САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПРИОРИТЕТНЫЙ НАЦИОНАЛЬНЫЙ ПРОЕКТ "ОБРАЗОВАНИЕ"

Проект «Инновационная образовательная среда в классическом университете»

Пилотный проект № 22 «Разработка и внедрение инновационной образовательной программы «Прикладные математика и физика»»

Физический факультет

Кафедра
Вычислительной физики

С.Л. Яковлев, Е.А. Яревский

Вычислительные алгоритмы

Учебно-методическое пособие

.

Санкт Петербург
2009 г.

- Рецензент: зав. кафедрой высшей математики и математической физики, д.ф.-м.н., проф. Буслаев В.С.
- Печатается по решению методической комиссии физического факультета СПбГУ.
- Рекомендовано Ученым советом физического факультета СПбГУ.

Вычислительные алгоритмы. – СПб., 2009

В учебно-методическом пособии рассмотрены важнейшие вычислительные задачи линейной алгебры. Подробно описаны наиболее эффективные алгоритмы решения этих задач, основанные на методах QR-разложений и методе Ланцоша. Пособие предназначено для студентов 5-7-го курсов, аспирантов, соискателей и других обучающихся.

S.L. Yakovlev, E.A. Yarevsky
Computational algorithms. St Petersburg, 2009

The major computational problems of Linear Algebra are considered. The most efficient algorithms for their solution which are based on QR-decomposition and Lanczos iteration method are given in details.

Lecture 1

Introduction and Preliminaries

1.1 The Basic Problem

For a given $n \times n$ matrix A, the *eigenvalue problem* is to find a scalar λ, called an *eigenvalue* of A, and a vector $\mathbf{x}$, called an *eigenvector* corresponding to the eigenvalue λ, such that

$$A\mathbf{x} = \lambda\mathbf{x}. \tag{1.1}$$

We may refer to the pair $(\lambda, \mathbf{x})$ as an *eigenpair.* The matrix A may be real or complex, and have other properties which we will discuss at length.

It is not clear when the matrix eigenvalue problem first arose, but it is likely that it originated in the study of a continuous problem, the motion of a vibrating string. The word "eigenvalue" is a corruption of a German word. Eigenvalues are also known as *proper values*, *characteristic roots*, as well as other terms.

Example A simple example that illustrates the importance of eigenvalues is the system of ordinary differential equations

$$\mathbf{y}'(t) = A\mathbf{y}, \tag{1.2}$$

with initial conditions

$$\mathbf{y}(0) = \mathbf{y}_0. \tag{1.3}$$

If $A\mathbf{z} = \lambda\mathbf{z}$, then

$$\mathbf{y}(t) = ae^{\lambda t}\mathbf{z},$$

where a is an arbitrary constant, satisfies the system (1.2). Of course, we need to satisfy the initial conditions (1.3), and in order to do so, we need to take a linear combination of *all* eigenvectors of A. $\square$

Now, from (1.1), we have

$$(A - \lambda I)\mathbf{x} = 0,$$

so that a solution exists if and only if

$$\det(A - \lambda I) = 0.$$

Now,

$$\det(A - \lambda I) = (a_{11} - \lambda)(a_{22} - \lambda) \dots (a_{nn} - \lambda) + \text{other terms} \tag{1.4}$$

where the other terms are polynomials in λ of degree $\leq n$. Since $\prod_{i=1}^{n}(a_{ii}-\lambda)$ is a polynomial of degree n,

$$\det(A - \lambda I) = \text{polynomial of degree} \leq n.$$

Note that by the classical expansion of the determinant, the other terms in (1.4) are of degree $\leq n$. Hence

$$(-1)^n \det(A - \lambda I) = \lambda^n + c_1\lambda^{n-1} + \cdots + c_{n-1}\lambda + c_n = \varphi_n(\lambda) \tag{1.5}$$

This polynomial (1.5) is known as the *characteristic polynomial.*

Now, it is easy to see from (1.4) that

$$c_1 = -\sum_{i=1}^{n} a_{ii} = -\sum_{i=1}^{n} \lambda_i$$

$$\text{i.e., } \operatorname{tr}(A) = \sum_{i=1}^{n} \lambda_i. \tag{1.6}$$

This relationship (1.6) is very useful, and is often used. In partcular, if all the eigenvalues of a matrix are desired, it can provide a quick check to see if the computed eigenvalues are reasonable.

Since $\det(A - \lambda I) =$ polynomial of degree n, we have immediately that every $n \times n$ matrix has n eigenvalues. Furthermore, since the coefficients of the characteristic polynomial are functions of the elements of A, the eigenvalues are a continuous function of the elements of the matrix. Note also, that if A is real, then the eigenvalues must be real or occur in complex conjugate pairs.

The Cayley-Hamilton Theorem states that every matrix satisfies its characteristic polynomial. There are many algorithms that make use of this nice property, for example consider the *Krylov vector sequence*

$$\eta_0 = \eta, \quad \eta_1 = A\eta, \quad \eta_2 = A^2\eta_1, \quad \dots, \eta_n = A^n\eta_{n-1}$$

where η is an arbitrary vector of length n and A is an $n \times n$ matrix. Then,

$$\varphi_n(A)\eta = A^n\eta + c_1A^{n-1}\eta + \cdots + c_{n-1}A\eta + c_n\eta = 0$$

which is equivalent to

$$c_1\eta_{n-1} + c_2\eta_{n-2} + \cdots + c_n\eta_0 = -\eta_n$$

or in matrix form

$$B\mathbf{c} = -\eta_n$$

where

$$B = \begin{bmatrix} \eta_{n-1} & \eta_{n-2} & \cdots & \eta_0 \end{bmatrix}, \qquad \mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}.$$

By solving this linear system we get the characteristic polynomial of A. Notice, however, that η must lie in the space of all eigenvectors of A otherwise B will be singular.

Another nice consequence of the Cayley-Hamilton Theorem is that A^n can be writen as a linear combination of the set of matrices $A^0, A^1, \ldots, A^{n-1}$ and by simple induction it can be shown that the same is true for any integer power of A. Since any analytic function of A has a power series extension, the same holds for any analytic function of A as well. Notice in particular that

$$A^{-1} = -\frac{1}{c_n}(A^{n-1} + c_1 A^{n-2} + \cdots + c_{n-1} I).$$

Obviously, the eigenvector is not unique, since if

$$A\mathbf{x} = \lambda\mathbf{x}, \qquad A(c\mathbf{x}) = \lambda(c\mathbf{x}),$$

so $c\mathbf{x}$ is also an eigenvector of A if $c \neq 0$. We usually impose the condition that $\|x\|_2 = 1$, but this is not sufficient to guarantee uniqueness as multiplication of an eigenvector by $e^{i\theta}$ will produce another eigenvector with the same norm. For this reason, the notion of different eigenvectors being linearly independent is generally assumed. The following theorem provides a sufficient condition for this to be true.

Theorem 1.1. *For an arbitrary $n \times n$ matrix A, if A has n distinct eigenvalues, then there are n linearly independent eigenvectors.*

In general, there may be fewer than n eigenvectors. For example, if

$$A = \begin{bmatrix} 2 & 100 \\ 0 & 2 \end{bmatrix}$$

then $\det(A - \lambda I) = (2 - \lambda)^2$ and hence, $\lambda_1 = \lambda_2 = 2$. But, since

$$\operatorname{rank}(A - 2I) = 1,$$

there is only one eigenvector, $\mathbf{x} = \mathbf{e_1} = \begin{bmatrix} 1 & 0 \end{bmatrix}^T$.

1.2 Well Known Example

A well known example that is of interest is the matrix

$$A = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ \vdots & & & \ddots & 1 \\ 0 & & \dots & & 0 \end{bmatrix}.$$

Since $\det(A - \lambda I) = (-\lambda)^n$, $\lambda_i = 0$ for $i = 1, 2, ..., n$. But, $\text{rank}(A - \lambda I) = (n-1)$. Notice that the $(n-1) \times (n-1)$ upper corner of A is the identity matrix. Thus, $\mathbf{x} = \mathbf{e_1}$ is the only eigenvector of this matrix.

Now, consider

$$A(\varepsilon) = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & & & \ddots & 1 \\ \varepsilon & 0 & \dots & & 0 \end{bmatrix}.$$

$$\det(A(\varepsilon) - \lambda I) = (-\lambda)^n + (-1)^{n-1}\varepsilon.$$

$$\text{So,} \qquad \lambda^n - \varepsilon = 0,$$

$$|\lambda_i| = |\varepsilon|^{\frac{1}{n}} \qquad \text{for } i = 1, 2, ..., n.$$

Since we have n distinct eigenvalues, we therefore have n eigenvectors. Note also that if $n = 10$, and $\varepsilon = 10^{-10}$, then

$$|\lambda_i| = 0.1.$$

Thus, a small perturbation in a (non-symmetric) matrix can cause a large (relative) perturbation in the eigenvalues. This is an example of a matrix that is *ill-conditioned* with regards to estimation (computation) of eigenvalues. We shall discuss this concept in greater detail later.

1.3 Useful Properties

We will need some other relationships in developing algorithms for computing eigenvalues. Some of these are:

1. If $A\mathbf{x} = \lambda\mathbf{x}$, then
$$(A + \alpha I)\mathbf{x} = (\lambda + \alpha)\mathbf{x}.$$

2. $A^2\mathbf{x} = A(A\mathbf{x}) = \lambda(A\mathbf{x}) = \lambda^2\mathbf{x}$ and
$$A^k\mathbf{x} = \lambda^k\mathbf{x}.$$

3. Let $P_k(A) = A^m(\alpha_0 A^k + \alpha_1 A^{k-1} + \cdots + \alpha_k I)$ where α_i are scalars, and m may be negative. If $P_k(A)\mathbf{x} = \mu\mathbf{x}$, then
$$\mu = \lambda^m(\alpha_0\lambda^k + \alpha_1\lambda^{k-1} + \cdots + \alpha_k) = p_k(\lambda).$$
Similarly, $P_k(A)\mathbf{x} = p_k(\lambda)\mathbf{x}$, and if F is an analytic function, $F(A)\mathbf{x} = f(\lambda)\mathbf{x}$ where f is a "scalar version" of F.

   We can often reconstruct the eigenvalues from $p_k(\lambda)$. But there can be difficulties. Suppose A has $+\lambda$ and $-\lambda$ as eigenvalues. Then A^2 has the eigenvalues λ^2 (twice). so, it may not be possible to compute the eigenvalues and eigenvectors from A^2.

   For example, consider
$$A = \begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix}.$$
A has eigenpairs $(+2, \mathbf{e_1})$ and $(-2, \mathbf{e_2})$, whereas for A^2, $\lambda_1 = \lambda_2 = 4$ and $x_1 = \alpha\mathbf{e_1} + \beta\mathbf{e_2}$ and $x_2 = \gamma\mathbf{e_1} + \delta\mathbf{e_2}$ for any α, β, γ, and δ provided
$$\begin{vmatrix} \alpha & \beta \\ \gamma & \delta \end{vmatrix} \neq 0.$$

4. If $B = Q^{-1}AQ$, then $B\mathbf{y} = \lambda\mathbf{y}$ i.e., B has the same eigenvalues as A, and $\mathbf{x} = Q\mathbf{y}$. We say that B is *similar* to A, and call the transformation $A \to Q^{-1}AQ$ a *similarity transformation* of A.

5. If A is real, then
$$\begin{aligned} A\mathbf{x} &= \lambda\mathbf{x}, & \mathbf{x} &= \mathbf{u} + i\mathbf{v} \\ A\bar{\mathbf{x}} &= \bar{\lambda}\bar{\mathbf{x}}, & \bar{\mathbf{x}} &= \bar{\mathbf{u}} - i\bar{\mathbf{v}}. \end{aligned}$$
where $\mathbf{u}$ and $\mathbf{v}$ are real valued. That is to say, if $(\lambda, \mathbf{x})$ is an eigenpair, then the complex conjugate of that eigenpair is also an eigenpair.

1.4 Spectral Decomposition

Let A have distinct eigenvalues and let X be the matrix whose columns are eigenvectors of A.
$$\begin{aligned} AX &= A[x_1, x_2, ..., x_n] \\ &= [\lambda_1 x_1, \lambda_2 x_2, ..., \lambda_n x_n] \\ &= X\Lambda \end{aligned}$$

where $\Lambda = \text{diag}(\lambda_1, \lambda_2, ..., \lambda_n)$. Therefore, by theorem 1.1:

$$\boxed{A = X\Lambda X^{-1}}.$$

This decomposition is referred to as the *spectral decomposition* of A, and any such A is called *diagonalizable*.

1.5 More Examples and Applications

1. Suppose A has n linearly independent eigenvectors. Then:

$$\begin{aligned} A^2 &= X\Lambda X^{-1} \cdot X\Lambda X^{-1} \\ &= X\Lambda^2 X^{-1} \\ &\vdots \\ A^k &= X\Lambda^k X^{-1} \end{aligned}$$

   and,

$$\begin{aligned} P_k(A) &= XP_k(\Lambda)X^{-1} \\ F(A) &= XF(\Lambda)X^{-1} \end{aligned}$$

   if F is an analytic function. One should note that these are not unique factorizations since the eigenvectors which comprise the columns of X are not unique.

2. If $|\lambda_i| < 1$ for $i = 1, ..., n$, then $A^k \to 0$ as $k \to \infty$.

3. Consider the following system of linear differential equations (assume that A is diagonalizable):

$$\mathbf{y}'^{(x)} = A\mathbf{y}(x) \qquad \text{with} \qquad \mathbf{y}(0) = \mathbf{y}^{(0)}.$$

   Now if $A = Q\Lambda Q^{-1}$,

$$Q^{-1}\mathbf{y}'^{(x)} = \Lambda Q^{-1}\mathbf{y}(x).$$

   Thus, if:

$$\begin{aligned} \mathbf{z}(x) &= Q^{-1}\mathbf{y}(x) \\ \mathbf{z}'(x) &= \Lambda\mathbf{z}(x) \end{aligned}$$

or

$$\begin{aligned}
z_j'(x) &= \lambda_j z_j(x) \\
z_j(x) &= e^{\lambda_j x} z_j^{(0)}(x) \\
y(x) &= Q \begin{bmatrix} e^{\lambda_1 x} & & & 0 \\ & e^{\lambda_2 x} & & \\ & & \ddots & \\ 0 & & & e^{\lambda_n x} \end{bmatrix} Q^{-1} \mathbf{y}^{(0)} \\
&\equiv e^{Ax} \mathbf{y}^{(0)}.
\end{aligned}$$

Hence, if Real$(\lambda_i) < 0$ for $i = 1, ..., n$, then

$$\mathbf{y}(x) \to \mathbf{0} \qquad \text{as} \qquad x \to \infty.$$

The matrix e^A is referred to as the *matrix exponential.* Recall that e^A can be represented as a polynomial in A.

1.6 Jordan Canonical Form

Of course, the situation becomes more complicated when A cannot be diagonalized. Then

$$\begin{aligned}
A &= QJQ^{-1}, \\
J &= \begin{bmatrix} J_1 & & & 0 \\ & J_2 & & \\ & & \ddots & \\ 0 & & & J_r \end{bmatrix}
\end{aligned}$$

with

$$J_i = \begin{bmatrix} \lambda_i & 1 & & 0 \\ & \ddots & \ddots & \\ & & & 1 \\ 0 & & & \lambda_i \end{bmatrix}_{n_i \times n_i}$$

Note that rank$(J_i - \lambda_i I) = (n_i - 1)$ so there is only one eigenvector that corresponds to λ_i. This decomposition is known as *Jordan Canonical Form*, or JCF.

Obviously,

$$\begin{aligned}
A^k &= QJ^kQ^{-1} \\
&= Q\begin{bmatrix} J_1^k & & & 0 \\ & J_2^k & & \\ & & \ddots & \\ 0 & & & J_r^k \end{bmatrix}Q^{-1} \\
J_i^k &= \begin{bmatrix} \lambda_i & 1 & & 0 \\ & \ddots & \ddots & \\ & & & 1 \\ 0 & & & \lambda_i \end{bmatrix}^k_{n_i\times n_i} \\
&= (\lambda_i I + K_i)^k
\end{aligned}$$

where

$$K_i = \begin{bmatrix} 0 & 1 & & 0 \\ & \ddots & \ddots & \\ & & & 1 \\ 0 & & & 0 \end{bmatrix}_{n_i\times n_i}.$$

Then, $(\lambda_i I + K_i)^k = \sum_{p=0}^{k}\binom{k}{p}\lambda_i^p K_i^{k-p}$, noting that

$$K_i^2 = \begin{bmatrix} 0 & 0 & 1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & & & 1 \\ & & 0 & & 0 \\ & & & & 0 \end{bmatrix}$$

$$q < n_i : \qquad K_i^q = \begin{bmatrix} \overbrace{0 \ \ \dots \ \ 0}^{q \text{ diags}} & 1 & & 0 \\ \ddots & & \ddots \quad \ddots & \\ & & & 1 \\ & 0 & & \left.\begin{matrix}0\\ \vdots \\ 0\end{matrix}\right\} \end{bmatrix}.$$

And as a result of the Cayley-Hamilton Theorem, if $q \geq n_i$, $K_i^q = 0$. Notice that raising K_i to the q^{th} power moves the 1's into the q^{th} diagonal, if we call the principal diagonal the 0^{th} diagonal as in MATLAB.

Thus for $k \geq n_i$,

$$J_i^k = \begin{bmatrix} \lambda_i^k & \binom{k}{1}\lambda_i^{k-1} & \binom{k}{2}\lambda_i^{k-2} & \cdots \\ & \ddots & \ddots & \\ & & & \\ & & & \lambda_i^k \end{bmatrix}_{n_i \times n_i}.$$

Here again, we note that when $|\lambda_i| < 1$ for $i = 1, ..., n$,

$$A^k \to \infty \qquad \text{as} \qquad k \to \infty.$$

But, convergence may be quite slow. For example:

$$\begin{aligned} A &= \begin{bmatrix} \frac{1}{2} & 1 \\ 0 & \frac{1}{2} \end{bmatrix}^{10} \\ &= \begin{bmatrix} \frac{1}{2^{10}} & \frac{10}{2^{10}} \\ 0 & \frac{1}{2^{10}} \end{bmatrix} \\ &\approx \begin{bmatrix} 10^{-3} & 10^{-2} \\ 0 & 10^{-3} \end{bmatrix}. \end{aligned}$$

As you can see from our exmple in §1.2, a small perturbation, such as $A \to A(\varepsilon)$ can greatly affect the Jordan Canonical Form. The JCF of A is given by $A = IAI$ (at least, this is one possibile form), but the JCF of $A(\varepsilon)$ is greatly different since J for this matrix is diagonal. For this reason, the Jordan Canonical Form is considered unstable.

1.7 Schur Decomposition

There are other decompositions which will play a major role in our studies. One such decomposition is the *Schur Decomposition.* A rather uninsightful proof of this decomposition is as follows: Let $A = QJQ^{-1}$ be the Jordan Canonical Form of A, and let $Q = UT$ be the QR-factorization of Q (hence $U^*U = I$ and T is upper triangular). Then:

$$\begin{aligned} A &= UTJT^{-1}U^* \\ &= U \begin{bmatrix} \times & \times & \cdots & \times \\ & \times & \ddots & \vdots \\ & & \ddots & \times \\ 0 & & & \times \end{bmatrix} \begin{bmatrix} J_1 & & & 0 \\ & J_2 & & \\ & & \ddots & \\ 0 & & & J_r \end{bmatrix} \begin{bmatrix} \times & \times & \cdots & \times \\ & \times & \ddots & \vdots \\ & & \ddots & \times \\ 0 & & & \times \end{bmatrix} U^* \\ &= URU^*, \end{aligned}$$

where R is upper triangular. Note, we shall use $\times$ to denote elements of a matrix that do not hold important values. This shorthand is quite useful for describing sparsity patterns.

Later, we shall give a more constructive proof of the Schur Decomposition.

Lecture 2

More Decompositions

2.1 Constructing the Schur Decomposition

We now give a more constructive proof that every matrix A has a *Schur Decomposition*

$$A = UTU^* \tag{2.1}$$

where U is a unitary matrix, T is an upper triangular matrix, and the diagonal elements of T are the eigenvalues of A.

Suppose that we have computed an eigenvector $\mathbf{x}_1$ with $\|\mathbf{x}_1\|_2 = 1$, whose first element, x_1, is real and non-negative. Now we can construct a Householder Matrix, $P^{(1)} = I - 2\mathbf{u}_1\mathbf{u}_1^*$ with $\|\mathbf{u}_1\|_2 = 1$, such that

$$\begin{aligned} P^{(1)}\mathbf{x}_1 &= \mathbf{e}_1 \\ \mathbf{x}_1 = P^{(1)^*}\mathbf{e}_1 = P^{(1)}\mathbf{e}_1 &= \begin{pmatrix} 1 - 2|u_1|^2 \\ -2u_2\bar{u}_1 \\ \vdots \\ -2u_n\bar{u}_1 \end{pmatrix} \\ |u_1|^2 &= \frac{1}{2}(1 - x_1). \end{aligned}$$

If $x_1 = \rho e^{i\theta}$, we will renormalize $\mathbf{x}_1$ so that $\mathbf{x}_1 := e^{-i\theta}\mathbf{x}_1$. Hence,

$$\begin{aligned} |u_1|^2 &= \frac{1}{2}(1 - \rho) \\ u_j &= -\frac{x_j}{2\bar{u}_1}. \end{aligned}$$

Note that Householder matrices also satisfy $P = P^*$ and $PP^* = P^*P = I$. It is important to note that we would not formally store P, but rather we would store $\mathbf{u}$ and perform the matrix-vector multiply, $P\mathbf{z}$, according to $\mathbf{z} - 2(\mathbf{u}^*\mathbf{z})\mathbf{u}$. The computational cost of this matrix-vector multiplier is also reduced from $\mathcal{O}\left(n^2\right)$ scalar multiplies to $\mathcal{O}\left(2n\right)$ multiplies.

The next step in the construction of the Schur Decomposition is the application of a similarity transformation using $P^{(1)}$.

$$\begin{aligned}
A^{(2)} &= P^{(1)*}AP^{(1)} \\
&= P^{(1)*}A[\mathbf{x}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n] \\
&= P^{(1)*}[\lambda \mathbf{x}_1, A\mathbf{p}_2, \ldots, A\mathbf{p}_n] \\
&= \begin{bmatrix} \mathbf{x}_1^* \\ \mathbf{p}_2^* \\ \vdots \\ \mathbf{p}_n^* \end{bmatrix} [\lambda \mathbf{x}_1, A\mathbf{p}_2, \ldots, A\mathbf{p}_n] \\
&= \begin{bmatrix} \lambda & \mathbf{x}_1^* A\mathbf{p}_2 & \ldots & \mathbf{x}_1^* A\mathbf{p}_n \\ 0 & \mathbf{p}_2^* A\mathbf{p}_2 & \ldots & \mathbf{p}_2^* A\mathbf{p}_n \\ \vdots & \vdots & & \vdots \\ 0 & \mathbf{p}_n^* A\mathbf{p}_2 & \ldots & \mathbf{p}_n^* A\mathbf{p}_n \end{bmatrix} \\
&= \begin{bmatrix} \lambda & \times & \ldots & \times \\ 0 & a_{22}^{(2)} & \ldots & a_{2n}^{(2)} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n2}^{(2)} & \ldots & a_{nn}^{(2)} \end{bmatrix} \\
&= \begin{bmatrix} \lambda & \times^T \\ \mathbf{0} & \tilde{A}^{(2)} \end{bmatrix}
\end{aligned}$$

Since we used a similarity transformation to construct $A^{(2)}$, the eigenvalues of $A^{(2)}$ equal those of A. Furthermore, λ is an eigenvalue of A and $A^{(2)}$, hence the remaining eigenvalues of A must be the eigenvalues of $\tilde{A}^{(2)}$. In addition, $\tilde{A}^{(2)}$ must have at least one eigenvector. So, we can repeat this process then acting only on $\tilde{A}^{(2)}$. The new $P^{(2)}$ will be of the form $\begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \tilde{P}^{(2)} \end{bmatrix}$.

This is an example of a technique known as *deflation* wherein one gradually reduces the dimensionality of the problem being solved. Ultimately, this process constructs the Schur Decomposition if the matrix has n eigenvectors as follows:

$$\begin{aligned}
A^{(n)} &= \begin{bmatrix} \lambda_1 & \times & \cdots & \times \\ & \lambda_2 & \ddots & \vdots \\ & & \ddots & \times \\ & 0 & & \lambda_n \end{bmatrix} = R \\
&= P^{(n-1)*}A^{(n-1)}P^{(n-1)} \\
&= P^{(n-1)*}P^{(n-2)*}\ldots P^{(1)*}AP^{(1)}\ldots P^{(n-2)}P^{(n-1)} \\
&= Q^*AQ \\
\therefore A &= QRQ^*
\end{aligned}$$

This construction has been shown to have very good numerical stability properties.

2.2 Normal Matrices and Departure from Normality

If $A = A^*$, then R is a diagonal matrix. Furthermore, if A is symmetric, $A = Q\Lambda Q^T$, where $\Lambda = \text{diag}(\lambda_1, ..., \lambda_n)$, and $Q^TQ = QQ^T = I$. This is a special case of normal matrices. A matrix is said to be *normal* if $AA^* = A^*A$. Matrices which are real and symmetric, skew-symmetric ($A = -A^T$), orthogonal, or hermitian, are other examples of normal matrices. But, complex symmetric matrices are not normal.

If A is normal,

$$A = QDQ^*$$

where $Q^*Q = I$, and $D = \text{diag}(\lambda_1, ..., \lambda_n)$.

The Schur decomposition gives

$$A = QTQ^*$$

where $T = D + N$, D is the diagonal part of T, and N is the strictly upper triangular part. Thus,

$$\begin{aligned}
\|A\|_F^2 &= \sum_{i,j} |a_{ij}|^2 \\
&= \|T\|_F^2 = \|D\|_F^2 + \|N\|_F^2 \\
\Delta(A) &= \|N\|_F^2 = \|A\|_F^2 - \|D\|_F^2 \\
&= \|A\|_F^2 - \sum_i |\lambda_i|^2 \\
&= < \textit{Departure from Normality} >
\end{aligned}$$

2.3 Murgnahan-Wintner Form

An eigenvector defines a one-dimensional subspace of $\mathbb{C}^n$. More generally, a subspace $\mathcal{S} \subseteq \mathbb{C}^n$ with the property

$$\mathbf{x} \in \mathcal{S} \qquad \Rightarrow \qquad A\mathbf{x} \in \mathcal{S}$$

is said to be an *invariant* subspace for A. Thus if

$$AX = XB \qquad X \in \mathbb{C}^{n\times k}, \qquad B \in \mathbb{C}^{k\times k}$$

then span(X) is invariant and

$$\begin{aligned}
B\mathbf{y} &= \lambda\mathbf{y} \Rightarrow \\
A(X\mathbf{y}) &= XB\mathbf{y} = \lambda X\mathbf{y}.
\end{aligned}$$

So, if X has full column rank, then $AX = XB$ implies $\lambda(B) \subseteq \lambda(A)$ (i.e., each eigenvalue of B is an eigenvalue of A). Thus, if we know an invariant subspace (that is spanned by X), then we can reduce our search for eigenvalues of A to those of the smaller matrix B, then continue to look in a different subspace for the remaining eigenvalues.

Now, assume that X has full rank, and $AX = XB$:

$$\begin{aligned} X &= Q \begin{bmatrix} R_1 \\ 0 \end{bmatrix}, \qquad Q \in \mathbb{C}^{n\times n}, R_1 \in \mathbb{C}^{p\times p} \\ AQ \begin{bmatrix} R_1 \\ 0 \end{bmatrix} &= Q \begin{bmatrix} R_1 \\ 0 \end{bmatrix} B \\ Q^*AQ \begin{bmatrix} R_1 \\ 0 \end{bmatrix} &= \begin{bmatrix} R_1 \\ 0 \end{bmatrix} B \\ Q^*AQ &\equiv \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix} = T \\ T \begin{bmatrix} R_1 \\ 0 \end{bmatrix} &= \begin{bmatrix} R_1 \\ 0 \end{bmatrix} B \\ T_{11}R_1 &= R_1 B \\ T_{21}R_1 &= 0 \Rightarrow T_{21} = 0. \end{aligned}$$

Note, the eigenvalues of T are those of A, and since X has full rank, R has full rank, and the conclusion on T_{21} follows. Hence,

$$Q^*AQ = \begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix}.$$

One can verify that

$$\lambda(A) = \lambda(T_{11}) \cup \lambda(T_{22}).$$

This is a generalization of the Schur Theorem.

Suppose that A is real. Then if $\lambda = \alpha + i\beta$ is an eigenvalue (where $\beta \neq 0$), $\bar{\lambda} = \alpha - i\beta$ is also an eigenvalue. Furthermore, $\mathbf{x} = \mathbf{u} + i\mathbf{v}$ is an eigenvector that corresponds with λ, and $\bar{\mathbf{x}} = \mathbf{u} - i\mathbf{v}$ is an eigenvector that corresponds with $\bar{\lambda}$. Thus $X = [\mathbf{u}, \mathbf{v}]$ is an invariant subspace since:

$$\begin{aligned} A[\mathbf{u},\mathbf{v}] \begin{bmatrix} 1 & 1 \\ i & -i \end{bmatrix} &= [\mathbf{u},\mathbf{v}] \begin{bmatrix} 1 & 1 \\ i & -i \end{bmatrix} \begin{bmatrix} \lambda & 0 \\ 0 & \bar{\lambda} \end{bmatrix} \\ A[\mathbf{u},\mathbf{v}] &= [\mathbf{u},\mathbf{v}]B \\ \text{where} \qquad B &= \begin{bmatrix} 1 & 1 \\ i & -i \end{bmatrix} \begin{bmatrix} \lambda & 0 \\ 0 & \bar{\lambda} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ i & -i \end{bmatrix}^{-1} = \begin{bmatrix} \alpha & \beta \\ -\beta & \alpha \end{bmatrix} \end{aligned}$$

If A is a real matrix, we can construct a real Q and T so that

$$Q^T A Q = \begin{bmatrix} T_{1,1} & \times & & & \cdots & & \times \\ & T_{2,2} & & & & & \\ & & \ddots & \ddots & & & \\ & & & T_{k,k} & \times & & \vdots \\ & & & & t_{2k+1,2k+1} & \ddots & \\ & 0 & & & & \ddots & \times \\ & & & & & & t_{n,n} \end{bmatrix}$$

where $T_{i,i}$ is a real 2×2 matrix, with complex eigenvalues. This is different than the Schur Form in that a quasi-upper triangular matrix is constructed ("quasi" in that there are 2×2 dense blocks along the diagonal). This is called the *Murgnahan-Wintner Form.*

2.4 Singular Value Decomposition

$$\begin{aligned} \underset{\substack{m \times n \\ m \geq n}}{A} &= U \Sigma V^* \end{aligned}$$

is the *Singular Value Decomposition*, where

$$U^* U = I_{m \times m}, \qquad V^* V = I_{n \times n}, \qquad \Sigma = \left[\begin{array}{ccc} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_n \\ \hline & 0 & \end{array}\right]_{m \times n},$$

each σ_i is real, and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$. Other forms also exist with related properties.

$$\text{rank}(A) = <\text{ number of singular values which are non-zero }> .$$

Note:

$$\begin{aligned} AA^* &= U \Sigma V^* V \Sigma^* U^* \\ &= U \Sigma \Sigma^* U^* \\ &= U \left[\begin{array}{ccc|c} \sigma_1^2 & & 0 & \\ & \ddots & & 0 \\ 0 & & \sigma_n^2 & \\ \hline & 0 & & 0 \end{array}\right]_{m \times m} U^* \end{aligned}$$

Thus, the columns of U are the eigenvectors of AA^*. Similarly, the columns of V are the eigenvectors of A^*A, and σ_i^2 are the eigenvalues of A^*A.

