

# Algorithmik kontinuierlicher Systeme

## *Direkte Verfahren für Lineare Gleichungssysteme*



- Direkte Verfahren, d.h. solche, die in endlich vielen Schritten das exakte Ergebnis liefern (exakte Rechnung voraus-gesetzt) basieren auf der Faktorisierung von  $A$ 
  - ▶  $A = A_1 A_2$  dann ist  $Ax = b$  äquivalent zu
 
$$A_1 y = b \quad \text{und} \quad A_2 x = y$$
- LR-Zerlegung
- Vorwärts/Rückwärts-Substitution für Dreiecksmatrizen:  $O(n^2)$
- Linpack ist der de-fakto Benchmark zur Messung der Leistungsfähigkeit von Supercomputern.

- Spezialfälle
  - Bandmatrizen
  - sym. pos. def. Matrizen
- Erweiterungen
  - Elimination mit orthogonalen Matrizen
  - Householder-Reflektionen
  - Givens-Rotationen
  - Lösung von low-rank-modifizierten LGS

- Bandmatrizen:
  - Beschränke die Elimination auf die Elemente, die innerhalb des Bandes und unterhalb der Diagonalen liegen
  - Aufwand bei Bandbreite einer Matrix der Ordnung  $n$  und Bandbreite  $m < n$  ist  $O(m^2n)$
- Wichtiger Spezialfall: Tridiagonale Matrizen
- Speichersparende Datenstruktur entscheidend!

$$\begin{bmatrix} b_1 & c_1 & 0 & \cdots & 0 & 0 \\ a_1 & b_2 & c_2 & & 0 & 0 \\ 0 & a_2 & b_3 & \ddots & 0 & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & b_{n-1} & c_{n-1} \\ 0 & 0 & 0 & \cdots & a_{n-1} & b_n \end{bmatrix}$$

- Tridiagonale Matrizen
- Struktur der LR-Zerlegung (siehe auch Übungsaufgabe):

$$L = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ l_1 & 1 & 0 & & 0 & 0 \\ 0 & l_2 & 1 & \ddots & 0 & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & l_{n-1} & 1 \end{bmatrix} \quad R = \begin{bmatrix} r_1 & c_1 & 0 & \cdots & 0 & 0 \\ 0 & r_2 & c_2 & & 0 & 0 \\ 0 & 0 & r_3 & \ddots & 0 & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & r_{n-1} & c_{n-1} \\ 0 & 0 & 0 & \cdots & 0 & r_n \end{bmatrix}$$

- Lösbar mit  $O(n)$  Operationen  
(genauer: je  $n-1$  Div., Add. und Mult. )

$$r_1 = b_1, l_1 = a_1 / r_1, r_2 = b_2 - l_1 * c_1, \dots, l_{n-1} = a_{n-1} / r_{n-1}, r_n = b_n - l_{n-1} * c_{n-1}$$

- Allgemeine dünn besetzte Matrizen: Fill-In
- Vorhandene Nullen in der Matrix werden durch die Elimination zerstört.
- Dies treibt die Zahl der Operationen und den Speicherbedarf nach oben
- Es wurden viele Algorithmen entwickelt, die den Fill-In durch eine geeignete Permutation der Matrix minimieren sollen: sog. **sparse matrix solver**.
- Lösungen für Spezialfälle:
  - *Nested Dissection, Minimum Degree, ...*
- Wenn solche Algorithmen für Parallelrechner entwickelt werden sollen, dann ist dies weiterhin ein heisses Forschungsthema

- Ist  $A$  **symmetrisch**, d.h.  $A = A^T$  und **positiv definit**, dann kann man  $A$  faktorisieren in

$$A = L \cdot D \cdot L^T$$

dabei ist  $L$  das  $L$  der LR-Zerlegung  $D$  der Diagonalanteil von  $R$ .

- wg positiv definit:  $D$  hat in der Diagonale nur positive Elemente, Alternative:

$$A = L \cdot D \cdot L^T = L \cdot D^{1/2} \cdot D^{1/2} \cdot L^T = \tilde{L} \cdot \tilde{L}^T$$

- Im Algorithmus kann man ausnützen, dass die beiden Dreiecksfaktoren gleich sind, man braucht nur einen der beiden explizit berechnen und spart so grob die Hälfte der Operationen.

