MA6606 — Computational Linear Algebra

Ya Yan Lu
Department of Mathematics
City University of Hong Kong

September 16, 2008

Contents

Chapter 1

More on Linear Algebra

Numerical (or computational) linear algebra is a subject that studies numerical methods for linear algebra problems. The numerical methods are given in terms of algorithms which can be implemented as computer programs. Three main problems are: (1) linear system of equation $Ax = b$, (2) least squares problems $\min ||Ax - b||$ and (3) eigenvalue problem $Ax = \lambda x$. To understand these methods, a solid background in linear algebra is essential. Some linear algebra topics are covered in this chapter.

1.1 Orthogonal matrix

Let x and y be vectors (of complex number) of length m, say

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}, \qquad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix},$$

we define the inner product of x and y as

$$x^* y = \overline{x}_1 y_1 + \overline{x}_2 y_2 + ... + \overline{x}_m y_m$$

where $\overline{x}_j$ is the complex conjugate of x_j, x^* is the row vector obtained from x by transpose and complex conjugate. Namely,

$$x^* = [\overline{x}_1, \overline{x}_2, ..., \overline{x}_m].$$

The two vectors are orthogonal to each other, if $x^* y = 0$.

This also gives rise to the definition of 2-norm (or, Euclidean norm):

$$||x||_2 = \sqrt{x^* x} = \sqrt{|x_1|^2 + |x_2|^2 + ... + |x_m|^2}.$$

A unit vector (in the 2-norm) is the vector with $||x||_2 = 1$.

Let Q be an $m \times m$ (complex) matrix, we write down its columns as q_1, q_2, ..., q_m. Namely,

$$Q = [q_1, q_2, ..., q_m].$$

The matrix Q is unitary if the column vectors are unit vectors and they are orthogonal to each other. More precisely,

$$q_i^* q_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

The above condition is precisely

$$Q^* Q = I,$$

where I is the identity matrix. As before, $Q^* = \overline{Q}^T$ is the transpose of $\overline{Q}$ and $\overline{Q}$ is the complex conjugate of Q. The above equation implies that Q^* is the inverse of Q, and of course, we also have $QQ^* = I$. Notice that the condition $QQ^* = I$ implies that the rows of Q are unit vectors and they are orthogonal to each other.

If Q is real and unitary (thus $Q^T = Q^{-1}$), then Q is called orthogonal.

A 1×1 unitary matrix is just a complex number, say a, satisfying $|a| = 1$. In other words, $a = e^{i\theta}$ for some real number θ.

A 2×2 orthogonal matrix can be written down as

$$Q = \begin{bmatrix} c & s \\ s & -c \end{bmatrix} \quad \text{or} \quad Q = \begin{bmatrix} c & -s \\ s & c \end{bmatrix}$$

where

$$c = \cos(\theta), \qquad s = \sin(\theta)$$

for some real number θ.

Theorem 1 *If Q is unitary and x is a column vector, then*

$$||Qx||_2 = ||x||_2.$$

Proof: $||Qx||_2 = \sqrt{(Qx)^* Qx} = \sqrt{x^* Q^* Qx} = \sqrt{x^* x} = ||x||_2$, since $Q^* Q = I$. □

1.2 Eigenvalue decomposition

If $A = A^*$, the matrix A is called Hermitian. It must be a square matrix (say $m \times m$). A real Hermitian matrix is a symmetric matrix ($A = A^T$). For a square matrix A, an eigenvalue λ is a solution of the polynomial equation:

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

In this case, there is a non-zero vector x, such that

$$(\lambda I - A)x = 0, \quad \text{or} \quad Ax = \lambda x.$$

The vector x is the eigenvector corresponding to the eigenvalue λ.

Theorem 2 *The eigenvalues of a Hermitian matrix are real.*

Proof: Let A be a Hermitian matrix and $Ax = \lambda x$, where λ is an eigenvalue and x is the corresponding eigenvector. If we multiply x^* to $Ax = \lambda x$, we obtain

$$x^*Ax = \lambda x^*x.$$

Now, if we take the transpose and complex conjugate (i.e. the $*$-operation) of the above identity, we obtain

$$(x^*Ax)^* = x^*A^*(x^*)^* = x^*Ax = (\lambda x^*x)^* = \overline{\lambda}x^*x.$$

Here, we have used $A^* = A$ and $(x^*)^* = x$. Therefore,

$$\lambda x^*x = \overline{\lambda}x^*x.$$

Since the eigenvector x is non-zero, $x^*x = ||x||_2^2 \neq 0$, we have

$$\lambda = \overline{\lambda}.$$

Therefore, λ is real. □

Theorem 3 *For a Hermitian matrix, the eigenvectors corresponding to distinct eigenvalues are orthogonal to each other.*

Proof: Let A be a Hermitian matrix, λ_1 and λ_2 are two eigenvalues of A, such that $\lambda_1 \neq \lambda_2$. From Theorem 2, we know that λ_1 and λ_2 are real. Let x_1 and x_2 be the corresponding eigenvectors. That is,

$$Ax_1 = \lambda_1 x_1, \quad Ax_2 = \lambda_2 x_2.$$

For $Ax_1 = \lambda_1 x_1$, we multiply x_2^*. Thus,

$$x_2^*Ax_1 = \lambda_1 x_2^*x_1.$$

Take the $*$-operation for both sides, we get

$$(x_2^*Ax_1)^* = x_1^*A^*(x_2^*)^* = x_1^*Ax_2 = (\lambda_1 x_2^*x_1)^* = \overline{\lambda}_1 x_1^*(x_2^*)^* = \lambda_1 x_1^*x_2.$$

For the equation $Ax_2 = \lambda_2 x_2$, we multiply x_1^*, thus

$$x_1^*Ax_2 = \lambda_2 x_1^*x_2.$$

Compare the two equations above, we obtain

$$\lambda_1 x_1^*x_2 = \lambda_2 x_1^*x_2.$$

This leads to

$$(\lambda_1 - \lambda_2)x_1^*x_2 = 0.$$

Since $\lambda_1 \neq \lambda_2$, we must have

$$x_1^*x_2 = 0.$$

Therefore, the two eigenvectors are orthogonal to each other. □

Vectors q_1, q_2, ..., q_k are called **orthonormal** if they are unit vectors (i.e. the 2-norm is 1) and they are orthogonal to each other. This implies

$$q_i^*q_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}.$$

Lemma 1 *Any given k orthonormal vectors of length m ($k < m$) can always be regarded as the first k columns of an $m \times m$ unitary matrix.*

Proof: Let q_1, q_2, ..., q_k be the given k orthonormal vectors. Let $\mathbb{C}^m$ be the m-dimensional space of all complex columns vectors of length m. Let $< q_1, q_2, ..., q_k >$ be the k-dimensional space spanned by the k orthonormal vectors. That is

$$< q_1, q_2, ..., q_k >= \{w \mid w \text{ is a linear combination of } q_1, q_2, ..., q_k\}.$$

Since $< q_1, q_2, ..., q_k >$ is a subspace of $\mathbb{C}^m$ and its dimension is smaller than m, we can always find a vector u in $\mathbb{C}^m$, but not in $< q_1, q_2, ..., q_k >$. Now, let

$$v = u - (q_1^* u)q_1 - (q_2^* u)q_2 - ... - (q_k^* u)q_k.$$

Obviously,

$$q_i^* v = 0 \quad \text{for } i = 1, 2, ..., k,$$

and $v \neq 0$ (otherwise $u \in < q_1, q_2, ..., q_k >$). We can thus let

$$q_{k+1} = v/||v||_2.$$

Then, the $k+1$ vectors q_1, ..., q_k, q_{k+1} are orthonormal. This process can be repeated untill we find the last vector q_m. Then, the matrix $Q = [q_1, q_2, ..., q_m]$ is unitary. □

The following theorem is the eigenvalue decomposition of a Hermitian matrix. It may be the most important result in linear lagebra.

Theorem 4 *(Eigenvalue decomposition) An $m \times m$ Hermitian matrix A can always be written as*

$$A = Q\Lambda Q^*, \text{ or } AQ = Q\Lambda, \text{ or } Q^* A Q = \Lambda,$$

where Q is unitary and Λ is real and diagonal. Meanwhile, the diagonal entries of Λ are the eigenvalues of A and the columnes of Q are the corresponding eigenvectors.

Proof: The proof is by induction. The case of $m = 1$ is obvious. You can choose $Q = 1$ and $\Lambda = A$. Let us assume that the eigenvalue decomposition is established for all Hermitian matrices of size $(m-1) \times (m-1)$. Now, for our $m \times m$ Hermitian matrix, we start with one eigenvalue λ_1 and its corresponding unit eigenvector q_1, such that $Aq_1 = \lambda_1 q_1$. From Lemma 1, we can find an unitary matrix

$$U = [q_1, u_2, u_3, ..., u_m].$$

Then,

$$U^* A U = U^*[Aq_1, Au_2, ..., Au_m] = \begin{bmatrix} q_1^* \\ u_2^* \\ \vdots \\ u_m^* \end{bmatrix} [\lambda_1 q_1, Au_2, ..., Au_m].$$

Since q_1 is an unit vector and it is orthogonal with u_k $(k = 2, ..., m)$, we have

$$U^*AU = \begin{bmatrix} \lambda_1 & * & ... & * \\ 0 & * & ... & * \\ \vdots & * & ... & * \\ 0 & * & ... & * \end{bmatrix}.$$

But, U^*AU is also Hermitian, thus

$$U^*AU = \begin{bmatrix} \lambda_1 & 0 & ... & 0 \\ 0 & * & ... & * \\ \vdots & * & ... & * \\ 0 & * & ... & * \end{bmatrix} = \begin{bmatrix} \lambda_1 & \\ & A_1 \end{bmatrix}$$

where A_1 is an $(m-1) \times (m-1)$ Hermitian matrix. Because of the induction assumption, we have an eigenvalue decomposition of A_1:

$$A_1 = Q_1 \Lambda_1 Q_1^*.$$

Thus,

$$A = U \begin{bmatrix} 1 & \\ & Q_1 \end{bmatrix} \begin{bmatrix} \lambda_1 & \\ & \Lambda_1 \end{bmatrix} \begin{bmatrix} 1 & \\ & Q_1^* \end{bmatrix} U^*.$$

Now, let

$$Q = U \begin{bmatrix} 1 & \\ & Q_1 \end{bmatrix}, \quad \Lambda = \begin{bmatrix} \lambda_1 & \\ & \Lambda_1 \end{bmatrix},$$

we have

$$A = Q \Lambda Q^*.$$

Clearly, Λ is real and diagonal. It is also easy to verify that Q is unitary and the first column of Q is q_1.

Now, if we let

$$Q = [q_1, q_2, ..., q_m], \quad \Lambda = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_m \end{bmatrix},$$

then, from $AQ = Q\Lambda$, we have

$$Aq_j = \lambda_j q_j, \quad j = 1, 2, ..., m.$$

Therefore, Λ and Q are the matrices for eigenvalues and eigenvectors, respectively. □

For the following non-symmetric matrix

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{bmatrix},$$

the three eigenvalues are

$$\lambda_1 = \lambda_2 = 1, \quad \lambda_3 = 3.$$

Corresponing to the double eigenvalue $\lambda = 1$, there is only one linearly independent eigenvector. For a Hermitian matrix, a double eigenvalue must have two linearly independent eigenvectors.

Theorem 5 *If λ is a multiple eigenvalue of a Hermitian matrix A with multiplicity k, then there are exactly k linearly independent eigenvectors corresponding to λ.*

Proof: From the eigenvalue decomposition of A, we have a set of orthonormal eigenvectors q_1, q_2, ..., q_m satisfying $Aq_j = \lambda_j q_j$. For a multiple eigenvalue λ of multiplicity k, there are k coresponding eigenvectors in the orthonormal set. They are obviously linearly independent. □

In general, it is not always possible to diagonalize a non-Hermitian matrix. When the matrix can be diagonalized, we have a non-singular matrix S and a diagonal matrix Λ, such that

$$A = S\Lambda S^{-1}, \quad AS = S\Lambda, \quad S^{-1}AS = \Lambda,$$

where the diagonal entries of Λ are the eigenvalues, the columns of S are the corresponding eigenvectors. The above is not always possible, because some multiple eigenvalues may not have enough linearly independent eigenvectors. Even when the above is possible, the matrix S maybe near singular. That is, the columns of S can be nearly linearly dependent on each other. In general, for non-Hermitian matrices, we do not attempt to calculate the eigenvalue decomposition. Instead, we use the so-called Schur decomposition.

Theorem 6 *(Schur decomposition) For any $m \times m$ complex matrix A, there is an unitary matrix Q and an upper triangular matrix T, such that*

$$A = QTQ^*, \quad AQ = QT, \quad Q^*AQ = T.$$

Meanwhile, the diagonals of T are the eigenvalues of A.

Proof: The proof is nearly identical to the proof of Theorem 4. □

Notice that Theorem 4 can be easily obtained from Theorem 6. Since A is Hermitian, then the upper triangular matrix $T = Q^*AQ$ is also Hermitian. Thus, T is real and diagonal.

1.3 Vector norm

A norm is a function defined on a vector space satisfying three basic conditions. Let us denote a norm (of some vector x) by $||x||$, we must have

1. for any x in the given vector space $||x|| \geq 0$, and $||x|| = 0$ if and only if $x = 0$;
2. for artitrary x and y in the given vector space,
$$||x + y|| \leq ||x|| + ||y||;$$
3. for any complex number α and any vector x,
$$||\alpha x|| = |\alpha|||x||.$$

To prove the 2-norm defined based on inner product is actually a norm, you need to verify the above three conditions. The first and the last conditions are easy to check. The second condition is the triangular inequality, which can be proved using the Cauchy-Schwarz inequality.

Lemma 2 *(Cauchy-Schwarz inequality) Let x and y be two column vectors of length m, then*

$$|x^*y| \leq ||x||_2||y||_2.$$

Proof: We can assume that the vectors x and y are non-zero vectors, otherwise, both sides of the above inequality are just zero. Now, for any complex t,

$$||x - ty||_2^2 \geq 0.$$

The left hand side above is actually a quadratic polynomial of t_1 and t_2, where $t = t_1 + it_2$ ($i = \sqrt{-1}$). We can find the minimum of the left hand side and the minimum should still be non-negative.

$$||x - ty||_2^2 = (x^* - \bar{t}y^*)(x - ty) = ||x||_2^2 - tx^*y - \bar{t}y^*x + |t|^2||y||_2^2$$

If we let $x^*y = \alpha + i\beta$ for real α and β, then

$$||x - ty||_2^2 = ||x||_2^2 - 2(\alpha t_1 - \beta t_2) + (t_1^2 + t_2^2)||y||_2^2.$$

The minimum of the above is reached at

$$t_1 = \frac{\alpha}{||y||_2^2}, \quad t_2 = -\frac{\beta}{||y||_2^2}.$$

Insert the above values of t_1 and t_2, we have

$$0 \leq \min ||x - ty||_2^2 = ||x||_2^2 - \frac{\alpha^2 + \beta^2}{||y||_2^2}.$$

This implies $|x^*y| \leq ||x|| \cdot ||y||$. □

Theorem 7 *For any two vectors of same length, the 2-norm satisfies the triangular inequality:*

$$||x + y||_2 \leq ||x||_2 + ||y||_2.$$

Proof: Notice that

$$||x + y||_2^2 = (x^* + y^*)(x + y) = ||x||_2^2 + x^*y + y^*x + ||y||_2^2 \leq ||x||_2^2 + |x^*y| + |y^*x| + ||y||_2^2.$$

Now, we use the Cauchy-Schwarz inequalities:

$$|x^*y| \leq ||x||_2||y||_2 \quad \text{and} \quad |y^*x| \leq ||x||_2||y||_2.$$

Thus,

$$||x + y||_2^2 \leq ||x||_2^2 + ||x||_2||y||_2 + ||x||_2||y||_2 + ||y||_2^2 = (||x||_2 + ||y||_2)^2.$$

Therefore, $||x + y||_2 \leq ||x||_2 + ||y||_2$. □

More generally, we can define the **vector p-norm** for $p \geq 1$ by

$$||x||_p = (|x_1|^p + |x_2|^p + ... + |x_m|^p)^{1/p}$$

A particulay simple and useful norm is the 1-norm for $p = 1$. That is,

$$||x||_1 = |x_1| + |x_2| + ... + |x_m|.$$

Moreover, the limit of $p \to \infty$ gives

$$||x||_\infty = \max_{1 \le j \le m} |x_j|.$$

This is the ∞-norm. For the general p-norm, we have the following Hölder inequality:

Lemma 3 *For p and q satisfying $1/p + 1/q = 1$ and $1 \le p,\ q \le \infty$, and for any two vectors of same length, we have*

$$|x^* y| \le ||x||_p \ ||y||_q.$$

The Cauchy-Schwarz inequality is the special case $p = q = 2$.

