Numerical linear algebra III

Pinta Titus

October 25, 2018

Contents I

Contents II

Polar Decomposition

Let A be a positive definite $n \times n$ matrix

Polar Decomposition

Let A be a positive definite $n \times n$ matrix

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

Polar Decomposition

Let A be a positive definite $n \times n$ matrix

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

$$\mu_i = \sqrt{\lambda_i}$$

Polar Decomposition

Let A be a positive definite $n \times n$ matrix

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

$$\mu_i = \sqrt{\lambda_i}$$

$$B = QMQ^* \Rightarrow Q^*BB = A$$

Polar Decomposition

Let A be a positive definite $n \times n$ matrix

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

$$\mu_i = \sqrt{\lambda_i}$$

$$B = QMQ^* \Rightarrow Q^*BB = A$$

Let A be a full-rank $n \times n$ matrix

Polar Decomposition

Let A be a positive definite $n \times n$ matrix

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

$$\mu_i = \sqrt{\lambda_i}$$

$$B = QMQ^* \Rightarrow Q^* BB = A$$

Let A be a full-rank $n \times n$ matrix

$$P = \sqrt{A^* A}, \; Px_i = \lambda_i x_i$$

Polar Decomposition

Let A be a positive definite $n \times n$ matrix

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

$$\mu_i = \sqrt{\lambda_i}$$

$$B = QMQ^* \Rightarrow Q^*BB = A$$

Let A be a full-rank $n \times n$ matrix

$$P = \sqrt{A^*A},\ \ Px_i = \lambda_i x_i$$

$$U = \begin{bmatrix} \frac{1}{\lambda_1}Ax_1 & \dots \frac{1}{\lambda_n}Ax_n \end{bmatrix} \begin{bmatrix} x_1 & \dots & x_n \end{bmatrix}^*$$

Polar Decomposition

Let A be a positive definite $n \times n$ matrix

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

$$\mu_i = \sqrt{\lambda_i}$$

$$B = QMQ^* \Rightarrow Q^* BB = A$$

Let A be a full-rank $n \times n$ matrix

$$P = \sqrt{A^* A}, \; P x_i = \lambda_i x_i$$

$$U = \begin{bmatrix} \frac{1}{\lambda_1} A x_1 & \dots \frac{1}{\lambda_n} A x_n \end{bmatrix} \begin{bmatrix} x_1 & \dots & x_n \end{bmatrix}^*$$

$$U P x_i = A x_i \Rightarrow A = UP$$

Hadamard Inequality

Let A be a $n \times n$ matrix

Hadamard Inequality

Let A be a $n \times n$ matrix

$$b_i = \frac{a_i}{\|a_i\|_2}$$

Hadamard Inequality

Let A be a $n \times n$ matrix

$$b_i = \frac{a_i}{\|a_i\|_2}$$

$$P = B^*B, \;\; Px_i = \lambda_i x_i$$

Hadamard Inequality

Let A be a $n \times n$ matrix

$$b_i = \frac{a_i}{\|a_i\|_2}$$

$$P = B^* B, \;\; P x_i = \lambda_i x_i$$

$$\det P = \prod_{k=1}^{n} \lambda_k \leq \left(\frac{1}{n} \sum_{k=1}^{n} \lambda_k \right)^n \leq \left(\frac{1}{n} \operatorname{tr} P \right)^n = \left(\frac{1}{n} \sum_{k=1}^{n} b_k^* b_k \right)^n$$

Hadamard Inequality

Let A be a $n \times n$ matrix

$$b_i = \frac{a_i}{\|a_i\|_2}$$

$$P = B^* B, \;\; P x_i = \lambda_i x_i$$

$$\det P = \prod_{k=1}^{n} \lambda_k \leq \left(\frac{1}{n} \sum_{k=1}^{n} \lambda_k \right)^n \leq \left(\frac{1}{n} \operatorname{tr} P \right)^n = \left(\frac{1}{n} \sum_{k=1}^{n} b_k^* b_k \right)^n$$

$$|\det M| \leq \sqrt{\det P} = 1$$

Hadamard Inequality

Let A be a $n \times n$ matrix

$$b_i = \frac{a_i}{\|a_i\|_2}$$

$$P = B^* B, \;\; P x_i = \lambda_i x_i$$

$$\det P = \prod_{k=1}^{n} \lambda_k \le \left(\frac{1}{n} \sum_{k=1}^{n} \lambda_k \right)^n \le \left(\frac{1}{n} \operatorname{tr} P \right)^n = \left(\frac{1}{n} \sum_{k=1}^{n} b_k^* b_k \right)^n$$

$$|\det M| \le \sqrt{\det P} = 1$$

$$|\det A| \le \prod_{k=1}^{n} \|a_k\|_2$$

Orthogonal Iteration

Use power iteration on multiple vectors

Orthogonal Iteration

Use power iteration on multiple vectors
Use orthogonalization to avoid all vectors converging to one eigenvector

Orthogonal Iteration

Use power iteration on multiple vectors
Use orthogonalization to avoid all vectors converging to one eigenvector
Let $A \in M^{n \times n}$ positive definite

Orthogonal Iteration

Use power iteration on multiple vectors
Use orthogonalization to avoid all vectors converging to one eigenvector
Let $A \in M^{n \times n}$ positive definite

$$B_0 = \begin{bmatrix} b_1 & \ldots b_k \end{bmatrix}$$

Orthogonal Iteration

Use power iteration on multiple vectors
Use orthogonalization to avoid all vectors converging to one eigenvector
Let $A \in M^{n \times n}$ positive definite

$$B_0 = \begin{bmatrix} b_1 & \dots b_k \end{bmatrix}$$

$$B_{n+1} = AB_n$$

Use Gram-Schmidt to orthogonalize B_k

Orthogonal Iteration

Use power iteration on multiple vectors
Use orthogonalization to avoid all vectors converging to one eigenvector
Let $A \in M^{n \times n}$ positive definite

$$B_0 = \begin{bmatrix} b_1 & \dots b_k \end{bmatrix}$$

$$B_{n+1} = AB_n$$

Use Gram-Schmidt to orthogonalize B_k If x_i are the eigenvectors for k dominating eigenvalues

$$\lim_{n \to \infty} b_{in} = x_i$$

Orthogonal Iteration

Use power iteration on multiple vectors
Use orthogonalization to avoid all vectors converging to one eigenvector
Let $A \in M^{n\times n}$ positive definite

$$B_0 = \begin{bmatrix} b_1 & \dots b_k \end{bmatrix}$$

$$B_{n+1} = AB_n$$

Use Gram-Schmidt to orthogonalize B_k If x_i are the eigenvectors for k dominating eigenvalues

$$\lim_{n\to\infty} b_{in} = x_i$$

We compute $\begin{bmatrix} A^n b_1 & \dots A^n b_k \end{bmatrix}$

Schur's Lemma

$$A \in M^{m,n},\ \ q^*A = \lambda q$$

Schur's Lemma

$$A \in M^{m,n}, \;\; q^* A = \lambda q$$

Schur's Lemma

$$A \in M^{m,n}, \;\; q^* A = \lambda q$$

$$Q = [Q_1 \;\; q], \;\; QQ^* = I$$

Schur's Lemma

$$A \in M^{m,n}, \;\; q^* A = \lambda q$$

$$Q = [Q_1 \;\, q], \;\; QQ^* = I$$

$$Q^* A Q = \begin{bmatrix} Q_1^* A Q_1 & Q_1^* A q \\ q^* A Q_1 & q^* A q \end{bmatrix}$$

Schur's Lemma

$$A \in M^{m,n}, \ q^*A = \lambda q$$

$$Q = [Q_1 \ q], \ QQ^* = I$$

$$Q^*AQ = \begin{bmatrix} Q_1^*AQ_1 & Q_1^*Aq \\ q^*AQ_1 & q^*Aq \end{bmatrix}$$

$$q^*AQ_1 = \lambda q^*Q_1 = 0, \ q^*Aq = \lambda q^*q = \lambda$$

Schur's Lemma

$$A \in M^{m,n}, \;\; q^* A = \lambda q$$

$$Q = [Q_1 \;\; q], \;\; QQ^* = I$$

$$Q^* A Q = \begin{bmatrix} Q_1^* A Q_1 & Q_1^* A q \\ q^* A Q_1 & q^* A q \end{bmatrix}$$

$$q^* A Q_1 = \lambda q^* Q_1 = 0, \;\; q^* A q = \lambda q^* q = \lambda$$

$$Q^* A Q = \begin{bmatrix} Q_1^* A Q_1 & Q_1^* A q \\ 0 & \lambda \end{bmatrix}$$

$B = Q_1^* A Q_1$, apply the reasoning for an eigenpair of B

Schur's Lemma

$$A \in M^{m,n},\ \ q^*A = \lambda q$$

$$Q = [Q_1\ q],\ \ QQ^* = I$$

$$Q^*AQ = \begin{bmatrix} Q_1^*AQ_1 & Q_1^*Aq \\ q^*AQ_1 & q^*Aq \end{bmatrix}$$

$$q^*AQ_1 = \lambda q^*Q_1 = 0,\ \ q^*Aq = \lambda q^*q = \lambda$$

$$Q^*AQ = \begin{bmatrix} Q_1^*AQ_1 & Q_1^*Aq \\ 0 & \lambda \end{bmatrix}$$

$B = Q_1^*AQ_1$, apply the reasoning for an eigenpair of B
A is similar with an upper triangular matrix

Schur's Lemma

$$A \in M^{m,n}, \;\; q^* A = \lambda q$$

$$Q = [Q_1 \;\, q], \;\; QQ^* = I$$

$$Q^* A Q = \begin{bmatrix} Q_1^* A Q_1 & Q_1^* A q \\ q^* A Q_1 & q^* A q \end{bmatrix}$$

$$q^* A Q_1 = \lambda q^* Q_1 = 0, \;\; q^* A q = \lambda q^* q = \lambda$$

$$Q^* A Q = \begin{bmatrix} Q_1^* A Q_1 & Q_1^* A q \\ 0 & \lambda \end{bmatrix}$$

$B = Q_1^* A Q_1$, apply the reasoning for an eigenpair of B
A is similar with an upper triangular matrix $\Rightarrow$ proof of Schur's lemma

QR Algorithm

The problem with the algorithm: eigenvectors are required

QR Algorithm

The problem with the algorithm: eigenvectors are required

$$A = QR$$

QR Algorithm

The problem with the algorithm: eigenvectors are required

$$A = QR$$

$Q = [Q_1 \ q]$, q not an eigenvector

QR Algorithm

The problem with the algorithm: eigenvectors are required

$$A = QR$$

$Q = [Q_1 \ q]$, q not an eigenvector

$$A = QR \Rightarrow Q^*A = R$$

QR Algorithm

The problem with the algorithm: eigenvectors are required

$$A = QR$$

$Q = [Q_1 \;\; q]$, q not an eigenvector

$$A = QR \Rightarrow Q^* A = R$$

$$\begin{bmatrix} Q_1^* \\ q^* \end{bmatrix} A = \begin{bmatrix} R_1^* \\ r^* \end{bmatrix}$$

QR Algorithm

The problem with the algorithm: eigenvectors are required

$$A = QR$$

$Q = [Q_1 \;\; q]$, q not an eigenvector

$$A = QR \Rightarrow Q^* A = R$$

$$\begin{bmatrix} Q_1^* \\ q^* \end{bmatrix} A = \begin{bmatrix} R_1^* \\ r^* \end{bmatrix}$$

$$r^* = [0 \; 0 \; \dots \; 0 \; r_{nn}]$$

QR Algorithm

The problem with the algorithm: eigenvectors are required

$$A = QR$$

$Q = [Q_1 \; q]$, q not an eigenvector

$$A = QR \Rightarrow Q^* A = R$$

$$\begin{bmatrix} Q_1^* \\ q^* \end{bmatrix} A = \begin{bmatrix} R_1^* \\ r^* \end{bmatrix}$$

$$r^* = [0 \; 0 \; \ldots \; 0 \; r_{nn}]$$

$$q^* = r^* A^{-1}$$

QR Algorithm

The problem with the algorithm: eigenvectors are required

$$A = QR$$

$Q = [Q_1 \;\; q]$, q not an eigenvector

$$A = QR \Rightarrow Q^* A = R$$

$$\begin{bmatrix} Q_1^* \\ q^* \end{bmatrix} A = \begin{bmatrix} R_1^* \\ r^* \end{bmatrix}$$

$$r^* = [0 \; 0 \;\; \ldots \;\; 0 \;\; r_{nn}]$$

$$q^* = r^* A^{-1}$$

q^* is an iteration of the power method for A^{-1} starting from r^*

QR Algorithm

$$A = QR, \ \widehat{A} = RQ$$

QR Algorithm

$$A = QR,\ \widehat{A} = RQ$$

$$A_k = Q_k R_k,\ A_{k+1} = R_k Q_k$$

QR Algorithm

$$A = QR, \ \widehat{A} = RQ$$

$$A_k = Q_k R_k, \ A_{k+1} = R_k Q_k$$

$$A_{k+1} = Q_k^* A_k Q_k \Rightarrow A_{k+1} = Q_k^* Q_{k-1}^* \dots Q_1^* A_1 Q_1 \dots Q_{k-1} Q_k$$

QR Algorithm

$$A = QR, \;\; \widehat{A} = RQ$$

$$A_k = Q_k R_k, \;\; A_{k+1} = R_k Q_k$$

$$A_{k+1} = Q_k^* A_k Q_k \Rightarrow A_{k+1} = Q_k^* Q_{k-1}^* \dots Q_1^* A_1 Q_1 \dots Q_{k-1} Q_k$$

$$\begin{aligned} Q_1 Q_2 \dots Q_{k+1} R_{k+1} \dots R_2 R_1 = & Q_1 Q_2 \dots Q_k A_{k+1} R_k \dots R_2 R_1 = \\ = Q_1 Q_2 \dots Q_k Q_k^* Q_{k-1}^* & \dots Q_1^* A_1 Q_1 \dots Q_{k-1} Q_k R_k R_{k-1} \dots R_2 R_1 = \\ = A_1 Q_1 \dots & Q_{k-1} Q_k R_k R_{k-1} \dots R_2 R_1 = A_1^{k+1} \end{aligned}$$