Let

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

rank$(A) = 2$, and $\lambda_i = 0$ for $i = 1, 2, 3$. But:

$$A^T A = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

so $\sigma_1 = \sigma_2 = 1$ and $\sigma_3 = 0$. Thus, the number of non-zero eigenvalues can only provide a lower bound for the rank of the matrix, yet the number of non-zero singular values tell us the rank.

Also,

$$\begin{aligned} \|A\|_F^2 &= \|U\Sigma V^*\|_F^2 = \|\Sigma\|_F^2 \\ &= \sigma_1^2 + \sigma_2^2 + \cdots + \sigma_n^2 \\ \Delta(A) = \|N\|_F^2 &= \sum_{i=1}^{n} (\sigma_i^2 - |\lambda_i|^2). \end{aligned}$$

The singular value decomposition also tells us how to construct a nearby matrix of lower rank. Let $A = U\Sigma V^* = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^*$ (i.e., express A as the sum of r-rank 1 matrices). Find a matrix, B, such that rank$(B) = k < r$ and

$$\|A - B\|_F = \min.$$

It is easy to show that the answer to this problem is:

$$B = A_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^*$$

and

$$\|A - B\|_F^2 = \sigma_{k+1}^2 + \cdots + \sigma_r^2.$$

A similar problem which is often of interest is the following:

Given A where $A = \hat{A} + E$ and E represents the error in A, furthermore $\|E\| < \varepsilon$ is known. Find a matrix, B, of minimal rank such that:

$$\|A - B\|_F < \varepsilon.$$

Lecture 3

More on SVD & Generation of Eigenvalue Problems

3.1 SVD and the Matrix 2-Norm

The 2-norm of a matrix A can be bounded using the SVD as follows:

$$\begin{aligned}\|A\|_2 &= \max_{\mathbf{x}\neq\mathbf{0}} \frac{\|A\mathbf{x}\|_2}{\|\mathbf{x}\|_2} \\ &= \max_{V^T\mathbf{x}\neq\mathbf{0}} \frac{\|U\Sigma V^T\mathbf{x}\|_2}{\|V^T\mathbf{x}\|_2} \\ &= \max_{\mathbf{y}\neq 0} \frac{\|\Sigma\mathbf{y}\|_2}{\|\mathbf{y}\|_2} \\ &= \max_{\mathbf{y}\neq 0} \sqrt{\frac{\sum_{i=1}^n \sigma_i^2 y_i^2}{\sum_{i=1}^n y_i^2}} \\ &\leq \sigma_1.\end{aligned}$$

However, this bound is achieved by setting $\mathbf{y} = \mathbf{e}_1$ (i.e., $\mathbf{x} = V\mathbf{e}_1$). Thus,

$$\|A\|_2 = \sigma_1(A).$$

$\|\cdot\|_2$ is sometimes called the *spectral norm.*

3.2 Canonical Correlation

Another useful application of the SVD is to describe the angle between two spaces. Let $A_{m\times n}$ and $B_{m\times p}$. Consider arbitrary vectors, $\mathbf{x}$ and $\mathbf{y}$ in the range of A and B respectively (i.e., $\mathbf{x} = A\xi$ and $\mathbf{y} = B\eta$). Then for a given ξ and η, the angle between $\mathbf{x}$ and $\mathbf{y}$ is defined as:

$$\cos(\mathbf{x},\mathbf{y}) = \frac{|\mathbf{x}^T\mathbf{y}|}{\|\mathbf{x}\|_2\|\mathbf{y}\|_2}.$$

We can then define the angle between range(A) and range(B) as follows:

$$\begin{aligned}\cos(A,B) &= \max_{\xi,\eta} \frac{|\mathbf{x}^T\mathbf{y}|}{\|\mathbf{x}\|_2\|\mathbf{y}\|_2}\\ &= \max_{\xi,\eta} \frac{|\xi^T A^T B\eta|}{\|A\xi\|_2\|B\eta\|_2}\\ &= \max_{\xi,\eta} \frac{|\xi^T A^T B\eta|}{\|\xi\|_2\|\eta\|_2}.\end{aligned}$$

Let us suppose further that A and B have QR-decompositions

$$A_{m\times n} = Q_{m\times m}\begin{bmatrix} R_{n\times n}\\ 0_{(m-n)\times n}\end{bmatrix}, \qquad B_{m\times r} = W_{m\times m}\begin{bmatrix} T_{r\times r}\\ 0_{(m-r)\times r}\end{bmatrix}.$$

Then,

$$\begin{aligned}\cos(A,B) &= \max_{\xi,\eta} \frac{\left|\xi^T\begin{bmatrix}R^T & 0^T\end{bmatrix}Q^T W\begin{bmatrix}T\\0\end{bmatrix}\eta\right|}{\left\|Q\begin{bmatrix}R\\0\end{bmatrix}\xi\right\|_2\left\|W\begin{bmatrix}T\\0\end{bmatrix}\eta\right\|_2}\\ &= \max_{\mathbf{g},\mathbf{h}} \frac{\left|\mathbf{g}^T\begin{bmatrix}I_n & 0\end{bmatrix}Q^T W\begin{bmatrix}I_r\\0\end{bmatrix}\mathbf{h}\right|}{\|\mathbf{g}\|_2\|\mathbf{h}\|_2},\end{aligned}$$

where $\mathbf{g} = R\xi$ and $\mathbf{h} = T\eta$.

If we partition $Q = \begin{bmatrix}Q_n & Q_{m-n}\end{bmatrix}$ and $W = \begin{bmatrix}W_r & W_{m-r}\end{bmatrix}$, then

$$\begin{bmatrix}I_n & 0\end{bmatrix}Q^T W\begin{bmatrix}I_r\\0\end{bmatrix} = Q_n^T W_r.$$

Recall that the largest eigenvalue of a symmetric matrix can be obtained using the Rayleigh Quotient:

$$\lambda_1(A) = \max_{\mathbf{x}\neq 0}\frac{|\mathbf{x}^T A\mathbf{x}|}{\|\mathbf{x}\|_2^2}.$$

Similarly,

$$\begin{aligned}\max_{\mathbf{x},\mathbf{y}\neq 0}\frac{|\mathbf{x}^T A\mathbf{y}|}{\|\mathbf{x}\|_2\|\mathbf{y}\|_2} &\leq \max_{\mathbf{x},\mathbf{y}}\frac{\|A\mathbf{x}\|_2\|\mathbf{y}\|_2}{\|\mathbf{x}\|_2\|\mathbf{y}\|_2}\\ &= \max_{\mathbf{x}\neq 0}\frac{\|A\mathbf{x}\|_2}{\|\mathbf{x}\|_2}\\ &= \sigma_1(A)\end{aligned}$$

where the inequality is obtained by the Cauchy-Schwarz theorem. Furthermore, the upper bound is attainable by letting $\mathbf{x}$ be the first column of U and $\mathbf{y}$ be the first column of V, and therefore

$$\sigma_1(A) = \max_{\mathbf{x},\mathbf{y}\neq 0} \frac{|\mathbf{x}^T A\mathbf{y}|}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2}.$$

It follows that the angle between range(A) and range(B) is given by

$$\cos(A, B) = \sigma_1(Q_n^T W_r) \leq 1.$$

In statistics, this calculation is also referred to as the *canonical correlation* of A and B. This concept also rises in biomedical applications.

3.3 SVD and Jordan Canonical Form

One potential problem that arises when computing the Jordan Canonical Form is that there is no *a priori* knowledge as to how many eigenvectors a matrix has, or, in particular, how many eigenvectors correspond to a particular eigenvalue. The SVD can be used to obtain this information.

Suppose we know that A has an eigenvalue λ. Then the SVD of $(A-\lambda I)$ tells us that

$$(A - \lambda I)V = U\Sigma.$$

Therefore, if

$$(A - \lambda I)v_j = 0$$

for $j = (n-k+1), ..., n$, it follow that there are k eigenvectors corresponding to the eigenvalue λ, and the eigenvectors are $\{v_j\}$ for these same j. Furthermore, we can conclude that the Jordan canonical form of A has k blocks corresponding to the eigenvalue λ.

3.4 The SVD and Least Squares

Finally, the SVD can be used for solving linear least squares problems. Suppose that we are given $A_{m\times n}$, with rank$(A) = r$, and an m-vector $\mathbf{b}$. We want to find a vector $\mathbf{x}$ of minimal Euclidian length such that $\|\mathbf{b} - A\mathbf{x}\|_2$ is minimized.

Now,

$$\begin{aligned}\|\mathbf{b} - A\mathbf{x}\|_2 &= \|\mathbf{b} - U\Sigma V^T\mathbf{x}\|_2 \\ &= \|U^T\mathbf{b} - \Sigma\mathbf{y}\|_2.\end{aligned}$$

If we let

$$U^T\mathbf{b} = \begin{bmatrix} c_1 \\ \vdots \\ c_r \\ d_1 \\ \vdots \\ d_{m-r} \end{bmatrix} = \begin{bmatrix} \mathbf{c} \\ \mathbf{d} \end{bmatrix} \quad \text{and} \quad \Sigma = \begin{bmatrix} \sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_r & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix},$$

then

$$\|\mathbf{b} - A\mathbf{x}\|_2^2 = \sum_{i=1}^{r}(c_i - \sigma_i y_i)^2 + \sum_{i=r+1}^{n} d_i^2.$$

Thus, the residual is minimized when $y_i = \frac{c_i}{\sigma_i}$ for $i = 1, ..., r$. Furthermore, $\mathbf{x} = V\mathbf{y}$, so $\|\mathbf{x}\|_2 = \|\mathbf{y}\|_2$, from which it follows that $\|\mathbf{x}\|_2$ is minimized by setting the remaining $y_i = 0$ for $i = r+1, ..., n$. Therefore, the solution is given by:

$$\begin{aligned} \hat{\mathbf{x}} &= V \begin{bmatrix} \frac{1}{\sigma_1} & & & & & \\ & \ddots & & & & \\ & & \frac{1}{\sigma_r} & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_r \\ d_1 \\ \vdots \\ d_{m-r} \end{bmatrix} \\ &= V \begin{bmatrix} \frac{1}{\sigma_1} & & & & & \\ & \ddots & & & & \\ & & \frac{1}{\sigma_r} & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix} U^T\mathbf{b} \\ &\equiv A^+\mathbf{b}. \end{aligned}$$

where A^+ is known as the *pseudo-inverse* of A. Note that once we have computed the SVD of A, a sequence of least squares problems can be easily solved.

3.5 Lower Rank Approximations Revisited

In light of our previous discussion, we can see why one might want to replace a matrix with one of lower rank. Suppose that the matrix $\begin{bmatrix} 1 & \times \\ 0 & 10^{-10} \end{bmatrix}$ is obtained after a long series of computations. One is left to wonder if this matrix is actually a rank-2 matrix, or is better represented as a rank-1 matrix in light of potential round-off error. If one were to solve a system of equations using this matrix, and if the 10^{-10} term were in exact arithmetic 0, the computed solution would be very inaccurate. Instead of forming

$$A^{-1}\mathbf{b} = \begin{bmatrix} 1 & \times \\ 0 & 10^{10} \end{bmatrix} \mathbf{b},$$

it may be more appropriate to form

$$\tilde{A}^{+}\mathbf{b} = \begin{bmatrix} 1 & \times \\ 0 & 0 \end{bmatrix} \mathbf{b}.$$

It is important to note that while the eigenvalues of a matrix can be extremely sensitive to small perturbations, the singular values are well-conditioned, which justifies the use of the pseduo-inverse in such cases.

3.6 Computational Considerations for Eigenvalue Problems

We have many different computational considerations associated with eigenvalue problems. The methods we will develop in the remainder of the term will depend largely on the matrix characteristics, and the actual quantities of interest.

The following are some characteristics which will affect the algorithm we choose to solve our problem, and the software that is developed:

1. Matrices:
    - (a) Real Valued:
        - i. Symmetric
        - ii. Non-symmetric
            - A. Skew-symmetric
            - B. Hard (e.g., ill-conditioned)
        - iii. Sparse
            - A. Structured (e.g., banded)
            - B. Unstructured (e.g., Markov matrices)
        - iv. Dense
            - A. Structured (Toeplitz, Hankel)
            - B. Unstructured
    - (b) Complex Valued:
        - i. Hermitian ($A = A^*$)
        - ii. Complex Symmetric ($A = A^T$)

iii. Arbitrary Values

2. Items of interest:
   (a) All eigenvalues (and eigenvectors?)
   (b) Only the largest eigenvalue (and the corresponding eigenvector?)
   (c) Only the smallest eigenvalue (and the corresponding eigenvector?)
   (d) Eigenvalues such that $\text{Real}(\lambda_j) < 0$
   (e) Eigenvalues such that $|\lambda_j - \alpha| < \beta$
   (f) Methods for varying computer architectures

Some examples of more difficult problems are:

1. Find the eigenvalues of a sequence of problems $A_1, A_2, A_3, \ldots$ where
$$A_{i+1} = A_i + \mathbf{u}_i\mathbf{u}_i^T,$$
or
$$A_{i+1} = \begin{bmatrix} A_i & \times \\ \times & \times \end{bmatrix}.$$

2. Find $(\lambda, \mathbf{x})$ such that
$$(\lambda^2 A + \lambda B + C)\mathbf{x} = \mathbf{0}.$$

3.7 Problems Which Can Be Transformed Into Standard Eigenvalue Problems

1. Root Finding

   Suppose one wanted to find all the roots of the polynomial
$$\lambda^n + c_{n-1}\lambda^{n-1} + c_{n-2}\lambda^{n-2} + \cdots + c_0 = 0. \qquad (3.1)$$
   Consider the matrix
$$C = \begin{bmatrix} 0 & 1 & & & 0 \\ \vdots & 0 & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ 0 & 0 & \ldots & 0 & 1 \\ -c_0 & -c_1 & \ldots & -c_{n-2} & -c_{n-1} \end{bmatrix}$$
   and the eigenvalue problem
$$C\mathbf{y} = \lambda\mathbf{y}.$$

Expanding this system of equations, we obtain

$$\begin{aligned} y_2 &= \lambda y_1 \\ &\vdots \\ y_n &= \lambda y_{n-1} \\ -c_0 y_1 - c_1 y_2 - \cdots - c_{n-1} y_n &= \lambda y_n. \end{aligned}$$

However, the first $n-2$ equations imply

$$y_j = \lambda^{j-1} y_1 \qquad \text{for } j = 1, 2, \ldots, n.$$

Substituting these relations into the last equation yields

$$(\lambda^n + c_{n-1}\lambda^{n-1} + c_{n-2}\lambda^{n-2} + \cdots + c_0) y_1 = 0.$$

Hence, any eigenvalue of C is a root of this polynomial. Furthermore, one could construct the characteristic polynomial by expansion of the determinant and discover that the characteristic polynomial of C has the same roots as our polynomial (3.1). For this reason, C is called the *companion matrix* of this polynomial.

There are other matrices which express a polynomial as a linear combination of orthogonal polynomials. One such matrix is called the *comrade matrix* which may be discussed at a later time.

2. The Generalized Eigenvalue Problem

Suppose we wanted to solve the *generalized eigenvalue problem*: Find a vector $\mathbf{x}$ and a scalar λ such that

$$A\mathbf{x} = \lambda B\mathbf{x}.$$

Of course, if B is invertible, we could solve the standard eigenvalue problem $B^{-1}A\mathbf{x} = \lambda\mathbf{x}$. However, this strategy often destroys sparsity and symmetry, which is a *very* desirable property for eigenvalue problems, as we shall see later.

Suppose A is symmetric, and B is symmetric and positive definite. Then, we could write $B = FF^T$. This decomposition could be computed using the Cholesky Factorization, but this is certainly not the only possibility, since if G is the Cholesky factor of B, and if Q is an orthogonal matrix, then

$$(GQ)(GQ)^T = GQQ^TG^T = GG^T = B$$

is another possible factorization of this form. Then, we can manipulate our generalized eigenvalue problem as follows:

$$\begin{aligned} A\mathbf{x} &= \lambda B\mathbf{x} \\ &= \lambda F F^T \mathbf{x} \\ F^{-1}A\mathbf{x} &= \lambda F^T \mathbf{x} \\ F^{-1}AF^{-T}F^T\mathbf{x} &= \lambda F^T\mathbf{x} \\ C\mathbf{y} &= \lambda \mathbf{y} \end{aligned}$$

where $C = F^{-1}AF^{-T}$ and $\mathbf{y} = F^T\mathbf{x}$. Now, we have a standard symmetric eigenvalue problem, since the matrix C is symmetric. Furthermore, the original eigenvalues have not changed, and the original eigenvectors are easily obtained. Finally, we note that the fact that we can convert the original problem into an equivalent symmetric eigenvalue problem implies that the eigenvalues of the original problem are real.

3. Non-Standard Eigenvalue Problems

We previously alluded to another type of eigenvalue problem which may arise from a discussion of systems of ordinary differential equations. Suppose that we wish to find a vector $\mathbf{x}$ and a scalar λ such that

$$(\lambda^2 A + \lambda B + C)\mathbf{x} = \mathbf{0}. \tag{3.2}$$

We can convert this problem into a generalized eigenvalue problem by first expressing $\mathbf{y} = \lambda\mathbf{x}$. Then (3.2) becomes

$$\begin{aligned} \lambda A\mathbf{y} + B\mathbf{y} + C\mathbf{x} &= 0 \\ \mathbf{y} &= \lambda\mathbf{x}. \end{aligned}$$

Hence,

$$\begin{bmatrix} B & C \\ I & 0 \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \mathbf{x} \end{bmatrix} = \lambda \begin{bmatrix} -A & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \mathbf{x} \end{bmatrix}.$$

We can then compute the eigenvalues of this $2n \times 2n$ generalized eigenvalue problem. This technique is similar to writing a second order ODE as 2 first order ODE's. We can also generalize this technique to higher order polynomial problems.

Lecture 4

Methods For Computing The Largest Eigenvalue in Modulus

4.1 The Power Method

Let A be an arbitrary $n \times n$ matrix with n linearly independent eigenvectors, so that

$$A\mathbf{x}_i = \lambda_i \mathbf{x}_i$$

for $i = 1, 2, ..., n$ where

$$|\lambda_1| \geq |\lambda_2| \geq ... \geq |\lambda_n|.$$

Algorithm 4.1. (The power method)

```
Let $\mathbf{u}^{(0)}$ be an arbitrary vector with $\|\mathbf{u}^{(0)}\|_2 = 1$.
while unconverged do
    $\mathbf{v}^{(k+1)} = A\mathbf{u}^{(k)}$,
    $c_{k+1} = 1/\|\mathbf{v}^{(k+1)}\|_2$,
    $\mathbf{u}^{(k+1)} = c_{k+1}\mathbf{v}^{(k+1)}$.
end while
```

It is clear that

$$\mathbf{u}^{(k+1)} = c_{k+1} A\mathbf{u}^{(k)},$$

and therefore,

$$\begin{aligned}
\mathbf{u}^{(1)} &= c_1 A\mathbf{u}^{(0)} \\
\mathbf{u}^{(2)} &= c_2 A\mathbf{u}^{(1)} = c_2 c_1 A^2\mathbf{u}^{(0)} \\
&\vdots \\
\mathbf{u}^{(k)} &= c_k c_{k-1} ... c_1 A^k\mathbf{u}^{(0)} \\
&= d_k A^k \mathbf{u}^{(0)}
\end{aligned}$$

where $d_k = \prod_{j=1}^{k} c_j$. Since the eigenvectors are linearly independent, we may express any vector as a linear combination of the eigenvectors. So,

$$\mathbf{u}^{(0)} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i.$$

Then,

$$\begin{aligned}
\mathbf{u}^{(k)} &= d_k A^k \left(\sum_{i=1}^{n} \alpha_i \mathbf{x}_i \right) \\
&= d_k \sum_{i=1}^{n} \alpha_i A^k \mathbf{x}_i \\
&= d_k \sum_{i=1}^{n} \alpha_i \lambda_i^k \mathbf{x}_i
\end{aligned}$$

Furthermore, since $\|\mathbf{u}^{(k)}\|_2 = 1$,

$$d_k = \frac{1}{\|A^k \mathbf{u}^{(0)}\|_2}.$$

Assumption 4.2. Let us now assume the following:

1. $\lambda_1 = \lambda_2 = \cdots = \lambda_p$,
2. $|\lambda_1| > |\lambda_{i+p}|$ for $i = 1, ..., n-p$,
3. $\sum_{i=1}^{p} |\alpha_i| \neq 0$.

Then,

$$\begin{aligned}
\mathbf{u}^{(k)} &= \frac{\lambda_1^k \sum_{i=1}^{p} \alpha_i \mathbf{x}_i + \sum_{j=p+1}^{n} \alpha_j \lambda_j^k \mathbf{x}_j}{\|\lambda_1^k \sum_{i=1}^{p} \alpha_i \mathbf{x}_i + \sum_{j=p+1}^{n} \alpha_j \lambda_j^k \mathbf{x}_j\|_2} \\
&= \frac{\left(\frac{\lambda_1}{|\lambda_1|}\right)^k \sum_{i=1}^{p} \alpha_i \mathbf{x}_i + \sum_{j=p+1}^{n} \alpha_j \left(\frac{\lambda_j}{|\lambda_1|}\right)^k \mathbf{x}_j}{\left\| \sum_{i=1}^{p} \alpha_i \mathbf{x}_i + \sum_{j=p+1}^{n} \alpha_j \left(\frac{\lambda_j}{|\lambda_1|}\right)^k \mathbf{x}_j \right\|_2}.
\end{aligned}$$

Therefore,

$$\mathbf{u}^{(k)} = \gamma_k \sum_{i=1}^{p} \alpha_i \mathbf{x}_i + \mathcal{O}\left(\left|\frac{\lambda_{p+1}}{\lambda_1}\right|^k\right), \tag{4.1}$$

where $|\gamma_k|$ tends to a constant as $k \to \infty$. And, since every linear combination of eigenvectors corresponding to the same eigenvalue is an eigenvector, we obtain:

Theorem 4.3. (Convergence of the Power Method) *Under the assumptions (4.2), the sequence of vectors generated from the power method, $\mathbf{u}^{(k)}$, converges to an eigenvector as $k \to \infty$.*

If $|\lambda_1| = |\lambda_2|$, but $\lambda_1 \neq \lambda_2$, then the power method will not necessarily converge to an eigenvector. For example, consider the matrix

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

For this matrix, $\lambda_1 = i$ and $\lambda_2 = -i$. If $\mathbf{u}^{(0)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, then $\mathbf{u}^{(k)}$ will not converge to an eigenvector. Later we shall consider in detail the problem of computing eigenvalues of equal moduli.

4.2 Shifts

From (4.1) we see that $\mathbf{u}^{(k)}$ will converge slowly to an eigenvector if $\frac{|\lambda_{p+1}|}{|\lambda_1|}$ is close to 1. One possible remedy is to use the power method on the matrix $(A - \alpha I)$. We refer to α as a *shift*. Recall that the eigenvalues of $(A - \alpha I)$ are

$$\lambda_1 - \alpha, \lambda_2 - \alpha, ..., \lambda_n - \alpha,$$

(i.e., shifted versions of the eigenvalues of A) and hence, the power method applied to $(A - \alpha I)$ yields a sequence of vectors

$$\mathbf{u}^{(k)} = \gamma_k \left(\sum_{i=1}^{p} \alpha_i \mathbf{x}_i + \sum_{j=p+1}^{n} \alpha_j \left(\frac{\lambda_j - \alpha}{|\lambda_1 - \alpha|} \right)^k \mathbf{x}_j \right).$$

This may look like the same statement as before, but it should not be interpreted as such, since some of the eigenvalue ratios in the second sum may now be greater than 1 in modulus. For example, if all the eigenvalues are real and positive, then if $\alpha > \lambda_1$,

$$0 > \lambda_1 - \alpha \geq \lambda_2 - \alpha \geq ... \geq \lambda_n - \alpha.$$

So, $\lambda_n - \alpha$ is the eigenvalue of largest magnitude, and so the power method can be used to find λ_n (in this case) if a shift is employed. In general, the power method will converge to the eigenvalue farthest from the shift if another distinct eigenvalue does not exist that is the same distance from the shift.

If we wish for the power method (with shift) to converge to an eigenvector that corresponds to λ_1, the best choice for α is the one for which

$$\max_{p+1 \leq i \leq n} \left| \frac{\lambda_i - \alpha}{\lambda_1 - \alpha} \right| = \min .$$

Since this clearly requires some knowledge of $\{\lambda_i\}_{i=p+1}^n$, a heuristic procedure is generally used for determining α.

4.3 Rayleigh Quotient

If the matrix A is symmetric, then the *Rayleigh Quotient*, defined by

$$\mu_k = \frac{\mathbf{u}^{(k)T} A \mathbf{u}^{(k)}}{\mathbf{u}^{(k)T} \mathbf{u}^{(k)}},$$

provides an improved estimate of the eigenvalue, λ_1. Recall that for a real symmetric matrix, the eigenvalues are real and the eigenvectors are orthonormal. Since

$$\mathbf{u}^{(k)} = \gamma_k \left(\sum_{i=1}^{p} \alpha_i \mathbf{x}_i + \sum_{j=p+1}^{n} \alpha_j \left(\frac{\lambda_j}{|\lambda_1|} \right)^k \mathbf{x}_j \right)$$

and

$$A\mathbf{u}^{(k)} = \gamma_k \left(\lambda_1 \sum_{i=1}^{p} \alpha_i \mathbf{x}_i + \sum_{j=p+1}^{n} \alpha_j \lambda_j \left(\frac{\lambda_j}{|\lambda_1|} \right)^k \mathbf{x}_j \right) .$$

It follows that

$$\begin{aligned}
\mu_k &= \frac{\mathbf{u}^{(k)T} A \mathbf{u}^{(k)}}{\mathbf{u}^{(k)T} \mathbf{u}^{(k)}} \\
&= \frac{\gamma_k^2 \left(\sum_{i=1}^{p} \alpha_i \mathbf{x}_i^T + \sum_{j=p+1}^{n} \alpha_j \left(\frac{\lambda_j}{|\lambda_1|} \right)^k \mathbf{x}_j^T \right) \left(\lambda_1 \sum_{i=1}^{p} \alpha_i \mathbf{x}_i + \sum_{j=p+1}^{n} \alpha_j \lambda_j \left(\frac{\lambda_j}{|\lambda_1|} \right)^k \mathbf{x}_j \right)}{\gamma_k^2 \left(\sum_{i=1}^{p} \alpha_i \mathbf{x}_i^T + \sum_{j=p+1}^{n} \alpha_j \left(\frac{\lambda_j}{|\lambda_1|} \right)^k \mathbf{x}_j^T \right) \left(\sum_{i=1}^{p} \alpha_i \mathbf{x}_i + \sum_{j=p+1}^{n} \alpha_j \left(\frac{\lambda_j}{|\lambda_1|} \right)^k \mathbf{x}_j \right)} \\
&= \frac{\lambda_1 \sum_{i=1}^{p} \alpha_i^2 + \sum_{j=p+1}^{n} \alpha_j^2 \lambda_j \left(\frac{\lambda_j}{|\lambda_1|} \right)^{2k}}{\sum_{i=1}^{p} \alpha_i^2 + \sum_{j=p+1}^{n} \alpha_j^2 \left| \frac{\lambda_j}{\lambda_1} \right|^{2k}} \\
&= \lambda_1 \left(1 + \mathcal{O} \left(\left| \frac{\lambda_{p+1}}{\lambda_1} \right|^{2k} \right) \right) .
\end{aligned}$$

Thus, the Rayleigh Quotient μ_k has roughly twice as many digits of accuracy as the components of $\mathbf{u}^{(k)}$.

4.4 Computing the Largest r Eigenvalues

Suppose we wanted the largest r eigenvalues. This could easily be acomplished using the power method. First, we can compute the largest eigenvalue λ_1 and its corresponding eigenvector $\mathbf{x}_1$ accurately. Then, we will perform the power method again, but this time forcing $\alpha_1 = 0$ by either choosing an initial guess that is orthogonal to $\mathbf{x}_1$, or by subtracting off the contribution to $\mathbf{v}^{(k)}$ of $\mathbf{x}_1$ at each iteration by defining

$$\underline{\mathrm{v}}^{(k)} = (I - \mathbf{x}_1\mathbf{x}_1^T)\mathbf{v}^{(k)},$$

and then let $\mathbf{u}^{(k+1)}$ be a normalized $\underline{\mathrm{v}}^{(k)}$. We could also work with the matrix

$$A^{(2)} = A - \lambda_1\mathbf{x}_1\mathbf{x}_1^T.$$

Once the second eigenvalue and eigenvector are computed, we can repeat the process, orthogonalizing against all previously computed eigenvectors until we have constructed r eigenvectors.

4.5 Distinct Eigenvalues of Equal Modulus

As we pointed out earlier, if $|\lambda_1| = |\lambda_2|$ and $\lambda_1 \neq \lambda_2$, then the power method will not converge for an arbitrary $\mathbf{u}^{(0)}$. One method of overcoming this difficulty is to employ an appropriately chosen shift. But, this shift might be complex. Also, the best shift can be very hard to determine. Fortunately, there are approaches which are more efficient which allow real arithmetic.

Let us assume that for the real matrix (no longer symmetric), A,

$$\begin{aligned}\lambda_1 &= \rho e^{i\theta}\\ \lambda_2 &= \rho e^{-i\theta}\end{aligned}$$

and $|\lambda_i| < |\lambda_1|$ for $i = 3, 4, ..., n$. Furthermore, if we wish to use real arithmetic in the numerical process, we should take $\mathbf{u}^{(0)}$ as a real vector. Suppose

$$\mathbf{u}^{(0)} = \alpha_1\mathbf{x}_1 + \alpha_2\mathbf{x}_2 + \sum_{i=3}^{n}\alpha_i\mathbf{x}_i.$$

Then,

$$\begin{aligned}\mathbf{u}^{(k)} &= d_k A^k\mathbf{u}^{(0)}\\ &= d_k(\alpha_1\lambda_1^k\mathbf{x}_1 + \alpha_2\lambda_2^k\mathbf{x}_2 + \sum_{i=3}^{n}\alpha_i\lambda_i^k\mathbf{x}_i)\\ &= d_k\left(\alpha_1\rho^k e^{ik\theta}\mathbf{x}_1 + \alpha_2\rho^k e^{-ik\theta}\mathbf{x}_2 + \sum_{i=3}^{n}\alpha_i\lambda_i^k\mathbf{x}_i\right),\end{aligned}$$

so that if $\theta \neq 0$, the elements of $\mathbf{u}^{(k)}$ will tend to oscillate.

Let us assume temporarily that $\alpha_i = 0$ for $i = 3, 4, ..., n$. Then λ_1 and λ_2 are the roots of the equation:

$$\lambda^2 + p\lambda + q = 0$$

for some p and q. Now define,

$$\mathbf{v}^{(k)} = A\mathbf{u}^{(k)}, \qquad \text{and} \qquad \mathbf{w}^{(k)} = A\mathbf{v}^{(k)}.$$

Then, since

$$\mathbf{u}^{(k)} = d_k(\alpha_1\lambda_1^k\mathbf{x}_1 + \alpha_2\lambda_2^k\mathbf{x}_2),$$

it follows that

$$\begin{aligned}\mathbf{w}^{(k)} + p\mathbf{v}^{(k)} + q\mathbf{u}^{(k)} &= d_k(\alpha_1(\lambda_1^2 + p\lambda_1 + q)\lambda^k\mathbf{x}_1 \\ &\qquad + \alpha_2(\lambda_2^2 + p\lambda_2 + q)\lambda^k\mathbf{x}_2) \\ &= 0.\end{aligned}$$

Clearly, it is not the case that $\alpha_j = 0$ for $j = 3, 4, ..., n$ in general, but one would expect that for a sufficiently large k, the vector generated from the power method, $\mathbf{u}^{(k)}$, would lie predominantly in a 2-dimensional subspace (i.e., α_j would be small for $j = 3, 4, ..., n$). Accordingly, we wish to determine $\{p_k, q_k\}$ so that

$$\mathbf{w}^{(k)} + p_k\mathbf{v}^{(k)} + q_k\mathbf{u}^{(k)} = \varepsilon^{(k)}$$

and

$$\|\varepsilon^{(k)}\|_2 = \min.$$

Thus, we will solve the linear least squares problem whose normal equations are given by:

$$\begin{bmatrix} \mathbf{v}^{(k)T}\mathbf{v}^{(k)} & \mathbf{v}^{(k)T}\mathbf{u}^{(k)} \\ \mathbf{v}^{(k)T}\mathbf{u}^{(k)} & \mathbf{u}^{(k)T}\mathbf{u}^{(k)} \end{bmatrix} \begin{bmatrix} p_k \\ q_k \end{bmatrix} = -\begin{bmatrix} \mathbf{v}^{(k)T}\mathbf{w}^{(k)} \\ \mathbf{u}^{(k)T}\mathbf{w}^{(k)} \end{bmatrix} \tag{4.2}$$

(i.e., $W^{(k)}\mathbf{c}^{(k)} = -\mathbf{h}^{(k)}$). It can easily be shown that if $\mathbf{u}^{(k)}$ belongs to a 1-dimensional subspace associated with the eigenvectors of A (i.e., α_1 or α_2 are zero), then $\det(W^{(k)}) = 0$.

