QR-Zerlegung mit Householder-Transformationen

Numerische Mathematik 1
WS 2011/12

Orthogonales Eliminieren

Sei $x \in \mathbb{R}^n$ ein Vektor $x \neq 0$.

Ziel: Ein orthogonales $H \in \mathbb{R}^{n,n}$ bestimmen, sodass

$$Hx = \pm \|x\| e_1,$$

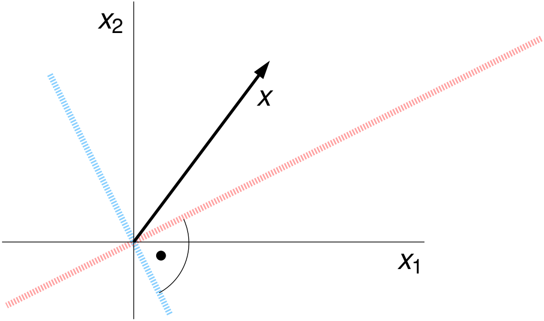
ein Vielfaches des ersten Einheitsvektors

$$e_1 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

ist.

Householder-Transformation

Das kann man durch Drehungen (Givens-Rotationen) oder durch Spiegelungen (Householder-Transformationen) erreichen.



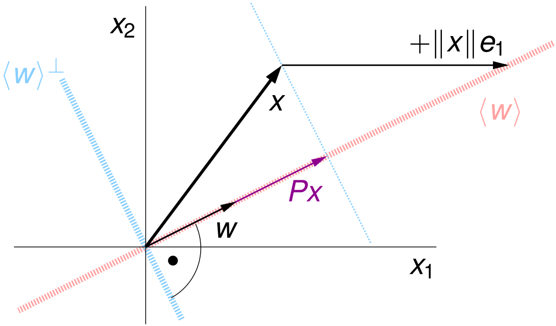
Householder-Transformation

Setzt man

$$\tilde{w} := x + \operatorname{sgn}(x_1)\|x\| e_1, \qquad \text{und } w := \frac{\tilde{w}}{\|\tilde{w}\|}$$

dann ist der Orthogonalprojektor auf $\langle w \rangle := \operatorname{span}(w)$ gegeben durch

$$P := ww^T = \left(\frac{\tilde{w}}{\|\tilde{w}\|}\right)\left(\frac{\tilde{w}^T}{\|\tilde{w}\|}\right) = \frac{\tilde{w}\tilde{w}^T}{\tilde{w}^T\tilde{w}}.$$



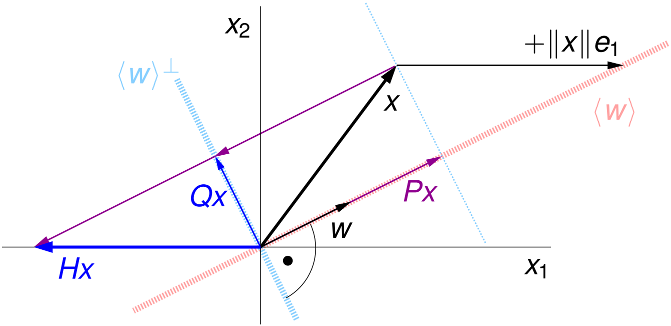
Householder-Transformation

Damit ist der (duale) Orthogonalprojektor auf $\langle w \rangle^\perp$ gegeben durch

$$Q := I - P = I - ww^T,$$

und entsprechend die Householder-Spiegelung (an $\langle w \rangle^\perp$)

$$H := I - 2P = I - 2ww^T.$$



Alternative Darstellung

Die Householder-Transformation $H \in \mathbb{R}^{n,n}$ hat also die Darstellung

$$H = I - 2ww^T = I - 2\frac{ww^T}{w^T w},$$

wobei $w \in \mathbb{R}^n$ mit $\|w\| = 1$.

Ist $v \in \mathbb{R}^n \setminus \{0\}$ aber ein linear abhängiger Vektor $v = \alpha \cdot w$, mit $\alpha \in \mathbb{R} \setminus \{0\}$ so gilt ebenfalls

$$\begin{aligned} I - 2\frac{vv^T}{v^T v} &= I - 2\frac{\alpha^2 ww^T}{\alpha^2 w^T w} \\ &= I - 2\frac{ww^T}{w^T w} = H. \end{aligned}$$

Normalisierte Darstellung

Ist $x \neq 0$ so ist der erste Eintrag von

$$\tilde{w} := x + \operatorname{sgn}(x_1)\|x\|e_1$$

ebenfalls nicht 0, d.h. $\tilde{w}_1 \neq 0$.

