Dr H. Fawzi

**Mathematical Tripos Part II: Michaelmas Term 2022**

# Numerical Analysis – Lecture 24

**Reduction to tridiagonal form** Given a symmetric matrix $A \in \mathbb{R}^{n\times n}$, we can find an orthogonal matrix $P$ such that $P^TAP = T$ is tridiagonal, using a sequence of Householder reflections.

**Revision 5.15** *Recall that a Householder reflection matrix is an orthogonal matrix of the form $H = I - 2\frac{\boldsymbol{u}\boldsymbol{u}^T}{\|\boldsymbol{u}\|_2^2}$, which represents a reflection about the hyperplane normal to $\boldsymbol{u}$. For any two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ such that $\|\boldsymbol{x}\|_2 = \|\boldsymbol{y}\|_2$, the Householder matrix corresponding to $\boldsymbol{u} = \boldsymbol{x} - \boldsymbol{y}$ satisfies $H\boldsymbol{x} = \boldsymbol{y}$.*

Let $H_1$ be an orthogonal matrix of the form

$$H_1 = \begin{bmatrix} 1 & 0 \\ 0 & \hat{H}_1 \end{bmatrix}$$

where $\hat{H}_1 \in \mathbb{R}^{(n-1)\times(n-1)}$ is a Householder matrix such that $\hat{H}_1 A[2:n,1]$ is proportional to the vector $(1,0,\dots,0) \in \mathbb{R}^{n-1}$. Then we see that

$$H_1AH_1^T = \begin{bmatrix} A_{11} & (\hat{H}_1A[2:n,1])^T \\ \hat{H}_1A[2:n,1] & \hat{H}_1A[2:n,2:n]\hat{H}_1^T \end{bmatrix} = \begin{bmatrix} * & \bullet & 0 & \dots & 0 \\ \bullet & \bullet & \bullet & \dots & \bullet \\ 0 & \bullet & \bullet & \dots & \bullet \\ \vdots & \vdots & & & \vdots \\ 0 & \bullet & \bullet & \dots & \bullet \end{bmatrix},$$

where $'*'$ (resp. $'\bullet'$) indicates an entry that is unchanged (resp. changed) by the orthogonal conjugation. Assuming we let $A = H_1AH_1^T$, let $H_2$ be an orthogonal matrix of the form

$$H_2 = \begin{bmatrix} I_2 & 0 \\ 0 & \hat{H}_2 \end{bmatrix}$$

where $\hat{H}_2 \in \mathbb{R}^{(n-2)\times(n-2)}$ is such that $\hat{H}_2A[3:n,2]$ is proportional to $(1,0,\dots,0) \in \mathbb{R}^{n-2}$. Then we see that

$$H_2AH_2^T = \begin{bmatrix} * & * & 0 & 0 & \dots & 0 \\ * & * & \bullet & 0 & \dots & 0 \\ 0 & \bullet & \bullet & \bullet & \dots & \bullet \\ 0 & 0 & \bullet & & & \bullet \\ \vdots & \vdots & \vdots & & & \vdots \\ 0 & 0 & \bullet & & \dots & \bullet \end{bmatrix}$$

Importantly, note that the zero entries in the first row and column are left unchanged. It is not hard to see that by continuing in the same way, we arrive at a sequence of Householder reflections that will bring $A$ into tridiagonal form. The complete algorithm can be described as follows.

REDUCTION TO SYMMETRIC TRIDIAGONAL FORM
Input: $A \in \mathbb{R}^{n\times n}$ symmetric
For $i = 1$ to $n-2$

- Let $\boldsymbol{x} = A[i+1:n, i] \in \mathbb{R}^{n-i}$
- Let $\boldsymbol{u} = \boldsymbol{x} \pm \|\boldsymbol{x}\|_2\boldsymbol{e}_1 \in \mathbb{R}^{n-i}$
- $A$[i+1:n , 1:n] $= A$[i+1:n , 1:n] $- 2\boldsymbol{u}(\boldsymbol{u}^TA$[i+1:n , 1:n]$)/\|\boldsymbol{u}\|_2^2$
- $A$[1:n , i+1:n] $= A$[1:n , i+1:n] $- 2(A$[1:n , i+1:n]$\boldsymbol{u})\boldsymbol{u}^T/\|\boldsymbol{u}\|_2^2$

The last two lines of the algorithm implement the conjugation by $H_i = \begin{bmatrix} I_i & 0 \\ 0 & \hat{H}_i \end{bmatrix}$: the first line corresponds to multiplying $A$ on the left by $H_i$, and the second line corresponds to multiplying $A$ on the right by $H_i^T$. Upon termination of the algorithm, the matrix $A$ has been reduced to symmetric tridiagonal form. The main computational cost of the algorithm is the computation of $\boldsymbol{u}^T A[i+1:n, 1:n]$ and $A[1:n, i+1:n]\boldsymbol{u}$ (when $A$ is symmetric, which we assume here, these are the same vector, up to a transpose). Computing these vectors requires $\mathcal{O}(n(n-i))$ operations. Thus we see that the total cost of the algorithm is $\mathcal{O}(n^3)$.