- Für die Cholesky-Zerlegung ist keine Pivotsuche nötig.
- Der Algorithmus ist auch ohne Pivotsuche stabil.
  - Es können keine 0-Pivots auftreten.
- André-Louis Cholesky, 1875 – 1918 (Geodät)
- Vorteile im Vergleich mit LR:
  - ▶ keine Pivotsuche notwendig,
  - ▶ Aufwand etwa halb so groß wie bei LR.
- Aber: funktioniert nur für symmetrisch positiv definite (SPD) Matrizen !



- Direkte Verfahren, d.h. solche, die in endlich vielen Schritten das exakte Ergebnis liefern (exakte Rechnung voraus-gesetzt) basieren auf der Faktorisierung von  $A$ 
  - ▶  $A = A_1 A_2$  dann ist  $Ax = b$  äquivalent zu
 
$$A_1 y = b \quad \text{und} \quad A_2 x = y$$
- LR-Zerlegung (Variante für SPD: Cholesky)
- Vorwärts/Rückwärts-Substitution für Dreiecksmatrizen:  $O(n^2)$
- Stabilität nur durch Pivotsuche
- Aufwand für LR ist  $O(n^3)$ 
  - ▶ Für Bandmatrizen ist der Aufwand wesentlich geringer
  - ▶ Insbesondere für tridiagonale Matrizen:  $O(n)$

- Faktorisierungsmethode:  $A = A_1 A_2 : A_1 y = b \text{ \& } A_2 x = y$
- LR-Zerlegung (mittels Gauss-Elimination /-scherung)
- Pivotisierung
- Aufwand : allgemein / Band / tridiagonal
- Cholesky-Verfahren

- Motivation:  
Durch Gauss-Scherungen kann ein gut konditioniertes Gl.-system in ein schlecht konditioniertes transformiert werden.
- Dies muss durch die (Spalten-) Pivotsuche verhindert werden.
- Verwendet man Rotationen oder Spiegelungen zur Elimination anstelle der Gauss-Scherungen hat man
  - ▶ Bessere numerische Eigenschaften (da die Spiegelungen und Rotationen „längentreu“ und „winkeltreu“ sind)
  - ▶ Orthogonale Matrizen (  $Q^T Q = Id$  )  
→ QR-Zerlegung

- Householder-Spiegelungen

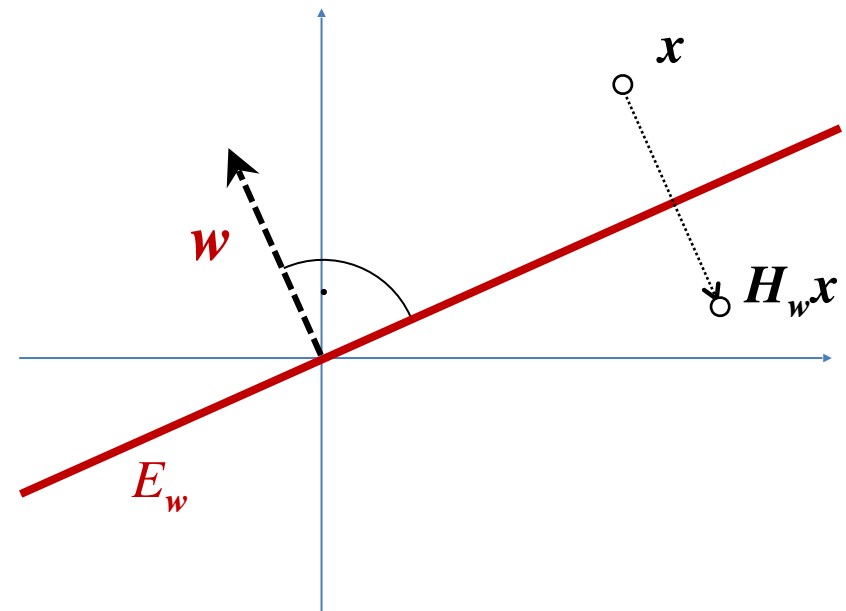
Spiegelung an einer Geraden ( $\mathbb{R}^2$ ), Ebene, ( $\mathbb{R}^3$ ),  
Hyper-Ebene ( $\mathbb{R}^n$ ) :

