric and positive definite. In fact, K is tridiagonal and M is diagonal in this very simple example, so M's Cholesky factor L is also diagonal, and $H = L^{-1}KL^{-T}$ is also symmetric and tridiagonal. In Chapter 5 we will consider a variety of algorithms for the symmetric tridiagonal eigenproblem. ◇

4.5.2. Singular Matrix Pencils and the Kronecker Canonical Form

Now we consider singular pencils $A - \lambda B$. Recall that $A - \lambda B$ is singular if either A and B are nonsquare or they are square and $\det(A - \lambda B) = 0$ for all values of λ. The next example shows that care is needed in extending the definition of eigenvalues to this case.

EXAMPLE 4.16. Let $A = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right]$ and $B = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right]$. Then by making arbitrarily small changes to get $A' = \left[\begin{smallmatrix} 1 & \epsilon_1 \\ \epsilon_2 & 0 \end{smallmatrix}\right]$ and $B' = \left[\begin{smallmatrix} 1 & \epsilon_3 \\ \epsilon_4 & 0 \end{smallmatrix}\right]$, the eigenvalues become ϵ_1/ϵ_3 and ϵ_2/ϵ_4, which can be arbitrary complex numbers. So the eigenvalues are *infinitely* sensitive. ◇

Despite this extreme sensitivity, singular pencils are used in modeling certain physical systems, as we describe below.

We continue by showing how to generalize the Jordan and Weierstrass forms to singular pencils. In addition to Jordan and "infinite Jordan" blocks, we get two new "singular blocks" in the canonical form.

THEOREM 4.13. Kronecker canonical form. *Let A and B be arbitrary rectangular m-by-n matrices. Then there are square nonsingular matrices P_L and P_R so that $P_L A P_R - \lambda P_L B P_R$ is block diagonal with four kinds of blocks:*

$$J_m(\lambda') - \lambda I = \begin{bmatrix} \lambda' - \lambda & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \lambda' - \lambda \end{bmatrix}, \quad \textit{m-by-m Jordan block;}$$

$$N_m = \begin{bmatrix} 1 & \lambda & & \\ & \ddots & \ddots & \\ & & \ddots & \lambda \\ & & & 1 \end{bmatrix}, \quad \textit{m-by-m Jordan block for } \lambda = \infty;$$

$$L_m = \begin{bmatrix} 1 & \lambda & & \\ & \ddots & \ddots & \\ & & 1 & \lambda \end{bmatrix}, \quad \textit{m-by-}(m+1) \textit{ right singular block};$$

$$L_m^T = \begin{bmatrix} 1 & & \\ \lambda & \ddots & \\ & \ddots & 1 \\ & & \lambda \end{bmatrix}, \quad (m+1)\textit{-by-}m \textit{ left singular block}.$$

We call L_m a right singular block since it has a right null vector $[\lambda^m, -\lambda^{m-1}, \ldots, \pm 1]$ for all λ. L_m^T has an analogous left null vector.

For a proof, see [110].

Just as Schur form generalized to regular matrix pencils in the last section, it can be generalized to arbitrary singular pencils as well. For the canonical form, perturbation theory and software, see [27, 79, 246].

Singular pencils are used to model systems arising in systems and control. We give two examples.

Application of Kronecker Form to Differential Equations

Suppose that we want to solve $B\dot{x} = Ax + f(t)$, where $A - \lambda B$ is a singular pencil. Write $P_L B P_R P_R^{-1} \dot{x} = P_L A P_R P_R^{-1} x + P_L f(t)$ to decompose the problem into independent blocks. There are four kinds, one for each kind in the Kronecker form. We have already dealt with $J_m(\lambda') - \lambda I$ and N_m blocks when we considered regular pencils and Weierstrass form, so we have to consider only L_m and L_m^T blocks. From the L_m blocks we get

$$\begin{bmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{y}_1 \\ \vdots \\ \dot{y}_{m+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_{m+1} \end{bmatrix} + \begin{bmatrix} g_1 \\ \vdots \\ g_m \end{bmatrix}$$

or

$$\begin{array}{rclcrcl} \dot{y}_2 & = & y_1 + g_1 & \text{or} & y_2(t) & = & y_2(0) + \int_0^t (y_1(\tau) + g_1(\tau))d\tau, \\ \dot{y}_3 & = & y_2 + g_2 & \text{or} & y_3(t) & = & y_3(0) + \int_0^t (y_2(\tau) + g_2(\tau))d\tau, \\ & \vdots & & & & & \\ \dot{y}_{m+1} & = & y_m + g_m & \text{or} & y_{m+1}(t) & = & y_{m+1}(0) + \int_0^t (y_m(\tau) + g_m(\tau))d\tau. \end{array}$$

