

# Algorithmik kontinuierlicher Systeme

## *Direkte Verfahren für Lineare Gleichungssysteme*



- Bitte Zweiergruppen für die Theorieübungen bilden

## Was ist Software Engineering?



Idee der  
Auftraggeber



Formulierung  
im Vertrag



Entwurf



Programm



Nach der  
Installation



Wunsch der  
Betroffenen

- Einfaches Beispiel (2 Gleichungen, 2 Unbekannte)

$$\begin{aligned} 2x_1 - 3x_2 &= 1 \\ 4x_1 + 2x_2 &= 10 \end{aligned} \quad \begin{bmatrix} 2 & -3 \\ 4 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 10 \end{bmatrix}$$

- Matrix-Schreibweise:

$$Ax = b$$

- Bei  $n$  Gleichungen und  $m$  Unbekannten ist
  - ▶  $A$  eine  $n \times m$  Matrix ( $n$  Zeilen und  $m$  Spalten)
  - ▶  $x$  ein  $m$ -Vektor
  - ▶  $b$  ein  $n$ -Vektor

- Wir betrachten (in diesem Kapitel) den Standardfall:  
 $n$  Gleichungen und  $n$  Unbekannte.

$$\begin{array}{cccccc} a_{11}x_1 + a_{12}x_2 + & \cdots & + a_{1n}x_n & = & b_1 \\ a_{21}x_1 + a_{22}x_2 + & \cdots & + a_{2n}x_n & = & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1}x_1 + a_{n2}x_2 + & \cdots & + a_{nn}x_n & = & b_n \end{array}$$

- Matrix-Vektor-Schreibweise:

$$Ax = b \quad \text{wobei} \quad A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}; x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}; b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

- Es gibt (sehr viele) Algorithmen zum Lösen von LGS
- Man unterscheidet zwei Klassen:
  - ▶ **direkte Verfahren:**  
dabei wird die Lösung nach endlich vielen Schritten erreicht (sofern exakt gerechnet wird!)
  - ▶ **iterative Verfahren:**  
man startet mit einem (geschätzten) Wert (z.B. 0-Vektor) und verbessert diesen iterativ (ggf. sehr viele Iterationen)
- Mit den iterativen Verfahren erhält man nur Näherungswerte, aber ....
- Tendenz:
 

vollbesetzte (kleine) Matrizen	→ direktes Verf.
dünnbesetzte (große) Matrizen	→ iteratives Verf.

Ein Beispiel:

$$\begin{array}{rrcrcl} x_1 & + & 2x_2 & - & x_3 & = & 3 \\ 3x_1 & + & 8x_2 & - & 2x_3 & = & 11 \\ -2x_1 & - & 2x_2 & + & 6x_3 & = & -3 \end{array}$$

---


$$\begin{array}{rrcrcl} x_1 & + & 2x_2 & - & x_3 & = & 3 & I \\ & & 2x_2 & + & x_3 & = & 2 & II - \frac{3}{1} \cdot I \\ & & 2x_2 & + & 4x_3 & = & 3 & III - \frac{-2}{1} \cdot I \end{array}$$

---


$$\begin{array}{rrcrcl} x_1 & + & 2x_2 & - & x_3 & = & 3 & I \\ & & 2x_2 & + & x_3 & = & 2 & II \\ & & & & 3x_3 & = & 1 & III - \frac{2}{2} \cdot II \end{array}$$

Ein Beispiel (cont'd):

$$\begin{array}{rclclcl} x_1 & + & 2x_2 & - & x_3 & = & 3 \\ & & 2x_2 & + & x_3 & = & 2 \\ & & & & 3x_3 & = & 1 \end{array}$$

Lösen durch Rückwärtssubstitution:

$$x_3 = \frac{1}{3}$$

$$x_2 = (2 - x_3) / 2 = \frac{5}{6}$$

$$x_1 = (3 - 2x_2 + x_3) = \frac{5}{3}$$

- Gegeben:
  - ▶ Eine (voll besetzte), nichtsinguläre Matrix  $A$
  - ▶ und eine rechte Seite  $b$
- Aufgabe: Löse die Gleichung  $Ax = b$
- Strategie: Faktorisiere  $A = A_1 A_2$  derart  
dass sich die Teilprobleme
  - ▶  $A_1 y = d$  und  $A_2 z = e$
 leicht lösen lassen
- Zweistufiges Lösen von  $Ax = b$  :  
**Erst  $A_1 y = b$ , dann  $A_2 x = y \Rightarrow Ax = A_1(A_2 x) = b$**



Die wichtigsten Verfahren:

- Die LR-Zerlegung:  $A = L R$   
wobei  $L$  eine linke (untere) Dreiecksmatrix und  $R$  eine rechte (obere) Dreiecksmatrix ist.  
(im Englischen: LU-decomposition)

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & r_{nn} \end{bmatrix}$$

- Die QR-Zerlegung:  $A = Q R$   
wobei  $Q$  eine orthogonale Matrix und  $R$  eine rechte (obere) Dreiecksmatrix ist.  
(im Englischen: QR-decomposition)

Eine  $n \times n$  Matrix  $Q$  heißt **orthogonal**, falls eine der folgenden (gleichwertigen) Bedingungen erfüllt ist:

- $Q^T Q = Id$  (= Einheitsmatrix)
- $Q Q^T = Id$
- Spalten oder Zeilen von  $Q$  bilden eine Orthonormalbasis
- Die Abbildung  $Q$  ist winkel- und längentreu,
- $Q$  erhält das Skalarprodukt:  $Qx \circ Qy = x \circ y$

Zur Erinnerung (Notation) : Skalarprodukt  $x \circ y = \sum_{i=1}^n x_i y_i$

Euklidische Norm (aka Länge):

$$||x|| = \sqrt{x \circ x}$$

Winkel:

$$\cos(\angle(x, y)) = \frac{x \circ y}{||x|| ||y||}$$

Wh.: Eine  $n \times n$  Matrix  $Q$  heißt **orthogonal**, falls  $Q^T Q = Id$

Speziell für  $n=2$  :

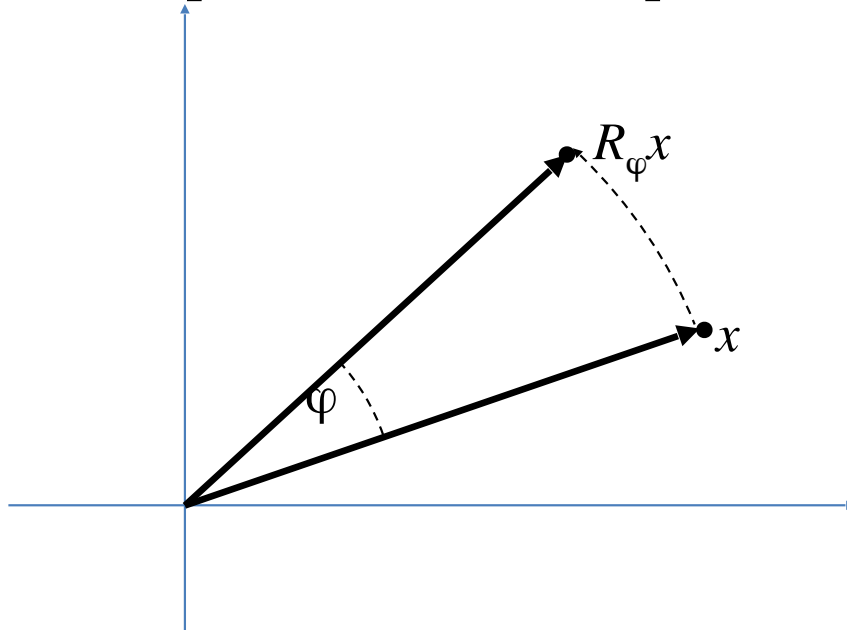
orthogonale  $2 \times 2$ -Matrizen sind