So, we can determine λ_1 and λ_2 as follows:

Algorithm 4.4. (Modified Power Method)

```
Let $\varepsilon, \eta > 0$ be prescribed tolerances, and perform several $(k)$
        steps of the power method. Then define $W^{(k)}$ as in (4.2).
**if** $\det(W^{(k)}) < \varepsilon$ **then**
    Continue with the power method.
**else**
```

```
        Compute $\mathbf{c}^{(k)}$.
        **if** $\|\mathbf{c}^{(k+1)} - \mathbf{c}^{(k)}\| \le \eta$ **then**
            Determine $\lambda_1$ and $\lambda_2$ from the roots of $\lambda^2 + p_k\lambda + q_k = 0$.
        **else**
            Repeat after another step of the power method.
        **end if**
**end if**
```

Note that complex arithmetic is not required in the calculation. Only after the vector $\mathbf{c}^{(k)}$ has converged are the (complex) eigenvalues computed.

The eigenvectors can be calculated in a simple manner once the eigenvalues are known. If

$$\mathbf{u} = \alpha_1\mathbf{x}_1 + \alpha_2\mathbf{x}_2,$$

then,

$$\mathbf{v} = \alpha_1\lambda_1\mathbf{x}_1 + \alpha_2\lambda_2\mathbf{x}_2$$

since $\mathbf{v} = A\mathbf{u}$. Now,

$$\begin{aligned} \mathbf{v} - \lambda_2\mathbf{u} &= \alpha_1(\lambda_1 - \lambda_2)\mathbf{x}_1 \\ \mathbf{v} - \lambda_1\mathbf{u} &= \alpha_2(\lambda_2 - \lambda_1)\mathbf{x}_2, \end{aligned}$$

and so,

$$\begin{aligned} \mathbf{x}_1 &= \frac{\mathbf{v} - \lambda_2\mathbf{u}}{\|\mathbf{v} - \lambda_2\mathbf{u}\|_2} \\ \mathbf{x}_2 &= \frac{\mathbf{v} - \lambda_1\mathbf{u}}{\|\mathbf{v} - \lambda_1\mathbf{u}\|_2} \end{aligned}$$

4.6 A Very Bad Method

One could attempt to find all n eigenvalues of $A_{n\times n}$at once in a manner similar to the method used in the previous section, but it is not recommended.

The Cayley-Hamilton Theorem tells us that if

$$\phi(\lambda) = \lambda^n - \alpha_1\lambda^{n-1} - ... - \alpha_n = 0,$$

then

$$\phi(A) = A^n - \alpha_1 A^{n-1} - ... - \alpha_n I = 0.$$

As an aside, we can note that

$$\begin{aligned}
A^n &= \alpha_1 A^{n-1} + ... + \alpha_n I \\
&\in \mathcal{P}_{n-1}(A) \\
\Rightarrow A^{n+1} &= AA^n = A(\alpha_1 A^{n-1} + ... + \alpha_n I) \\
&= \alpha_1 A^n + ...\alpha_n A \\
&= \alpha_1^2 A^{n-1} + \alpha_2(1+\alpha_1)A^{n-2} + ... + \alpha_n(1+\alpha_1)I \\
&\in \mathcal{P}_{n-1}(A) \\
&\vdots \\
\Rightarrow A^m &\in \mathcal{P}_{n-1}(A). \\
\text{Also, } \alpha_n I &= A^n - \alpha_1 A^{n-1} - ... - \alpha_{n-1} A \\
\Rightarrow \alpha_n A^{-1} &= A^{n-1} - \alpha_1 A^{n-2} - ... - \alpha_{n-1} I \\
\text{i.e., } A^{-1} &\in \mathcal{P}_{n-1}(A).
\end{aligned}$$

Returning to our bad algorithm,

$$\phi(A)\mathbf{u}^{(0)} = A^n\mathbf{u}^{(0)} - \alpha_1 A^{n-1}\mathbf{u}^{(0)} - ... - \alpha_n \mathbf{u}^{(0)}.$$

So, if we store the *Krylov Sequence* $\mathbf{v}^{(0)} = \mathbf{u}^{(0)}$, $\mathbf{v}^{(i)} = A\mathbf{v}^{(i-1)}$, for $i = 1, ..., n$, we can retrieve the characteristic polynomial by solving the system of equations:

$$[\mathbf{v}^{(n-1)}, ..., \mathbf{v}^{(0)}] \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \mathbf{v}^{(n)}.$$

Then we can determine the eigenvalues from those of the Companion Matrix of $\phi(\lambda)$.

Not only does this method only transform our problem into one with a well defined structure, but the method generates a system of equations where the left-most columns of the matrix are all strongly dominated by the same eigenvector of A, so the system is highly ill-conditioned. Furthermore, we have performed n, $\mathcal{O}\left(n^2\right)$ operations (matrix-vector multiplies), followed by a $\mathcal{O}\left(\frac{1}{3}n^3\right)$ linear systems solve, yielding another eigenvalue problem. Even if it wasn't ill conditioned, it makes for a computationally expensive algorithm.

4.7 The Inverse Power Method

Algorithm 4.5. (Inverse Power Method)

```
Let u be an arbitrary vector with $\|u\|_2 = 1$, and $\mu_0$ be an arbitrary (real) scalar.
while $\mu_k$ is unconverged do
    Solve for $\mathbf{w}^{(k+1)}$: $(A - \mu_k I)\mathbf{w}^{(k+1)} = \mathbf{u}$
    Update: $\mu_k = \frac{\mathbf{w}^{(k+1)T} A \mathbf{w}^{(k+1)}}{\mathbf{w}^{(k+1)T} \mathbf{w}^{(k+1)}}$
end while
```

We can make several variations in this procedure. For instance, our base algorithm can be replaced by:

$$(A - \mu_k I)\mathbf{w}^{(k+1)} = \frac{\mathbf{w}^{(k)}}{\|\mathbf{w}^{(k)}\|_2}.$$

Here we would obtain cubic convergence. Also, it is often desirable to fix $\mu_k = \mu$. In this manner, the factorization of $(A - \mu_k I)$ will not need to be recomputed at each iteration, possibly saving computational work.

Lecture 5

Finding Other Eigenvalues and Perturbation Theory

5.1 The Inverse Power Method

A very effective method for increasing the rate of convergence of the power method is the method of inverse iteration, also known as the *inverse power method*. This method can be used to find *any* eigenvalue, provided an appropriate shift is used. We shall discuss this algorithm for real, symmetric matrices, but it is applicable to more general situations.

Algorithm 5.1. (Inverse Power Method)

```
Let $\mathbf{u}$ be an arbitrary vector with $\|u\|_2 = 1$, and $\mu_0$ be an arbitrary (real) scalar.
while $\mu_k$ is unconverged do
    Solve for $\mathbf{w}^{(k+1)}$: $(A - \mu_k I)\mathbf{w}^{(k+1)} = \mathbf{u}$
    Update: $\mu_k = \frac{\mathbf{w}^{(k+1)T} A \mathbf{w}^{(k+1)}}{\mathbf{w}^{(k+1)T}\mathbf{w}^{(k+1)}}$
end while
```

Again, let us write

$$\mathbf{u} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i, \qquad \text{where } \sum_{i=1}^{n} \alpha_i^2 = 1,$$

and $\{\mathbf{x}_k\}$ are the eigenvectors of A. Let us further assume that $\lambda_1 = \lambda_2 = ... = \lambda_p$ and that u has some contibution in the direction of an eigenvector that corresponds to λ_1 (i.e., $\sum_{i=1}^{p} |\alpha_i| > 0$).

If μ_0 is not an eigenvalue of A, then,

$$\begin{aligned}\mathbf{w}^{(1)} &= (A - \mu_0 I)^{-1}\mathbf{u} \\ &= \sum_{i=1}^{n} \frac{\alpha_i}{\lambda_i - \mu_0}\mathbf{x}_i\end{aligned}$$

Hence, if μ_0 is a good approximation to λ_1, we would expect $\mathbf{w}^{(1)}$ to be a good approximation to an eigenvector that corresponds to λ_1. In general,

$$\mathbf{w}^{(k)} = \sum_{i=1}^{n} \frac{\alpha_i}{\lambda_i - \mu_{k-1}} \mathbf{x}_i,$$

and hence,

$$A\mathbf{w}^{(k)} = \sum_{i=1}^{n} \frac{\alpha_i \lambda_i}{\lambda_i - \mu_{k-1}} \mathbf{x}_i.$$

And by orthogonality of the eigenvectors ($A = A^T$),

$$\begin{aligned}\mu_{k+1} &= \frac{\mathbf{w}^{(k+1)T} A \mathbf{w}^{(k+1)}}{\mathbf{w}^{(k+1)T} \mathbf{w}^{(k+1)}} \\ &= \frac{\sum_{i=1}^{n} \frac{\alpha_i^2 \lambda_i}{(\lambda_i - \mu_k)^2}}{\sum_{j=1}^{n} \frac{\alpha_j^2}{(\lambda_j - \mu_k)^2}}.\end{aligned}$$

Hence,

$$\begin{aligned}\mu_{k+1} - \lambda_1 &= \frac{\sum_{i=1}^{n} \alpha_i^2 \frac{\lambda_i - \lambda_1}{(\lambda_i - \mu_k)^2}}{\sum_{j=1}^{n} \frac{\alpha_j^2}{(\lambda_j - \mu_k)^2}} \\ &= \frac{\sum_{i=p+1}^{n} \alpha_i^2 \frac{\lambda_i - \lambda_1}{(\lambda_i - \mu_k)^2}}{\sum_{j=1}^{n} \frac{\alpha_j^2}{(\lambda_j - \mu_k)^2}} \quad \text{if } \lambda_1 = \lambda_2 = \ldots = \lambda_p\ , \\ &= (\lambda_1 - \mu_k)^2 \left\{ \frac{\sum_{i=p+1}^{n} \alpha_i^2 \frac{\lambda_i - \lambda_1}{(\lambda_i - \mu_k)^2}}{\sum_{j=1}^{p} \alpha_j^2 + \sum_{j=p+1}^{n} \alpha_j^2 \frac{(\lambda_1 - \mu_k)^2}{(\lambda_j - \mu_k)^2}} \right\}.\end{aligned}$$

Thus,

$$\frac{\mu_{k+1} - \lambda_1}{(\mu_k - \lambda_1)^2} = \frac{\sum_{i=p+1}^{n} \alpha_i^2 \frac{\lambda_i - \lambda_1}{(\lambda_i - \mu_k)^2}}{\sum_{j=1}^{p} \alpha_j^2 + \sum_{j=p+1}^{n} \alpha_j^2 \frac{(\lambda_1 - \mu_k)^2}{(\lambda_j - \mu_k)^2}}.$$

Therefore, if $\mu_k \to \lambda_1$ as $k \to \infty$,

$$\lim_{k\to\infty} \frac{u_{k+1} - \lambda_1}{(\mu_k - \lambda_1)^2} = \frac{\sum_{i=p+1}^{m} \frac{\alpha_i^2}{\lambda_i - \lambda_1}}{\sum_{j=1}^{p} \alpha_j^2}.$$

Thus, if the inverse power method converges, the rate of convergence is quadratic. It can be shown that in most situations, μ_k *will* converge to an eigenvalue.

It was mentioned earlier that if μ_0 is a good approximation to λ_1, we would expect $\mathbf{w}^{(1)}$ to be a good approximation to an eigenvector that corresponds to λ_1. However, if μ_0 is a good approximation to λ_j, then we would expect that $\mathbf{w}^{(1)}$ would be a good approximation to an eigenvector that corresponds to λ_j. We could proceed as before with this goal in mind, and show that in this case, if μ_k converges to λ_j, it would do so quadratically.

5.2 Note: Linear Systems Error Bounds

If we wish to solve $A\mathbf{x} = \mathbf{b}$ for the vector $\mathbf{x}$, and we actually obtain from our algorithm the vector ξ, then if we express $\mathbf{x} = \xi + \mathbf{e}$ for some vector $\mathbf{e}$ which represents computational errors, then we can deine the *residual*, $\mathbf{r}$,

$$\begin{aligned} \mathbf{r} &\equiv \mathbf{b} - A\xi \\ &= A(\mathbf{x} - \xi) \\ &= A\mathbf{e} \\ \Rightarrow \qquad \mathbf{e} &= A^{-1}\mathbf{r}. \end{aligned}$$

So, we can place the following bounds on the error vector:

$$\begin{aligned} \|\mathbf{r}\|_2 &\le \|A\|_2 \|\mathbf{e}\|_2 \\ \Rightarrow \qquad \frac{\|\mathbf{r}\|_2}{\|A\|_2} &\le \|\mathbf{e}\|_2 \le \|A^{-1}\|_2 \|\mathbf{r}\|_2. \end{aligned}$$

Notice that these bounds are directly related to the matrix, A and its inverse. It is rather surprising to note that the bounds that we will develop for eigenvalue and eigenvector perturbation are not of this form.

5.3 Error Bounds on Eigenvalues

Let us suppose A is symmetric (i.e., $A = A^T$). Then given an estimate of an eigenvector, $\mathbf{y}$, we can use the Rayleigh Quotient as an estimate of the eigenvalue, $\alpha = \frac{\mathbf{y}^T A\mathbf{y}}{\mathbf{y}^T\mathbf{y}}$. We know that

$$\lambda_{min}(A) \le \alpha \le \lambda_{max}(A).$$

Let us define the *residual* with respect to the eigenvalue problem, $\mathbf{r}$, as

$$\mathbf{r} = A\mathbf{y} - \alpha\mathbf{y} = (A - \alpha I)\mathbf{y}.$$

Then,

$$\begin{aligned} \frac{\mathbf{r}^T\mathbf{r}}{\mathbf{y}^T\mathbf{y}} &= \frac{\mathbf{y}^T(A - \alpha I)^T(A - \alpha I)\mathbf{y}}{\mathbf{y}^T\mathbf{y}} \\ &= \frac{\mathbf{y}^T(A_\alpha I)^2\mathbf{y}}{\mathbf{y}^T\mathbf{y}} \\ &\le \max\{(A - \alpha I)^2\}. \end{aligned}$$

So,

$$\min_i |\lambda_i - \alpha|^2 \le \frac{\mathbf{r}^T\mathbf{r}}{\mathbf{y}^T\mathbf{y}} = \frac{\|\mathbf{r}\|_2^2}{\|\mathbf{y}\|_2^2} \le \max_i |\lambda_i - \alpha|^2.$$

Hence, there must be at least one eigenvalue inside of the interval surrounding our estimate, $[\alpha - \frac{\|\mathbf{r}\|_2}{\|\mathbf{y}\|_2}, \alpha + \frac{\|\mathbf{r}\|_2}{\|\mathbf{y}\|_2}]$. One should notice that this bound is much easier to compute than those for solving linear systems of equations.

5.4 Solutions to Nearby Eigenvalue Problems

Suppose we have found an eigenpair, $(\lambda_i, \mathbf{x}_i)$, of the symmetric matrix, A. How close is λ_i to an eigenvalue of the nearby matrix $(A + \varepsilon E)$ where E is symmetric, $\|E\|_2 = 1$, and $\varepsilon > 0$ is a small parameter? We have

$$\begin{aligned} \mathbf{r} &= (A + \varepsilon E)\mathbf{x}_i - \lambda_i \mathbf{x}_i \\ &= \varepsilon E \mathbf{x}_i \end{aligned}$$

which yields

$$\|\mathbf{r}\|_2 \leq \varepsilon.$$

Therefore, there must be an eigenvalue of $(A + \varepsilon E)$ inside the interval $[\lambda_i - \varepsilon, \lambda_i + \varepsilon]$.

5.5 The Gerschgorin Disk Theroem

Given $A\mathbf{x} = \lambda\mathbf{x}$, looking at one row of this system of equations,

$$\begin{aligned} a_{rr}x_r + \sum_{s \neq r} a_{rs}x_s &= \lambda x_r \\ (a_{rr} - \lambda)x_r &= -\sum_{s \neq r} a_{rs}x_s \end{aligned}$$

Let us assume that $|x_r| \geq |x_s|$ for $s = 1, 2, ..., n$. Then,

$$\begin{aligned} (a_{rr} - \lambda) &= -\sum_{s \neq r} a_{rs}\frac{x_s}{x_r} \\ |a_{rr} - \lambda| &\leq \sum_{s \neq r} |a_{rs}| \equiv \rho_r \end{aligned}$$

Hence, the union of all disks, $|a_{ii} - \mu| \leq \rho_i$, contain all the eigenvalues of A.

For example, let

$$A_{n \times n} = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}.$$

Then $\rho_1 = \rho_n = 1$, and $\rho_j = 2$ for $j = 2, 3, ..., (n-1)$. The Gerschgorin Disk Theorem tells us that all the eigenvalues lie in the disk, $|\lambda - 2| \leq 2$. Furthermore, since $A = A^T$, $0 \leq \lambda \leq 4$. We could also conclude that the matrix $A + \sigma I$ is positive definite for any $\sigma > 0$.

Another trick that we can use is to rescale off disgonal elements using the similarity trasformation $B = DAD^{-1}$ where D is a diagonal matrix. Then the elements of B are given by $b_{ij} = a_{ij}\frac{d_i}{d_j}$. Then the Gerschgorin Disk Theorem tells us:

$$|a_{ii} - \lambda| = |b_{ii} - \lambda| \leq \sum_{i \neq j} |a_{ij}| \left| \frac{d_i}{d_j} \right| .$$

To make this effect more clear, suppose $A = \begin{bmatrix} 1 & 1000 \\ 10^{-9} & 1 \end{bmatrix}$. Then we know all of the eigenvalues lie on a disk $|\lambda - 1| \leq 1000$. But if we choose D such that $d_1 = 1$ and $d_2 = 10^6$, then

$$B = DAD^{-1} = \begin{bmatrix} 1 & 1000\frac{d_1}{d_2} \\ 10^{-9}\frac{d_2}{d_1} & 1 \end{bmatrix} .$$

We can conclude that all of the eigenvalues of A lie in the disk $|\lambda - 1| \leq 10^{-3}$. Since B is symmetric, we can also conclude that the eigenvalues are real and lie in the interval $[1 - 10^{-3}, 1 + 10^{-3}]$.

One can also prove as a corollary to the Gerschgorin Disk Theorem that if any of the disks are disjoint, then an eigenvalue must lie in that disk.

Lecture 6

More Perturbation Theory and Error Bounds, and Jacobi's Algorithm

6.1 The Condition Number of an Eigenvalue

If $A\mathbf{x} = \lambda\mathbf{x}$, we call $\mathbf{x}$ an eigenvector, or more precisely, a *right eigenvector* of A. Similarly, if $\mathbf{y}^*A = \lambda\mathbf{y}^*$, we call $\mathbf{y}$ a *left eigenvector* of A. Note that any left eigenvector of A is a right eigenvector of A^*. If the A is diagonalizable, then we can write

$$\mathbf{y}_j^*\mathbf{x}_i = 0 \qquad \text{for } i \neq j.$$

Consider the matrix $(A+\varepsilon B)$ with right eigenvector $\mathbf{x}_1(\varepsilon)$, and eigenvalue $\lambda_1(\varepsilon)$. Assume further that $|b_{ij}| \leq 1$. Then it can be shown that since A is diagonalizable, $\lambda_1(\varepsilon)$ can be expressed as a power series in ε such that

$$\lambda_1(\varepsilon) = \lambda_1 + k_1\varepsilon + k_2\varepsilon^2 + \cdots .$$

So,

$$\frac{\lambda_1(\varepsilon) - \lambda_1}{\varepsilon} \to k_1, \qquad \text{as } \varepsilon \to 0.$$

Similarly, $\mathbf{x}_1(\varepsilon)$ can be expressed as a power series in ε such that

$$\mathbf{x}_1(\varepsilon) = \mathbf{x}_1 + \varepsilon\mathbf{z}_1 + \varepsilon^2\mathbf{z}_2 + \cdots .$$

Since A is diagonalizable, we can express the vectors $\mathbf{z}_i$ as linear combinations of the eigenvectors of A, i.e. $\mathbf{z}_i = \sum_{j=1}^n t_{ji}\mathbf{x}_j$. So,

$$\begin{aligned}\mathbf{x}_1(\varepsilon) &= \mathbf{x}_1 + \varepsilon\sum_{j=1}^n t_{j1}\mathbf{x}_j + \varepsilon^2\sum_{j=1}^n t_{j2}\mathbf{x}_j + \cdots \\ &= (1 + \varepsilon t_{11} + \varepsilon^2 t_{12} + \cdots)\mathbf{x}_1 + (\varepsilon t_{21} + \varepsilon^2 t_{22} + \cdots)\mathbf{x}_2 \\ &\quad + \cdots + (\varepsilon t_{n1} + \varepsilon^2 t_{n2} + \cdots)\mathbf{x}_n.\end{aligned}$$

Since this is an eigenvector, let us rescale it so that the coefficient of $\mathbf{x}_1$ is 1, then

$$\begin{aligned}\mathbf{x}_1(\varepsilon) &= \mathbf{x}_1 + \left(\sum_{j=1}^{\infty} \varepsilon^j \tilde{t}_{2j}\right) \mathbf{x}_2 + \left(\sum_{j=1}^{\infty} \varepsilon^j \tilde{t}_{3j}\right) \mathbf{x}_3 + \cdots + \left(\sum_{j=1}^{\infty} \varepsilon^j \tilde{t}_{nj}\right) \mathbf{x}_n \\ &\equiv \mathbf{x}_1 + \sum_{j=2}^{n} f_j(\varepsilon)\mathbf{x}_j.\end{aligned}$$

Before we proceed, let us make two notational definitions and observations. Let $s(\lambda_i) \equiv |\mathbf{y}_i^* \mathbf{x}_i|$, and also, $\beta_{ij} \equiv \mathbf{y}_i^* B \mathbf{x}_j$. Thus, $|s_i| \leq \|\mathbf{y}_i\|_2 \|\mathbf{x}_i\|_2 = 1$, and $|\beta_{ij}| \leq \sigma_1(B) = \|B\|_2$.

Now,

$$(A + \varepsilon B)\mathbf{x}_1(\varepsilon) = \lambda_1(\varepsilon)\mathbf{x}_1(\varepsilon).$$

Looking at the terms that do not include ε, we obtain the equation,

$$A\mathbf{x}_1 = \lambda_1 \mathbf{x}_1.$$

The terms of order ε yield the equation:

$$\begin{aligned}&\varepsilon A(\tilde{t}_{21}\mathbf{x}_2 + \tilde{t}_{31}\mathbf{x}_3 + \cdots + \tilde{t}_{n1}\mathbf{x}_n) + \varepsilon B\mathbf{x}_1 \\ &\qquad = \varepsilon(k_1\mathbf{x}_1 + \lambda_1\tilde{t}_{21}\mathbf{x}_2 + \lambda_1\tilde{t}_{31}\mathbf{x}_3 + \cdots + \lambda_1\tilde{t}_{n1}\mathbf{x}_n).\end{aligned}$$

That is,

$$\begin{aligned}A\left(\sum_{i=2}^{n} \tilde{t}_{i1}\mathbf{x}_i\right) + B\mathbf{x}_1 &= \lambda_1\left(\sum_{i=2}^{n} \tilde{t}_{i1}\mathbf{x}_i\right) + k_1\mathbf{x}_1 \\ (A - \lambda_1 I)\left(\sum_{i=2}^{n} \tilde{t}_{i1}\mathbf{x}_i\right) + B\mathbf{x}_1 &= k_1\mathbf{x}_1 \\ \sum_{i=2}^{n} \tilde{t}_{i1}(\lambda_i - \lambda_1)\mathbf{x}_i + B\mathbf{x}_1 &= k_1\mathbf{x}_1.\end{aligned}$$

So, due to the orthogonality of the right and left eigeivectors, multiplying this equation on the left by $\mathbf{y}_1^*$ yields

$$\mathbf{y}_1^* B\mathbf{x}_1 = k_1 \mathbf{y}_1^* \mathbf{x}_1$$

and therefore

$$|k_1| = \left|\frac{\mathbf{y}_1^* B\mathbf{x}_1}{\mathbf{y}_1^*\mathbf{x}_1}\right| = \frac{|\beta_{11}|}{s(\lambda_1)}.$$

Hence,

$$|\lambda_1(\varepsilon) - \lambda_1| = \frac{|\beta_{11}|}{s(\lambda_1)}\varepsilon + \mathcal{O}\left(\varepsilon^2\right).$$

That is to say, the perturbation in the eigenvalue is proportional to $s(\lambda_i)^{-1}$ for each i. Thus we call $s(\lambda_i)^{-1}$ the *condition number of* λ_i. Furthermore,

$$\left|\frac{\lambda_i(\varepsilon) - \lambda_i}{\varepsilon}\right| \to \frac{\|B\|_2}{s(\lambda_i)}$$

as $\varepsilon \to 0$.

6.2 Perturbation of Eigenvectors

To find the first order perturbation of the eigenvectors, we proceed as before, arriving at the equation

$$\sum_{i=2}^{n} \tilde{t}_{i1}(\lambda_i - \lambda_1)\mathbf{x}_i + B\mathbf{x}_1 = k_1\mathbf{x}_1.$$

Now, we shall multiply on the left by $\mathbf{y}_j^*$, yielding

$$\begin{aligned} &\mathbf{y}_j^* \sum_{i=2}^{n} \tilde{t}_{i1}(\lambda_i - \lambda_1)\mathbf{x}_i + \mathbf{y}_j^* B\mathbf{x}1 = k_1\mathbf{y}_j^*\mathbf{x}_1 \\ \Rightarrow \qquad &(\lambda_j - \lambda_1)\tilde{t}_{j1}\mathbf{y}_j^*\mathbf{x}_j + \beta_{j1} = 0 \end{aligned}$$

for $j = 2, 3, ..., n$. Therefore, $|\tilde{t}_{j1}| = \left|\frac{\beta_{j1}}{(\lambda_j - \lambda_1)s(\lambda_j)}\right|$ and

$$\|\mathbf{x}_1(\varepsilon) - \mathbf{x}_1\|_2 \le \varepsilon \sum_{j=2}^{n} \left|\frac{\beta_{j1}}{(\lambda_1 - \lambda_j)s(\lambda_j)}\right| + \mathcal{O}\left(\varepsilon^2\right).$$

Again, we notice that the perturbation in the eigenvector is proportional to $s(\lambda_j)^{-1}$, but in this case, each $s(\lambda_j)$ may contribute. Furthermore, if the eigenvalues are close to one another, we may have difficulty computing eigenvectors.

6.3 Perversity Theorem

One of the more troublesome (or troubling may be more appropriate) results related to the computation of eigenvalues and eigenvectors is the following theorem:

Theorem 6.1. *(Perversity Theorem) Let A have distinct eigenvalues. If, for some eigenvalue λ, $s(\lambda) < 1$, then there exists a matrix E such that λ is a repeated eigenvalue of $A + E$, and*

$$\frac{\|E\|_2}{\|A\|_2} \le \frac{s(\lambda)}{\sqrt{1 - s^2(\lambda)}}.$$

Thus, even if the eigenvalues are distinct, if one eigenvalue is sufficiently ill-conditioned, then computation of the eigenvalues, and especially the eigenvectors, may be very difficult.

6.4 Jacobi's Algorithm, Preliminary Details

For this discussion, we shall assume that A is symmetric, and all the eigenvalues of the matrix are desired. Note that since A is symmetric, all the eigenvalues are well-conditioned.

The basic idea behind Jacobi's Algorithm is that by performing transformations of the form $A^{(k+1)} = Q_k^T A^{(k)} Q_k$, where Q_k is an orthogonal matrix, we can reduce A to its Schur Form. Under the assumption that A is symmetric, this will yield a diagonal matrix, and an orthogonal set of eigenvectors.

The particular orthogonal matrices we will employ are called *Jacobi Rotations.* Recall that in 2-D, if we wish to rotate a vector by an angle θ, we pre-multiply it by the matrix

$$T = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}.$$

This concept can be generalized to higher dimensions, whereby a vector is rotated about one of its principle axes, via pre-multiplication by the matrix,

$$T_{ij} = \begin{bmatrix} I & & & & \\ & \cos\theta & & \sin\theta & \\ & & I & & \\ & -\sin\theta & & \cos\theta & \\ & & & & I \end{bmatrix}.$$

T_{ij} is the identity matrix with its (i, j) and (j, i) elements replaced by $\pm\sin\theta$, respectively, and its (i, i) and (j, j) elements replaced by $\cos\theta$. It is easy to verify that this matrix is orthogonal.

The basic question which will be discussed in the next lecture is, "How do we choose (i_k, j_k) to define our matrices T_{i_k,j_k}?" We will choose (i_k, j_k) so that the off-diagonal elements become smaller, but how do we assure this? Let D_A denote the diagonal matrix whose non-zero elements are the diagonal of the matrix, A. Let us also define,

$$t^2(A) \equiv \|A - D_A\|_F^2 = \sum_{i\neq j} |a_{ij}|^2.$$

We shall describe a sequence as monotonic (decreasing), if

$$t^2(A^{(k+1)}) \leq t^2(A^{(k)}).$$

Let $C = T_{ij}^T A T_{ij}$. Then $\|C\|_F = \|A\|_F$, and

$$\begin{aligned} t^2(C) - t^2(A) &= \|C - D_C\|_F^2 - \|A - D_A\|_F^2 \\ &= \|T_{ij}^T A T_{ij} - D_C\|_F^2 - \|A - D_A\|_F^2 \\ &= (\sum_{i,j} a_{ij}^2 - \sum_i c_{ii}^2) - (\sum_{i,j} a_{ij}^2 - \sum_i a_{ii}^2) \\ &= \sum_i (a_{ii}^2 - c_{ii}^2) \\ &= a_{ii}^2 + a_{jj}^2 - c_{ii}^2 - c_{jj}^2. \end{aligned}$$

The last line is a result of the structure of T_{ij}.

Now let us reduce our scope from T_{ij}, C, and A, to the matrices T,

$$\tilde{C} = \begin{bmatrix} c_{ii} & c_{ij} \\ c_{ji} & c_{jj} \end{bmatrix}, \qquad \text{and} \qquad \tilde{A} = \begin{bmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{bmatrix}.$$

Notice that $\tilde{A}$ and $\tilde{C}$ are symmetric, and $\tilde{C} = T^T \tilde{A} T$. Furthermore,

$$\|\tilde{A}\|_F^2 = a_{ii}^2 + a_{jj}^2 + 2a_{ij}^2.$$

$\tilde{C}$ is similar. Thus,

$$\begin{aligned} t^2(C) - t^2(A) &= (a_{ii}^2 + a_{jj}^2) - (c_{ii}^2 + c_{jj}^2) \\ &= (\|\tilde{A}\|_F^2 - 2a_{ij}^2) - (\|\tilde{C}\|_F^2 - 2c_{ij}^2) \\ &= 2(c_{ij}^2 - a_{ij}^2). \end{aligned}$$

Now, we can conclude that our method is monotonic if for every iteration, $|c_{ij}| < |a_{ij}|$, which is very simple to check.