**Remark 5.16** *Observe that there is a choice of sign to be made at each iteration, as one can reflect $\boldsymbol{x}$ into either $\|\boldsymbol{x}\|_2 \boldsymbol{e}_1$ or $-\|\boldsymbol{x}\|_2 \boldsymbol{e}_1$. In practice one chooses $\boldsymbol{u} = \boldsymbol{x} + \operatorname{sgn}(\boldsymbol{x}_1)\|\boldsymbol{x}\|_2 \boldsymbol{e}_1$ to avoid subtracting numbers that can be too close to each other.*

Summary: to compute the eigenvalues of a symmetric matrix $A$, we start by putting $A$ into tridiagonal form using the algorithm above (phase 1), then we apply the QR iterations to the tridiagonal matrix (phase 2). The computational cost of phase 1 is $\mathcal{O}(n^3)$, while the computational cost of phase 2 is generally in $\mathcal{O}(n^2)$, as each QR factorization requires $\mathcal{O}(n)$, and we typically need a constant number of QR iterations per eigenvalue.

## 5.2 Nonsymmetric eigenvalue decomposition

In this section we assume $A \in \mathbb{R}^{n\times n}$ is a general nonsymmetric matrix and we discuss the problem of computing its eigenvalues (which can be complex). For nonsymmetric matrices, there are multiple "eigenvalue-revealing" factorizations:

- Eigenvalue decomposition: $A = VDV^{-1}$ where $V \in \mathbb{C}^{n\times n}$ invertible, and $D$ diagonal
- Schur decomposition: $A = PTP^*$ where $P \in \mathbb{C}^{n\times n}$ unitary, and $T$ is upper triangular. The eigenvalues of $A$ appear on the diagonal of $T$

Since $A$ is real, it is desirable to have a decomposition that involves only real numbers. By pairing together complex conjugate pairs of eigenvalues in a Schur decomposition, one obtains the

- Real Schur decomposition: $A = QTQ^T$ where $Q \in \mathbb{R}^{n\times n}$ is orthogonal and $T = \begin{bmatrix} T_{11} & & * \\ 0 & \ddots & \\ 0 & 0 & T_{nn} \end{bmatrix} \in \mathbb{R}^{n\times n}$ is *quasi upper-triangular*, i.e., each $T_{ii}$ is either $1\times 1$, or $2\times 2$ with complex conjugate eigenvalues.

Eigenvalue algorithms for nonsymmetric matrices proceed in two phases, just like in the symmetric case.

**Phase 1:** We construct an orthogonal matrix $U \in \mathbb{R}^{n\times n}$ such that $U^T AU$ is in *upper Hessenberg form*. A matrix $H$ is said to be in upper Hessenberg form if $H_{ij} = 0$ for all $i > j+1$, i.e.,

$$H = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}.$$

Such a reduction to Hessenberg form can be obtained using exactly the same Householder-based algorithm described earlier; when applied to nonsymmetric matrices it yields a Hessenberg matrix. The computational cost of this phase is $\mathcal{O}(n^3)$.

**Phase 2:** Assuming $A$ is in upper Hessenberg form, we apply QR iterations. The basic QR iteration is the same as for symmetric matrices:

BASIC QR ITERATION
Input: $A \in \mathbb{R}^{n\times n}$. Let $A^{(0)} = A$. For $k = 0, 1, 2, \ldots$

- $QR = \operatorname{qr}(A^{(k)})$
- $A^{(k+1)} = Q^T A^{(k)} Q = RQ$

If the eigenvalues of $A$ all have distinct magnitudes then one can show that $A^{(k)}$ will converge to an upper-triangular matrix. However this method is not practical because of its slow convergence, and because it requires the eigenvalues to have distinct magnitudes.

Just like in the symmetric case, we use shifting and deflation to make the algorithm practical. The choice of shift needs to take into account the fact that eigenvalues are not necessarily real. Thus, instead of using $A^{(k)}_{nn}$ as the shift, we use the *Francis double shift*: Let $\begin{bmatrix} A^{(k)}_{n-1,n-1} & A^{(k)}_{n-1,n} \\ A^{(k)}_{n,n-1} & A^{(k)}_{n,n} \end{bmatrix}$ be the $2 \times 2$ bottom-right block of $A^{(k)}$, and let $a, b$ be its two eigenvalues. Then we apply two successive QR steps using $a$ and $b$ as shifts:

- $Q_1 R_1 = A^{(k)} - aI, \quad A^{(k+1/2)} = Q_1^T A^{(k)} Q_1$
- $Q_2 R_2 = A^{(k+1/2)} - bI, \quad A^{(k+1)} = Q_2^T A^{(k+1/2)} Q_2$

It turns out that one can perform these two successive steps in a single step $A^{(k+1)} = Q^T A^{(k)} Q$, without ever needing to manipulate complex numbers. The resulting algorithm produces a real Schur factorization of $A$.

Computational cost: the QR factorization of a matrix in upper Hessenberg form can be computing using Givens rotations in $\mathcal{O}(n^2)$ operations. If we assume the number of QR iterations needed is proportional to $n$ (the number of eigenvalues to compute), this means that the total cost of phase 2 is $\mathcal{O}(n^3)$. Thus the total cost of the algorithm (phase 1 + phase 2) is in $\mathcal{O}(n^3)$.