

3. Auflösung Linearer Gleichungssysteme (GS)

▣ Dtr. zunächst lineares GS in der allgemeinen Form

$$(1) \text{ Ges } x \in \mathbb{R}^n: Ax = b \text{ in } \mathbb{R}^n$$

oder ausgeschrieben

$$(1) \text{ Ges. } x = (x_1, x_2, \dots, x_n)^T \in \mathbb{R}^n:$$

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2$$

$$\vdots$$

$$\vdots$$

$$\vdots$$

$$\vdots$$

$$a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n$$

und setzen voraus, dass die Systemmatrix $A = [a_{ij}]_{i,j=1..n}$ regulär ist, d.h.

$$(2) \exists A^{-1} \quad (\text{gdw. } |A| := \det A \neq 0)$$

gdw. Spaltenvekt. lin. unabh. gdw. Zeilenvekt. lin. unabh.

und folglich das GS (1) die eindeutige Lsg. $u = A^{-1}b$ hat.

▣ Bsp.: Lineares GS $K_n u_n = f_n$ in $\mathbb{R}^{n=N_n}$, das bei der FE-Diskretisierung von RWA entsteht, siehe Kap. 2!

Kap. 5. Optimierungsprobleme und deren numerische Lsg.

▣ Kapitel 3 hat folgende Abs.:

3.1. Direkte Verfahren: Folien 32 und 33

3.2. Iterative Verfahren: Folien 34 - 37

▣ Iterative Verfahren zur Lsg. nichtlin. GS der Art
 Ges. $x \in \mathbb{R}^n: F(x) = 0$ in \mathbb{R}^n
 $u_n \in \mathbb{R}^{N_n}: K_n(u_n) - f_n = 0$ in $\mathbb{R}^{n=N_n}$
 werden im Kap. 5 "Optimierung" behandelt!

3.1. Direkte Verfahren

3.1.1. Das Gaußsche Eliminationsverfahren und seine Interpretation als LU-Zerlegung

■ Eliminationschritt: Überführung in Dreiecksgestalt

• Bez.: $A^{(0)} = [a_{ij}^{(0)}] := A = [a_{ij}]_{i,j=1,\dots,n}$

$$b^{(0)} = [b_i^{(0)}] := b = [b_i]_{i=1,\dots,n}$$

Oberer Index (k) bedeutet k-ter Eliminationschritt

• k=1: Im 1. Schritt eliminieren wir x_1 aus der 2. bis n-ten Gleichung mit geeigneten Vielfachen der 1. Gleichung:

$$A^{(1)} x = b^{(1)}$$

mit

$$A^{(1)} = \left[\begin{array}{c|ccc} u_{11} & u_{12} & \dots & u_{1n} \\ \hline 0 & a_{22}^{(1)} & \dots & a_{2n}^{(1)} \\ \vdots & \vdots & \text{Rest-} & \vdots \\ 0 & a_{n2} & \dots & a_{nn} \end{array} \right], \quad b^{(1)} = \begin{bmatrix} c_1 \\ b_2^{(1)} \\ \vdots \\ b_n^{(1)} \end{bmatrix}$$

Pivotelement (green arrow pointing to $a_{22}^{(1)}$)

wobei

$$u_{1j} = a_{1j}^{(0)} = a_{1j}, \quad j = 1, 2, \dots, n$$

$$l_{i1} = a_{i1}^{(0)} / a_{11}^{(0)}, \quad i = 2, \dots, n$$

(3)⁽¹⁾

$$a_{ij}^{(1)} := a_{ij}^{(0)} - l_{i1} u_{1j}, \quad i, j = 2, \dots, n$$

und

$$c_1 = b_1^{(0)} = b_1$$

$$b_i^{(1)} = b_i^{(0)} - l_{i1} c_1, \quad i = 2, 3, \dots, n = \overline{2:n}$$