Lecture 7

Jacobi Algorithm and Tri-Diagonal Form

7.1 Jacobi Algorithm

As described in the previous section, the Jacobi Algorithm will construct a sequence of matrices $\{A^{(k)}\}$ via unitary similarity transformations using rotation matrices of the form

$$T_{ij} = \begin{bmatrix} I & & & & \\ & \cos\theta & & \sin\theta & \\ & & I & & \\ & -\sin\theta & & \cos\theta & \\ & & & & I \end{bmatrix}.$$

Hence,

$$A^{(k+1)} = T^T_{i_k,j_k} A^{(k)} T_{i_k,j_k}$$

and if A is a symmetric matrix, $A^{(k)} \to \Lambda$ as $k \to \infty$ (i.e.,

$$t^2(A^{(k)}) = \sum_{i \neq j} (a^{(k)}_{ij})^2 \to 0$$

as $k \to \infty$). Note that in doing so, we also obtain an orthogonal set of eigenvectors as $\prod_k T_{i_k,j_k}$, if an infinite number of iterations are performed in exact arithmetic. But, after a finite number of iterations, we still obtain an orthogonal set of estimates for the eigenvectors.

We have already shown that if we define $C = T^T_{ij} A T_{ij}$, then $\|A\|^2_F = \|C\|^2_F$, and $t^2(C) - t^2(A) = 2(c^2_{ij} - a^2_{ij})$. Our goal is to define a monotonic (decreasing) sequence, hence we want $|c_{ij}| < |a_{ij}|$. We can express c_{ij} in terms of elements of A and θ as

$$\begin{aligned} c_{ij} &= a_{ij}(\cos^2\theta - \sin^2\theta) + (a_{jj} - a_{ii})\cos\theta\sin\theta \\ &= a_{ij}\cos(2\theta) + \frac{1}{2}(a_{jj} - a_{ii})\sin(2\theta). \end{aligned}$$

The classical approach (although clearly not the only possible approach) is to choose θ such that $c_{ij} = 0$. This yields the equation

$$0 = a_{ij} \cos(2\theta) + \frac{1}{2}(a_{jj} - a_{ii}) \sin(2\theta)$$

which can be rearranged to obtain

$$\tan(2\theta) = \tau = \frac{2a_{ij}}{a_{ii} - a_{jj}},$$

and if we define $t = \tan\theta$, we obtain the quadratic equation

$$0 = t^2 + 2\frac{t}{\tau} - 1$$

which has the solution

$$t = \frac{-1 \pm \sqrt{1+\tau^2}}{\tau}.$$

If we always choose the smallest t, then we are always defining θ such that $|\theta| \leq \frac{\pi}{4}$. This choice will improve the computational stability of the method. We can then define the matrix T_{ij} according to

$$\cos\theta = c = \frac{1}{\sqrt{1+t^2}}, \qquad \sin\theta = s = tc.$$

A different perspective on the how to generate the matrix C is to look at the eigenvalues of $\tilde{A}$.

7.2 Choice Of Indices

One question remains: How does one choose (i, j)? One could choose (I, J) such that

$$|a_{IJ}| \geq \max_{i \neq j} |a_{ij}|$$

(i.e., the largest off-diagonal element is to be annihilated). Then

$$a_{IJ}|^2 \geq \frac{1}{n(n-1)} t^2(A^{(k)}).$$

So,

$$\begin{aligned} t^2(A^{(k+1)}) &= t^2(A^{(k)}) - 2a_{IJ}^2 \\ &\leq t^2(A^{(k)}) - \frac{2}{n(n-1)} t^2(A^{(k)}) \\ &= \left(1 - \frac{2}{n(n-1)}\right) t^2(A^{(k)}) \\ &\leq \left(1 - \frac{2}{n(n-1)}\right)^k t^2(A^{(0)}). \end{aligned}$$

In actuality, we have shown convergence provided that

$$|a_{IJ}|^2 \geq \frac{1}{n(n-1)} \sum_{i \neq j} (a_{ij})^2$$

(i.e., a_{IJ} is larger than the average off-diagonal element).

This analysis shows that convergence is at least linear. It has, however, been shown that asymptotically, convergence is quadratic; i.e.,

$$t(A^{(k+1)}) = c[t(A^{(k)})]^2$$

for some c, provided k is sufficiently large.

A second alternative is to sequentially walk through the indices. This will eliminate the rather costly search to find the maximal off-diagonal element. Forsythe and Henrici proved convergence for this method under the condition $|\theta| \leq \frac{\pi}{4}$.

A third approach is to implement a threshold-based search. Suppose, for example, that $|a_{ij}| \leq 1$. Then, in the first sweep, one could choose to annihilate all elements $> \frac{1}{2}$. In the next sweep, you could ennihilate all elements $> \frac{1}{4}$, and so on.

7.3 Some Implementation Details

What should be chosen as a termination criterion? We know that with infinite precision calculation, for all k,

$$t^2(A^{(k+1)}) < t^2(A^{(k)}).$$

So, if this condition is violated, it must be a result of roundoff errors, so we can stop our computation. An alternative to this is to look at the size of the off diagonals, relative to the matrix. One might choose

$$\frac{t^2(A^{(k+1)})}{\|A\|_F^2} \leq \varepsilon$$

as grounds for termination.

How many iterations are required for convergence? If we use the sequential walk through the indices to define one sweep (the second alternative), $N = \frac{n(n-1)}{2}$ rotations are used in one sweep. The heuristic that the number of sweeps is $O(\log N)$ tends to be reasonable.

Which architecture is most suited for this algorithm? On a serial computer, the Jacobi algorithm is considered impractical for very large problems. However, we can exploit the fact that the rotation T_{ij} only affects the i^{th}

and j^{th} rows and columns of the matrix to obtain a large amount of parallelism in our implementation. Consider, for example, the following rotations applied to a 4×4 matrix:

$(1,2)$	$(3,4)$
$(1,3)$	$(2,4)$
$(1,4)$	$(2,3)$

Each pair of rotations shown can be performed simultaneously, and independently (assuming all row rotations are performed before performing any column rotations). These 6 rotations span all rotations needed for one sweep. In general, approximately $\frac{N}{2}$ processors can be used efficiently with this type of implementation.

How accurate is the algorithm? Let us define the following:

$$\begin{aligned} r &: \# \text{ of sweeps} \\ d_i &: \text{computed eigenvalues} \\ u &: \text{machine precision.} \end{aligned}$$

Wilkinson showed that

$$\sum_{i=1}^{n} (d_i - \lambda_i)^2 \leq (N + k_r)\|A\|_F u.$$

A refined error analysis by Demmel and Veselic shows that if A is positive definite,

$$\frac{|\hat{\lambda}_i - \lambda_i(A)|}{\lambda_i(A)} \approx u\kappa_2(D^{-1}AD^{-1})$$

where $D = \text{diag}(\sqrt{a_{11}}, \sqrt{a_{22}}, ..., \sqrt{a_{nn}})$. The termination criterion plays a role in the above analysis as well.

7.4 Reduction to Tri-Diagonal Form

Clearly, the most significant amount of work required by the Jacobi Algorithm is the application of the rotations to eliminate all of the off-diagonal elements. If we had a tri-diagonal matrix to work with at each step, the application of these rotations would preserve the tri-diagonal structure, and hence our computational workload would be dramatically reduced. Fortunately, we can always perform a unitary similarity transformation on any symmetric matrix to reduce it to a tri-diagonal matrix.

Consider the rotation matrix

$$T = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}.$$

We wish to choose θ so that if we apply T to the vector $\mathbf{a} = \begin{bmatrix} a \\ b \end{bmatrix}$, we obtain $T\mathbf{a} = \alpha\mathbf{e}_1$:

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \sqrt{a^2+b^2} \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

It follows that

$$\sin\theta = \frac{b}{\sqrt{a^2+b^2}}, \qquad \cos\theta = \frac{a}{\sqrt{a^2+b^2}}.$$

Similarly, given an arbitrary symmetric matrix A, we can pick T_{23} so that the 3^{rd} element of the first column is eliminated. Then, forming the similarity transformation $A^{(\prime)} = T_{23}AT_{32}^T$,

$$A^{(\prime)} = \begin{bmatrix} \times & \times & 0 & \times & \dots \\ \times & \times & \dots & & \\ 0 & \vdots & & & \\ \times & & & & \\ \vdots & & & & \end{bmatrix}.$$

Note that this would not have been the case if we had chosen T_{13} for this purpose.

We could proceed down the first column, and across the first row:

$$A^{(\prime\prime)} = T_{24}AT_{24}^T = \begin{bmatrix} \times & \times & 0 & 0 & \times\dots \\ \times & \times & \dots & & \\ 0 & \vdots & & & \\ 0 & & & & \\ \times & & & & \\ \vdots & & & & \end{bmatrix},$$

ultimately yielding

$$\begin{aligned} A^{(2)} &= T_{2n}T_{2,n-1}\cdots T_{23}AT_{23}^TT_{24}^T\cdots T_{2n}^T \\ &= \left[\begin{array}{c|cccc} a_{11} & a_{21}^{(2)} & 0 & \dots & 0 \\ \hline a_{21}^{(2)} & a_{22}^{(2)} & a_{32}^{(2)} & \dots & \\ 0 & a_{32}^{(2)} & & & \\ \vdots & \vdots & & & \\ 0 & & & & \end{array}\right]. \end{aligned}$$

We then proceed to eliminate all elements beyond the third of the second column and row in the same fashion:

$$A^{(3)} = T_{3n}T_{3,n-1}\cdots T_{34}AT_{34}^TT_{35}^T\cdots T_{3n}^T$$
$$= \left[\begin{array}{cc|cccc} a_{11} & a_{21}^{(2)} & 0 & \dots & & 0 \\ a_{21}^{(2)} & a_{22}^{(2)} & a_{32}^{(3)} & 0 & \dots & 0 \\ \hline 0 & a_{32}^{(3)} & a_{33}^{(3)} & a_{34}^{(3)} & \dots & \\ \vdots & 0 & a_{34}^{(3)} & & & \\ & \vdots & \vdots & & & \\ 0 & 0 & & & & \end{array}\right].$$

Hence, after performing $\frac{n(n-1)}{2}$ row and column rotations (which is essentially one sweep of the Jacobi Algorithm), we obtain a tri-diagonal matrix with the same eigenvalues as A. Hence we have obtained the decomposition,

$$A = QJQ^T$$

where

$$J = \begin{bmatrix} a_1 & b_1 & 0 & \dots & 0 \\ b_1 & a_2 & b_2 & \ddots & \vdots \\ 0 & b_2 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & a_{n-1} & b_{n-1} \\ 0 & \dots & 0 & b_{n-1} & a_n \end{bmatrix}.$$

J is called a *Jacobi matrix.* Once we have computed the eigenvalues and eigenvectors of J, we obtain the decomposition

$$J = Z\Lambda Z^T$$

of J, which yields the decomposition

$$A = QJQ^T = QZ\Lambda Z^TQ^T,$$

of A. It follows that the columns of QZ are the eigenvectors of A.

The reduction to tri-diagonal form does not need to be accomplished by Givens rotations as described above. Householder transformations could be used to eliminate all elements below the sub-diagonal in a column in one step (and similarly across the row). The Lanczos Method also produces a tridiagonal matrix, as we shall later see.

If A is not symmetric, one often performs the above reduction steps, but instead of obtaining a tri-diagonal matrix, an upper Hessenberg matrix is obtained. More about this structure will be said in the next lecture.

7.5 Polynomial Recurrence Methods

Once we have a tridiagonal matrix J, methods which compute the roots of polynomials, such as Newton's Method, can be used to find the largest eigenvalue, the smallest eigenvalue, and all eigenvalues inside of an interval. These methods take advantage of recurrence relations developed by expanding the determinant of $(J - \lambda I)$.

Let

$$J_k = \begin{bmatrix} a_1 & b_1 & & & \\ b_1 & \ddots & \ddots & & \\ & \ddots & a_{k-1} & b_{k-1} \\ & & b_{k-1} & a_k \end{bmatrix}$$

(i.e., J_k is the leading $k \times k$ minor of J). Furthermore, let

$$\delta_k(\lambda) = \det(J_k - \lambda I).$$

So,

$$\begin{aligned} \delta_0(\lambda) &\equiv 1, \\ \delta_1(\lambda) &= (a_1 - \lambda), \\ \delta_2(\lambda) &= \det \begin{bmatrix} a_1 - \lambda & b_1 \\ b_1 & a_2 - \lambda \end{bmatrix} \\ &= (a_2 - \lambda)\delta_1(\lambda) - b_1^2 \delta_0(\lambda), \\ \delta_{k+1}(\lambda) &= \det \left[\begin{array}{ccc|c} & & & 0 \\ & J_k - \lambda I & & \vdots \\ & & & 0 \\ & & & b_k \\ \hline 0 & \dots & 0 \;\; b_k & a_{k+1} - \lambda \end{array} \right] \\ &= (a_{k+1} - \lambda)\delta_k(\lambda) - b_k \det \left[\begin{array}{ccc|c} & J_{k-1} - \lambda I & & 0 \\ \hline 0 & \dots & 0 \;\; b_{k-1} & b_k \end{array} \right] \\ &= (a_{k+1} - \lambda)\delta_k(\lambda) - b_k^2 \delta_{k-1}(\lambda). \end{aligned}$$

Note that all orthogonal polynomials satisfy similar three-term recurrence relations, so perhaps they could be used in methods of this nature.

We could use iterative methods like Newton's Method to find λ such that $\delta_n(\lambda) = 0$. In these methods, the above recurence relation would be invaluable. We can easily compute derivatives of $\delta_n(\lambda)$ as according to

$$\delta'_{k+1}(\lambda) = (a_{k+1} - \lambda)\delta'_k(\lambda) - \delta_k(\lambda) - b_k^2 \delta'_{k-1}(\lambda).$$

Then we could iterate as follows:

$$\lambda^{(n+1)} = \lambda^{(n)} - \frac{\delta_n(\lambda^{(n)})}{\delta'_n(\lambda^{(n)})}$$

using recursive evaluations of δ_n and δ'_n. This method parallelizes very well, for if we have r estimates of eigenvalues, then we could use r processors simultaneously. The only problem is that good initial guesses are needed.

Other methods include the construction of *Sturm Sequences*. These sequences tell you how many eigenvalues are to the right and left of a particular value, λ^*, based upon the signs of $\delta_j(\lambda^*)$ for $j = 1, 2, ..., n$. These sequences can be very useful for isolating eigenvalues within an interval, and for obtaining good estimates for use with Newton's Method. It can also be useful for determining if a matrix is positive definite.

Lecture 8

Non-Symmetric Methods and QR Iteration

8.1 Polynomial Recurrence Methods

Suppose $A \neq A^T$, A is real valued, and has been reduced to an Upper Hessenberg matrix:

$$H = \begin{bmatrix} \times & \times & \dots & & \times \\ \times & \times & \dots & & \\ 0 & \times & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \\ 0 & \dots & 0 & \times & \times \end{bmatrix},$$

where $H = QAQ^T$, and $Q^TQ = I$. If we only need to find a few eigenvalues, several of the ideas previously introduced for J will work for H. So we must ask, how do we evaluate $\det(H - \lambda I)$ for a particular eigenvalue estimate, λ?

Suppose we had a matrix, $\bar{H}$ such that

$$\bar{H} = \begin{bmatrix} \times & \times & \dots & \times & \bar{h}_{1n} \\ \bar{h}_{21} & \times & \dots & \times & 0 \\ 0 & \bar{h}_{32} & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \times & 0 \\ 0 & \dots & 0 & \bar{h}_{n,n-1} & 0 \end{bmatrix},$$

Then, $\det(\bar{H}) = \bar{h}_{1n}(\bar{h}_{21}\bar{h}_{32}\cdots\bar{h}_{n,n-1})$, which is easily computed.

To achieve this, we shall add to the last column of H a linear combination of the other columns of H such that the resulting last column is of the desired form. This is equivalent to post multiplying H by a matrix, X:

$$\bar{H} = HX = H[\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_{n-1}, \mathbf{x}]$$

where $x_n = 1$. Then

$$\begin{aligned}\det(\bar{H}) &= \det(H)\det(X) \\ &= \det(H)\cdot(-1)^n(\det(I_{n-1})) \\ &= \det(H)\cdot(-1)^n = (-1)^n\det(H).\end{aligned}$$

Note that we have expanded $\det X$ about the last row, so there would be a sign change if n is odd. But the characteristic polynomial of $\bar{H}$ has the same roots as that of H, so $\lambda(\bar{H}) = \lambda(H)$.

Now, we need to find x_i so that we can take the determinant of $(\bar{H}-\lambda I)$. Choosing x_i so that the last $(n-1)$ elements of $(H-\lambda I)$ are eliminated is equivalent to solving the system of equations:

$$\begin{array}{lllll} x_1h_{21} & +x_2(h_{22}-\lambda) & +\dots & +x_{n-1}h_{2,n-1} & = -h_{2n}, \\ & x_2h_{21} & +x_3(h_33-\lambda) \quad +\dots & +x_{n-1}h_{3.n-1} & = -h_{3n}, \\ & & \ddots & \vdots & \vdots \\ & & & x_{n-1}h_{n,n-1} & = -(h_{nn}-\lambda). \end{array} \tag{8.1}$$

Notice that this $(n-1)\times(n-1)$ linear system is upper triangular, thus the solution can be obtained in $\mathcal{O}\left(\frac{n^2}{2}\right)$ operations. Then,

$$\det(H-\lambda I) = (h_{1n}+(x_1(h_{11}-\lambda)+x_2h_{12}+\dots+x_{n-1}h_{1,n-1}))(h_{21}h_{32}\cdots h_{n,n_1}),$$

and since we are only concerned with the roots of the characteristic polynomial, we can ignore the factor $(h_{21}\cdots h_{n,n-1})$.

As before, we may wish to compute $\frac{d}{d\lambda}(\det(H-\lambda I))$, or ignoring the extra factor,

$$\frac{d}{d\lambda}\left(\frac{\det(H-\lambda I)}{h_{21}\cdots h_{n,n-1}}\right) = \frac{d\,x_1}{d\lambda}(h_{11}-\lambda) + \frac{d\,x_2}{d\lambda}h_{12} + \dots + \frac{d\,x_{n-1}}{d\lambda}h_{1,n-1} - x_1.$$

The terms $\frac{d\,x_i}{d\lambda}$ can be found by differentiating the system of equations (8.1) that defined the x_i, yielding:

$$\begin{array}{lllll} x_1'h_{21} & +x_2'(h_{22}-\lambda) & +\dots & +x_{n-1}'h_{2,n-1} & = x_2, \\ & x_2'h_{21} & +x_3'(h_33-\lambda) \quad +\dots & +x_{n-1}'h_{3.n-1} & = x_3, \\ & & \ddots & \vdots & \vdots \\ & & & x_{n-1}'h_{n,n-1} & = 1. \end{array} \tag{8.2}$$

If a second (or higher) derivative is required, we differentiate (8.2), which entails replacing x_i in the righthand side vector with $2x_i'$ (and obviously replacing the variables x_i' in the system with the new variables x_i'').

Lecture 9

QR Iteration

9.1 The Basic Iteration

In this lecture we will discuss the next evolutionary step from the power method to a practical method for computing all of the eigenvalues and eigenvectors of a matrix A, the QR iteration:

Algorithm 9.1. **(QR Iteration)**

```
Let $A \equiv A^{(0)}$.
while unconverged do
    Form a $QR$ decomposition of $A^{(k)}$ such that $Q$ is orthogonal,
        and $R$ is upper triangular: $A^{(k)} = Q^{(k)}R^{(k)}$.
    Multiply the factors in reverse order: $A^{(k+1)} = R^{(k)}Q^{(k)}$.
end while
```

It is easily seen that each iteration performs a unitary similarity transformation of $A^{(k)}$:

$$A^{(k+1)} = R^{(k)}Q^{(k)} = Q^{(k\,T)}A^{(k)}Q^{(k)}.$$

In general,

$$A^{(k+1)} = Q^{(k\,T)} \cdots Q^{(0\,T)}AQ^{(0)} \cdots Q^{(k)}.$$

9.2 Aside: LR Iteration

Another algorithm for computing eigenvalues, called *LR iteration* was devised by Rutishauser. This algorithm is not as numerically stable as QR iterations, but can be used for some sparse data structures.

Algorithm 9.2. **(LR Iteration)**

```
Let $A \equiv A^{(0)}$.
while unconverged do
    Form an $LU$ decomposition of $A^{(k)}$ such that $L$ is lower
        triangular, and $R$ is upper triangular: $A^{(k)} = L^{(k)}R^{(k)}$.
    Multiply the factors in reverse order: $A^{(k+1)} = R^{(k)}L^{(k)}$.
end while
```

If L is non-singular, this iteration amounts to the application of similarity transformations on A since $A^{(k+1)} = L^{(k)\,-1}A^{(k)}L^{(k)}$.

9.3 Convergence of QR Iteration

Let us assume

$$|\lambda_1| > |\lambda_2| > ... > |\lambda_n| > 0.$$

Perhaps the use of a shift is necesary for the assumption that the eigenvalues have non-zero modulus, but we shall assume it for simplicity. This means that $A = X\Lambda X^{-1}$, where Λ is diagonal, and ordered by modulus as described above. Then,

$$\begin{aligned} A^k &= X\Lambda^k X^{-1} \\ &\equiv X\Lambda^k Y. \end{aligned}$$

Let us take a QR decomposition of X, and an LU decomposition of Y. Thus $X = QR$, where R is upper triangular, and Q is unitary; and $Y = LU$, where L is unit lower triangular, and U is upper triangular. Note that a QR decomposition will always exist, but the LU decomposition exists if and only if

$$\det\left(\begin{bmatrix} y_{11} & \cdots & y_{1k} \\ \vdots & \ddots & \vdots \\ y_{k1} & \cdots & y_{kk} \end{bmatrix}\right) \neq 0 \qquad \text{for } k = 1, 2, ..., n.$$

We shall say more about this later.

Inserting these decompositions into our expression for A^k:

$$\begin{aligned} A^k &= QR\Lambda^k LU \\ &= QR(\Lambda^k L\Lambda^{-k})\Lambda^k U. \end{aligned}$$

Note:

$$\Lambda^k L\Lambda^{-k} = \begin{bmatrix} 1 & & 0 \\ & \ddots & \\ l_{ij}\left(\frac{\lambda_i}{\lambda_j}\right)^k & \ddots & \\ & & 1 \end{bmatrix} \qquad (i > j).$$

So,

$$\Lambda^k L \Lambda^{-k} = I + E^{(k)},$$

where $E^{(k)} \to 0$ as $k \to \infty$ since $0 < \frac{|\lambda_i|}{|\lambda_j|} < 1$ for $i > j$. Thus,

$$\begin{aligned} A^k &= QR(I + E^{(k)})\Lambda^k U \\ &= Q(R + RE^{(k)}R^{-1}R)\Lambda^k U \\ &= Q(I + RE^{(k)}R^{-1})R\Lambda^k U \\ &\equiv Q(I + F^{(k)})R\Lambda^k U, \end{aligned}$$

where $F^{(k)} \to 0$ as $k \to \infty$. We can now define a QR decomposition of $I + F^{(k)} = \tilde{Q}^{(k)}\tilde{R}^{(k)}$ such that as $F^{(k)} \to 0$, $\tilde{Q}^{(k)} \to I$ and $\tilde{R}^{(k)} \to I$. Then,

$$\begin{aligned} A^k &= Q\tilde{Q}^{(k)}\tilde{R}^{(k)}R\Lambda^k U \\ &= (Q\tilde{Q}^{(k)})(\tilde{R}^{(k)}R\Lambda^k U) \qquad (9.1) \\ &= \text{(unitary)(upper triangular)}. \end{aligned}$$

Thus, we have created a QR decomposition of A^k. To establish a link between this decomposition and a sequence of iterates $A^{(k)}$ generated by QR iteration, we employ the following result:

Proposition Let $P^{(0)}, \ldots, P^{(k)}$ be a sequence of orthogonal matrices, and let A be an invertible matrix. Then, for $j = 0, \ldots, k$,

$$A^j = P^{(j)}T^{(j)},$$

where $T^{(j)}$ is upper triangular, if and only if the sequence $A^{(0)} \equiv A, A^{(1)}, \ldots, A^{(k)}$ defined by

$$A^{(k)} = P^{(k)*}AP^{(k)}$$

is a sequence of QR iterates of A; i.e. for $j = 0, \ldots, k-1$,

$$A^{(j+1)} = Q^{(j)*}A^{(j)}Q^{(j)}$$

for some orthogonal matrix $Q^{(j)}$, and $Q^{(j)*}A^{(j)}$ is upper triangular.

By this proposition,

$$A^{(k)} = P^{(k)*}AP^{(k)}$$

is a QR iterate for each k. Thus,

$$A^{(k)} = \tilde{Q}^{(k)*}Q^*AQ\tilde{Q}^{(k)}.$$

But,

$$A = X\Lambda X^{-1} = (QR)\Lambda(R^{-1}Q^*).$$

So,

$$A^{(k)} = \tilde{Q}^{(k)*}R\Lambda R^{-1}\tilde{Q}^{(k)}$$

which converges to an upper triangular matrix as $k \to \infty$, due to the convergence of $\tilde{Q}^{(k)*}$ to I. We can apply the converse of the propostion to conclude that for any sequence $\{A^{(k)}\}$ of QR iterates of A, we obtain a QR decomposition of A^k of the form (9.1), where $\tilde{Q}^{(k)}$ and $\tilde{R}^{(k)}$ must both converge to I.

The rate of convergence is dependent on the spacing of the eigenvalues. In particular, $\frac{\lambda_i}{\lambda_j}$ is responsible for the rate at which $\Lambda^k L \Lambda^{-k} \to I$.

9.4 Disorder of Eigenvalues

What happens if we cannot construct the LU decomposition? That is, suppose

$$\det\left(\begin{bmatrix} y_{11} & \cdots & y_{1k} \\ \vdots & \ddots & \vdots \\ y_{k1} & \cdots & y_{kk} \end{bmatrix}\right) = 0$$

(i.e., we have encountered a zero along the diagonal of $Y^{(k)}$ during Gaussian Elimination). We shall then permute the rows of Y such that

$$\Pi Y = LU$$

exists. Thus,

$$\begin{aligned} A^k &= X\Lambda^k Y \\ &= X\Lambda^k \Pi^T LU \\ &= (X\Pi^T)(\Pi\Lambda^k\Pi^T)LU. \end{aligned}$$

Then define

$$\Delta^k = \Pi\Lambda^k\Pi^T = \operatorname{diag}(\lambda_{i_1}^k, ..., \lambda_{i_n}^k).$$

Now, Δ^k is Λ^k with the eigenvalues re-ordered, and $(X\Pi^T)$ are the eigenvectors reordered in a corresponding manner.

Now we can proceed with the analysis as before, and the convergence rate depends on the rate at which the (p, q) element of $(\Delta^k L \Delta^{-k})$ tends to zero. But,

$$(\Delta^k L \Delta^{-k})_{pq} = \left(\frac{\lambda_{i_p}}{\lambda_{i_q}}\right)^k l_{pq}.$$

So, it is possible for $|\lambda_{i_p}| > |\lambda_{i_q}|$, and we should be concerned that the method might not converge. Fortunately, this is not the case. In our construction of Π, we only permuted rows if a zero was encountered along the diagonal. In applying the permutation, we introduced a zero element in the (p, q) element of L, so $(\Delta^k L \Delta^{-k})_{pq} = 0$ in this case.

If there were a block of eigenvalues with the same modulus, say

$$|\lambda_t| = |\lambda_{t+1}| = ... = |\lambda_{t+s}|,$$

then the QR iteration would converge to an upper triangular matrix, with the exception of an $s \times s$ dense block along the diagonal.

9.5 Uniqueness of the Hessenberg Form

It can be shown that the Hessenberg form of A (i.e., $A = QHQ^*$) has the same uniqueness property as the QR factorization. Recall that the QR factorization is "unique" provided the starting vector, $\mathbf{q}_1$, is specified, and the signs of the diagonal elements of R are specified. For Hessenberg form, uniqueness can be shown provided $\mathbf{q}_1$ is specified, within pre and post-multiplication of H by a diagonal matrix, $D = \text{diag}(\pm 1, ..., \pm 1)$ (i.e., multiplication of each vector $\mathbf{q}_i$ by ± 1). If complex arithmetic is necessary, then D will be diagonal with arbitrary numbers of unit modulus.

We will now show that the first column of Q (i.e., $\mathbf{q}_1$) determines H and Q with an arbitrary sign choice. Assume that real arithmetic is to be used.

Given a vector, $\mathbf{q}_1$, such that $\|\mathbf{q}_1\|_2 = 1$,

$$\begin{aligned} A\mathbf{q}_1 &= h_{11}\mathbf{q}_1 + h_{21}\mathbf{q}_2, \\ \Rightarrow \qquad \mathbf{q}_1^* A\mathbf{q}_1 &= h_{11}. \\ \text{And,} \qquad h_{21}\mathbf{q}_2 &= A\mathbf{q}_1 - h_{11}\mathbf{q}_1, \\ \Rightarrow \qquad |h_{21}| &= \|A\mathbf{q}_1 - h_{11}\mathbf{q}_1\|_2. \end{aligned}$$

Here is where an arbitrary sign choice enters the process. Let $h_{21} > 0$, then

$$\mathbf{q}_2 = \frac{1}{h_{21}}(A\mathbf{q}_1 - h_{11}\mathbf{q}_1).$$

Note that if we prescribed $h_{21} < 0$ the only change to $\mathbf{q}_2$ would be a sign change. Furthermore, this process fails if $h_{21} = 0$. This failure corresponds to the case where $\mathbf{q}_1$ is an eigenvector of A. In this case, we can pick any $\mathbf{q}_2$ that is orthogonal to $\mathbf{q}_1$, and we must redefine our concept of "uniqueness".

In general,

$$\begin{aligned} A\mathbf{q}_k &= h_{1k}\mathbf{q}_1 + h_{2k}\mathbf{q}_2 + \cdots + h_{k+1,k}\mathbf{q}_{k+1} \\ \Rightarrow \qquad \mathbf{q}_i^* A\mathbf{q}_k &= h_{ik} \qquad \text{for } i \le k, \\ \text{and,} \qquad |h_{k+1,k}| &= \|A\mathbf{q}_k - h_{1k}\mathbf{q}_1 - \cdots - h_{kk}\mathbf{q}_k\|_2. \end{aligned}$$

Again, we may pick the sign of this element, so let $h_{k+1,k} > 0$ as before. Similarly, the process may breakdown if (for instance) $\mathbf{q}_k$ is in a subspace spanned by k eigenvectors of A, which we could deal with in the same manner as before. If this is not the case,

$$\mathbf{q}_{k+1} = \frac{1}{h_{k+1,k}}(A\mathbf{q}_k - h_{1k}\mathbf{q}_1 - ... - h_{kk}\mathbf{q}_k),$$

and the Hessenberg form is thus constructed “uniquely”.

The following theorem summarizes the nature of this uniqueness. We say that an upper Hessenberg matrix H is *unreduced* if all subdiagonal elements are nonzero.