Based on the p-norm, we can define a weighted p-norm, using a non-singular diagonal matrix W:

$$W = \begin{bmatrix} w_1 & & & \\ & w_2 & & \\ & & \ddots & \\ & & & w_m \end{bmatrix}$$

(assuming $w_i \neq 0$ for all i). Here is the weighted 2-norm:

$$||x||_W = \sqrt{\sum_{i=1}^{m} |w_i x_i|^2} = ||Wx||_2.$$

The notation $|| \cdot ||_W$ is not a standard notation.

1.4 Matrix norm

For two matrices of same size (say, both 3×3), one matrix may have some large entries and the other matrix may have only small entries. To distinguish these two matrices, we could attach a number to each matrix. This would be the matrix norm. Naturally, the matrix with larger entries would have a larger norm on average. However, there is not a unique way of defining the matrix norm. Like vector norms, there are many different matrix norms.

An important class of matrix norms is the **induced** matrix norms. This is related to what the matrix will do for vectors. Let A be a complex $m \times n$ matrix, an important aspect of the matrix is that it takes x in $\mathbb{C}^n$ (the vector space of complex column vectors of length n) to $y = Ax$ in $\mathbb{C}^m$. That is to say, the matrix A represents a **linear operator** that maps x to y. Since x and y can be mesured by their norms in $\mathbb{C}^n$ and $\mathbb{C}^m$, respectively, we will introduce the matrix norm as the maximum of the ratio $||y||/||x||$ among all non-zero x. Therefore, the matrix norm is the largest possible increasing (or decreasing) factor when it acts on vectors.

More precisely, let $||\cdot||_{(n)}$ be a vector norm for $\mathbb{C}^n$ and $||\cdot||_{(m)}$ be a vector norm for $\mathbb{C}^m$, (these two norms do not have to be the same, as $\mathbb{C}^n$ and $\mathbb{C}^m$ are different vector spaces), we define the **induced matrix norm** of A as:

$$||A||_{(m,n)} = \sup_{0 \neq x \in \mathbb{C}^n} \frac{||Ax||_{(m)}}{||x||_{(n)}}.$$

Since the right hand side above does not change when x is replaced by αx for any constant α, we can assume $||x||_{(n)} = 1$ and use the definition

$$||A||_{(m,n)} = \sup_{x \in \mathbb{C}^n,\ ||x||_{(n)}=1} ||Ax||_{(m)}.$$

Since the induced matrix norm is defined as the maximum above, for any vector x, we always have

$$||Ax||_{(m)} \leq ||A||_{(m,n)}\ ||x||_{(n)}.$$

If the norms for x and $y = Ax$ in the definition of the induced matrix norm are the same, then the induced norm will use the same name as the norm for x and y. That is, if the p-norm is used for both $x \in \mathbb{C}^n$ and $Ax \in \mathbb{C}^m$, then the induced matrix norm is the **matrix p-norm**. Among them, the most important matrix norm is the matrix 2-norm, which is defined using vector 2-norms for both x and $y = Ax$. Therefore,

$$||A||_2 = \sup_{x \in \mathbb{C}^n,\ ||x||_2=1} ||Ax||_2.$$

Example: Consider

$$A = \begin{bmatrix} 1 & 2 \\ 1 & 1 \\ 3 & 0 \end{bmatrix},$$

if we use $||\cdot||_2$ for $\mathbb{C}^2$ and $||\cdot||_1$ for $\mathbb{C}^3$, then induced matrix norm of A is

$$||A||_{1,2} = \sup_{|x_1|^2+|x_2|^2=1} (|x_1 + 2x_2| + |x_1 + x_2| + |3x_1|) = \sqrt{34}.$$

Another matrix norm, which is often used because it is easy to compute, is the **Frobenius norm**. It is not an "induced matrix norm". For an $m \times n$ matrix A, it is like the vector 2-norm, when the matrix A is regarded as a vector of length mn. That is,

$$||A||_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{1/2}$$

where a_{ij} is the (i, j) entry of A.

These matrix norms defined above are actually "norms" following the definition (the three conditions) in the previous section. In this case, the set of $m \times n$ matrices are considered as a "vector space". We have

Theorem 8 *The induced matrix norms and the Frobenius norm satisfy the following three conditions (for a general definition of norm):*

1. $||A|| \geq 0$, *and* $||A|| = 0$ *only if* $A = 0$ *(the zero matrix),*

2. $||A + B|| \leq ||A|| + ||B||$, *where* A *and* B *have the same size.*

3. $||\alpha A|| = |\alpha| \ ||A||$, *where* α *is a complex scalar.*

For an $l \times m$ matrix A, and an $m \times n$ matrix B, we have an $l \times n$ matrix $C = AB$. We can consider how the matrix norm of C is bounded by the matrix norms of A and B.

Theorem 9 *Let* $||\cdot||_{(l)}$, $||\cdot||_{(l)}$, *and* $||\cdot||_{(l)}$ *be norms on* $\mathbb{C}^l$, $\mathbb{C}^m$, *and* $\mathbb{C}^n$, *respectively, then the induced norms satisfy*

$$||AB||_{(l,n)} \leq ||A||_{(l,m)} \ ||B||_{(m,n)}.$$

Proof: For any vector $x \in \mathbb{C}^n$, we have

$$||ABx||_{(l)} \leq ||A||_{(l,m)} \ ||Bx||_{(m)} \leq ||A||_{(l,m)} \ ||B||_{(m,n)}||x||_{(n)}.$$

The result is then clear from the definition of the matrix induced norm. □

For the Frobenius norm, we have a similar result.

Theorem 10 *The Frobenius norm of matrices satisfy*

$$||AB||_F \leq ||A||_F \ ||B||_F.$$

Proof: Let A and B be matrices of size (l, m) and (m, n), respectively, we write down the rows of A and columns of B:

$$A = \begin{bmatrix} a_1^* \\ a_2^* \\ \vdots \\ a_l^* \end{bmatrix}, \quad B = [b_1, b_2, ..., b_n].$$

Then,

$$||A||_F^2 = \sum_{i=1}^{l} ||a_i||_2^2, \quad ||B||_F^2 = \sum_{j=1}^{n} ||b_j||_2^2.$$

The produce $C = AB$ is a matrix of size (l, n). The (i, j) entry of C is

$$c_{ij} = a_i^* b_j.$$

From the Cauchy-Schwarz inequality, we have

$$|c_{ij}|^2 \leq ||a_i||_2^2 \ ||b_j||_2^2.$$

Thus,

$$||AB||_F^2 = \sum_{i=1}^{l}\sum_{j=1}^{n} |c_{ij}|^2 \leq \sum_{i=1}^{l}\sum_{j=1}^{n} ||a_i||_2^2 \ ||b_j||_2^2 = \sum_{i=1}^{l} ||a_i||_2^2 \ \sum_{j=1}^{n} ||b_j||_2^2 = ||A||_F^2 \ ||B||_F^2.$$

This completes our proof. □

The following theorem states that when a unitary matrix is multiplied to a matrix, the 2-norm and the Frobenius norm are unchanged.

Theorem 11 *For any $m \times n$ matrix A, $m \times m$ unitary matrix Q_1 and $n \times n$ unitary matrix Q_2, we have*

$$||A||_2 = ||Q_1 A||_2 = ||AQ_2||_2, \quad ||A||_F = ||Q_1 A||_F = ||AQ_2||_F.$$

The 1-norm and the ∞-norm of a matrix have explicit formulas. The following theorem states that the 1-norm is the maximum of column sum, and the ∞-norm is the maximum row sum.

Theorem 12 *Let a_{ij} be the (i,j) entry of an $m \times n$ matrix A, then*

$$\begin{aligned} ||A||_1 &= \max_{1 \le j \le n} \sum_{i=1}^{m} |a_{ij}|, \\ ||A||_\infty &= \max_{1 \le i \le m} \sum_{j=1}^{n} |a_{ij}|. \end{aligned}$$

The matrix 2-norm is somewhat more complicated. We first establish the following lemma:

Lemma 4 *Let d_1, d_2, ..., d_m be non-negative numbers, then*

$$\sup_{x \neq 0} \frac{d_1|x_1|^2 + d_2|x_2|^2 + ... + d_m|x_m|^2}{|x_1|^2 + |x_2|^2 + ... + |x_m|^2} = \max_{1 \le j \le m} d_j.$$

If we introduce a diagonal matrix $D = \mathrm{diag}(d_1, d_2, ..., d_m)$, the left hand side above is the so-called **Rayleigh quotient**

$$r_D(x) = \frac{x^* D x}{x^* x}.$$

Thus,

$$\sup_{0 \neq x \in \mathbb{C}^m} r_D(x) = \sup_{x \in \mathbb{C}^m,\ ||x||_2 = 1} x^* D x = \max_{1 \le 1 \le m} d_i.$$

Now, we can establish a result on the 2-norm of an $m \times m$ Hermitian matrix.

Theorem 13 *Let A be an $m \times m$ Hermitian matrix, λ_1, λ_2, ..., λ_m be the eigenvalues of A, then*

$$||A||_2 = \max_{1 \le j \le m} |\lambda_j|.$$

Proof: We start with the eigenvalue decomposition

$$A = Q \Lambda Q^*,$$

where Q is unitary and Λ is the real diagonal matrix of the eigenvalues. Therefore,

$$||A||_2 = ||\Lambda||_2 = \sup_{z \neq 0} \frac{||\Lambda z||_2}{||z||_2} = \sup_{z \neq 0} \sqrt{\frac{z^* \Lambda^2 z}{z^* z}} = \max_{1 \le j \le m} |\lambda_j|.$$

This completes the proof. □

Finally, we have the following result for the 2-norm of a general $m \times n$ matrix.

Theorem 14 *For any complex matrix A, we have*

$$||A||_2 = \text{ square root of the largest eigenvalue of } A^*A.$$

Proof: We consider an $m \times n$ matrix A. From the definition, we have

$$||A||_2 = \sup_{x \neq 0} \frac{||Ax||_2}{||x||_2} = \sup_{x \neq 0} \sqrt{\frac{x^*A^*Ax}{x^*x}}.$$

Let the eigenvalue decomposition of A^*A be

$$A^*A = UDU^*,$$

where U is unitary, $D = \text{diag}(d_1, d_2, ..., d_n)$ is the diagonal matrix of the eigenvalues of A^*A (which must be non-negative). We let $z = U^*x$, then $x^*x = z^*z$ and

$$||A||_2 = \sup_{z \neq 0} \sqrt{\frac{z^*Dz}{z^*z}} = \max_{1 \leq j \leq n} \sqrt{d_j}.$$

1.5 Singular value decomposition

Let A be $m \times n$ and complex, we have the following singular value decomposition (SVD)

$$A = U\Sigma V^*$$

where U is an $m \times m$ unitary matrix, V is an $n \times n$ unitary matrix, Σ is an $m \times n$ diagonal matrix given as

$$\Sigma = \begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \ddots \end{bmatrix}$$

where $\sigma_1 \geq \sigma_2 \geq ... \geq 0$. Notice that Σ is not a square matrix in general. The diagonal entries of Σ are the singular values of A.

To give a proof for the existence of SVD, we need the following result: *Let*

$$C = \begin{bmatrix} \sigma_1 & w^* \\ 0 & B \end{bmatrix}$$

be an $m \times m$ matrix, where B is $(m-1) \times (n-1)$, σ_1 is a scalar and w^* is a row vector of length $m-1$. If $||C||_2 = |\sigma_1|$, then $w^* = 0$.

This result can be proved from

$$|\sigma_1| = ||C||_2 \geq \frac{||Cx||_2}{||x||_2}$$

for the particular choice of

$$x = \begin{bmatrix} \bar{\sigma}_1 \\ w \end{bmatrix}.$$

The right hand side can be proved to be greater than or equal to $\sqrt{|\sigma_1|^2 + ||w||_2^2}$.

To prove the SVD, we start with $\sigma_1 = ||A||_2$ and argue that there must be a unit vector, say v_1, such that

$$||Av_1||_2 = \sigma_1.$$

This is so, because

$$\sigma_1 = \sup_{||x||_2=1} ||Ax||_2$$

and the maximum must be reached for some unit vector. Now, for the unit vector

$$u_1 = \frac{Av_1}{||Av_1||_2}$$

we have $Av_1 = \sigma_1 u_1$. Now, we can find unitary matrices U_1 and V_1, such that u_1 and v_1 are their first columns. If we write down $U_1^* A V_1$, we realize that the first column must be σ_1 followed by zeros. That is

$$U_1^* A V_1 = \begin{bmatrix} \sigma_1 & w^* \\ 0 & B \end{bmatrix} = C.$$

It is not difficult to realize that the matrix 2-norm is un-changed when unitary matrices are multiplied from left or right. This leads to

$$||C||_2 = ||A||_2 = \sigma_1.$$

Therefore, $w^* = 0$. Now the matrix B has one row and one column less than matrix A. By induction, we assume that the SVD of B is already calculated:

$$B = U_2 \Sigma_2 V_2^*.$$

We therefore have

$$A = U_1 \begin{bmatrix} 1 & \\ & U_2 \end{bmatrix} \begin{bmatrix} \sigma_1 & \\ & \Sigma_2 \end{bmatrix} \begin{bmatrix} 1 & \\ & V_2^* \end{bmatrix} V_1^*.$$

This gives rise to the SVD decomposition $A = U\Sigma V^*$, since U and V defined below are also unitary:

$$U = U_1 \begin{bmatrix} 1 & \\ & U_2 \end{bmatrix}, \quad V^* = \begin{bmatrix} 1 & \\ & V_2^* \end{bmatrix} V_1^*, \quad \Sigma = \begin{bmatrix} \sigma_1 & \\ & \Sigma_2 \end{bmatrix}.$$

1.6 More on SVD

Once the SVD of a matrix is calculated, many mathematical properties of the matrix are revealed.

1. The rank of the matrix A equals to the number of non-zero singular values of A.

2. Suppose the matrix A has rank r and that the singular value decomposition of A is $A = U\Sigma V^*$. We write down the matrix U by its columns $U = [u_1, u_2, ..., u_m]$ and $V = [v_1, v_2, ..., v_n]$. Then range(A) is spanned by the r vectors u_1, u_2, ..., u_r. Here range(A) is the set of vectors that can be written as a linear combination of the columns of A. It is also called the column space of A. This results mean: if a vector b can be written as a sum (with suitable coefficients) of the columns of A, i.e, $b = Ax$ for some x, then b can be written as a sum (with suitable coefficients) of u_1, u_2, ..., u_r.

3. The null space of A is all these vectors y, such that $Ay = 0$. If $rank(A) = r$, then null(A) is spanned by v_{r+1}, v_{r+2}, ..., v_n.

4. In the proof of SVD, we already see that $||A||_2 = \sigma_1$. We also have

$$||A||_F = \sqrt{\sigma_1^2 + \sigma_2^2 + ... + \sigma_r^2}$$

assuming $rank(A) = r$.

5. The non-zero singular values of A are the square roots of the eigenvalues of A^*A and AA^*.

6. If A is Hermitian, i.e., $A = A^*$, the singular values of A are the absolute values of the eigenvalues of A.

7. If A is square, say $m \times m$, then $|\det(A)| = \sigma_1\sigma_2...\sigma_m$. This is so, because $|\det(U)| = |\det(V)| = 1$.

8. If A has rank r, then

$$A = \sum_{j=1}^{r} \sigma_j u_j v_j^*.$$

That is, A is a sum of r rank-1 matrices.