Dann erfüllt der Vektor

$$v := \frac{1}{\tilde{w}_1} \cdot \tilde{w} \in \mathbb{R}^n$$

die Bedingung $v_1 = 1$.

Normalisierte Darstellung

Somit ist die Householder-Transformation gegeben durch

$$\begin{aligned} H &= I - 2P = I - 2\frac{vv^T}{v^Tv} = I - \frac{2}{\|v\|^2}vv^T \\ &= I - \beta vv^T, \end{aligned}$$

wobei

$$\beta := \frac{2}{\|v\|^2}.$$

Berechnung von v und β

```
function [beta, v] = house(x)

n = length(x);

if( norm(x) < n*eps )
    beta = 0;
    v = eye(n,1);
else
    % bereche w
    w = ...

    % berechne v
    v = w/w(1);

    % berechne beta
    beta = ...
end
```

Vorteil

Die Householder-Transformation kann man also als

$$H = I - \beta v v^T$$

schreiben, wobei $\beta \in \mathbb{R}$ und $v \in \mathbb{R}^n$ die Form

$$v = \begin{bmatrix} 1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

hat.

	H	$I - \beta v v^T$
Speicherbedarf:	n^2	n

H anwenden

Es sei $\beta \in \mathbb{R}$, $v \in \mathbb{R}^n$ und die Householder-Transformation

$$H = I - \beta v v^T \in \mathbb{R}^{n,n}.$$

Dann ist für $X \in \mathbb{R}^{n,\ell}$ gerade

$$\begin{aligned} HX &= (I - \beta v v^T)X = X - \beta v v^T X \\ &= X - \beta v \underbrace{\left(v^T X\right)}_{=:X_1 \in \mathbb{R}^{1,\ell} \text{ in } \mathcal{O}(n\ell)} = X - \beta v X_1 \\ &= X - \beta \underbrace{(v X_1)}_{=:X_2 \in \mathbb{R}^{n,\ell} \text{ in } \mathcal{O}(n\ell)} = \underbrace{X - \beta X_2}_{\text{in } \mathcal{O}(n\ell)} =: Y. \end{aligned}$$

	H	$I - \beta v v^T$
Laufzeit, Matrix-Vektor-Multiplikation:	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
Laufzeit, Matrix-Matrix-Multiplikation:	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$

H anwenden in Matlab

```
function Y = h_anwenden(v, beta, X)

Y = zeros(size(X));

% X1 in der ersten Zeile von Y speichern

% X2 in Y speichern

% Y ausrechnen
Y = ...
```

QR-Zerlegung (Formal)

Wir wollen eine QR-Zerlegung der Matrix

$$A = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & a_{23}^{(1)} \\ a_{31}^{(1)} & a_{32}^{(1)} & a_{33}^{(1)} \\ a_{41}^{(1)} & a_{42}^{(1)} & a_{43}^{(1)} \\ a_{51}^{(1)} & a_{52}^{(1)} & a_{53}^{(1)} \end{bmatrix} \in \mathbb{R}^{5,3}$$

mittels Householder-Transformationen berechnen.

QR-Zerlegung (Formal)

$$A = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & a_{23}^{(1)} \\ a_{31}^{(1)} & a_{32}^{(1)} & a_{33}^{(1)} \\ a_{41}^{(1)} & a_{42}^{(1)} & a_{43}^{(1)} \\ a_{51}^{(1)} & a_{52}^{(1)} & a_{53}^{(1)} \end{bmatrix}$$

Schritt 1: Berechne $v_1 \in \mathbb{R}^5$ und β_1 sodass mit

$$H_1 := I - \beta_1 v_1 v_1^T \in \mathbb{R}^{5,5}$$

und

$$\hat{H}_1 := H_1 \qquad \text{gilt} \qquad \hat{H}_1 A = \begin{bmatrix} a_{11}^{(2)} & a_{12}^{(2)} & a_{13}^{(2)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} \\ 0 & a_{32}^{(2)} & a_{33}^{(2)} \\ 0 & a_{42}^{(2)} & a_{43}^{(2)} \\ 0 & a_{52}^{(2)} & a_{53}^{(2)} \end{bmatrix}.$$