Theorem 9.3. (Implicit Q Theorem) *Suppose that $Q^TAQ = H$ and $V^TAV = G$, where Q and V are orthogonal, and H and G are upper Hessenberg. Let k be the smallest integer such that $h_{k+1,k} = 0$, with the convention that $k = n$ if H is unreduced. If $\mathbf{q}_1 = \mathbf{v}_1$, then $\mathbf{q}_i = \pm\mathbf{v}_i$ and $|h_{i,i-1}| = |g_{i,i-1}|$ for $i = 2, \ldots, k$. Furthermore, if $k < n$, then $g_{k+1,k} = 0$.*

Lecture 10

Practical QR Iterations

10.1 Shift Strategy

We have seen how convergence depends upon the spacing between the eigenvalues, in particular $\frac{\lambda_{i+1}}{\lambda_i}$, which can often be near 1. We have also seen how the QR Iterations are related to the Power Method, so perhaps we can use a shift as before. Now, instead of working with $A^{(k)}$, we will iterate as follows:

Algorithm 10.1. Shifted QR Iteration

```
Let $A^{(0)} \equiv A$.
while unconverged do
    Choose a shift $\sigma_k$
    Compute the QR factorization of $A^{(k)}$ shifted:
        $Q^{(k)}R^{(k)} = A^{(k)} - \sigma_k I$
    Multiply in reverse order and undo the shift:
        $A^{(k+1)} \equiv R^{(k)}Q^{(k)} + \sigma_k I$
end while
```

A more practical application of this algorithm is suggested by the following:

$$\begin{aligned} A^{(0)} - \sigma_0 I &= Q^{(0)}R^{(0)} \\ A^{(1)} &= R^{(0)}Q^{(0)} + \sigma_0 I \\ &= Q^{(0)T}(A^{(0)} - \sigma_0 I)Q^{(0)} + \sigma_0 I \end{aligned}$$

Thus, if A had already been reduced to upper Hessenberg form, we could apply shifted QR iterations by first subtracting the shift σ_k from the diagonal elements of A, then performing $n-1$ rotations on the left, then the transpose of the same rotations on the right, and then adding σ back into the diagonal elements. This would require $\mathcal{O}\left(n^2\right)$ operations per iteration in the unsymmetric case, and $\mathcal{O}\left(n\right)$ operations in the symmetric case.

It is easy to see that this shift strategy has not changed the eigenvalues computed. It has hopefully served only to accelerate convergence:

$$\begin{aligned} A^{(1)} &= Q^{(0)T}(A^{(0)} - \sigma_0 I)Q^{(0)} + \sigma_0 Q^{(0)T}Q^{(0)} \\ &= Q^{(0)T}A^{(0)}Q^{(0)}, \\ \Rightarrow A^{(k+1)} &= Q^{(k)T}A^{(k)}Q^{(k)} \\ &= (Q^{(0)}Q^{(1)}\cdots Q^{(k)})^T A(Q^{(0)}\cdots Q^{(k)})^T. \end{aligned}$$

This result looks like the same result as for the unshifted QR iterations, but now, the $Q^{(j)}$ are based on the shift parameter σ_j as well as A.

10.2 The Single Shift Strategy

How does one choose a shift σ_k? The simplest tactic is known as the *single shift strategy*. It is based on the fact that shifting by an eigenvalue causes deflation to occur in one step, at least in exact arithmetic. Therefore, it is suggested that we attempt to choose an approximate eigenvalue to be the shift.

Suppose that A has been reduced to an upper Hessenberg matrix H. Then we can choose the (n,n) element of $A^{(k)}$ to be the shift. It has been shown by Stewart that this strategy yields quadratic convergence in the unsymmetric case, and cubic in the symmetric case.

In the symmetric case, however, Wilkinson has given heuristic reasons why an alternative shift, known as the *Wilkinson shift*, is a better choice. Let $H = A^{(k)}$. The Wilkinson shift is then defined to be the eigenvalue of the 2×2 matrix

$$\tilde{H} = \begin{bmatrix} h_{n-1,n-1} & h_{n-1,n} \\ h_{n,n-1} & h_{nn} \end{bmatrix}$$

that is closer to h_{nn}.

10.3 The Double Shift Strategy

Unfortunately, in the unsymmetric case, the single shift strategy does not work well if the eigenvalues of the block $\tilde{H}$ are complex, since h_{nn} tends to be a poor approximation to an eigenvalue.

Let $H = A^{(k)}$, and let a_1 and a_2 be the eigenvalues of $\tilde{H}$. If they are complex, then we must have $a_1 = \overline{a}_2$. We can proceed as follows using these complex shifts to obtain the next iterate $A^{(k+1)}$:

$$\begin{aligned} H - a_1 I &= U_1 R_1 \quad (QR \text{ factorization}) \\ H_1 &= R_1 U_1 + a_1 I \\ H_1 - a_2 I &= U_2 R_2 \quad (QR \text{ factorization}) \\ A^{(k+1)} = H_2 &= R_2 U_2 + a_2 I \end{aligned}$$

It is easy to see that

$$H_2 = (U_1U_2)^H H(U_1U_2).$$

Suppose that we let

$$M = U_1U_2R_2R_1.$$

Then

$$\begin{aligned} M &= U_1U_2R_2R_1 \\ &= U_1(H_1 - a_2I)R_1 \\ &= U_1H_1R_1 - a_2U_1R_1 \\ &= U_1(R_1U_1 + a_1I)R_1 - a_2U_1R_1 \\ &= (H - a_1I)^2 + a_1(H - a_1I) - a_2(H - a_1I) \\ &= H^2 - 2a_1H + a_1^2I + a_1H - a_1^2I - a_2H + a_1a_2I \\ &= H^2 - (a_1 + a_2)H + a_1a_2I \\ &= H^2 - \text{tr}(\tilde{H})H + \det(\tilde{H})I \end{aligned}$$

and therefore M is a real matrix. It follows from the fact that $M = ZR$, where $Z = U_1U_2$ and $R = R_2R_1$, is a QR factorization of M that Z can be chosen to be real, and thus $H_2 = Z^T HZ$ is also real.

Since our initial matrix H and final matrix H_2 are both real, it is desirable to compute H_2 without using complex arithmetic. We could proceed by computing M, computing the QR factorization $M = ZR$ and applying Z to H to compute H_2, but constructing M directly can be computationally costly ($\mathcal{O}\left(n^3\right)$ operations since H is Hessenberg), so this is not a practical approach.

10.4 Double Implicit Shifts

Fortunately, we can show that its explicit construction of M is not necessary to generate H_2. We can apply the Implicit Q Theorem and construct a matrix Z_1 such that $Z_1^T HZ_1$ is upper Hessenberg, and Z_1 has the same column as the matrix Z that triangularizes M. To compute this first column of Z, we note that Z can be written as a product of Householder matrices $Z = P_1 \cdots P_{n-1}$ where each P_i is designed to zero elements in column i of M below the diagonal. It follows that $P_i\mathbf{e}_1 = \mathbf{e}_1$ for $i > 1$, and therefore $Z\mathbf{e}_1 = P_1\mathbf{e}_1$.

We can compute P_1 very efficiently, as it is a Householder matrix constructed so that $P_1M\mathbf{e}_1$ is a mutiple of $\mathbf{e}_1$. Therefore, it is only necessary to construct the first column of M, rather than the entire matrix. This column has only three nonzero elements, so it can be computed in $O(1)$ operations.

Once we compute P_1, we can apply it directly to H. We then need to construct the remaining Householder matrices $P_2, \ldots, P_{n-1}$, but we know how to do so without computing the remainder of M: they transform $P_1^T H P_1$ to upper Hessenberg form.

We shall use the following 6×6 example to illustrate the process.

$$\begin{aligned} M &= \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} & h_{15} & h_{16} \\ h_{21} & h_{22} & h_{23} & h_{24} & h_{25} & h_{26} \\ 0 & h_{32} & h_{33} & h_{34} & h_{35} & h_{36} \\ 0 & 0 & h_{43} & h_{44} & h_{45} & h_{46} \\ 0 & 0 & 0 & h_{54} & h_{55} & h_{56} \\ 0 & 0 & 0 & 0 & h_{65} & h_{66} \end{bmatrix} \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} & h_{15} & h_{16} \\ h_{21} & h_{22} & h_{23} & h_{24} & h_{25} & h_{26} \\ 0 & h_{32} & h_{33} & h_{34} & h_{35} & h_{36} \\ 0 & 0 & h_{43} & h_{44} & h_{45} & h_{46} \\ 0 & 0 & 0 & h_{54} & h_{55} & h_{56} \\ 0 & 0 & 0 & 0 & h_{65} & h_{66} \end{bmatrix} \\ &- s \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} & h_{15} & h_{16} \\ h_{21} & h_{22} & h_{23} & h_{24} & h_{25} & h_{26} \\ 0 & h_{32} & h_{33} & h_{34} & h_{35} & h_{36} \\ 0 & 0 & h_{43} & h_{44} & h_{45} & h_{46} \\ 0 & 0 & 0 & h_{54} & h_{55} & h_{56} \\ 0 & 0 & 0 & 0 & h_{65} & h_{66} \end{bmatrix} + tI \\ &= \begin{bmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \end{bmatrix}. \end{aligned}$$

And,

$$\begin{aligned} M\mathbf{e}_1 &= \begin{bmatrix} h_{11}h_{11} + h_{12}h_{21} \\ h_{21}h_{11} + h_{22}h_{21} \\ h_{32}h_{21} \\ 0 \\ 0 \\ 0 \end{bmatrix} - s \begin{bmatrix} h_{11} \\ h_{21} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + t \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} h_{11}^2 + h_{12}h_{21} - sh_{11} + t \\ h_{21}(h_{11} + h_{22} - s) \\ h_{21}h_{32} \\ 0 \\ 0 \\ 0 \end{bmatrix} \equiv \begin{bmatrix} x \\ y \\ z \\ 0 \\ 0 \\ 0 \end{bmatrix}. \end{aligned}$$

We shall now define Given's Rotations, R_{12} and R_{13} to eliminate the 2^{nd}

and 3^{rd} elements of this matrix:

$$R_{13}R_{12}\begin{bmatrix} x \\ y \\ z \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \sqrt{x^2+y^2+z^2} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

We shall apply similarity transformations to H using these rotations yielding

$$\begin{aligned} C &= R_{13}R_{12}HR_{12}^TR_{13}^T \\ &= R_{13}\begin{bmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & h_{33} & h_{34} & h_{35} & h_{36} \\ 0 & 0 & h_{43} & h_{44} & h_{45} & h_{46} \\ 0 & 0 & 0 & h_{54} & h_{55} & h_{56} \\ 0 & 0 & 0 & 0 & h_{65} & h_{66} \end{bmatrix} R_{13}^T \\ &= \begin{bmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \times & 0 & \times & h_{44} & h_{45} & h_{46} \\ 0 & 0 & 0 & h_{54} & h_{55} & h_{56} \\ 0 & 0 & 0 & 0 & h_{56} & h_{66} \end{bmatrix}. \end{aligned}$$

We finish one Double Implicit Shift iteration by reducing this matrix to upper Hessenberg Form. The entire process will require $\mathcal{O}\left(8n^2\right)$ multiplications per iteration.

As for the rate of convergence of this method, if

$$A^{(k)} = \begin{bmatrix} F^{(k)} & G^{(k)} \\ G^{(k)T} & H^{(k)} \end{bmatrix}$$

where $G^{(k)} = \varepsilon K^{(k)}$ and $\varepsilon < \frac{1}{3}|\lambda_{n-r} - \lambda_n|$, then

$$\|G^{(k+1)}\|_E \leq \frac{10r\varepsilon^3}{|\lambda_{n-r} - \lambda_n|^2}.$$

However, this result is an asymptotic result, which is difficult to observe due to its cubic nature. Generally, after about 2 major iterations the lower 1×1 or 2×2 block decouples from the rest of the matrix, and we can proceed with a matrix of smaller dimension.

10.5 Banded Matrices

One can show that for symmetric matrices, the bandwidth is preserved when using QR iterations. Let $A = A^T$ be a matrix with bandwidth $2m-1$. Then

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1m} & & & \\ \vdots & & & \ddots & & \\ a_{1m} & & & & \ddots & \\ & \ddots & & & & a_{n-m+1,n} \\ & & \ddots & & & \vdots \\ & & & a_{n-m+1,n} & \cdots & a_{nn} \end{bmatrix}.$$

Then, according to the QR factorization of A,

$$A = QR = [\mathbf{q}_1, ..., \mathbf{q}_n]R$$

$$\begin{bmatrix} a_{11} \\ \vdots \\ a_{1m} \\ 0 \\ \vdots \\ 0 \end{bmatrix} = r_{11} \begin{bmatrix} q_{11} \\ \vdots \\ q_{1m} \\ q_{1,m+1} \\ \vdots \\ q_{1n} \end{bmatrix}.$$

So, $q_{1,m+1} = ... = q_{1n} = 0$. Similarly,

$$A\mathbf{e}_2 = r_{12}\mathbf{q}_1 + r_{22}\mathbf{q}_2.$$

So, $q_{2,m+2} = ... = q_{2n} = 0$. And so,

$$Q = \begin{bmatrix} q_{11} & \cdots & & q_{n-m+1,1} & \cdots & q_{n1} \\ \vdots & & & \vdots & & \vdots \\ q_{1m} & & & & & \\ & \ddots & & & & \\ & & \ddots & \vdots & & \vdots \\ & & & q_{n-m+1,n} & \cdots & q_{nn} \end{bmatrix}.$$

Then,

$$\begin{aligned} A^{(1)} &= RQ \\ &= Q^T A Q \\ &= Q^T A^T Q \\ &= A^{(1)T}. \end{aligned}$$

So, QR iteration has preserved the symmetry of A in $A^{(1)}$. Furthermore,

$$\begin{aligned}
A^{(1)}\mathbf{e}_1 &= RQ\mathbf{e}_1 \\
&= R\mathbf{q}_1 \\
&= \begin{bmatrix} \mathbf{r}_1^T \\ \vdots \\ \mathbf{r}_m^T \\ \mathbf{r}_{m+1}^T \\ \vdots \\ \mathbf{r}_n^T \end{bmatrix} \begin{bmatrix} q_{11} \\ \vdots \\ q_{1m} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{r}_1^T\mathbf{q}_1 \\ \vdots \\ \mathbf{r}_m^T\mathbf{q}_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \\
A^{(1)}\mathbf{e}_2 &= \begin{bmatrix} \mathbf{r}_1^T\mathbf{q}_1 \\ \vdots \\ \mathbf{r}_{m+1}^T\mathbf{q}_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.
\end{aligned}$$

And so forth, yielding,

$$A^{(1)} = \begin{bmatrix} a_{11}^{(1)} & \cdots & a_{n-m+1,1}^{(1)} & \cdots & a_{n1}^{(1)} \\ \vdots & & \vdots & & \vdots \\ a_{1m}^{(1)} & & & & \\ & \ddots & & & \\ & & \ddots \quad \vdots & & \vdots \\ & & a_{n-m+1,n}^{(1)} & \cdots & a_{nn}^{(1)} \end{bmatrix}.$$

However, $A^{(1)} = A^{(1)T}$, so

$$A^{(1)} = \begin{bmatrix} a_{11}^{(1)} & \cdots & a_{1m}^{(1)} & & & \\ \vdots & & & \ddots & & \\ a_{1m}^{(1)} & & & & \ddots & \\ & \ddots & & & & a_{n-m+1,n}^{(1)} \\ & & \ddots & & & \vdots \\ & & & a_{n-m+1,n}^{(1)} & \cdots & a_{nn}^{(1)} \end{bmatrix}.$$

So, if one had a symmetric matrix with bandwidth $2m + 1$, and one did not wish to reduce this to tridiagonal form, one could perform QR Iterations while preserving bandwidth.

10.6 Skew-symmetric Matrices

A *Skew-symmetric matrix* S is a real-valued matrix which satisfies $S^T = -S$. These matrices are normal matrices, have 0's along the diagonal, and have eigenvalues that are either 0 or are purely imaginary (which occur in complex conjugate pairs).

To find the eigenvalues of these matrices, the double implicit shift strategy for QR iterations can be employed. We will show in the homework that the iterates formed in this manner will maintain the skew-symmetric property. Thus, if these matrices have been reduced to tri-diagonal form via unitary similarity transformations, the zero values along the diagonal should be preserved, and only $n - 1$ values need to be stored and computed at each iteration.

10.7 Constructing the Singular Value Decomposition

To construct the SVD of a matrix A, we would like to exploit the fact that the orthogonal matrices U and V are the Schur vectors of AA^T and A^TA respectively. This suggests that we could use QR Iterations on these matrices, but explicitly forming these matrices is undesirable. Our first step should be to reduce A to some more simple structure via orthogonal transformations. Since the SVD involves pre- and post- multiplication by different orthogonal matrices, this seems natural for the reduction process as well. In doing so, we can reduce the matrix to Upper Bi-diagonal Form.

To perform this process, we shall perform one step of the standard QR Factorization process, constructing a Householder matrix, P_1, which eliminates all elements below the diagonal. We will then apply a Householder transformation, Q_1 on the right so that the elements to the right of the super-diagonal in the first row are eliminated. This transformation will only act on the columns $(2, ..., n)$, so the zeros along the first column will be preserved as shown below. I am giving what appears to be a square example, but this does not need to be the case.

$$A = \begin{bmatrix} a_{11} & a_{12} & \mathbf{a}_{13}^T \\ a_{21} & a_{22} & \mathbf{a}_{23}^T \\ \mathbf{a}_{31} & \mathbf{a}_{32} & A_{33} \end{bmatrix}.$$

$$A^{(0.5)} \equiv P_1 A = \begin{bmatrix} a_{11} & a_{12} & \mathbf{a}_{13}^T \\ 0 & a_{22}^{(0.5)} & \mathbf{a}_{23}^{(0.5)T} \\ \mathbf{0} & \mathbf{a}_{32}^{(0.5)} & A_{33}^{(0.5)} \end{bmatrix}.$$

$$A^{(1)} \equiv A^{(0.5)} Q_1 = \begin{bmatrix} a_{11} & a_{12} & \mathbf{a}_{13}^T \\ 0 & a_{22}^{(0.5)} & \mathbf{a}_{23}^{(0.5)T} \\ \mathbf{0} & \mathbf{a}_{32}^{(0.5)} & A_{33}^{(0.5)} \end{bmatrix} \begin{bmatrix} 1 & 0 & \mathbf{0}^T \\ 0 & q_{22} & \mathbf{q}_{23}^T \\ \mathbf{0} & \mathbf{q}_{32} & Q_{33} \end{bmatrix}.$$

$$= \begin{bmatrix} a_{11} & b_{12} & \mathbf{0}^T \\ 0 & a_{22}^{(1)} & \mathbf{a}_{23}^{(1)T} \\ \mathbf{0} & \mathbf{a}_{32}^{(1)} & A_{33}^{(1)} \end{bmatrix}.$$

We will then repeat this process to eliminate the non-zeros below the diagonal in the second column of $A^{(1)}$ (i.e., eliminate $\mathbf{a}_{32}^{(1)}$) and eliminate the non-zeros to the right of the super-diagonal in the second row (i.e., convert $\mathbf{a}_{23}^{(1)T}$ to $b_{23}\mathbf{e}_1^T$). And so forth, reducing the matrix A to Upper Bi-diagonal Form. Note that at the n^{th} step of the above process, we do not need to apply a matrix Q_n on the right, but we do need to apply P_n on the left if A is not square and $A_{m \times n}$ with $m > n$. The reverse is true if $m < n$ (i.e., we apply only Q_n). For the remainder of the discussion, I shall assume $m > n$.

Now, we have

$$A^{(n)} = P_n \cdots P_1 A Q_1 \cdots Q_{n-1} = PAQ = \begin{bmatrix} B \\ 0 \end{bmatrix}.$$

Thus, if

$$\begin{aligned} B &= X \Sigma Y^T, \\ A &= PX \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} Y^T Q \\ &= U \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} V^T. \end{aligned}$$

Recall that Householder Transformations are symmetric, and orthogonal.

We could proceed by constructing $B^T B$ and BB^T, but this has poor numerical stability properties. Instead, we will use the matrix

$$\tilde{B} = \begin{bmatrix} 0 & B \\ B^T & 0 \end{bmatrix}.$$

This matrix has eigenpairs given by:

$$\begin{aligned}
\begin{bmatrix} 0 & B \\ B^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} &= \lambda \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \\
\text{i.e.,} \qquad B\mathbf{y} &= \lambda \mathbf{x} \\
\text{and,} \qquad B^T\mathbf{x} &= \lambda \mathbf{y}. \\
\text{So,} \qquad B^T B\mathbf{y} &= \lambda B^T \mathbf{x} \\
&= \lambda^2 \mathbf{y} \\
&= \sigma^2 \mathbf{y}. \\
\text{And,} \qquad B B^T\mathbf{x} &= \lambda B \mathbf{y} \\
&= \lambda^2 \mathbf{x} \\
&= \sigma^2 \mathbf{x}.
\end{aligned}$$

So $\tilde{B}$ has eigenpairs given by

$$(\lambda, \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}) \qquad \text{and} \qquad (-\lambda, \begin{bmatrix} \mathbf{x} \\ -\mathbf{y} \end{bmatrix})$$

where $\mathbf{x}$ is an eigenvector of BB^T and $\mathbf{y}$ is an eigenvector of B^TB, each of which has corresponding eigenvalue λ^2 for their respective problem. Note that we could have chosen $(-\lambda, \begin{bmatrix} -\mathbf{x} \\ \mathbf{y} \end{bmatrix})$ as an alternative since this is just a scalar multiple (-1) of the other choice.

There is one more modification to this procedure that we can make. The matrix, $\tilde{B}$ is of the form

$$\begin{aligned}
\tilde{B} &= \begin{bmatrix} 0 & B \\ B^T & 0 \end{bmatrix} \\
&= \begin{bmatrix}
0 & & & & \alpha_1 & \beta_1 & & \\
 & 0 & & & & \alpha_2 & \ddots & \\
 & & \ddots & & & & \ddots & \beta_{n-1} \\
 & & & 0 & & & & \alpha_n \\
\alpha_1 & & & & 0 & & & \\
\beta_1 & \alpha_2 & & & & 0 & & \\
 & \ddots & \ddots & & & & \ddots & \\
 & & \beta_{n-1} & \alpha_n & & & & 0
\end{bmatrix}.
\end{aligned}$$

This matrix has a very large bandwidth that will get filled quickly wtih QR Iterations. If we say these rows and columns are ordered $(1, 2, ..., 2n)$ then applying the column permutation $(n+1, 1, n+2, 2, ...)$ and likewise to the rows, we obtain the matrix,

$$\hat{B} = \Pi \tilde{B} \Pi^T = \begin{bmatrix} 0 & \alpha_1 & & & & \\ \alpha_1 & 0 & \beta_1 & & & \\ & \beta_1 & 0 & \alpha_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \beta_{n-1} & \ddots & \alpha_n \\ & & & & \alpha_n & 0 \end{bmatrix}.$$

An alternative approach is to implicity apply QR iteration to the symmetric matrix $B^T B$. At each iteration, the lower 2×2 block of the matrix $A^{(k)} = B^{(k)*} B^{(k)}$, where $B^{(0)} \equiv B$, can be computed in order to obtain the Wilkinson shift, and the Implicit Q Theorem can be applied as in the double implicit shift strategy to uniquely determine the matrix that upper-triangularizes $A^{(k)}$. In this case, the additional Householder matrices are selected so as to restore $B^{(k)}$ to bidiagonal form.

Lecture 11

The Generalized Eigenvalue Problem

11.1 Comments on QR Iterations

QR iterations are useful for finding all the eigenvalues of dense matrices, particularly for real matrices, as complex eigenvalues can be found without resorting to complex arithmetic. Complex matrices can also be used with this method. For symmetric matrices, if you pre-process the matrix so that it is reduced a tri-diagonal matrix, iterations only require $\mathcal{O}(n)$ floating point operations. QR iterations will also preserve bandwidth for symmetric matrices. For skew symmetric matrices the iterations will preserve the skew tridiagonal structure that these matrices can be reduced to, finding the purely imaginary eigenvalues with real operations. The method is also useful for finding singular values.

The basic procedure is as follows:

1. Reduce the matrix to one of the following forms:
   (a) If $A = A^T$, reduce to tridiagonal form.
   (b) If $A^T = -A$, reduce to skew tridiagonal form.
   (c) If A is not one of the above, reduce to upper Hesenberg form.
   (d) If singular values are desired, reduce to bi-diagonal form.
2. Apply QR iterations with double implicit shifts. This will preserve the reduced form of the matrix after each iteration.
3. The product of rotation matrices (Householder reflections, or Gram-Schmidt projections) yields Q from the Murgnahan-Wintner form (or Schur form if the eigenvalues are real-valued). $A^{(\infty)}$ yields R from the Murgnahan-Wintner form. The desired eigenvalues can be extracted from the block diagonal.

11.2 The Generalized Eigenvalue Problem

Suppose we wish to find eigenpairs $(\lambda, \mathbf{x})$ such that

$$A\mathbf{x} = \lambda B\mathbf{x}. \tag{11.1}$$

This problem is much more delicate than the standard eigenvalue problem as shown in this simple 3×3 example.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{x}_i = \lambda_i \begin{bmatrix} 1 & 0 & 0 \\ 0 & \varepsilon & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{x}_i.$$

Clearly, $\mathbf{x}_1 = \mathbf{e}_1$ with corresponding eigenvalue $\lambda_1 = 1$. This pair was simple enough, but the problems appear in the other two pairs. The second pair is $(\mathbf{e}_2, \frac{1}{\varepsilon})$. This is numerically difficult to obtain, as a small perturbation in the matrix can yield a very large change in this eigenvalue. The third pair is even more difficult to deal with. The third pair is $(\mathbf{e}_3, \lambda)$; i.e., any value for λ will satisfy the generalized eigenvalue equation (11.1) if $\mathbf{e}_3$ is used for $\mathbf{x}$. Thus there an infinte number of potential eigenvalues. This scenario was clearly not possible in the standard case, and will be difficult to detect with a computational scheme. The basic structure of this problem which corresponds to the Jordan Canonical Form is called the *Kronecker Canonical Form*. It is rather complicated to describe, so we shall not do so here.

11.3 The QZ Algorithm

The basic algorithm associated with this problem was derived by Moler and Stewart, and is called the *QZ algorithm*. The basic idea of the algorithm is to use different orthogonal matrices on the left and right of A and B (i.e., replace A with QAZ and B with QBZ) to simultaneously reduce the matrices to upper triangular form. Then, the eigenvalues are easily obtained via the diagonal elements (i.e., $\lambda = \frac{a_{ii}}{b_{ii}}$). Notice that pre- or post-multiplication of both sides of (11.1) by a unitary matrix is analogous to performing a similarity transformation in the standard eigenvalue problem, since if Q and Z are orthogonal,

$$\begin{aligned} QA\mathbf{x} &= \lambda QI\mathbf{x} \\ \Rightarrow QAQ^T\mathbf{y} &= \lambda\mathbf{y}, \end{aligned}$$

and

$$\begin{aligned} AZ\mathbf{x} &= \lambda IZ\mathbf{x} \\ \Rightarrow Z^TAZ\mathbf{x} &= \lambda\mathbf{x}. \end{aligned}$$

Without loss of generality, we shall assume that B is an upper triangular matrix. If this is not the case, we shall compute the QR factorization of $B = QR$, then multiply both sides of the generalized eigenvalue equation by the matrix Q^T, yielding the desired forms. Thus,

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \lambda \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ 0 & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & b_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

Now, we shall apply the rotation $Q_{n-1,n}$ to zero out the a_{n1} element, and perform the same operation on B:

$$Q_{n-1,n}A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ \vdots & \ddots & \ddots & \vdots \\ a'_{n-1,1} & a'_{n-1,2} & \dots & a'_{n-1,n} \\ 0 & a'_{n2} & \dots & a'_{nn} \end{bmatrix},$$

$$Q_{n-1,n}B = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ 0 & \ddots & \dots & b_{2n} \\ \vdots & \vdots & b'_{n-1,n-1} & b'_{n-1,n} \\ 0 & \dots & b'_{n,n-1} & b'_{nn} \end{bmatrix}.$$

However, this introduces a new non-zero element below the diagonal in B, so we must apply the rotation $Z_{n-1,n}$ on the right of A and B to return B to triangular form. It should be clear that this does not introduce new non-zero elements to A, as only the $n-1^{st}$ column and the n^{th} column of A are modified. This process of applying rotations on the left and right can proceed until A is upper Hessenberg, for when we try to eliminate an element on the sub-diagonal of A, it introduces a non-zero element in the same location of B, and similarly, applying a rotation on the right to eliminate the newly created element on B will place a new non-zero in the spot that was just eliminated in A. So,

$$QAZ = H, \qquad QBZ = T$$

where H is upper Hessenberg, and T is upper triangular, yielding the generalized eigenvalue equation

$$H\mathbf{y} = \lambda T\mathbf{y} \quad \Rightarrow \quad HT^{-1}T\mathbf{y} = \lambda T\mathbf{y},$$

if we assume that T is invertible. Now, we have the simpler problem of applying QR iterations to HT^{-1} with double implicit shifts.

It would be nice to think that we could do this without explicitly constructing T^{-1}, and this is indeed the case. To construct the shift, we only need to know the bottom right 3×2 block of T (one element of which is known to be 0 since T is triangular). Thus, we need 5 elements of T^{-1}, and these 5 elements are easily constructed. If real arithmetic is used in the complete implementation, B will tend to a Murgnahan-Wintner type of Block Triangular matrix (with 2×2 blocks that correspond to complex eigenvalues), and A will tend to triangular as iterations proceed.

11.4 Symmetric Positive Definite Generalized Eigenvalue Problem

Suppose $A = A^T$ and B is symmetric positive definite, and we wish to solve the generalized eigenvalue problem (11.1). Clearly, the QZ algorithm will not preserve the symmetry of the problem, so it may be more practical to convert it into a symmetric standard eigenvalue problem as follows: Let $B = FF^T$, which could be performed using Cholesky factorization. Then,

$$\begin{aligned} A\mathbf{x} &= \lambda B\mathbf{x} \\ &= \lambda F F^T \mathbf{x} \\ \Rightarrow \qquad F^{-1}A\mathbf{x} &= \lambda F^T\mathbf{x} \\ \Rightarrow \qquad F^{-1}AF^{-T}F^T\mathbf{x} &= \lambda F^T\mathbf{x} \\ \Rightarrow \qquad F^{-1}AF^{-T}\mathbf{y} &= \lambda \mathbf{y}. \end{aligned}$$

We can now compute the eigenvalues and eigenvectors of $C = F^{-1}AF^{-T}$. The power method might be a good choice for this problem, as C would not need to be formed explicitly, as would be necessary for QR iterations to proceed.

11.5 Constrained Maximization of the Quadratic Form

Often one wishes to find

$$\max_{\mathbf{x}^T\mathbf{x}=1} \mathbf{x}^T A\mathbf{x}$$

for a symmetric matrix A. This is the same problem as finding the largest eigenvalue of A. However, suppose you also want to impose the additional constraint

$$C^T\mathbf{x} = \mathbf{0}.$$

One application where this problem arises is in statistics. Suppose we just solved the least squares problem

$$\min_{\mathbf{x}} \|\mathbf{b} - C^T\mathbf{x}\|_2.$$

The normal equations tell us that

$$C^TC\widehat{\mathbf{x}} = C^T\mathbf{b} \quad \Rightarrow \quad r = \mathbf{b} - C\widehat{\mathbf{x}}, \quad C^T\mathbf{r} = 0.$$

Then, we desire knowledge of the maximum value of $\sum_{i=1}^{n-1}(r_{i+1}-r_i)^2$. However,

$$\begin{aligned}\sum_{i=1}^{n-1}(r_{i+1}-r_i) &= \begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}_{1\times n-1} \begin{bmatrix}-1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1\end{bmatrix}_{n-1\times n} \begin{bmatrix}r_1 \\ \vdots \\ r_n\end{bmatrix}. \\ &= \mathbf{1}^T B\mathbf{r}.\end{aligned}$$

So, we are maximizing

$$\sum_{i=1}^{n-1}(r_{i+1}-r_i)^2 = \mathbf{r}^T B^T B\mathbf{r} \equiv \mathbf{r}^T A\mathbf{r}$$

where $A = B^TB = A^T$.