QR Algorithm

$$A = QR, \ \widehat{A} = RQ$$

$$A_k = Q_k R_k, \ A_{k+1} = R_k Q_k$$

$$A_{k+1} = Q_k^* A_k Q_k \Rightarrow A_{k+1} = Q_k^* Q_{k-1}^* \dots Q_1^* A_1 Q_1 \dots Q_{k-1} Q_k$$

$$\begin{aligned} Q_1 Q_2 \dots Q_{k+1} R_{k+1} \dots R_2 R_1 =& Q_1 Q_2 \dots Q_k A_{k+1} R_k \dots R_2 R_1 = \\ = Q_1 Q_2 \dots Q_k Q_k^* Q_{k-1}^* \dots Q_1^* A_1 Q_1 \dots Q_{k-1} Q_k R_k R_{k-1} \dots R_2 R_1 = \\ = A_1 Q_1 \dots Q_{k-1} Q_k R_k R_{k-1} \dots R_2 R_1 = A_1^{k+1} \end{aligned}$$

$$U_k = Q_1 \dots Q_k$$

$$T_k = R_1 \dots R_k$$

QR Algorithm

$$A = QR,\ \widehat{A} = RQ$$

$$A_k = Q_k R_k,\ A_{k+1} = R_k Q_k$$

$$A_{k+1} = Q_k^* A_k Q_k \Rightarrow A_{k+1} = Q_k^* Q_{k-1}^* \dots Q_1^* A_1 Q_1 \dots Q_{k-1} Q_k$$

$$\begin{aligned} Q_1 Q_2 \dots Q_{k+1} R_{k+1} \dots R_2 R_1 &= Q_1 Q_2 \dots Q_k A_{k+1} R_k \dots R_2 R_1 = \\ &= Q_1 Q_2 \dots Q_k Q_k^* Q_{k-1}^* \dots Q_1^* A_1 Q_1 \dots Q_{k-1} Q_k R_k R_{k-1} \dots R_2 R_1 = \\ &= A_1 Q_1 \dots Q_{k-1} Q_k R_k R_{k-1} \dots R_2 R_1 = A_1^{k+1} \end{aligned}$$

$$U_k = Q_1 \dots Q_k$$
$$T_k = R_1 \dots R_k$$

$$A_k = U_k^* A_1 U_k,\ U_k T_k = A_1^k$$

QR Algorithm

We want to show $A_k \approx R$ upper triangular as $k \to \infty$

QR Algorithm

We want to show $A_k \approx R$ upper triangular as $k \to \infty$
Assume A has distinct eigenvalues in absolute value $\in \mathbb{R}^*$

QR Algorithm

We want to show $A_k \approx R$ upper triangular as $k \to \infty$
Assume A has distinct eigenvalues in absolute value $\in \mathbb{R}^*$

$$A = A_1 = V\Lambda V^{-1},\ U_k T_k = A_1^k = V\Lambda^k V^{-1}$$

QR Algorithm

We want to show $A_k \approx R$ upper triangular as $k \to \infty$
Assume A has distinct eigenvalues in absolute value $\in \mathbb{R}^*$

$$A = A_1 = V\Lambda V^{-1},\ U_k T_k = A_1^k = V\Lambda^k V^{-1}$$

$$U_k = V\Lambda^k V^{-1} T_k^{-1},\ U_k^* = T_k V\Lambda^{-k} V^{-1}$$

QR Algorithm

We want to show $A_k \approx R$ upper triangular as $k \to \infty$
Assume A has distinct eigenvalues in absolute value $\in \mathbb{R}^*$

$$A = A_1 = V\Lambda V^{-1},\; U_k T_k = A_1^k = V\Lambda^k V^{-1}$$

$$U_k = V\Lambda^k V^{-1} T_k^{-1},\; U_k^* = T_k V\Lambda^{-k} V^{-1}$$

$$A_k = T_k V\Lambda^k V^{-1} A_1 V\Lambda^{-k} V^{-1} T_k^{-1} = T_k V\Lambda V^{-1} T_k^{-}1$$

QR Algorithm

We want to show $A_k \approx R$ upper triangular as $k \to \infty$
Assume A has distinct eigenvalues in absolute value $\in \mathbb{R}^*$

$$A = A_1 = V\Lambda V^{-1},\ U_k T_k = A_1^k = V\Lambda^k V^{-1}$$

$$U_k = V\Lambda^k V^{-1} T_k^{-1},\ U_k^* = T_k V\Lambda^{-k} V^{-1}$$

$$A_k = T_k V\Lambda^k V^{-1} A_1 V\Lambda^{-k} V^{-1} T_k^{-1} = T_k V\Lambda V^{-1} T_k^- 1$$

$$A_k = T_k V\Lambda V^{-1} T_k^{-1} = T_k A T_k^{-1}$$

QR Algorithm

We want to show $A_k \approx R$ upper triangular as $k \to \infty$
Assume A has distinct eigenvalues in absolute value $\in \mathbb{R}^*$

$$A = A_1 = V \Lambda V^{-1}, \; U_k T_k = A_1^k = V \Lambda^k V^{-1}$$

$$U_k = V \Lambda^k V^{-1} T_k^{-1}, \; U_k^* = T_k V \Lambda^{-k} V^{-1}$$

$$A_k = T_k V \Lambda^k V^{-1} A_1 V \Lambda^{-k} V^{-1} T_k^{-1} = T_k V \Lambda V^{-1} T_k^- 1$$

$$A_k = T_k V \Lambda V^{-1} T_k^{-1} = T_k A T_k^{-1}$$

$$A_k T_k = T_k A$$

Convergence of the QR Algorithm

Assume:

$$\lambda_1 > \lambda_2 > \ldots > \lambda_n > 0$$

Convergence of the QR Algorithm

Assume:

$$\lambda_1 > \lambda_2 > \ldots > \lambda_n > 0$$

$A = Q\Lambda Q^*$ the eigenvalue decomposition of A, A positive-definite

Convergence of the QR Algorithm

Assume:

$$\lambda_1 > \lambda_2 > \ldots > \lambda_n > 0$$

$A = Q\Lambda Q^*$ the eigenvalue decomposition of A, A positive-definite
$Q^* = LU$ the LU factorization, U has non-negative components

Convergence of the QR Algorithm

Assume:

$$\lambda_1 > \lambda_2 > \ldots > \lambda_n > 0$$

$A = Q\Lambda Q^*$ the eigenvalue decomposition of A, A positive-definite
$Q^* = LU$ the LU factorization, U has non-negative components

$$A^k = Q\Lambda^k LU = U_k T_k$$

Convergence of the QR Algorithm

Assume:

$$\lambda_1 > \lambda_2 > \ldots > \lambda_n > 0$$

$A = Q \Lambda Q^*$ the eigenvalue decomposition of A, A positive-definite
$Q^* = LU$ the LU factorization, U has non-negative components

$$A^k = Q \Lambda^k LU = U_k T_k$$

$$Q \Lambda^k L \Lambda^{-k} = U_k T_k U^{-1} \Lambda^{-k}$$

Convergence of the QR Algorithm

Assume:

$$\lambda_1 > \lambda_2 > \ldots > \lambda_n > 0$$

$A = Q\Lambda Q^*$ the eigenvalue decomposition of A, A positive-definite
$Q^* = LU$ the LU factorization, U has non-negative components

$$A^k = Q\Lambda^k LU = U_k T_k$$

$$Q\Lambda^k L\Lambda^{-k} = U_k T_k U^{-1}\Lambda^{-k}$$

$$(\Lambda^k L\Lambda^{-k})_{ij} = \begin{cases} l_{ij}\left(\frac{\lambda_i}{\lambda_j}\right)^k, & i > j \\ 1, & i = j \\ 0, & i < j \end{cases}$$

Convergence of the QR Algorithm

Assume:

$$\lambda_1 > \lambda_2 > \ldots > \lambda_n > 0$$

$A = Q\Lambda Q^*$ the eigenvalue decomposition of A, A positive-definite
$Q^* = LU$ the LU factorization, U has non-negative components

$$A^k = Q\Lambda^k LU = U_k T_k$$

$$Q\Lambda^k L\Lambda^{-k} = U_k T_k U^{-1}\Lambda^{-k}$$

$$(\Lambda^k L\Lambda^{-k})_{ij} = \begin{cases} l_{ij}\left(\frac{\lambda_i}{\lambda_j}\right)^k, & i > j \\ 1, & i = j \\ 0, & i < j \end{cases}$$

$$\Lambda^k L\Lambda^{-k} \to I$$

Convergence of the QR Algorithm

$$Q\Lambda^k L \Lambda^{-k} \to Q$$

Convergence of the QR Algorithm

$$Q\Lambda^k L\Lambda^{-k} \to Q$$

$$U_k T_k U^{-1}\Lambda^{-k} \to Q$$

Convergence of the QR Algorithm

$$Q\Lambda^k L\Lambda^{-k} \to Q$$

$$U_k T_k U^{-1}\Lambda^{-k} \to Q$$

$Q = QI$ the QR factorization of Q

Convergence of the QR Algorithm

$$Q\Lambda^k L\Lambda^{-k} \to Q$$

$$U_k T_k U^{-1}\Lambda^{-k} \to Q$$

$Q = QI$ the QR factorization of Q
U_k is orthonormal and $T_k U^{-1}\Lambda^{-k}$ is upper-triangular

Convergence of the QR Algorithm

$$Q\Lambda^k L \Lambda^{-k} \to Q$$

$$U_k T_k U^{-1} \Lambda^{-k} \to Q$$

$Q = QI$ the QR factorization of Q
U_k is orthonormal and $T_k U^{-1} \Lambda^{-k}$ is upper-triangular
$T_k,\ U^{-1} \Lambda^{-k}$ have positive entries

Convergence of the QR Algorithm

$$Q\Lambda^k L \Lambda^{-k} \to Q$$

$$U_k T_k U^{-1} \Lambda^{-k} \to Q$$

$Q = QI$ the QR factorization of Q
U_k is orthonormal and $T_k U^{-1} \Lambda^{-k}$ is upper-triangular
$T_k,\ U^{-1}\Lambda^{-k}$ have positive entries

$$U_k \to Q$$

Convergence of the QR Algorithm

$$Q\Lambda^k L\Lambda^{-k} \to Q$$

$$U_k T_k U^{-1}\Lambda^{-k} \to Q$$

$Q = QI$ the QR factorization of Q
U_k is orthonormal and $T_k U^{-1}\Lambda^{-k}$ is upper-triangular
$T_k, \; U^{-1}\Lambda^{-k}$ have positive entries

$$U_k \to Q$$

$$A_k = U_k^* A U_k \to Q^* A Q = \Lambda$$

Cholesky Iteration

B positive definite $n \times n$ matrix

Cholesky Iteration

B positive definite $n \times n$ matrix The Cholesky iteration:

$$B_0 = B, \; B_k = C_k^* C_k, \; B_{k+1} = C_k C_k^*$$

Cholesky Iteration

B positive definite $n \times n$ matrix The Cholesky iteration:

$$B_0 = B, \; B_k = C_k^* C_k, \; B_{k+1} = C_k C_k^*$$

$$B = C^* C, \; \widehat{B} = CC^*$$

Cholesky Iteration

B positive definite $n \times n$ matrix The Cholesky iteration:

$$B_0 = B,\ B_k = C_k^* C_k,\ B_{k+1} = C_k C_k^*$$

$$B = C^* C,\ \widehat{B} = CC^*$$

$$B_{ij} = \sum_{p=1}^{i} C_{ip}^* C_{pj} = \sum_{p=1}^{i} C_{pi} C_{pj}$$

Cholesky Iteration

B positive definite $n \times n$ matrix The Cholesky iteration:

$$B_0 = B,\ B_k = C_k^* C_k,\ B_{k+1} = C_k C_k^*$$

$$B = C^* C,\ \widehat{B} = CC^*$$

$$B_{ij} = \sum_{p=1}^{i} C_{ip}^* C_{pj} = \sum_{p=1}^{i} C_{pi} C_{pj}$$

$$\widehat{B}_{ij} = \sum_{p=j}^{n} C_{ip} C_{pj}^* = \sum_{p=j}^{n} C_{ip} C_{jp}$$

Cholesky Iteration

B positive definite $n \times n$ matrix The Cholesky iteration:

$$B_0 = B, \; B_k = C_k^* C_k, \; B_{k+1} = C_k C_k^*$$

$$B = C^* C, \; \widehat{B} = CC^*$$

$$B_{ij} = \sum_{p=1}^{i} C_{ip}^* C_{pj} = \sum_{p=1}^{i} C_{pi} C_{pj}$$

$$\widehat{B}_{ij} = \sum_{p=j}^{n} C_{ip} C_{pj}^* = \sum_{p=j}^{n} C_{ip} C_{jp}$$

$$B_{kk} = \sum_{p=1}^{k} C_{kp}^2, \; \widehat{B}_{kk} = \sum_{p=k}^{n} C_{pk}^2$$

Cholesky Iteration

$$\sum_{k=1}^{m} B_{kk} = \sum_{k=1}^{m} \sum_{p=1}^{k} C_{kp}^2 = \sum_{k=1}^{m} \sum_{p=1}^{m} C_{kp}^2 = \sum_{i=1}^{m} \sum_{j=1}^{m} C_{ji}^2, \quad m \leq n$$

Cholesky Iteration

$$\sum_{k=1}^{m} B_{kk} = \sum_{k=1}^{m} \sum_{p=1}^{k} C_{kp}^2 = \sum_{k=1}^{m} \sum_{p=1}^{m} C_{kp}^2 = \sum_{i=1}^{m} \sum_{j=1}^{m} C_{ji}^2, \quad m \le n$$

$$\sum_{k=1}^{m} \widehat{B}_{kk} = \sum_{k=1}^{m} \sum_{p=k}^{n} C_{pk}^2 = \sum_{k=1}^{m} \sum_{p=1}^{n} C_{pk}^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} C_{ji}^2, \quad m \le n$$

Cholesky Iteration

$$\sum_{k=1}^{m} B_{kk} = \sum_{k=1}^{m}\sum_{p=1}^{k} C_{kp}^2 = \sum_{k=1}^{m}\sum_{p=1}^{m} C_{kp}^2 = \sum_{i=1}^{m}\sum_{j=1}^{m} C_{ji}^2, \quad m \le n$$

$$\sum_{k=1}^{m} \widehat{B}_{kk} = \sum_{k=1}^{m}\sum_{p=k}^{n} C_{pk}^2 = \sum_{k=1}^{m}\sum_{p=1}^{n} C_{pk}^2 = \sum_{i=1}^{m}\sum_{j=1}^{n} C_{ji}^2, \quad m \le n$$

$$\sum_{k=1}^{m} \widehat{B}_{kk} - B_{kk} = \sum_{i=1}^{m}\sum_{j=m+1}^{n} C_{ji}^2$$

