Jim Lambers
MAT 610
Summer Session 2009-10
Lecture 14 Notes

These notes correspond to Sections 7.3 and 8.2 in the text.

The Eigenvalue Problem: Power Iterations

The Unsymmetric Eigenvalue Problem

We now consider the problem of computing eigenvalues of an $n \times n$ matrix A. For simplicity, we assume that A has eigenvalues $\lambda_1, \ldots, \lambda_n$ such that

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|.$$

We also assume that A is diagonalizable, meaning that it has n linearly independent eigenvectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$ such that $A\mathbf{x}_i = \lambda_i \mathbf{x}_i$ for $i = 1, \ldots, n$.

Suppose that we continually multiply a given vector $\mathbf{x}^{(0)}$ by A, generating a sequence of vectors $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots$ defined by

$$\mathbf{x}^{(k)} = A\mathbf{x}^{(k-1)} = A^k \mathbf{x}^{(0)}, \quad k = 1, 2, \ldots.$$

Because A is diagonalizable, any vector in $\mathbb{R}^n$ is a linear combination of the eigenvectors, and therefore we can write

$$\mathbf{x}^{(0)} = c_1\mathbf{x}_1 + c_2\mathbf{x}_2 + \cdots + c_n\mathbf{x}_n.$$

We then have

$$\begin{aligned}
\mathbf{x}^{(k)} &= A^k \mathbf{x}^{(0)} \\
&= \sum_{i=1}^{n} c_i A^k \mathbf{x}_i \\
&= \sum_{i=1}^{n} c_i \lambda_i^k \mathbf{x}_i \\
&= \lambda_1^k \left[c_1 \mathbf{x}_1 + \sum_{i=2}^{n} c_i \left(\frac{\lambda_i}{\lambda_1} \right)^k \mathbf{x}_i \right].
\end{aligned}$$

Because $|\lambda_1| > |\lambda_i|$ for $i = 2, \ldots, n$, it follows that the coefficients of $\mathbf{x}_i$, for $i = 2, \ldots, n$, converge to zero as $k \to \infty$. Therefore, the direction of $\mathbf{x}^{(k)}$ converges to that of $\mathbf{x}_1$. This leads to the most basic method of computing an eigenvalue and eigenvector, the *Power Method:*

Choose an initial vector $\mathbf{q}_0$ such that $\|\mathbf{q}_0\|_2 = 1$
for $k = 1, 2, \ldots$ **do**
    $\mathbf{z}_k = A\mathbf{q}_{k-1}$
    $\mathbf{q}_k = \mathbf{z}_k / \|\mathbf{z}_k\|_2$
end

This algorithm continues until $\mathbf{q}_k$ converges to within some tolerance. If it converges, it converges to a unit vector that is a scalar multiple of $\mathbf{x}_1$, an eigenvector corresponding to the largest eigenvalue, λ_1. The rate of convergence is $|\lambda_1/\lambda_2|$, meaning that the distance between $\mathbf{q}_k$ and a vector parallel to $\mathbf{x}_1$ decreases by roughly this factor from iteration to iteration.

It follows that convergence can be slow if λ_2 is almost as large as λ_1, and in fact, the power method fails to converge if $|\lambda_2| = |\lambda_1|$, but $\lambda_2 \neq \lambda_1$ (for example, if they have opposite signs). It is worth noting the implementation detail that if λ_1 is negative, for example, it may appear that $\mathbf{q}_k$ is not converging, as it "flip-flops" between two vectors. This is remedied by normalizing $\mathbf{q}_k$ so that it is not only a unit vector, but also a positive number.

Once the normalized eigenvector $\mathbf{x}_1$ is found, the corresponding eigenvalue λ_1 can be computed using a Rayleigh quotient. Then, *deflation* can be carried out by constructing a Householder reflection P_1 so that $P_1\mathbf{x}_1 = \mathbf{e}_1$, as discussed previously, and then P_1AP_1 is a matrix with block upper-triangular structure. This decouples the problem of computing the eigenvalues of A into the (solved) problem of computing λ_1, and then computing the remaining eigenvalues by focusing on the lower right $(n-1) \times (n-1)$ submatrix.