This means that we can choose y_1 as an arbitrary integrable function and use the above recurrence relations to get a solution. This is because we have one more unknown than equation, so the the ODE is *underdetermined.* From the L_m^T blocks we get

$$\begin{bmatrix} 0 & & \\ 1 & \ddots & \\ & \ddots & 0 \\ & & 1 \end{bmatrix} \begin{bmatrix} \dot{y}_1 \\ \vdots \\ \dot{y}_m \end{bmatrix} = \begin{bmatrix} 1 & & \\ 0 & \ddots & \\ & \ddots & 1 \\ & & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} + \begin{bmatrix} g_1 \\ \vdots \\ g_{m+1} \end{bmatrix}$$

or

$$\begin{aligned} 0 &= y_1 + g_1, \\ \dot{y}_1 &= y_2 + g_2, \\ &\vdots \\ \dot{y}_{m-1} &= y_m + g_m, \\ \dot{y}_m &= g_{m+1}. \end{aligned}$$

Starting with the first equation, we solve to get

$$\begin{aligned} y_1 &= -g_1, \\ y_2 &= -g_2 - \dot{g}_1, \\ &\vdots \\ y_m &= -g_m - \dot{g}_{m-1} - \cdots - \frac{d^{m-1}}{dt^{m-1}} g_1 \end{aligned}$$

and the *consistency condition* $g_{m+1} = -\dot{g}_m - \cdots - \frac{d^m}{dt^m} g_1$. So unless the g_i satisfy this equation, there is no solution. Here we have one more equation than unknown, and the subproblem is *overdetermined.*

Application of Kronecker Form to Systems and Control Theory

The *controllable subspace* of $\dot{x}(t) = Ax(t) + Bu(t)$ is the space in which the *system state* $x(t)$ can be "controlled" by choosing the *control input* $u(t)$ starting at $x(0) = 0$. This equation is used to model (feedback) control systems, where the $u(t)$ is chosen by the control system engineer to make $x(t)$ have certain desirable properties, such as boundedness. From

$$\begin{aligned} x(t) &= \int_0^t e^{A(t-\tau)} Bu(\tau) d\tau = \int_0^t \sum_{i=0}^{\infty} \frac{(t-\tau)^i}{i!} A^i Bu(\tau) d\tau \\ &= \sum_{i=0}^{\infty} A^i B \int_0^t \frac{(t-\tau)^i}{i!} u(\tau) d\tau \end{aligned}$$

one can prove the controllable space is span$\{[B, AB, A^2B, \ldots, A^{n-1}B]\}$; any components of $x(t)$ outside this space cannot be controlled by varying $u(t)$. To compute this space in practice, in order to determine whether the physical system being modeled can in fact be controlled by input $u(t)$, one applies a QR-like algorithm to the singular pencil $[B, A - \lambda I]$. For details, see [78, 246, 247].

4.5.3. Nonlinear Eigenvalue Problems

Finally, we consider the *nonlinear eigenvalue problem* or *matrix polynomial*

$$\sum_{i=0}^{d} \lambda^i A_i = \lambda^d A_d + \lambda^{d-1} A_{d-1} + \cdots + \lambda A_1 + A_0. \tag{4.7}$$

Suppose for simplicity that the A_i are n-by-n matrices and A_d is nonsingular.

DEFINITION 4.10. *The* characteristic polynomial *of the matrix polynomial* (4.7) *is* $p(\lambda) = \det(\sum_{i=0}^{d} \lambda^i A_i)$. *The roots of* $p(\lambda) = 0$ *are defined to be the* eigenvalues. *One can confirm that* $p(\lambda)$ *has degree* $d \cdot n$, *so there are* $d \cdot n$ *eigenvalues. Suppose that* γ *is an eigenvalue. A nonzero vector* x *satisfying* $\sum_{i=0}^{d} \gamma^i A_i x = 0$ *is a* right eigenvector *for* γ. *A left eigenvector* y *is defined analogously by* $\sum_{i=0}^{d} \gamma^i y^* A_i = 0$.