Cholesky Iteration

$$\sum_{k=1}^{m} B_{kk} = \sum_{k=1}^{m}\sum_{p=1}^{k} C_{kp}^2 = \sum_{k=1}^{m}\sum_{p=1}^{m} C_{kp}^2 = \sum_{i=1}^{m}\sum_{j=1}^{m} C_{ji}^2, \quad m \le n$$

$$\sum_{k=1}^{m} \widehat{B}_{kk} = \sum_{k=1}^{m}\sum_{p=k}^{n} C_{pk}^2 = \sum_{k=1}^{m}\sum_{p=1}^{n} C_{pk}^2 = \sum_{i=1}^{m}\sum_{j=1}^{n} C_{ji}^2, \quad m \le n$$

$$\sum_{k=1}^{m} \widehat{B}_{kk} - B_{kk} = \sum_{i=1}^{m}\sum_{j=m+1}^{n} C_{ji}^2$$

$$m = n \Rightarrow \operatorname{tr}(B) = \operatorname{tr}(\widehat{B})$$

Cholesky Iteration

$$\sum_{k=1}^{m} B_{kk} = \sum_{k=1}^{m}\sum_{p=1}^{k} C_{kp}^2 = \sum_{k=1}^{m}\sum_{p=1}^{m} C_{kp}^2 = \sum_{i=1}^{m}\sum_{j=1}^{m} C_{ji}^2, \quad m \le n$$

$$\sum_{k=1}^{m} \widehat{B}_{kk} = \sum_{k=1}^{m}\sum_{p=k}^{n} C_{pk}^2 = \sum_{k=1}^{m}\sum_{p=1}^{n} C_{pk}^2 = \sum_{i=1}^{m}\sum_{j=1}^{n} C_{ji}^2, \quad m \le n$$

$$\sum_{k=1}^{m} \widehat{B}_{kk} - B_{kk} = \sum_{i=1}^{m}\sum_{j=m+1}^{n} C_{ji}^2$$

$$m = n \Rightarrow \mathrm{tr}(B) = \mathrm{tr}(\widehat{B})$$

$$m < n \Rightarrow \sum_{k=1}^{m} \widehat{B}_{kk} \ge \sum_{k=1}^{m} B_{kk}$$

Cholesky Iteration

$$\mathrm{tr}(B) = \mathrm{tr}(B_0) = \ldots = \mathrm{tr}(B_k)$$

Cholesky Iteration

$$\mathrm{tr}(B) = \mathrm{tr}(B_0) = \ldots = \mathrm{tr}(B_k)$$

$$\widehat{B}_{mm} - B_{mm} = \sum_{k=1}^{m}(\widehat{B}_{kk} - B_{kk}) - \sum_{k=1}^{m-1}(\widehat{B}_{kk} - B_{kk}) \geq 0$$

Cholesky Iteration

$$\mathrm{tr}(B) = \mathrm{tr}(B_0) = \ldots = \mathrm{tr}(B_k)$$

$$\widehat{B}_{mm} - B_{mm} = \sum_{k=1}^{m} (\widehat{B}_{kk} - B_{kk}) - \sum_{k=1}^{m-1} (\widehat{B}_{kk} - B_{kk}) \geq 0$$

$(B_k)_{ii}$ is monotonous and bounded

$$\lim_{k\to\infty} (B_k)_{ii} = (B_\infty)_{ii}$$

Cholesky Iteration

$$\mathrm{tr}(B) = \mathrm{tr}(B_0) = \ldots = \mathrm{tr}(B_k)$$

$$\widehat{B}_{mm} - B_{mm} = \sum_{k=1}^{m} (\widehat{B}_{kk} - B_{kk}) - \sum_{k=1}^{m-1} (\widehat{B}_{kk} - B_{kk}) \geq 0$$

$(B_k)_{ii}$ is monotonous and bounded

$$\lim_{k \to \infty} (B_k)_{ii} = (B_\infty)_{ii}$$

$$\sum_{p=1}^{m} \widehat{B}_{pp} - B_{pp} = \sum_{i=1}^{m} \sum_{j=m+1}^{n} C_{ji}^2 \to 0$$

Cholesky Iteration

$$\mathrm{tr}(B) = \mathrm{tr}(B_0) = \ldots = \mathrm{tr}(B_k)$$

$$\widehat{B}_{mm} - B_{mm} = \sum_{k=1}^{m} (\widehat{B}_{kk} - B_{kk}) - \sum_{k=1}^{m-1} (\widehat{B}_{kk} - B_{kk}) \geq 0$$

$(B_k)_{ii}$ is monotonous and bounded

$$\lim_{k\to\infty} (B_k)_{ii} = (B_\infty)_{ii}$$

$$\sum_{p=1}^{m} \widehat{B}_{pp} - B_{pp} = \sum_{i=1}^{m} \sum_{j=m+1}^{n} C_{ji}^2 \to 0$$

$$C_{ji} \to 0, \quad j \geq i+1$$

Cholesky Iteration

$$\mathrm{tr}(B) = \mathrm{tr}(B_0) = \ldots = \mathrm{tr}(B_k)$$

$$\widehat{B}_{mm} - B_{mm} = \sum_{k=1}^{m} (\widehat{B}_{kk} - B_{kk}) - \sum_{k=1}^{m-1} (\widehat{B}_{kk} - B_{kk}) \geq 0$$

$(B_k)_{ii}$ is monotonous and bounded

$$\lim_{k \to \infty} (B_k)_{ii} = (B_\infty)_{ii}$$

$$\sum_{p=1}^{m} \widehat{B}_{pp} - B_{pp} = \sum_{i=1}^{m} \sum_{j=m+1}^{n} C_{ji}^2 \to 0$$

$$C_{ji} \to 0, \quad j \geq i+1$$

C converges to a diagonal matrix

Cholesky Iteration

$$\mathrm{tr}(B) = \mathrm{tr}(B_0) = \ldots = \mathrm{tr}(B_k)$$

$$\widehat{B}_{mm} - B_{mm} = \sum_{k=1}^{m} (\widehat{B}_{kk} - B_{kk}) - \sum_{k=1}^{m-1} (\widehat{B}_{kk} - B_{kk}) \geq 0$$

$(B_k)_{ii}$ is monotonous and bounded

$$\lim_{k \to \infty} (B_k)_{ii} = (B_\infty)_{ii}$$

$$\sum_{p=1}^{m} \widehat{B}_{pp} - B_{pp} = \sum_{i=1}^{m} \sum_{j=m+1}^{n} C_{ji}^2 \to 0$$

$$C_{ji} \to 0, \quad j \geq i + 1$$

C converges to a diagonal matrix

B_k converges to a diagonal matrix

Another Convergence Proof for QR

For A positive definite

$$B = A^2 = A^* A$$

Another Convergence Proof for QR

For A positive definite

$$B = A^2 = A^* A$$

$$x \neq 0 \Rightarrow x^* A x \geq 0 \Rightarrow x^* Q^* A Q x \geq 0 \Rightarrow x^* A_k x \geq 0$$

Another Convergence Proof for QR

For A positive definite

$$B = A^2 = A^* A$$

$$x \neq 0 \Rightarrow x^* A x \geq 0 \Rightarrow x^* Q^* A Q x \geq 0 \Rightarrow x^* A_k x \geq 0$$

$$A = QR \Rightarrow B = R^* Q^* Q R = R^* R$$

Another Convergence Proof for QR

For A positive definite

$$B = A^2 = A^* A$$

$$x \neq 0 \Rightarrow x^* A x \geq 0 \Rightarrow x^* Q^* A Q x \geq 0 \Rightarrow x^* A_k x \geq 0$$

$$A = QR \Rightarrow B = R^* Q^* Q R = R^* R$$

$$A_0 = A, \; A_k = Q_k R_k, \; A_{k+1} = R_k Q_k$$

Another Convergence Proof for QR

For A positive definite

$$B = A^2 = A^*A$$

$$x \neq 0 \Rightarrow x^*Ax \geq 0 \Rightarrow x^*Q^*AQx \geq 0 \Rightarrow x^*A_kx \geq 0$$

$$A = QR \Rightarrow B = R^*Q^*QR = R^*R$$

$$A_0 = A,\; A_k = Q_kR_k,\; A_{k+1} = R_kQ_k$$

$$A_k^2 = B_k = R_k^*R_k$$

Another Convergence Proof for QR

For A positive definite

$$B = A^2 = A^* A$$

$$x \neq 0 \Rightarrow x^* A x \geq 0 \Rightarrow x^* Q^* A Q x \geq 0 \Rightarrow x^* A_k x \geq 0$$

$$A = QR \Rightarrow B = R^* Q^* Q R = R^* R$$

$$A_0 = A,\ A_k = Q_k R_k,\ A_{k+1} = R_k Q_k$$

$$A_k^2 = B_k = R_k^* R_k$$

$$A_{k+1}^2 = A_{k+1} A_{k+1}^* = R_k R_k^* = B_{k+1}$$

Another Convergence Proof for QR

For A positive definite

$$B = A^2 = A^* A$$

$$x \neq 0 \Rightarrow x^* A x \geq 0 \Rightarrow x^* Q^* A Q x \geq 0 \Rightarrow x^* A_k x \geq 0$$

$$A = QR \Rightarrow B = R^* Q^* Q R = R^* R$$

$$A_0 = A,\ A_k = Q_k R_k,\ A_{k+1} = R_k Q_k$$

$$A_k^2 = B_k = R_k^* R_k$$

$$A_{k+1}^2 = A_{k+1} A_{k+1}^* = R_k R_k^* = B_{k+1}$$

The QR iteration for A is the Cholesky iteration for $B \Rightarrow$ converges

Improved Convergence Result for QR

For A normal

Improved Convergence Result for QR

For A normal

$$\exists P^2 = AA^* = A^*A$$

Improved Convergence Result for QR

For A normal

$$\exists P^2 = AA^* = A^*A$$

$A = UP$ polar decomposition of $A = A_0$

Improved Convergence Result for QR

For A normal

$$\exists P^2 = AA^* = A^*A$$

$A = UP$ polar decomposition of $A = A_0$

$$P_0 = Q_0 R_0 \Rightarrow A_0 = U_0 Q_0 R_0$$

Improved Convergence Result for QR

For A normal

$$\exists P^2 = AA^* = A^*A$$

$A = UP$ polar decomposition of $A = A_0$

$$P_0 = Q_0 R_0 \Rightarrow A_0 = U_0 Q_0 R_0$$

$$A_0 = U_0 Q_0 R_0 \Rightarrow A_1 = R_0 U_0 Q_0$$

Improved Convergence Result for QR

For A normal

$$\exists P^2 = AA^* = A^*A$$

$A = UP$ polar decomposition of $A = A_0$

$$P_0 = Q_0R_0 \Rightarrow A_0 = U_0Q_0R_0$$

$$A_0 = U_0Q_0R_0 \Rightarrow A_1 = R_0U_0Q_0$$

$$U_0Q_0R_0 = Q_0R_0U_0 \Rightarrow R_0U_0 = Q_0^*U_0Q_0R_0$$

Improved Convergence Result for QR

For A normal

$$\exists P^2 = AA^* = A^*A$$

$A = UP$ polar decomposition of $A = A_0$

$$P_0 = Q_0 R_0 \Rightarrow A_0 = U_0 Q_0 R_0$$

$$A_0 = U_0 Q_0 R_0 \Rightarrow A_1 = R_0 U_0 Q_0$$

$$U_0 Q_0 R_0 = Q_0 R_0 U_0 \Rightarrow R_0 U_0 = Q_0^* U_0 Q_0 R_0$$

$$A_1 = Q_0^* U_0 Q_0 R_0 Q_0 = Q_0^* U_0 Q_0 Q_1 R_1$$

Improved Convergence Result for QR

For A normal

$$\exists P^2 = AA^* = A^*A$$

$A = UP$ polar decomposition of $A = A_0$

$$P_0 = Q_0 R_0 \Rightarrow A_0 = U_0 Q_0 R_0$$

$$A_0 = U_0 Q_0 R_0 \Rightarrow A_1 = R_0 U_0 Q_0$$

$$U_0 Q_0 R_0 = Q_0 R_0 U_0 \Rightarrow R_0 U_0 = Q_0^* U_0 Q_0 R_0$$

$$A_1 = Q_0^* U_0 Q_0 R_0 Q_0 = Q_0^* U_0 Q_0 Q_1 R_1$$

$$U_1 = Q_0^* U_0 Q_0$$

Improved Convergence Result for QR

Use induction, for k to prove A_k normal and

$$A_k = U_k Q_k R_k$$

Improved Convergence Result for QR

Use induction, for k to prove A_k normal and

$$A_k = U_k Q_k R_k$$

By definition $R_k Q_k = Q_{k+1} R_{k+1} = P_k$

Improved Convergence Result for QR

Use induction, for k to prove A_k normal and

$$A_k = U_k Q_k R_k$$

By definition $R_k Q_k = Q_{k+1} R_{k+1} = P_k$

$$A_{k+1} = U_k Q_k R_k = U_k P_k$$

$$U_k = Q_k^* \dots Q_0^* U Q_0 \dots Q_k$$

Improved Convergence Result for QR

Use induction, for k to prove A_k normal and

$$A_k = U_k Q_k R_k$$

By definition $R_k Q_k = Q_{k+1} R_{k+1} = P_k$

$$A_{k+1} = U_k Q_k R_k = U_k P_k$$

$$U_k = Q_k^* \dots Q_0^* U Q_0 \dots Q_k$$

P_k converges to a diagonal matrix and U_k is a rotation

Improved Convergence Result for QR

Use induction, for k to prove A_k normal and

$$A_k = U_k Q_k R_k$$

By definition $R_k Q_k = Q_{k+1} R_{k+1} = P_k$

$$A_{k+1} = U_k Q_k R_k = U_k P_k$$

$$U_k = Q_k^* \dots Q_0^* U Q_0 \dots Q_k$$

P_k converges to a diagonal matrix and U_k is a rotation
In a sense A_k oscillates near a diagonal matrix

Improved Convergence Result for QR

Use induction, for k to prove A_k normal and

$$A_k = U_k Q_k R_k$$

By definition $R_k Q_k = Q_{k+1} R_{k+1} = P_k$

$$A_{k+1} = U_k Q_k R_k = U_k P_k$$

$$U_k = Q_k^* \dots Q_0^* U Q_0 \dots Q_k$$

P_k converges to a diagonal matrix and U_k is a rotation
In a sense A_k oscillates near a diagonal matrix
No true convergence is guaranteed

QR for Complex Eigenvalues

$$A_0 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

QR for Complex Eigenvalues

$$A_0 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

$$Q_0 = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}, \ R_0 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

QR for Complex Eigenvalues

$$A_0 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

$$Q_0 = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}, \; R_0 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$A_1 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

QR for Complex Eigenvalues

$$A_0 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

$$Q_0 = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}, \; R_0 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$A_1 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

$$Q_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \; R_1 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

QR for Complex Eigenvalues

$$A_0 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

$$Q_0 = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}, \; R_0 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$A_1 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

$$Q_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \; R_1 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

QR for Complex Eigenvalues

$$A_0 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

$$Q_0 = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}, \; R_0 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$A_1 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

$$Q_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \; R_1 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

A_k has eigenvalues $-\mathrm{i}$, i

QR for Complex Eigenvalues

$$A_0 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

$$Q_0 = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}, \; R_0 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$A_1 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

$$Q_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \; R_1 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

A_k has eigenvalues $-\mathrm{i}$, i

Larger matrices get non-convergent 2×2 blocks with conjugate eigenvalues

QR for Complex Eigenvalues

More general

$$A_0 = \begin{bmatrix} a & -b \\ b & a \end{bmatrix}$$

QR for Complex Eigenvalues

More general

$$A_0 = \begin{bmatrix} a & -b \\ b & a \end{bmatrix}$$

$$\lambda_1(A) = \overline{\lambda_2(A)} = a + bi$$

QR for Complex Eigenvalues

More general

$$A_0 = \begin{bmatrix} a & -b \\ b & a \end{bmatrix}$$

$$\lambda_1(A) = \overline{\lambda_2(A)} = a + bi$$

$$A_{2k} = \begin{bmatrix} a & -b \\ b & a \end{bmatrix}, A_{2k+1} = \begin{bmatrix} a & b \\ -b & a \end{bmatrix}$$

