TANA15/Lecture 5 - Contents

Computing Eigenvalues

- Decoupling, Similarity Transformations, The Schur Decomposition.
- Hessenberg Decomposition. The QR Algorithm. Shifts.

Applications

- Google PageRank.

Similarity Transformations

Definition If $A = XBX^{-1}$ then we say that A and B are *similar* and X is called a *similarity transformation*.

Lemma If A and B are *similar* then $\lambda(A) = \lambda(B)$.

Remark A *similarity transformation* preserves eigenvalues. Specific matrices to use includes Gauss transformations, Householder recleftions and Givens rotations.

Sensitivity

Let $A \in \mathbb{C}^{n\times n}$ be non–defective and let $(\hat{x}, \hat{\lambda})$ be an *approximate* eigenpair of A, with $\|\hat{x}\|_2 = 1$, and put $r = A\hat{x} - \hat{\lambda}\hat{x}$.

Proposition There is an eigenvalue λ of A such that

$$|\lambda - \hat{\lambda}| \leq \kappa_2(X)\|r\|_2.$$

Corollary If A is Symmetric or Hermitean then

$$|\lambda - \hat{\lambda}| \leq \|r\|_2.$$

Remark This is often called the *Bauer-Fike* Theorem.

Example Let $(\hat{\lambda}, \hat{x}) = (1, (0, 0, 1)^T)$ and consider the matrix

$$A = \begin{pmatrix} 3 & 2 & 0 \\ 2 & 4 & \varepsilon \\ 0 & \varepsilon & 1 \end{pmatrix},$$

The residual is

$$r = A\hat{x} - \hat{\lambda}\hat{x} = \begin{pmatrix} 0 \\ \varepsilon \\ 1 \end{pmatrix} - 1 \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ \varepsilon \\ 0 \end{pmatrix}.$$

Since the matrix is symmetric $\kappa_2(X) = 1$ and

$$|\lambda_3 - 1| \leq \kappa_2(X)\|r\|_2 = |\varepsilon|.$$

Remark A small change to a_{ij} leads to a small change in the eigenvalues λ_k.

The Decoupling theorem

Theorem Suppose A has a block-structure

$$A = \begin{pmatrix} A_1 & B \\ 0 & A_2 \end{pmatrix},$$

then $\lambda(A) = \lambda(A_1) \cup \lambda(A_2)$.

Corollary If T is an *upper triangular* matrix then its eigenvalues are the diagonal elements, i.e. $\lambda_i = T_{ii}$.

Remark If $\hat{\lambda}_i$ is an eigenvalue then we get the corresponding eigenvector $\hat{x}_i$ efficiently by *inverse iteration*.

Algorithm Let $A^{(0)} = A$. Generate a sequence of similar matrices,

$$A^{(k+1)} = X_k A^{(k)} X_k^{-1}, \qquad k = 1, 2, \ldots$$

such that

$$\lim_{k\to\infty} A^{(k)} = T, \qquad \text{or} \qquad \lim_{k\to\infty} A^{(k)} = D,$$

where T is *upper triangular* and D is *diagonal*.

Question What types of similarity transformations are needed? Not every matrix can be written $A = XDX^T$, with X orthogonal.

The Schur decomposition

Theorem Every matrix $A \in \mathbb{R}^{n\times n}$ has a *Schur decomposition*, i.e.

$$A = QTQ^H,$$

where T is upper triangular and Q is unitary.

Corollary If A is *Hermitean* then T is *diagonal* and Q the eigenvector matrix.

Remarks Neither T or Q are unique. The eigenvalues of a matrix A can be computed by using only *reflections* or *rotations*.

The QR algorithm

Algorithm Put $A_0 = A$ and do

$$A_k = Q_k R_k, \text{ and } A_{k+1} = R_k Q_k, \text{ for } k = 1, 2, \ldots$$

In each step compute the QR decomposition of A_k and multiply the factors in reverse order. Need $\mathcal{O}(n^3)$ operations/step.

Proposition The sequence of matrices $\{A_k\}$ are *similar*.

Remark *If* the algorithm converges to an upper triangular matrix then we have the eigenvalues of A.

Proposition It holds that

$$A_{k+1} = S_k^H A S_k, \quad S_k = Q_0 Q_1 \cdots Q_k.$$

Also S_{k-1} provides an orthonormal basis for Range(A^k).