EXAMPLE 4.17. Consider Example 4.1 once again. The ODE arising there in equation (4.3) is $M\ddot{x}(t) + B\dot{x}(t) + Kx(t) = 0$. If we seek solutions of the form $x(t) = e^{\lambda_i t} x_i(0)$, we get $e^{\lambda_i t}(\lambda_i^2 M x_i(0) + \lambda_i B x_i(0) + K x_i(0)) = 0$, or $\lambda_i^2 M x_i(0) + \lambda_i B x_i(0) + K x_i(0) = 0$. Thus λ_i is an eigenvalue and $x_i(0)$ is an eigenvector of the matrix polynomial $\lambda^2 M + \lambda B + K$. ◇

Since we are assuming that A_d is nonsingular, we can multiply through by A_d^{-1} to get the equivalent problem $\lambda^d I + A_d^{-1} A_{d-1} \lambda^{d-1} + \cdots + A_d^{-1} A_0$. Therefore, to keep the notation simple, we will assume $A_d = I$ (see section 4.6 for the general case). In the very simplest case where each A_i is 1-by-1, i.e., a scalar, the original matrix polynomial is equal to the characteristic polynomial.

We can turn the problem of finding the eigenvalues of a matrix polynomial into a standard eigenvalue problem by using a trick analogous to the one used to change a high-order ODE into a first-order ODE. Consider first the simplest case $n = 1$, where each A_i is a scalar. Suppose that γ is a root. Then the vector $x' = [\gamma^{d-1}, \gamma^{d-2}, \ldots, \gamma, 1]^T$ satisfies

$$\begin{aligned} Cx' &\equiv \begin{bmatrix} -A_{d-1} & -A_{d-2} & \cdots & \cdots & \cdots & -A_0 \\ 1 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & 1 & 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & 1 & 0 \end{bmatrix} x' = \begin{bmatrix} -\sum_{i=0}^{d-1} \gamma^i A_i \\ \gamma^{d-1} \\ \vdots \\ \gamma^2 \\ \gamma \end{bmatrix} \\ &= \begin{bmatrix} \gamma^d \\ \gamma^{d-1} \\ \vdots \\ \gamma^2 \\ \gamma \end{bmatrix} = \gamma x'. \end{aligned}$$

Thus x' is an eigenvector and γ is an eigenvalue of the matrix C, which is called the *companion matrix* of the polynomial (4.7).

(The Matlab routine `roots` for finding roots of a polynomial applies the Hessenberg QR iteration of section 4.4.8 to the companion matrix C, since this is currently one of the most reliable, if expensive, methods known [100, 117, 241]. Cheaper alternatives are under development.)

The same idea works when the A_i are matrices. C becomes an $(n \cdot d)$-by-$(n \cdot d)$ *block companion matrix*, where the 1's and 0's below the top row become n-by-n identity and zero matrices, respectively. Also, x' becomes

$$x' = \begin{bmatrix} \gamma^{d-1}x \\ \gamma^{d-2}x \\ \vdots \\ \gamma x \\ x \end{bmatrix},$$

where x is a right eigenvector of the matrix polynomial. It again turns out that $Cx' = \gamma x'$.

EXAMPLE 4.18. Returning once again to $\lambda^2 M + \lambda B + K$, we first convert it to $\lambda^2 + \lambda M^{-1}B + M^{-1}K$ and then to the companion matrix

$$C = \begin{bmatrix} -M^{-1}B & -M^{-1}K \\ I & 0 \end{bmatrix}.$$

This is the same as the matrix A in equation 4.4 of Example 4.1. ◇

Finally, Question 4.16 shows how to use matrix polynomials to solve a problem in *computational geometry.*

4.6. Summary

The following list summarizes all the canonical forms, algorithms, their costs, and applications to ODEs described in this chapter. It also includes pointers to algorithms exploiting symmetry, although these are discussed in more detail in the next chapter. Algorithms for sparse matrices are discussed in Chapter 7.

- $A - \lambda I$
  - Jordan form: For some nonsingular S,

$$A - \lambda I = S \cdot \text{diag}\left(\ldots, \begin{bmatrix} \lambda_i - \lambda & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i - \lambda \end{bmatrix}^{n_i \times n_i}, \ldots \right) \cdot S^{-1}.$$