QR for Complex Eigenvalues

More general

$$A_0 = \begin{bmatrix} a & -b \\ b & a \end{bmatrix}$$

$$\lambda_1(A) = \overline{\lambda_2(A)} = a + bi$$

$$A_{2k} = \begin{bmatrix} a & -b \\ b & a \end{bmatrix}, A_{2k+1} = \begin{bmatrix} a & b \\ -b & a \end{bmatrix}$$

For a general matrix A with distinct eigenvalues the QR Algorithm converges to a block upper triangular matrix

QR for Complex Eigenvalues

More general

$$A_0 = \begin{bmatrix} a & -b \\ b & a \end{bmatrix}$$

$$\lambda_1(A) = \overline{\lambda_2(A)} = a + bi$$

$$A_{2k} = \begin{bmatrix} a & -b \\ b & a \end{bmatrix}, A_{2k+1} = \begin{bmatrix} a & b \\ -b & a \end{bmatrix}$$

For a general matrix A with distinct eigenvalues the QR Algorithm converges to a block upper triangular matrix
For a general matrix with eigenvalues $a + bi$, $a - bi$ the QR Algorithm goes towards A_{2k}, A_{2k+1}

QR for General Matrices

Let A, B and $\lambda_{\max}(A) < \lambda_{\min}(B)$

QR for General Matrices

Let A, B and $\lambda_{\text{max}}(A) < \lambda_{\text{min}}(B)$

$$\rho(AB^{-1}) \leq \frac{\rho(A)}{\rho(B)} \leq 1$$

QR for General Matrices

Let A, B and $\lambda_{\max}(A) < \lambda_{\min}(B)$

$$\rho(AB^{-1}) \leq \frac{\rho(A)}{\rho(B)} \leq 1$$

For a general diagonalizable A, $\det A \neq 0$ we have from the QR algorithm

$$A^k = U_k R_k, \; A_{k+1} = U_k A U_k^*$$

QR for General Matrices

Let A, B and $\lambda_{\max}(A) < \lambda_{\min}(B)$

$$\rho(AB^{-1}) \leq \frac{\rho(A)}{\rho(B)} \leq 1$$

For a general diagonalizable A, $\det A \neq 0$ we have from the QR algorithm

$$A^k = U_k R_k, \ A_{k+1} = U_k A U_k^*$$

$$A = P \Lambda P^{-1}, \ P^{-1} = LU, \ P = QR$$

QR for General Matrices

Let A, B and $\lambda_{\max}(A) < \lambda_{\min}(B)$

$$\rho(AB^{-1}) \leq \frac{\rho(A)}{\rho(B)} \leq 1$$

For a general diagonalizable A, $\det A \neq 0$ we have from the QR algorithm

$$A^k = U_k R_k,\ A_{k+1} = U_k A U_k^*$$

$$A = P \Lambda P^{-1},\ P^{-1} = LU,\ P = QR$$

With similar argument with the block LU factoriaztion can be used

QR for General Matrices

Let A, B and $\lambda_{\max}(A) < \lambda_{\min}(B)$

$$\rho(AB^{-1}) \leq \frac{\rho(A)}{\rho(B)} \leq 1$$

For a general diagonalizable A, $\det A \neq 0$ we have from the QR algorithm

$$A^k = U_k R_k, \; A_{k+1} = U_k A U_k^*$$

$$A = P \Lambda P^{-1}, \; P^{-1} = LU, \; P = QR$$

With similar argument with the block LU factoriaztion can be used
Convergence is not guaranteed but the eigenvalues of the block upper triangular matrix converge to the real eigenvalues

The Explicitly Shifted QR Algorithm

$$A - \mu I = QR$$

The Explicitly Shifted QR Algorithm

$$A - \mu I = QR$$

$$\hat{A} - \mu I = RQ$$

The Explicitly Shifted QR Algorithm

$$A - \mu I = QR$$

$$\hat{A} - \mu I = RQ$$

$$\begin{bmatrix} Q \\ q^* \end{bmatrix} (A - \mu I) = \begin{bmatrix} R \\ r^* \end{bmatrix}$$

The Explicitly Shifted QR Algorithm

$$A - \mu I = QR$$

$$\hat{A} - \mu I = RQ$$

$$\begin{bmatrix} Q \\ q^* \end{bmatrix} (A - \mu I) = \begin{bmatrix} R \\ r^* \end{bmatrix}$$

$$r = r_{nn} e_n$$

The Explicitly Shifted QR Algorithm

$$A - \mu I = QR$$

$$\hat{A} - \mu I = RQ$$

$$\begin{bmatrix} Q \\ q^* \end{bmatrix} (A - \mu I) = \begin{bmatrix} R \\ r^* \end{bmatrix}$$

$$r = r_{nn} e_n$$

$$q^* (A - \mu I) = r_{nn} e_n^*$$

The Explicitly Shifted QR Algorithm

$$A - \mu I = QR$$

$$\hat{A} - \mu I = RQ$$

$$\begin{bmatrix} Q \\ q^* \end{bmatrix} (A - \mu I) = \begin{bmatrix} R \\ r^* \end{bmatrix}$$

$$r = r_{nn} e_n$$

$$q^* (A - \mu I) = r_{nn} e_n^*$$

$$q^* = r_{nn} e_n^* (A - \mu I)^{-1}$$

The Explicitly Shifted QR Algorithm

$$A - \mu I = QR$$

$$\hat{A} - \mu I = RQ$$

$$\begin{bmatrix} Q \\ q^* \end{bmatrix} (A - \mu I) = \begin{bmatrix} R \\ r^* \end{bmatrix}$$

$$r = r_{nn} e_n$$

$$q^*(A - \mu I) = r_{nn} e_n^*$$

$$q^* = r_{nn} e_n^* (A - \mu I)^{-1}$$

QR with explicit shifts is equivalent to the Inverse Power method

QR Decomposition of a Matrix Polynomial

Apply the QR Algorithm with shifts $\kappa_1, \ldots, \kappa_m$

$$A_0 = A,\ A_k - \kappa_k I = Q_k R_k,\ A_{k+1} - \kappa_{k+1} I = R_k Q_k$$

QR Decomposition of a Matrix Polynomial

Apply the QR Algorithm with shifts $\kappa_1, \ \ldots, \ \kappa_m$

$$A_0 = A, \ A_k - \kappa_k I = Q_k R_k, \ A_{k+1} - \kappa_{k+1} I = R_k Q_k$$

$$A_k - \kappa_k I = Q_k R_k, \ R_k = Q_k^* (A_k - \kappa_k I)$$

QR Decomposition of a Matrix Polynomial

Apply the QR Algorithm with shifts $\kappa_1, \ \ldots, \ \kappa_m$

$$A_0 = A, \ A_k - \kappa_k I = Q_k R_k, \ A_{k+1} - \kappa_{k+1} I = R_k Q_k$$

$$A_k - \kappa_k I = Q_k R_k, \ R_k = Q_k^*(A_k - \kappa_k I)$$

$$A_{k+1} - \kappa_{k+1} = Q_k^*(A_k - \kappa_k I) Q_k$$

QR Decomposition of a Matrix Polynomial

Apply the QR Algorithm with shifts $\kappa_1, \ \ldots, \ \kappa_m$

$$A_0 = A, \ A_k - \kappa_k I = Q_k R_k, \ A_{k+1} - \kappa_{k+1} I = R_k Q_k$$

$$A_k - \kappa_k I = Q_k R_k, \ R_k = Q_k^*(A_k - \kappa_k I)$$

$$A_{k+1} - \kappa_{k+1} = Q_k^*(A_k - \kappa_k I) Q_k$$

$$A_{k+1} - \kappa_{k+1} = (Q_0 \ldots Q_k)^*(A_0 - \kappa_0 I)(Q_0 \ldots Q_k)$$

$$Q_0 \ldots Q_k R_k \ldots R_0 = Q_0 \ldots Q_{k-1}(A_k - \kappa_k I) R_{k-1} \ldots R_0 =$$
$$= (A - \kappa_0 I)(Q_1 \ldots Q_{k-1}) R_{k-1} \ldots R_0 = (A - \kappa_0 I) \ldots (A - \kappa_k I)$$

Error Bounds

$$A = \begin{bmatrix} B & h \\ g^* & \mu \end{bmatrix}$$

Error Bounds

$$A = \begin{bmatrix} B & h \\ g^* & \mu \end{bmatrix}$$

μ is the Rayleigh quotient of e_n

$$\mu = e_n^* A e_n$$

Error Bounds

$$A = \begin{bmatrix} B & h \\ g^* & \mu \end{bmatrix}$$

μ is the Rayleigh quotient of e_n

$$\mu = e_n^* A e_n$$

$$A - \kappa I = \begin{bmatrix} B - \kappa I & h \\ g^* & \mu - \kappa \end{bmatrix} = \begin{bmatrix} P & f \\ e^* & \alpha \end{bmatrix} \begin{bmatrix} R & p \\ 0 & r \end{bmatrix}$$

Error Bounds

$$A = \begin{bmatrix} B & h \\ g^* & \mu \end{bmatrix}$$

μ is the Rayleigh quotient of e_n

$$\mu = e_n^* A e_n$$

$$A - \kappa I = \begin{bmatrix} B - \kappa I & h \\ g^* & \mu - \kappa \end{bmatrix} = \begin{bmatrix} P & f \\ e^* & \alpha \end{bmatrix} \begin{bmatrix} R & p \\ 0 & r \end{bmatrix}$$

$$\hat{A} - \kappa I = \begin{bmatrix} \hat{B} - \kappa I & \hat{h} \\ \hat{g}^* & \hat{\mu} - \kappa \end{bmatrix} = \begin{bmatrix} R & p \\ 0 & r \end{bmatrix} \begin{bmatrix} P & f \\ e^* & \alpha \end{bmatrix}$$

Error Bounds

$$\|f\|_2^2 + \alpha^2 = \|e\|_2^2 + \alpha^2 = 1 \Rightarrow \|f\|_2^2 = \|e\|_2^2$$

Error Bounds

$$\|f\|_2^2 + \alpha^2 = \|e\|_2^2 + \alpha^2 = 1 \Rightarrow \|f\|_2^2 = \|e\|_2^2$$

$$g^* = e^* R \Rightarrow \|e\|_2 \leq \|R^{-1}\|_2 \|g\|_2$$

Error Bounds

$$\|f\|_2^2 + \alpha^2 = \|e\|_2^2 + \alpha^2 = 1 \Rightarrow \|f\|_2^2 = \|e\|_2^2$$

$$g^* = e^* R \Rightarrow \|e\|_2 \leq \|R^{-1}\|_2 \|g\|_2$$

$$\begin{bmatrix} P^* & e \\ f^* & \alpha \end{bmatrix} \begin{bmatrix} B - \kappa I & h \\ g^* & \mu - \kappa \end{bmatrix} = \begin{bmatrix} R & p \\ 0 & r \end{bmatrix}$$

Error Bounds

$$\|f\|_2^2 + \alpha^2 = \|e\|_2^2 + \alpha^2 = 1 \Rightarrow \|f\|_2^2 = \|e\|_2^2$$

$$g^* = e^* R \Rightarrow \|e\|_2 \leq \|R^{-1}\|_2 \|g\|_2$$

$$\begin{bmatrix} P^* & e \\ f^* & \alpha \end{bmatrix} \begin{bmatrix} B - \kappa I & h \\ g^* & \mu - \kappa \end{bmatrix} = \begin{bmatrix} R & p \\ 0 & r \end{bmatrix}$$

$$f^* h + \alpha(\mu - \kappa) = r$$

Error Bounds

$$\|f\|_2^2 + \alpha^2 = \|e\|_2^2 + \alpha^2 = 1 \Rightarrow \|f\|_2^2 = \|e\|_2^2$$

$$g^* = e^* R \Rightarrow \|e\|_2 \leq \|R^{-1}\|_2 \|g\|_2$$

$$\begin{bmatrix} P^* & e \\ f^* & \alpha \end{bmatrix} \begin{bmatrix} B - \kappa I & h \\ g^* & \mu - \kappa \end{bmatrix} = \begin{bmatrix} R & p \\ 0 & r \end{bmatrix}$$

$$f^* h + \alpha(\mu - \kappa) = r$$

$$|r| = |f^* h + \alpha(\mu - \kappa)| \leq |f^* h| + |\alpha||\mu - \kappa| \leq \|f\|_2 \|h\|_2 + |\mu - \kappa|$$

Error Bounds

$$\|f\|_2^2 + \alpha^2 = \|e\|_2^2 + \alpha^2 = 1 \Rightarrow \|f\|_2^2 = \|e\|_2^2$$

$$g^* = e^* R \Rightarrow \|e\|_2 \leq \|R^{-1}\|_2 \|g\|_2$$

$$\begin{bmatrix} P^* & e \\ f^* & \alpha \end{bmatrix} \begin{bmatrix} B - \kappa I & h \\ g^* & \mu - \kappa \end{bmatrix} = \begin{bmatrix} R & p \\ 0 & r \end{bmatrix}$$

$$f^* h + \alpha(\mu - \kappa) = r$$

$$|r| = |f^* h + \alpha(\mu - \kappa)| \leq |f^* h| + |\alpha||\mu - \kappa| \leq \|f\|_2 \|h\|_2 + |\mu - \kappa|$$

$$|r| \leq \|R^{-1}\|_2 \|g\|_2 \|h\|_2 + |\mu - \kappa|$$

Error Bounds

$$\|f\|_2^2 + \alpha^2 = \|e\|_2^2 + \alpha^2 = 1 \Rightarrow \|f\|_2^2 = \|e\|_2^2$$

$$g^* = e^* R \Rightarrow \|e\|_2 \leq \|R^{-1}\|_2 \|g\|_2$$

$$\begin{bmatrix} P^* & e \\ f^* & \alpha \end{bmatrix} \begin{bmatrix} B - \kappa I & h \\ g^* & \mu - \kappa \end{bmatrix} = \begin{bmatrix} R & p \\ 0 & r \end{bmatrix}$$

$$f^* h + \alpha(\mu - \kappa) = r$$

$$|r| = |f^* h + \alpha(\mu - \kappa)| \leq |f^* h| + |\alpha||\mu - \kappa| \leq \|f\|_2 \|h\|_2 + |\mu - \kappa|$$

$$|r| \leq \|R^{-1}\|_2 \|g\|_2 \|h\|_2 + |\mu - \kappa|$$

$$\hat{g} = r g^* \Rightarrow \|\hat{g}\|_2 \leq \|R^{-1}\|_2^2 \|g\|_2^2 \|h\|_2 + \|R^{-1}\|_2 \|g\|_2 |\mu - \kappa|$$