Theorem Suppose $A = A^T$ and $|\lambda_1| > \ldots > |\lambda_n|$. Then

$$A_k \to D = \mathrm{diag}(\lambda_i) \text{ as } k \to \infty.$$

Remark The proof is very similar to the convergence proof for the power method. In the non symmetric case $A_k \to T$, where T is upper triangular.

Example Perform $k = 100$ QR steps. In Matlab

```
>> A=[ 3 4 1 ; 4 5 -1 ; 1 -1 6];
>> Ak=A;
>> for k=1:100, [Q,R]=qr(Ak); Ak=R*Q;,end;

  Ak =
      8.1388    0.0000   -0.0000
      0.0000    6.2909    0.0000
      0.0000   -0.0000   -0.4297
```

The computed eigenvalues have 15 correct digits. Note that the eigenvectors are not saved during the QR process.

Observation Computing the QR decomposition of a full matrix A_k is very expensive. For a practically viable algorithm we need to reduce the computational work.

Question How to find a similarity transformation X so that it is easy to compute the QR decomposition of $B = XAX^{-1}$?

The Hessenberg Decomposition

Definition A matrix H is *Hessenberg* if $H_{ij} = 0$ for $i > j+1$.

Proposition Every matrix $A \in \mathbb{R}^{n\times n}$ can be written as $A = QHQ^H$, where H is Hessenberg and Q is orthogonal.

Remarks If A is Hermitean or Symmetric then the corresponding Hessenberg matrix is tridiagonal.

In Matlab `H=hess(A);`

Example Suppose A is a 5×5 matrix. First select a Householder reflection such that $H_1 A(2:5,1) = \alpha e_1$. Then,

$$\tilde{H}_1 A \tilde{H}_1^T = \begin{pmatrix} x & x & x & x & x \\ + & + & + & + & + \\ 0 & + & + & + & + \\ 0 & + & + & + & + \\ 0 & + & + & + & + \end{pmatrix} \tilde{H}_1^T = \begin{pmatrix} x & + & + & + & + \\ x & + & + & + & + \\ 0 & + & + & + & + \\ 0 & + & + & + & + \\ 0 & + & + & + & + \end{pmatrix} = A_2.$$

Next select a reflection such that $H_2 A_2(3:5,2) = \alpha e_1$. Then

$$\tilde{H}_2 A_2 \tilde{H}_2^T = \begin{pmatrix} x & x & x & x & x \\ x & x & x & x & x \\ 0 & + & + & + & + \\ 0 & 0 & + & + & + \\ 0 & 0 & + & + & + \end{pmatrix} \tilde{H}_2^T = \begin{pmatrix} x & x & + & + & + \\ x & x & + & + & + \\ 0 & x & + & + & + \\ 0 & 0 & + & + & + \\ 0 & 0 & + & + & + \end{pmatrix} = A_3.$$

For the final step select a Householder reflection such that $H_3 A_3(4:5,3) = \alpha e_1$. Then,

$$\tilde{H}_3 A_3 \tilde{H}_3^T = \begin{pmatrix} x & x & x & x & x \\ x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & + & + & + \\ 0 & 0 & 0 & + & + \end{pmatrix} \tilde{H}_3^T = \begin{pmatrix} x & x & x & + & + \\ x & x & x & + & + \\ 0 & x & x & + & + \\ 0 & 0 & x & + & + \\ 0 & 0 & 0 & + & + \end{pmatrix} = A_4.$$

Remarks Need $n-2$ reflections. Don't need $Q = \tilde{H}_3 \tilde{H}_2 \tilde{H}_1$.

If A is Symmetric/Hermitean then the Hessenberg form is tridiagonal.

Hessenberg/*QR* step

The decomposition $A_k = Q_k R_k$ is computed using $n-1$ Givens Rotations.

$$G_{34} G_{23} G_{12} \begin{pmatrix} x & x & x & x \\ x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \end{pmatrix} = G_{34} G_{23} \begin{pmatrix} + & + & + & + \\ 0 & + & + & + \\ 0 & x & x & x \\ 0 & 0 & x & x \end{pmatrix} =$$

$$G_{34} \begin{pmatrix} x & x & x & x \\ 0 & + & + & + \\ 0 & 0 & + & + \\ 0 & 0 & x & x \end{pmatrix} = \begin{pmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & + & + \\ 0 & 0 & 0 & + \end{pmatrix} = R_k.$$

We have computed $A_k = Q_k R_k$ with $Q_k^T = G_{34} G_{23} G_{12}$.