QR-Zerlegung (Formal)

$$\hat{H}_1 A = \begin{bmatrix} a_{11}^{(2)} & a_{12}^{(2)} & a_{13}^{(2)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} \\ 0 & a_{32}^{(2)} & a_{33}^{(2)} \\ 0 & a_{42}^{(2)} & a_{43}^{(2)} \\ 0 & a_{52}^{(2)} & a_{53}^{(2)} \end{bmatrix}$$

Schritt 2: Berechne $v_2 \in \mathbb{R}^4$ und β_2 sodass mit

$$H_2 := I - \beta_2 v_2 v_2^T \in \mathbb{R}^{4,4}$$

und

$$\hat{H}_2 := \begin{bmatrix} 1 & \\ & H_2 \end{bmatrix} \qquad \text{gilt} \qquad \hat{H}_2 \hat{H}_1 A = \begin{bmatrix} a_{11}^{(3)} & a_{12}^{(3)} & a_{13}^{(3)} \\ 0 & a_{22}^{(3)} & a_{23}^{(3)} \\ 0 & 0 & a_{33}^{(3)} \\ 0 & 0 & a_{43}^{(3)} \\ 0 & 0 & a_{53}^{(3)} \end{bmatrix}.$$

QR-Zerlegung (Formal)

$$\hat{H}_2 \hat{H}_1 A = \begin{bmatrix} a_{11}^{(3)} & a_{12}^{(3)} & a_{13}^{(3)} \\ 0 & a_{22}^{(3)} & a_{23}^{(3)} \\ 0 & 0 & a_{33}^{(3)} \\ 0 & 0 & a_{43}^{(3)} \\ 0 & 0 & a_{53}^{(3)} \end{bmatrix}$$

Schritt 3: Berechne $v_3 \in \mathbb{R}^3$ und β_3 sodass mit

$$H_3 := I - \beta_3 v_3 v_3^T \in \mathbb{R}^{3,3}$$

und

$$\hat{H}_3 := \begin{bmatrix} 1 & & \\ & 1 & \\ & & H_3 \end{bmatrix} \qquad \text{gilt} \qquad \hat{H}_3 \hat{H}_2 \hat{H}_1 A = \begin{bmatrix} a_{11}^{(4)} & a_{12}^{(4)} & a_{13}^{(4)} \\ 0 & a_{22}^{(4)} & a_{23}^{(4)} \\ 0 & 0 & a_{33}^{(4)} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

QR-Zerlegung (Formal)

$$\underbrace{\hat{H}_3 \hat{H}_2 \hat{H}_1}_{=:Q^T} A = \begin{bmatrix} a_{11}^{(4)} & a_{12}^{(4)} & a_{13}^{(4)} \\ 0 & a_{22}^{(4)} & a_{23}^{(4)} \\ 0 & 0 & a_{33}^{(4)} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} =: R.$$

Dann ist Q orthogonal, R obere Dreiecksmatrix und somit

$$A = QR$$

eine QR-Zerlegung von A.

QR-Zerlegung (im Rechner)

Es werden die H_i, $\hat{H}_i$, etc. **nicht** explizit gebildet.

Man rechnet nur sog. Elementarreflektoren v_1, v_2, ... aus und speichert diese in den frei werdenden Null-Elementen (die erste 1 muss nicht gespeichert werden).

Dabei braucht man ein extra Array für die β_1, β_2, ...