Convergence Rates

$$\|g_{k+1}\|_2 \leq \|R_k^{-1}\|_2^2 \|g_k\|_2^2 \|h_k\|_2 + \|R_k^{-1}\|_2 \|g_k\|_2 |\mu_k - \kappa_k|$$

Convergence Rates

$$\|g_{k+1}\|_2 \leq \|R_k^{-1}\|_2^2 \|g_k\|_2^2 \|h_k\|_2 + \|R_k^{-1}\|_2 \|g_k\|_2 |\mu_k - \kappa_k|$$

Use a Rayleigh quotient shift $\kappa_k = \mu_k$

Convergence Rates

$$\|g_{k+1}\|_2 \leq \|R_k^{-1}\|_2^2 \|g_k\|_2^2 \|h_k\|_2 + \|R_k^{-1}\|_2 \|g_k\|_2 |\mu_k - \kappa_k|$$

Use a Rayleigh quotient shift $\kappa_k = \mu_k$

$$\|g_{k+1}\|_2 \leq \sigma^2 \|g_k\|_2^2$$

For a proof for the bounds of $\|R^{-1}\|_2^2$ and $\|h_k\|_2^2$ see [4]

Convergence Rates

$$\|g_{k+1}\|_2 \leq \|R_k^{-1}\|_2^2 \|g_k\|_2^2 \|h_k\|_2 + \|R_k^{-1}\|_2 \|g_k\|_2 |\mu_k - \kappa_k|$$

Use a Rayleigh quotient shift $\kappa_k = \mu_k$

$$\|g_{k+1}\|_2 \leq \sigma^2 \|g_k\|_2^2$$

For a proof for the bounds of $\|R^{-1}\|_2^2$ and $\|h_k\|_2^2$ see [4]
$\|g_k\|$ converges at least quadratically

Convergence Rates

$$\|g_{k+1}\|_2 \leq \|R_k^{-1}\|_2^2 \|g_k\|_2^2 \|h_k\|_2 + \|R_k^{-1}\|_2 \|g_k\|_2 |\mu_k - \kappa_k|$$

Use a Rayleigh quotient shift $\kappa_k = \mu_k$

$$\|g_{k+1}\|_2 \leq \sigma^2 \|g_k\|_2^2$$

For a proof for the bounds of $\|R^{-1}\|_2^2$ and $\|h_k\|_2^2$ see [4]
$\|g_k\|$ converges at least quadratically
If A is Hermitean $h_k = g_k$

Convergence Rates

$$\|g_{k+1}\|_2 \leq \|R_k^{-1}\|_2^2 \|g_k\|_2^2 \|h_k\|_2 + \|R_k^{-1}\|_2 \|g_k\|_2 |\mu_k - \kappa_k|$$

Use a Rayleigh quotient shift $\kappa_k = \mu_k$

$$\|g_{k+1}\|_2 \leq \sigma^2 \|g_k\|_2^2$$

For a proof for the bounds of $\|R^{-1}\|_2^2$ and $\|h_k\|_2^2$ see [4]
$\|g_k\|$ converges at least quadratically
If A is Hermitean $h_k = g_k$

$$\|g_{k+1}\|_2 \leq \sigma^2 \|g_k\|_2^3$$

Convergence Rates

$$\|g_{k+1}\|_2 \leq \|R_k^{-1}\|_2^2 \|g_k\|_2^2 \|h_k\|_2 + \|R_k^{-1}\|_2 \|g_k\|_2 |\mu_k - \kappa_k|$$

Use a Rayleigh quotient shift $\kappa_k = \mu_k$

$$\|g_{k+1}\|_2 \leq \sigma^2 \|g_k\|_2^2$$

For a proof for the bounds of $\|R^{-1}\|_2^2$ and $\|h_k\|_2^2$ see [4]
$\|g_k\|$ converges at least quadratically
If A is Hermitean $h_k = g_k$

$$\|g_{k+1}\|_2 \leq \sigma^2 \|g_k\|_2^3$$

$\|g_k\|$ converges at least cubic

Practical QR Algorithm

Use Householder transformation to reduce A to Hessenberg form

Practical QR Algorithm

Use Householder transformation to reduce A to Hessenberg form
Use QR with Rayleigh quotient to approximate the last eigenvalue

$$\kappa_i = e_n^* A_i e_n$$

Practical QR Algorithm

Use Householder transformation to reduce A to Hessenberg form
Use QR with Rayleigh quotient to approximate the last eigenvalue

$$\kappa_i = e_n^* A_i e_n$$

When g is close 0, μ is close to an eigenvalue

$$\|g\| \leq \epsilon \|A\|_F$$

Practical QR Algorithm

Use Householder transformation to reduce A to Hessenberg form
Use QR with Rayleigh quotient to approximate the last eigenvalue

$$\kappa_i = e_n^* A_i e_n$$

When g is close 0, μ is close to an eigenvalue

$$\|g\| \leq \epsilon \|A\|_F$$

Use a deflation technique to reduce A to a $(n-1) \times (n-1)$ matrix

Practical QR Algorithm

Use Householder transformation to reduce A to Hessenberg form
Use QR with Rayleigh quotient to approximate the last eigenvalue

$$\kappa_i = e_n^* A_i e_n$$

When g is close 0, μ is close to an eigenvalue

$$\|g\| \leq \epsilon \|A\|_F$$

Use a deflation technique to reduce A to a $(n-1) \times (n-1)$ matrix
If a subdiagonal element is close to 0 deflate

Practical QR Algorithm

Use Householder transformation to reduce A to Hessenberg form
Use QR with Rayleigh quotient to approximate the last eigenvalue

$$\kappa_i = e_n^* A_i e_n$$

When g is close 0, μ is close to an eigenvalue

$$\|g\| \leq \epsilon \|A\|_F$$

Use a deflation technique to reduce A to a $(n-1) \times (n-1)$ matrix
If a subdiagonal element is close to 0 deflate
Apply inverse power method to get eigenvectors from eigenvalues

Wilkinson's Shift

The QR with Rayleigh quotient cannot produce complex eigenvalues

Wilkinson's Shift

The QR with Rayleigh quotient cannot produce complex eigenvalues
Wilkinson's shift introduces complex arithmetic

Wilkinson's Shift

The QR with Rayleigh quotient cannot produce complex eigenvalues
Wilkinson's shift introduces complex arithmetic
Denote by B_k the 2×2 matrix $A(n-1:n, n-1:n)$

Wilkinson's Shift

The QR with Rayleigh quotient cannot produce complex eigenvalues
Wilkinson's shift introduces complex arithmetic
Denote by B_k the 2×2 matrix $A(n-1:n, n-1:n)$
Denote α_k, β_k the eigenvalues of B_k

Wilkinson's Shift

The QR with Rayleigh quotient cannot produce complex eigenvalues
Wilkinson's shift introduces complex arithmetic
Denote by B_k the 2×2 matrix $A(n-1:n, n-1:n)$
Denote α_k, β_k the eigenvalues of B_k
Use as shift the eigenvalue that minimizes

$$\kappa_k = \min\{|\alpha_k e_n^* A|, |\beta_k e_n^* A|\}$$

Wilkinson's Shift

The QR with Rayleigh quotient cannot produce complex eigenvalues
Wilkinson's shift introduces complex arithmetic
Denote by B_k the 2×2 matrix $A(n-1:n, n-1:n)$
Denote α_k, β_k the eigenvalues of B_k
Use as shift the eigenvalue that minimizes

$$\kappa_k = \min\{|\alpha_k e_n^* A|, |\beta_k e_n^* A|\}$$

Iterate and deflate as in the Practical QR Algorithm

Wilkinson's Shift

The QR with Rayleigh quotient cannot produce complex eigenvalues
Wilkinson's shift introduces complex arithmetic
Denote by B_k the 2×2 matrix $A(n-1:n, n-1:n)$
Denote α_k, β_k the eigenvalues of B_k
Use as shift the eigenvalue that minimizes

$$\kappa_k = \min\{|\alpha_k e_n^* A|, |\beta_k e_n^* A|\}$$

Iterate and deflate as in the Practical QR Algorithm For Hermitean matrices there can be a loss of cubic convergence

Implicit Double Shifts

Avoid complex arithmetic

Implicit Double Shifts

Avoid complex arithmetic

If κ is a candidate for a Wilkinson shift, so is $\overline{\kappa}$

Implicit Double Shifts

Avoid complex arithmetic
If κ is a candidate for a Wilkinson shift, so is $\overline{\kappa}$ Start with Hessenberg matrix H

$$(H - \kappa I)(H - \overline{\kappa} I) = H^2 - 2\Re(\kappa)H + |\kappa|^2$$

Implicit Double Shifts

Avoid complex arithmetic
If κ is a candidate for a Wilkinson shift, so is $\overline{\kappa}$ Start with Hessenberg matrix H

$$(H - \kappa I)(H - \overline{\kappa} I) = H^2 - 2\Re(\kappa)H + |\kappa|^2$$

$$(H - \kappa I)(H - \overline{\kappa} I) = QR$$

Implicit Double Shifts

Avoid complex arithmetic
If κ is a candidate for a Wilkinson shift, so is $\overline{\kappa}$ Start with Hessenberg matrix H

$$(H - \kappa I)(H - \overline{\kappa} I) = H^2 - 2\Re(\kappa)H + |\kappa|^2$$

$$(H - \kappa I)(H - \overline{\kappa} I) = QR$$

$$\hat{H} = RQ$$

Implicit Double Shifts

Avoid complex arithmetic
If κ is a candidate for a Wilkinson shift, so is $\overline{\kappa}$ Start with Hessenberg matrix H

$$(H - \kappa I)(H - \overline{\kappa} I) = H^2 - 2\Re(\kappa)H + |\kappa|^2$$

$$(H - \kappa I)(H - \overline{\kappa} I) = QR$$

$$\hat{H} = RQ$$

Making both shifts at once = Francis' shift

Implicit Double Shifts

Avoid complex arithmetic
If κ is a candidate for a Wilkinson shift, so is $\overline{\kappa}$ Start with Hessenberg matrix H

$$(H - \kappa I)(H - \overline{\kappa} I) = H^2 - 2\Re(\kappa)H + |\kappa|^2$$

$$(H - \kappa I)(H - \overline{\kappa} I) = QR$$

$$\hat{H} = RQ$$

Making both shifts at once = Francis' shift
$H^2 - 2\Re(\kappa)H + |\kappa|^2$ is hard to compute $\mathcal{O}(n^3)$

Implicit Q Theorem

The matrices $Q, \ H$ from $A = QHQ^*$ is determined, up to the modulus, by q_1

$$AQ = QH$$

Implicit Q Theorem

The matrices $Q,\ H$ from $A = QHQ^*$ is determined, up to the modulus, by q_1

$$AQ = QH$$

$$Aq_1 = q_1 h_{11} + q_2 h_{21} \Rightarrow h_{11} = q_1^* A q_1$$

Implicit Q Theorem

The matrices Q, H from $A = QHQ^*$ is determined, up to the modulus, by q_1

$$AQ = QH$$

$$Aq_1 = q_1 h_{11} + q_2 h_{21} \Rightarrow h_{11} = q_1^* A q_1$$

$$q_2 h_{21} = Aq_1 - h_{11} q_1 \Rightarrow h_{21} = \|Aq_1 - h_{11} q_1\|_2$$

Implicit Q Theorem

The matrices $Q, \ H$ from $A = QHQ^*$ is determined, up to the modulus, by q_1

$$AQ = QH$$

$$Aq_1 = q_1 h_{11} + q_2 h_{21} \Rightarrow h_{11} = q_1^* A q_1$$

$$q_2 h_{21} = Aq_1 - h_{11} q_1 \Rightarrow h_{21} = \|Aq_1 - h_{11} q_1\|_2$$

Assume $q_k, \ \ldots q_k$ have been determined

Implicit Q Theorem

The matrices $Q, \ H$ from $A = QHQ^*$ is determined, up to the modulus, by q_1

$$AQ = QH$$

$$Aq_1 = q_1 h_{11} + q_2 h_{21} \Rightarrow h_{11} = q_1^* A q_1$$

$$q_2 h_{21} = Aq_1 - h_{11} q_1 \Rightarrow h_{21} = \|Aq_1 - h_{11} q_1\|_2$$

Assume $q_k, \ \ldots q_k$ have been determined

$$Aq_k = h_{1k} q_1 + h_{2k} q_2 + \ldots h_{kk} q_k + h_{(k+1)k} q_{k+1} \Rightarrow h_{ik} = q_i^* A q_i$$

Implicit Q Theorem

The matrices $Q, \; H$ from $A = QHQ^*$ is determined, up to the modulus, by q_1

$$AQ = QH$$

$$Aq_1 = q_1 h_{11} + q_2 h_{21} \Rightarrow h_{11} = q_1^* A q_1$$

$$q_2 h_{21} = Aq_1 - h_{11} q_1 \Rightarrow h_{21} = \|Aq_1 - h_{11} q_1\|_2$$

Assume $q_k, \; \ldots q_k$ have been determined

$$Aq_k = h_{1k} q_1 + h_{2k} q_2 + \ldots h_{kk} q_k + h_{(k+1)k} q_{k+1} \Rightarrow h_{ik} = q_i^* A q_i$$

$$q_{k+1} = \|Aq_k - (h_{1k} q_1 + h_{2k} q_2 + \ldots h_{kk} q_k)\|_2$$

Francis' Double Shift

Denote by c the first column of $H^2 - 2\Re(\kappa)H + |\kappa|^2$

Francis' Double Shift

Denote by c the first column of $H^2 - 2\Re(\kappa)H + |\kappa|^2$

Use Householder transformation $Q_0 c = \|c\|_2 e_1$

Francis' Double Shift

Denote by c the first column of $H^2 - 2\Re(\kappa)H + |\kappa|^2$
Use Householder transformation $Q_0 c = \|c\|_2 e_1$

$$H_1 = Q_0^* H Q_0$$

Francis' Double Shift

Denote by c the first column of $H^2 - 2\Re(\kappa)H + |\kappa|^2$
Use Householder transformation $Q_0 c = \|c\|_2 e_1$

$$H_1 = Q_0^* H Q_0$$

Reduce H_1 to Hessenberg form $\hat{H}$

$$\hat{H} = Q_1^* H_1 Q_1$$

Francis' Double Shift

Denote by c the first column of $H^2 - 2\Re(\kappa)H + |\kappa|^2$
Use Householder transformation $Q_0 c = \|c\|_2 e_1$

$$H_1 = Q_0^* H Q_0$$

Reduce H_1 to Hessenberg form $\hat{H}$

$$\hat{H} = Q_1^* H_1 Q_1$$

$$\hat{Q} = Q_0 Q_1$$

$\hat{Q}$ has to be the Hessenberg reduction of H to $\hat{H}$

Francis' Double Shift

Denote by c the first column of $H^2 - 2\Re(\kappa)H + |\kappa|^2$
Use Householder transformation $Q_0 c = \|c\|_2 e_1$

$$H_1 = Q_0^* H Q_0$$

Reduce H_1 to Hessenberg form $\hat{H}$

$$\hat{H} = Q_1^* H_1 Q_1$$

$$\hat{Q} = Q_0 Q_1$$

$\hat{Q}$ has to be the Hessenberg reduction of H to $\hat{H}$

$$\hat{H} = Q^* H Q$$

Francis' Double Shift

Denote by c the first column of $H^2 - 2\Re(\kappa)H + |\kappa|^2$
Use Householder transformation $Q_0 c = \|c\|_2 e_1$

$$H_1 = Q_0^* H Q_0$$

Reduce H_1 to Hessenberg form $\hat{H}$

$$\hat{H} = Q_1^* H_1 Q_1$$

$$\hat{Q} = Q_0 Q_1$$

$\hat{Q}$ has to be the Hessenberg reduction of H to $\hat{H}$

$$\hat{H} = Q^* H Q$$

Iterate this algorithm until close to Real Schur form

Final Notes on the QR

The eigenvalues usually appear ordered in the Schur form

Final Notes on the QR

The eigenvalues usually appear ordered in the Schur form
When there is disorder in the eigenvalues the algorithm may not converge