9. Let $A_k = \sum_{j=1}^{k} \sigma_j u_j v_j^*$, this matrix is a sum of k rank-1 matrices. For $k \leq r = rank(A)$, we know that A_k must have rank k. Now the matrix A_k is the closest rank-k matrix to A, under the matrix 2-norm and the Frobenius norm. Let B be an arbitrary matrix of the same size as A and assume $rank(B) \leq k$, then

$$||A - B||_2 \geq ||A - A_k||_2 = \sigma_{k+1}$$
$$||A - B||_F \geq ||A - A_k||_F = \sqrt{\sigma_{k+1}^2 + ... + \sigma_r^2}$$

1.7 Exercises

1. Let A be a non-singular $m \times m$ upper triangular matrix. Show that A^{-1} is also upper triangular.

2. Show that if a matrix A is both triangular and unitary, then it is diagonal.

3. What can be said about the eigenvalues of a unitary matrix?

4. Let S be an $m \times m$ skew-hermitian, i.e. $S^* = -S$.

   - Show that the eigenvalues of S are pure imaginary.
   - Show that $I - S$ is nonsingular.
   - Show that $Q = (I - S)^{-1}(I + S)$ is unitary.

5. Let u and v be m-vectors and $A = I + uv^*$. Show that if A is nonsingular, then $A^{-1} = I + \alpha uv^*$ for some scalar α. Find an expression for α. For what u and v is A singular. If it is singular, what is $null(A)$?

6. Let $||\cdot||$ be a "induced matrix norm" for $m \times m$ matrices. Show that $||A|| \geq |\lambda|$ if λ is an eigenvalue of A.

7. Let x be a m-vector, A be an $m \times n$ matrix, show that

   - $||x||_\infty \leq ||x||_2 \leq \sqrt{m}||x||_\infty$.
   - $\frac{1}{\sqrt{n}}||A||_\infty \leq ||A||_2 \leq \sqrt{m}||A||_\infty$.

8. Let u and v be two m-vectors and $E = uv^*$, show that $||E||_2 = ||u||_2||v||_2$.

9. Determine SVDs of the following matrices (by hand calculation):

$$(a)\begin{pmatrix} 3 & 0 \\ 0 & -2 \end{pmatrix}, (b)\begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix}, (c)\begin{pmatrix} 0 & 2 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, (d)\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}, (e)\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}.$$

10. Consider the matrix

$$A = \begin{pmatrix} -2 & 11 \\ -10 & 5 \end{pmatrix}.$$

    Determine, on paper, a real SVD of A. What are the 1-, 2-, ∞-, and Frobenius norms of A?

Chapter 2

QR Factorization and Least Squares

2.1 Householder reflections

Given a column vector $x = (x_1, x_2, ..., x_m)^*$ in $\mathfrak{C}^m$, we can find a unitary matrix H, such that

$$Hx = \begin{bmatrix} \sigma \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Since $(Hx)^*(Hx) = x^*x = |\sigma|^2$, we have $|\sigma| = ||x||_2$.

It is very easy to show that if w is a unit vector, then $I - 2ww^*$ is unitary. If v is some vector, we can normalize v to get $w = v/||v||_2$. Therefore,

$$H = I - 2ww^* = I - \frac{2}{v^*v}vv^*$$

is a unitary matrix, for any vector v. If we try

$$Hx = (I - 2ww^*)x = \begin{bmatrix} \sigma \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

we get

$$x - \begin{bmatrix} \sigma \\ 0 \\ \vdots \\ 0 \end{bmatrix} = 2(w^*x)w$$

Define the above vector as v, we also require $w = v/||v||_2$, then

$$v = 2\frac{v^*x}{||v||_2}\frac{v}{||v||_2}$$

or $v^*v = 2v^*x$. This leads to

$$\overline{\sigma}x_1 = \sigma\overline{x}_1$$

That is, the number $\overline{\sigma}x_1$ must be real. Now, if $x_1 = |x_1|e^{i\theta}$, we must have

$$\sigma = \pm||x||_2 e^{i\theta}.$$

The matrix H constructed above is the Householder reflection for the vector x. Notice that there are in general two Householder reflections for a given vector x.

In the case that x_1 is real (i.e. $\theta = 0$ or $\theta = \pi$), we can take σ as $||x||_2$ or $-||x||_2$. Usually, we want to avoid a subtraction in computing the first component of v (which is $x_1 - \sigma$). Therefore, if $x_1 \geq 0$, we take $\sigma = -||x||_2$. Otherwise, we take $\sigma = ||x||_2$.

2.2 QR factorization

The QR factorization is $A = QR$, where Q is unitary and R is upper triangular. This is useful for least squares problems (in section 2.3) and for eigenvalue problems. A special case of the least squares problem is the linear system of equations $Ax = b$. Therefore, the QR factorization is also useful for solving $Ax = b$. This factorization can be calculated using Householder reflections. We find H_1, H_2, ..., H_k, such that

$$H_k H_{k-1} ... H_2 H_1 A = R$$

If the matrix A is $m \times n$, the integer k is $m - 1$ if $m \leq n$ and $k = n$ if $m > n$. Notice that A is a general $m \times n$ matrix. The matrix Q is actually $(H_k ... H_2 H_1)^{-1}$. Most of the time (for the purpose of solving linear system of equations, least squares problems and in other applications), you do not need to explicitly calculate the matrix Q. Since H_j is unitary for $j = 1, 2, ..., k$, we conclude that their product $H_k ... H_2 H_1$ is also unitary. Therefore, $(H_k ... H_2 H_1)^* = (H_k ... H_2 H_1)^{-1}$ is also unitary.

The matrix H_1 is the Householder reflection based on the first column of the matrix A. Therefore,

$$H_1 A = \begin{bmatrix} * & * & ... & * \\ 0 & * & ... & * \\ \vdots & \vdots & & \vdots \\ 0 & * & ... & * \end{bmatrix}$$

For this purpose, we take the first column of A as x, then define v by

$$v = x - \sigma e_1$$

where e_1 is the first column of the identity matrix, $\sigma = \pm||x||_2$, and define H_1 by

$$H_1 = I - \frac{2}{v^* v} v v^*.$$

Next, we work on the second column. But it is the second column of $H_1 A$, not the second column of A. Besides, we start with the second column of $H_1 A$ from row 2 to row m only. We do not want to touch the first row. We define x as the second column of $H_1 A$ from row 2 to row m. In MATLAB, this can be written as

$$x = (H_1 A)(2 : m, 2).$$

Then, we define v (which is a vector of length $m-1$) by the same formula

$$v = x - \sigma e_1$$

where e_1 is the first column of the $(m-1) \times (m-1)$ identity matrix, and we define $\tilde{H}_2$ by

$$\tilde{H}_2 = I - \frac{2}{v^* v} v v^*.$$

Since, we do not want to do anything for the first column, we set

$$H_2 = \begin{bmatrix} 1 & \\ & \tilde{H}_2 \end{bmatrix}.$$

Now, we use H_2 to multiply $H_1 A$, we should obtain

$$H_2 H_1 A = \begin{bmatrix} * & * & * & ... & * \\ 0 & * & * & ... & * \\ 0 & 0 & * & ... & * \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & * & ... & * \end{bmatrix}.$$

The third step is similar, we define x as the part of column 3 of $H_2 H_1 A$ from row 3 to row m, then define v and $\tilde{H}_3$ in a similar fashion. Finally, we define

$$H_3 = \begin{bmatrix} I_2 & \\ & \tilde{H}_3 \end{bmatrix}$$

where T_2 is the 2×2 identity matrix. Then, the matrix $H_3 H_2 H_1 A$ will have zero entries in the first three columns and below the diagonal.

If $m \leq n$, we stop at H_{m-1}. If $m > n$, we stop at H_n. This gives rise to our definition of k earlier.

In the special case that $m > n$, the upper triangular matrix R can be written as

$$R = \begin{bmatrix} R_1 \\ 0 \end{bmatrix}$$

where R_1 is a square $n \times n$ upper triangular matrix and 0 is a $(m-n) \times n$ zero matrix. This leads to

$$A = QR = Q_1 R_1$$

where Q_1 is a $m \times n$ matrix and it is the first n columns of Q. More precisely, let

$$Q = [q_1, q_2, ..., q_m]$$

where q_j is the j-th column of Q. Then

$$Q_1 = [q_1, q_2, ..., q_n].$$

We will call $A = Q_1 R_1$ the *reduced* QR factorization.

The reduced QR factorization can be calculated by the Gram-Schmidt orthogonalization process. This is a standard textbook material in linear algebra. There is a variant called modified Gram-Schmidt orthogonalization process which has a different computing priority (but does the same thing mathematically). The modified Gram-Schmidt process has better numerical properties. However, the QR process based on Householder reflections is supposed to be the best.

2.3 Least squares problems

Let A be a complex $m \times n$ matrix. We assume $m > n$ and rank$A = n$, then $Ax = b$ has m equations for n unknowns. Usually, the linear system $Ax = b$ has no solution (Sometimes, it does). However, we can look for the minimum

$$\min_{x \in \mathfrak{C}^n} ||Ax - b||_2.$$

The minimum always exists and we assume the minimum is reached at x_*. That is

$$\min_{x \in \mathfrak{C}^n} ||Ax - b||_2 = ||Ax_* - b||_2.$$

The main results concerning the least squares problem are:

- $b - Ax_*$ is orthogonal to range(A). Here, range(A) is the vector space spanned by the columns of A. In other words, a vector in range(A) is a linear combination of the columns of A. Alternatively, if $y \in$ range(A), then $y = Az$ for some $z \in \mathfrak{C}^n$. In particular, the columns of A themself are vectors in range(A). In any case, we have

  $$y^*(b - Ax_*) = 0$$

  if $y = Az$ for any $z \in \mathfrak{C}^n$.

- x_* can be computed by

  $$A^*Ax = A^*b.$$

- The vector b has an orthogonal projection in range(A), which is

  $$Ax_* = A(A^*A)^{-1}A^*b$$

  The matrix $P = A(A^*A)^{-1}A^*$ is the projection matrix.

Although we can solve x_* from $A^*Ax_* = A^*b$. It is usually not the most efficient method. Besides, in finite precision calculations, this method based on A^*A can give less accurate solutions. The method based on QR factorization is prefered.

We notice that

$$||Ax - b||_2^2 = ||QRx - QQ^*b||_2^2 = ||Rx - Q^*b||_2^2$$

As before, we let

$$R = \begin{bmatrix} R_1 \\ 0 \end{bmatrix}$$

where R_1 is $n \times n$ upper triangular. Since rank$(A) = n =$ rank$(R) =$ rank(R_1), the matrix R_1 is non-singular. Let us partition Q^*b as

$$Q^*b = \begin{bmatrix} \beta \\ \gamma \end{bmatrix}$$

where β is a vector of length n (we call it n-vector), and γ is a vector of length $m-n$. This leads to

$$||Rx - Q^*b||_2^2 = ||R_1x - \beta||_2^2 + ||\gamma||_2^2$$

In order to obtain a minimum, we can force the first term to be zero. There is nothing we can do for the second term. Therefore, x_* can be calculated from $R_1x_* = \beta$ and

$$\min_x ||Ax - b||_2 = ||\gamma||_2$$

2.4 Givens rotation

Another important orthogonal matrix is the Givens rotation. A general $m \times m$ Givens rotation is the same as the identity matrix, except four entries, say at (i,i), (i,j), (j,i) and (j,j). For a given angle θ and a pair of integers i and j, we have

$$G_{ij}(\theta) = \begin{bmatrix} 1 & & & & & & & & \\ & 1 & & & & & & & \\ & & \ddots & & & & & & \\ & & & c & & -s & & & \\ & & & & \ddots & & & & \\ & & & s & & c & & & \\ & & & & & & \ddots & & \\ & & & & & & & 1 & \\ & & & & & & & & 1 \end{bmatrix}$$

where $c = \cos(\theta)$, $s = \sin(\theta)$. The (i,i) and (j,j) entries are c and the (i,j) and (j,i) entries are $-s$ and s, respectively. (We assume $i < j$). For a given vector x, we can produce a zero at its j-th component, by finding θ, such that

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} x_i \\ x_j \end{bmatrix} = \begin{bmatrix} \sqrt{x_i^2 + x_j^2} \\ 0 \end{bmatrix}.$$

This can be done, if

$$c = \cos\theta = \frac{x_i}{\sqrt{x_i^2 + x_j^2}}, \quad s = \sin\theta = \frac{-x_j}{\sqrt{x_i^2 + x_j^2}}$$

The Given rotation can also be used to compute QR factorization. It is useful when A already has a lot of zeros below the main diagonal.

2.5 Exercises

1. Determine the (a) eigenvalues, (b) determinant, and (c) singular values of a Householder reflector.

2. Let A have the form

$$A = \begin{bmatrix} R \\ S \end{bmatrix}$$

where R is $n \times n$ upper triangular, and S is $(m-n) \times n$ and dense (assuming $m > n$). Describe an algorithm using Householder reflections for reducing A to upper triangular form. Your algorithm should not "fill in" the zeros in R and thus require fewer operations than the general algorithm for QR factorization.

3. If $A = R + uv^*$, where R is an upper triangular matrix, and u and v are column vectors, describe an efficient algorithm to compute the QR factorization of A. (Hint: use Given rotation).

4. Let $x \in \Re^m$ and let H be a Householder reflection such that $Hx = \pm||x||_2 e_1$. Let $G_{1,2}$, $G_{2,3}$, ..., $G_{m-1,m}$ be Givins rotations and let $G = G_{1,2}G_{2,3}...G_{m-1,m}$. Suppose $Gx = \pm||x||_2 e_1$. Must H equal G? You need to give a proof or a counterexample.

5. Let A be $m \times n$ and have full rank. Show that

$$\begin{bmatrix} I & A \\ A^* & 0 \end{bmatrix} \begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}$$

has a solution where x minimizes $||Ax - b||_2$.

6. Let A be $m \times n$, $(m < n)$ and of full rank (i.e. rank$(A) = m$). Then, the linear system $Ax = b$ has infinite many solutions. Actually, the general solution depends on $n - m$ arbitrary parameters. How do you find the unique solution with the minimum 2-norm?

7. Describe a method to solve the constrained least squares problem:

$$\min_{x \in P} ||Ax - b||_2$$

where

$$P = \{x \in \mathfrak{C}^n \mid c^*x = d\}$$

for a given vector c and a scalar d.

8. Write a MATLAB program for calculating the QR factorization of the following 10×10 matrix

$$A = \begin{bmatrix} 1 & 1 & ... & ... & 1 \\ 1 & 1 & ... & ... & 1 \\ & 2 & \ddots & & \vdots \\ & & \ddots & 1 & 1 \\ & & & 9 & 1 \end{bmatrix}$$

using Givens rotations. Submit the MATLAB program and a list for the diagonal entries of R.

Chapter 3

Linear System of Equations

In this chapter, we discuss numerical methods for solving the linear system of equations $Ax = b$. Although it is a very classical problem, numerical methods have special requirements. Two algorithms are presented. The first is the Gaussian elliminaiton with partial pivoting. The second method is for a symmetric positive definite matrix A. The method is the Cholesky decomposition. We also introduce the concept of *condition number*. In the last section, we develop the *backward stability analysis*. When you solve $Ax = b$ by some method using finite precision floating point numbers, you will get $\tilde{x}$ which is not the exact x. It is important to know how accurate $\tilde{x}$ is. It turns out that it depends on the condition number of A and it depends on whether or not your algorithm is *stable*.

3.1 Cholesky decomposition

A Hermitian matrix A is positive definite, if

$$x^* Ax > 0$$

for any non-zero vector x. Using the eigenvalue decomposition of A, we can prove that this is equivalent to the condition that all eigenvalues of A are positive.

Theorem 15 *Let A be a Hermitian matrix, then A is positive definite, if and only if all eigenvalues of A are positive.*

Proof: Assume A is positive definite. Let λ be an eigenvalue of A and $x \neq 0$ be the corresponding eigenvector. Thus,

$$Ax = \lambda x.$$

We can multiply x^*, thus

$$\lambda x^* x = x^* Ax > 0.$$

Therefore, $\lambda > 0$.

Assume all eigenvalues of A are positive. We have the eigenvalue decomposition:

$$A = Q\Lambda Q^*,$$

where Q is unitary and Λ is the diagonal matrix with the positive eigenvalues on the diagonal. Then, for any $x \neq 0$, we have

$$x^*Ax = x^*Q\Lambda Q^*x = y^*\Lambda y = \lambda_1|y_1|^2 + \lambda_2|y_2|^2 + ... + \lambda_m|y_m|^2 > 0,$$

where $y = Q^*x$ $ne0$ and λ_1, λ_2, ..., λ_m are the eigenvalues. □

Another condition is that A can be written as

$$A = SS^*$$

where S is a non-singular lower triangular matrix. In fact, we can insist that the diagonal entries of S are all positive. This is called the Cholesky decomposition.