This method can be impractical, however, due to the contamination of smaller eigenvalues by roundoff error from computing the larger ones and then deflating. An alternative is to compute several eigenvalues "at once" by using a block version of the Power Method, called *Orthogonal Iteration*. In this method, A is multiplied by an $n \times r$ matrix, with $r > 1$, and then the normalization of the vector computed by the power method is generalized to the *orthogonalization* of the block, through the QR factorization. The method is as follows:

Choose an $n \times r$ matrix Q_0 such that $Q_0^H Q_0 = I_r$
for $k = 1, 2, \ldots$ **do**
    $Z_k = AQ_{k-1}$
    $Z_k = Q_k R_k$ (QR Factorization)
end

Generally, this method computes a convergent sequence $\{Q_k\}$, as long as Q_0 is not deficient in the directions of certain eigenvectors of A^H, and $|\lambda_r| > |\lambda_{r+1}|$. From the relationship

$$R_k = Q_k^H Z_k = Q_k^H A Q_{k-1},$$

we see that if Q_k converges to a matrix Q, then $Q^H AQ = R$ is upper-triangular, and because $AQ = QR$, the columns of Q span an invariant subspace.

Furthermore, if $Q^\perp$ is a matrix whose columns span $(\text{range}(Q))^\perp$, then

$$\begin{bmatrix} Q^H \\ (Q^\perp)^H \end{bmatrix} A \begin{bmatrix} Q & Q^\perp \end{bmatrix} = \begin{bmatrix} Q^HAQ & Q^HAQ^\perp \\ (Q^\perp)^HAQ & (Q^\perp)^HAQ^\perp \end{bmatrix} = \begin{bmatrix} R & Q^HAQ^\perp \\ 0 & (Q^\perp)^HAQ^\perp \end{bmatrix}.$$

That is, $\lambda(A) = \lambda(R) \cup \lambda((Q^\perp)^HAQ^\perp)$, and because R is upper-triangular, the eigenvalues of R are its diagonal elements. We conclude that Orthogonal Iteration, when it converges, yields the largest r eigenvalues of A.

If we let $r = n$, then, if the eigenvalues of A are separated in magnitude, then generally Orthogonal Iteration converges, yielding the Schur Decomposition of A, $A = QTQ^H$. However, this convergence is generally quite slow. Before determining how convergence can be accelerated, we examine this instance of Orthogonal Iteration more closely.

For each integer k, we define $T_k = Q_k^HAQ_k$. Then, from the algorithm for Orthogonal Iteration, we have

$$T_{k-1} = Q_{k-1}^HAQ_{k-1} = Q_{k-1}^HZ_k = (Q_{k-1}^HQ_k)R_k,$$

and

$$\begin{aligned} T_k &= Q_k^HAQ_k \\ &= Q_k^HAQ_{k-1}Q_{k-1}^HQ_k \\ &= Q_k^HZ_kQ_{k-1}^HQ_k \\ &= Q_k^HQ_kR_k(Q_{k-1}^HQ_k) \\ &= R_k(Q_{k-1}^HQ_k). \end{aligned}$$

That is, T_k is obtained from T_{k-1} by computing the QR factorization of T_{k-1}, and then multiplying the factors in reverse order. Equivalently, T_k is obtained by applying a unitary similarity transformation to T_{k-1}, as

$$T_k = R_k(Q_{k-1}^HQ_k) = (Q_{k-1}^HQ_k)^HT_{k-1}(Q_{k-1}^HQ_k) = U_k^HT_{k-1}U_k.$$

If Orthogonal Iteration converges, then T_k converges to an upper-triangular matrix $T = Q^HAQ$ whose diagonal elements are the eigenvalues of A. This simple process of repeatedly computing the QR factorization and multiplying the factors in reverse order is called the *QR Iteration*, which proceeds as follows:

Choose Q_0 so that $Q_0^HQ_0 = I_n$ $T_0 = Q_0^HAQ_0$
for $k = 1, 2, \ldots$ **do** (QR factorization)
    $T_{k-1} = U_kR_k$
    $T_k = R_kU_k$
end

It is this version of Orthogonal Iteration that serves as the cornerstone of an efficient algorithm for computing all of the eigenvalues of a matrix. As described, QR iteration is prohibitively expensive, because $O(n^3)$ operations are required in *each* iteration to compute the QR factorization of T_{k-1}, and typically, many iterations are needed to obtain convergence. However, we will see that with a judicious choice of Q_0, the amount of computational effort can be drastically reduced.

It should be noted that if A is a real matrix with complex eigenvalues, then Orthogonal Iteration or the QR Iteration will not converge, due to distinct eigenvalues having equal magnitude. However, the *structure* of the matrix T_k in QR Iteration generally will converge to "quasi-upper-triangular" form, with 1×1 or 2×2 diagonal blocks corresponding to real eigenvalues or complex-conjugate pairs of eigenvalues, respectively. It is this type of convergence that we will seek in our continued development of the QR Iteration.