Final Notes on the QR

The eigenvalues usually appear ordered in the Schur form
When there is disorder in the eigenvalues the algorithm may not converge
The QR Algorithm usually shows slow initial convergence until the quadratic convergence gets into action

Final Notes on the QR

The eigenvalues usually appear ordered in the Schur form
When there is disorder in the eigenvalues the algorithm may not converge
The QR Algorithm usually shows slow initial convergence until the quadratic convergence gets into action
For large matrices in Hessenberg form elements from the top may get small and require deflation

Final Notes on the QR

The eigenvalues usually appear ordered in the Schur form

When there is disorder in the eigenvalues the algorithm may not converge

The QR Algorithm usually shows slow initial convergence until the quadratic convergence gets into action

For large matrices in Hessenberg form elements from the top may get small and require deflation

There is an algorithm to reduce a complex Hessenberg matrix to a real one

Arnoldi Iteration

Compute Hessenberg form of A

Arnoldi Iteration

Compute Hessenberg form of A
Householder method cannot give partial results

Arnoldi Iteration

Compute Hessenberg form of A
Householder method cannot give partial results

$$AQ = QH \Rightarrow AQ_n = Q_{n+1}\hat{H}$$

Arnoldi Iteration

Compute Hessenberg form of A
Householder method cannot give partial results

$$AQ = QH \Rightarrow AQ_n = Q_{n+1}\hat{H}$$

Where Q_k is the matrix with the firs k columns of Q and

$$\hat{H} = \begin{bmatrix} h_{11} & h_{12} & \dots & h_{1n} \\ h_{21} & h_{22} & \dots & h_{2n} \\ 0 & h_{32} & \dots & h_{3n} \\ \dots & \dots & \dots & \dots \\ 0 & \dots & h_{n+1n-1} & h_{n+1n} \end{bmatrix}$$

Compute Hessenberg form of A
Householder method cannot give partial results

$$AQ = QH \Rightarrow AQ_n = Q_{n+1}\hat{H}$$

Where Q_k is the matrix with the firs k columns of Q and

$$\hat{H} = \begin{bmatrix} h_{11} & h_{12} & \dots & h_{1n} \\ h_{21} & h_{22} & \dots & h_{2n} \\ 0 & h_{32} & \dots & h_{3n} \\ \hdotsfor{4} \\ 0 & \dots & h_{n+1n-1} & h_{n+1n} \end{bmatrix}$$

$$Aq_n = h_{1n}q_1 + h_{2n}q_2 + \ldots + h_{nn}q_n + h_{n+1}q_{n+1}$$

Compute Hessenberg form of A
Householder method cannot give partial results

$$AQ = QH \Rightarrow AQ_n = Q_{n+1}\hat{H}$$

Where Q_k is the matrix with the firs k columns of Q and

$$\hat{H} = \begin{bmatrix} h_{11} & h_{12} & \dots & h_{1n} \\ h_{21} & h_{22} & \dots & h_{2n} \\ 0 & h_{32} & \dots & h_{3n} \\ \dots & \dots & \dots & \dots \\ 0 & \dots & h_{n+1n-1} & h_{n+1n} \end{bmatrix}$$

$$Aq_n = h_{1n}q_1 + h_{2n}q_2 + \ldots + h_{nn}q_n + h_{n+1}q_{n+1}$$

Similar construction with the implicit Q theorem

Arnoldi Iteration

$$K_k(A, b) = \operatorname{span}\left(b,\ Ab,\ A^2 b,\ \ldots,\ A^{k-1} b\right)$$

Arnoldi Iteration

$$K_k(A,b) = \operatorname{span}\left(b,\ Ab,\ A^2 b,\ \ldots,\ A^{k-1} b\right)$$

$$q_1 = \frac{b}{\|b\|_2} \Rightarrow K_n(A,b) = \operatorname{span}\left(q_1,\ \ldots,\ q_n\right)$$

Arnoldi Iteration

$$K_k(A, b) = \text{span}\,(b,\ Ab,\ A^2 b,\ \ldots,\ A^{k-1} b)$$

$$q_1 = \frac{b}{\|b\|_2} \Rightarrow K_n(A, b) = \text{span}\,(q_1,\ \ldots,\ q_n)$$

Arnoldi process = construct an orthogonal base of $K_n(A, b)$

Arnoldi Iteration

$$K_k(A, b) = \operatorname{span}\left(b,\ Ab,\ A^2 b,\ \ldots,\ A^{k-1} b\right)$$

$$q_1 = \frac{b}{\|b\|_2} \Rightarrow K_n(A, b) = \operatorname{span}\left(q_1,\ \ldots,\ q_n\right)$$

Arnoldi process = construct an orthogonal base of $K_n(A, b)$ Define K_n the matrix of the first n Krylov vectors

Arnoldi Iteration

$$K_k(A, b) = \operatorname{span}\left(b,\ Ab,\ A^2 b,\ \ldots,\ A^{k-1} b\right)$$

$$q_1 = \frac{b}{\|b\|_2} \Rightarrow K_n(A, b) = \operatorname{span}\left(q_1,\ \ldots,\ q_n\right)$$

Arnoldi process = construct an orthogonal base of $K_n(A, b)$ Define K_n the matrix of the first n Krylov vectors

$$K_n = \bar{Q}\bar{R} \Rightarrow Q_n = \bar{Q}$$

Arnoldi Iteration

$$K_k(A, b) = \text{span}\,(b,\ Ab,\ A^2 b,\ \ldots,\ A^{k-1} b)$$

$$q_1 = \frac{b}{\|b\|_2} \Rightarrow K_n(A, b) = \text{span}\,(q_1,\ \ldots,\ q_n)$$

Arnoldi process = construct an orthogonal base of $K_n(A, b)$ Define K_n the matrix of the first n Krylov vectors

$$K_n = \bar{Q}\bar{R} \Rightarrow Q_n = \bar{Q}$$

From the power method $\Rightarrow K_n$ encodes information about the dominant eigenvalue

Arnoldi Iteration

$$K_k(A, b) = \mathrm{span}\left(b,\ Ab,\ A^2 b,\ \ldots,\ A^{k-1} b\right)$$

$$q_1 = \frac{b}{\|b\|_2} \Rightarrow K_n(A, b) = \mathrm{span}\left(q_1,\ \ldots,\ q_n\right)$$

Arnoldi process = construct an orthogonal base of $K_n(A, b)$ Define K_n the matrix of the first n Krylov vectors

$$K_n = \bar{Q}\bar{R} \Rightarrow Q_n = \bar{Q}$$

From the power method $\Rightarrow K_n$ encodes information about the dominant eigenvalue
Arnoldi iteration isolates the information about the dominating eigenvalue via QR

Krylov Subspaces Projection

Define $P : K_n(A, b) \to K_n(A, b)$

$$x \to P_{K_n} A x$$

Krylov Subspaces Projection

Define $P : K_n(A, b) \to K_n(A, b)$

$$x \to P_{K_n} Ax$$

Q_n is an orthonormal base

$$P = Q_n Q_n^* A$$

Krylov Subspaces Projection

Define $P : K_n(A, b) \to K_n(A, b)$

$$x \to P_{K_n} Ax$$

Q_n is an orthonormal base

$$P = Q_n Q_n^* A$$

Change the basis of P

$$P_{Q_n} = Q_n^* A Q_n$$

The Rayleigh Ritz quotient of the vectors q_i

Krylov Subspaces Projection

Define $P : K_n(A, b) \to K_n(A, b)$

$$x \to P_{K_n} Ax$$

Q_n is an orthonormal base

$$P = Q_n Q_n^* A$$

Change the basis of P

$$P_{Q_n} = Q_n^* A Q_n$$

The Rayleigh Ritz quotient of the vectors q_i Main ideas of Arnoldi iteration

$$K_n = Q_n R_n, \quad H_n = Q_n^* A Q_n, \quad AQ_n = Q_{n+1} \overline{H_n}$$

Arnoldi Eigenvalues Approximations

The eigenvalues of $\overline{H_n}$ ar called the Arnoldi estimates

Arnoldi Eigenvalues Approximations

The eigenvalues of $\overline{H_n}$ ar called the Arnoldi estimates

$$x \in K_n(A, b) \Rightarrow x = \sum_{k=0}^{n-1} c_k A^k b = p(A)b$$

Arnoldi Eigenvalues Approximations

The eigenvalues of $\overline{H_n}$ ar called the Arnoldi estimates

$$x \in K_n(A, b) \Rightarrow x = \sum_{k=0}^{n-1} c_k A^k b = p(A)b$$

Imposing $c_n = 1$, find p_n that minimizes $\|p_n(A)b\|_2$

Arnoldi Eigenvalues Approximations

The eigenvalues of $\overline{H_n}$ ar called the Arnoldi estimates

$$x \in K_n(A, b) \Rightarrow x = \sum_{k=0}^{n-1} c_k A^k b = p(A)b$$

Imposing $c_n = 1$, find p_n that minimizes $\|p_n(A)b\|_2$

$$p_n(A)b = A^n b - Q_n y$$

Arnoldi Eigenvalues Approximations

The eigenvalues of $\overline{H_n}$ ar called the Arnoldi estimates

$$x \in K_n(A, b) \Rightarrow x = \sum_{k=0}^{n-1} c_k A^k b = p(A)b$$

Imposing $c_n = 1$, find p_n that minimizes $\|p_n(A)b\|_2$

$$p_n(A)b = A^n b - Q_n y$$

We have a linear least square problem, so we have orthogonality

Arnoldi Eigenvalues Approximations

The eigenvalues of $\overline{H_n}$ ar called the Arnoldi estimates

$$x \in K_n(A,b) \Rightarrow x = \sum_{k=0}^{n-1} c_k A^k b = p(A)b$$

Imposing $c_n = 1$, find p_n that minimizes $\|p_n(A)b\|_2$

$$p_n(A)b = A^n b - Q_n y$$

We have a linear least square problem, so we have orthogonality

$$p_n(A)b \perp K_n A, b \Rightarrow Q_n^* p_n(A)b = 0$$

Arnoldi Eigenvalues Approximations

The eigenvalues of $\overline{H_n}$ ar called the Arnoldi estimates

$$x \in K_n(A, b) \Rightarrow x = \sum_{k=0}^{n-1} c_k A^k b = p(A)b$$

Imposing $c_n = 1$, find p_n that minimizes $\|p_n(A)b\|_2$

$$p_n(A)b = A^n b - Q_n y$$

We have a linear least square problem, so we have orthogonality

$$p_n(A)b \perp K_n A, b \Rightarrow Q_n^* p_n(A)b = 0$$

$$A = QHQ^* \Rightarrow Q_n Q p(H) Q^* b = 0$$

Arnoldi Eigenvalues Approximations

$Q_n Q p(H) Q^* b = 0 \Rightarrow$ the first n elements of the first column of $p_n H$ are 0

Arnoldi Eigenvalues Approximations

$Q_n Q p(H) Q^* b = 0 \Rightarrow$ the first n elements of the first column of $p_n H$ are 0

$$p_n(H_n) = 0$$

Arnoldi Eigenvalues Approximations

$Q_n Q p(H) Q^* b = 0 \Rightarrow$ the first n elements of the first column of $p_n H$ are 0

$$p_n(H_n) = 0$$

From Cayley-Hamilton theorem p_n is the caracteristic polynomial of $\overline{H_n}$

Arnoldi Eigenvalues Approximations

$Q_n Q p(H) Q^* b = 0 \Rightarrow$ the first n elements of the first column of $p_n H$ are 0

$$p_n(H_n) = 0$$

From Cayley-Hamilton theorem p_n is the caracteristic polynomial of $\overline{H_n}$

$$p_n(q_i^* A q_i) = 0, \quad i = \overline{1, n}$$

Arnoldi Eigenvalues Approximations

$Q_n Q p(H) Q^* b = 0 \Rightarrow$ the first n elements of the first column of $p_n H$ are 0

$$p_n(H_n) = 0$$

From Cayley-Hamilton theorem p_n is the caracteristic polynomial of $\overline{H_n}$

$$p_n(q_i^* A q_i) = 0, \quad i = \overline{1, n}$$

$q_i^* A q_i$ are the eigenvalues of H_n

Arnoldi Eigenvalues Approximations

$Q_n Q p(H) Q^* b = 0 \Rightarrow$ the first n elements of the first column of $p_n H$ are 0

$$p_n(H_n) = 0$$

From Cayley-Hamilton theorem p_n is the caracteristic polynomial of $\overline{H_n}$

$$p_n(q_i^* A q_i) = 0, \quad i = \overline{1, n}$$

$q_i^* A q_i$ are the eigenvalues of H_n
As n increases H_n is closer to H so the eigenvalues of H_n are close to the eigenvalues of H

Arnoldi Eigenvalues Approximations

$Q_n Q p(H) Q^* b = 0 \Rightarrow$ the first n elements of the first column of $p_n H$ are 0

$$p_n(H_n) = 0$$

From Cayley-Hamilton theorem p_n is the caracteristic polynomial of $\overline{H_n}$

$$p_n(q_i^* A q_i) = 0, \quad i = \overline{1, n}$$

$q_i^* A q_i$ are the eigenvalues of H_n
As n increases H_n is closer to H so the eigenvalues of H_n are close to the eigenvalues of H
$A = QHQ^*$ so the eigenvalues of A and H are the same

Arnoldi Eigenvalues Approximations

$Q_n Qp(H)Q^* b = 0 \Rightarrow$ the first n elements of the first column of $p_n H$ are 0

$$p_n(H_n) = 0$$

From Cayley-Hamilton theorem p_n is the caracteristic polynomial of $\overline{H_n}$

$$p_n(q_i^* A q_i) = 0, \quad i = \overline{1, n}$$

$q_i^* A q_i$ are the eigenvalues of H_n
As n increases H_n is closer to H so the eigenvalues of H_n are close to the eigenvalues of H
$A = QHQ^*$ so the eigenvalues of A and H are the same
Arnoldi iteration may give geometric convergence