- $k=2$: Im 2. Schritt eliminieren wir x_2 aus 3. bis n -ter Gleichung analog:

$$A^{(2)} x = b^{(2)}$$

mit

$$(3)^{(2)} \quad A^{(2)} = \begin{bmatrix} u_{11} & u_{12} & \dots & u_{13} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{23} & \dots & u_{2n} \\ \hline 0 & 0 & \dots & a_{33}^{(2)} & \dots & a_{3n}^{(2)} \\ 0 & 0 & \dots & a_{43}^{(2)} & \dots & a_{4n}^{(2)} \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \dots & a_{n3}^{(2)} & \dots & a_{nn}^{(2)} \end{bmatrix}, \quad b^{(2)} = \begin{bmatrix} c_1 \\ c_2 \\ b_3^{(2)} \\ b_4^{(2)} \\ \vdots \\ b_n^{(2)} \end{bmatrix}$$

wobei

$$u_{2j} = a_{2j}^{(1)}, \quad j = 2, 3, \dots, n;$$

$$l_{i2} = a_{i2}^{(1)} / a_{22}^{(1)}, \quad i = 3, 4, \dots, n;$$

$$a_{ij}^{(2)} = a_{ij}^{(1)} - l_{i2} u_{2j}, \quad i, j = 3, 4, \dots, n;$$

und

$$c_2 = b_2^{(1)};$$

$$b_i^{(2)} = b_i^{(1)} - l_{i2} c_2, \quad i = 3, 4, \dots, n = \overline{3, n}.$$

etc.

- Nach insgesamt $(n-1)$ Schritten erhalten wir schließlich ein GS in der Form

$$(3) \quad U x = c$$

mit der oberen (upper) Dreiecksmatrix

$$U = \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & u_{nn} \end{bmatrix} \quad \text{und der RS } c = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$$

wobei $u_{nn} = a_{nn}^{(n-1)}$.

■ Rückwärts einsetzen!

Das gestaffelte GS (3) läßt sich nun leicht von unten nach oben auflösen:

$$x_n = \frac{1}{u_{nn}} \cdot c_n$$

$$x_i = \frac{1}{u_{ii}} \left(c_i - \sum_{j=i+1}^n u_{ij} x_j \right)$$

$$i = n-1, n-2, \dots, 1 = n-1 \quad (-1) \quad 1$$

■ Durchführbarkeit: Pivotelement $\neq 0$

$$u_{kk} = a_{kk}^{(k-1)} \neq 0 \quad \forall k = 1, 2, \dots, n$$

Um $a_{kk}^{(k-1)} = 0$ zu vermeiden, macht man
 → Pivotsuche in der Restmatrix

- Totalpivotsuche & Spalten- und Zeilentausch
 $(i, j) \in \{k, \dots, n\} : |a_{ij}^{(k-1)}| \geq |a_{kk}^{(k-1)}| \quad \forall i, j = \overline{k, n}$
- Spaltenpivotsuche & Spaltentausch
- Zeilenpivotsuche & Zeilentausch

■ Hauptaufwand (Operationen der Art $z := x + ay$)

1. Zur Berechnung von $\tilde{U} = [\]$

$$(n-1)^2 + (n-2)^2 + \dots + 2^2 + 1^2 \approx \frac{n^3}{3} = O(n^3)$$

2. Zur Berechnung von c (& Vorwärtseinsetzen)

$$(n-1) + (n-2) + \dots + 2 + 1 \approx \frac{n^2}{2} = O(n^2)$$

3. Zur Berechnung von x (& Rückwärtseinsetzen)

$$(n-1) + (n-2) + \dots + 2 + 1 \approx \frac{n^2}{2} = O(n^2)$$

BW·n

BW·n

BW·n

- Abspeicherung: der Zwischenergebnisse nach $(k-1)$ Schritten läßt sich in der Form

$$\begin{bmatrix}
 u_{11} & \dots & u_{1k} & \dots & u_{1n} \\
 L_{21} & u_{22} & \dots & u_{2k} & \dots & u_{2n} \\
 \vdots & \ddots & & \vdots & & \vdots \\
 L_{k-1,1} & & u_{k-1,k-1} & u_{k-1,k} & \dots & u_{k-1,n} \\
 L_{k1} & \dots & L_{kk-1} & a_{kk}^{(k-1)} & & a_{kn}^{(k-1)} \\
 \vdots & & \vdots & \vdots & & \vdots \\
 L_{n1} & \dots & L_{nk-1} & a_{nk}^{(k-1)} & & a_{nn}^{(k-1)}
 \end{bmatrix}$$

mit ungefähr n^2 Speicherplätzen realisieren, falls A und b überschrieben werden können.