Eine **Hyper-Ebene** wird  
durch einen senkrechten  
Vektor  $w$  beschrieben:

$$E_w = \{x : x \circ w = 0\}$$

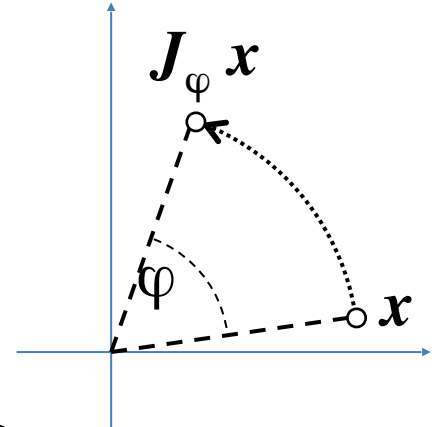
$$H_w : x \mapsto x - \frac{2 \cdot w \circ x}{w \circ w} \cdot w$$

$$H_w = Id - \frac{2}{w \circ w} \cdot w \cdot w^T$$



- Givens-Rotationen (Jacobi-Rotationen)
- Rotation in  $\mathbb{R}^2$ :

$$\mathbf{J}_\varphi : \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



- Rotationen in  $\mathbb{R}^3$ : Rotation um eine Achse.  
Givens-/Jacobi-Rotationen sind Rotationen um eine Koordinatenachse

$$\mathbf{J}_{21}(\varphi) = \begin{bmatrix} \cos \varphi & -\sin \varphi & 0 \\ \sin \varphi & \cos \varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{J}_{13}(\varphi) = \begin{bmatrix} \cos \varphi & 0 & \sin \varphi \\ 0 & 1 & 0 \\ -\sin \varphi & 0 & \cos \varphi \end{bmatrix} \quad \mathbf{J}_{32}(\varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \varphi & -\sin \varphi \\ 0 & \sin \varphi & \cos \varphi \end{bmatrix}$$

- Givens-Rotationen (Jacobi-Rotationen)
- Allgemein im  $\mathbb{R}^n$  : nur Rotation zweier Koordinaten

$$J_{ij}(\varphi) = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & \cos \varphi & -\sin \varphi & \\ & & & \sin \varphi & \cos \varphi & \\ & & & & & 1 & \\ & & & & & & \ddots & \\ & & & & & & & 1 \end{bmatrix}$$

$j$ 
 $i$

Ansätze:

1. mit einer Householder-Spiegelungen in einer Spalte Nullen einfügen (außer Diagonalelement)

→ nach  $n-1$  Schritten erhält man die Dreiecksmatrix  $R$

2. Mit einer Givens-Rotationen ein Element (unterhalb der Diagonalen) zu Null machen

→ nach  $n(n-1)/2$  Schritten erhält man die Dreiecksmatrix  $R$

- ▶ Inverse :  $J_{ij}(\varphi)^{-1} = J_{ij}(-\varphi) = J_{ij}(\varphi)^T$  ;  
 $H_w^{-1} = H_w = H_w^T$
- ▶ Givens-Rotationen und Householder-Spiegelungen benötigen keine Pivotsuche (die Vertauschung ist als Spezialfall enthalten)
  - ★ deshalb auch leichter parallelisierbar
- ▶ Spiegelungen und Rotationen erhalten (in der 2-Norm) die Kondition des Ausgangssystems (siehe später)
- ▶ Aufwand (arithmetische Op.) im Vergleich zu LR :  
 etwa 2x (Householder-Spiegelungen) bzw. 4x (Givens-Rotationen)
  - ▶ Bei Givensrotationen gibt es eine Variante, die nur gleich teuer wie Householder ist.
  - ▶ Dazu muss man zusätzliche Skalierungen einführen: rationale Givensrotationen, siehe unten.