**Drehungen**

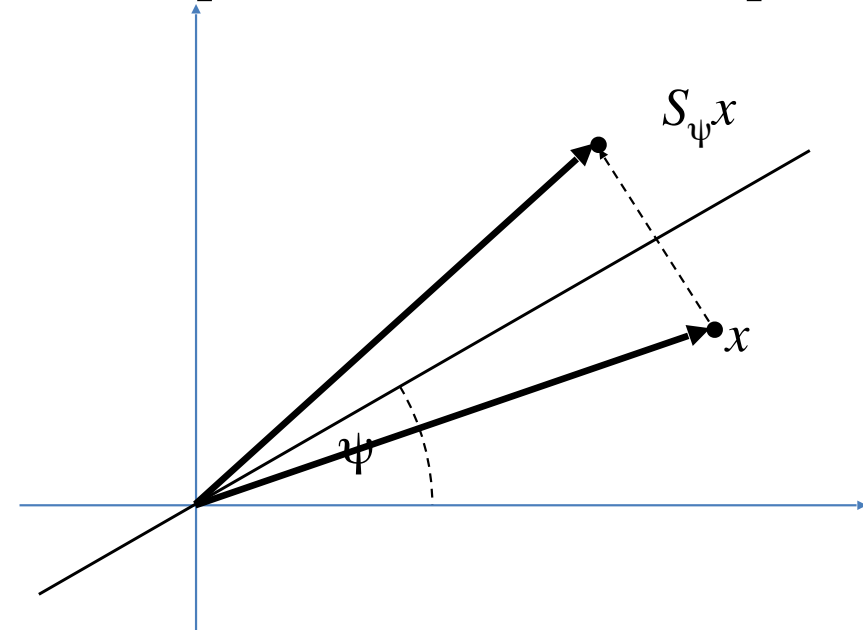
oder

**Spiegelungen**

$$R_\varphi = \begin{bmatrix} \cos(\varphi) & -\sin(\varphi) \\ \sin(\varphi) & \cos(\varphi) \end{bmatrix}$$



$$S_\psi = \begin{bmatrix} \cos(2\psi) & \sin(2\psi) \\ \sin(2\psi) & -\cos(2\psi) \end{bmatrix}$$



- Gleichungssysteme mit Dreiecksstruktur:  $Lx = c$

$$L = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix}; c = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}; \begin{array}{l} x_1 = c_1/l_{11}; \\ x_2 = (c_2 - l_{21}x_1)/l_{22} \\ \vdots \\ x_n = \left( c_n - \sum_{j=1}^{n-1} l_{nj}c_j \right) / l_{nn} \end{array}$$

- Lösung durch Vorwärtssubstitution:

```
for j=1..n
    x[j] = c[j]
    for i=1..j-1
        x[j] -= L[j,i]*x[i]
    x[j] = x[j]/L[j,j]
```

Kosten:

$n$  Divisionen,  
 $n(n-1)/2$  Multiplikationen  
 $n(n-1)/2$  Additionen

gesamt:  $O(n^2)$

- Gleichungssysteme mit Dreiecksstruktur:  $Rx = c$

$$R = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & r_{nn} \end{bmatrix} c = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}; \quad \begin{aligned} x_n &= c_n / r_{nn}; \\ x_{n-1} &= (c_{n-1} - r_{n-1,n}x_n) / r_{n-1,n-1} \\ &\vdots \\ x_1 &= \left( c_1 - \sum_{j=2}^n r_{1j}c_j \right) / r_{11} \end{aligned}$$

- Lösung durch Rückwärtssubstitution:

```
for j=n..1
    x[j] = c[j];
    for i=j+1..n
        x[j] -= R[j,i]*x[i]
    x[j] = x[j]/R[j,j] ;
```

Kosten:

$n$  Divisionen,  
 $n(n-1)/2$  Multiplikationen  
 $n(n-1)/2$  Additionen

gesamt:  $O(n^2)$

- Erst Gleichungssystem  $Qy = d$  mit orthonormaler Matrix  $Q$ :

Lösung:  $y = Q^T d$

Aufwand:  $n^2$  Multiplikation,  $n(n-1)$  Additionen

- Dann Gleichungssysteme  $Rx = y$  mit rechter oberer Dreiecksmatrix  $R$ . Das löst man analog wie oben

Rückwärtssubstitution:

$$x_n = y_n / r_{n,n}$$

$$x_{n-1} = (y_{n-1} - r_{n-1,n} x_n) / r_{n-1,n-1}$$

.....

- Aufwand wie bei der Vorwärtssubstitution:  $O(n^2)$

1. Wie kann man die Matrix  $A$  faktorisieren?

$$A = LR$$

bzw.

$$A = QR$$

Welche Algorithmen?

Welcher Aufwand ist erforderlich?

(Anzahl der arithmetischen Operationen)

- Der klassische Gauss-Eliminations-Algorithmus beruht auf Transformationen mittels *Zeilenoperationen* (und Permutationen=Zeilenvertauschungen).
- Dabei ist es das Ziel, die Matrix sukzessive (mittels Zeilenoperationen) in Dreiecksgestalt zu transformieren

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{n1} & \cdots & & a_{nn} \end{bmatrix} \rightarrow \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & * & & * \\ \vdots & & \ddots & \vdots \\ 0 & * & \cdots & * \end{bmatrix} \rightarrow \cdots \rightarrow \begin{bmatrix} * & * & \cdots & * \\ 0 & * & & * \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & * \end{bmatrix}$$

- Zeilenoperationen ergeben sich durch Multiplikation mit geeigneten Matrizen, sog. *Gauß-Scherung*
- Die Beschreibung mit Matrizen ist für die Analyse
- die Implementierung erfolgt anders, durch (Unter-)Programme mit denen die Matrizen bearbeitet werden.