- Schur form: For some unitary Q, $A - \lambda I = Q(T - \lambda I)Q^*$, where T is triangular.
- Real Schur form of real A: For some real orthogonal Q, $A - \lambda I = Q(T - \lambda I)Q^T$, where T is real quasi-triangular.
- Application to ODEs: Provides solution of $\dot{x}(t) = Ax(t) + f(t)$.
- Algorithm: Do Hessenberg reduction (Algorithm 4.6), followed by QR iteration to get Schur form (Algorithm 4.5, implemented as described in section 4.4.8). Eigenvectors can be computed from the Schur form (as described in section 4.2.1).
- Cost: This costs $10n^3$ flops if eigenvalues only are desired, $25n^3$ if T and Q are also desired, and a little over $27n^3$ if eigenvectors are also desired. Since not all parts of the algorithm can take advantage of the Level 3 BLAS, the cost is actually higher than a comparison with the $2n^3$ cost of matrix multiply would indicate: instead of taking $(10n^3)/(2n^3) = 5$ times longer to compute eigenvalues than to multiply matrices, it takes 23 times longer for $n = 100$ and 19 times longer for $n = 1000$ on an IBM RS6000/590 [10, page 62]. Instead of taking $(27n^3)/(2n^3) = 13.5$ times longer to compute eigenvalues and eigenvectors, it takes 41 times longer for $n = 100$ and 60 times longer for $n = 1000$ on the same machine. Thus computing eigenvalues of nonsymmetric matrices is expensive. (The symmetric case is *much* cheaper; see Chapter 5.)
- LAPACK: `sgees` for Schur form or `sgeev` for eigenvalues and eigenvectors; `sgeesx` or `sgeevx` for error bounds too.
- Matlab: `schur` for Schur form or `eig` for eigenvalues and eigenvectors.
- Exploiting symmetry: When $A = A^*$, better algorithms are discussed in Chapter 5, especially section 5.3.

- Regular $A - \lambda B$ ($\det(A - \lambda B) \not\equiv 0$)
  - Weierstrass form: For some nonsingular P_L and P_R,

$$A - \lambda B = P_L \cdot \mathrm{diag}\left(\text{Jordan}, \begin{bmatrix} 1 & \lambda & & \\ & \ddots & \ddots & \\ & & \ddots & \lambda \\ & & & 1 \end{bmatrix}^{n_i \times n_i} \right) P_R^{-1}.$$

  - Generalized Schur form: For some unitary Q_L and Q_R, $A - \lambda B = Q_L(T_A - \lambda T_B)Q_R^*$, where T_A and T_B are triangular.

- Generalized real Schur form of real A and B: For some real orthogonal Q_L and Q_R, $A - \lambda B = Q_L(T_A - \lambda T_B)Q_R^T$, where T_A is real quasi-triangular and T_B is real triangular.
- Application to ODEs: Provides solution of $B\dot{x}(t) = Ax(t) + f(t)$, where the solution is uniquely determined but may depend nonsmoothly on the data (*impulse response*).
- Algorithm: Hessenberg/triangular reduction followed by QZ iteration (QR applied implicitly to AB^{-1}).
- Cost: Computing T_A and T_B costs $30n^3$. Computing Q_L and Q_R in addition costs $66n^3$. Computing eigenvectors as well costs a little less than $69n^3$ in total. As before, Level 3 BLAS cannot be used in all parts of the algorithm.
- LAPACK: `sgges` for Schur form or `sggev` for eigenvalues; `sggesx` or `sggevx` for error bounds too.
- Matlab: `eig` for eigenvalues and eigenvectors.
- Exploiting symmetry: When $A = A^*$, $B = B^*$, and B is positive definite, one can convert the problem to finding the eigenvalues of a single symmetric matrix using Theorem 4.12. This is done in LAPACK routines `ssygv`, `sspgv` (for symmetric matrices in "packed storage"), and `ssbgv` (for symmetric band matrices).

- Singular $A - \lambda B$
  - Kronecker form: For some nonsingular P_L and P_R,

$$A - \lambda B = P_L \cdot \operatorname{diag}\left(\text{Weierstrass}, \begin{bmatrix} 1 & \lambda & & \\ & \ddots & \ddots & \\ & & 1 & \lambda \end{bmatrix}^{n_i \times n_i}, \begin{bmatrix} 1 & & \\ \lambda & \ddots & \\ & \ddots & 1 \\ & & \lambda \end{bmatrix}^{m_i \times m_i} \right) P_R^{-1}.$$

  - Generalized upper triangular form: For some unitary Q_L and Q_R, $A - \lambda B = Q_L(T_A - \lambda T_B)Q_R^*$, where T_A and T_B are in generalized upper triangular form, with diagonal blocks corresponding to different parts of the Kronecker form. See [79, 246] for details of the form and algorithms.
  - Cost: The most general and reliable version of the algorithm can cost as much as $O(n^4)$, depending on the details of the Kronecker Structure; this is much more than for regular $A - \lambda B$. There is also a slightly less reliable $O(n^3)$ algorithm [27].

- Application to ODEs: Provides solution of $B\dot{x}(t) = Ax(t) + f(t)$, where the solution may be overdetermined or underdetermined.
- Software: NETLIB/linalg/guptri.