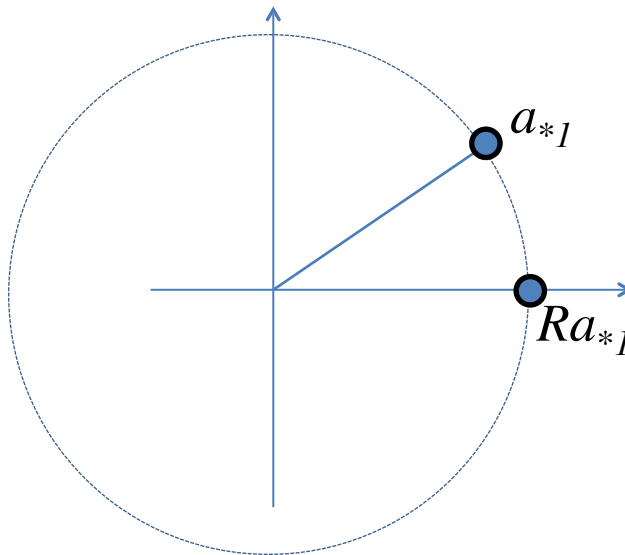


- 2x2-Fall: Gesucht eine Rotationsmatrix so dass

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} * & * \\ 0 & * \end{bmatrix} \text{ wobei } c = \cos(\varphi), s = \sin(\varphi)$$

d.h. Der Vektor  $\begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix}$  muss auf ein Vielfaches des Einheitsvektors abgebildet (gedreht) werden.

1. Fall:  
 $a_{11} > 0$



$$c = \cos(\varphi) = \frac{a_{11}}{\sqrt{a_{11}^2 + a_{21}^2}}$$

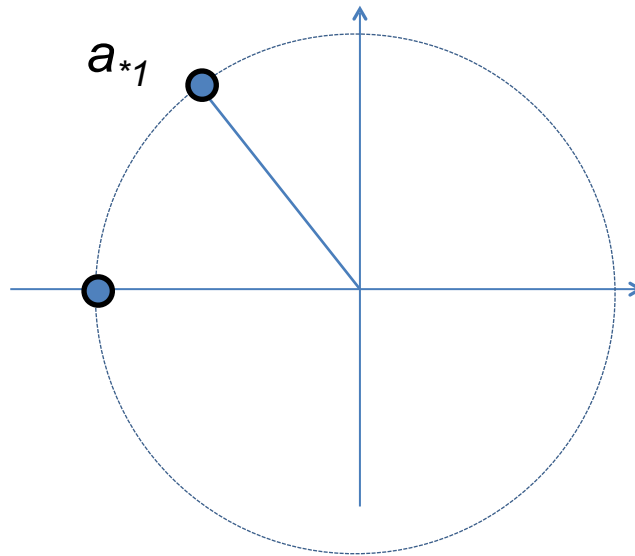
$$s = \sin(\varphi) = \frac{-a_{21}}{\sqrt{a_{11}^2 + a_{21}^2}}$$

- 2x2-Fall: Gesucht eine Rotationsmatrix so dass

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} * & * \\ 0 & * \end{bmatrix} \text{ wobei } c = \cos(\varphi), s = \sin(\varphi)$$

d.h. Der Vektor  $\begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix}$  muss auf ein Vielfaches des Einheitsvektors abgebildet (gedreht) werden.

2. Fall:  
 $a_{11} < 0$



$$c = \cos(\varphi) = \frac{-a_{11}}{\sqrt{a_{11}^2 + a_{21}^2}}$$

$$s = \sin(\varphi) = \frac{a_{21}}{\sqrt{a_{11}^2 + a_{21}^2}}$$

- Allgemeiner Fall:

$$J_{ij}(\varphi) = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & c & & -s & \\ & & & \ddots & & \\ & & s & & c & \\ & & & & & \ddots & \\ & & & & & & 1 \end{bmatrix} \begin{matrix} \leftarrow j \\ \leftarrow i \end{matrix}$$