Multiply $A_{k+1} = R_k Q_k = R_k G_{12}^T G_{23}^T G_{34}^T$. We obtain

$$\begin{pmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \\ 0 & 0 & 0 & x \end{pmatrix} G_{12}^T G_{23}^T G_{34}^T = \begin{pmatrix} + & + & x & x \\ + & + & x & x \\ 0 & 0 & x & x \\ 0 & 0 & 0 & x \end{pmatrix} G_{23}^T G_{34}^T =$$

$$\begin{pmatrix} x & + & + & x \\ x & + & + & x \\ 0 & + & + & x \\ 0 & 0 & 0 & x \end{pmatrix} G_{34}^T = \begin{pmatrix} x & x & + & + \\ x & x & + & + \\ 0 & x & + & + \\ 0 & 0 & + & + \end{pmatrix} = A_{k+1}.$$

Note that $A_{k+1} = R_k Q_k$ is Hessenberg. Need $2(n-1)$ Givens rotations. Don't need to keep the rotations G_{12}, G_{23} and G_{34}.

The QR Algorithm

Algorithm Compute one eigenvalue by

1. Hessenberg reduction $A := \text{Hess}(A)$.
2. Save elements $E := A(1\!:\!2, 1)$.
3. **while** $|A(n-1, n)| < \text{tol}$

    for $j = 1 : n-1$
    Create Rotation $G_{j,j+1}$ using E.
    Rotate rows $A := G_{j,j+1}A$.
    Save elements $E := A(j+1\!:\!j+2, j+1)$.
    Rotate columns $A := AG^T_{j,j+1}$.
    end

   end

Question What happens if eigenvalues are complex? Algorithms for computing eigenvalues are *iterative*. Why?

Theorem There is no explicit formula for the solution of polynomial equations of degree five or higher.

This is called the *Abel-Ruffini* theorem.

Remark If there were an explicit formula for eigenvalues we could use the *companion* matrix to get an explicit formula for polynomials.

Shifted QR algorithm

The convergence can be increased by using shifts.

$$A_k - s_k I = Q_k R_k, \qquad A_{k+1} = R_k Q_k + s_k I.$$

Lemma It holds that $A_{k+1} = Q_k^H A_k Q_k$ so A_k and A_{k+1} are similar.

Remark The element $(A_k)_{i,i-1}$ tends to zero with a rate equal to

$$\gamma = \left| \frac{\lambda_i - s_k}{\lambda_{i-1} - s_k} \right|.$$

Hence if $\lambda_i \approx s_k$ we get *very* fast convergence.

Shift selection strategies

Single shift Select $s_k = (A_k)_{n,n}$.

Example We select a Hessenberg matrix A and perform a few QR steps. In Matlab

```
>> A0 = [2 1 1 1; 1 3 1 1 ; 0 1 4 1 ; 0 0 1 5];
>> s=A0(4,4); [Q,R]=qr(A0-s*eye(4));
>> A1=R*Q+s*eye(4)
   A =                  A1 =
     2    1    1    1     1.50   0.08 -0.49 -0.89
     1    3    1    1     0.59   2.64 -0.45 -0.49
     0    1    4    1        0   0.54  4.60  0.80
     0    0    1    5        0      0  1.47  5.25
```

The matrix A_1 is Hessenberg and the new shift $s_1 = 5.25$.

We perform a few more QR steps to obtain

```
A4 =
      1.4007   -0.3106    0.2598    0.6786
      0.1916    2.7684   -0.2204    1.0612
           0    0.1136    3.8583    0.9829
           0         0   -0.0081    5.9727
```

Remark Fast convergence since $|(A_3)_{4,3}/(A_4)_{4,3}| \approx 29.2$.

Finally we see that

```
A6 =
      1.4164   -0.4228    0.2554    0.6012
      0.0948    2.7618   -0.2617    1.0482
           0    0.0493    3.8531    1.0535
           0         0   -0.0000    5.9688
```

Remark Here $|(A_6)_{4,3}| = 5.1875 \cdot 10^{-11}$. Proceed to use decoupling and shift with $s_k = (A_6)_{3,3}$.