Theorem 16 *A matrix A is Hermitian positive definite if and only if $A = SS^*$ for some non-singular lower triangular matrix S.*

Proof: If $A = SS^*$, then A is obviously Hermitian. Meanwhile, for any non-zero x, $S^*x = y$ is non-zero (because S is non-singular). Thus,

$$x^*Ax = x^*SS^*x = y^*y > 0.$$

Therefore, A is positive definite.

Next, we want to show that a Hermitian positive definite matrix always has a Cholesky decomposition. This is proved by induction. The case for 1×1 matrix is obvious. In this case, $A = a_{11}$ is positive. Then, $S = \sqrt{a_{11}}$. We assume that a Cholesky decomposition is available for any $(m-1) \times (m-1)$ Hermitian positive definite matrices. Let us write A as

$$A = \begin{bmatrix} a_{11} & w^* \\ w & K \end{bmatrix}$$

where w is a $m-1$ vector, K is $(m-1) \times (m-1)$ and Hermitian positive definite (why?), and $a_{11} > 0$ (why?). We then have

$$A = \begin{bmatrix} \sqrt{a_{11}} & \\ w/\sqrt{a_{11}} & I \end{bmatrix} \begin{bmatrix} 1 & \\ & K - ww^*/a_{11} \end{bmatrix} \begin{bmatrix} \sqrt{a_{11}} & w^*/\sqrt{a_{11}} \\ & I \end{bmatrix}.$$

It turns out that the matrix $B = K - ww^*/a_{11}$ is still Hermitian positive definite (Why?). Now, suppose we have a Cholesky decomposition for B, say $B = S_1S_1^*$, then $A = SS^*$ for

$$S = \begin{bmatrix} \sqrt{a_{11}} & \\ w/\sqrt{a_{11}} & S_1 \end{bmatrix}.$$

This completes the proof. □

This leads to the following algorithm for Cholesky decomposition:

for $k = 1, 2, ..., m$
  $a_{kk} := \sqrt{a_{kk}}$
  for $i = k+1, ..., m$
    $a_{ik} := a_{ik}/a_{kk}$

```
        end
        for j = k + 1, ..., m
            for i = j, ..., m
                a_ij := a_ij − a_ik ā_jk
            end
        end
    end
```

This algorithm transforms the lower triangular part of A to S. The above algorithm is backward stable (the concept will be introduced in section 5 of this chapter). Wilkinson first established the backward stability for computing Cholesky decomposition in 1968.

3.2 LU decomposition

The LU decomposition of a matrix A is the relationship

$$A = LU$$

where L is a unit lower triangular matrix, U is an upper triangular matrix. Unit lower means that the diagonal entries are all 1. Consider the matrix A written as

$$A = \begin{bmatrix} a_{11} & b^* \\ c & D \end{bmatrix}$$

If $a_{11} \neq 0$, we have

$$A = \begin{bmatrix} 1 & \\ c/a_{11} & I \end{bmatrix} \begin{bmatrix} a_{11} & b^* \\ 0 & D - cb^*/a_{11} \end{bmatrix}$$

If the matrix $B = D - \frac{1}{a_{11}} cb^*$ has a LU decomposition $B = L_1 U_1$, then $A = LU$, where

$$L = \begin{bmatrix} 1 & \\ c/a_{11} & L_1 \end{bmatrix}, \quad U = \begin{bmatrix} a_{11} & b^* \\ & U_1 \end{bmatrix}.$$

You can then write down an algorithm similar to the algorithm for Cholesky decomposition. We notice that if $a_{11} = 0$, we can not start the first step. In general, a matrix A may or may not have a LU decomposition. But for certain special matrices, LU decomposition is always possible.

3.3 Partial pivoting

For a general matrix, even if LU decomposition exists, it can lead to inaccurate solutions for the linear system $Ax = b$, if a small diagonal entries are encountered. The diagonal entries (a_{11} in A, the first entry of $D - \frac{1}{a_{11}} cb^*$ in the second step, etc) are called the pivots for LU decomposition. To obtain more reliable numerical solutions, we use row exchanges to find the largest entry in the column as the pivot. In the first step, we look for the largest entry in the first column of A (in absolute value), say $a_{p_1,1}$, then we exchange row 1 with row p_1. In the

second step, we exchange row 2 with row p_2 (if $p_2 > 2$). This is to move the largest entry in the first column of $D - \frac{1}{a_{11}} cb^*$ to the $(2,2)$ position in the original matrix.

The LU decomposition with partial pivoting computes $PA = LU$, where P is the permutation matrix that represents all the row exchanges. Since L is a unit lower triangular matrix, the diagonal entries of L always equal to 1 and they do not need to be stored. Thus, we use the lower triangular part of A (without the diagonal) to store the matrix L and use the upper triangular part of A (including the diagonal) to store the matrix U. The matrix P is not explicitly formed. Instead, we use the integer vector p (length $n-1$) for the pivoting index. If for the 3rd column, we exchanged the 3rd row with the 5-th row, then p(3)=5. The algorithm is as follows:

For $k = 1, 2, ..., m-1$,

Find p_k*, such that*

$$|a_{p_k,k}| = \max\{|a_{kk}|, |a_{k+1,k}|, ..., |a_{mk}|\}.$$

If $p_k > k$*, swap the* k*-th row with the* p_k*-th row.*

Reset the k*-th column for matrix* L*.*

$$a_{ik} := a_{ik}/a_{kk} \quad \textit{for} \quad i = k+1, ..., m.$$

Update the trailing $(m-k) \times (m-k)$ *matrix:*

$$a_{ij} := a_{ij} - a_{ik}a_{kj} \quad \textit{for} \quad i, j = k+1, ..., m.$$

end.

Here is an example:

$$\begin{aligned} A &= \begin{pmatrix} 1 & 5 & 3 \\ 2 & 4 & -2 \\ 3 & 0 & -1 \end{pmatrix} \Longrightarrow \begin{pmatrix} 3 & 0 & -1 \\ 2 & 4 & -2 \\ 1 & 5 & 3 \end{pmatrix} \Longrightarrow \begin{pmatrix} 3 & 0 & -1 \\ 2/3 & 4 & -4/3 \\ 1/3 & 5 & 10/3 \end{pmatrix} \\ &\Longrightarrow \begin{pmatrix} 3 & 0 & -1 \\ 1/3 & 5 & 10/3 \\ 2/3 & 4 & -4/3 \end{pmatrix} \Longrightarrow \begin{pmatrix} 3 & 0 & -1 \\ 1/3 & 5 & 10/3 \\ 2/3 & 4/5 & -4 \end{pmatrix} \end{aligned}$$

Finally,

$$L = \begin{pmatrix} 1 & & \\ 1/3 & 1 & \\ 2/3 & 4/5 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 3 & 0 & -1 \\ & 5 & 10/3 \\ & & -4 \end{pmatrix}.$$

Also, we found $p_1 = 3$ and $p_2 = 3$. We can obtain matrix P as follows: $P = P_2 P_1$, where P_1 is the permutation matrix that exchanges the 1st and 3rd rows, P_2 is the permutation matrix that exchanges the 2nd and 3rd rows:

$$P_1 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad P_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \quad P = P_2 P_1 = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

If we start with a 3×3 identity matrix, exchange the 1st row with the 3rd row, then exchange the 2nd and 3rd row, we also get the right P.

To solve a system of equations $Ax = b$, we can use the decomposition $PA = LU$. First, we find Pb, then we solve $Ly = Pb$, finally, we solve $Ux = y$. Let us consider:

$$\begin{pmatrix} 1 & 5 & 3 \\ 2 & 4 & -2 \\ 3 & 0 & -1 \end{pmatrix} x = \begin{pmatrix} 4 \\ 10 \\ 7 \end{pmatrix}.$$

We have:

$$Pb = \begin{pmatrix} 7 \\ 4 \\ 10 \end{pmatrix}.$$

Now solve y based on:

$$\begin{pmatrix} 1 & & \\ 1/3 & 1 & \\ 2/3 & 4/5 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 4 \\ 10 \end{pmatrix}.$$

We get:

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 5/3 \\ 4 \end{pmatrix}.$$

Finally, we solve:

$$\begin{pmatrix} 3 & 0 & -1 \\ & 5 & 10/3 \\ & & -4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 5/3 \\ 4 \end{pmatrix}.$$

We get:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \\ -1 \end{pmatrix}.$$

3.4 Condition number

If an algorithm for $f(x)$ is backward stable, then it is a good algorithm. However, when it is used to compute a function $f(x)$, the result may still be inaccurate. Even though the computer result following the algorithm becomes $f(\tilde{x})$, for $\tilde{x}$ close to x. It is still possible that $f(\tilde{x})$ and $f(x)$ are not close. For example,

$$x = 1000.3, \quad y = 2000.4, \quad z = -3001.7$$

we approximate x, y and z by rounding to 4 decimal digits:

$$\tilde{x} = 1000, \quad \tilde{y} = 2000, \quad \tilde{z} = -3002.$$

The relartive errors for $\tilde{x}$, $\tilde{y}$ and $\tilde{z}$ are reasonably small. But

$$f(x, y, z) = x + y + z = -1, \quad f(\tilde{x}, \tilde{y}, \tilde{z}) = -2.$$

The relative error of $f(\tilde{x}, \tilde{y}, \tilde{z})$ is 100%. When this happens, we can not blame the roundings (to 4 digits) we used, or the algorithm we add the three numbers. It is simply a fact that this problem itself is sentitive to small errors in inputs. We will call such problem ill-conditioned. We will define a number called condition number. When the condition number is large, the problem itself (i.e. the function $f(x)$ itself, it has nothing to do with any algorithm) is ill-conditioned. Thus a small error in inputs will lead to large errors in the output. Ill-conditioned problems are difficult to calculate (to obtain accurate solution) because of its own nature. This has nothing to do with algorithms. It is a indication that the function is sensitive to small errors in inputs.

As an example, we consider the function

$$f(x) = x_1 + x_2 + x_3, \quad \text{for} \quad x = (x_1, x_2, x_3)$$

We assume $\tilde{x} = (\tilde{x}_1, \tilde{x}_2, \tilde{x}_2)$ is close to x. For this purpose, we use vector 1-norm. Therefore, we assume

$$\frac{||\tilde{x} - x||_1}{||x||_1} \leq \epsilon.$$

Here ϵ is just some small number. It has nothing to do with machine epsilon. Meanwhile, x_j, $\tilde{x}_j$ are just real numbers and they do not need to be floating point numbers. In this case, we have

$$|f(\tilde{x}) - f(x)| \leq ||\tilde{x} - x||_1$$

Thus,

$$\left| \frac{f(\tilde{x}) - f(x)}{f(x)} \right| \leq \kappa(x) \frac{||\tilde{x} - x||_1}{||x||_1}$$

where

$$\kappa(x) = \frac{||x||_1}{|f(x)|} = \frac{|x_1| + |x_2| + |x_3|}{|x_1 + x_2 + x_3|}$$

is the condition number.

For a more general function $f(x)$, we start with some norm for x and assume

$$\frac{||\tilde{x} - x||}{||x||} \leq \epsilon$$

then we attempt to find κ, such that

$$\frac{||f(\tilde{x}) - f(x)||}{||f(x)||} \leq \kappa\epsilon + O(\epsilon^2)$$

A norm for f is also needed, because f may be vector functions. Meanwhile, κ should be the smallest number (may depends on x) such that the above is valid. This leads to the following formal definition for the "relative" condition number:

$$\kappa = \lim_{\delta \to 0} \sup_{||\tilde{x} - x|| \leq \delta} \left(\frac{||f(\tilde{x}) - f(x)||}{||f(x)||} \cdot \frac{||x||}{||\tilde{x} - x||} \right)$$

If $f(x)$ is a scalar differentiable function of a scalar x, then

$$\kappa = \left| \frac{x f'(x)}{f(x)} \right|.$$

If $x \in \mathfrak{C}^n$, $f(x) \in \mathfrak{C}^m$ and f is differentiable, then

$$\kappa = \frac{||x|| \cdot ||J||}{||f(x)||}$$

where J is the Jacobian matrix

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

and $||J||$ is the induced matrix norm associated with the vector norm in $\mathfrak{C}^n$ used for x and the vector norm in $\mathfrak{C}^m$ for $f(x)$.

If f is the linear function $f(x) = Ax$ for an $m \times n$ matrix, then $J = A$ and the condition number is

$$\kappa = \frac{||A||\ ||x||}{||Ax||}.$$

Writing down $x = A^{-1}Ax$ and following $||x|| \leq ||A^{-1}||\ ||Ax||$, it is easy to show that

$$\kappa \leq ||A||\ ||A^{-1}||.$$

For a given matrix A, the number $||A||\ ||A^{-1}||$ shows up repeatedly for the conditioning of various problems with A. We thus call it the condition number of the matrix A, denoted by $\kappa(A)$.

For the linear system, $Ax = b$. If we regard A as fixed, b as the input and x the output, we have the function

$$x = A^{-1}b = f(b).$$

Thus, the condition number for this function is

$$\kappa = \frac{||b||\ ||A^{-1}||}{||x||} \leq \kappa(A).$$

For the linear system $Ax = b$, we can also take b as fixed, A as the input and x the output. This implies that we consider the function

$$x = A^{-1}b = f(A)$$

Assuming A is an $m \times m$ matrix, the input space is $\mathfrak{C}^{m \times m}$, and the output space is $\mathfrak{C}^m$. For $\tilde{A}$ close to A, let us assume that the solution is $\tilde{x}$, i.e. $\tilde{A}\tilde{x} = b$. Let $\tilde{A} = A + \delta A$, $\tilde{x} = x + \delta x$. We have $(A + \delta A)(x + \delta x) = b$, or

$$Ax + \delta Ax + A(\delta x) + ... = b$$

Therefore,

$$\delta x \approx A^{-1}(\delta A)x$$

This leads to

$$\frac{||\delta x||}{||x||} \leq ||A^{-1}||\ ||\delta A|| + h.o.t. = ||A^{-1}||\ ||A||\ \frac{||\delta A||}{||A||} + h.o.t.$$

Therefore, $\kappa(A) = ||A^{-1}||\ ||A||$ is the condition number for $f(A) = A^{-1}b$. That is, it is the condition number for the problem of linear system of equations with A as input and b as fixed constant vector.

3.5 Backward stability analysis

3.5.1 Floating point arithmetic

Most computations on computers use finite precision numbers. A single precision number uses 32 bits and a double precision number uses 64 bits. We have the machine epsilon defined as

$$\epsilon_M = \begin{cases} 2^{-24} & \text{for single precision} \\ 2^{-53} & \text{for double precision} \end{cases}$$

If x is real number, x will be rounded to $fl(x)$, where $fl(x)$ is the floating number closest to x. We have $|fl(x) - x| \leq \epsilon_M |x|$, or $fl(x) = x(1+\epsilon)$ with $|\epsilon| \leq \epsilon_M$.

Now we consider the arithmetic operations between two floating point numbers. Let x and y be floating point numbers (thus, $fl(x) = x$ and $fl(y) = y$), we can add them: $x + y$. The question is what will we get on a computer for $x + y$. It is not difficult to see that $x + y$ is not (in general) a floating point number. Therefore, we can not hope to get the exact answer. Most recent computers are made such that the answer is $fl(x + y)$. That, the computer result for $x + y$ is the nearest floating point number to the exact answer. This is clearly the best we can hope for. However, this does not mean that the computer will actually add the two numbers exactly, then find the closest floating point number. In any case, for other basic operations between two floating point numbers (subtraction, multiplication and division), we also assume that is true. Namely, the computer result for $x * y$, $x - y$ and x/y are $fl(x * y)$, $fl(x - y)$ and $fl(x/y)$, respectively.