$\begin{matrix} \uparrow & \uparrow \\ j & i \end{matrix}$

$$c = \cos(\varphi) = \frac{\sigma \cdot a_{jj}}{\sqrt{a_{jj}^2 + a_{ij}^2}}, \quad s = \sin(\varphi) = \frac{-\sigma \cdot a_{ij}}{\sqrt{a_{jj}^2 + a_{ij}^2}}, \quad \text{wobei } \sigma = \text{sign}(a_{jj})$$

- $A$  eine  $m \times n$  - Matrix:

```

for j=1..nstar    # nstar wie unten definiert
  for i=j+1..m
    # bestimme c und s in  $J_{i,j}$  wie oben beschrieben
     $\mathbf{A} = \mathbf{J}_{i,j} \mathbf{A}$  # aber nicht als Matrixmultiplikation
    # dann ist das Element  $a_{i,j} = 0$ 
  
```

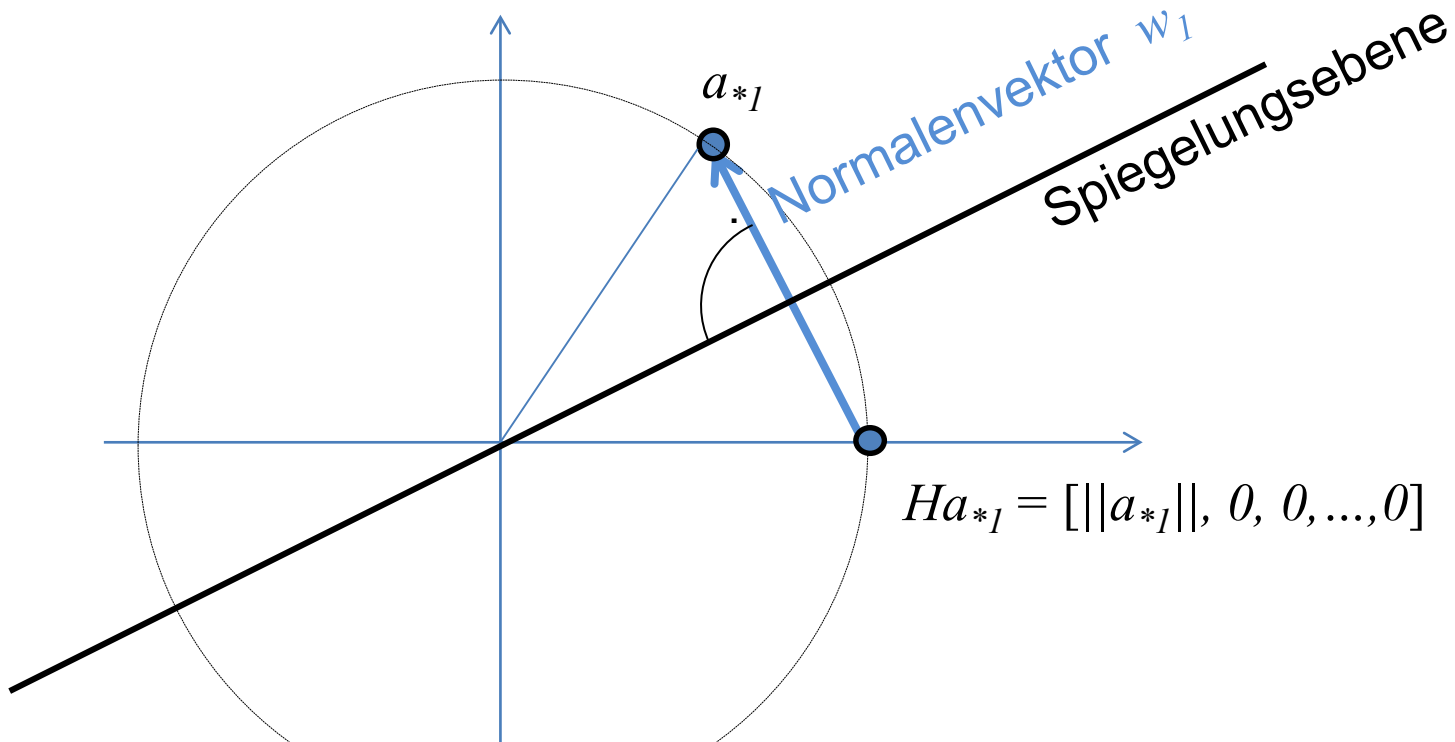
- Reihenfolge so, dass einmal geschaffene 0-en nicht wieder zerstört werden.
  - ▶  $R = J_{m,n^*} \dots J_{2,1} A$  (hierbei ist  $n^* = \min\{m-1, n\}$ )
  - ▶  $A = J_{2,1}^T \dots J_{m,n^*}^T R = Q \cdot R$
  - ▶ Das Produkt aller Rotationen  $J_{i,j}^T$  ist eine orthogonale  $m \times m$ -Matrix.
  - ▶ In den meisten Anwendungen muss dieses Produkt nicht explizit mit berechnet werden.