QR-Zerlegung (im Rechner)

Mit obigen Bezeichnungen durchläuft man die Abfolge

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & a_{23}^{(1)} \\ a_{31}^{(1)} & a_{32}^{(1)} & a_{33}^{(1)} \\ a_{41}^{(1)} & a_{42}^{(1)} & a_{43}^{(1)} \\ a_{51}^{(1)} & a_{52}^{(1)} & a_{53}^{(1)} \end{bmatrix} \rightsquigarrow \begin{bmatrix} a_{11}^{(2)} & a_{12}^{(2)} & a_{13}^{(2)} \\ \color{red}{v_{1,2}} & a_{22}^{(2)} & a_{23}^{(2)} \\ \color{red}{v_{1,3}} & a_{32}^{(2)} & a_{33}^{(2)} \\ \color{red}{v_{1,4}} & a_{42}^{(2)} & a_{43}^{(2)} \\ \color{red}{v_{1,5}} & a_{52}^{(2)} & a_{53}^{(2)} \end{bmatrix}$$

$$\beta = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \qquad \beta = \begin{bmatrix} \beta_1 & 0 & 0 \end{bmatrix}$$

$$\rightsquigarrow \begin{bmatrix} a_{11}^{(3)} & a_{12}^{(3)} & a_{13}^{(3)} \\ \color{red}{v_{1,2}} & a_{22}^{(3)} & a_{23}^{(3)} \\ \color{red}{v_{1,3}} & \color{red}{v_{2,2}} & a_{33}^{(3)} \\ \color{red}{v_{1,4}} & \color{red}{v_{2,3}} & a_{43}^{(3)} \\ \color{red}{v_{1,5}} & \color{red}{v_{2,4}} & a_{53}^{(3)} \end{bmatrix} \rightsquigarrow \begin{bmatrix} a_{11}^{(4)} & a_{12}^{(4)} & a_{13}^{(4)} \\ \color{red}{v_{1,2}} & a_{22}^{(4)} & a_{23}^{(4)} \\ \color{red}{v_{1,3}} & \color{red}{v_{2,2}} & a_{33}^{(4)} \\ \color{red}{v_{1,4}} & \color{red}{v_{2,3}} & \color{red}{v_{3,2}} \\ \color{red}{v_{1,5}} & \color{red}{v_{2,4}} & \color{red}{v_{3,3}} \end{bmatrix}$$

$$\beta = \begin{bmatrix} \beta_1 & \beta_2 & 0 \end{bmatrix} \qquad \beta = \begin{bmatrix} \beta_1 & \beta_2 & \beta_3 \end{bmatrix}$$

wobei $\quad v_{i,j} := v_i\Big|_j \quad$ den j-ten Eintrag von v_i bezeichnet.

QR-Zerlegung (in LAPACK)

```
      SUBROUTINE DGEQRF( M, N, A, LDA, TAU, WORK, LWORK, INFO )
*
*     .. Scalar Arguments ..
      INTEGER            INFO, LDA, LWORK, M, N
*     ..
*     .. Array Arguments ..
      DOUBLE PRECISION   A( LDA, * ), TAU( * ), WORK( * )
*     ..
*
*  Purpose
*  =======
*
*  DGEQRF computes a QR factorization of a real M-by-N matrix A:
*  A = Q * R.
*
*  Arguments
*  =========
*
*  M       (input) INTEGER
*          The number of rows of the matrix A.  M >= 0.
*
*  N       (input) INTEGER
*          The number of columns of the matrix A.  N >= 0.
*
*  A       (input/output) DOUBLE PRECISION array, dimension (LDA,N)
*          On entry, the M-by-N matrix A.
*          On exit, the elements on and above the diagonal of the array
*          contain the min(M,N)-by-N upper trapezoidal matrix R (R is
*          upper triangular if m >= n); the elements below the diagonal,
*          with the array TAU, represent the orthogonal matrix Q as a
*          product of min(m,n) elementary reflectors (see Further
*          Details).
*
*  LDA     (input) INTEGER
```

QR-Zerlegung in Matlab

```
function [QR, betas] = qr_zerlegung(A)

[n, m] = size(A);
min_nm = min([n-1, m]);

betas = zeros(1, min_nm);

for i=1:min_nm
   [beta_i, v_i] = house(...);

   % A^(i) --> A^(i+1)
   ... h_anwenden(...) ...

   A(i+1:n,i) = v_i(2:end);
   betas(i) = beta_i;
end

QR = A;
```

Problem

Hat man die QR-Zerlegung in der Form

$$\begin{bmatrix} a_{11}^{(4)} & a_{12}^{(4)} & a_{13}^{(4)} \\ v_{1,2} & a_{22}^{(4)} & a_{23}^{(4)} \\ v_{1,3} & v_{2,2} & a_{33}^{(4)} \\ v_{1,4} & v_{2,3} & v_{3,2} \\ v_{1,5} & v_{2,4} & v_{3,3} \end{bmatrix}$$
$$\beta = \begin{bmatrix} \beta_1 & \beta_2 & \beta_3 \end{bmatrix}$$

gespeichert, so kann man

$$Q = \hat{H}_1 \hat{H}_2 \hat{H}_3$$

und ebenso (da Householder-Transformationen symmetrisch sind)

$$Q^T = \hat{H}_3 \hat{H}_2 \hat{H}_1$$

nicht so einfach mit einem Vektor (oder mit einer Matrix) Multiplizieren.