3.5.2 Backward stability

It is important to notice that if a calculation involves more than 1 operations, we will not get the nearest floating point number to the exact answer. For example, if x, y and z are all floating point numbers and we try to add them together on a computer. The computer output for $x + y + z$ is not (in general) $fl(x + y + z)$. What we get will depend on how we add them. The computer result for $(x + y) + z$ is usually different from that of $x + (y + z)$. In general, the computer result to calculate a function $f(x)$ (where x may be vector or matrix, involving floating point numbers) depends on your algorithm. An algorithm is like a computer program, it specifies a sequence of calculations to find $f(x)$. Since we can not get $fl(f(x))$ (which is the best you can hope for using floating point numbers), what can we say about the computer result for $f(x)$ based on some algorithm, say T? We will denote this computer result by $\tilde{f}_T(x)$. In may instances, we can prove that

$$\tilde{f}_T(x) = f(\tilde{x})$$

for $\tilde{x}$ close to x. This means that the computer result based on algorithm T is the exact result of a slightly different input. In this case, we should be happy with the algorithm T. We say that the algorithm T for computing the function $f(x)$ is backward stable.

Consider the function $f(x, y, z) = x + y + z$, where x (now a scalar), y and z are floating point numbers. If the algorithm T is $f(x, y, z) = (x + y) + z$, then we get

$$\tilde{f}_T(x, y, z) = fl(fl(x + y) + z).$$

To prove backward stability, we use $fl(a) = a(1 + \epsilon)$ for $|\epsilon| \leq \epsilon_M$. Here,

$$\tilde{f}_T(x, y, z) = [(x + y)(1 + \epsilon_1) + z](1 + \epsilon_2)$$

for $|\epsilon_j| \leq \epsilon_M$. Therefore,

$$\tilde{f}_T(x, y, z) = \tilde{x} + \tilde{y} + \tilde{z}$$

with

$$\begin{aligned} \tilde{x} &= x(1 + \epsilon_1)(1 + \epsilon_2) \\ \tilde{y} &= y(1 + \epsilon_1)(1 + \epsilon_2) \\ \tilde{z} &= z(1 + \epsilon_2). \end{aligned}$$

Obviously,

$$\left|\frac{\tilde{x} - x}{x}\right| \leq 2\epsilon_M + O(\epsilon_M^2), \qquad \left|\frac{\tilde{y} - y}{y}\right| \leq 2\epsilon_M + O(\epsilon_M^2), \qquad \left|\frac{\tilde{z} - z}{z}\right| \leq \epsilon_M.$$

Back to the general function $f(x)$. To prove backward stability, we need to show that $\tilde{f}_T(x) = f(\tilde{x})$. In the above example, the vector x is replaced by three scalars. The requirement that $\tilde{x}$ is close to x is replaced by the conditions that each scalar with a tilde is close to the original one without the tilde. We observe that "close" is measured by a relative error bounded by a constant multiplied by machine epsilon. For a general case, we may not always want to do this for each scalar in x. It is often easier to show that

$$\frac{||\tilde{x} - x||}{||x||} \leq C\epsilon_M + O(\epsilon_M^2)$$

using a suitable vector norm (or a matrix norm, if x is a matrix). The constant C should be independent of x and should not be too large. In any event, $C\epsilon_M$ most be much smaller than 1. When x is a vector (matrix), C may depends the length (sizes) of x, say n. If C depends on n like $2n^2 + 10n$, then it is not too bad. We will still consider this as backward stable. On the other hand, if C is like 2^n, then, we can no longer accept this as backward stable. Since for single precision, $\epsilon_M = 2^{-24}$. If $C = 2^n = 2^{24}$ for $n = 24$, we may have the relative error (left hand side) around 1. Thus, we can not say $\tilde{x}$ is close to x anymore.

3.5.3 Backward stability of triangular solver

To solve a linear system, $Ax = b$, we are essentially calculating the function $f(A, b) = A^{-1}b$. In this case, the inputs are the matrix A and vector b. The output is $x = A^{-1}b$. We will consider the special case, where the coefficient matrix is upper triangular, say R. We assume R is $m \times m$ and we have

$$r_{ij} = 0, \quad \text{if} \quad i > j.$$

We also assume that the matrix R is invertible. Thus, $r_{jj} \neq 0$ for $j = 1, 2, ..., m$.

There is a trivial algorithm for solving $Rx = b$. It is called back substitution (BS). From the last equation

$$r_{mm}x_m = b_m$$

we can solve x_m, then we substitute into equation $m-1$:

$$r_{m-1,m-1}x_{m-1} + r_{m-1,m}x_m = b_{m-1}$$

and we can solve x_{m-1}. We continue to solve x_{m-2}, ..., x_2, and eventially, we use the first equation to solve x_1. This is a well defined algorithm. We can prove that this algorithm, say BS, is backward stable.

For this purpose, we actually assume that the entries of R and b are floating point numbers. The computer result following algorithm BS will be denoted as

$$\tilde{x} = \tilde{f}_{BS}(A, b)$$

The point here is to prove that

$$\tilde{x} = \tilde{f}_{BS}(R, b) = f(\tilde{R}, \tilde{b})$$

for $\tilde{R}$ close to R and $\tilde{b}$ close to b. In fact, we can take $\tilde{b} = b$. The main theorem states that

$$\left| \frac{\tilde{r}_{ij} - r_{ij}}{r_{ij}} \right| \leq c_{ij}\epsilon_M + O(\epsilon_M^2).$$

The coefficients c_{ij} are the entries of the following matrix

$$C = \begin{bmatrix} m & 1 & 2 & ... & ... & m-2 & m-1 \\ & m-1 & 1 & ... & ... & m-3 & m-2 \\ & & \ddots & & & \vdots & \vdots \\ & & & 4 & 1 & 2 & 3 \\ & & & & 3 & 1 & 2 \\ & & & & & 2 & 1 \\ & & & & & & 1 \end{bmatrix}.$$

3.6 Exercises

1. Let A be the following symmetric matrix

$$A = \begin{bmatrix} 4 & 2 & -2 & 2 \\ 2 & 2 & 1 & 1 \\ -2 & 1 & 14 & -1 \\ 2 & 1 & -1 & \beta \end{bmatrix}$$

find the values of β, such that A is positive definite.

2. Let A be a symmetric positive definite matrix, show that A has a decomposition:

$$A = UU^T$$

where U is *upper* triangular with positive diagonal entries. Describe an algorithm for calculating the matrix U.

3. Let A be a tridiagonal matrix, say

$$A = \begin{bmatrix} a_1 & c_1 & & \\ b_1 & a_2 & \ddots & \\ & \ddots & \ddots & c_{m-1} \\ & & b_{m-1} & a_m \end{bmatrix}$$

Thus, A can be stored in three vectors. Describe a method to calculate the LU decomposition of A, using only three vectors. For LU decomposition of A with partial pivoting, how many vectors are needed?

4. Gaussian elimination can be used to compute the inverse A^{-1} of a nonsingular matrix A, (say $m \times m$), though it is rarely necessary to do so.

   (a) Describe an algorithm for computing A^{-1} by solving m system of equations, and show that its asymptotic operation count is $8m^3/3$ flops.

   (b) Describe a variant of your algorithm, taking advantage of sparsity, that reduces the operation count to $2m^3$ flops.

   (c) Suppose one wishes to solve n system of equations $AX = B$, where B is $m \times n$. What is the asymptotic operation count for doing this (i) directly from LU decomposition and (ii) with a preliminary computation of A^{-1}.

5. Let L be a unit lower triangular matrix (lower triangular and diagonal entries are all 1), consider the problem of solving $Lx = b$. Describe an algorithm for this (it should be forward substitution) and show that this algorithm is backward stable for $m = 4$, where L is $m \times m$. Namely, the computer result $\tilde{x}$ (obtained using floating point operations) satisfies $\tilde{L}\tilde{x} = \tilde{b}$. Give a precise definition of $\tilde{L}$ and $\tilde{b}$, and show that $\tilde{L}$ is close to L and $\tilde{b}$ is close to b. Is it possible to prove backward stability with $\tilde{b} = b$?

6. Consider the Cholesky decomposition of a 2×2 real symmetric positive definite matrix

$$A = \begin{bmatrix} a & b \\ b & c \end{bmatrix} = SS^T = \begin{bmatrix} \sqrt{a} & \\ b/\sqrt{a} & \sqrt{c - b^2/a} \end{bmatrix} \begin{bmatrix} \sqrt{a} & b/\sqrt{a} \\ & \sqrt{c - b^2/a} \end{bmatrix}.$$

Let s_{ij} be the (i, j) entry of S, we use the following algorithm to calculate S:

$$s_{11} = \sqrt{a}, \quad s_{21} = b/s_{11}, \quad s_{22} = \sqrt{c - s_{21}^2}.$$

Show that this algorithm is backward stable. Namely, the computer result (using floating point operations) $\tilde{S}$ satisfies $\tilde{S}\tilde{S}^T = \tilde{A}$ for some $\tilde{A}$ close to A.

Chapter 4

Matrix Eigenvalue Problems

In this chapter, we study numerical methods for matrix eigenvalue problems. The first section explains why the standard linear algebra method for eigenvalue problems has difficulties with finite precision calculations. Some relative elementary methods are dsceribed in section 2. The next three sections present the pwerful method QR method. The last section gives an efficient new method for symmetric tridiagonal matrices.

4.1 Introduction

Let A be a square matrix, an eigenvalue λ is a solution of

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

Corresponding to an eigenvalue λ, we have an eigenvector x which is a non-zero solution of

$$(A - \lambda I)x = 0, \quad \text{or} \quad Ax = \lambda x.$$

The eigenvectors are determined up to an arbitrary constant. This chapter briefly describes some algorithms for computing the eigenvalues and eigenvectors.

The characteristic polynomial is

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

Classical linear algebra textbooks usually suggest that you can find the polynomial

$$p(\lambda) = \lambda^n + c_{n-1}\lambda^{n-1} + ... + c_1 x + c_0$$

then find the zeros of $p(\lambda)$ for the eigenvalues. This works for small (say, 2×2 or 3×3) matrices. But remember that we are doing numerical computations. For large matrices, we only find approximate values for the polynomial coefficients c_{n-1}, ..., c_1, c_0. What is surprising is that **a small error in polynomial coefficients can lead to very large error in the zeros of the polynomial.**

Wilkinson's Example: For $n = 20$, $p(\lambda) = (\lambda - 1)(\lambda - 2)...(\lambda - 20)$ can be regarded as the characteristic polynomial of the diagonal matrix

$$A = \text{diag}\{1, 2, ..., 20\}.$$

We can write down c_0 and c_{19},

$$p(\lambda) = \lambda^{20} - 210\lambda^{19} + ... - 20!$$

Now, let us introduce a small *error* to the coefficient c_{19}, say $\epsilon = 10^{-9}$, and consider the polynomial

$$\tilde{p}(\lambda) = \lambda^{20} + (-210 + \epsilon)\lambda^{19} + ... - 20! = (\lambda - 1)(\lambda - 2)...(\lambda - 20) + \epsilon\lambda^{19}$$

Now, if we try to solve $\tilde{p}(\lambda) = 0$, we will get 3 *complex conjugate* pairs. In particular, the original zeros 15, 16 now becomes

$$15.457790724 \pm 0.899341526i$$

The conclusion is that the zeros of a polynomial are very sensitive to the its coefficients. If we only calculate the polynomial coefficients approximately, we can not have accurate eigenvalues in general. In other words, the approach based on characteristic polynomial is bad.

4.2 Power, inverse power and Rayleigh quotient iterations

Power iterations: Let A be an $n \times n$ matrix, λ_1, λ_2, ..., λ_n be its eigenvalues. Let λ_1 be the largest (actually dominant) eigenvalue in absolute value. That is

$$|\lambda_1| > |\lambda_j| \quad \text{for} \quad j = 2, 3, ..., n.$$

The power iteration method can be used to calculate the eigenvector corresponding to λ_1.

- Set x_0 as an arbitrary vector (initial guess);
- For $k = 1, 2, ...,$

$$x_k = \frac{Ax_{k-1}}{||Ax_{k-1}||}$$

The basic theory is that for almost any initial vector x_0, the sequence $\{x_k\}$ "converges" to a unit eigenvector:

$$\lim_{k\to\infty} [Ax_k - \lambda_1 x_k] = 0$$

Inverse Power Iterations: Let A be $n \times n$ and non-singular, λ_1, $\lambda - 2$, ..., λ_n be the eigenvalues of A. We assume

$$0 < |\lambda_1| < |\lambda_j| \quad \text{for} \quad j \neq 1.$$

The inverse power iteration method can be used to find the eigenvector corresponding to λ_1. It is mathematically equivalent to the power method applied to A^{-1} (the eigenvalues of A^{-1} are $1/\lambda_1$, $1/\lambda_2$, ..., $1/\lambda_n$, with the dominant eigenvalue being $1/\lambda_1$). But we do not find A^{-1} first and then use the power iteration method.

- Set x_0 as an arbitrary non-zero vector (initial guess).
- For $k = 1, 2, 3, ...$, solve z from

$$Az = x_{k-1}$$

and set x_k as

$$x_k = \frac{z}{||z||}.$$

Notice that

$$x_k = \frac{(A^{-1})^k x_0}{||(A^{-1})^k x_0||}.$$

Rayleigh Quotient Iteration: If x is an eigenvector of the matrix A, how do you calculate the eigenvalue (when x is given)? We have

$$Ax = \lambda x, \quad \text{thus} \quad x^T Ax = \lambda x^T x.$$

Therefore,

$$\lambda = \frac{x^T Ax}{x^T x}.$$

The term $x^T Ax/(x^T x)$ is the Rayleigh quotient. The Rayleigh quotient iteration method is a procedure to calculate an eigenvalue and an eigenvector of a matrix. The basic idea is to apply the inverse power method to $A - \lambda I$, where λ is the approximate eigenvalue.

- Let $x_0 =$ arbitrary initial guess.
- For $k = 1, 2, 3, ...$,

$$\lambda = \frac{x_{k-1}^T A x_{k-1}}{x_{k-1}^T x_{k-1}}$$

solve z from

$$(A - \lambda I)z = x_{k-1}$$

set x_k as

$$x_k = \frac{z}{||z||}.$$

The method has a very fast convergence. But it is not known which eigenvalue/eigenvector pair will it converge to, for an arbitrary initial guess x_0. The method developed in the next a few sections allows us to find all the eigenvalues (and eigenvectors) of a matrix in a more systematic way.

4.3 Reduction to tridiagonal / upper Hessenberg matrices

The eigenvalue problem of a real symmetric matrix can be reduced to the eigenvalue problem of a real symmetric tridiagonal matrix. We use Householder reflections to find the matrix T (symmetric and tridiagonal) and an orthogonal matrix Q, such that

$$A = QTQ^T.$$

For a general (complex) matrix, we can find a unitary matrix U and an upper Hessenberg matrix H, such that $A = UHU^*$. We notice that (for the real symmetric case)

$$\det(\lambda I - A) = \det(Q(\lambda I - T)Q^T) = \det Q \det(\lambda I - T) \det Q^T = \det(\lambda I - T).$$

In fact, from $QQ^T = I$, we obtain $\det(Q)\det(Q)^T = (\det(Q))^2 = 1$ or $\det(Q) = \pm 1$.