The Symmetric Eigenvalue Problem

The Power Method, when applied to a symmetric matrix to obtain its largest eigenvalue, is more effective than for a general matrix: its rate of convergence $|\lambda_2/\lambda_1|^2$, meaning that it generally converges twice as rapidly.

Let A be an $n \times n$ symmetric matrix. Even more rapid convergence can be obtained if we consider a variation of the Power Method. *Inverse Iteration* is the Power Method applied to $(A - \mu I)^{-1}$. The algorithm is as follows:

Choose $\mathbf{x}_0$ so that $\|\mathbf{x}_0\|_2 = 1$
for $k = 0, 1, 2, \ldots$ **do**
  Solve $(A - \mu I)\mathbf{z}_k = \mathbf{x}_k$ for $\mathbf{z}_k$
  $\mathbf{x}_{k+1} = \mathbf{z}_k / \|\mathbf{z}_k\|_2$
end

Let A have eigenvalues $\lambda_1, \ldots, \lambda_n$. Then, the eigenvalues of $(A - \mu I)^{-1}$ matrix are $1/(\lambda_i - \mu)$, for $i - 1, 2, \ldots, n$. Therefore, this method finds the eigenvalue that is closest to μ.

Now, suppose that we vary μ from iteration to iteration, by setting it equal to the *Rayleigh quotient*

$$r(\mathbf{x}) = \frac{\mathbf{x}^H A \mathbf{x}}{\mathbf{x}^H \mathbf{x}},$$

of which the eigenvalues of A are constrained extrema. We then obtain *Rayleigh Quotient Iteration*:

Choose a vector $\mathbf{x}_0$, $\|\mathbf{x}_0\|_2 = 1$
for $k = 0, 1, 2, \ldots$ **do**
  $\mu_k = \mathbf{x}_k^H A \mathbf{x}_k$
  Solve $(A - \mu_k I)\mathbf{z}_k = \mathbf{x}_k$ for $\mathbf{z}_k$
  $\mathbf{x}_{k+1} = \mathbf{z}_k / \|\mathbf{z}_k\|_2$
end

When this method converges, it converges *cubically* to an eigenvalue-eigenvector pair. To see this, consider the diagonal 2×2 matrix

$$A = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}, \quad \lambda_1 > \lambda_2.$$

This matrix has eigenvalues λ_1 and λ_2, with eigenvectors $\mathbf{e}_1$ and $\mathbf{e}_2$. Suppose that $\mathbf{x}_k = \begin{bmatrix} c_k & s_k \end{bmatrix}^T$, where $c_k^2 + s_k^2 = 1$. Then we have

$$\mu_k = r(\mathbf{x}_k) = \begin{bmatrix} c_k & s_k \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} c_k \\ s_k \end{bmatrix} = \lambda_1 c_k^2 + \lambda_2 s_k^2.$$

From

$$A - \mu_k I = \begin{bmatrix} \lambda_1 - (\lambda_1 c_k^2 + \lambda_2 s_k^2) & 0 \\ 0 & \lambda_2 - (\lambda_1 c_k^2 + \lambda_2 s_k^2) \end{bmatrix} = (\lambda_1 - \lambda_2) \begin{bmatrix} s_k^2 & 0 \\ 0 & -c_k^2 \end{bmatrix},$$

we obtain

$$\mathbf{z}_k = \frac{1}{\lambda_1 - \lambda_2} \begin{bmatrix} c_k/s_k^2 \\ -s_k/c_k^2 \end{bmatrix} = \frac{1}{c_k^2 s_k^2 (\lambda_1 - \lambda_2} \begin{bmatrix} c_k^3 \\ -s_k^3 \end{bmatrix}.$$

Normalizing yields

$$\mathbf{x}_{k+1} = \frac{1}{\sqrt{c_k^6 + s_k^6}} \begin{bmatrix} c_k^3 \\ -s_k^3 \end{bmatrix},$$

which indicates cubic convergence to a vector that is parallel to $\mathbf{e}_1$ or $\mathbf{e}_2$, provided $|c_k| \neq |s_k|$.

It should be noted that Inverse Iteration is also useful for a general (unsymmetric) matrix A, for finding *selected* eigenvectors after computing the Schur decomposition $A = QTQ^H$, which reveals the eigenvalues of A, but not the eigenvectors. Then, a computed eigenvalue can be used as the shift μ, causing rapid convergence to a corresponding eigenvector. In fact, in practice a single iteration is sufficient. However, when no such information about eigenvalues is available, Inverse Iteration is far more practical for a symmetric matrix than an unsymmetric matrix, due to the superior convergence of the Power Method in the symmetric case.