ALGORITHMS FOR EIGENVALUE PROBLEMS

- **The Power method**
- **The QR algorithm**
- **Practical QR algorithms: use of Hessenberg form and shifts**
- **The symmetric QR method**
- **The Jacobi method**

Basic algorithm: The power method

➤ Basic idea is to generate the sequence of vectors $A^k v_0$ where $v_0 \neq 0$ – then normalize.

➤ Most commonly used normalization: ensure that the largest component of the approximation is equal to one.

The Power Method

1. Choose a nonzero initial vector $v^{(0)}$.
2. For $k = 1, 2, \ldots,$ until convergence, Do:
3. $\quad \alpha_k = \text{argmax}_{i=1,\ldots,n} |(Av^{(k-1)})_i|$
4. $\quad v^{(k)} = \frac{1}{\alpha_k} A v^{(k-1)}$
5. EndDo

➤ $\text{argmax}_{i=1,..,n}|x_i| \equiv$ the component x_i with largest modulus

Convergence of the power method

THEOREM Assume there is one eigenvalue λ_1 of A, s.t. $|\lambda_1| > |\lambda_j|$, for $j \neq i$, and that λ_1 is semi-simple. Then either the initial vector $v^{(0)}$ has no component in Null$(A - \lambda_1 I)$ or $v^{(k)}$ converges to an eigenvector associated with λ_1 and $\alpha_k \to \lambda_1$.

Proof in the diagonalizable case.

➤ $v^{(k)}$ is = vector $A^k v^{(0)}$ normalized by a certain scalar $\hat{\alpha}_k$ in such a way that its largest component is 1.

➤ Decompose initial vector $v^{(0)}$ in the eigenbasis as:

$$v^{(0)} = \sum_{i=1}^{n} \gamma_i u_i$$

➤ Each u_i is an eigenvector associated with λ_i.

➤ Note that $A^k u_i = \lambda_i^k u_i$

$$\begin{aligned} v^{(k)} &= \frac{1}{scaling} \times \sum_{i=1}^{n} \lambda_i^k \gamma_i u_i \\ &= \frac{1}{scaling} \times \left[\lambda_1^k \gamma_1 u_1 + \sum_{i=2}^{n} \lambda_i^k \gamma_i u_i \right] \\ &= \frac{1}{scaling'} \times \left[u_1 + \sum_{i=2}^{n} \left(\frac{\lambda_i}{\lambda_1} \right)^k \frac{\gamma_i}{\gamma_1} u_i \right] \end{aligned}$$

➤ Second term inside bracket converges to zero. QED

➤ Proof suggests that the convergence factor is given by

$$\rho_D = \frac{|\lambda_2|}{|\lambda_1|}$$

where λ_2 is the second largest eigenvalue in modulus.

Example: Consider a 'Markov Chain' matrix of size $n = 55$. Dominant eigenvalues are $\lambda = 1$ and $\lambda = -1$ ➤ the power method applied directly to A fails. (Why?)

➤ We can consider instead the matrix $I + A$ The eigenvalue $\lambda = 1$ is then transformed into the (only) dominant eigenvalue $\lambda = 2$

Iteration	Norm of diff.	Res. norm	Eigenvalue
20	0.639D-01	0.276D-01	1.02591636
40	0.129D-01	0.513D-02	1.00680780
60	0.192D-02	0.808D-03	1.00102145
80	0.280D-03	0.121D-03	1.00014720
100	0.400D-04	0.174D-04	1.00002078
120	0.562D-05	0.247D-05	1.00000289
140	0.781D-06	0.344D-06	1.00000040
161	0.973D-07	0.430D-07	1.00000005

The Shifted Power Method

➤ In previous example shifted A into $B = A + I$ before applying power method. We could also iterate with $B(\sigma) = A + \sigma I$ for any positive σ

Example: With $\sigma = 0.1$ we get the following improvement.

Iteration	Norm of diff.	Res. Norm	Eigenvalue
20	0.273D-01	0.794D-02	1.00524001
40	0.729D-03	0.210D-03	1.00016755
60	0.183D-04	0.509D-05	1.00000446
80	0.437D-06	0.118D-06	1.00000011
88	0.971D-07	0.261D-07	1.00000002

➤ *Question:* What is the best shift-of-origin σ to use?