- Matrix polynomials $\sum_{i=0}^{d} \lambda^i A_i$ [118]
  - If $A_d = I$ (or A_d is square and well-conditioned enough to replace each A_i by $A_d^{-1}A_i$), then linearize to get the standard problem

$$\begin{bmatrix} -A_{d-1} & -A_{d-2} & \cdots & \cdots & \cdots & -A_0 \\ I & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & I & 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & I & 0 \end{bmatrix} - \lambda I.$$

  - If A_d is ill-conditioned or singular, linearize to get the pencil

$$\begin{bmatrix} -A_{d-1} & -A_{d-2} & \cdots & \cdots & \cdots & -A_0 \\ I & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & I & 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & I & 0 \end{bmatrix} - \lambda \begin{bmatrix} A_d & & & & \\ & I & & & \\ & & I & & \\ & & & \ddots & \\ & & & & I \end{bmatrix}.$$

4.7. References and Other Topics for Chapter 4

For a general discussion of properties of eigenvalues and eigenvectors, see [139]. For more details about perturbation theory of eigenvalues and eigenvectors, see [161, 237, 52], and chapter 4 of [10]. For a proof of Theorem 4.7, see [69]. For a discussion of Weierstrass and Kronecker canonical forms, see [110, 118]. For their application to systems and control theory, see [246, 247, 78]. For applications to computational geometry, graphics, and mechanical CAD, see [181, 182, 165]. For a discussion of parallel algorithms for the nonsymmetric eigenproblem, see [76].

4.8. Questions for Chapter 4

QUESTION 4.1. *(Easy)* Let A be defined as in equation (4.1). Show that $\det(A) = \prod_{i=1}^{b} \det(A_{ii})$ and then that $\det(A - \lambda I) = \prod_{i=1}^{b} \det(A_{ii} - \lambda I)$. Conclude that the set of eigenvalues of A is the union of the sets of eigenvalues of A_{11} through A_{bb}.

QUESTION 4.2. *(Medium; Z. Bai)* Suppose that A is *normal*; i.e., $AA^* = A^*A$. Show that if A is also triangular, it must be diagonal. Use this to show that an n-by-n matrix is normal if and only if it has n orthonormal eigenvectors. Hint: Show that A is normal if and only if its Schur form is normal.

QUESTION 4.3. *(Easy; Z. Bai)* Let λ and μ be distinct eigenvalues of A, let x be a right eigenvector for λ, and let y be a left eigenvector for μ. Show that x and y are orthogonal.

QUESTION 4.4. *(Medium)* Suppose A has distinct eigenvalues. Let $f(z) = \sum_{i=-\infty}^{+\infty} a_i z^i$ be a function which is defined at the eigenvalues of A. Let $Q^*AQ = T$ be the Schur form of A (so Q is unitary and T upper triangular).

1. Show that $f(A) = Qf(T)Q^*$. Thus to compute $f(A)$ it suffices to be able to compute $f(T)$. In the rest of the problem you will derive a simple recurrence formula for $f(T)$.

2. Show that $(f(T))_{ii} = f(T_{ii})$ so that the diagonal of $f(T)$ can be computed from the diagonal of T.

3. Show that $Tf(T) = f(T)T$.

4. From the last result, show that the ith superdiagonal of $f(T)$ can be computed from the $(i-1)$st and earlier subdiagonals. Thus, starting at the diagonal of $f(T)$, we can compute the first superdiagonal, second superdiagonal, and so on.

QUESTION 4.5. *(Easy)* Let A be a square matrix. Apply either Question 4.4 to the Schur form of A or equation (4.6) to the Jordan form of A to conclude that the eigenvalues of $f(A)$ are $f(\lambda_i)$, where the λ_i are the eigenvalues of A. This result is called the *spectral mapping theorem.*

This question is used in the proof of Theorem 6.5 and section 6.5.6.

QUESTION 4.6. *(Medium)* In this problem we will show how to solve the *Sylvester* or *Lyapunov* equation $AX - XB = C$, where X and C are m-by-n, A is m-by-m, and B is n-by-n. This is a system of mn linear equations for the entries of X.

1. Given the Schur decompositions of A and B, show how $AX - XB = C$ can be transformed into a similar system $A'Y - YB' = C'$, where A' and B' are upper triangular.

2. Show how to solve for the entries of Y one at a time by a process analogous to back substitution. What condition on the eigenvalues of A and B guarantees that the system of equations is nonsingular?

3. Show how to transform Y to get the solution X.

QUESTION 4.7. *(Medium)* Suppose that $T = \begin{bmatrix} A & C \\ 0 & B \end{bmatrix}$ is in Schur form. We want to find a matrix S so that $S^{-1}TS = \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}$. It turns out we can choose S of the form $\begin{bmatrix} I & R \\ 0 & I \end{bmatrix}$. Show how to solve for R.