- Für die Lösung eines Gleichungssystems:  
Rechte Seite mit den Rotationen mit-transformieren.
- Aufwand: ca. 4× so hoch wie bei LR-Zerlegung.
- Verbesserung: rationale Givens-Rotationen
  - ▶ Rotationen werden durch eine Skalierung so modifiziert, dass die Diagonalelemente alle gleich 1 sind. Alle so erfolgten Skalierungen werden separat auf multipliziert: In dieser Variante kann man ca. die Hälfte des Rechenaufwandes sparen.
- Für die numerische Stabilität ist keine Pivotsuche nötig,
- QR ist stabiler und besser konditioniert als LR
 
$$\kappa_2(A) = \kappa_2(R) , \kappa_2(Q) = 1,$$
- Häufig für  $m > n$  angewandt, also für überbestimmte Gleichungssysteme.
  - ▶ siehe später: Ausgleichsrechnung.

- Eine QR-Zerlegung kann man auch mit Householder-Spiegelungen) anstelle von (Givens-) Rotationen erzeugen
- Dazu müssen  $n-1$  Householder-Spiegelungen

$$H_j = Id - \frac{2}{\mathbf{w}_j \circ \mathbf{w}_j} \cdot \mathbf{w}_j \cdot \mathbf{w}_j^T \quad H_j \mathbf{x} = \mathbf{x} - \frac{2 \cdot \mathbf{w}_j \circ \mathbf{x}}{\mathbf{w}_j \circ \mathbf{w}_j} \cdot \mathbf{w}_j$$

- konstruiert werden, die in der  $j$ -ten Spalte die erforderlichen Nullen erzeugen
- Man kann hier mit einer einzigen Transformation gleich die ganze Spalte eliminieren, indem man den Vektor  $\mathbf{w}_j$  „geeignet“ wählt



Householder Spiegelung mit  $w_1 = [a_{11} - \|a_{*1}\|, a_{21}, \dots, a_{n1}]^T$   
 bildet die erste Spalte  $a_{*1}$  auf  $[||a_{*1}||, 0, \dots, 0]^T$  ab.

- Vorgehen wie bei LR-Zerlegung:

- 1. Householder-Spiegelung:

$$H_1 A = \begin{bmatrix} * & * & \dots & * \\ 0 & * & \dots & * \\ \vdots & \vdots & \ddots & \vdots \\ 0 & * & \dots & * \end{bmatrix}$$

- 2. Householder-Spiegelung:

$$H_2 H_1 A = \begin{bmatrix} * & * & * & \dots & * \\ 0 & * & * & \dots & * \\ 0 & 0 & * & & * \\ 0 & 0 & \vdots & \ddots & \vdots \\ 0 & 0 & * & \dots & * \end{bmatrix}$$



- (n-1)-te Householder-Spiegelung:

$$H_{n-1} \cdots H_2 H_1 A = \begin{bmatrix} * & * & \cdots & * & * \\ 0 & * & \cdots & * & * \\ 0 & 0 & \ddots & \ddots & \vdots \\ 0 & 0 & \ddots & * & * \\ 0 & 0 & \cdots & 0 & * \end{bmatrix}$$

- Man setzt dann  $R = H_{n-1} \cdots H_2 H_1 A$  und

$$Q = (H_{n-1} \cdots H_2 H_1)^{-1} = H_1 H_2 \cdots H_{n-2} H_{n-1}$$

... und alles passt!