To solve this problem, we can use Lagrange multipliers. Let

$$\phi(\mathbf{x},\lambda,\mu) = \mathbf{x}^TA\mathbf{x} - \lambda\mathbf{x}^T\mathbf{x} + 2\mathbf{x}^TC\mu.$$

We then set $\nabla_x\phi = 0$. This yields

$$\begin{aligned} 0 &= A\mathbf{x} - \lambda\mathbf{x} + C\mu \\ &= C^TA\mathbf{x} - \lambda C^T\mathbf{x} + C^TC\mu \\ &= C^TA\mathbf{x} + C^TC\mu \\ \therefore \qquad \mu &= (C^TC)^{-1}C^TA\mathbf{x}. \end{aligned}$$

Thus, we obtain our solution by finding the largest eigenvalue of

$$\begin{aligned} &A\mathbf{x} - \lambda\mathbf{x} + C(C^TC)^{-1}C^TA\mathbf{x} = 0 \\ &\text{i.e., } [I - C(C^TC)^{-1}C^T]A\mathbf{x} = \lambda\mathbf{x}. \end{aligned}$$

Notice that $P = (I - C(C^TC)^{-1}C^T)$ is a symmetric matrix, as is A by assumption, but the product is in general no longer symmetric. However,

$$\begin{aligned} P^2 &= PP \\ &= I - 2C(C^TC)^{-1}C^T + C(C^TC)^{-1}C^TC(C^TC)^{-1}C^T \\ &= I - C(C^TC)^{-1}C^T \\ &= P. \end{aligned}$$

Thus P is a *projection matrix*. Therefore, instead of looking for the largest eigenvalue of PA, we shall look for that of P^2A. However,

$$\lambda(P^2A) = \lambda(P(PA)) = \lambda((PA)P) = \lambda(PAP),$$

and PAP is a symmetric matrix. Furthermore, since C is an $n \times k$ matrix,

$$\lambda_{n-k+1} = ... = \lambda_n = 0.$$

Now, we can substitute the QR factorization of C into P:

$$\begin{aligned}
C &= Q \begin{bmatrix} R \\ 0 \end{bmatrix}. \\
P &= I_n - Q \begin{bmatrix} R \\ 0 \end{bmatrix} (R^T R)^{-1} \begin{bmatrix} R^T, & 0 \end{bmatrix} Q^T \\
&= I_n - Q \begin{bmatrix} I_k & 0 \\ 0 & 0 \end{bmatrix} Q^T \\
&= Q \begin{bmatrix} 0 & 0 \\ 0 & I_{n-k} \end{bmatrix} Q^T. \\
\therefore \quad PAP &= Q \begin{bmatrix} 0 & 0 \\ 0 & I_{n-k} \end{bmatrix} Q^T A Q \begin{bmatrix} 0 & 0 \\ 0 & I_{n-k} \end{bmatrix} Q^T.
\end{aligned}$$

Since we are interested in computing the largest eigenvalue of PAP, we can ignore the outer Q and Q^T as this is a similarity transformation, and look at the largest eigenvalue of

$$\begin{bmatrix} 0 & 0 \\ 0 & I_{n-k} \end{bmatrix} Q^T A Q \begin{bmatrix} 0 & 0 \\ 0 & I_{n-k} \end{bmatrix}.$$

Let us now define

$$\hat{A} = Q^T A Q = \begin{bmatrix} \hat{A}_{11} & \hat{A}_{12} \\ \hat{A}_{21} & \hat{A}_{22} \end{bmatrix}.$$

Then,

$$\begin{bmatrix} 0 & 0 \\ 0 & I_{n-k} \end{bmatrix} Q^T A Q \begin{bmatrix} 0 & 0 \\ 0 & I_{n-k} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & \hat{A}_{22} \end{bmatrix}.$$

Therefore, $\lambda(PAP) = \lambda(\hat{A}_{22}) \cup \{0\}$.

So, we have two algorithms for solving our problem. First, we can construct P, and then find the eigenvalues of PAP. Alternatively, we can compute the QR factorization of C, construct $\hat{A}_{22}$, and find the eigenvalues of $\hat{A}_{22}$.

Which algorithm should be used? $\hat{A}_{22}$ has none of the data structure that previously existed, but we now have a smaller problem to deal with. Therefore, if we want to preserve the data structure, we should use the first algorithm. This is particularly true if only a few constraints are needed, for then we can use the power method to exploit the data structure.

Lecture 12

Constrained Least Squares and Introduction to Lanczos Method

12.1 Constrained Least Squares Problem

12.1.1 Lagrange Multipliers

Suppose we were given $A_{m\times n}$, $\mathbf{b}$, and a scalar α, and wished to find the vector $\mathbf{x}$ such that $\|\mathbf{b}-A\mathbf{x}\|_2$ is minimized, while maintaining $\|\mathbf{x}\|_2 = \alpha$. We could solve this problem by using Lagrange multipliers, constructing the potential function

$$\begin{aligned}\phi(x,\lambda) &= \|\mathbf{b}-A\mathbf{x}\|_2^2 + \lambda(\|\mathbf{x}\|_2^2 - \alpha)\\ &= (\mathbf{b}^T - \mathbf{x}^T A^T)(\mathbf{b}-A\mathbf{x}) + \lambda(\mathbf{x}^T\mathbf{x} - \alpha^2).\end{aligned}$$

Setting the gradient of this function with respect to x (and not λ) equal to zero yields the equation

$$\nabla_x \phi = -2A^T\mathbf{b} + 2A^T A\mathbf{x} + 2\lambda\mathbf{x} = 0$$

which has the solution

$$\mathbf{x} = (A^T A + \lambda I)^{-1} A^T \mathbf{b}.$$

provided the inverse of $A^T A + \lambda I$ exists. Substituting this result into the constraint $\|\mathbf{x}\|_2^2 = \alpha^2$ yields

$$\psi(\lambda) = \mathbf{b}^T A (A^T A + \lambda I)^{-2} A^T \mathbf{b} - \alpha^2 = 0.$$

Let $A = U\Sigma V^T$ be the singular value decomposition of A. Then our constraint equation becomes

$$\begin{aligned}\psi(\lambda) = 0 &= \mathbf{b}^T U\Sigma V^T (V\Sigma^T U^T U\Sigma V^T + \lambda I)^{-2} V\Sigma^T U^T \mathbf{b} - \alpha^2 \\ &= \mathbf{b}^T U\Sigma V^T (V(\Sigma^T\Sigma + \lambda I)V^T)^{-2} V\Sigma^T U^T \mathbf{b} - \alpha^2 \\ &= \mathbf{b}^T U\Sigma V^T (V(\Sigma^T\Sigma + \lambda I)V^T V(\Sigma^T\Sigma + \lambda I)V^T)^{-1} V\Sigma^T U^T \mathbf{b} - \alpha^2 \\ &= \mathbf{b}^T U\Sigma(\Sigma^T\Sigma + \lambda I)^{-2}\Sigma^T U^T \mathbf{b} - \alpha^2.\end{aligned}$$

Letting $\beta = U^T\mathbf{b}$, we arrive at

$$\psi(\lambda) = \sum_{i=1}^{n} \frac{\beta_i^2 \sigma_i^2}{(\sigma_i^2 + \lambda)^2} - \alpha^2 = 0.$$

Thus, $\psi(\lambda)$ has a pole whenever $\lambda = -\sigma_i^2$. Furthermore, $\psi(\lambda)$ decreases from ∞ to $-\alpha^2$ as λ goes from $-\sigma_n^2$ to ∞. Hence, after computing the SVD of A, we could use a non-linear equation solver, such as Newton's method, to find $\hat{\lambda}$ such that the constraint equation is satisfied. Then, we can obtain the solution vector, $\mathbf{x}$. The algorithm is summarized as:

Algorithm 12.1. (Constrained Least Squares Solution)

Given A, $\mathbf{b}$, and α^2,
Compute the SVD of A. (Lots of work)
Compute $\beta = U^T\mathbf{b}$. (Trivial)
Solve $\psi(\lambda) = 0$ for $\hat{\lambda}$. (Some work)
Solve for $\mathbf{x}$ either $(A^TA + \hat{\lambda})\mathbf{x} = A^T\mathbf{b}$ or $V(\Sigma^T\Sigma + \hat{\lambda}I)V^T\mathbf{x} = V^T\Sigma\beta$. (Some work)

12.1.2 Quadratic Eigenvalue Problem

An alternative method for solving the Constrained Least Squares Problem is to construct the $(n+1)\times(n+1)$ system of equations:

$$\begin{bmatrix} (A^TA + \lambda I)^2 & A^T\mathbf{b} \\ \mathbf{b}^TA & \alpha^2 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \xi \end{bmatrix} = 0.$$

That is,

$$\begin{aligned}(A^TA - \lambda I)^2\mathbf{u} + A^T\mathbf{b}\xi &= 0, &(12.1)\\ b^TA\mathbf{u} + \alpha^2\xi &= 0. &(12.2)\end{aligned}$$

Equation (12.1) says

$$\mathbf{u} = -(A^TA + \lambda I)^{-2}A^T\mathbf{b}\xi$$

which can be substituted into (12.2) yielding

$$(-\mathbf{b}^TA(A^TA + \lambda I)^{-2}A^T\mathbf{b} + \alpha^2)\xi = 0.$$

Setting $\xi = 1$ to obtain non-trivial solutions yields the same equation generated from the Lagrange multipliers approach. If, however, we solved (12.2) for ξ to obtain

$$\xi = -\frac{1}{\alpha^2}\mathbf{b}^T A\mathbf{u}$$

and substituted this expression into (12.1), we obtain

$$[(A^T A + \lambda I)^2 - \frac{1}{\alpha^2}A^T\mathbf{b}\mathbf{b}^T A]\mathbf{u} = 0.$$

Hence, if we can solve this quadratic eigenvalue problem, perhaps using the method discussed earlier in the quarter, we can avoid computing the SVD of A.

12.2 Introduction to Lanczos Method

The Lanczos Method is primarily used for computing the eigenvalues of A where A is symmetric (which we shall assume throughout this discussion), although it can be extended for non-symmetric matrices. The basis of the method is in computing the coefficients of the characteristic polynomial:

$$\phi(\lambda) = \xi_0 + \xi_1\lambda + ... + \xi_{n-1}\lambda^{n-1} - \lambda^n.$$

Recall that if λ is an eigenvalue, $\phi(\lambda) = 0$. Furthermore, the Cayley-Hamilton Theorem tells us that A satisfies its own characteristic equation; i.e.

$$\phi(A) = \xi_0 I + \xi_1 A + ... + \xi_{n-1}A^{n-1} - A^n = 0.$$

Thus, if we let $\mathbf{q}$ be an arbitrary vector such that $\|\mathbf{q}\|_2 = 1$, we have

$$\phi(A)\mathbf{q} = \xi_0\mathbf{q} + \xi_1 A\mathbf{q} + ... + \xi_{n-1}A^{n-1}\mathbf{q} - A^n\mathbf{q} = 0.$$

This suggests that we can compute the coefficients of the characteristic polynomial by solving the system of equations

$$\begin{bmatrix}\mathbf{q}, & A\mathbf{q}, & \dots, & A^{n-1}\mathbf{q}\end{bmatrix}\xi = A^n\mathbf{q}$$

where $\xi = \begin{bmatrix}\xi_0 & \xi_1 & ... & \xi_{n-1}\end{bmatrix}^T$. If we define the Krylov Sequence $\{\mathbf{q}_i\}$ according to $\mathbf{q}_0 = \mathbf{q}$, $\mathbf{q}_{i+1} = A\mathbf{q}_i$ for $i = 0, ..., n-1$, this can be expressed as

$$\begin{bmatrix}\mathbf{q}_0 & \mathbf{q}_1 & \cdots & \mathbf{q}_{n-1}\end{bmatrix}\xi = \mathbf{q}_n,$$

or simply $Q\xi = \mathbf{q}_n$.

Unfortunately, this method breaks down very easily. For example, if $\mathbf{q}_0$ is an eigenvector of A, then rank$(Q) = 1$. Similarly, if $\mathbf{q}_0$ is in a subspace spanned by only r eigenvectors of A, then rank$(Q) = r$. Furthermore, even if $\mathbf{q}_0$ has non-zero components in the direction of each eigenvector of A,

Q will be highly ill-conditioned for large n, because $\mathbf{q}_k$ is the result of k iterations of the power method without rescaling the iterates. Hence the last columns of Q look very much like the eigenvector that corresponds to the largest eigenvalue of A. This problem becomes even more dramatic if the eigenvalues are spaced far apart in modulus, which is a property we would ordinarily like for solving an eigenvalue problem.

Instead of looking at $Q\xi = \mathbf{q}_n$, we will look at the normal equations $Q^TQ\xi = Q^T\mathbf{q}_n$ (which is even more horribly ill-conditioned). The $(r+1, s+1)$ element of Q^TQ is given by

$$(Q^TQ)_{r+1,s+1} = \mathbf{q}_r^T\mathbf{q}_s = \mathbf{q}_0^T A^{rT} A^s \mathbf{q}_0 = \mathbf{q}_0^T A^{r+s}\mathbf{q}_0 \equiv \mu_{r+s}.$$

Thus,

$$Q^TQ = \begin{bmatrix} \mu_0 & \mu_1 & \mu_2 & \cdots & \mu_{n-1} \\ \mu_1 & \mu_2 & \mu_3 & \cdots & \mu_n \\ \mu_2 & \mu_3 & \mu_4 & & \vdots \\ \vdots & \vdots & & \ddots & \mu_{2n-3} \\ \mu_{n-1} & \mu_n & \cdots & \mu_{2n-3} & \mu_{2n-2} \end{bmatrix}.$$

Notice that each counter diagonal has the same value. Any such matrix is called a *Hankel Matrix.* Systems of equations involving Hankel matrices can be solved using $\mathcal{O}\left(n^2\right)$ operations. In our case, Q^TQ is symmetric positive semi-definite.

If $A\mathbf{z}_i = \lambda\mathbf{z}_i$ for $i = 1, 2, ..., n$, then $\mathbf{q} = \sum_{i=1}^n \alpha_i\mathbf{z}_i$. Hence,

$$\mu_r = \mathbf{q}A^r\mathbf{q} = \sum_{i=1}^n \lambda_i^r\alpha_i^2 = \int \lambda^r d\alpha(\lambda).$$

Thus, μ_r is a Stieltjes integral, where the distribution function $\alpha(\lambda)$ is a step function which takes the value α_i^2 for λ between λ_i and λ_{i+1}. We mention this only for reference purposes, not as a computational scheme.

If $A \neq A^T$, then

$$\mathbf{q}_{r+1}^T\mathbf{q}_{s+1} = \mathbf{q}_0^T(A^r)^TA^s\mathbf{q}_0$$

does not lead to a Hankel Matrix. Instead, we must define two sequences,

$$\begin{aligned} \mathbf{q}_r &= A^r\mathbf{q}_0 \qquad \text{for } r = 0, 1, ..., n-1 \\ \mathbf{p}_s &= (A^T)^s\mathbf{p}_0 \qquad \text{for } s = 0, 1, ..., n-1. \end{aligned}$$

Now,

$$(P^TQ)_{r+1,s+1} = \mathbf{p}_0^T[(A^T)^r]^TA^s\mathbf{q}_0 = \mathbf{p}_0A^{r+s}\mathbf{q}_0 \equiv \mu_{r+s}.$$

This matrix is a Hankel Matrix. Furthermore, it is symmetric, but it is not likely to be positive semi-definite. We can then express $\mathbf{q}_0$ as a linear combination of the right eigenvectors of A, $\mathbf{q}_0 = \sum_{i=1}^n \alpha_i\mathbf{z}_i$ where $A\mathbf{z}_i = \lambda_i\mathbf{z}_i$, and $\mathbf{p}_0$ as a linear combination of the left eigenvectors of A, $\mathbf{p}_0 = \sum_{j=1}^{n} \beta_j \mathbf{v}_j$ where $A^T\mathbf{v}_i = \lambda_i \mathbf{v}_i$. Then, since $\mathbf{v}_j^T\mathbf{z}_i = 0$ for $i \neq j$, and $\mathbf{v}_i^T\mathbf{z}_i = s_i$,

$$\mu_{r+s} = \sum_{i=1}^{n} \alpha_i \beta_i s_i \lambda_i^{r+s} = \int \lambda^{r+s} d\gamma(\lambda)$$

only if $\alpha_i\beta_i > 0$ so that the measure $\gamma(\alpha)$ is positive and increasing. Sometimes, one ignores this detail and performs formal manipulations anyway.

Returning to the symmetric case, we have an efficient method for solving the normal equations, but, as mentioned before, the system is horribly ill-conditioned. To alleviate our troubles, we will try to find a different basis for our polynomial vector space, and hence a different representation for the characteristic polynomial:

$$\begin{aligned}\phi(\lambda) &= \xi_0 + \xi_1\lambda + ... + \xi_{n-1}\lambda^{n-1} - \lambda^n \\ &= \eta_0 p_0(\lambda) + \eta_1 p_1(\lambda) + ... + \eta_{n-1}p_{n-1}(\lambda) + \eta_n p_n(\lambda).\end{aligned}$$

One possibility would be to choose the Chebyshev polynomials. These polynomials form an orthogonal family with respect to the weight function $\frac{1}{\sqrt{x^2-1}}$, not the measure $\alpha(\lambda)$. If we choose to proceed in this fashion, we would need to manipulate $\nu_r = \int p_r(\lambda) d\alpha(\lambda)$ instead of $\mu_r = \int \lambda^r d\alpha(\lambda)$. The Lanczos method will generate a set of polynomials $\{p_i(\lambda)\}_{i=0}^{n}$ that are orthogonal with respect to $\alpha(\lambda)$. That is,

$$\int p_r(\lambda)p_s(\lambda)d\alpha(\lambda) = 0 \qquad \text{for } r \neq s.$$

Note that all orthogonal polynomials satisfy a 3-term recurrence relationship

$$\begin{aligned}p_{r+1}(\lambda) = (\lambda - \alpha_{r+1})p_r(\lambda) - \beta_r^2 p_{r-1}(\lambda),\\ p_{-1}(\lambda) = 0, \qquad p_0(\lambda) = 1.\end{aligned}$$

Thus,

$$\begin{aligned}p_{r+1}(A) &= (A - \alpha_{r+1}I)p_r(A) - \beta_r^2 p_{r-1}(A) \\ \Rightarrow p_{r+1}(A)\mathbf{q}_0 &= (A - \alpha_{r+1}I)p_r(A)\mathbf{q}_0 - \beta_r^2 p_{r-1}(A)\mathbf{q}_0,\end{aligned}$$

(i.e., we have generated a sequence of orthogonal vectors). Let $\mathbf{x}_r = p_r(A)\mathbf{q}_0$. Then,

$$\mathbf{x}_{r+1} = (A - \alpha_{r+1}I)\mathbf{x}_r - \beta_r^2\mathbf{x}_{r-1}.$$

We will show how to choose the recursion coefficients α_i and β_i such that $\mathbf{x}_r^T\mathbf{x}_s = 0$ for $r \neq s$.

Lecture 13

Lanczos and Orthogonal Polynomials

13.1 Chebyshev Polynomials

Given a matrix A, we devised an algorithm for finding the coefficients of its characteristic polynomial $\phi(\lambda)$. Recall

$$\phi(\lambda) = \xi_0 + \xi_1\lambda + ... + \xi_{n-1}\lambda^{n-1} - \lambda^n.$$

Unfortunately, this method involved solving a horribly ill-conditioned system of linear equations. To solve this problem, we proposed finding a new basis for the space of polynomials of degree n, and finding the coefficients of ϕ with respect to this new basis. Hence, we wish to define $\{p_j(\lambda)\}_{i=0}^{n}$ and compute $\{\eta_j\}_{j=0}^{n-1}$ such that

$$\phi(\lambda) = \eta_0 p_0(\lambda) + \eta_1 p_1(\lambda) + \cdots + \eta_{n-1}p_{n-1}(\lambda) - \eta_n p_n(\lambda).$$

We would like $p_j(\lambda)$ to be a monic polynomial of degree j, and $\{p_j(\lambda)\}$ to be an orthogonal family. With our choice of monic polynomials, we can conclude $\eta_n = -1$ (as with the standard polynomial basis). We mentioned two possibilities; the first was to pre-select a family of such polynomials, the second was to construct one on the fly.

Suppose A is symmetric positive definite. Let us define $\tilde{A} = I - \alpha A$, with $\alpha = \frac{2}{\lambda_1+\lambda_n}$, so that each eigenvalue λ of $\tilde{A}$ satisfies $-1 \le \lambda \le 1$. We can then choose the *Chebyshev polynomials*

$$\begin{aligned} p_j(\lambda) &= \cos(j\cos^{-1}\lambda) \qquad \text{for } |\lambda| \le 1, \\ &= \cosh(j\cosh^{-1}\lambda) \qquad \text{otherwise.} \end{aligned}$$

Each polynomial in this family undergoes equi-oscillation between ± 1 on the interval $|\lambda| \le 1$. Also, as a family, they are the smallest orthogonal polynomials on the interval $|\lambda| \le 1$, and they grow the fastest outside of this interval.

Furthermore, the 3-term recurrence relationship for these polynomials is

$$p_{j+1}(\lambda) = 2\lambda p_j(\lambda) - p_{j-1}(\lambda). \tag{13.1}$$

If we define $\mathbf{p}_{-1} = \mathbf{0}$ and choose $\mathbf{p}_0$ arbitrarily, we can post-multiply the above expression (13.1) by $\mathbf{p}_0$ and hence define a sequence of vectors, according to

$$\mathbf{p}_{j+1} = 2\tilde{A}\mathbf{p}_j - \mathbf{p}_{j-1},$$

or, equivalently, $\mathbf{p}_j = p_j(\tilde{A})\mathbf{p}_0$. We can then find the coefficients of the characteristic polynomial for $\tilde{A}$ by solving the system $P\eta = \mathbf{p}_n$, where $P = \begin{bmatrix}\mathbf{p}_0 & \cdots & \mathbf{p}_{n-1}\end{bmatrix}$ and $\eta = \begin{bmatrix}\eta_0 & \cdots & \eta_{n-1}\end{bmatrix}$. If we solve this system via the normal equations $P^TP\eta = P^T\mathbf{p}_n$, and write

$$\mathbf{p}_0 = \sum_{i=1}^{n} \alpha_i \mathbf{u}_i$$

where each $\mathbf{u}_i$ is an eigenvector of $\tilde{A}$ satisfying

$$\tilde{A}\mathbf{u}_i = \lambda_i \mathbf{u}_i, \quad \|\mathbf{u}_i\|_2 = 1, \quad i = 1, \ldots, n,$$

then

$$\begin{aligned}
(P^TP)_{r+1,s+1} &= (p_r(\tilde{A})\mathbf{p}_0, p_s(\tilde{A})\mathbf{p}_0) \\
&= \mathbf{p}_0^T p_r(\tilde{A}) p_s(\tilde{A})\mathbf{p}_0 \\
&= \sum_{i=1}^{n} \alpha_i^2 p_r(\lambda_i) p_s(\lambda_i) \\
&= \int p_r(\lambda) p_s(\lambda)\, d\hat{\pi}(\lambda).
\end{aligned}$$

But,

$$\begin{aligned}
p_r(\lambda)p_s(\lambda) &= \cos(r\theta)\cos(s\theta) \\
&= \frac{1}{2}(\cos((r+s)\theta) + \cos(|r-s|\theta))
\end{aligned}$$

where $\theta = \cos^{-1}\lambda$. Therefore, if we define

$$\alpha_j = \int p_j(\lambda)\, d\hat{\pi}(\lambda),$$

then

$$\begin{aligned}
\int p_r(\lambda)p_s(\lambda)\, d\hat{\pi}(\lambda) &= \frac{1}{2}\int (p_{r+s}(\lambda) + p_{|r-s|}(\lambda))\, d\hat{\pi}(\lambda) \\
&= \frac{1}{2}(\alpha_{r+s} + \alpha_{|r-s|}) \\
&= (P^TP)_{r+1,s+1}.
\end{aligned}$$

Note that we will compute the inner products, and exploit this "α" representation:

$$P^TP = \frac{1}{2}\begin{bmatrix} \alpha_0 & \alpha_1 & \alpha_2 & \dots & \alpha_{n-1} \\ \alpha_1 & \alpha_2 & \alpha_3 & \dots & \alpha_n \\ \alpha_2 & \alpha_3 & \alpha_4 & & \vdots \\ \vdots & \vdots & & \ddots & \alpha_{2n-3} \\ \alpha_{n-1} & \alpha_n & \dots & \alpha_{2n-3} & \alpha_{2n-2} \end{bmatrix} + \frac{1}{2}\begin{bmatrix} \alpha_0 & \alpha_1 & \alpha_2 & \dots & \alpha_{n-1} \\ \alpha_1 & \alpha_0 & \alpha_1 & \ddots & \vdots \\ \alpha_2 & \alpha_1 & \ddots & \ddots & \alpha_2 \\ \vdots & \ddots & \ddots & \alpha_0 & \alpha_1 \\ \alpha_{n-1} & \dots & \alpha_2 & \alpha_1 & \alpha_0 \end{bmatrix}.$$

$$= \frac{1}{2} < \text{Hankel} > + \frac{1}{2} < \text{Toeplitz} > .$$

Recall that any matrix that is constant along its diagonals is called a *Toeplitz Matrix*. Just as there are fast (i.e., $\mathcal{O}(n^2)$) solvers for Hankel Matrices, there are Fast Toeplitz solvers, and a recent development due to Heinig has yielded algorithms to solve Hankel+Toeplitz matrices in $\mathcal{O}(n^2)$ operations, without explicitly separating the two parts.

This yields a more stable computation of $\phi(\lambda)$. To compute the eigenvalues, we still need to find the roots of $\phi(\lambda)$. If we were to use Newton's method, we would need derivatives, so we would differentiate the recurrence relation as discussed in an earlier lecture.

13.2 Determining The Orthogonal Polynomials "On The Fly"

Since Chebyshev polynomials are not orthogonal with respect to $\hat{\pi}(\lambda)$, how would we construct a family of orthogonal polynomials "on the fly"? All orthogonal polynomials on the real line satisfy a 3 term recurrence relationship of the form:

$$p_{j+1}(\lambda) = (\lambda - \alpha_{j+1})p_j(\lambda) - \beta_j^2 p_{j-1}(\lambda) \tag{13.2}$$

where $p_{-1}(\lambda) = 0$ and $p_0(\lambda) = 1$. Furthermore, given any set of coefficients $\{\alpha_j, \beta_j\}$, there exists a measure on which $\{p_j(\lambda)\}$ forms an orthogonal family. And, for any measure, there exists a set of parameters $\{\alpha_j, \beta_j\}$ such that an orthogonal family can be constructed using (13.2).

Then, by plugging A into (13.2), and post-multiplying by a non-zero vector $\mathbf{p}_0$ with $\|\mathbf{p}_0\|_2 = 1$ as follows:

$$p_{j+1}(A)\mathbf{p}_0 = (A - \alpha_{j+1}I)p_j(A)\mathbf{p}_0 - \beta_j^2 p_{j-1}(A)\mathbf{p}_0,$$

we can generate a sequence of vectors $\mathbf{x}_j = p_j(A)\mathbf{x}_0$ with $\mathbf{x}_0 = \mathbf{p}_0$ and $\mathbf{x}_{-1} = \mathbf{0}$. Now, we can choose α_j and β_j such that this sequence of vectors are orthogonal for $j = 0, ..., n-1$.

We wish to constuct $\mathbf{x}_1$ so that it is orthogonal to $\mathbf{x}_0$. We have

$$\mathbf{x}_1 = (A - \alpha_1 I)\mathbf{x}_0$$

and by taking the inner product of both sides with $\mathbf{x}_0$, we obtain

$$\begin{aligned}\Rightarrow 0 = \mathbf{x}_0^T\mathbf{x}_1 &= \mathbf{x}_0^T(A-\alpha_1 I)\mathbf{x}_0\\ \Rightarrow \alpha_1 &= \frac{\mathbf{x}_0^T A\mathbf{x}_0}{\mathbf{x}_0^T\mathbf{x}_0}.\end{aligned}$$

We now have 2 vectors, and we can proceed by induction. Suppose we have $j+1$ mutually orthogonal vectors $\{\mathbf{x}_0, ..., \mathbf{x}_j\}$. Then,

$$\begin{aligned}\mathbf{x}_{j+1} &= (A-\alpha_{j+1}I)\mathbf{x}_j - \beta_j^2\mathbf{x}_{j-1}.\\ \mathbf{x}_j^T\mathbf{x}_{j+1} &= \mathbf{x}_j^T(A-\alpha_{j+1}I)\mathbf{x}_j - \beta_j^2\mathbf{x}_j^T\mathbf{x}_{j-1}\\ &= \mathbf{x}_j^T(A-\alpha_{j+1}I)\mathbf{x}_j.\\ \mathbf{x}_j^T\mathbf{x}_{j+1} &= 0\\ \Rightarrow \alpha_{j+1} &= \frac{\mathbf{x}_j^T A\mathbf{x}_j}{\mathbf{x}_j^T\mathbf{x}_j}.\end{aligned}$$

Similarly,

$$\begin{aligned}\mathbf{x}_{j-1}^T\mathbf{x}_{j+1} &= \mathbf{x}_{j-1}^T(A-\alpha_{j+1}I)\mathbf{x}_j - \beta_j^2\mathbf{x}_{j-1}^T\mathbf{x}_{j-1}\\ &= \mathbf{x}_{j-1}^T A\mathbf{x}_j - \beta_j^2\mathbf{x}_{j-1}^T\mathbf{x}_{j-1}.\\ \mathbf{x}_{j-1}^T\mathbf{x}_{j+1} &= 0\\ \Rightarrow \beta_j^2 &= \frac{\mathbf{x}_{j-1}^T A\mathbf{x}_j}{\mathbf{x}_{j-1}^T\mathbf{x}_{j-1}}.\end{aligned}$$

How do we know $\beta_j > 0$? Since A is symmetric,

$$\beta_j^2 = \frac{\mathbf{x}_{j-1}^T A\mathbf{x}_j}{\mathbf{x}_{j-1}^T\mathbf{x}_{j-1}} = \frac{\mathbf{x}_j^T A\mathbf{x}_{j-1}}{\mathbf{x}_{j-1}^T\mathbf{x}_{j-1}}.$$

But,

$$\begin{aligned}\mathbf{x}_j &= (A-\alpha_j I)\mathbf{x}_{j-1} - \beta^2\mathbf{x}_{j-2}.\\ \therefore \mathbf{x}_j^T\mathbf{x}_j &= \mathbf{x}_j^T A\mathbf{x}_{j-1}.\end{aligned}$$

So,

$$\beta_j^2 = \frac{\mathbf{x}_j^T\mathbf{x}_j}{\mathbf{x}_{j-1}^T\mathbf{x}_{j-1}}.$$

Now, we have constructed $\mathbf{x}_{j+1}$ so that it is orthogonal to $\mathbf{x}_j$ and $\mathbf{x}_{j-1}$. Is $\mathbf{x}_k^T\mathbf{x}_{j+1} = 0$ for $k < j-1$? We have

$$\begin{aligned}\mathbf{x}_k^T\mathbf{x}_{j+1} &= \mathbf{x}_k^T A\mathbf{x}_j - \alpha_{j+1}\mathbf{x}_k^T\mathbf{x}_j - \beta_j^2\mathbf{x}_k^T\mathbf{x}_{j-1}\\ &= \mathbf{x}_k^T A\mathbf{x}_j\\ &= \mathbf{x}_j^T A\mathbf{x}_k.\end{aligned}$$

But,

$$\begin{aligned} \mathbf{x}_{k+1} &= (A-\alpha_{k+1}I)\mathbf{x}_k - \beta_k^2\mathbf{x}_{k-1} \\ \Rightarrow \mathbf{x}_j^T\mathbf{x}_{k+1} &= \mathbf{x}_j^T A\mathbf{x}_k - \alpha_{k+1}\mathbf{x}_j^T\mathbf{x}_k - \beta_k^2\mathbf{x}_j^T\mathbf{x}_{k-1} \\ \text{i.e., } \mathbf{x}_j^T A\mathbf{x}_k &= 0. \end{aligned}$$

So, $\mathbf{x}_k^T\mathbf{x}_{j+1} = 0$ for $k = 0, 1, 2, ..., j$.