➤ Easy to answer the question when all eigenvalues are real.

Assume all eigenvalues are real and labeled decreasingly:

$$\lambda_1 > \lambda_2 \geq \lambda_2 \geq \cdots \geq \lambda_n,$$

Then: If we shift A to $A - \sigma I$:

The shift σ that yields the best convergence factor is:

$$\sigma_{opt} = \frac{\lambda_2 + \lambda_n}{2}$$

✍1 Plot a typical convergence factor $\phi(\sigma)$ as a function of σ. Determine the minimum value and prove the above result.

Inverse Iteration

Observation: The eigenvectors of A and A^{-1} are identical.

➤ Idea: use the power method on A^{-1}.

➤ Will compute the eigenvalues closest to zero.

➤ Shift-and-invert Use power method on $\boxed{(A - \sigma I)^{-1}}$.

➤ will compute eigenvalues closest to σ.

➤ Rayleigh-Quotient Iteration: use $\sigma = \frac{v^T A v}{v^T v}$ (best approximation to λ given v).

➤ Advantages: fast convergence in general.

➤ Drawbacks: need to factor A (or $A - \sigma I$) into LU.

The QR algorithm

➤ The most common method for solving small (dense) eigenvalue problems. The basic algorithm:

QR algorithm (basic)

1. Until Convergence Do:
2. Compute the QR factorization $A = QR$
3. Set $A := RQ$
4. EndDo

➤ "Until Convergence" means "Until A becomes close enough to an upper triangular matrix"

➤ Note: $A_{new} = RQ = Q^H(QR)Q = Q^H AQ$

➤ A_{new} *is Unitarily similar to* A → Spectrum does not change

➤ Convergence analysis complicated – but insight: we are implicitly doing a QR factorization of A^k:

	QR-Factorize:	Multiply backward:	
Step 1	$A_0 = Q_0R_0$	$A_1 = R_0Q_0$	
Step 2	$A_1 = Q_1R_1$	$A_2 = R_1Q_1$	
Step 3:	$A_2 = Q_2R_2$	$A_3 = R_2Q_2$	Then:

$$\begin{aligned}[Q_0Q_1Q_2][R_2R_1R_0] &= Q_0Q_1A_2R_1R_0 \\ &= Q_0(Q_1R_1)(Q_1R_1)R_0 \\ &= Q_0A_1A_1R_0, \qquad A_1 = R_0Q_0 \rightarrow \\ &= \underbrace{(Q_0R_0)}_{A}\underbrace{(Q_0R_0)}_{A}\underbrace{(Q_0R_0)}_{A} = A^3\end{aligned}$$

➤ $[Q_0Q_1Q_2][R_2R_1R_0]$ == QR factorization of A^3

➤ This helps analyze the algorithm (details skipped)

➤ Above basic algorithm is never used as is in practice. Two variations:

(1) Use shift of origin and

(2) Start by transforming A into an Hessenberg matrix

Practical QR algorithms: Shifts of origin

Observation: (from theory): Last row converges fastest. Convergence is dictated by

$$\frac{|\lambda_n|}{|\lambda_{n-1}|}$$

where we assume: $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_{n-1}| > |\lambda_n|$.

➤ For simplicity we will consider the situation when all eigenvalues are real.

➤ As $k \rightarrow \infty$ the last row (except $a_{nn}^{(k)}$) converges to zero quickly ..

➤ .. and $a_{nn}^{(k)}$ converges to eigenvalue of smallest magnitude.

$$A^{(k)} = \left(\begin{array}{ccccc|c} . & . & . & . & . & a \\ . & . & . & . & . & a \\ . & . & . & . & . & a \\ . & . & . & . & . & a \\ . & . & . & . & . & a \\ \hline a & a & a & a & a & a \end{array}\right)$$

➤ Idea: Apply QR algorithm to $A^{(k)} - \mu I$ with $\mu = a_{nn}^{(k)}$. Note: eigenvalues of $A^{(k)} - \mu I$ are shifted by μ (eigenvectors unchanged). $\rightarrow$ Shift matrix by $+\mu I$ after iteration.