Arnoldi Eigenvalues Approximations

$Q_n Qp(H)Q^* b = 0 \Rightarrow$ the first n elements of the first column of $p_n H$ are 0

$$p_n(H_n) = 0$$

From Cayley-Hamilton theorem p_n is the caracteristic polynomial of $\overline{H_n}$

$$p_n(q_i^* A q_i) = 0, \quad i = \overline{1, n}$$

$q_i^* A q_i$ are the eigenvalues of H_n
As n increases H_n is closer to H so the eigenvalues of H_n are close to the eigenvalues of H
$A = QHQ^*$ so the eigenvalues of A and H are the same
Arnoldi iteration may give geometric convergence
The algorithm breaks down when K_n is not of full rank

Arnoldi Eigenvalues Approximations

$Q_n Qp(H)Q^* b = 0 \Rightarrow$ the first n elements of the first column of $p_n H$ are 0

$$p_n(H_n) = 0$$

From Cayley-Hamilton theorem p_n is the caracteristic polynomial of $\overline{H_n}$

$$p_n(q_i^* A q_i) = 0, \quad i = \overline{1, n}$$

$q_i^* A q_i$ are the eigenvalues of H_n
As n increases H_n is closer to H so the eigenvalues of H_n are close to the eigenvalues of H
$A = QHQ^*$ so the eigenvalues of A and H are the same
Arnoldi iteration may give geometric convergence
The algorithm breaks down when K_n is not of full rank
Arnoldi iteration is not fully understood (Trefethen)

Lanczos Iterations

Lanczos iteration = Arnoldi for Hermitean matrices

Lanczos Iterations

Lanczos iteration = Arnoldi for Hermitean matrices
A simplification of computation is easily achieved

Lanczos Iterations

Lanczos iteration = Arnoldi for Hermitean matrices
A simplification of computation is easily achieved

$$h_{ij} = q_j^* A q_i, \ \overline{h_{ji}} = q_i^* A^* q_j$$

Lanczos Iterations

Lanczos iteration = Arnoldi for Hermitean matrices
A simplification of computation is easily achieved

$$h_{ij} = q_j^* A q_i,\ \overline{h_{ji}} = q_i^* A^* q_j$$

$$Aq_n = h_{n-1n} q_{n-1} + h_{nn} q_n + h_{n+1n} q_{n+1}$$

Lanczos Iterations

Lanczos iteration = Arnoldi for Hermitean matrices
A simplification of computation is easily achieved

$$h_{ij} = q_j^* A q_i, \; \overline{h_{ji}} = q_i^* A^* q_j$$

$$Aq_n = h_{n-1n} q_{n-1} + h_{nn} q_n + h_{n+1n} q_{n+1}$$

Both iterations tend to find first eigenvalues closer to the extreme of the spectrum

Lanczos Iterations

Lanczos iteration = Arnoldi for Hermitean matrices
A simplification of computation is easily achieved

$$h_{ij} = q_j^* A q_i, \ \overline{h_{ji}} = q_i^* A^* q_j$$

$$Aq_n = h_{n-1n} q_{n-1} + h_{nn} q_n + h_{n+1n} q_{n+1}$$

Both iterations tend to find first eigenvalues closer to the extreme of the spectrum
Lanczos gets the vectors q_k orthogonal by construction

Lanczos Iterations

Lanczos iteration = Arnoldi for Hermitean matrices
A simplification of computation is easily achieved

$$h_{ij} = q_j^* A q_i, \ \overline{h_{ji}} = q_i^* A^* q_j$$

$$Aq_n = h_{n-1n} q_{n-1} + h_{nn} q_n + h_{n+1n} q_{n+1}$$

Both iterations tend to find first eigenvalues closer to the extreme of the spectrum
Lanczos gets the vectors q_k orthogonal by construction
On finite precision there may be a loss of orthogonality

Lanczos Iterations

Lanczos iteration = Arnoldi for Hermitean matrices
A simplification of computation is easily achieved

$$h_{ij} = q_j^* A q_i, \ \overline{h_{ji}} = q_i^* A^* q_j$$

$$Aq_n = h_{n-1n} q_{n-1} + h_{nn} q_n + h_{n+1n} q_{n+1}$$

Both iterations tend to find first eigenvalues closer to the extreme of the spectrum
Lanczos gets the vectors q_k orthogonal by construction
On finite precision there may be a loss of orthogonality
The loss of orthogonality may introduce ghost eigenvalues

Sylvester's Law of Inertia

Define the congruence relation

$$B = SAS^*$$

Sylvester's Law of Inertia

Define the congruence relation

$$B = SAS^*$$

Assume A is symmetric and real with rank r

Sylvester's Law of Inertia

Define the congruence relation

$$B = SAS^*$$

Assume A is symmetric and real with rank r
A is orthogonal diagonalizable so $A \sim D$

Sylvester's Law of Inertia

Define the congruence relation

$$B = SAS^*$$

Assume A is symmetric and real with rank r
A is orthogonal diagonalizable so $A \sim D$
Reorder D so first p elements are positive, next q negative and the remaining 0

Sylvester's Law of Inertia

Define the congruence relation

$$B = SAS^*$$

Assume A is symmetric and real with rank r
A is orthogonal diagonalizable so $A \sim D$
Reorder D so first p elements are positive, next q negative and the remaining 0
There is a subspace P of dimension p on which A is positive definite

Sylvester's Law of Inertia

Define the congruence relation

$$B = SAS^*$$

Assume A is symmetric and real with rank r
A is orthogonal diagonalizable so $A \sim D$
Reorder D so first p elements are positive, next q negative and the remaining 0
There is a subspace P of dimension p on which A is positive definite
Let $\hat{P}$ be another subspace on witch A is positive definite

Sylvester's Law of Inertia

Define the congruence relation

$$B = SAS^*$$

Assume A is symmetric and real with rank r

A is orthogonal diagonalizable so $A \sim D$

Reorder D so first p elements are positive, next q negative and the remaining 0

There is a subspace P of dimension p on which A is positive definite

Let $\hat{P}$ be another subspace on witch A is positive definite

A is negative on $N \Rightarrow \hat{P} \cap N = 0$

Sylvester's Law of Inertia

Define the congruence relation

$$B = SAS^*$$

Assume A is symmetric and real with rank r
A is orthogonal diagonalizable so $A \sim D$
Reorder D so first p elements are positive, next q negative and the remaining 0
There is a subspace P of dimension p on which A is positive definite
Let $\hat{P}$ be another subspace on witch A is positive definite
A is negative on $N \Rightarrow \hat{P} \cap N = 0$

$$\dim \hat{P} + \dim N \leq r \Rightarrow \dim \hat{P} \leq \dim P$$

Sylvester's Law of Inertia

Define the congruence relation

$$B = SAS^*$$

Assume A is symmetric and real with rank r
A is orthogonal diagonalizable so $A \sim D$
Reorder D so first p elements are positive, next q negative and the remaining 0
There is a subspace P of dimension p on which A is positive definite
Let $\hat{P}$ be another subspace on witch A is positive definite
A is negative on $N \Rightarrow \hat{P} \cap N = 0$

$$\dim \hat{P} + \dim N \leq r \Rightarrow \dim \hat{P} \leq \dim P$$

In any diagonal representation of A the number of positive eigenvalues is the same

Sylvester's Law Inertia

An eigenvalue algorithm based on Sylvester's Inertia

Sylvester's Law Inertia

An eigenvalue algorithm based on Sylvester's Inertia
By multiplication $B = UDU^*$ we see that A and B have the same number of positive eigenvalues

Sylvester's Law Inertia

An eigenvalue algorithm based on Sylvester's Inertia
By multiplication $B = UDU^*$ we see that A and B have the same number of positive eigenvalues
A is symmetric

$$A = LDL^*$$

Sylvester's Law Inertia

An eigenvalue algorithm based on Sylvester's Inertia
By multiplication $B = UDU^*$ we see that A and B have the same number of positive eigenvalues
A is symmetric

$$A = LDL^*$$

Count the number p of positive elements in D

Sylvester's Law Inertia

An eigenvalue algorithm based on Sylvester's Inertia
By multiplication $B = UDU^*$ we see that A and B have the same number of positive eigenvalues
A is symmetric

$$A = LDL^*$$

Count the number p of positive elements in D
Find a κ for which $A - \kappa I = LDL^*$ has $p - 1$ positive eigenvalues

Sylvester's Law Inertia

An eigenvalue algorithm based on Sylvester's Inertia
By multiplication $B = UDU^*$ we see that A and B have the same number of positive eigenvalues
A is symmetric

$$A = LDL^*$$

Count the number p of positive elements in D
Find a κ for which $A - \kappa I = LDL^*$ has $p - 1$ positive eigenvalues
Use Gershgorin Circle Theorem for an initial guess of κ

Sylvester's Law Inertia

An eigenvalue algorithm based on Sylvester's Inertia
By multiplication $B = UDU^*$ we see that A and B have the same number of positive eigenvalues
A is symmetric

$$A = LDL^*$$

Count the number p of positive elements in D
Find a κ for which $A - \kappa I = LDL^*$ has $p - 1$ positive eigenvalues
Use Gershgorin Circle Theorem for an initial guess of κ
Use bisection to find the eigenvalue in $(0, \kappa)$

Sylvester's Equation

$$AX + XB = C$$

Sylvester's Equation

$$AX + XB = C$$

Define $T : M^n \to M^n$

$$T(X) = AX + XB$$

Sylvester's Equation

$$AX + XB = C$$

Define $T : M^n \to M^n$

$$T(X) = AX + XB$$

T is a linear operator

Sylvester's Equation

$$AX + XB = C$$

Define $T : M^n \to M^n$

$$T(X) = AX + XB$$

T is a linear operator The problem can be rewritten as

$$Mv = y$$

Sylvester's Equation

$$AX + XB = C$$

Define $T : M^n \to M^n$

$$T(X) = AX + XB$$

T is a linear operator The problem can be rewritten as

$$Mv = y$$

Simple observation: let us identify $v = \operatorname{vect} X$

Sylvester's Equation

$$AX + XB = C$$

Define $T : M^n \to M^n$

$$T(X) = AX + XB$$

T is a linear operator The problem can be rewritten as

$$Mv = y$$

Simple observation: let us identify $v = \operatorname{vect} X$

$$\operatorname{vect}(AX) = (I \otimes A)v, \ \operatorname{vect}(XB) = (B^T \otimes I)v$$

Sylvester's Equation

$$AX + XB = C$$

Define $T : M^n \to M^n$

$$T(X) = AX + XB$$

T is a linear operator The problem can be rewritten as

$$Mv = y$$

Simple observation: let us identify $v = \operatorname{vect} X$

$$\operatorname{vect}(AX) = (I \otimes A)v, \; \operatorname{vect}(XB) = (B^T \otimes I)v$$

$$M = (I \otimes A) + (B^T \otimes I)$$

Generalized Eigenvalues Problem

Let A, B be hermitean, positive definite

$$AQ = Q\Lambda_A, \;\; BU = U\Lambda_B$$

Generalized Eigenvalues Problem

Let A, B be hermitean, positive definite

$$AQ = Q\Lambda_A, \; BU = U\Lambda_B$$

$$Ax = \lambda Bx$$

Generalized Eigenvalues Problem

Let A, B be hermitean, positive definite

$$AQ = Q\Lambda_A, \; BU = U\Lambda_B$$

$$Ax = \lambda Bx$$

In matrix form $AX = BX\Lambda$

Generalized Eigenvalues Problem

Let A, B be hermitean, positive definite

$$AQ = Q\Lambda_A, \; BU = U\Lambda_B$$

$$Ax = \lambda Bx$$

In matrix form $AX = BX\Lambda$

$$\Lambda_B^{-\frac{1}{2}} U^* B U \Lambda_B^{-\frac{1}{2}} = I$$

Generalized Eigenvalues Problem

Let A, B be hermitean, positive definite

$$AQ = Q\Lambda_A,\ \ BU = U\Lambda_B$$

$$Ax = \lambda Bx$$

In matrix form $AX = BX\Lambda$

$$\Lambda_B^{-\frac{1}{2}} U^* B U \Lambda_B^{-\frac{1}{2}} = I$$

$\Lambda_B^{-\frac{1}{2}} U^* A U \Lambda_B^{-\frac{1}{2}}$ is symmetric

$$\Lambda_B^{-\frac{1}{2}} U^* A U \Lambda_B^{-\frac{1}{2}} = P^* \Lambda_{A'} P$$

Generalized Eigenvalues Problem

$$U\Lambda_B^{-\frac{1}{2}}P = M$$

Generalized Eigenvalues Problem

$$U\Lambda_B^{-\frac{1}{2}}P = M$$

$$M^*AM = P^*\Lambda_B^{-\frac{1}{2}}U^*AU\Lambda_B^{-\frac{1}{2}}P = \Lambda_{A'}$$

Generalized Eigenvalues Problem

$$U\Lambda_B^{-\frac{1}{2}}P = M$$

$$M^*AM = P^*\Lambda_B^{-\frac{1}{2}}U^*AU\Lambda_B^{-\frac{1}{2}}P = \Lambda_{A'}$$

$$M^*BM = P^*\Lambda_B^{-\frac{1}{2}}U^*BU\Lambda_B^{-\frac{1}{2}}P = P^*\Lambda_B^{-\frac{1}{2}}\Lambda_B\Lambda_B^{-\frac{1}{2}}P = P^*P = I$$

Generalized Eigenvalues Problem

$$U\Lambda_B^{-\frac{1}{2}}P = M$$

$$M^*AM = P^*\Lambda_B^{-\frac{1}{2}}U^*AU\Lambda_B^{-\frac{1}{2}}P = \Lambda_{A'}$$

$$M^*BM = P^*\Lambda_B^{-\frac{1}{2}}U^*BU\Lambda_B^{-\frac{1}{2}}P = P^*\Lambda_B^{-\frac{1}{2}}\Lambda_B\Lambda_B^{-\frac{1}{2}}P = P^*P = I$$

$$\begin{cases} M^*AM = \Lambda_{A'} \\ M^*BM = I \end{cases} \Rightarrow \begin{cases} M^*AM = \Lambda_{A'} \\ \Lambda_{A'}M^*BM = \Lambda_{A'} \end{cases}$$