Algorithm 13.1. Generation of Orthogonal Vectors and Polynomials

```
Given $\mathbf{x}_0$.
$\alpha_1 = \frac{\mathbf{x}_0^T A\mathbf{x}_0}{\mathbf{x}_0^T\mathbf{x}_0}$.
$\mathbf{x}_1 = (A-\alpha_1)I\mathbf{x}_0$.
for $k = 1, \ldots, n$,
    $\alpha_{k+1} = \frac{\mathbf{x}_k^T A\mathbf{x}_k}{\mathbf{x}_k^T\mathbf{x}_k}$.
    $\beta_k^2 = \frac{\mathbf{x}_k^T\mathbf{x}_k}{\mathbf{x}_{k-1}^T\mathbf{x}_{k-1}}$.
    $\mathbf{x}_{k+1} = (A-\alpha_{k+1}I)\mathbf{x}_k - \beta_k^2\mathbf{x}_{k-1}$.
end
```

An alternative perspective on this computation is,

$$A\begin{bmatrix}\mathbf{x}_0, & \ldots, & \mathbf{x}_{n-1}\end{bmatrix} = \begin{bmatrix}\mathbf{x}_0, & \ldots, & \mathbf{x}_{n-1}\end{bmatrix}\begin{bmatrix} \alpha_1 & \beta_1^2 & 0 & \ldots & 0 \\ 1 & \alpha_2 & \beta_2^2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 1 & \alpha_{n-1} & \beta_{n-1}^2 \\ 0 & \ldots & 0 & 1 & \alpha_n \end{bmatrix}$$

(i.e., $AX = XJ$ or $A = XJX^{-1}$ where J is a tridiagonal matrix). Thus, finding the eigenvalues of A is the same as finding the eigenvalues of J. Furthermore, there exists a matrix, $D = \text{diag}(1, d_1, d_2, ..., d_{n-1})$ where $d_i = 1$ if $\beta_i^2 = 0$, $d_i = \frac{1}{\sqrt{\beta_i^2}}$ otherwise. Then, $K = D^{-1}JD$ where

$$K = \begin{bmatrix} \alpha_1 & \beta_1 & 0 & \ldots & 0 \\ \beta_1 & \alpha_2 & \beta_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \beta_{n-2} & \alpha_{n-1} & \beta_{n-1} \\ 0 & \ldots & 0 & \beta_{n-1} & \alpha_n \end{bmatrix}.$$

Thus, $A = QKQ^{-1}$. Furthermore, we can show that $Q^{-1} = Q^T$.

This algorithm requires storage of only 2 vectors at a time. It is optimal for large sparse matrices as only a matrix-vector multiplier is required. It yields the coefficients needed to define an orthogonal family of polynomials or it defines a tridiagonal matrix, that is similar to A. Hence, it is useful as a preliminary computation for QR iteration, or for yielding the characteristic polynomial to be used with Newton iterations, or some other root finding algorithm.

If we want eigenvectors, we need to store all of the vectors $\mathbf{x}_j$, as well as needing to find the eigenvectors of the tridiagonal matrix, K. Often, one performs this algorithm, and then uses a method to regenerate the $\mathbf{x}_j$ which were not stored. Secondly, this algorithm will break down if $\mathbf{x}_0$ is an eigenvector, since under these circumstances, α_1 is an eigenvalue, and then $\mathbf{x}_1 = \mathbf{0}$. This is not entirely bad, though, because now we can deflate our matrix and restart. Similarly, if $\mathbf{x}_0$ is in a subspace spanned by p eigenvectors, $\mathbf{x}_p = \mathbf{0}$ and we can again deflate.

Lecture 14

Lanczos Algorithm and Error Estimates

14.1 Steepest Descent

Let A be an $n \times n$ symmetric positive definite matrix, with eigenvalues

$$\lambda_1 \geq \cdots \geq \lambda_n > 0,$$

and corresponding eigenfunctions $\mathbf{z}_i$, $i = 1, \ldots, n$. We have seen that the *Rayleigh quotient*

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

achieves its maximum value at λ_1, and its minimum at λ_n.

This suggests a simple algorithm for finding these extremal eigenvalues. We choose an arbitrary starting vector $\mathbf{q}_1$, with $\|\mathbf{q}_1\|_2 = 1$, and then follow the direction of steepest ascent to obtain an improved approximation to λ_1, while following the direction of steepest descent to improve our approximation of λ_n. We can continue this process until our approximations converge to the desired eigenvalues.

From multivariable calculus, we know that the direction of steepest descent of $r(\mathbf{x})$ from $\mathbf{x}$ is $\nabla r(\mathbf{x})$, and the direction of steepest descent is $-\nabla r(\mathbf{x})$. Therefore, we can refine our algorithm as follows:

```
Choose arbitary $\mathbf{q}_1$ so that $\|\mathbf{q}_1\|_2 = 1$
**for** $k = 1, 2, \ldots$
    $Q_k = \begin{bmatrix} \mathbf{q}_1 & \cdots & \mathbf{q}_k \end{bmatrix}$
    Let $T_k = Q_k^T A Q_k$
    Let $T_k = S_k M_k S_k^T$ be the Schur decomposition of $T_k$
        with $\operatorname{diag}(M_k) = (\theta_1, \ldots, \theta_k)$, $\theta_1 \geq \cdots \geq \theta_k$
    $\mathbf{u}_k = Q_k S_k \mathbf{e}_1$
    $\mathbf{v}_k = Q_k S_k \mathbf{e}_k$
    Choose $\mathbf{q}_{k+1}$ orthogonal to $\{\mathbf{q}_1, \ldots, \mathbf{q}_k\}$ so that
        $\text{span}(\mathbf{q}_1, \ldots, \mathbf{q}_{k+1}) = \text{span}(\mathbf{q}_1, \ldots, \mathbf{q}_k, \nabla r(\mathbf{u}_k))$ and
        $\text{span}(\mathbf{q}_1, \ldots, \mathbf{q}_{k+1}) = \text{span}(\mathbf{q}_1, \ldots, \mathbf{q}_k, \nabla r(\mathbf{v}_k))$
end
```

Since each approximation M_k to λ_1 and m_k to λ_n is obtained by taking the maximum and minimum values of $r(\mathbf{x})$ over a larger subspace, clearly these approximations converge to λ_1 and λ_n, respectively, while the vectors $\mathbf{u}_k$ and $\mathbf{v}_k$ converge to the corresponding eigenvectors.

It may seem impossible to expand the subspace spanned by the columns of Q_k as prescribed in the above algorithm, but it is actually quite simple to do so, since

$$\nabla r(\mathbf{x}) = \frac{2}{\mathbf{x}^T\mathbf{x}}(A\mathbf{x} - r(\mathbf{x})\mathbf{x}).$$

Therefore $\nabla r(\mathbf{x}) \in \text{span}(\mathbf{x}, A\mathbf{x})$. It follows that for each k, the columns of Q_k span the Krylov subspace

$$\mathcal{K}(A, \mathbf{q}_1, k) = \text{span}(\mathbf{q}_1, A\mathbf{q}_1, \ldots, A^{k-1}\mathbf{q}_1).$$

Let

$$K(A, \mathbf{q}_1, k) = \left[\begin{array}{ll} \mathbf{q}_1 & A\mathbf{q}_1, \ldots, A^{k-1}\mathbf{q}_1 \end{array}\right].$$

Then Q_k can be constructed by orthogonalizing the columns of $K(A, \mathbf{q}_1, k)$. This could be accomplished by computing the QR factorization, but this is not practical because the columns of $K(A, \mathbf{q}_1, k)$ need to be computed up front, and these columns tend to become linearly dependent as k increases, as we saw when we discussed the power method. The Lanczos algorithm is an alternative approach that computes the columns of Q_k on the fly.

Let

$$K(A, \mathbf{q}_1, k) = Q_k R_k$$

be the QR factorization of $K(A, \mathbf{q}_1, k)$. Then,

$$Q_k^T A K(A, \mathbf{q}_1, k) = H_k$$

is an upper Hessenberg matrix. It follows from

$$Q_n^T A Q_n Q_n^T K(A, \mathbf{q}_1, k) = (Q_n^T A Q_n) R_k = H_k$$

that $T_n = Q_n^T A Q_n$ is Hessenberg, and, due to its symmetry, is in fact tridigagonal. We will see that the Lanczos algorithm computes the entries of T_n and the columns of Q_n simultaneously.

14.2 The Lanczos Algorithm

Let A be a symmetric matrix. Then A can be reduced to a symmetric tridiagonal matrix via orthogonal similarity transformations as we have shown in previous lectures. Hence $A = Q_n T_n Q_n^T$, and

$$T_n = \begin{bmatrix} \alpha_1 & \beta_1 & 0 & \dots & 0 \\ \beta_1 & \alpha_2 & \beta_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \beta_{n-2} & \alpha_{n-1} & \beta_{n-1} \\ 0 & \dots & 0 & \beta_{n-1} & \alpha_n \end{bmatrix}.$$

Then, $AQ = QT$ implies

$$\begin{aligned} A\mathbf{q}_j &= \beta_{j-1}\mathbf{q}_{j-1} + \alpha_j \mathbf{q}_j + \beta_j \mathbf{q}_{j+1}. \\ \therefore \qquad \beta_j \mathbf{q}_{j+1} &= A\mathbf{q}_j - \beta_{j-1}\mathbf{q}_{j-1} - \alpha_j \mathbf{q}_j, \\ \Rightarrow \qquad |\beta_j| &= \|A\mathbf{q}_j - \beta_{j-1}\mathbf{q}_{j-1} - \alpha_j \mathbf{q}_j\|_2, \\ \text{and,} \qquad \mathbf{q}_j &= A\mathbf{q}_j - \beta_{j-1}\mathbf{q}_{j-1} - \alpha_j \mathbf{q}_j. \end{aligned}$$

Which suggests the following algorithm:

Algorithm 14.1. (Lanczos Algorithm)

```
Initially, $\mathbf{r}_0 = \mathbf{q}_1$ such that $\|\mathbf{q}_1\|_2 = 1$, $\beta_0 = 1$, $\mathbf{q}_0 = \mathbf{0}$, and $k = 0$.
while $\beta_k \neq 0$
    $\mathbf{q}_{k+1} = \mathbf{r}_k / \beta_k$.
    $k = k + 1$.
    $\alpha_k = \mathbf{q}_k^T A \mathbf{q}_k$.
    $\mathbf{r}_k = (A - \alpha_k I)\mathbf{q}_k - \beta_{k-1}\mathbf{q}_{k-1}$.
    $\beta_k = \|\mathbf{r}_k\|_2$.
end
```

Notice that if A is positive definite, each $\alpha_k > 0$. Also, if $\lambda_1 = ... = \lambda_p$, then there exists $\beta_{i_1} = ... = \beta_{i_{p-1}} = 0$, and the basic algorithm breaks down. However, this only means that we can deflate our matrix, and restart with a smaller one.

14.3 A Posteriori Error Estimate

Suppose we have performed k steps of the Lanczos Algorithm. Then, let $Q_k = \begin{bmatrix} \mathbf{q}_1, & \dots & \mathbf{q}_k \end{bmatrix}$, $Q_k^T Q_k = I$ and

$$AQ_k = Q_k T_k + \mathbf{r}_k \mathbf{e}_k^T$$

where T_k is the leading $k \times k$ block of T. Let $T_k = S_k M_k S_k^T$ be the spectral decomposition of T_k. The diagonal elements $\mu_1, \dots, \mu_k$ of M_k are known as the *Ritz values*. Then, the columns of $Y_k = Q_k S_k = \begin{bmatrix} \mathbf{y}_1, & \dots, & \mathbf{y}_k \end{bmatrix}$, known as *Ritz vectors*, should approximate k eigenvectors of A. Since $\|\mathbf{y}_j\|_2 = 1$, our analysis earlier in the quarter showed that the interval

$$|\lambda - \mu_j| \le \gamma_j \equiv \|A\mathbf{y}_j - \mu_j \mathbf{y}_j\|_2$$

contains an eigenvalue of A. Furthermore, one can show that this is an exclusive interval as well (i.e., there exists at least one eigenvalue outside of the interval also).

Also,

$$\begin{aligned} & & AQ_k &= Q_k T_k + \mathbf{r}_k \mathbf{e}_k^T. \\ &\Rightarrow & AQ_k S_k &= Q_k T_k S_k + \mathbf{r}_k \mathbf{e}_k^T S_k \\ & & &= Q_k S_k M_k + \mathbf{r}_k \mathbf{e}_k^T S_k, \\ & & AY_k &= Y_k M_k + \mathbf{r}_k \mathbf{e}_k^T S_k. \\ &\Rightarrow & A\mathbf{y}_j &= \mu_j \mathbf{y}_j + (\mathbf{e}_k^T S \mathbf{e}_j)\mathbf{r}_k. \\ &\therefore & \gamma_j &= \|A\mathbf{y}_j - \mu_j \mathbf{y}_j\|_2 \\ & & &= \|\mathbf{r}_k\|_2 |s_{kj}| \\ & & &= |\beta_k||s_{kj}| \end{aligned}$$

since $\mathbf{q}_{k+1} = \mathbf{r}_k / \beta_k$ and $\|\mathbf{q}_k\|_2 = 1$. Notice also that s_{kj} is the j^{th} element in the bottom row of S_k (i.e., the last element in the j^{th} eigenvector). Thus,

$$|\lambda - \mu_j| \le |\beta_k||s_{kj}|.$$

Thus, if k eigenvalues are desired, we can quickly check the last element of the eigenvectors to avoid performing too many iterations.

Suppose we want the smallest eigenvalue, and $\mu_1 \le \mu_2 \le ... \le \mu_k$, after k steps. If we aren't satisfied with our error bounds, we can restart with $\mathbf{q}_1 = \mathbf{y}_1$. To see this, let $\eta = Q_k \xi$. Then,

$$\min_{\eta \ne \mathbf{0}} \frac{\eta^T A \eta}{\eta^T \eta} = \min_{\xi \ne \mathbf{0}} \frac{\xi^T Q_k^T A Q_k \xi}{\xi^T \xi} = \mu_1$$

because $Q_k^T A Q_k = T_k$. Hence using this as the restart vector is akin to applying Lanczos in a steepest descent mode. Along a similar line, Hestenes and Karusch, and earlier still by Kantarovich, analyzed the following scheme:

Algorithm 14.2. Steepest Descent for the Eigenvalue Problem

```
Given an initial vector $\mathbf{x}_1$ with $\|\mathbf{x}_1\|_2 = 1$.
while unconverged
```

```
    $\theta_k = \mathbf{x}_k^T A\mathbf{x}_k$.
    $\mathbf{z}_k = A\mathbf{x}_k - \theta_k\mathbf{x}_k$.
    $\mathbf{y}_{k+1} = a\mathbf{x}_k + b\mathbf{x}_k$ where $\begin{bmatrix} a \\ b \end{bmatrix}$ is the eigenvector of $\left(\begin{bmatrix} \mathbf{x}_k^T \\ \mathbf{z}_k^T \end{bmatrix} A \begin{bmatrix} \mathbf{x}_k, & \mathbf{z}_k \end{bmatrix}\right)$
        that corresponds to its smallest eigenvalue.
    $\mathbf{x}_{k=1} = \mathbf{y}_{k+1}/\|\mathbf{y}_{k+1}\|_2$.
end
```

14.4 Kaniel-Paige Convergence Theory

In the previous section, we obtained error estimates for the Ritz values of T_k, but we still know nothing about the rate of convergence. The following theorem, from *Kaniel-Paige theory*, sheds light on this aspect of the Lanczos algorithm.

Theorem 14.1 Let A be an $n \times n$ symmetric matrix with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$ and corresponding orthonormal eigenvectors $\mathbf{z}_1, \ldots, \mathbf{z}_n$. If $\theta_1 \geq \cdots \geq \theta_k$ are the eigenvalues of the matrix T_k obtained after k steps of the Lanczos iteration, then

$$\lambda_1 \geq \theta_1 \geq \lambda_1 - (\lambda_1 - \lambda_n)\frac{\tan(\phi_1)^2}{(c_{k-1}(1 + 2\rho_1))^2}$$

where $\cos(\phi_1) = |\mathbf{q}_1^T\mathbf{z}_1|$, $\rho_1 = (\lambda_1 - \lambda_2)/(\lambda_2 - \lambda_n)$, and $c_{k-1}(x)$ is the Chebyshev polynomial of degree $k-1$.

Proof Since θ_1 is the largest eigenvalue of T_k, and each Lanczos vector $\mathbf{q}_k$ is of the form $\mathbf{q}_k = p_{k-1}(A)\mathbf{q}_1$, where p_k is a polynomial of degree k, it follows that

$$\begin{aligned} \theta_1 &= \max_{\mathbf{y}\neq\mathbf{0}} \frac{\mathbf{y}^T T_k \mathbf{y}}{\mathbf{y}^T\mathbf{y}} \\ &= \max_{\mathbf{y}\neq\mathbf{0}} \frac{\mathbf{y}^T Q_k^T A Q_k \mathbf{y}}{\mathbf{y}^T Q_k^T Q_k \mathbf{y}} \\ &= \max_{p\in\mathcal{P}_{k-1}} \frac{\mathbf{q}_1^T p(A) A p(A)\mathbf{q}_1}{\mathbf{q}_1^T p(A)^2 \mathbf{q}_1} \end{aligned}$$

where $\mathcal{P}_{k-1}$ is the space of all polynomials of degree at most $k-1$. Clearly, $\theta_1 = r(Q_k\mathbf{y})$ for some $\mathbf{y}$, so $\theta_1 \leq \lambda_1$. To establish the lower bound on θ_1, we write $\mathbf{q}_1 = \sum_{i=1}^n d_i \mathbf{z}_i$, and obtain

$$
\begin{aligned}
\frac{\mathbf{q}_1^T p(A) A p(A) \mathbf{q}_1}{\mathbf{q}_1^T p(A)^2 \mathbf{q}_1} &= \frac{\sum_{i=1}^n d_i^2 p(\lambda_i)^2 \lambda_i}{\sum_{i=1}^n d_i^2 p(\lambda_i)^2} \\
&= \frac{\sum_{i=1}^n d_i^2 p(\lambda_i)^2 (\lambda_i + \lambda_1 - \lambda_1)}{\sum_{i=1}^n d_i^2 p(\lambda_i)^2} \\
&= \lambda_1 - \frac{\sum_{i=2}^n d_i^2 p(\lambda_i)^2 (\lambda_1 - \lambda_i)}{\sum_{i=1}^n d_i^2 p(\lambda_i)^2} \\
&\geq \lambda_1 - (\lambda_1 - \lambda_n) \frac{\sum_{i=2}^n d_i^2 p(\lambda_i)^2}{d_1^2 p(\lambda_1)^2 + \sum_{i=2}^n d_i^2 p(\lambda_i)^2}
\end{aligned}
$$

We can obtain a sharp bound by choosing

$$
p(x) = c_{k-1}\left(1 + 2\frac{x - \lambda_n}{\lambda_2 - \lambda_n}\right).
$$

The argument to $c_{k-1}(x)$ is constructed to map the interval $[\lambda_n, \lambda_2]$ to $[-1, 1]$. Since $|c_{k-1}(x)| \leq 1$ on $[-1, 1]$ and grows vary rapidly outside this interval, it follows that $p(x)$ is large at λ_1 and small at the other eigenvalues. Therefore, using the fact that $d_1 = \mathbf{q}_1^T \mathbf{z}_1$, we obtain

$$
\begin{aligned}
\theta_1 &= \max_{p \in \mathcal{P}_{k-1}} \frac{\mathbf{q}_1^T p(A) A p(A) \mathbf{q}_1}{\mathbf{q}_1^T p(A)^2 \mathbf{q}_1} \\
&\geq \lambda_1 - (\lambda_1 - \lambda_n) \frac{\sum_{i=2}^n d_i^2}{d_1^2 c_{k-1}(1 + 2\rho_1)^2} \\
&\geq \lambda_1 - (\lambda_1 - \lambda_n) \frac{1 - d_1^2}{d_1^2} \frac{1}{c_{k-1}(1 + 2\rho_1)^2} \\
&\geq \lambda_1 - (\lambda_1 - \lambda_n) \frac{\tan(\phi_1)^2}{c_{k-1}(1 + 2\rho_1)^2}
\end{aligned}
$$

□

Similar bounds can be established for θ_k, regarding its rate of convergence to λ_n.

14.5 Block Lanczos

Often one wishes to use several vectors at a time with the Lanczos procedure. The previous algorithm can be easily modified to accomodate this. Suppose we started with p initial orthonormal vectors as the columns of Q_1. We will construct the decomposition $A = QJQ^T$, where $Q = \begin{bmatrix} Q_1, & \dots, & Q_r \end{bmatrix}$ and

Q is orthogonal, and J is block tridiagonal, namely

$$J = \begin{bmatrix} A_1 & B_1 & 0 & \dots & 0 \\ B_1^T & A_2 & B_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & B_{r-2}^T & A_{r-1} & B_{r-1} \\ 0 & \dots & 0 & B_{r-1}^T & A_r \end{bmatrix}.$$

Note that $n = p \cdot r$ and each sub-block within J is $p \times p$.

$$\begin{aligned} A &= QJQ^T \\ AQ &= QJ \\ AQ_1 &= Q_1A_1 + Q_2B_1^T \\ \Rightarrow Q_1^TAQ_1 &= A_1Q_1^TQ_2B_1^T \\ &= A_1 \end{aligned}$$

assuming the columns of Q_2 are orthonormal, and orthogonal to those of Q_1. Then,

$$\begin{aligned} Q_2B_1^T &= AQ_1 - Q_1A_1 \equiv Z_1. \\ \therefore Q_1^TZ_1 &= Q_1^TAQ_1 - A = 0. \end{aligned}$$

So, the columns of Z_1 are orthogonal to those of Q_1. Furthermore,

$$\begin{aligned} Z_1^TZ_1 &= B_1Q_2^TQ_2B_1^T \\ &= B_1B_1^T. \end{aligned}$$

So, one possible way to construct B_1 would be via a Cholesky Factorization of $Z_1^TZ_1$, hence $Q_2 = Z_1B_1^{-T}$, but this is a bad choice. Instead, we will compute the QR factorization of Z_1, and let the transpose of the R factor be B_1, hence B_1 is lower triangular, and $Z_1 = Q_2B_1^T$.

We now have constructed A_1, a lower triangular B_1, and Q_1 (which was given) and Q_2, reducing A to the form

$$\begin{bmatrix} A_1 & B_1 & 0 & \dots & 0 \\ B_1^T & \times & \times & \dots & \times \\ 0 & \times & \times & \dots & \times \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \times & \times & \dots & \times \end{bmatrix}.$$

We shall now proceed inductively. After $j-1$ steps we have constructed $A_1, ..., A_{j-1}$, lower triangular matrices $B_1, ..., B_{j-1}$, and matrices with mutually orthonormal columns, $Q_1, ..., Q_j$. Then,

$$AQ_j = Q_{j-1}B_{j-1} + Q_jA_j + Q_{j+1}B_j^T.$$

So,

$$\begin{aligned}Q_j^T Q_{j+1} B_j^T &= Q_j^T A Q_j - Q_j^T Q_j A_j - Q_j^T Q_{j-1} B_{j-1}\\ &= Q_j^T A Q_j - A_j.\\ Z_j &\equiv A Q_j - Q_j A_j - Q_{j-1} B_{j-1}\\ &= Q_{j+1} B_j^T.\end{aligned}$$

Then, we construct Q_{j+1} and B_j from the QR factorization of Z_j, and define B_j to be the transpose of the R factor, as before. It can be shown that all orthogonalities for Q_{j+1} are maintained as necesary, and so $A_j = Q_j^T A Q_j$.

We have not only constructed a block tridiagonal matrix, but it has upper triangular blocks along the sub-diagonal, and lower triangular blocks along the super-diagonal. Thus, the resulting J matrix is banded, with bandwidth $2p+1$.

14.6 More Error Bounds

We can generalize the Kaniel-Paige theory to determine error bounds based on the block Lanczos procedure. After t steps, we have

$$J_t = \begin{bmatrix} A_1 & B_1 & 0 & \dots & 0 \\ B_1^T & A_2 & B_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & B_{t-2}^T & A_{t-1} & B_{t-1} \\ 0 & \dots & 0 & B_{t-1}^T & A_t \end{bmatrix}.$$

Suppose A has eigenvalues $\lambda_1 \geq \dots \geq \lambda_n$ with a complete set of corresponding eigenvectors $\mathbf{z}_1, ..., \mathbf{z}_n$, and $\lambda_p > \lambda_{p+1}$ (recall, p initial vectors are used to define Q_1). Furthermore, suppose J_t has eigenvalues $\mu_1, ..., \mu_{tp}$. Then $A = Z\Lambda Z^T$, and $W = Z^T Q_1 = \begin{bmatrix} W_1 \\ W_2 \end{bmatrix}$ where W_1 is a $p \times p$ block. If W_1 is nonsingular, then $\sigma_{min}(W_1) > 0$ and

$$\lambda_k \geq \mu_k \geq \lambda_k - \varepsilon_k^2$$

where

$$\varepsilon_k^2 = \frac{(\lambda_1 - \lambda_k)\tan^2\theta_k}{T_{t-1}^2(\frac{1+\nu_k}{1-\nu_k})}.$$

and $\theta_k = \cos^{-1}(\sigma_{min}(W_1))$, $\nu_k = \frac{\lambda_k - \lambda_{p+1}}{\lambda_k - \lambda_1}$, and $T_s(\nu)$ is the s^{th} Chebyshev polynomial of the first kind. As mentioned in the previous lecture, these polynomials are the smallest polynomials inside the interval $[-1, 1]$, and increase the fastest outside the interval. Thus, the denominator has a term which is potentially quite large.

For example, if A is a 1000×1000 matrix, and we choose 2 initial vectors to define Q_1 (i.e., $p = 2$), and these vectors yield $\sigma_{min}(W_1) = 0.4$, and if the eigenvalues of A are ordered with $\lambda_1 = 0$, $\lambda_2 = 0.1$, $\lambda_3 = 0.5$, and $\lambda_{1000} = 1$, then after 10 steps of Block Lanczos (i.e., $t = 10$),

$$\lambda_1 \leq \mu_1 \leq \lambda_1 + 2.6 \times 10^{-8},$$
$$\lambda_2 \leq \mu_2 \leq \lambda_2 + 3.6 \times 10^{-7}.$$

It should be noted however that the spacing between the second and third eigenvalues is rather large. Typically, if A is the result of a discretized differential operator, eigenvalues will be be clustered together, so such a large separation between the second and third eigenvalues is unlikely. If a sequence of problems are to be solved, and the eigenvalues do not change very much from one problem to the next, this set of bounds could prove useful in obtaining estimates for the number of Lanczos iterations required, and good block sizes to use. Typically, a small p is chosen, such as 2 or 3.

Lecture 15

Lanczos Technical Details and Interlaced Eigenvalues

15.1 Why Study Lanczos?

The Lanczos algorithm applied to a symmetric matrix generates a sequence of orthonormal vectors $\{\mathbf{q}_1, ..., \mathbf{q}_k\}$ after k steps. These vectors serve to tridiagonalize A in such a manner that

$$AQ_k = Q_k T_k + \mathbf{r}_k \mathbf{e}_k^T,$$

where $Q_k = \begin{bmatrix} \mathbf{q}_1 & \cdots & \mathbf{q}_k \end{bmatrix}$ and

$$T_k = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{k-2} & \alpha_{k-1} & \beta_{k-1} \\ & & & \beta_{k-1} & \alpha_k \end{bmatrix}.$$

Why should we study the Lanczos algorithm?

- In its basic form, without worrying about several technical details, it is easy to implement. More importantly, it is especially convenient for large sparse matrices as it only requires the matrix A for matrix-vector multiplication.
- At each intermediate step it yields a symmetric, tridiagonal matrix, T_k, whose eigenvalues are easy to calculate.
- The eigenvalues of T_k approximate the eigenvalues of A.
- Lanczos is the optimal Krylov Subspace method in the absence of roundoff errors.

This last statement deserves some further commentary. In Lanczos, given any vector of unit length $\mathbf{q}_1$, and assuming $\beta_0 = 0$ and $\mathbf{q}_0 = \mathbf{0}$, we generate a sequence of $\{\alpha_k\}$, $\{\beta_k\}$, and$\{\mathbf{r}_k\}$, where

$$\mathbf{r}_k = (A - \alpha_k I)\mathbf{q}_k - \beta_{k-1}\mathbf{q}_{k-1},$$

and $\mathbf{q}_{k+1} = \frac{1}{\beta_k}\mathbf{r}_k$. Thus,

$$\begin{aligned} \beta_1\mathbf{q}_2 &= \mathbf{r}_1 = (A - \alpha_1 I)\mathbf{q}_1 \\ &= p_1(A)\mathbf{q}_1. \\ \beta_2\mathbf{q}_3 &= \frac{1}{\beta_1}(A - \alpha_2 I)(A - \alpha_1 I)\mathbf{q}_2 - \beta_1\mathbf{q}_1. \\ &= p_2(A)\mathbf{q}_1. \\ \Rightarrow \qquad \mathbf{q}_k &= p_{k-1}(A)\mathbf{q}_1 \end{aligned}$$

Thus, the vectors $\{\mathbf{q}_1, ..., \mathbf{q}_k\}$ span the same subspace as the Krylov subspace $\{\mathbf{q}_1, A\mathbf{q}_1, ..., A^{k-1}\mathbf{q}_1\}$. In other words, any vector which can be expressed as a linear combination of the Lanczos vectors $\{\mathbf{q}_j\}$ can also be expressed as a linear combination of the Krylov subspace vectors $\{A^{j-1}\mathbf{q}_1\}$. So,

$$\begin{aligned} \lambda_{\max}(A) &= \max_{\mathbf{x}\neq\mathbf{0}} \frac{\mathbf{x}^T A\mathbf{x}}{\mathbf{x}^T\mathbf{x}} \\ &\leq \max_{\mathbf{c}\neq\mathbf{0}} \frac{\mathbf{c}^T Q_k^T A Q_k\mathbf{c}}{\mathbf{c}^T Q_k^T Q_k\mathbf{c}} \\ &\leq \max_{\mathbf{c}\neq\mathbf{0}} \frac{\mathbf{c}^T T_k\mathbf{c}}{\mathbf{c}^T\mathbf{c}} \\ &\leq \lambda_{\max}(T_k). \end{aligned} \tag{15.1}$$

This is the sense in which Lanczos is optimal. The largest eigenvalue of T_k is a maximum of Rayleigh quotients over all vectors within the k^{th} Krylov subspace generated by A and $\mathbf{q}_1$. The fact that this maximum equals the largest eigenvalue of T_k tells us that there is no other Krylov subspace method which will yield a better estimate for the largest eigenvalue of A than the largest eigenvalue of T_k after k steps, with this starting vector. Given a good choice of $\mathbf{q}_1$, we will get excellent convergence to the largest eigenvalue. But this dependence on the starting vector can be a downfall of the method as well, as a bad choice of $\mathbf{q}_1$ will require many iterations before a reasonable estimate can be obtained.

Unfortunately, there are numerical difficulties associated with the robustness of the algorithm. As we shall see later, this is related to a loss of orthogonality of the vectors $\{\mathbf{q}_k\}$. As a result, the computed eigenvalues of T_n can be quite different from the eigenvalues of A, despite the fact that they should be identical.