Interpretation als LU-Zerlegung:

Ü 3.1 Man zeige, daß die Eliminations-
schritte $(3)^{(1)}, (3)^{(2)}, \dots, (3)^{(n-1)}$
äquivalent zur LU-Zerlegung
von A sind, d.h.

$$A = LU = \begin{bmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & & \ddots & \\ l_{n1} & l_{n2} & \dots & l_{nn} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & u_{nn} \end{bmatrix}$$

Lower Upper

mit den im Gauß-Algorithmus
erzeugten l_{ij} und u_{ij} !

Damit ist die Auflösung des GS

$$\begin{array}{l} A \mathbf{x} = \mathbf{b} \\ \underbrace{LU}_{=C} \mathbf{x} = \mathbf{b} \end{array}$$

äquivalent zu den folgenden Schritten:

1. Faktorisierung: $A = LU$ mit $Q = O(n^3)$ ops
2. Vorwärtseinsehen: $L\mathbf{c} = \mathbf{b}$ mit $Q = O(n^2)$ ops
3. Rückwärtseinsehen: $U\mathbf{x} = \mathbf{c}$ mit $Q = O(n^2)$ ops

■ Implementierung:

Folie 32 f

$$Ax = b \quad \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{nn} & a_{n2} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \quad A^{(0)}x = b^{(0)}$$

1. Schritt:

$$u_{1j} = a_{1j}^{(0)} = a_{1j} \quad j = \overline{1, n}$$

$$j = \overline{1, n}$$

$$l_{in} = a_{in}^{(0)} / a_{1n}^{(0)} \quad i = \overline{2, n}$$

$$i = \overline{2, n}$$

$$a_{ij}^{(1)} = a_{ij}^{(0)} - l_{i1} u_{1j} \quad i, j = \overline{2, n}$$

$$i, j = \overline{2, n}$$

$$c_1 = b_1^{(0)} = b_1$$

$$b_i^{(1)} = b_i^{(0)} - l_{i1} c_1$$

1 → k

Initialisieren: $A^{(0)} = [a_{ij}^{(0)}] = A = [a_{ij}]$
 $b^{(0)} = [b_i^{(0)}] = b = [b_i]$

Zerlegen / Vorwärtsrechnen:

FOR $k = 1$ STEP 1 UNTIL $n-1$ DO

FOR $i = k+1$ STEP 1 UNTIL n DO

$$l_{ik} := a_{ik}^{(k-1)} / a_{kk}^{(k-1)}$$

$$b_i^{(k)} := b_i^{(k-1)} - l_{ik} b_k^{(k-1)}$$

FOR $j = k+1$ STEP 1 UNTIL n DO

$$a_{ij}^{(k)} := a_{ij}^{(k-1)} - l_{ik} a_{kj}^{(k-1)}$$

ENDFOR

ENDFOR

ENDFOR

⋮
⋮

- ILU-Zerlegung \rightarrow Prädiktionierung
 \rightarrow Iterationsverfahren

= Unvollständige (**I**ncomplete) **L**U-Zerlegung:

Man spricht von einer ILU-Zerlegung von A , falls die Koeffizienten l_{ij} und u_{ij} nach den Formeln

$$(3)^{(1)} \quad \dots \quad (3)^{(n-1)}$$

nur

$$\forall (i,j) \in \mathcal{M} \stackrel{\text{Maske 2.6.}}{=} \mathcal{M}_{\text{NNE}} := \{(i,j) : a_{ij} \neq 0\}$$

berechnet und sonst einfach 0 gesetzt werden. Dann erhalten wir eine Zerlegung der Art

$$(4) \quad A = \tilde{L} \tilde{U} + R, \quad \text{d. h. } C = \tilde{L} \tilde{U} \neq A$$

Rest

Insbesondere gilt aber

$$R = 0 \quad \text{für } \mathcal{M} := \{(i,j) : i,j = \overline{1,n}\}$$

Zu was sind ILU-Zerlegungen gut ?