QR algorithm with shifts

1. Until row $a_{in}, 1 \le i < n$ converges to zero DO:
2. Obtain next shift (e.g. $\mu = a_{nn}$)
3. $A - \mu I = QR$
5. Set $A := RQ + \mu I$
6. EndDo

➤ Convergence (of last row) is cubic at the limit! [for symmetric case]

➤ Result of algorithm:

$$A^{(k)} = \left(\begin{array}{ccccc|c} . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ \hline 0 & 0 & 0 & 0 & 0 & \lambda_n \end{array}\right)$$

➤ Next step: deflate, i.e., apply above algorithm to $(n-1) \times (n-1)$ upper block.

Practical algorithm: Use the Hessenberg Form

Recall: Upper Hessenberg matrix is such that

$$a_{ij} = 0 \text{ for } i > j + 1$$

Observation: QR algorithm preserves Hessenberg form (and tridiagonal symmetric form). Results in substantial savings: $O(n^2)$ flops per step instead of $O(n^3)$

Transformation to Hessenberg form

➤ Want $H_1 A H_1^T = H_1 A H_1$ to have the form shown on the right

➤ Consider the first step only on a 6×6 matrix

$$\begin{pmatrix} \star & \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star & \star \\ 0 & \star & \star & \star & \star & \star \\ 0 & \star & \star & \star & \star & \star \\ 0 & \star & \star & \star & \star & \star \\ 0 & \star & \star & \star & \star & \star \end{pmatrix}$$

➤ Choose a w in $H_1 = I - 2ww^T$ to make the first column have zeros from position 3 to n. So $w_1 = 0$.

➤ Apply to left: $B = H_1 A$

➤ Apply to right: $A_1 = BH_1$.

Main observation: the Householder matrix H_1 which transforms the column $A(2:n,1)$ into e_1 works only on rows 2 to n. When applying the transpose H_1 to the right of $B = H_1 A$, we observe that only columns 2 to n will be altered. So the first column will retain the desired pattern (zeros below row 2).

➤ Algorithm continues the same way for columns 2, ...,$n-2$.

QR algorithm for Hessenberg matrices

➤ Need the "Implicit Q theorem"

Suppose that $Q^T AQ$ is an unreduced upper Hessenberg matrix. Then columns 2 to n of Q are determined uniquely (up to signs) by the first column of Q.

➤ In other words if $V^T AV = G$ and $Q^T AQ = H$ are both Hessenberg and $V(:,1) = Q(:,1)$ then $V(:,i) = \pm Q(:,i)$ for $i = 2:n$.

Implication: To compute $A_{i+1} = Q_i^T AQ_i$ we can:

➤ Compute 1st column of Q_i [== scalar $\times A(:,1)$]

➤ Choose other columns so Q_i = unitary, and A_{i+1} = Hessenberg.

➤ W'll do this with Givens rotations:

Example: With $n = 5$:

$$A = \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{pmatrix}$$

1. Choose $G_1 = G(1,2,\theta_1)$ so that $(G_1^T A_0)_{21} = 0$

$$➤\ A_1 = G_1^T A G_1 = \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ + & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{pmatrix}$$

2. Choose $G_2 = G(2,3,\theta_2)$ so that $(G_2^T A_1)_{31} = 0$

$$➤\ A_2 = G_2^T A_1 G_2 = \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & + & * & * & * \\ 0 & 0 & 0 & * & * \end{pmatrix}$$

3. Choose $G_3 = G(3,4,\theta_3)$ so that $(G_3^T A_2)_{42} = 0$

$$➤\ A_3 = G_3^T A_2 G_3 = \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & + & * & * \end{pmatrix}$$

4. *Choose* $G_4 = G(4,5,\theta_4)$ *so that* $(G_4^T A_3)_{53} = 0$

➤ $A_4 = G_4^T A_3 G_4 = \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{pmatrix}$

➤ Process known as "Bulge chasing"

➤ Similar idea for the symmetric (tridiagonal) case

The QR algorithm for symmetric matrices

➤ Most common approach used : reduce to tridiagonal form and apply the QR algorithm with shifts.

➤ Householder transformation to Hessenberg form yields a tridiagonal matrix because

$$HAH^T = A_1$$

is symmetric and also of Hessenberg form ➤ it is tridiagonal symmetric.