[illegible]

- $N_{ij}(\alpha)$  A addiert zur  $i$ -ten Zeile von A das  $\alpha$ -fache der  $j$ -ten.
- $(N_{ij}(\alpha))^{-1} = N_{ij}(-\alpha)$

$$\begin{bmatrix}
 1 & & & & \\
 & \ddots & & & \\
 & & 1 & & \\
 & & & \ddots & \\
 & & & & 1
 \end{bmatrix}
 \begin{bmatrix}
 0 & & & & \\
 \vdots & & & & \\
 0 & & & & \\
 & & & & \\
 & & & &
 \end{bmatrix}
 \begin{bmatrix}
 0 & & & & \\
 & & & & \\
 & & & & \\
 & & & & \\
 & & & &
 \end{bmatrix}
 \begin{bmatrix}
 r_{11} & \dots & * & * & * & \dots & * \\
 0 & \ddots & & \vdots & \vdots & & \vdots \\
 \vdots & \ddots & r_{j-1,j-1} & * & \vdots & & \vdots \\
 0 & & 0 & r_{jj} & * & \dots & * \\
 \vdots & & \vdots & 0 & \vdots & & \vdots \\
 0 & \dots & 0 & \tilde{a}_{ij} & * & \dots & * \\
 \vdots & & \vdots & * & \vdots & & \vdots \\
 \vdots & & \vdots & \vdots & \vdots & & \vdots \\
 0 & \dots & 0 & * & * & \dots & *
 \end{bmatrix}
 =
 \begin{bmatrix}
 r_{11} & \dots & * & * & * & \dots & * \\
 0 & \ddots & & \vdots & \vdots & & \vdots \\
 \vdots & \ddots & r_{j-1,j-1} & * & \vdots & & \vdots \\
 0 & & 0 & r_{jj} & * & \dots & * \\
 \vdots & & \vdots & 0 & \vdots & & \vdots \\
 0 & \dots & 0 & 0 & \tilde{*} & \dots & \tilde{*} \\
 \vdots & & \vdots & * & \vdots & & \vdots \\
 \vdots & & \vdots & \vdots & \vdots & & \vdots \\
 0 & \dots & 0 & * & * & \dots & *
 \end{bmatrix}$$

$$N_{ij}(\alpha) \cdot \tilde{A} = \tilde{A}$$

dabei ist  $\alpha = -\frac{\tilde{a}_{ij}}{r_{jj}}$  und  $r_{jj}$  das Pivotelement

## Verwendung der Zeilenoperationen

$$N_j(\alpha) = N_{j+1}(\alpha_{j+1}) \dots N_n(\alpha_n)$$

Mit  $N_1(\alpha)$  kann die erste Spalte unter die Diagonalen auf 0 gebracht werden.

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ -\frac{a_{21}}{a_{11}} & 1 & & 0 \\ \vdots & & \ddots & \vdots \\ -\frac{a_{n1}}{a_{11}} & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & * & & * \\ \vdots & & \ddots & \vdots \\ 0 & * & \cdots & * \end{bmatrix}$$

Vorgehen spaltenweise von links nach rechts, so dass einmal erzeugte Nullen nicht mehr ruiniert werden.

Die rechte Seite des Gleichungssystems muss mittransformiert werden.

Dieser Algorithmus ist völlig äquivalent zu dem Gauss-Eliminationsverfahren aus der Mathematik-Vorlesung.

- Wenn man sich also einfach nur die Eliminationsfaktoren der Gauss-Elimination in einer unteren Dreiecksmatrix  $L$  merkt, generiert man eine Faktorisierung der Matrix  $A = L R$
- Geschickte Programmierer verwenden hierzu genau den durch Elimination frei werdenden Speicherplatz in der Originalmatrix  $A$

$$\begin{aligned}
 A &= N_1(-\alpha_1)N_1(\alpha_1)A && \text{(Elimination der 1. Spalte)} \\
 &= N_1(-\alpha_1)N_2(-\alpha_2)N_2(\alpha_2)N_1(\alpha_1)A && \text{(Elimination der 2. Spalte)} \\
 &= N_1(-\alpha_1)N_2(-\alpha_2)N_3(-\alpha_3)N_3(\alpha_3)N_1(\alpha_2)N_1(\alpha_1)A && \text{(3. Spalte)} \\
 &= \vdots \\
 &= L \cdot R
 \end{aligned}$$

$$L = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \ddots & \vdots \\ \vdots & \cdots & \ddots & 0 \\ l_{n1} & \cdots & l_{n,n-1} & 1 \end{bmatrix} \quad R = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & & r_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & r_{nn} \end{bmatrix}$$