QUESTION 4.8. *(Medium; Z. Bai)* Let A be m-by-n and B be n-by-m. Show that the matrices

$$\begin{pmatrix} AB & 0 \\ B & 0 \end{pmatrix} \text{ and } \begin{pmatrix} 0 & 0 \\ B & BA \end{pmatrix}$$

are similar. Conclude that the nonzero eigenvalues of AB are the same as those of BA.

QUESTION 4.9. *(Medium; Z. Bai)* Let A be n-by-n with eigenvalues $\lambda_1, \ldots, \lambda_n$. Show that

$$\sum_{i=1}^{n} |\lambda_i|^2 = \min_{\det(S)\neq 0} \|S^{-1}AS\|_F^2.$$

QUESTION 4.10. *(Medium; Z. Bai)* Let A be an n-by-n matrix with eigenvalues $\lambda_1, \ldots, \lambda_n$.

1. Show that A can be written $A = H + S$, where $H = H^*$ is Hermitian and $S = -S^*$ is skew-Hermitian. Give explicit formulas for H and S in terms of A.

2. Show that $\sum_{i=1}^{n} |\Re\lambda_i|^2 \leq \|H\|_F^2$.

3. Show that $\sum_{i=1}^{n} |\Im\lambda_i|^2 \leq \|S\|_F^2$.

4. Show that A is normal ($AA^* = A^*A$) if and only if $\sum_{i=1}^{n} |\lambda_i|^2 = \|A\|_F^2$.

QUESTION 4.11. *(Easy)* Let λ be a simple eigenvalue, and let x and y be right and left eigenvectors. We define the *spectral projection* P corresponding to λ as $P = xy^*/(y^*x)$. Prove that P has the following properties.

1. P is uniquely defined, even though we could use any nonzero scalar multiples of x and y in its definition.

2. $P^2 = P$. (Any matrix satisfying $P^2 = P$ is called a *projection matrix.*)

3. $AP = PA = \lambda P$. (These properties motivate the name *spectral* projection, since P "contains" the left and right invariant subspaces of λ.)

4. $\|P\|_2$ is the condition number of λ.

QUESTION 4.12. *(Easy; Z. Bai)* Let $A = \left[\begin{smallmatrix} a & c \\ 0 & b \end{smallmatrix}\right]$. Show that the condition numbers of the eigenvalues of A are both equal to $(1 + (\frac{c}{a-b})^2)^{1/2}$. Thus, the condition number is large if the difference $a - b$ between the eigenvalues is small compared to c, the offdiagonal part of the matrix.

QUESTION 4.13. *(Medium; Z. Bai)* Let A be a matrix, x be a unit vector ($\|x\|_2 = 1$), μ be a scalar, and $r = Ax - \mu x$. Show that there is a matrix E with $\|E\|_F = \|r\|_2$ such that $A + E$ has eigenvalue μ and eigenvector x.

QUESTION 4.14. *(Medium; Programming)* In this question we will use a Matlab program to plot eigenvalues of a perturbed matrix and their condition numbers. (It is available at HOMEPAGE/Matlab/eigscat.m.) The input is

a = input matrix,
err = size of perturbation,
m = number of perturbed matrices to compute.

The output consists of three plots in which each symbol is the location of an eigenvalue of a perturbed matrix:

"o" marks the location of each unperturbed eigenvalue.
"x" marks the location of each perturbed eigenvalue, where a real perturbation matrix of norm err is added to a.
"." marks the location of each perturbed eigenvalue, where a complex perturbation matrix of norm err is added to a.

A table of the eigenvalues of A and their condition numbers is also printed.

Here are some interesting examples to try (for as large an m as you want to wait; the larger the m the better, and m equal to a few hundred is good).

```
(1) a = randn(5) (if a does not have complex eigenvalues,
                  try again)
        err=1e-5, 1e-4, 1e-3, 1e-2, .1, .2

(2) a = diag(ones(4,1),1);  err=1e-12, 1e-10, 1e-8

(3) a=[[1 1e6 0     0]; ...
      [0 2   1e-3  0]; ...
      [0 0   3    10]; ...
      [0 0  -1     4]]
        err=1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3

(4) [q,r]=qr(randn(4,4));a=q*diag(ones(3,1),1)*q'
        err=1e-16, 1e-14, 1e-12, 1e-10, 1e-8

(5) a = [[1 1e3 1e6];[0 1 1e3];[0 0 1]],
        err=1e-7, 1e-6, 5e-6, 8e-6, 1e-5, 1.5e-5, 2e-5

(6) a = [[1   0   0   0   0    0]; ...
         [0   2   1   0   0    0]; ...
         [0   0   2   0   0    0]; ...
         [0   0   0   3 1e2 1e4]; ...
         [0   0   0   0   3 1e2]; ...
         [0   0   0   0   0    3]]
        err= 1e-10, 1e-8, 1e-6, 1e-4, 1e-3
```