Tridiagonal form preserved by QR similarity transformation

Practical method

➤ How to implement the QR algorithm with shifts?

➤ It is best to use Givens rotations – can do a shifted QR step without explicitly shifting the matrix..

➤ Two most popular shifts:

$s = a_{nn}$ and $s =$ smallest e.v. of $A(n-1:n, n-1:n)$

The Jacobi algorithm for symmetric matrices

➤ Main idea: Rotation matrices of the form

$$J(p,q,\theta) = \begin{pmatrix} 1 & \cdots & 0 & & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & \cdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ 0 & \cdots & 0 & & \cdots & & 1 \end{pmatrix} \begin{matrix} \\ \\ p \\ \\ q \\ \\ \\ \end{matrix}$$

$c = \cos\theta$ and $s = \sin\theta$ are so that $J(p,q,\theta)^T A J(p,q,\theta)$ has a zero in position (p,q) (and also (q,p))

➤ Frobenius norm of matrix is preserved – but diagonal elements become larger ➤ convergence to a diagonal.

- Let $B = J^T A J$ (where $J \equiv J_{p,q,\theta}$).
- Look at 2×2 matrix $B([p,q],[p,q])$ (matlab notation)
- Keep in mind that $a_{pq} = a_{qp}$ and $b_{pq} = b_{qp}$

$$\begin{pmatrix} b_{pp} & b_{pq} \\ b_{qp} & b_{qq} \end{pmatrix} = \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} a_{pp} & a_{pq} \\ a_{qp} & a_{qq} \end{pmatrix} \begin{pmatrix} c & s \\ -s & c \end{pmatrix} = \ldots$$
$$= \left[\begin{array}{c|c} c^2 a_{pp} + s^2 a_{qq} - 2sc\, a_{pq} & (c^2 - s^2) a_{pq} - sc(a_{qq} - a_{pp}) \\ * & c^2 a_{qq} + s^2 a_{pp} + 2sc\, a_{pq} \end{array} \right]$$

- Want: $(c^2 - s^2) a_{pq} - sc(a_{qq} - a_{pp}) = 0$

$$\frac{c^2 - s^2}{2sc} = \frac{a_{qq} - a_{pp}}{2a_{pq}} \equiv \tau$$

- Letting $t = s/c\ (= \tan\theta)$ $\to$ quad. equation

$$t^2 + 2\tau t - 1 = 0$$

- $t = -\tau \pm \sqrt{1+\tau^2} = \frac{1}{\tau \pm \sqrt{1+\tau^2}}$
- Select sign to get a smaller t so $\theta \le \pi/4$.
- Then : $c = \frac{1}{\sqrt{1+t^2}}$; $s = c * t$
- Implemented in matlab script `jacrot(A,p,q)` –

- Define: $A_O = A - \text{Diag}(A)$ $\equiv$ A 'with its diagonal entries replaced by zeros'
- Observations: (1) Unitary transformations preserve $\|.\|_F$. (2) Only changes are in rows and columns p and q.
- Let $B = J^T A J$ (where $J \equiv J_{p,q,\theta}$). Then:

$$a_{pp}^2 + a_{qq}^2 + 2a_{pq}^2 = b_{pp}^2 + b_{qq}^2 + 2b_{pq}^2 = b_{pp}^2 + b_{qq}^2$$

because $b_{pq} = 0$. Then, a little calculation leads to:

$$\begin{aligned} \|B_O\|_F^2 &= \|B\|_F^2 - \sum b_{ii}^2 = \|A\|_F^2 - \sum b_{ii}^2 \\ &= \|A\|_F^2 - \sum a_{ii}^2 + \sum a_{ii}^2 - \sum b_{ii}^2 \\ &= \|A_O\|_F^2 + (a_{pp}^2 + a_{qq}^2 - b_{pp}^2 - b_{qq}^2) \\ &= \|A_O\|_F^2 - 2a_{pq}^2 \end{aligned}$$

- $\|A_O\|_F$ will decrease from one step to the next.

✍2 Let $\|A_O\|_I = \max_{i \ne j} |a_{ij}|$. Show that

$$\|A_O\|_F \le \sqrt{n(n-1)}\|A_O\|_I$$

✍3 Use this to show convergence in the case when largest entry is zeroed at each step.