Lösung

```
function Y = q_anwenden(QR, betas, X, transponiert)

if( transponiert )

   % hier soll Q^T X ausgerechnet werden
   for i=...
      ... h_anwenden(...) ...
   end

else

   % hier soll Q X ausgerechnet werden
   for i=...
      ... h_anwenden(...) ...
   end

end
```

Q-Anwenden (in LAPACK)

```
      SUBROUTINE DORMQR( SIDE, TRANS, M, N, K, A, LDA, TAU, C, LDC,
     $                   WORK, LWORK, INFO )
*
*  Purpose
*  =======
*
*  DORMQR overwrites the general real M-by-N matrix C with
*
*                  SIDE = 'L'     SIDE = 'R'
*  TRANS = 'N':      Q * C          C * Q
*  TRANS = 'T':      Q**T * C       C * Q**T
*
*  where Q is a real orthogonal matrix defined as the product of k
*  elementary reflectors
*
*        Q = H(1) H(2) . . . H(k)
*
*  as returned by DGEQRF. Q is of order M if SIDE = 'L' and of order N
*  if SIDE = 'R'.
*
*  Arguments
*  =========
*
*  TRANS   (input) CHARACTER*1
*          = 'N':  No transpose, apply Q;
*          = 'T':  Transpose, apply Q**T.
*
*  A       (input) DOUBLE PRECISION array, dimension (LDA,K)
*          The i-th column must contain the vector which defines the
*          elementary reflector H(i), for i = 1,2,...,k, as returned by
*          DGEQRF in the first k columns of its array argument A.
*          A is modified by the routine but restored on exit.
*
*  SIDE    (input) CHARACTER*1
```

Lineares Ausgleichsproblem

Sei $A \in \mathbb{R}^{n,m}$ eine Matrix mit vollem Spaltenrang und $b \in \mathbb{R}^n$. Meistens ist auch $n \gg m$.

Ziel: Das *lineare Ausgleichsproblem* lösen, d.h. ein $x^* \in \mathbb{R}^m$ bestimmen mit

$$\min_{x \in \mathbb{R}^m} \|Ax - b\| = \|Ax^* - b\| .$$

Lineares Ausgleichsproblem

Hat man eine QR-Zerlegung von $A \in \mathbb{R}^{n,m}$ berechnet so ist

$$\begin{aligned}
\min_{x\in\mathbb{R}^m} \|Ax-b\| &= \min_{x\in\mathbb{R}^m} \|QRx-b\| = \min_{x\in\mathbb{R}^m} \left\|Q(Rx-Q^Tb)\right\| \\
&= \min_{x\in\mathbb{R}^m} \left\|Rx-Q^Tb\right\|,
\end{aligned}$$

d.h. mit den Bezeichnungen $R =: \begin{bmatrix} R_1 \\ 0 \end{bmatrix}$ und $Q^Tb =: \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$,

wobei $R_1 \in \mathbb{R}^{m,m}$, $b_1 \in \mathbb{R}^m$, $b_2 \in \mathbb{R}^{n-m}$, hat man

$$\begin{aligned}
&\min_{x\in\mathbb{R}^m} \|Ax-b\|^2 = \min_{x\in\mathbb{R}^m} \left\| \begin{bmatrix} R_1 \\ 0 \end{bmatrix} x - \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \right\|^2 \\
= \;& \min_{x\in\mathbb{R}^m} \|R_1x-b_1\|^2 + \|0-b_2\|^2 = \|b_2\| + \min_{x\in\mathbb{R}^m} \|R_1x-b_1\|^2,
\end{aligned}$$

mit der Lösung

$$x^* = R^{-1}b_1$$

und dem Residuum

$$\|Ax^*-b\| = \|b_2\|.$$

Ausgleichsproblem lösen (in Matlab)

```
function [x, res] = ausgleichsproblem(A, b)

[n, m] = size(A);

[QR, beta] = qr_zerlegung(A);

QTb = q_anwenden(QR, beta, b, true);

...

x = ...;
res = ...;
```

Asymptotische Laufzeit

Es sei $m \in \mathbb{N}$ fest, $n \in \mathbb{N}$ mit $n > m$ variabel, $A \in \mathbb{R}^{n,m}$ und $b \in \mathbb{R}^n$.