- Aufwand  $O(n^3)$  (etwa doppelt so hoch wie bei LR)

Born: 5 May 1904 in Rockford, Illinois, USA

Died: 4 July 1993 in Malibu, California, USA

Director of the Oak-Ridge National Laboratory, 1948-1969

*... Householder transformations are now routinely taught in courses throughout the world, as is the systematic use of norms in linear algebra, which he pioneered ...*



- Die oben beschriebenen Verfahren kann man auch für nicht quadratische Matrizen durchführen
- Ergebnis falls  $m < n$  , ( $m=3$ )

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1m} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2m} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3m} \end{bmatrix} = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & \cdots & r_{1m} \\ 0 & r_{22} & r_{23} & \cdots & r_{2m} \\ 0 & 0 & r_{33} & \cdots & r_{3m} \end{bmatrix}$$

- $Q = J_{2,1}^T \cdot J_{3,1}^T \cdot J_{3,2}^T$  bzw.  $Q = H_1 \cdot H_2$

- Ergebnis falls  $m > n$ , ( $n=3$ )

$$\begin{aligned}
 A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} \end{bmatrix} &= \begin{bmatrix} q_{11} & q_{12} & \cdots & q_{1n} \\ q_{21} & q_{22} & \cdots & q_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ q_{n1} & q_{n2} & \cdots & q_{nn} \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \\ \vdots & \vdots & 0 \\ 0 & 0 & 0 \end{bmatrix} = \\
 &= \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \\ \vdots & \vdots & \vdots \\ q_{n1} & q_{n2} & q_{n3} \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{bmatrix}
 \end{aligned}$$

$$\begin{aligned}
 Q &= J_{2,1}^T \cdots J_{n,1}^T \cdot \\
 &\quad J_{3,2}^T \cdots J_{n,2}^T \cdot J_{4,3}^T \cdots J_{n,3}^T
 \end{aligned}$$

$$Q = H_1 \cdot H_2 \cdot H_3$$

- in diesem Fall sind die Spalten  $q_{n+1}, \dots, q_m$  ohne Bedeutung (und werden meist weg gelassen)

- Problem: Gegeben linear unabhängige Vektoren  $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_k$
- Finde *orthonormale* Vektoren  $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_k$  so dass
  - ▶  $\text{span}\{\mathbf{u}_1\} = \text{span}\{\mathbf{b}_1\},$
  - ▶  $\text{span}\{\mathbf{u}_1, \mathbf{u}_2\} = \text{span}\{\mathbf{b}_1, \mathbf{b}_2\},$
  - ▶ ...
  - ▶  $\text{span}\{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_k\} = \text{span}\{\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_k\}$
- Standard Verfahren (LinAlg): Gram-Schmidt-Verfahren

$$\mathbf{u}_1 = \frac{\mathbf{b}_1}{\|\mathbf{b}_1\|}; \quad \tilde{\mathbf{u}}_2 = \mathbf{b}_2 - (\mathbf{b}_2 \circ \mathbf{u}_1) \cdot \mathbf{u}_1, \mathbf{u}_2 = \frac{\tilde{\mathbf{u}}_2}{\|\tilde{\mathbf{u}}_2\|}; \quad \dots$$

- Das Gram-Schmidt-Verfahren ist numerisch instabil
  - (das modifizierte Gram-Schmidt-Verfahren ist besser)



- Besser ist: QR-Zerlegung von

$$B = \begin{bmatrix} | & | & | & | \\ \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_k \\ | & | & | & | \end{bmatrix} = \begin{bmatrix} | & | & | & | \\ \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_k \\ | & | & | & | \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & \vdots & r_{1k} \\ 0 & r_{22} & & r_{2k} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & r_{kk} \end{bmatrix}$$