Your assignment is to try these examples and compare the regions occupied by the eigenvalues (the so-called *pseudospectrum*) with the bounds described in section 4.3. What is the difference between real perturbations and complex perturbations? What happens to the regions occupied by the eigenvalues as the perturbation err goes to zero? What is limiting size of the regions as err goes to zero (i.e., how many digits of the computed eigenvalues are correct)?

QUESTION 4.15. *(Medium; Programming)* In this question we use a Matlab program to plot the diagonal entries of a matrix undergoing unshifted QR iteration. The values of each diagonal are plotted after each QR iteration, each diagonal corresponding to one of the plotted curves. (The program is available at HOMEPAGE/Matlab/qrplt.m and also shown below.) The inputs are

a = input matrix,
m = number of QR iterations,

and the output is a plot of the diagonals.

Examples to try this code on are as follows (choose m large enough so that the curves either converge or go into cycles):

```
a = randn(6);
b = randn(6); a = b*diag([1,2,3,4,5,6])*inv(b);
a = [[1 10];[-1 1]]; m = 300
a = diag((1.5*ones(1,5)).\verb+^+(0:4)) +
    .01*(diag(ones(4,1),1)+diag(ones(4,1),-1)); m=30
```

What happens if there are complex eigenvalues?

In what order do the eigenvalues appear in the matrix after many iterations?

Perform the following experiment: Suppose that a is n-by-n and symmetric. In Matlab, let perm=(n:-1:1). This produces a list of the integers from n down to 1. Run the iteration for m iterations. Let a=a(perm,perm); we call this "flipping" a, because it reverses the order of the rows and columns of a. Run the iteration again for m iterations, and again form a=a(perm,perm). How does this value of a compare with the original value of a? You should not let m be too large (try $m = 5$) or else roundoff will obscure the relationship you should see. (See also Corollary 5.4 and Question 5.25.)

Change the code to compute the error in each diagonal from its final value (do this just for matrices with all real eigenvalues). Plot the log of this error versus the iteration number. What do you get asymptotically?

```
hold off
e=diag(a);
for i=1:m,
   [q,r]=qr(a);dd=diag(sign(diag(r)));r=dd*r;q=q*dd;a=r*q; ...
   e=[e,diag(a)];
end
clg
plot(e','w'),grid
```

QUESTION 4.16. *(Hard; Programming)* This problem describes an application of the nonlinear eigenproblem to computer graphics, computational geometry, and mechanical CAD; see also [181, 182, 165].

Let $F = [f_{ij}(x_1, x_2, x_3)]$ be a matrix whose entries are polynomials in the three variables x_i. Then $\det(F) = 0$ will (generally) define a two-dimensional surface S in 3-space. Let $x_1 = g_1(t)$, $x_2 = g_2(t)$, and $x_3 = g_3(t)$ define a (one-dimensional) curve C parameterized by t, where the g_i are also polynomials. We want to find the intersection $S \cap C$. Show how to express this as an eigenvalue problem (which can then be solved numerically). More generally, explain how to find the intersection of a surface $\det(F(x_1, \ldots, x_n)) = 0$ and curve $\{x_i = g_i(t),\ 1 \le i \le n\}$. At most how many discrete solutions can there be, as a function of n, the dimension d of F, and the maximum of the degrees of the polynomials f_{ij} and g_k?

Write a Matlab program to solve this problem, for $n = 3$ variables, by converting it to an eigenvalue problem. It should take as input a compact description of the entries of each $f_{ij}(x_k)$ and $g_i(t)$ and produce a list of the intersection points. For instance, it could take the following inputs:

- Array NumTerms(1:d,1:d), where NumTerms(i,j) is the number of terms in the polynomial $f_{ij}(x_1, x_2, x_3)$.
- Array Sterms(1:4, 1:TotalTerms), where TotalTerms is the sum of all the entries in NumTerms(.,.). Each column of Sterms represents one term in one polynomial: The first NumTerms(1,1) columns of Sterms represent the terms in f_{11}, the second Numterm(2,1) columns of Sterms represent the terms in f_{21}, and so on. The term represented by Sterms(1:4,k) is $\mathrm{Sterm}(4,k) \cdot x_1^{\mathrm{Sterm}(1,k)} \cdot x_2^{\mathrm{Sterm}(2,k)} \cdot x_3^{\mathrm{Sterm}(3,k)}$.
- Array tC(1:3) contains the degrees of polynomials g_1, g_2, and g_3 in that order.
- Array Curve(1: tC(1)+tC(2)+tC(3)+3) contains the coefficients of the polynomials g_1, g_2, and g_3, one polynomial after the other, from the constant term to the highest order coefficient of each.

