Lecture 33. The Arnoldi Iteration

Despite the many names and acronyms that have proliferated in the field of Krylov subspace matrix iterations, these algorithms are built upon a common foundation of a few fundamental ideas. One can take various approaches to describing this foundation. Ours will be to consider the Arnoldi process, a Gram–Schmidt-style iteration for transforming a matrix to Hessenberg form.

The Arnoldi/Gram–Schmidt Analogy

Suppose, to pass the time while marooned on a desert island, you challenged yourself to devise an algorithm to reduce a nonhermitian matrix to Hessenberg form by orthogonal similarity transformations, proceeding column by column from a prescribed first column q_1. To your surprise, you would probably find you could solve this problem in an hour and still have time to gather coconuts for dinner. The method you would come up with goes by the name of the Arnoldi iteration. If A is hermitian, the Hessenberg matrix becomes tridiagonal, an n-term recurrence relation becomes a three-term recurrence relation, and the name changes to the Lanczos iteration, to be discussed in Lecture 36.

Here is an analogy. For computing the QR factorization $A = QR$ of a matrix A, we have discussed two methods in this book: Householder reflections, which triangularize A by a succession of orthogonal operations, and Gram–Schmidt orthogonalization, which orthogonalizes A by a succession of triangular operations. Though Householder reflections lead to a more nearly

orthogonal matrix Q in the presence of rounding errors, the Gram–Schmidt process has the advantage that it can be stopped part-way, leaving one with a reduced QR factorization of the first n columns of A. The problem of computing a Hessenberg reduction $A = QHQ^*$ of a matrix A is exactly analogous. There are two standard methods: Householder reflections (applied now on two sides of A rather than one) and the Arnoldi iteration. Thus Arnoldi is the analogue of Gram–Schmidt for similarity transformations to Hessenberg form rather than QR factorization. Like Gram–Schmidt, it has the advantage that it can be stopped part-way, leaving one with a partial reduction to Hessenberg form that is exploited in various manners to form iterative algorithms for eigenvalues or systems of equations.

Thus, this lecture is to Lecture 26 as Lecture 8 is to Lecture 10.

We can summarize the four algorithms just mentioned in a table:

	$A = QR$	$A = QHQ^*$
orthogonal structuring	Householder	Householder
structured orthogonalization	Gram–Schmidt	Arnoldi

For the remainder of this book, m and $n < m$ are positive integers, A is a real or complex $m \times m$ matrix, and $\| \cdot \| = \| \cdot \|_2$. In addition, one further character will now appear in the drama, an m-vector that we shall denote by b. The Arnoldi process needs this vector in order to get started. For applications to eigenvalue problems, we typically assume that b is random. For applications to systems of equations, as considered in later lectures, it will be the right-hand side, or more generally, the initial residual (see Exercise 35.5).

Mechanics of the Arnoldi Iteration

A complete reduction of A to Hessenberg form by an orthogonal similarity transformation might be written $A = QHQ^*$, or $AQ = QH$. However, in dealing with iterative methods we take the view that m is huge or infinite, so that computing the full reduction is out of the question. Instead we consider the first n columns of $AQ = QH$. Let Q_n be the $m \times n$ matrix whose columns are the first n columns of Q:

$$Q_n = \left[\begin{array}{c|c|c|c} q_1 & q_2 & \cdots & q_n \end{array}\right]. \tag{33.1}$$

Here and in the lectures ahead, it would be consistent with our usage elsewhere in the book to put hats on the symbols Q_n, since these matrices are rectangular, but to keep the formulas uncluttered we do not do this.

Let $\tilde{H}_n$ be the $(n+1)\times n$ upper-left section of H, which is also a Hessenberg matrix:

$$\tilde{H}_n = \begin{bmatrix} h_{11} & \cdots & h_{1n} \\ h_{21} & h_{22} & \\ & \ddots & \ddots & \vdots \\ & & h_{n,n-1} & h_{nn} \\ & & & h_{n+1,n} \end{bmatrix}. \tag{33.2}$$

Then we have

$$AQ_n = Q_{n+1}\tilde{H}_n, \tag{33.3}$$

that is,

$$\begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix} \left[\begin{array}{c|c|c} & & \\ q_1 & \cdots & q_n \\ & & \end{array}\right] = \left[\begin{array}{c|c|c} & & \\ q_1 & \cdots & q_{n+1} \\ & & \end{array}\right] \begin{bmatrix} h_{11} & \cdots & h_{1n} \\ h_{21} & & \vdots \\ & \ddots & \\ & & h_{n+1,n} \end{bmatrix}.$$

The nth column of this equation can be written as follows:

$$Aq_n = h_{1n}q_1 + \cdots + h_{nn}q_n + h_{n+1,n}q_{n+1}. \tag{33.4}$$

In words, q_{n+1} satisfies an $(n+1)$-term recurrence relation involving itself and the previous Krylov vectors.