To find the decomposition $A = QTQ^T$, we actually find

$$Q^T AQ = T$$

and Q is obtained as a product of Householder reflections. To illustrate this, we consider the case of a 4×4 matrix A

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

The main steps are:

1. Find H_1 (Householder reflection), such that

$$H_1 A H_1^T = \begin{bmatrix} * & * & 0 & 0 \\ * & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{bmatrix}.$$

2. Find H_2 (Householder reflection), such that

$$H_2 H_1 A H_1^T H_2^T = \begin{bmatrix} * & * & 0 & 0 \\ * & * & * & 0 \\ 0 & * & * & * \\ 0 & 0 & * & * \end{bmatrix} = T.$$

Now, the matrix Q is given by

$$Q^T = H_2 H_1, \quad \text{or} \quad Q = H_1^T H_2^T = H_1 H_2.$$

The matrix T is also symmetric, since

$$T^T = Q^T A^T (Q^T)^T = Q^T AQ = T.$$

For the matrix H_1, we construct this matrix by a Householder reflection that works on rows 2, 3, 4 (keep row 1 unchanged). This will produce two zeros in row 3 and row 4. What about H_1^T multiplied from the right side of the matrix? It will work on the columns 2, 3 and 4, and it will not change column 1. Let us write down the matrix A as

$$A = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ a_2 & * & * & * \\ a_3 & * & * & * \\ a_4 & * & * & * \end{bmatrix}.$$

We construct the 3×3 Householder reflection $\tilde{H}_1$ such that

$$\tilde{H}_1 \begin{bmatrix} a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} \sigma_1 \\ 0 \\ 0 \end{bmatrix}$$

where $\sigma_1 = \pm\sqrt{a_2^2 + a_3^2 + a_4^2}$ and

$$\tilde{H}_1 = I - \frac{2}{v_1^T v_1} v_1 v_1^T, \quad \text{for} \quad v_1 = \begin{bmatrix} a_2 - \sigma_1 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} a_2 \mp \sqrt{a_2^2 + a_3^2 + a_4^2} \\ a_3 \\ a_4 \end{bmatrix}.$$

Now the 4×4 matrix H_1 is given by

$$H_1 = \begin{bmatrix} 1 & \\ & \tilde{H}_1 \end{bmatrix}.$$

Now, we have

$$H_1 A H_1^T = \begin{bmatrix} a_1 & \sigma_1 & 0 & 0 \\ \sigma_1 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{bmatrix}.$$

Let us assume that $H_1 A H_1^T$ can be further written as

$$H_1 A H_1^T = \begin{bmatrix} a_1 & \sigma_1 & 0 & 0 \\ \sigma_1 & b_2 & b_3 & b_4 \\ 0 & b_3 & * & * \\ 0 & b_4 & * & * \end{bmatrix}.$$

Now we find H_2 based on the Householder reflection for row 3 and row 4. We have

$$H_2 H_1 A H_1^T H_2^T = \begin{bmatrix} a_1 & \sigma_1 & 0 & 0 \\ \sigma_1 & b_2 & \sigma_2 & 0 \\ 0 & \sigma_2 & * & * \\ 0 & 0 & * & * \end{bmatrix},$$

where $\sigma_2 = \pm\sqrt{b_3^2 + b_4^2}$ and

$$H_2 = \begin{bmatrix} 1 & & \\ & 1 & \\ & & \tilde{H}_2 \end{bmatrix}, \quad \tilde{H}_2 = I - \frac{2}{v_2^T v_2} v_2 v_2^T, \quad v_2 = \begin{bmatrix} b_3 - \sigma_2 \\ b_4 \end{bmatrix} = \begin{bmatrix} b_3 \mp \sqrt{b_3^2 + b_4^2} \\ b_4 \end{bmatrix}.$$

Next, we give some details for an efficient implementation of this tridiagonalization process. Let us denote the matrix A by

$$A = \begin{bmatrix} a_1 & \alpha_1^T \\ \alpha_1 & \hat{A} \end{bmatrix}$$

where $\alpha_1^T = (a_2, a_3, a_4)$, and

$$H_1 A H_1^T = \begin{bmatrix} a_1 & \beta_1^T \\ \beta_1 & A_1 \end{bmatrix}$$

where $\beta_1^T = (\sigma_1, 0, 0)$. The matrix $\hat{A}$ is the given 3×3 matrix obtained from A by retaining the last three rows and columns. The matrix A_1 is what we need to calculate. They are related by

$$A_1 = \tilde{H}_1 \hat{A} \tilde{H}_1^T.$$

Since $\tilde{H}_1$ has a special simple form, we can efficiently evaluate A_1. Let $\gamma = 2/(v_1^T v_1)$, we have $\tilde{H}_1 = I - \gamma v_1 v_1^T$ and

$$A_1 = (I - \gamma v_1 v_1^T)\hat{A}(I - \gamma v_1 v_1^T) = \hat{A} - \gamma \hat{A} v_1 \; v_1^T - \gamma v_1 \; v_1^T \hat{A} + \gamma^2 v_1 \; v_1^T \hat{A} v_1 \; v_1^T$$

This can be written as

$$A_1 = \hat{A} + g v_1^T + v_1 g^T$$

for

$$g = -\gamma u + \frac{\gamma^2 (v_1^T u)}{2} v_1 \quad \text{where} \quad u = \hat{A} v_1.$$

Thus, the evaluation of A_1 follows these steps:

- $u = \hat{A} v_1$
- $\tau = v_1^T u$,
- $g = -\gamma u + \frac{\gamma^2 \tau}{2} v_1$
- $A_1 = \hat{A} + g v_1^T + v_1 g^T$.

For a general $n \times n$ matrix, the first and last two steps require about $2n^2$ operations each. Each elements of A_1 requires 4 operations to calculate, but A_1 is a symmetric matrix and only about $n^2/2$ entries need to be calculated. The 2nd and 3rd steps require $O(n)$ operations. The total number of operations required for evaluating $H_1 A H_1^T$ is around $4n^2$. The whole process of reduction to a tridiagonal matrix requires about $\frac{4}{3}n^3$ operations.

4.4 The QR algorithm

The QR algorithm is a widely used method for calculating the eigenvalues and eigenvectors. Since a general real symmetric matrix A can be reduced to a symmetric tridiagonal matrix T, we concentrate on the eigenvalue problem of T here. The basic idea is the following transformation:

$$\begin{aligned} QR &= T - sI \\ \hat{T} &= sI + RQ. \end{aligned}$$

For a given tridiagonal matrix T, we choose a real number s, then find the QR factorization of the matrix $T - sI$. This gives $T = sI + QR$, but we calculate the new matrix $\hat{T} = sI + RQ$. Since in general $QR \neq RQ$, we expect $\hat{T} \neq T$. However, we have

- $\hat{T}$ is also symmetric tridiagonal;
- $\hat{T}$ has the same eigenvalues as T.

Since $\hat{T} = sI + Q^T QRQ = Q^T(sI + QR)Q = Q^T TQ$, we obtain the symmetry $\hat{T}^T = \hat{T}$ and the fact of same eigenvalues (see similar arguments in the previous section for A and T). The fact that $\hat{T}$ is also tridiagonal, can be proved.

If we denote the original matrix T (obtained in the previous section through the reduction from A) as T_0 and $\hat{T}$ by T_1, then we can repeat the above transformation for T_1 to obtain T_2, etc. That is, for $k = 0, 1, 2, ...$

- find a real number s_k (somehow), and find the QR factorization of $T_k - s_k I$,

$$QR = T_k - s_k I$$

- calculate T_{k+1}

$$T_{k+1} = s_k I + RQ.$$

The hope is that

$$T_k \to \begin{bmatrix} * & * & & & \\ * & \ddots & \ddots & & \\ & \ddots & \ddots & * & \\ & & * & * & 0 \\ & & & 0 & \lambda_1 \end{bmatrix} = \begin{bmatrix} \tilde{T}_0 & \\ & \lambda_1 \end{bmatrix}$$

as $k \to \infty$, where λ_1 is an eigenvalue of A (also T_0, T_1, ...), but it is not necessarily the largest or smallest eigenvalue. More precisely, we hope the (n, n) entry of T_k converges to an eigenvalue and the $(n-1, n)$ entry (which is the same as the $(n, n-1)$ entry) converges to 0. If this works out, we find one eigenvalue λ_1 and then we continue with the $(n-1) \times (n-1)$ matrix $\tilde{T}_0$.

The number s (or s_k) is called the shift. One possible choice is to choose s_k as the (n, n) entry of T_k. The Wilkinson shift is to choose s_k as **the eigenvalue of the trailing** 2×2 **matrix of** T_k **which is closer to the** (n, n) **entry.** More precisely, we denote

$$T_k = \begin{bmatrix} \alpha_1^{(k)} & \beta_1^{(k)} & & \\ \beta_1^{(k)} & \ddots & \ddots & \\ & \ddots & \alpha_{n-1}^{(k)} & \beta_{n-1}^{(k)} \\ & & \beta_{n-1}^{(k)} & \alpha_n^{(k)} \end{bmatrix}$$

Thus, s_k is an eigenvalue of $\begin{pmatrix} \alpha_{n-1}^{(k)} & \beta_{n-1}^{(k)} \\ \beta_{n-1}^{(k)} & \alpha_n^{(k)} \end{pmatrix}$. Since this matrix has two eigenvalues, we choose s_k to be the eigenvalue that is closer to $\alpha_n^{(k)}$.

4.5 Givens rotation

For given a and b, we can have a 2×2 orthogonal matrix

$$G = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$$

(where $c^2 + s^2 = 1$), such that

$$G \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} * \\ 0 \end{bmatrix}$$

One possible solution is

$$c = \frac{a}{\sqrt{a^2 + b^2}}, \quad s = \frac{b}{\sqrt{a^2 + b^2}}$$

In that case, we actually have

$$G \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sqrt{a^2 + b^2} \\ 0 \end{bmatrix}.$$

The orthogonal matrix G is called Givens rotation, since it can be regarded as a plane rotation of angle θ, where $c = \cos(\theta)$ and $s = \sin(\theta)$.

For a general vector x, say

$$x = \begin{bmatrix} \vdots \\ a \\ \vdots \\ b \\ \vdots \end{bmatrix}$$

where the i-th and j-th elements of x are a and b, respectively. We can take

$$G_{ij} = \begin{bmatrix} \ddots & & & & \\ & c & & s & \\ & & \ddots & & \\ & -s & & c & \\ & & & & \ddots \end{bmatrix},$$

such that

$$G_{ij}x = \begin{bmatrix} \vdots \\ \sqrt{a^2 + b^2} \\ \vdots \\ 0 \\ \vdots \end{bmatrix}$$

The matrix G_{ij} is the same as the identity matrix, except the four entries (i, i), (i, j), (j, i) and (j, j). The formulas for c and s are the same as before. The vector $G_{ij}x$ has the same elements as x, except the i-th and j-th elements.

The QR algorithm for matrix eigenvalue problem requires the QR factorization of a tridiagonal matrix $T - sI$ in each step. Since the tridiagonal matrix has many zero entries, it is important to minimize the required computation by taking advantage of this. Givens rotation can be used here.

Let us consider the following 4×4 case.

$$T - sI = \begin{bmatrix} * & * & & \\ * & * & * & \\ & * & * & * \\ & & * & * \end{bmatrix}.$$

In the first step, we produce a zero at the location $(2,1)$. This can be achieved by a Givens rotation for rows 1 and 2 (based on the entries at (1,1) and (2,1)), say G_{12}.

$$G_{12}(T-sI) = \begin{bmatrix} \spadesuit & \spadesuit & \spadesuit & \\ 0 & \spadesuit & \spadesuit & \\ & * & * & * \\ & & * & * \end{bmatrix},$$

where $\spadesuit$ is a new entry (in general non-zero) calculated in this step. Next, we use the two entries at (2,2) and (3,2) to define a Givens rotation and produce a zero at (3,2).

$$G_{23}G_{12}(T-sI) = \begin{bmatrix} * & * & * & \\ & \spadesuit & \spadesuit & \spadesuit \\ & 0 & \spadesuit & \spadesuit \\ & & * & * \end{bmatrix},$$

Finally, we use the two entries at (3,3) and (4,3) to define the Givens rotation G_{34}. We have

$$G_{34}G_{23}G_{12}(T-sI) = \begin{bmatrix} * & * & * & \\ & * & * & * \\ & & \spadesuit & \spadesuit \\ & & 0 & \spadesuit \end{bmatrix} = R$$

Now, we let $Q^T = G_{34}G_{23}G_{12}$ or $Q = G_{12}^T G_{23}^T G_{34}^T$, we have $T - sI = QR$.

This is followed by calculating the tridiagonal matrix $\hat{T}$ in $\hat{T} - sI = RQ = RG_{12}^T G_{23}^T G_{34}^T$. We can minimize our computation effort by multiplying G_{12}^T, G_{23}^T and G_{34}^T one-by-one. We have

$$R = \begin{bmatrix} * & * & * & \\ & * & * & * \\ & & * & * \\ & & & * \end{bmatrix}$$

Then

$$RG_{12}^T = \begin{bmatrix} \spadesuit & \spadesuit & * & \\ \spadesuit & \spadesuit & * & * \\ & & * & * \\ & & & * \end{bmatrix}$$

where $\spadesuit$ is a new entry calculated in this step. Next,

$$RG_{12}^T G_{23}^T = \begin{bmatrix} * & \spadesuit & 0 & \\ * & \spadesuit & \spadesuit & * \\ & \spadesuit & \spadesuit & * \\ & & & * \end{bmatrix}$$

Actually, because of symmetry, the $(1,2)$ entry must equal the $(2,1)$. Thus, only 4 entries have to be calculated in this step. Finally,

$$RG_{12}^T G_{23}^T G_{34}^T = \begin{bmatrix} * & * & & \\ * & * & \spadesuit & 0 \\ & * & \spadesuit & \spadesuit \\ & & \spadesuit & \spadesuit \end{bmatrix} = \hat{T} - sI$$

Again, the (2,3) entry should be equal to (3,2) entry and only 4 new entries need to be calculated.

In general, for an $n \times n$ matrix T, the new matrix $\hat{T}$ can be obtained in $O(n)$ operations. There are $n-1$ Givens rotations. When a Givens rotation (or its transpose) is multiplied from left (or right), only 5 (or 4) new entries have to be calculated, requiring $O(1)$ operations.

4.6 Divide and conquer method

This is a method for the eigenvalue problem of symmetric tridiagonal matrices. It was originally developed by J. J. M Cuppen in 1981 and later enhanced by others. It is suitable for calculating all eigenvalues and eigenvectors of a symmetric tridiagonal matrix. The QR method with Wilkonson's shift is an excellent method, but it is not as efficient as this method if all eigenvectors (and eigenvalues) are required. However, if only the eigenvalues are needed, the QR method is more efficient.

Let T be the following symmetric tridiagonal matrix:

$$T = \begin{bmatrix} a_1 & b_1 & & \\ b_1 & a_2 & \ddots & \\ & \ddots & \ddots & b_{m-1} \\ & & b_{m-1} & a_m \end{bmatrix},$$

we take $k \approx m/2$ and write T as

$$T = \begin{bmatrix} T_1 & \\ & T_2 \end{bmatrix} + b_k v v^T$$

where

$$T_1 = \begin{bmatrix} a_1 & b_1 & & \\ b_1 & a_2 & \ddots & \\ & \ddots & \ddots & b_{k-1} \\ & & b_{k-1} & a_k - b_k \end{bmatrix}, \quad T_2 = \begin{bmatrix} a_{k+1} - b_k & b_{k+1} & & \\ b_{k+1} & a_{k+2} & \ddots & \\ & \ddots & \ddots & b_{m-1} \\ & & b_{m-1} & a_m \end{bmatrix}$$

and

$$v^T = [0, ..., 0, 1, 1, 0, ..., 0].$$

The important feature here is that the matrix $b_k v v^T$ is a rank-1 matrix. Now, we assume that the eigenvalue decomposition of T_1 and T_2 are already calculated:

$$T_s = Q_s D_s Q_s^T, \quad s = 1, 2$$

where Q_1, Q_2 are orthogonal, D_1 and D_2 are diagonal. We have

$$T = Q \left[D + b_k z z^T \right] Q^T$$

where

$$Q = \begin{bmatrix} Q_1 & \\ & Q_2 \end{bmatrix}, \quad D = \begin{bmatrix} D_1 & \\ & D_2 \end{bmatrix}, \quad z = Q^T v.$$

Therefore, the eigenvalue problem of T is related to the eigenvalue problem of $D+b_k zz^T$. More precisely, let (λ, w) be an eigenpair of T, i.e., $Tw = \lambda w$, then

$$[D + b_k zz^T]u = \lambda u$$

where

$$u = Q^T w.$$

Let $D = \text{diag}(d_1, d_2, ..., d_m)$, $z^T = (z_1, z_2, ..., z_m)$, we have $b_k zz^T u = \lambda u - Du$ or

$$b_k(z^T u)z = \begin{bmatrix} (\lambda - d_1)u_1 \\ (\lambda - d_2)u_2 \\ \vdots \\ (\lambda - d_m)u_m \end{bmatrix}$$

Thus, the vector in the right hand side is proportional to z. Let

$$b_k(z^T u) = \gamma$$

then

$$\gamma z_j = (\lambda - d_j)u_j$$

Solve u_j and insert into $b_k(z^T u) = \gamma$, we get

$$b_k \sum_{j=1}^{m} \frac{z_j^2}{\lambda - d_j} = 1. \tag{4.1}$$

In summary, we can solve the nonlinear equation (4.1) for the eigenvalue λ. We should get m eigenvalues. For each eigenvalue, the corresponding unit eigenvector is given by

$$u = \frac{(\lambda I - D)^{-1} z}{||\text{above}||_2} \tag{4.2}$$

The method is supposed to be recursive. The eigenvalue problem for the smaller tridiagonal matrices T_1 and T_2 are obtained in a similar fashion by breaking the matrices into even smaller matrices. The total number of operations required for this method is $4m^3/3$. If the original matrix A is a general symmetric matrix, then a step to reduce A to tridiagonal T is required. It is also necessary to keep the orthogonal matrix for this reduction step and to transform the eigenvectors of T to those of A. This requires the additional of $8m^3/3$ operations. Therefore, the total required number of operations is $4m^3$ for computing all m eigenvalues and eigenvectors of A.