Your program should also compute error bounds for the computed answers. This will be possible only when the eigenproblem can be reduced to one for which the error bounds in Theorems 4.4 or 4.5 apply. You do not have to provide error bounds when the eigenproblem is a more general one. (For a description of error bounds for more general eigenproblems, see [10, 237].

Write a second Matlab program that plots S and C for the case $n = 3$ and marks the intersection points.

Are there any limitations on the input data for your codes to work? What happens if S and C do not intersect? What happens if S lies in C?

Run your codes on at least the following examples. You should be able to solve the first five by hand to check your code.

1. $g_1 = t$, $g_2 = 1 + t$, $g_3 = 2 + t$, $F = \begin{bmatrix} x_1 + x_2 + x_3 & 0 \\ 0 & 3x_1 + 5x_2 - 7x_3 + 10 \end{bmatrix}$.

2. $g_1 = t^3$, $g_2 = 1 + t^3$, $g_3 = 2 + t^3$, $F = \begin{bmatrix} x_1 + x_2 + x_3 & 0 \\ 0 & 3x_1 + 5x_2 - 7x_3 + 10 \end{bmatrix}$.

3. $g_1 = t^2$, $g_2 = 1 + t^2$, $g_3 = 2 + t^2$, $F = \begin{bmatrix} x_1 + x_2 + x_3 & 0 \\ 0 & 3x_1 + 5x_2 - 7x_3 + 10 \end{bmatrix}$.

4. $g_1 = t^2$, $g_2 = 1 + t^2$, $g_3 = 2 + t^2$, $F = \begin{bmatrix} 1 & 0 \\ 0 & 3x_1 + 5x_2 - 7x_3 + 9 \end{bmatrix}$.

5. $g_1 = t^2$, $g_2 = 1 + t^2$, $g_3 = 2 + t^2$, $F = \begin{bmatrix} x_1 + x_2 + x_3 & 0 \\ 0 & 3x_1 + 5x_2 - 7x_3 + 8 \end{bmatrix}$.

6. $g_1 = t^2$, $g_2 = 1 + t^2$, $g_3 = 2 + t^2$, $F = \begin{bmatrix} x_1 + x_2 + x_3 & x_1 \\ x_3 & 3x_1 + 5x_2 - 7x_3 + 10 \end{bmatrix}$.

7. $g_1 = 7 - 3t + t^5$, $g_2 = 1 + t^2 + t^5$, $g_3 = 2 + t^2 - t^5$,

$$F = \begin{bmatrix} x_1x_2 + x_3^5 & 3 - x_2^2 & 5 + x_1 + x_2 + x_3 + x_1x_2 + x_1x_3 + x_2x_3 \\ x_2 - 7x_3^5 & 1 - x_1^2 + x_1x_2x_3^3 & 3 + x_1 + 3x_3 - 9x_2x_3 \\ 2 & 3x_1 + 5x_2 - 7x_3 + 8 & x_1^3 - x_2^4 + 4x_3^5 \end{bmatrix}.$$

You should turn in

- mathematical formulation of the solution in terms of an eigenproblem.
- the algorithm in at most two pages, including a road map to your code (subroutine names for each high level operation). It should be easy to see how the mathematical formulation leads to the algorithm and how the algorithm matches the code.

— At most how many discrete solutions can there be?

— Do all compute eigenvalues represent actual intersections? Which ones do?

— What limits does your code place on the input for it to work correctly?

— What happens if S and C do not intersect?

— What happens if S contains C?

- mathematical formulation of the error bounds.
- the algorithm for computing the error bounds in at most two pages, including a road map to your code (subroutine names for each high-level operation). It should be easy to see how the mathematical formulation leads to the algorithm and how the algorithm matches the code.
- program listing.

For each of the seven examples, you should turn in

- the original statement of the problem.
- the resulting eigenproblem.
- the numerical solutions.
- plots of S and C; do your numerical solutions match the plots?
- the result of substituting the computed answers in the equations defining S and C: are they satisfied (to within roundoff)?