The Arnoldi iteration is simply the modified Gram–Schmidt iteration that implements (33.4). The following algorithm should be compared with Algorithm 8.1.

Algorithm 33.1. Arnoldi Iteration

b = arbitrary, $q_1 = b/\|b\|$
for $n = 1, 2, 3, \ldots$
    $v = Aq_n$
    for $j = 1$ **to** n
        $h_{jn} = q_j^* v$
        $v = v - h_{jn}q_j$
    $h_{n+1,n} = \|v\|$ [see Exercise 33.2 concerning $h_{n+1,n} = 0$]
    $q_{n+1} = v/h_{n+1,n}$

The reader can see at a glance how simple the Arnoldi process is. In a high-level language such as MATLAB, it can be implemented in less than a dozen lines. The matrix A appears only in the product Aq_n, which can be computed by a black box procedure as described in the last lecture.

QR Factorization of a Krylov Matrix

The power of the Arnoldi process lies in the various interpretations that can be made of it, and in the algorithms these suggest. For a first interpretation, consider the recurrence (33.4). It is evident from this formula that the vectors $\{q_j\}$ form bases of the successive *Krylov subspaces* generated by A and b, defined as follows:

$$\mathcal{K}_n = \langle b, Ab, \ldots, A^{n-1}b \rangle = \langle q_1, q_2, \ldots, q_n \rangle \subseteq \mathbb{C}^m . \tag{33.5}$$

Moreover, since the vectors q_j are orthonormal, these are orthonormal bases. Thus the Arnoldi process can be described as the systematic construction of orthonormal bases for successive Krylov subspaces.

To express this observation in matrix form, let us define K_n to be the $m \times n$ *Krylov matrix*

$$K_n = \left[\begin{array}{c|c|c|c} b & Ab & \cdots & A^{n-1}b \end{array} \right] . \tag{33.6}$$

Then K_n must have a reduced QR factorization

$$K_n = Q_n R_n , \tag{33.7}$$

where Q_n is the same matrix as above. In the Arnoldi process, neither K_n nor R_n is formed explicitly. Doing so would make for an unstable algorithm, since these are exceedingly ill-conditioned matrices in general, as the columns of K_n all tend to approximate the same dominant eigenvector of A. However, (33.6) and (33.7) give an intuitive explanation of why the Arnoldi process leads to effective methods for determining certain eigenvalues. Clearly K_n might be expected to contain good information about the eigenvalues of A with largest modulus, and the QR factorization might be expected to reveal this information by peeling off one approximate eigenvector after another, starting with the dominant one.

The explanation just given may remind the reader of a similar discussion that appeared earlier in this book. The relationship between (33.6)–(33.7) and the Arnoldi algorithm is analogous to that between simultaneous iteration and the QR algorithm for computing eigenvalues of matrices. One is easy to understand but unstable, the other is subtler but stabler. The difference is

that, whereas the Arnoldi iteration is based upon the QR factorization (33.7) of the matrix whose columns are $b, Ab, \ldots, A^{n-1}b$, simultaneous iteration and the QR algorithm are based upon the QR factorization (28.16) of the matrix whose columns are $A^n e_1, \ldots, A^n e_m$. We can summarize this parallel in another table:

	quasi-direct	iterative
straightforward but unstable	simultaneous iteration	(33.6)–(33.7)
subtle but stable	QR algorithm	Arnoldi

Projection onto Krylov Subspaces

Another way to view the Arnoldi process is as a computation of projections onto successive Krylov subspaces. To see this, note that the product $Q_n^* Q_{n+1}$ is the $n \times (n+1)$ identity, i.e., the $n \times (n+1)$ matrix with 1 on the main diagonal and 0 elsewhere. Therefore $Q_n^* Q_{n+1} \tilde{H}_n$ is the $n \times n$ Hessenberg matrix obtained by removing the last row of $\tilde{H}_n$:

$$H_n = \begin{bmatrix} h_{11} & \cdots & h_{1n} \\ h_{21} & h_{22} & \\ & \ddots & \ddots & \vdots \\ & & h_{n,n-1} & h_{nn} \end{bmatrix}. \tag{33.8}$$