Dann ist die asymptotische Laufzeit (für $n \to \infty$) zur Lösung des linearen Ausgleichsproblems

$$\min_{x \in \mathbb{R}^m} \|Ax - b\|,$$

nach obiger Methode

$$\mathcal{O}(n).$$

Würde man erst orthogonales $Q \in \mathbb{R}^{n,n}$ und eine obere Dreiecksmatrix $R \in \mathbb{R}^{n,m}$ berechnen mit $QR = A$, so würde schon die notwendige Matrix-Vektor-Multiplikation $Q^T b$

$$\mathcal{O}(n^2)$$

brauchen.

Berechnung der SVD

Um die SVD einer Matrix der Form

$$A = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix}$$

zu berechnen, wendet man Householder-Transformationen (von links und rechts) an.

Berechnung der SVD

Man durchläuft die Abfolge

$$A \rightsquigarrow \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{bmatrix} \rightsquigarrow \begin{bmatrix} * & * & 0 & 0 \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{bmatrix} \rightsquigarrow \begin{bmatrix} * & * & 0 & 0 \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \end{bmatrix}$$

$$\rightsquigarrow \begin{bmatrix} * & * & 0 & 0 \\ 0 & * & * & 0 \\ 0 & 0 & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \end{bmatrix} \rightsquigarrow \begin{bmatrix} * & * & 0 & 0 \\ 0 & * & * & 0 \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \\ 0 & 0 & 0 & * \\ 0 & 0 & 0 & * \\ 0 & 0 & 0 & * \end{bmatrix} \rightsquigarrow \begin{bmatrix} * & * & 0 & 0 \\ 0 & * & * & 0 \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} =: B$$

Berechnung der SVD

Dann kann man die Eigenwerte von

$$B^T B = \begin{bmatrix} * & 0 & 0 & 0 & 0 & 0 & 0 \\ * & * & 0 & 0 & 0 & 0 & 0 \\ 0 & * & * & 0 & 0 & 0 & 0 \\ 0 & 0 & * & * & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} * & * & 0 & 0 \\ 0 & * & * & 0 \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} * & * & 0 & 0 \\ * & * & * & 0 \\ 0 & * & * & * \\ 0 & 0 & * & * \end{bmatrix},$$

einer symmetrischen positiv semi-definiten Tridiagonalmatrix, berechnen.

Eigenwerte von $B^T B$ (in LAPACK)

```
      SUBROUTINE DSTEV( JOBZ, N, D, E, Z, LDZ, WORK, INFO )
*
*     .. Scalar Arguments ..
      CHARACTER          JOBZ
      INTEGER            INFO, LDZ, N
*     .. Array Arguments ..
      DOUBLE PRECISION   D( * ), E( * ), WORK( * ), Z( LDZ, * )
*
*  Purpose
*  =======
*
*  DSTEV computes all eigenvalues and, optionally, eigenvectors of a
*  real symmetric tridiagonal matrix A.
*
*  Arguments
*  =========
*
*  JOBZ    (input) CHARACTER*1
*          = 'N':  Compute eigenvalues only;
*          = 'V':  Compute eigenvalues and eigenvectors.
*
*  N       (input) INTEGER
*          The order of the matrix.  N >= 0.
*
*  D       (input/output) DOUBLE PRECISION array, dimension (N)
*          On entry, the n diagonal elements of the tridiagonal matrix
*          A.
*          On exit, if INFO = 0, the eigenvalues in ascending order.
*
*  E       (input/output) DOUBLE PRECISION array, dimension (N-1)
*          On entry, the (n-1) subdiagonal elements of the tridiagonal
*          matrix A, stored in elements 1 to N-1 of E.
*          On exit, the contents of E are destroyed.
*
```