To solve m eigenvalues from (4.1), we need to assume that $z_j \neq 0$ for all j and the m diagonal entries of D are distinct. What happens if these conditions are not satisfied? If $z_j = 0$, then we get an eigenvalue $\lambda = d_j$, the corresponding eigenvector is e_j (the j-th column of the identity matrix). If some d_i equals d_j, then we introduce an orthogonal matrix G, such that

$$G(D + b_k zz^T)G^T = D + b_k(Gz)(Gz)^T$$

and the j-th (or i-th) component of Gz is zero. The matrix G is a Givens rotation for rows i and j and it differs from the identity matrix in only four entries. In particular, $GDG^T = D$.

A main problem is that the eigenvectors calculated from (4.2) are not necessarily orthogonal to each other, when the eigenvalues are calculated approximately and when some eigenvalues are close to each other. This is the reason that the method is not widely used for more than 10 years after the publication of Cuppen's paper. This problem was solved by Gu and Eisenstat (1995). While the calculated eigenvalues are approximate eigenvalues of $D + b_k zz^T$, they are the eact eigenvalues of $D + b_k \tilde{z}\tilde{z}^T$ for some $\tilde{z}$. From this, you can get the exact eigenvectors of $D + b_k \tilde{z}\tilde{z}^T$, which should be orthogonal to each other.

4.7 Exercises

1. Let A be a 3×3 real symmetric matrix. Let the eigenvalues of A be λ_1, λ_2 and λ_3, and the corresponding unit eigenvectors be q_1, q_2 and q_3. We assume that $\lambda_1 \neq \lambda_j$ for $j = 2, 3$. Furthermore, we can assume that the eigenvectors are orthogonal to each other. Let x_0 be a vector that is close to q_1:

$$x_0 = q_1 + \epsilon_2 q_2 + \epsilon_3 q_3,$$

where ϵ_1, ϵ_2 are small constants. Calculate x_1 by one step of the Rayleigh quotient method.

2. For the following symmetric matrix A, find a symmetric tridiagonal matrix T, such that A and T have the same eigenvalues.

$$A = \begin{bmatrix} 2 & 1 & 0 & 0 \\ 1 & 1 & 2 & 2 \\ 0 & 2 & 0 & -1 \\ 0 & 2 & -1 & 4 \end{bmatrix}.$$

3. (Relation between QR method and the Rayleigh quotient iteration) Let A be a real $n \times n$ symmetric (non-singular) matrix and $e_n = [0, 0, ..., 0, 1]^T$ be the last column of the identity matrix. Define vector x_1 by

$$Az = e_n, \quad x_1 = z/||z||.$$

Show that $x_1 = \pm q_n$, where q_n is the last column of the matrix Q and $A = QR$ is the QR factorization of A. If $\hat{A} = RQ$, show that the (n, n) entry of $\hat{A}$ is the Rayleigh quotient $x_1^T A x_1 = x_1^T A x_1/(x_1^T x_1)$.

4. Find the QR factorization of

$$T = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 2 & \epsilon \\ 0 & \epsilon & 0 \end{pmatrix}$$

then find $\hat{T} = RQ$.

5. Let A be a real symmetric $n \times n$ matrix, δA be a small real symmetric perturbation. Let the eigenvalues and the unit eigenvectors of A be λ_1, λ_2, ..., λ_n and q_1, q_2, ..., q_n, respectively. Assume further that λ_1 is a single eigenvalue. Let $\lambda_1 + \delta\lambda$ be the eigenvalue of $A + \delta A$ which is close to λ_1 and $q_1 + \delta q$ be the corrresponding eigenvector. Since eigenvectors are up to a constant, we assume δq is orthogonal to q_1. Find an approximate formula for $\delta\lambda$. If we write down $\delta q = \sigma_2 q_2 + \sigma_3 q_3 + ... + \sigma_n q_n$, find an approximate formula for σ_j.

6. Suppose we have a 3×3 matrix and wish to introduce zeros by left- and/or right-multiplications by unitary matrices Q_j such as Householder reflectors or Givens rotations. Consider the following three matrix structures:

$$(i)\begin{bmatrix} * & * & 0 \\ 0 & * & * \\ 0 & 0 & * \end{bmatrix} \quad (ii)\begin{bmatrix} * & * & 0 \\ * & 0 & * \\ 0 & * & * \end{bmatrix} \quad (iii)\begin{bmatrix} * & * & 0 \\ 0 & 0 & * \\ 0 & 0 & * \end{bmatrix}$$

   For each one, decide which of the following situations holds, and justify your claim.

   (a) Can be obtained by a sequence of left-multiplications by matrices Q_j;

   (b) Not (a), but can be obtained by a sequence of left- and right-multiplications by matrices Q_j;

   (c) Cannot be obtained by any sequence of left- and right-multiplications by matrices Q_j.

7. Let T be a real symmetric tridiagonal matrix, $T = QR$ is its QR factorization. Which entries of R and Q are in general non-zero? Show that $\hat{T} = RQ$ is also symmetric tridiagonal. Explain how Givens rotations or 2×2 Householder reflections can be used in the computation of the QR factorization of T, reducing the operation count to $O(m)$, if T is $m \times m$.

8. To compute the SVD of a matrix A, we can first reduce A to a bi-diagonal matrix. Starting with the QR factorization of the matrix A, i.e. $A = QR$, describe a method to factor the upper triangular matrix R as

$$R = U^* B W$$

   where B is bi-diagonal (only (i, i) and $(i, i+1)$ entries are non-zero) and W and U are unitary. If you have found the SVD of B, what is the SVD of A?

9. How many eigenvalues does

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 2 & 1 \\ 0 & 0 & 1 & 3 \end{pmatrix}$$

   have in the interval $[1, 2]$?

Chapter 5

Iterative Methods

In this chapter, we consider two iterative methods. The first section (conjugate gradient method) is about the linear system $Ax = b$, where A is large sparse symmetric positive definite. Here the size of A is large, thus a direct method by Cholesky decomposition is expensive. But A is sparse — only very few non-zeros for each row or each column, thus it is efficient to multiply A with any given vector. Iterative methods produce a sequence of approximate solutions x_1, x_2, ... In section 2, we introduce the Lanczos method, it is mainly used as an iterative method for the eigenvalue problems $Ax = \lambda x$, where A is a large sparse symmetric matrix. Actually, the Lanczos method is another method that reduces A to a symmetric trodiagonal matrix, but the process can be truncated to give approximate solutions of the eigenvalue problem.

5.1 The Conjugate Gradient Method

5.1.1 Background

The conjugate gradient method is a method for solving $Ax = b$, where A is a symmetric positive definite matrix. It is an iterative method that produces the sequence of approximations: x_1, x_2, x_3, ... Let A be $m \times m$, define the Krylov space by

$$\mathcal{K}_n =< b, Ab, A^2b, ..., A^{n-1}b >$$

This is the vector space spanned by the vectors b, Ab, ..., $A^{n-1}b$. It is the "column space" of the Krylov matrix

$$K_n = [b, Ab, A^2b, ..., A^{n-1}b].$$

The conjugate gradient method finds $x_n \in \mathcal{K}_n$ which solves the minimization problem

$$\min_{x \in \mathcal{K}_n} (x - x_*)^T A(x - x_*)$$

where $x_* = A^{-1}b$ is the exact solution. However, since

$$(x - x_*)^T A(x - x_*) = 2\phi(x) - b^T A^{-1} b, \quad \text{for} \quad \phi(x) = \frac{1}{2} x^T Ax - x^T b.$$

It is equivalent to say that x_n solves

$$\min_{x \in \mathcal{K}_n} \phi(x).$$

5.1.2 1-D optimization problem

For a given point x_{n-1} and a given direction p_{n-1}, we have a line that passes through x_{n-1} along the direction of p_{n-1}. The points on the line are given by

$$x_{n-1} + \alpha p_{n-1} \quad \text{for} \quad \alpha \in \mathbb{R}$$

Alternatively, we denote this line by

$$x_{n-1} + < p_{n-1} >$$

where $< p_{n-1} >$ is a 1-D vector space. We can minimize the function ϕ along this line

$$\min_{x \in x_{n-1} + < p_{n-1} >} \phi(x) = \min_{\alpha \in \mathbb{R}} \phi(x_{n-1} + \alpha p_{n-1})$$

Now, $\phi(x_{n-1} + \alpha p_{n-1})$ is a quadratic polynomial of α, its minimum is reached at

$$\alpha_n = \frac{r_{n-1}^T p_{n-1}}{p_{n-1}^T A p_{n-1}}$$

The minimum is obtained at $x_{n-1} + \alpha_n p_{n-1}$.

If x_{n-1} happens to be a conjugate gradient iteration, i.e., x_{n-1} minimizes $\phi(x)$ in $\mathcal{K}_{n-1}$. The above procedure gives

$$\tilde{x}_n = x_{n-1} + \alpha_n p_{n-1}$$

Of course, $\tilde{x}_n$ is usually not x_n which minimizes ϕ in $\mathcal{K}_n$. However, we will find a special way of choosing p_{n-1}, such that $\tilde{x}_n = x_n$.

5.1.3 Subspace minimization problem

We now look for $x_n \in \mathcal{K}_n$ such that

$$\phi(x_n) = \min_{x \in \mathcal{K}_n} \phi(x)$$

We assume that $\mathcal{K}_n$ has the following basis

$$p_0, p_1, ..., p_{n-1}$$

Now,

$$\min_{x \in \mathcal{K}_n} \phi(x) = \min_{\alpha_1, \alpha_2, ..., \alpha_n \in \mathbb{R}} \phi(\alpha_1 p_0 + \alpha_2 p_1 + ... + \alpha_n p_{n-1})$$

To find the minimum, we solve the system

$$\frac{\partial \phi}{\partial \alpha_i} = 0 \quad \text{for} \quad i = 1, 2, ..., n.$$

In fact,

$$\frac{\partial \phi}{\partial \alpha_i} = p_{i-1}^T A(\alpha_1 p_0 + \alpha_2 p_1 + ... + \alpha_n p_{n-1}) - p_{i-1}^T b$$

Therefore, we have the system for α_1, α_2, ..., α_n:

$$C \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix} = \begin{pmatrix} p_0^T b \\ p_1^T b \\ \vdots \\ p_{n-1}^T b \end{pmatrix}$$

where the $(i+1, j+1)$ entry of C is $p_i^T A p_j$.

If we assume that

$$p_i^T A p_j = 0 \text{ if } i \neq j$$

then the matrix C is diagonal and α_i is easily solved

$$\alpha_i = \frac{p_{i-1}^T b}{p_{i-1}^T A p_{i-1}}.$$

Furthermore, if we assume that $\{p_0, p_1, ..., p_{i-1}\}$ is a basis for $\mathcal{K}_i$ for all i (we only assume that for $i = n$ earlier), then

$$x_{n-1} = \alpha_1 p_0 + \alpha_2 p_1 + ... + \alpha_{n-1} p_{n-2}$$

is the conjugate gradient iteration that minimizes ϕ in $\mathcal{K}_{n-1}$ and

$$x_n = x_{n-1} + \alpha_n p_{n-1}$$

Indeed, you can show that the formula for α_n here is equivalent to the formula in last section. Therefore, the subspace minimization problem can be solved by 1-D optimization process under these assumptions on the search vectors p_0, p_1, ..., p_{n-1}.

5.1.4 Orthogonal residual

Clearly, we need a simple way to find these vectors p_0, p_1, ... It turns out that the following property on the residual is very important.

Let x_n be the n-th conjugate gradient iteration, $r_n = b - Ax_n$ be the residual, then

$$r_n \perp \mathcal{K}_n.$$

To prove this, we need a result for least squares problem. Let B be $m \times n$ (assuming $m \geq n$ and $\text{rank}(B) = n$), if y_n solves the least squares problem

$$||By_n - d||_2 = \min_{y \in \mathbb{R}^n} ||By - d||_2$$

then

$$By_n - d \perp By \quad \text{for any} \quad y \in \mathbb{R}^n$$

It can be easily shown by using the formula $y_n = (B^T B)^{-1} B^T d$.

Now, x_n solves the subspace minimization problem

$$\min_{x \in \mathcal{K}_n} \sqrt{(x - x_*)^T A (x - x_*)}$$

we need to change this to a standard least squares problem, then use the above result. We let

$$x = K_n y, \quad x_n = K_n y_n$$

where K_n is the Krylov matrix (which we assume is full rank) and also introduce a Cholesky decomposition $A = S^T S$ for some non-singular upper triangular matrix S, then

$$\sqrt{(x - x_*)^T A (x - x_*)} = ||By - d||_2$$

where

$$B = SK_n, \quad d = Sx_*$$

Since y_n solves the least squares problem $\min ||By - d||$, we have

$$By_n - d \perp By \quad \text{for any} \quad y \in \mathbb{R}^n$$

This is the same as

$$r_n \perp x \quad \text{for any} \quad x \in \mathcal{K}_n$$

5.1.5 The next conjugate direction

Suppose x_j is the conjugate gradient iteration that solves the subspace minimization problem $\min_{x \in \mathcal{K}_j} \phi(x)$, it is not difficult to realize that

$$\mathcal{K}_n =< x_1, x_2, ..., x_n >=< r_0, r_1, ..., r_{n-1} >$$

where $r_0 = b - Ax_0 = b$. We also assume that

$$\mathcal{K}_j =< p_0, p_1, ..., p_{j-1} > \quad \text{for} \quad j \leq n$$

The question now is how to choose p_n, such that

- $\mathcal{K}_{n+1} =< p_0, p_1, ..., p_n >$;
- $p_n^T A p_j = 0$ for $n = 0, 1, 2, ..., n-1$.

To satisfy the first condition, we realize that $r_n = b - Ax_n$ is in $\mathcal{K}_{n+1}$ (and not in $\mathcal{K}_n$), therefore, we can choose

$$p_n = r_n + \text{a component in} \quad \mathcal{K}_n$$

to satisfy the second condition. The component in $\mathcal{K}_n$ can be written as

$$\beta_n p_{n-1} + (*)p_{n-2} + ... + (*)p_0$$

since $\{p_0, p_1, ..., p_{n-1}\}$ is a basis of $\mathcal{K}_n$. We use the condition $p_j^T A p_n = p_n^T A p_j = 0$ (since $A = A^T$) to find the coefficients. For $j \leq n-2$, we have

$$0 = p_j^T A p_n = p_j^T A r_n + (*) p_j^T A p_j$$

Now, $p_j^T A r_n = r_n^T (A p_j) = 0$, since $p_j \in \mathcal{K}_{n-1}$ or $A p_j \in \mathcal{K}_n$ (and $r_n \perp \mathcal{K}_n$ as in the last section), therefore, $(*) = 0$. Meanwhile, we obtain

$$p_n = r_n + \beta_n p_{n-1} \quad \text{for} \quad \beta_n = \frac{r_n^T A p_{n-1}}{p_{n-1}^T A p_{n-1}}$$

5.1.6 The conjugate gradient iteration

We now have the following conjugate gradient method:

- Let $x_0 = 0$, $r_0 = b$, $p_0 = r_0$.
- For $n = 1, 2, 3, ...$

$$\begin{aligned} \alpha_n &= \frac{r_{n-1}^T r_{n-1}}{p_{n-1}^T A p_{n-1}} \\ x_n &= x_{n-1} + \alpha_n p_{n-1} \\ r_n &= r_{n-1} - \alpha_n A p_{n-1} \\ \beta_n &= \frac{r_n^T r_n}{r_{n-1}^T r_{n-1}} \\ p_n &= r_n + \beta_n p_{n-1} \end{aligned}$$

We notice that the formulas for α_n and β_n are different. But they are equivalent to the formulas in previous sections. One step of this algorithm requires

- Evaluate $v = Ap_{n-1}$;
- $2m$ operations for $p_{n-1}^T v = p_{n-1}^T A p_{n-1}$;
- $2m$ operations for $x_n = x_{n-1} + \alpha_n p_{n-1}$;
- $2m$ operations for $r_n = r_{n-1} - \alpha_n v = r_{n-1} - \alpha_n A p_{n-1}$;
- $2m$ operations for $r_n^T r_n$;
- $2m$ operations for $p_n = r_n + \beta_n p_{n-1}$

This is a total of $10m$ operations, plus one matrix vector multiplication.