From (33.3) we accordingly have

$$H_n = Q_n^* A Q_n. \tag{33.9}$$

This matrix can be interpreted as the representation in the basis $\{q_1, \ldots, q_n\}$ of the orthogonal projection of A onto $\mathcal{K}_n$. Is it clear what this interpretation means? Here is a precise statement. Consider the linear operator $\mathcal{K}_n \to \mathcal{K}_n$ defined as follows: given $v \in \mathcal{K}_n$, apply A to it, then orthogonally project Av back into the space $\mathcal{K}_n$. Since the orthogonal projector of $\mathbb{C}^m$ onto $\mathcal{K}_n$ is $Q_n Q_n^*$, this operator can be written $Q_n Q_n^* A$ with respect to the standard basis of $\mathbb{C}^m$. With respect to the basis of columns of Q_n, it can therefore be written $Q_n^* A Q_n$.

The kind of projection just described comes up throughout applied and numerical mathematics. In another context it is known as the *Rayleigh–Ritz* procedure; not coincidentally, in the diagonal elements of H_n one recognizes the Rayleigh quotients of A with respect to the vectors q_j. This projection process is also one of the ideas underlying finite element methods for solution

of partial differential equations, as well as their younger relatives known as spectral methods.

Since H_n is a projection of A, one might imagine that its eigenvalues would be related to those of A in a useful fashion. These n numbers,

$$\{\theta_j\} = \{\text{eigenvalues of } H_n\}, \tag{33.10}$$

are called the *Arnoldi eigenvalue estimates* (at step n) or *Ritz values* (with respect to $\mathcal{K}_n$) of A. In the next lecture we shall see that some of these numbers may be extraordinarily accurate approximations to some of the eigenvalues of A, even for $n \ll m$.

We summarize the developments of this lecture in a theorem, to be compared with Theorem 28.3.

Theorem 33.1. *The matrices Q_n generated by the Arnoldi iteration are reduced QR factors of the Krylov matrix* (33.6)*:*

$$K_n = Q_n R_n. \tag{33.11}$$

The Hessenberg matrices H_n are the corresponding projections

$$H_n = Q_n^* A Q_n, \tag{33.12}$$

and the successive iterates are related by the formula

$$AQ_n = Q_{n+1} \tilde{H}_n. \tag{33.13}$$

Exercises

33.1. Let $A \in \mathbb{C}^{m\times m}$ and $b \in \mathbb{C}^m$ be arbitrary. Show that any $x \in K_n$ is equal to $p(A)b$ for some polynomial p of degree $\le n-1$.

33.2. Suppose Algorithm 33.1 is executed for a particular A and b until at some step n, an entry $h_{n+1,n} = 0$ is encountered.

(a) Show how (33.13) can be simplified in this case. What does this imply about the structure of a full $m \times m$ Hessenberg reduction $A = QHQ^*$ of A?

(b) Show that $\mathcal{K}_n$ is an invariant subspace of A, i.e., $A\mathcal{K}_n \subseteq \mathcal{K}_n$.

(c) Show that if the Krylov subspaces of A generated by b are defined by $\mathcal{K}_k = \langle b, Ab, \ldots, A^{k-1}b\rangle$ as in (33.5), then $\mathcal{K}_n = \mathcal{K}_{n+1} = \mathcal{K}_{n+2} = \cdots$.

(d) Show that each eigenvalue of H_n is an eigenvalue of A.

(e) Show that if A is nonsingular, then the solution x to the system of equations $Ax = b$ lies in $\mathcal{K}_n$.

The appearance of an entry $h_{n+1,n} = 0$ is called a *breakdown* of the Arnoldi iteration, but it is a breakdown of a benign sort. For applications in computing eigenvalues (Lecture 34) or solving systems of equations (Lecture 35), because of (d) and (e), a breakdown usually means that convergence has occurred and the iteration can be terminated. Alternatively, a new orthonormal vector q_{n+1} could be selected at random and the iteration then continued.

33.3. (a) Suppose Algorithm 33.1 is executed for a particular A and b and runs to completion ($n = m$), with no breakdown of the kind described in the last exercise. Show that this implies that the minimal polynomial of A is of degree m.

(b) Conversely, suppose that the minimal polynomial of A is of degree m. Show that this does not imply that for a particular choice of b, Algorithm 33.1 will necessarily run to completion.

(c) Explain why the result of (a) does not contradict Exercise 25.1(b).