C = "guter" Prädiktionierer
für Iterationsverfahren (z. B. Abs. 3.2)

Man hofft: $\kappa(C^{-1}A) \ll \kappa(A) !!$
Kondition

3.1.2. Zerlegung spezieller Matrizen

■ Bandmatrizen:

$$A = \begin{bmatrix} \text{---} & & & & & \\ & \text{---} & & & & \\ & & \text{---} & & & \\ & & & \text{---} & & \\ & & & & \text{---} & \\ & & & & & \text{---} \end{bmatrix} = LU = \begin{bmatrix} 1 & & & & & \\ & 2 & & & & \\ & & \dots & & & \\ & & & & & \\ & & & & 1 & \\ & & & & & 1 \end{bmatrix} \begin{bmatrix} \text{---} & & & & & \\ & \text{---} & & & & \\ & & \text{---} & & & \\ & & & \text{---} & & \\ & & & & \text{---} & \\ & & & & & \text{---} \end{bmatrix}$$

Ü 3.2

Man zeige, daß

$$l_{ij} = 0 \text{ und } u_{ij} = 0 \quad \forall |i-j| \geq m,$$

falls $a_{ij} = 0 \quad \forall |i-j| \geq m = BW$

Resultate:

1. Bei der LU-Zerlegung bleibt die BW von A in L und U erhalten, aber eventuell innerhalb des Bandes von A vorhandene Nullen können zerstört werden (\Downarrow "Fill-in")!
2. Benötigte Arithmetik:
 - Faktorisierung (Zerlegung): $O \approx BW^2 \cdot n = m^2 \cdot n$,
 - Vor- und Rückwärtseinsetzen: $O \approx BW \cdot n = m \cdot n$,
3. Speicherplatzbedarf: $M \approx BW \cdot n = m \cdot n$.

■ Profilmatrizen:

$$A = \begin{bmatrix} \text{[Matrix with profile]} \end{bmatrix} = LU = \begin{bmatrix} \text{[L with profile]} \end{bmatrix} \begin{bmatrix} \text{[U with profile]} \end{bmatrix}$$

$$A = \begin{bmatrix} \text{[Matrix with profile]} \end{bmatrix} = \tilde{U} \tilde{L} = \begin{bmatrix} \text{[L-tilde with profile]} \end{bmatrix} \begin{bmatrix} \text{[U-tilde with profile]} \end{bmatrix}$$

Profil überträgt sich entsprechend auf Faktoren.!

■ Symmetrische Matrizen ($A=A^T$): LDL^T

$\Rightarrow LDL^T$ - Zerlegung

$$A = \underbrace{L}_{=U} \underbrace{D}_{\tilde{D}} \underbrace{L^T}_{\tilde{U}^T} = \tilde{U} \tilde{D} \tilde{U}^T$$

$\tilde{U} = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & -1 \end{bmatrix}$

mit $D = \begin{bmatrix} \diagdown \end{bmatrix}$
 $\tilde{D} = \begin{bmatrix} \diagdown \end{bmatrix}$
 Diagonalmatrizen

■ SPD - Matrizen ($A=A^T > 0$):

\Rightarrow Cholesky - Zerlegung

$$A = U^T U = \tilde{U} \tilde{U}^T \text{ mit } U = \begin{bmatrix} \text{[Matrix]} \end{bmatrix}, \tilde{U} = \begin{bmatrix} \text{[Matrix]} \end{bmatrix}$$

Cholesky - Zerlegung ($A=S^T S = R R^T$)
 siehe Skriptum S. 148-154.

FE-Gleichungssystem $K_h u_h = f_h$:

Aus den Eigenschaften von K_h

- $n = O(h^{-d})$
- $NNE = O(n) = O(h^{-d})$
- $BW = m = O(h^{-(d-1)})$
- K_h spd, falls $a(\cdot, \cdot)$ sym. und positiv
- $\kappa(K_h) = O(h^{-2})$ für PDg 1. 2. Ordnung

ergeben sich sofort die folgende Charakteristika der direkten Verfahren:

1. $M = \text{Memorg} \approx BW \cdot n = O(h^{-2d+1})$
2. $Q = \text{Arithmetik} \approx BW^2 n = O(h^{-3d+2})$
3. $s = \text{Verlust an gültigen Ziffern} = (g \kappa(K_h))$

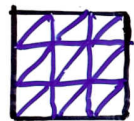
Daraus ergibt sich folgende Situation:

- $d=1$: $M = O(h^{-1})$, $Q = O(h^{-1})$ \rightarrow (asym.) optimal!
- $d=2,3$: Starkes Anwachsen von M und Q für $h \rightarrow 0$:
 \rightarrow optimal wäre: $M, Q = O(n) = O(h^{-d})$!
- $d=1,2,3$: Verlust an gültigen Ziffern ist unabh. von d !