15.2 Interlaced Eigenvalues for Rank 1 Updates

Often one would want to compute eigenvalues of a rank one modification of the symmetric matrix A, namely, the eigenvalues of

$$A^{(1)} = A + \mathbf{v}\mathbf{v}^T.$$

Let us call the eigenpairs of A, $(\lambda_1, \mathbf{u}_1), ..., (\lambda_n, \mathbf{u}_n)$, and the eigenpairs of $A^{(1)}$, $(\mu_1, \mathbf{x}_1), ..., (\mu_n, \mathbf{x}_n)$. Then, if $\mathbf{x}$ is an eigenvector of $A^{(1)}$,

$$A^{(1)}\mathbf{x} = (A + \mathbf{v}\mathbf{v}^T)\mathbf{x} = \mu\mathbf{x}.$$

This implies that

$$\begin{aligned} & (A - \mu I)\mathbf{x} = -\mathbf{v}\mathbf{v}^T\mathbf{x}. \\ \Rightarrow \quad & \mathbf{x} = -(A - \mu I)^{-1}\mathbf{v}\mathbf{v}^T\mathbf{x}. \\ \Rightarrow \quad & \mathbf{v}^T\mathbf{x} + \mathbf{v}^T(A - \mu I)^{-1}\mathbf{v}\mathbf{v}^T\mathbf{x} = 0. \\ \Rightarrow \quad & (\mathbf{v}^T\mathbf{x})(1 + \mathbf{v}^T(A - \mu I)^{-1}\mathbf{v}) = 0. \end{aligned}$$

If $\mathbf{v}$ is an eigenvector of $A^{(1)}$ corresponding to an eigenvalue other than μ, then $\mathbf{v}^T\mathbf{x} = 0$, but we shall assume that this is not the case. Then we can define the function

$$\begin{aligned} \phi(\mu) &= 1 + \mathbf{v}^T(A - \mu I)^{-1}\mathbf{v} \\ &= 1 + \mathbf{v}^T(U\Lambda U^T - \mu I)^{-1}\mathbf{v} \\ &= 1 + \mathbf{v}^T(U(\Lambda - \mu I)U^T)^{-1}\mathbf{v} \\ &= 1 + \mathbf{v}^T U(\Lambda - \mu I)^{-1}U^T\mathbf{v} \\ &= 1 + \mathbf{w}^T(\Lambda - \mu I)^{-1}\mathbf{w} \\ &= 1 + \sum_{i=1}^{n} \frac{w_i^2}{\lambda_i - \mu}. \end{aligned}$$

where $w_i = \mathbf{e}_i^T U^T\mathbf{v}$. Therefore, finding the zeros of $\phi(\mu)$ is equivalent to finding the eigenvalues of $A^{(1)}$. If we make the assumption $\lambda_1 > \lambda_2 > \cdots > \lambda_n > 0$, then we can easily sketch $\phi(\mu)$. Clearly, as μ approaches each λ_i from the left, $\phi(\mu)$ increases toward infinity. As μ approaches each λ_i from the right, $\phi(\mu)$ decreases toward $-\infty$. Finally, as μ tends toward $\pm\infty$, $\phi(\mu)$ tends toward 1. A simple sketch of this behavior is shown in Figure 15.1.

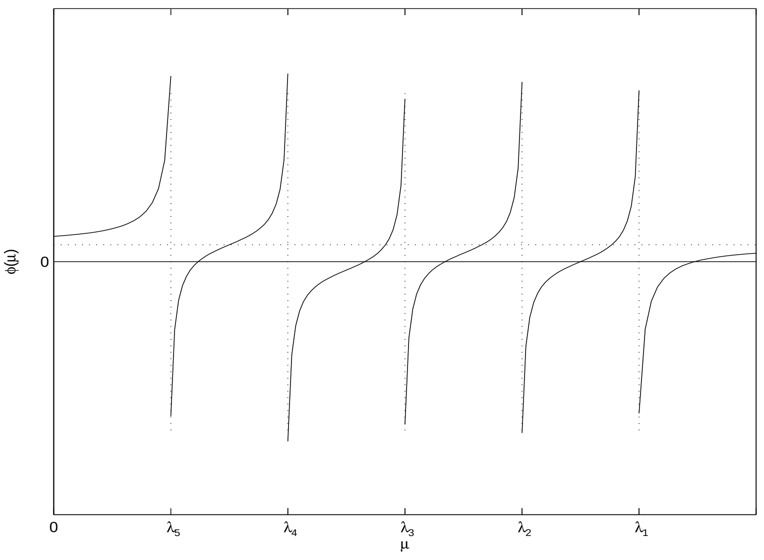

Figure 15.1: Sketch of $\phi(\mu)$ for $A_{5\times 5}$.

We can place an upper bound on μ_1 as follows:

$$\begin{aligned}
\mu_1 &= \frac{\mathbf{x}_1^T(A+\mathbf{v}\mathbf{v}^T)\mathbf{x}_1}{\mathbf{x}_1^T\mathbf{x}_1} \\
&= \max_{\mathbf{x}\neq\mathbf{0}} \frac{\mathbf{x}^T(A+\mathbf{v}\mathbf{v}^T)\mathbf{x}}{\mathbf{x}^T\mathbf{x}} \\
&\leq \max_{\mathbf{x}\neq\mathbf{0}} \frac{\mathbf{x}^TA\mathbf{x}}{\mathbf{x}^T\mathbf{x}} + \max_{\mathbf{x}\neq\mathbf{0}} \frac{(\mathbf{x}^T\mathbf{v})^2}{\|\mathbf{x}\|_2^2} \\
&\leq \lambda_1 + \|\mathbf{v}\|_2^2,
\end{aligned}$$

with the last inequality resulting from the Cauchy-Schwartz Inequality. For example, the eigenvalues of

$$A = \begin{bmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}, \text{and } A^{(1)} = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}$$

interlace since $A^{(1)} = A + \mathbf{e}_1\mathbf{e}_1^T$.

Sometimes $\phi(\mu)$ is used on parallel architectures as part of a divide and conquer scheme, developed by Cuppen. Given the matrix T_{2n}, we can partition the matrix using the concept of a rank 1 update:

$$T_{2n} = \left[\begin{array}{cccc|cccc}
a_1 & b_1 & & & & & & \\
b_1 & a_2 & b_2 & & & & & \\
 & \ddots & \ddots & \ddots & & & & \\
 & & b_{n-1} & a_n & b_n & & & \\ \hline
 & & & b_n & a_{n+1} & b_{n+1} & & \\
 & & & & \ddots & \ddots & \ddots & \\
 & & & & & b_{2n-2} & a_{2n-1} & b_{2n-1} \\
 & & & & & & b_{2n-1} & a_{2n}
\end{array}\right]$$

$$= \left[\begin{array}{cccc|cccc}
a_1 & b_1 & & & & & & \\
b_1 & a_2 & b_2 & & & & & \\
 & \ddots & \ddots & \ddots & & & & \\
 & & b_{n-1} & a_n - b_n & 0 & & & \\ \hline
 & & & 0 & a_{n+1} - b_n & b_{n+1} & & \\
 & & & & \ddots & \ddots & \ddots & \\
 & & & & & b_{2n-2} & a_{2n-1} & b_{2n-1} \\
 & & & & & & b_{2n-1} & a_{2n}
\end{array}\right]$$

$$+ \left[\begin{array}{cc|cc}
 & & & \\
 & b_n & b_n & \\ \hline
 & b_n & b_n & \\
 & & &
\end{array}\right]$$

$$= \left[\begin{array}{c|c}
T_n^{(1)} & 0 \\ \hline
0 & T_{n'}^{(1)}
\end{array}\right] + b_n(\mathbf{e}_n + \mathbf{e}_{n+1})(\mathbf{e}_n + \mathbf{e}_{n+1})^T.$$

Then, we could recurse downward, applying the same trick to $T_n^{(1)}$ and $T_{n'}^{(1)}$, until 2×2 blocks remain. Then, eigenvalues can be found using $\phi(\mu)$ and perhaps Newton's method.

15.3 Interlaced Eigenvalues and Augmented Matrices

Suppose we take a symmetric matrix A and augment it so that

$$A^{(1)} = \begin{bmatrix} A & \mathbf{b} \\ \mathbf{b}^T & c \end{bmatrix}.$$

How do the eigenvalues of $A^{(1)}$ relate to those of A?

The system of equations that defines the eigenvalue problem tells us that

$$(A - \mu I)\mathbf{x} + \mathbf{b}\xi = 0, \tag{15.2}$$
$$\mathbf{b}^T\mathbf{x} + (c - \mu)\xi = 0. \tag{15.3}$$

Solving (15.2) for $\mathbf{x}$ and plugging the result into (15.3) yields

$$\mathbf{x} = -(A - \mu I)^{-1}\mathbf{b}\xi,$$
$$\big(- \mathbf{b}^T(A - \mu I)^{-1}\mathbf{b} + (c - \mu)\big)\xi = 0.$$

As in the previous section, if $\mathbf{x}$ is not the eigenvector of A corresponding to the eigenvalue μ, then $\xi \neq 0$, so then finding the zeros of $\phi(\mu)$ where

$$\begin{aligned}\phi(\mu) &= (c - \mu) - \beta^T(\Lambda - \mu I)^{-1}\beta \\ &= c - \mu - \sum_{i=1}^{n} \frac{\beta^2}{\lambda_i - \mu}\end{aligned}$$

yields the eigenvalues. As before, we can generate a quick sketch of this function and see that

$$\mu_1 > \lambda_1 > \mu_2 > ... > \lambda_n > \mu_{n+1}$$

under the same assumptions as before.

Clearly, as you increase the dimension of the matrix, the largest eigenvalue of the augmented matrix is larger than the largest eigenvalue of the original matrix. Similarly, the smallest of the augmented matrix is smaller than the smallest of the original matrix.

15.4 The Breakdown of Lanczos

Returning to the discussion of Lanczos, the eigenvalues of T_k do not exactly interlace the eigenvalues of A; after all, there are only k eigenvalues of T_k while A has n. However, we can show that the smallest eigenvalue of A is smaller than the smallest eigenvalue of T_k, and we have already seen that the largest eigenvalue of A is larger than that of T_k. But Kaniel and Paige proved some bounds that are enlightening.

Let $\{\lambda_i\}$ denote the set of eigenvalues of A, and $\{\mu_i\}$ denote the set of eigenvalues of T_k. Furthermore let us assume that these eigenvalues are in descending order ($\lambda_1 \geq ... \geq \lambda_n$ and $\mu_1 \geq ... \geq \mu_k$), and $\{\mathbf{z}_i\}$ are the corresponding eigenvectors of A. Then after k steps of Lanczos with blocks of size 1 (i.e., classical Lanczos),

$$\lambda_1 \geq \mu_1 \geq \lambda_1 - \frac{(\lambda_1 - \lambda_n)\tan^2\phi_1}{[C_{k-1}(1 + 2\rho_1)]^2},$$

where $\cos(\phi_1) = |\mathbf{q}_1^T \mathbf{z}_1|$, $\rho_1 = \frac{\lambda_1 - \lambda_2}{\lambda_2 - \lambda_n}$, and $C_{k-1}(x)$ is the $k-1^{st}$ Chebyshev polynomial of the first kind.

We can see that the governing parameters are the separation between the largest and smallest eigenvalues of A, the separations between λ_1 and λ_2, as well as that of λ_2 and λ_n. The angle between the initial guess vector and the leading eigenvector plays an important role. Furthermore, the denominator will increase dramatically with k.

Similarly, we can relate the smallest eigenvalues of A and T_k:

$$\lambda_n \le \mu_k \le \lambda_n + \frac{(\lambda_n - \lambda_1)\tan^2\phi_n}{[C_{k-1}(1+2\rho_n)]^2}$$

where $\cos(\phi_n) = |\mathbf{q}_1^T \mathbf{z}_n|$, and $\rho_n = \frac{\lambda_{n-1} - \lambda_n}{\lambda_1 - \lambda_n}$.

We have noted in the past that for differential operators, the eigenvalues tend to cluster near the smallest eigenvalue with relatively few outliers. To find the smallest eigenvalues in this case, we can work with $(A - \sigma I)^{-1}$ where σ is chosen to yield a better representation of the eigenvector. Suppose

$$\lambda_i = 0.1, 0.12, 0.13, ..., 2.0, 2.1, 2.2, 10, 11.$$

We might want to work with $(A - 0.5I)^{-1}$ or even A^{-1} to obtain the smallest eigenvalues. If we wanted information about eigenvalues clustered near 2, we could pick σ near 2 as well.

Unfortunately, the Lanczos algorithm does not live up to all of the claims made using exact arithmetic for floating point computations. As k (or n) grows large, the vectors $\{\mathbf{q}_i\}$ lose orthogonality due to the use of the 3-term recurrence relation. By using this, we explicitly construct $\mathbf{q}_{k+1}$ orthogonal to $\mathbf{q}_k$ and $\mathbf{q}_{k-1}$, and basically hope that the result in exact arithmetic that the remaining vectors are also orthogonal still holds, instead of explicitly forcing the result. This hope gives Lanczos its efficiency, but also suggests the use of re-orthogonalization if we can detect the breakdown of the algorithm. So we can use the Gram-Schmidt process,

$$\mathbf{r}_k := \mathbf{r}_k - \sum_{i=1}^{k-1} (\mathbf{r}_k^T \mathbf{q}_i)\mathbf{q}_i,$$

where the := operation denotes redefinition (overwriting). The obvious drawbacks of this process are that we must store all of the previous vectors $\mathbf{q}_i$ that we must use to compute the inner products.

Paige (1971, 1976) performed detailed studies of the numerical properties and made the following algorithmic adjustments. We shall use $\wedge$ to denote quantities computed from the classical Lanczos algorithm. Then,

$$A\widehat{Q_k} = \widehat{Q_k}\widehat{T_k} + \widehat{\mathbf{r}_k}\mathbf{e}_k^T + E_k,$$

where E_k denotes a matrix of errors, and $\|E_k\|_2 \approx \varepsilon_{\text{mach}}\|A\|_2$. Also, keep in mind that $\widehat{Q_k}^T\widehat{Q_k} \neq I$. From this we can conclude that

$$|\widehat{\mathbf{q}_{k+1}}^T\widehat{\mathbf{q}_i}| \approx \frac{|\widehat{\mathbf{r}_k}^T\widehat{\mathbf{q}_i}| + \varepsilon_{\text{mach}}\|A\|_2}{|\widehat{\beta_k}|}.$$

Thus, if $\widehat{\beta_k}$ gets small, we can expect to have big problems. Furthermore, if we denote the computed eigenpairs of $\widehat{T_k}$ as $(\widehat{\mu_k}, \widehat{\mathbf{s}_k})$, and $\widehat{Y_k} = \text{fl}(\widehat{Q_k}\widehat{S_k})$ (i.e., $\widehat{Y_k}$ is a matrix containing the computed approximations to eigenvectors of A), then

$$|\widehat{\mathbf{q}_{k+1}}^T\widehat{\mathbf{y}_i}| \approx \frac{\varepsilon_{\text{mach}}\|A\|_2}{|\widehat{\beta_k}||\widehat{s_{ki}}|},$$
$$\|A\widehat{\mathbf{y}_i} - \widehat{\mu_i}\widehat{\mathbf{y}_i}\|_2 \approx |\widehat{\beta_k}||\widehat{s_{ki}}|.$$

So one possible solution to the problem is to calculate $\|A\widehat{\mathbf{y}_i} - \widehat{\mu_i}\widehat{\mathbf{y}_i}\|_2$, and if the result is small, then $\widehat{\mathbf{y}_i}$ is a good approximation for an eigenvector, so we shall declare $(\widehat{\mu_i}, \widehat{\mathbf{y}_i})$ an eigenpair, and then reorthogonalize the future iterates against $\widehat{\mathbf{y}_i}$. This strategy, known as *selective reorthogonalization*, effectively provides a way of deflating the matrix A.

Other solutions involve ignoring the problem, and letting Lanczos run for $\sim 3n$ iterations, yielding $3n$ eigenvalue and eigenvector estimates, many of which are not good, but there should be n that are good. Another strategy, called *complete reorthogonalization*, simply involves reorthogonalizing at every step.

15.5 Netlib

Dongarra and Gross have assembled a set of algorithms which the scientific computing community has generally accepted as being confident. These codes are not perfect for every situation, but are very good for the majority of problems. The codes are freely available for download at

`http://www.netlib.org`

Lecture 16

Unsymmetric Lanczos, Orthogonal Polynomials, and Quadrature

16.1 Unsymmetric Lanczos

We have rather thoroughly discussed the Lanczos algorithm as it is applied to symmetric matrices. What can be done when A is not symmetric? Previously, we constructed matrices T and Q such that $Q^T AQ = T$ where T is symmetric and tridiagonal, and Q is orthogonal. This is not possible if $A \neq A^T$. Instead, we shall attempt to construct matrices Q and T such that

$$Q^{-1}AQ = T = \begin{bmatrix} \alpha_1 & \gamma_1 & & & \\ \beta_1 & \alpha_2 & \gamma_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{n-2} & \alpha_{n-1} & \gamma_{n-1} \\ & & & \beta_{n-1} & \alpha_n \end{bmatrix}.$$

Note that further difficulty arises when we attempt to compute the eigenvalues of T since QR iterations only preserve the tridiagonal structure when $T = T^T$.

In the generation of the algorithm, we shall attempt to form

$$\begin{aligned} A &= QTQ^{-1} \\ A^T &= Q^{-T}T^TQ^T \\ &\equiv PT^TP^{-1}. \end{aligned}$$

That is, we will form sequences of vectors $\{\mathbf{q}_i\}$ based on A, and $\{\mathbf{p}_i\}$ based on A^T. Since $P = Q^{-T}$, we have orthogonality relations

$$\begin{aligned} PQ^T &= I \\ Q^T P &= I \end{aligned}$$

Looking at our equations for A and A^T,

$$\begin{aligned} AQ &= QT \\ \Rightarrow A\mathbf{q}_k &= \beta_k \mathbf{q}_{k+1} + \alpha_k \mathbf{q}_k + \gamma_{k-1}\mathbf{q}_{k-1}. \\ A^T P &= PT \\ \Rightarrow A^T \mathbf{p}_k &= \gamma_k \mathbf{p}_{k+1} + \alpha_k \mathbf{p}_k + \beta_{k-1}\mathbf{p}_{k-1}. \end{aligned}$$

So,

$$\mathbf{p}_k^T A \mathbf{q}_k = \alpha_k.$$

Furthermore,

$$\begin{aligned} \beta_k \mathbf{q}_{k+1} &= \mathbf{r}_k \\ &= (A - \alpha_k I)\mathbf{q}_k - \gamma_{k-1}\mathbf{q}_{k-1}. \\ \gamma_k \mathbf{p}_{k+1} &= \mathbf{s}_k \\ &= (A - \alpha_k I)^T \mathbf{p}_k - \beta_{k-1}\mathbf{p}_{k-1}. \end{aligned}$$

Thus,

$$\begin{aligned} \mathbf{p}_{k+1}^T \mathbf{q}_{k+1} &= 1 \\ &= \frac{\mathbf{s}_k^T \mathbf{r}_k}{\gamma_k \beta_k}. \\ \Rightarrow \gamma_k &= \frac{\mathbf{s}_k^T \mathbf{r}_k}{\beta_k}. \end{aligned}$$

Therefore, γ_k and β_k are not defined uniquely. By convention, one often chooses $\beta_k = \|\mathbf{r}_k\|_2$.

Algebraically, everything looks fine, but computationally, it leaves much to be desired. If $\mathbf{r}_k = \mathbf{0}$, then iterations terminate. This results in the formation of an invariant subspace of A, namely $[\mathbf{q}_1, ..., \mathbf{q}_k]$. Furthermore, the eigenvalues of T_k are eigenvalues of A. So, this is not a bad situation to be in. Similarly, if $\mathbf{s}_k = \mathbf{0}$, iterations terminate with the vectors $[\mathbf{p}_1, ..., \mathbf{p}_k]$ forming an invariant subspace of A^T, and the eigenvalues of T_k are again eigenvalues of A.

The bad situation is if $\mathbf{s}_k^T \mathbf{r}_k = 0$. Then we truly break down since β_k and γ_k cannot be constructed. This situation is a mathematical possibility as well as a computational possibility.

One solution to this problem is due to Parlett. He devised a look-ahead strategy. The idea is to augment the sequences, but now, T_k is not tridiagonal. Instead you get a tridiagonal matrix, with a step associated with the look-ahead iterations, when they are needed:

$$T_k = \begin{bmatrix} \times & \times & & & & & & & \\ \times & \times & \times & & & & & & \\ & \times & \times & \times & & & & & \\ & & \times & \times & \times & \times & \times & & \\ & & & \times & \times & \times & \times & & \\ & & & & \times & \times & \times & & \\ & & & & & \times & \times & \times & \\ & & & & & & \times & \times & \times \\ & & & & & & & \times & \times \end{bmatrix}.$$

Once we have generated T_k, two problems arise. First, mathematically, how do the eigenvalues of T_k relate to those of A? And, secondly, how do we compute the eigenvalues of T_k? This second problem is a legitimate problem since T_k is not symmetric, so standard QR iterations will introduce a filling of the upper right quadrant of non-zeros, resulting in an upper Hessenberg form. Newton's method produces real valued roots of polynomials given real initial guesses, so perhaps we could use a complex set of initial guesses. Alternatively, we could use a method based on finding roots of quadratics, such as Muller's Method. The generalization of QR iterations referred to as LR iterations will preserve the tridiagonal form, but there are numerical issues with this algorithm. The first question has yet to be resolved as well.

An alternative to Lanczos is to use *Arnoldi iteration*, which will generate an upper Hesenberg form:

$$\begin{aligned} AQ &= QH. \\ \Rightarrow A\mathbf{q}_1 = h_{11}\mathbf{q}_1 + h_{21}\mathbf{q}_2. & \\ \therefore h_{11} &= \mathbf{q}_1^T A\mathbf{q}_1, \\ h_{21} &= \|A\mathbf{q}_1 - h_{11}\mathbf{q}_1\|_2, \\ \mathbf{q}_2 &= \frac{1}{h_{21}}(A\mathbf{q}_1 - h_{11}\mathbf{q}_1). \end{aligned}$$

One proceeds in a similar manner to complete the construction. This construction is also used in some methods for solving $A\mathbf{x} = \mathbf{b}$. One of these methods is known as *Quasi-Minimal Residual* (QMR).

16.2 Orthogonal Polynomials

Let $p_j(x)$ be a polynomial of degree less than or equal to j. Then we say a family of polynomials $\{p_k(x)\}$ are *orthogonal with respect to* $w(x)$ if

$$\int_a^b p_r(x)p_s(x)w(x)dx = 0 \qquad \text{for } r \neq s.$$

Alternatively, given a distribution function $f(x)$, the family is said to be orthogonal with respect to f if

$$\int p_r(x)p_s(x)df = 0 \qquad \text{for } r \neq s.$$

With this representation, we can include discontinuous distribution functions. The family of polynomials are called *orthonormal* if they are orthogonal and $\int p_r(x)p_r(x)df = 1$.

All families of orthogonal polynomial will satisfy a 3-term recurrence relationship of the form

$$p_{r+1}(x) = (x - \alpha_{r+1})p_r(x) - \beta_r^2 p_{r-1}(x), \qquad p_0(x) = 0, \quad p_1(x) = 1.$$

This implies that

$$xp_r(x) = p_{r+1}(x) + \alpha_{r+1}p_r(x) + \beta_r^2 p_{r-1}(x). \tag{16.1}$$

Let us now define a vector of polynomials in x,

$$\pi_n(x) = \begin{bmatrix} p_o(x) \\ \vdots \\ p_n(x) \end{bmatrix}.$$

Then (16.1) tells us

$$x\pi_n(x) = \begin{bmatrix} \alpha_1 & 1 & & & \\ \beta_1^2 & \alpha_2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{n-2}^2 & \alpha_{n-1} & 1 \\ & & & \beta_{n-1}^2 & \alpha_n \end{bmatrix} \pi_n(x) + p_n(x)\mathbf{e}_n.$$

$$\therefore x\pi_n(x) = T\pi_n(x) + p_n(x)\mathbf{e}_n.$$

Thus, if we wanted to find the roots of $p_n(x)$ (i.e., all $x = \lambda_i$ such that $p_n(\lambda_i) = 0$), we should solve the eigenvalue problem

$$\lambda_i \pi_i = T\pi_i.$$

Recall that for this matrix T, there exists a diagonal matrix D such that

$$DTD^{-1} = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{n-2} & \alpha_{n-1} & \beta_{n-1} \\ & & & \beta_{n-1} & \alpha_n \end{bmatrix},$$

which is clearly symmetric. Hence, shifted QR iterations would work quite well here. Furthermore, this formulation shows that the roots are clearly real valued and distinct if each $\beta_i \neq 0$. We can also find bounds on these roots from Gerschgorin disks, and other means.

16.3 Gaussian Quadrature

Suppose we wanted to numerically evaluate (approximate) an integral. We can follow the suggestion of Riemann summation, dividing the interval into regions (quadrants), and approximating the integrand as a sum of area calculations,

$$\int_a^b g(x)df(x) \approx \sum_{i=1}^n w_i g(x_i).$$

The points of evaluation of g, $\{x_i\}$, are called *nodes*, and the coefficients that proceed them, $\{w_i\}$, are called *weights*. One method for choosing the weights and nodes is to first pick nodes, perhaps uniformly spaced across $[a, b]$, then construct the interpolant of g based on these nodes, and then selecting the weights so that the interpolant is integrated exactly. But we can clearly get better methods if we choose the nodes and weights more carefully.

We shall use a technique known as *moment matching*. The k^{th} moment is

$$\mu_k = \int_a^b x^k df.$$

We then pick n, and select weights and nodes so that the first $2n$ moments are computed exactly:

$$\mu_k = \sum_{i=1}^n w_i x_i^k, \qquad \text{for } k = 0, 1, ..., 2n-1.$$

Since we have $2n$ free parameters to choose from it is reasonable to think that this method might succeed, but this system of equations is non-linear, so we cannot be sure without performing some analysis first.

Suppose $g(x)$ is a polynomial of degree $2n-1$ (i.e., $g \in \mathcal{P}_{2n-1}$). We shall show that there exists $\{w_i\}$ and $\{x_i\}$ such that

$$\int_a^b g(x)df = \sum_{i=1}^n w_i g(x_i).$$

Furthermore, for more general functions, $G(x)$,

$$\int_a^b G(x)df = \sum_{i=1}^n w_i G(x_i) + E[G]$$

where

1. x_i are real, distinct, and $a < x_i < b$ for $i = 1, ..., n$.

2. $w_i > 0$.

3. $E[G] = \frac{G^{(2n)}(\xi)}{(2n)!} \int_a^b \prod_{i=1}^n (x - x_i)^2 df$.

Notice that this method is exact for polynomials of degree $2n - 1$ since the error functional depends on the $(2n)^{th}$ derivative of G.

To prove this, we shall construct an orthonormal family of polynomials $\{p_i(x)\}$, so that

$$\int_a^b p_r(x)p_s(x)df = \begin{cases} 0 & r \neq s, \\ 1 & r = s. \end{cases}$$

We choose $\{x_i\}$ to be the roots of the n^{th} polynomial in this family. Now, we will construct the interpolant, L_{n-1}, of $g(x)$ through the nodes:

$$\begin{aligned} l_i(x) &= \frac{p_n(x)}{(x - x_i)p_n'(x_i)} \\ \therefore l_i(x_j) &= \frac{p_n(x_j)}{(x_j - x_i)p_n'(x_j)} \\ &= \begin{cases} 0 & i \neq j \\ \lim_{x \to x_i} \frac{p_n(x) - p_n(x_i)}{x - x_i} \cdot \frac{1}{p_n'(x_i)} = 1 & i = j. \end{cases} \\ l_i(x) &\in \mathcal{P}_{n-1} \\ L_{n-1}(x) &= \sum_{i=1}^n g(x_i)l_i(x). \\ \therefore L_{n-1}(x_j) &= \sum_{i=1}^n g(x_i)l_i(x_j) = g(x_j)l_j(x_j) = g(x_j). \end{aligned}$$

We shall now look at the interpolation error function

$$e(x) = g(x) - L_{n-1}(x).$$

Clearly, since $g \in \mathcal{P}_{2n-1}$, $e \in \mathcal{P}_{2n-1}$. Since $e(x)$ has roots at each of the roots of $p_n(x)$, we can factor e so that

$$e(x) = p_n(x)r(x),$$

then $r \in \mathcal{P}_{n-1}$. The integral of g can be written as

$$\begin{aligned}
\int_a^b g(x)df &= \int_a^b L_{n-1}(x)df + \int_a^b p_n(x)r(x)df \\
&= \int_a^b L_{n-1}(x)df \\
&= \int_a^b \sum_{i=1}^n g(x_i)l_i(x)df \\
&= \sum_{i=1}^n g(x_i) \int_a^b l_i(x)df \\
&\equiv \sum_{i=1}^n g(x_i)w_i
\end{aligned}$$

where $w_i = \int_a^b l_i(x)df = \int_a^b \frac{p_n(x)}{(x-x_i)p_n'(x_i)}df$. If this formula were easy to evaluate exactly, we could use it to find the weights. If not, once we have computed the nodes, we could use moment matching, and solve an $n \times n$ system of linear equations to obtain the weights.

It is easy to show that the weights are positive. Since the interpolation basis functions $l_i \in \mathcal{P}_{n-1}$, $l_i^2 \in \mathcal{P}_{2n-2}$,

$$0 < \int_a^b l_i^2(x)df = \sum_{j=1}^n w_j l_i^2(x_j) = w_i.$$

References

[1] C. Lanczos. *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators.* J. Res. Nat. Bur. Standards, 45:255-282, 1950.

[2] C. Lanczos. *Solution of systems of linear equations by minimized iterations.* J. Res. Nat. Bur. Standards, 49:33-53, 1952.

[3] W. E. Arnoldi. *The principle of minimized iterations in the solution of the matrix eigenvalue problem.* Quart. Appl. Math., 9:17-29, 1951

[4] G. Golub and R. Underwood. *The block Lanczos method for computing eigenvalues.* In J. Rice, editor, Mathematical Software III, pages 364-377. Academic Press, New York, 1977.

[5] G. Golub and C. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, third edition, 1996.

[6] G. Golub and J. H. Wilkinson. *Ill-conditioned eigensystems and the computation of the Jordan canonical form.* SIAM Rev., 18(4):578-619, 1976.

[7] J. H. Wilkinson. *The Algebraic Eigenvalue Problem.* Clarendon Press, Oxford, UK, 1965.

[8] B. N. Parlett. *The Symmetric Eigenvalue Problem.* Prentice-Hall, Englewood Cliffs, NJ, 1980. Reprinted as Classics in Applied Mathematics 20, SIAM, Philadelphia, 1997.

[9] H. Yang. *Conjugate gradient methods for the Rayleigh quotient minimization of generalized eigenvalue problems.* Computing, 51(1):79-94, 1993.

[10] C. F. Van Loan. *A symplectic method for approximating all the eigenvalues of a Hamiltonian matrix.* Linear Algebra Appl., 61:233-251, 1984.

[11] C. F. Van Loan. *Computational Frameworks for the Fast Fourier Transform.* SIAM, Philadelphia, 1992.

[12] 1G. W. Stewart and J.-G. Sun. *Matrix Perturbation Theory.* Academic Press, New York, 1990.

[13] Y. Saad. *Chebyshev acceleration techniques for solving nonsymmetric eigenvalue problems.* Math. Comp., 42(166):567-588, 1984. 384

[14] Y. Saad. *Least squares polynomials in the complex plane and their use for solving nonsymmetric linear systems.* SIAM J. Numer. Anal., 24(1):155-169, 1987. 385

[15] Y. Saad. *Numerical solution of large nonsymmetric eigenvalue problems.* Comp. Phys. Comm., 53:71-90, 1989. 386

[16] Y. Saad. *SPARSKIT: A basic tool-kit for sparse matrix computation, version 2, 1994.* Software available at http://www.cs.umn.edu/ saad 387

[17] Y. Saad. *Numerical Methods for Large Eigenvalue Problems* Halsted Press, New York, 1992. 388

[18] Y. Saad. *Iterative Methods for Linear Systems.* PWS Publishing, Boston, 1996. 389

[19] Y. Saad and M. H. Schultz. *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems.* SIAM J. Sci. Statist. Comput., 7:856-869, 1986. 390

[20] B. N. Parlett and D. Scott. *The Lanczos algorithm with selective orthogonalization.* Math. Comput., 33:217-238, 1979.

[21] S. G. Nash and A. Sofer. *Linear and Nonlinear Programming.* McGraw-Hill, New York, 1995.

Contents