Generalized Eigenvalues Problem

$$U\Lambda_B^{-\frac{1}{2}}P = M$$

$$M^*AM = P^*\Lambda_B^{-\frac{1}{2}}U^*AU\Lambda_B^{-\frac{1}{2}}P = \Lambda_{A'}$$

$$M^*BM = P^*\Lambda_B^{-\frac{1}{2}}U^*BU\Lambda_B^{-\frac{1}{2}}P = P^*\Lambda_B^{-\frac{1}{2}}\Lambda_B\Lambda_B^{-\frac{1}{2}}P = P^*P = I$$

$$\begin{cases} M^*AM = \Lambda_{A'} \\ M^*BM = I \end{cases} \Rightarrow \begin{cases} M^*AM = \Lambda_{A'} \\ \Lambda_{A'}M^*BM = \Lambda_{A'} \end{cases}$$

$$AX = BX\Lambda$$

Generalized Eigenvalues Problem

$$U\Lambda_B^{-\frac{1}{2}}P = M$$

$$M^*AM = P^*\Lambda_B^{-\frac{1}{2}}U^*AU\Lambda_B^{-\frac{1}{2}}P = \Lambda_{A'}$$

$$M^*BM = P^*\Lambda_B^{-\frac{1}{2}}U^*BU\Lambda_B^{-\frac{1}{2}}P = P^*\Lambda_B^{-\frac{1}{2}}\Lambda_B\Lambda_B^{-\frac{1}{2}}P = P^*P = I$$

$$\begin{cases} M^*AM = \Lambda_{A'} \\ M^*BM = I \end{cases} \Rightarrow \begin{cases} M^*AM = \Lambda_{A'} \\ \Lambda_{A'}M^*BM = \Lambda_{A'} \end{cases}$$

$$AX = BX\Lambda$$

If B has full rank $\Rightarrow B^{-1}AX = X\Lambda$

QZ Algorithm

Let A, B hermitean positive definite

QZ Algorithm

Let A, B hermitean positive definite
Let $B_n \to B$, $\det(B_n) \neq 0$

QZ Algorithm

Let A, B hermitean positive definite
Let $B_n \to B$, $\det(B_n) \neq 0$
$Q_k^*(AB_k^{-1})Q_k = R_k$ Schur decomposition

QZ Algorithm

Let A, B hermitean positive definite
Let $B_n \to B$, $\det(B_n) \neq 0$
$Q_k^*(AB_k^{-1})Q_k = R_k$ Schur decomposition

$$B_k^{-1}Q_k = Z_k S_k^{-1} \Rightarrow Z_k^* B_k^{-1} Q_k = S_k^{-1}$$

QZ Algorithm

Let A, B hermitean positive definite
Let $B_n \to B$, $\det(B_n) \neq 0$
$Q_k^*(AB_k^{-1})Q_k = R_k$ Schur decomposition

$$B_k^{-1}Q_k = Z_k S_k^{-1} \Rightarrow Z_k^* B_k^{-1} Q_k = S_k^{-1}$$

$$Q_k^* B_k Z_k = R_k S_k, \; Q_k^* A Z_k = S_k$$

QZ Algorithm

Let A, B hermitean positive definite
Let $B_n \to B$, $\det(B_n) \neq 0$
$Q_k^*(AB_k^{-1})Q_k = R_k$ Schur decomposition

$$B_k^{-1}Q_k = Z_k S_k^{-1} \Rightarrow Z_k^* B_k^{-1} Q_k = S_k^{-1}$$

$$Q_k^* B_k Z_k = R_k S_k, \; Q_k^* A Z_k = S_k$$

$$Q^* BZ = RS = T, \; Q^* AZ = S$$

QZ Algorithm

Let A, B hermitean positive definite
Let $B_n \to B$, $\det(B_n) \neq 0$
$Q_k^*(AB_k^{-1})Q_k = R_k$ Schur decomposition

$$B_k^{-1}Q_k = Z_k S_k^{-1} \Rightarrow Z_k^* B_k^{-1} Q_k = S_k^{-1}$$

$$Q_k^* B_k Z_k = R_k S_k, \; Q_k^* A Z_k = S_k$$

$$Q^* BZ = RS = T, \; Q^* AZ = S$$

$$\det(A - \lambda B) = \det(QZ^*) \prod_{k=1}^{n} (t_{ii} - \lambda s_{ii})$$

QZ Algorithm

Let A, B hermitean positive definite
Let $B_n \to B$, $\det(B_n) \neq 0$
$Q_k^*(AB_k^{-1})Q_k = R_k$ Schur decomposition

$$B_k^{-1}Q_k = Z_k S_k^{-1} \Rightarrow Z_k^* B_k^{-1} Q_k = S_k^{-1}$$

$$Q_k^* B_k Z_k = R_k S_k, \; Q_k^* A Z_k = S_k$$

$$Q^* BZ = RS = T, \; Q^* AZ = S$$

$$\det(A - \lambda B) = \det(QZ^*) \prod_{k=1}^{n} (t_{ii} - \lambda s_{ii})$$

$$\lambda = \frac{t_{ii}}{s_{ii}}, \; s_{ii} \neq 0$$

QZ Algorithm

Use orthogonal transformations to make A Hessenberg and B upper triangular

QZ Algorithm

Use orthogonal transformations to make A Hessenberg and B upper triangular

$$B_1 = QB, \; A_1 = QA$$

QZ Algorithm

Use orthogonal transformations to make A Hessenberg and B upper triangular

$$B_1 = QB, \; A_1 = QA$$

$$\hat{A} = UA_1, \; \hat{B} = UB_1$$

QZ Algorithm

Use orthogonal transformations to make A Hessenberg and B upper triangular

$$B_1 = QB, \; A_1 = QA$$

$$\hat{A} = UA_1, \; \hat{B} = UB_1$$

Use one transformation to restore B triangular form and one to zero an element of A

QZ Algorithm

Use orthogonal transformations to make A Hessenberg and B upper triangular

$$B_1 = QB, \; A_1 = QA$$

$$\hat{A} = UA_1, \; \hat{B} = UB_1$$

Use one transformation to restore B triangular form and one to zero an element of A
With A Hessenberg and B upper triangular

$$(\hat{A} - \lambda\hat{B}) = Q^*(A - \lambda B)Z$$

QZ Algorithm

Use orthogonal transformations to make A Hessenberg and B upper triangular

$$B_1 = QB, \; A_1 = QA$$

$$\hat{A} = UA_1, \; \hat{B} = UB_1$$

Use one transformation to restore B triangular form and one to zero an element of A
With A Hessenberg and B upper triangular

$$(\hat{A} - \lambda \hat{B}) = Q^*(A - \lambda B)Z$$

Let M be the 2×2 lower submatrix of AB^{-1} with eigenvalues $a, \; b$

QZ Algorithm

Use orthogonal transformations to make A Hessenberg and B upper triangular

$$B_1 = QB, \; A_1 = QA$$

$$\hat{A} = UA_1, \; \hat{B} = UB_1$$

Use one transformation to restore B triangular form and one to zero an element of A
With A Hessenberg and B upper triangular

$$(\hat{A} - \lambda \hat{B}) = Q^*(A - \lambda B)Z$$

Let M be the 2×2 lower submatrix of AB^{-1} with eigenvalues $a, \; b$
Let v be the first column $(AB^{-1} - aI)(AB^{-1} - bI)$

QZ Algorithm

Use orthogonal transformations to make A Hessenberg and B upper triangular

$$B_1 = QB, \;\; A_1 = QA$$

$$\hat{A} = UA_1, \;\; \hat{B} = UB_1$$

Use one transformation to restore B triangular form and one to zero an element of A
With A Hessenberg and B upper triangular

$$(\hat{A} - \lambda \hat{B}) = Q^*(A - \lambda B)Z$$

Let M be the 2×2 lower submatrix of AB^{-1} with eigenvalues $a, \; b$
Let v be the first column $(AB^{-1} - aI)(AB^{-1} - bI)$
v can be computed in $\mathcal{O}(1)$

QZ Algorithm

We need to transform A to an upper triangular matrix

QZ Algorithm

We need to transform A to an upper triangular matrix

$$\hat{A} = Q_0 A$$

QZ Algorithm

We need to transform A to an upper triangular matrix

$$\hat{A} = Q_0 A$$

Q_0 also reduces v to e_1

QZ Algorithm

We need to transform A to an upper triangular matrix

$$\hat{A} = Q_0 A$$

Q_0 also reduces v to e_1 Similarly, we need Z_0 to reduce B to an upper triangular matrix

QZ Algorithm

We need to transform A to an upper triangular matrix

$$\hat{A} = Q_0 A$$

Q_0 also reduces v to e_1 Similarly, we need Z_0 to reduce B to an upper triangular matrix

$$b_n Z_0 = e_n^*$$

Repeat the procces and denote

$$Q = Q_0 Q_1 \ldots Q_n, \; Z = Z_0 Z_1 \ldots Z_n$$

QZ Algorithm

We need to transform A to an upper triangular matrix

$$\hat{A} = Q_0 A$$

Q_0 also reduces v to e_1 Similarly, we need Z_0 to reduce B to an upper triangular matrix

$$b_n Z_0 = e_n^*$$

Repeat the procces and denote

$$Q = Q_0 Q_1 \dots Q_n, \; Z = Z_0 Z_1 \dots Z_n$$

By the implicit Q theorem, Q applies the Francis iteration to AB^{-1}

$$S = Q^* A B^{-1} Q$$

QZ Algorithm

We need to transform A to an upper triangular matrix

$$\hat{A} = Q_0 A$$

Q_0 also reduces v to e_1 Similarly, we need Z_0 to reduce B to an upper triangular matrix

$$b_n Z_0 = e_n^*$$

Repeat the procces and denote

$$Q = Q_0 Q_1 \dots Q_n, \; Z = Z_0 Z_1 \dots Z_n$$

By the implicit Q theorem, Q applies the Francis iteration to AB^{-1}

$$S = Q^* A B^{-1} Q$$

By construction Q and Z simultaneously reduce A and B to Schur form

The Singular Values

The Jacobi method is based on Givens Rotations

The Singular Values

The Jacobi method is based on Givens Rotations
A is hermitean

The Singular Values

The Jacobi method is based on Givens Rotations
A is hermitean
J_{ij} rotates a_i so a_{ij} is 0

The Singular Values

The Jacobi method is based on Givens Rotations
A is hermitean
J_{ij} rotates a_i so a_{ij} is 0
$J_{ij} A J_{ij}^*$ has 2 extra 0

The Singular Values

The Jacobi method is based on Givens Rotations
A is hermitean
J_{ij} rotates a_i so a_{ij} is 0
$J_{ij} A J_{ij}^*$ has 2 extra 0
Repeat the rotations for all off-diagonal entries of A

The Singular Values

The Jacobi method is based on Givens Rotations
A is hermitean
J_{ij} rotates a_i so a_{ij} is 0
$J_{ij} A J_{ij}^*$ has 2 extra 0
Repeat the rotations for all off-diagonal entries of A
The algorithm produces orthogonaly similar matrices to A

The Singular Values

The Jacobi method is based on Givens Rotations
A is hermitean
J_{ij} rotates a_i so a_{ij} is 0
$J_{ij} A J_{ij}^*$ has 2 extra 0
Repeat the rotations for all off-diagonal entries of A
The algorithm produces orthogonaly similar matrices to A

$$A_k = J_k A_{k-1} J_k^* \Rightarrow A_n = JAJ^*$$

The Singular Values

The Jacobi method is based on Givens Rotations
A is hermitean
J_{ij} rotates a_i so a_{ij} is 0
$J_{ij}AJ_{ij}^*$ has 2 extra 0
Repeat the rotations for all off-diagonal entries of A
The algorithm produces orthogonaly similar matrices to A

$$A_k = J_k A_{k-1} J_k^* \Rightarrow A_n = JAJ^*$$

The algorithm goes towards a diagonal form

The Singular Values

The Jacobi method is based on Givens Rotations
A is hermitean
J_{ij} rotates a_i so a_{ij} is 0
$J_{ij} A J_{ij}^*$ has 2 extra 0
Repeat the rotations for all off-diagonal entries of A
The algorithm produces orthogonaly similar matrices to A

$$A_k = J_k A_{k-1} J_k^* \Rightarrow A_n = JAJ^*$$

The algorithm goes towards a diagonal form
Only 4 entries are affected at each step $\Rightarrow$ easy to compute in parallel

Pseudo-inverse

For $A \in M^{n\times m}$

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

Pseudo-inverse

For $A \in M^{n\times m}$

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

$$\sigma_k^+ = \begin{cases} 0, & \sigma_k = 0 \\ \frac{1}{\sigma_k} & \end{cases}$$

Pseudo-inverse

For $A \in M^{n \times m}$

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

$$\sigma_k^+ = \begin{cases} 0, & \sigma_k = 0 \\ \frac{1}{\sigma_k} & \end{cases}$$

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

Pseudo-inverse

For $A \in M^{n \times m}$

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

$$\sigma_k^+ = \begin{cases} 0, & \sigma_k = 0 \\ \frac{1}{\sigma_k} & \end{cases}$$

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

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

Pseudo-inverse

For $A \in M^{n \times m}$

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

$$\sigma_k^+ = \begin{cases} 0, & \sigma_k = 0 \\ \frac{1}{\sigma_k} & \end{cases}$$

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

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

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

Pseudo-inverse

For $A \in M^{n \times m}$

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

$$\sigma_k^+ = \begin{cases} 0, & \sigma_k = 0 \\ \frac{1}{\sigma_k} \end{cases}$$

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

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

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

$$(AA^+)^* = (U\Sigma V^* V\Sigma^+ U^*)^* = U\Sigma V^* V\Sigma^+ U^* = (AA^+)$$

Pseudo-inverse

For $A \in M^{n\times m}$

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

$$\sigma_k^+ = \begin{cases} 0, & \sigma_k = 0 \\ \frac{1}{\sigma_k} & \end{cases}$$

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

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

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

$$(AA^+)^* = (U\Sigma V^* V\Sigma^+ U^*)^* = U\Sigma V^* V\Sigma^+ U^* = (AA^+)$$

$$(A^+A)^* = (V\Sigma^+ U^* U\Sigma V^*)^* = V\Sigma^+ U^* U\Sigma V^* = A^+A$$

References I

- L. N. Trefethen, D. Bau, Numerical linear algebra, SIAM, 1997
- J. Demmel, Applied numerical linear algebra, SIAM, 1997
- P. G. Ciarlet, Introduction to numerical linear algebra and optimization, Cambridge University Press, 1989
- G. W. Stewart, Matrix algorithms volumes I II, SIAM, 1998
- G. W. Stewart, Introduction to matrix computations, Academic Press, 1973
- G. W. Stewart, Ji-guang Sun, Matrix perturbation theory, Academic Press, 1990
- G. H. Golub, C. van Loan, Matrix computations, John Hopkins University Press, 1996

References II

- N. J Higham, Accuracy and stability of numerical algorithms, SIAM, 2002
- J. Wilkinson, The algebraic eigenvalue problem, Oxford University Press, 1988
- C. D. Meyer, Matrix analysis and applied linear algebra, SIAM, 2000
- P. Lancaster, M, Tismenetsky, The theory of matrices, Academic Press, 1985
- G. Strang, Introduction to Linear Algebra, Wellesley-Cambridge Press, 2016
- R. Bhatia, Matrix Analysis, Springer, 1997

References III

R. S. Leite, N. C. Saldanha, C Tomei, The Asymptotics of Wilkinson's Shift: Loss of Cubic Convergence, Foundations of Computational Mathematics, 2010

J. H. Wilkinson The QR algorithm for real symmetric matrices with multiple eigenvalues, The Computer Journal, 1965