- Dann:  $\mathbf{b}_1 = r_{11} \mathbf{q}_1$ ,  
 $\mathbf{b}_2 = r_{12} \mathbf{q}_1 + r_{22} \mathbf{q}_2$ ,  
 ....  
 $\mathbf{b}_k = r_{1k} \mathbf{q}_1 + r_{2k} \mathbf{q}_2 + \dots + r_{kk} \mathbf{q}_k$ ,

- Dies zeigt: Die (ersten  $k$ )-Spalten  $Q$  sind die gesuchten orthonormalen Vektoren

- Wie kann man ausnützen, dass zwei Gleichungssysteme

$$\begin{array}{lcl} & Ax & = b \\ \text{„ähnliche“ Koeffizientenmatrizen haben?} & \hat{A}\hat{x} & = \hat{b} \end{array}$$

- Problem: ohne weiteres Wissen muss die Zerlegung (LR oder QR) mit  $O(n^3)$  Aufwand neu durchgeführt werden.
  - Möglichkeit 1: Eliminationsvorgang im Detail analysieren und prüfen, ob und welche Teile sich wieder verwenden lassen. (abhängig vom konkreten Problem und vom Geschick des Algorithmenentwicklers)
  - Für spezielle „Störungen“ nutzen spezieller Formeln: Hier als Beispiel die Sherman-Morrison-Woodbury-Formel für Rang-1-Störungen.

- Ist  $A$  nicht singulär,  $u$  und  $v$  Vektoren, so dass  $v^T A^{-1} u \neq -1$ , dann gilt:

$$(A + uv^T) \quad \text{ist nicht singulär}$$

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^T A^{-1}}{(1 + v^T A^{-1}u)}$$

(Sherman-Morrison-Woodbury-Formel)

- $A + uv^T$  ist eine Rang-1-Modifikation der Matrix  $A$ .  
Spezialfälle davon sind:
  - Modifikation eines einzelnen Elements von  $A$
  - Modifikation einer einzelnen Spalte von  $A$
  - Modifikation einer einzelnen Zeile von  $A$
- Das Herleiten der Formel ist eine nette Übungsaufgabe.
- Die numerische Stabilität hängt von den Daten ab, muss also in jedem Einzelfall sicher gestellt werden.



Die Sherman-Morrison-Woodbury-Formel darf man nicht naiv anwenden:

Die Inverse von  $A$  wird natürlich **nicht** explizit berechnet, sondern nur ihre LR-Zerlegung  $A=LR$  (die wir ja schon kennen, weil wir ja annehmen dass  $Ax=b$  schon gelöst wurde):

$$(A + uv^T)x = b$$

$$x = (A + uv^T)^{-1}b = A^{-1}b - \frac{A^{-1}uv^T A^{-1}b}{1 + v^T A^{-1}u}$$

Also:

$Ap = b$  durch einmal Vorwärts-Rückwärtssubstitution

$Aq = u$  durch noch einmal Vorwärts-Rückwärtssubstitution

und dann ergibt sich die Lösung  $x = p - \frac{qv^T p}{1 + v^T q}$

- Lineare Gleichungssysteme kann man direkt oder iterativ lösen
- Direkte Verfahren liefern nach endlich vielen Schritten das exakte Ergebnis (theoretisch, wenn Arithmetik exakt),
- sie basieren auf der Faktorisierung von  $A$ 
  - $A = A_1 A_2$  dann ist  $Ax = b$  äquivalent zu  $A_1 y = b$  und  $A_2 x = y$
- LR-Zerlegung (mit Gauss-Scherung): Teilprobleme lösen durch Vorwärts-/ Rückwärts-Substitution
  - nur Pivotsuche garantiert (in den meisten Fällen) Stabilität
- QR-Zerlegung (mit Givens-Rot. oder Householder Sp.)
- Aufwand: LR- und QR-Zerlegung :  $O(n^3)$   
Lösung der Teilprobleme  $O(n^2)$
- Spezialvorlesungen aus der Mathematik, oder im WS  
ANLA: „Algorithmen der numerischen Linearen Algebra“