5.1.7 Optimal polynomial problem

Let A be an $m \times m$ positive definite matrix, we can define a norm by:

$$||y||_A = \sqrt{y^* A y}$$

for any vector of length m. We already know that the CG solution x_n solves the following minimization problem:

$$\min_{x \in \mathcal{K}_n} \phi(x).$$

It turns out that $\phi(x)$ is closely related to $||x - x_*||_A^2$, where $x_* = A^{-1}b$ is the exact solution. Therefore, x_n also solves

$$\min_{x \in \mathcal{K}_n} ||x - x_*||_A.$$

In other words,

$$||x_n - x_*||_A \leq ||x - x_*||_A \quad \text{for any} \quad x \in \mathcal{K}_n. \tag{5.1}$$

We also introduce the error:

$$e_n = x_* - x_n.$$

In particular, $e_0 = x_*$ since $x_0 = 0$. If we write down x_n as

$$x_n = c_1 b + c_2 Ab + ... + c_n A^{n-1} b$$

then

$$e_n = [I - c_1 A - c_2 A^2 - ... - c_n A^n] A^{-1} b = p_n(A) e_0$$

where p_n is a polynomial of degree n and $p_n(0) = 1$. While x_n is related to the polynomial p_n, a vector in $\mathcal{K}_n$ can be related to a polynomial q. Then (5.1) can be written as

$$||p_n(A)e_0||_A \leq ||q(A)e_0||_A \tag{5.2}$$

for any polynomial q with degree $\leq n$ and $q(0) = 1$.

5.1.8 Rate of convergence

The inequality (5.2) gives us a way to establish an estimate for the rate of convergence. We can try to choose some q, such that the right hand side of (5.2) is small. It turns out that the Chebyshev polynomial is useful for this purpose. The left hand side of (5.2) is just $||e_n||_A$. After some calculation with the Chebyshev polynomials, we can establish the following:

$$\frac{||e_n||_A}{||e_0||_A} \leq 2 \left[\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right]^n \tag{5.3}$$

where $\kappa = \lambda_1 / \lambda_m$ is the condition number of A (λ_1 and λ_m are the largest and smallest eigenvalues of A).

5.2 Lanczos method

5.2.1 Lanczos tridiagonalization process

Let A be an $m \times m$ real symmetric matrix, we can find an orthgonal matrix Q, such that

$$AQ = QT$$

where T is real symmetric tridiagonal. The first method is based on Householder reflections (or Givens rotations). The Lanczos method is an alternative. As a matter of fact, the matrix Q is not unique. The first column can be any unit vector. But if the first column of Q is fixed, the other columns are determined up to a plus or minus sign (Q is real here) in the general case. Let us write the columns of Q:

$$Q = [q_1, q_2, ..., q_m]$$

and write down the entries of T

$$T = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_{m-1} \\ & & \beta_{m-1} & \alpha_m \end{bmatrix}$$

We can obtain an algorithm for computing Q and T by comparing columns of $AQ = QT$. For the first column, we have

$$Aq_1 = \alpha_1 q_1 + \beta_1 q_2.$$

As we have mentioned earlier, q_1 is an arbitrary (but unit) vector. We know that q_2 is orthgonal with q_1 (i.e., $q_1^* q_2 = 0$). Thus,

$$q_1^* A q_1 = \alpha_1 q_1^* q_1 + 0 = \alpha_1$$

Next, we define

$$w = Aq_1 - \alpha_1 q_1.$$

It is possible that w is a zero vector (when q_1 is an eigenvector of A and α_1 is the corresponding eigenvalue). But in general, $w \neq 0$. Then we define

$$\beta_1 = ||w||_2$$

and

$$q_2 = w/\beta_1.$$

Next, we compare the second columns of $AQ = QT$. That is

$$Aq_2 = \beta_1 q_1 + \alpha_2 q_2 + \beta_2 q_3$$

This gives rise to

$$\begin{aligned} \alpha_2 &= q_2^* A q_2 \\ w &= Aq_2 - \beta_1 q_1 - \alpha_2 q_2 \\ \beta_2 &= ||w||_2 \\ q_3 &= w/\beta_2 \quad \text{if} \quad \beta_2 \neq 0 \end{aligned}$$

If $\beta_2 = 0$, we can continue with an arbitrary q_3 which is a unit vector orthogonal to q_1 and q_2. The above is basically the general step, if you change the subscripts 1, 2 and 3 to $n-1$, n and $n+1$, respectively.

For symmetric matrices, the methods discussed in a previous chapter on matrix eigenvalue problems require such a tridiagonalization process. However, the Lanczos method is not competitive with the Householder method, because the columns of Q are not always orthgonal to each other due to round off errors. However, the Lanczos method becomes useful when A is large and sparse and when we truncate the Lanczos process and use approximation. For some n much less than m, we have

$$AQ_n \approx Q_n T_n$$

where Q_n is the matrix of first n columns of Q, T_n is the leading $n \times n$ block of T. If you think about the above equation, you can see that the first $n-1$ columns are exact, but the last column is

$$Aq_n \approx \beta_{n-1} q_{n-1} + \alpha_n q_n$$

The exact relationship is

$$Aq_n = \beta_{n-1}q_{n-1} + \alpha_n q_n + \beta_n q_{n+1}$$

When this truncation is used, we can use the eigenvalues of T_n to approximate a few largest and smallest eigenvalues of A, we can also use it to approximate

$$f(A)b$$

where $f(A)$ is some matrix function, b is a given vector. As a special case, when $f(A) = A^{-1}$ we are actually finding approximate solutions of $Ax = b$. It turns that this is related to the conjugate gradient method.

We first notice that

$$Q_n^* A Q_n = T_n.$$

For a non-zero vector b, we can also introduce the Krylov matrix

$$K_n = [b, Ab, A^2b, ..., A^{n-1}b].$$

Now, if we start the Lanczos process with $q_1 = b/||b||_2$, then

$$K_n = Q_n R_n$$

is a reduced QR factorization of K_n. If you use the relationship $AQ = QT$, and re-write the columns as

$$\begin{aligned} b &= ||b||_2 q_1 \\ Ab &= ||b||_2(Aq_1) = ||b||_2(\alpha_1 q_1 + \beta_1 q_2) = (*)q_1 + (*)q_2 \end{aligned}$$

and we can also see that

$$Ab^2 = (*)q_1 + (*)q_2 + (*)q_3$$

This gives rise to the reduced QR factorization.

5.2.2 Approximating eigenvalues

For a real symmetric matrix A (of size $m \times m$, m may be large), we can use the Lanczos method to find a few largest and smallest eigenvalues. We start with an initial vector b, let $q_1 = b/||b||_2$ and calculate the coefficients of tridiagonal matrix T. We may terminate at step n and calculate the eigenvalues of T_n, then the extreme eigenvalues of T_n are the approximate eigenvalues of A.

Let λ_1 is the smallest eigenvalue of A, then

$$\lambda_1 = \min_{x \neq 0} \frac{x^T A x}{x^T x} = \min_{||x||=1} x^T A x.$$

Associated with the vector b, we have the Krylov subspace:

$$\mathcal{K}_n =< b, Ab, .., A^{n-1}b > .$$

Let $\lambda_1^{(n)}$ be the smallest eigenvalue of T_n, then we can prove that

$$\lambda_1^{(n)} = \min_{0 \neq x \in \mathcal{K}_n} \frac{x^T A x}{x^T x}.$$

This means that $\lambda_1^{(n)}$ is the best result we can get by minimizing the Rayleigh quotient in the Krylov subspace. This is in analog to the conjugate gadient method, where the n-th iteration x_n solves the minimization problem of $\phi(x)$ in $\mathcal{K}_n$.

To prove the above result, we start with a vector x in $\mathcal{K}_n$:

$$x = y_1 q_1 + y_2 q_2 + ... + y_n q_n = Q_n y.$$

This is so, because the Lanczos vectors $\{q_1, q_2, ...q_n\}$ is a basis for $\mathcal{K}_n$. This is related to the earlier result that the Krylov matrix K_n has a reduced QR factorization $K_n = Q_n R_n$. For this vector x,

$$\frac{x^T A x}{x^T x} = \frac{y^T Q_n^T A Q_n y}{y^T Q_n^T Q_n y} = \frac{y^T T_n y}{y^T y}.$$

Thus,

$$\min_{0 \neq x \in \mathcal{K}_n} \frac{x^T A x}{x^T x} = \min_{y \neq 0} \frac{y^T T_n y}{y^T y} = \lambda_1^{(n)}.$$

Corresponding to the eigenvalue $\lambda_1^{(n)}$ of T_n, we have an unit eigenvector y, then the approximate eigenvector for A is $x = Q_n y$. Therefore, the Lanczos method gives the best approximation to the eigenvector, if the approximation is restricted in the Krylov subspace $\mathcal{K}_n$.

All this is true, if we change the smallest eigenvalue to the largest eigenvalue and change the minimum to maximum for the Rayleigh quotient $x^T A x/(x^T x)$. In reality, the Lanczos method is used to find the approximations of a few largest and smallest eigenvalues of A.

Similar to the case of conjugate gradient method, the Lanczos method also has related to an optimal polynomial problem. For the original matrix A, we have its characteristic polynomial $p(\lambda) = \det(\lambda I - A)$. The degree of $p(\lambda)$ is m and the leading term is λ^m. The coefficient of the leading term is 1. This is called a monic polynomial. In linear algebra, we know that $p(A) = 0$ which is the $m \times m$ zero matrix. If we have a vector b, of course, $p(A)b = 0$ (the 0 vector). Now,

for $n < m$, we can look for a monic polynomial of degree n, say $p^n(\lambda)$ (here, the superscript does not mean power), such that $||p^n(A)b||$ is minimized. The answer is

$$p^n(\lambda) = \det(\lambda I - T_n).$$

That is, $p^n(\lambda)$ is the characteristic polynomial of T_n, where $q_1 = b/||b||_2$ is used to start the Lanczos tridiagonalization process as before. More precise, we can prove that

$$||p^n(A)b||_2 \leq ||q^n(A)b||_2 \tag{5.4}$$

where q^n is an arbitrary monic polynomial of degress n. Monic means that the coefficient of λ^n in $q^n(\lambda)$ is 1. That is

$$q^n(\lambda) = \lambda^n + (*)\lambda^{n-1} + ... + (*)\lambda + (*).$$

Of course, p^n is also a monic polynomial of degree n.

As a special application of the above result, we consider A a $m \times m$ matrix with only three distinct eigenvalues (assuming $m > 3$), say

$$\lambda_1, \quad \lambda_2, \quad \lambda_3.$$

Now, if we start with a vector b which has non-zero components in eigenvectors of all three eigenvalues, then we claim that

$$p^3(\lambda) = (\lambda - \lambda_1)(\lambda - \lambda_2)(\lambda - \lambda_3).$$

This is so, because if we choose $q^3(\lambda) = (\lambda - \lambda_1)(\lambda - \lambda_2)(\lambda - \lambda_3)$, then we have

$$||q^3(A)b||_2 = 0$$

This leads to

$$||p^3(A)b||_2 = 0.$$

Therefore, $p^3 = (\lambda - \lambda_1)(\lambda - \lambda_2)(\lambda - \lambda_3)$, because other monic polynomials of degress 3 can not produce a zero. In other words, if a matrix A has only three distinct eigenvalues, we can always find the three eigenvalues using T_3.

To prove (5.4), we need to re-write

$$\min_{q^n \in P^n} ||q(A)b||_2$$

as a least squares problem involving Q_n (the reduced QR factorization of K_n is useful here), where P^n is the set of all monic polynomials of degree n, then show that p^n is related to the solution of that least squares problem.

5.2.3 Approximating $f(A)b$

To approximate $f(A)b$, we start a Lanczos process with $q_1 = b/||b||_2$, then truncated at some n steps. From the approximate relationship:

$$AQ_n \approx Q_n T_n$$

we generalize to

$$f(A)Q_n \approx Q_n f(T_n).$$

Thus,

$$f(A)q_1 \approx Q_n f(T_n)e_1$$

where $e_1 = (1, 0, ..., 0)^*$ is the first column of the $n \times n$ identity matrix. Therefore,

$$f(A)b \approx ||b||_2 Q_n f(T_n)e_1$$

This has a lot of applications. Consider the heat equation

$$u_t = \Delta u$$

If we discretize the spatial variables in a suitable way, we have a vector U approximating the values of u at various spatial grid points, and the partial differential equation is changed to

$$\frac{dU}{dt} = -AU$$

where A is symmetric, sparse (if finite difference or finite element methods are sued) and positive definite. The exact solution is

$$U(t) = e^{-At}U(0)$$

assuming that initial condition is given as $t = 0$. Similarly, we consider the Schrödinger equation (fundamental in quantum mechanics)

$$iu_t = (\Delta - q)u$$

If we discretize the spatial variables first, we can replace the operator $-\Delta + q$ by a symmetric matrix A, then the equation is changed to

$$\frac{dU}{dt} = iAU$$

The exact solution is

$$U(t) = e^{iAt}U(0).$$

This might be the first case where the Lanczos method is shown to be useful.

Another example is the wave equation:

$$u_{tt} = \Delta u$$

If we discretize the spatial variables, we end up with

$$\frac{d^2U}{dt^2} = -AU$$

where A replaces the operator $-\Delta$. It turns out that we have a two step method:

$$U(t_2) + U(t_0) = 2\cos(h\sqrt{A})U(t_1)$$

where $t_j = t_0 + jh$ and h is the step size. Assume $U(t_0)$ and $U(t_1)$ are known, we can then calculate $U(t_2)$ based on a Lanczos approximation for $\cos(h\sqrt{A})U(t_1)$.

5.3 Exercises

1. Using the standard notations for the Conjugate Gradient method, where x_n is the n-th iteration of the approximate solution (for $Ax = b$, assuming $x_0 = 0$), r_n is the residual, p_n is the n-th A-conjugate direction, show that

   (a)

   $$\alpha_n = \frac{p_{n-1}^T b}{p_{n-1}^T A p_{n-1}} = \frac{r_{n-1}^T p_{n-1}}{p_{n-1}^T A p_{n-1}} = \frac{r_{n-1}^T r_{n-1}}{p_{n-1}^T A p_{n-1}}$$

   (b)

   $$\beta_n = -\frac{r_n^T A p_{n-1}}{p_{n-1}^T A p_{n-1}} = \frac{r_n^T r_n}{r_{n-1}^T r_{n-1}}.$$

2. If A is not symmetric, then the Lanczos method should be replaced by the Arnoldi method. It is a method for reduction of A to upper Hessenberg matrix H, i.e., $AQ = QH$, where Q is unitary and H is upper Hessenberg. Starting from an arbitrary unit vector as the first column of Q, write down an algorithm for calculating other columns of Q and the entries of H. Demonstrate this for matrix

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

   If Q_n is the matrix of first n columns of Q, H_n is the $n \times n$ leading block of H, show that

   $$H_n = Q_n^* A Q_n$$

   Can you generalize it to

   $$p(H_n) = Q_n^* p(A) Q_n$$

   where $p(x)$ is a polynomial of x?

3. Let T be an $m \times m$ symmetric tridiagonal matrix ($m > 3$), T_3 is the 3×3 leading block of T. Let $p_3(\lambda)$ be a polynomial of degree 3. Compare the matrix $p_3(T_3)$ with the 3×3 leading block of $p_3(T)$. Which entries are the same?

4. We have derived the conjugate gradient method as an iterative minimization of $\phi(x) = \frac{1}{2}x^T Ax - x^T b$. Another way to minimize the same function — far slower, in general — is by the method of steepest descent.

   (a) Find a formula for $\nabla\phi(x)$, the steepest descent method uses $p_n = -\nabla\phi(x_n)$ as the direction for 1-D minimization.

   (b) Determine the formula α_n for the 1-D minmization

   $$\min_{\alpha} \phi(x_{n-1} + \alpha p_{n-1})$$

   (c) Write down the full steepest descent algorithm.