Example We select a new Hessenberg matrix A and perform several QR steps using $s_k = (A_k)_{4,4}$. In Matlab

```
>> A= [ 2  -1   6   7
        3  -2   1   1
        0   4  -3   2
        0   0  -2   3];
>> I = eye(4);
>> for k=1:20
     s=A(4,4);[Q,R]=qr(A-s*I); A=R*Q+s*I;
   end
```

What happens now?

After 20 QR steps we obtain

```
A20 =
     -3.0327   -5.6708   -3.3364   -4.2027
      1.8228   -3.8883   -0.5321    0.8355
           0    0.0000    2.2067    5.4178
           0         0   -0.9909    4.7143
```

Observation The lower 2×2 block has the eigenvalues $\lambda_{3,4} = 3.46 \pm 1.94i$. We never introduce complex numbers in the computations.

Can still use decoupling. There is an analytic formula for the 2×2 case.

Double shift Select s_k as an eigenvalue of the block $(A_k)(n-1:n, n-1:n)$. In Matlab

```
for k=1:5
   s=max(eig(A(3:4,3:4)));
   [Q,R]=qr(A-s*eye(4));
   A=R*Q+s*eye(4);
end
```

The second shift is $s_2 = 3.2644 + 2.1334i$. Complex numbers are introduced.

After 5 QR steps with complex shifts we obtain

```
A5 =
-3.75-1.01i -5.55-0.38i  2.62+3.22i -0.10+3.04i
 1.66+0.00i -3.16+1.00i -0.28-0.51i  0.76-1.44i
 0.00+0.00i -0.05+0.00i  3.46-1.94i  3.15+3.99i
 0.00+0.00i  0.00+0.00i  0.00+0.00i  3.46+1.94i
```

Remark If A is real complex numbers should be avoided. Use decoupling on 2×2 blocks instead.

The Practical QR algorithm

A practical implementation includes the steps

- Hessenberg Reduction $A := \text{Hess}(A)$.
- Select a shift s_k using a strategy.
- The QR step is implemented using Givens rotations.
- If any $|A(j+1,j)| <$ tol then use *decoupling*:

$$A := \begin{pmatrix} A_1 & B \\ 0 & A_2 \end{pmatrix}.$$

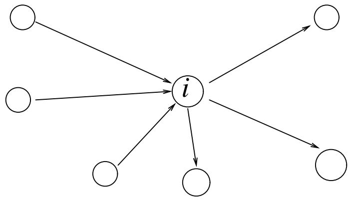
- If we find a 2×2 block. Use the analytic formula.
- Computed eigenvectors using *Inverse iteration*.

Remark The Matlab function `eig` implements this. Its difficult to set tolerances.

Application: Google Page Rank

Google ranks about $45 \cdot 10^9$ webb pages (2011). The ability to identify high quality webb pages is a large part of Googles success.

- The ranking is based on the link structure of the internet and has to be recomputed often.
- A *Web Crawler* downloads webb pages, collects *keywoards* for indexing, and finds links to, and from, webb pages.
- All webb pages relevant to a certain search phrase are retrived. They are displayed in the order given by the their *PageRank*.



Each webb page is assigned an index $i = 1, \ldots, N$.

The *PageRank* $r_i \in [0, 1])$ is a quality measure for webb pages. It is based on the set of *inlinks* I_i and *outlinks* O_i.

Idea Good webb pages get links from many other good webpages.

Definition The Google PageRank is r_i for webb page i satisfies,

$$r_i = \sum_{j \in I_i} \frac{r_j}{N_j}.$$

Remarks This means that the rank of a page j is divided equally between the its outlinks. This is a matrix equation

$$r = Ar, \qquad A_{i,j} = \begin{cases} 1/N_j, & \text{if page } j \text{ links to page } i, \\ 0, & \text{otherwise.} \end{cases}$$

Note If page j has at least one outlink then the corresponding column $A(:, j)$ sums to 1. A is the *Transition matrix*.

Definition If page j lacks outlinks then change the corresponding column to

$$A(:, j) = e/N, \qquad e = (1, 1, \ldots, 1)^T.$$

Lemma The largest eigenvalue of the modified Google transition matrix is $\lambda_{max} = 1$ and the corresponding eigenvector r has elements $0 \leq r_i \leq 1$.

Remarks We need one eigenvector of a matrix A of dimension $N = 45 \cdot 10^9$. The **only** realistic choice is the *Power Method*.