Praktisch bedeutet das für 100 MFlop Rechner:

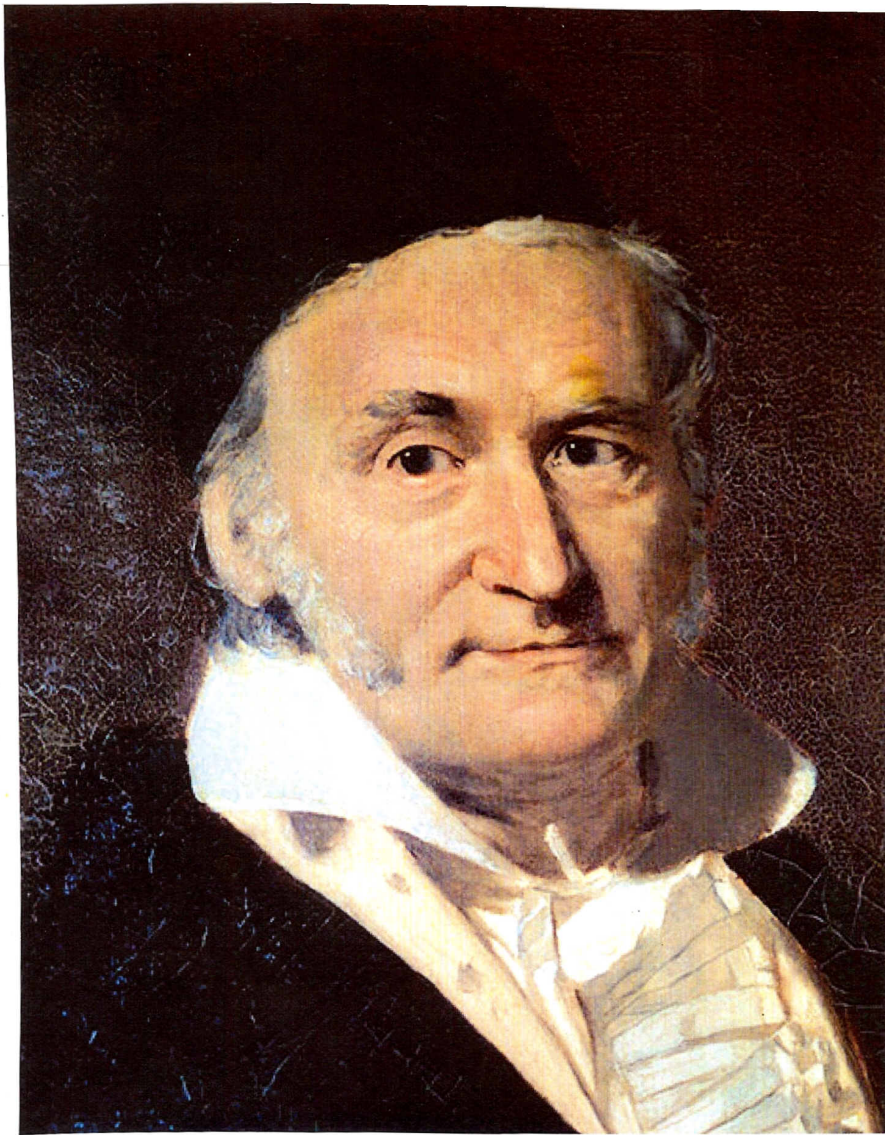
am Modellbsp.: $-\Delta u = f$ in $\Omega = (0,1)^2$

$u = g$ auf $\partial\Omega$



$\downarrow h$

h^{-1}	$d=2: 2D$			$d=3: 3D$		
	CPU-Zeit	M	s	CPU-Zeit	M	s
50	30 ms	488 KB	3	65 min	1192 MB	3
100	0.5 s	3,6 MB	4	5,8 Tage	38146 MB	4
500	5.2 min	476 MB	5	3,1 Jahre	1220 GB	5
1000	1.8 h	3816 MB	6			6
2000	22,2 h	30528 MB	7			7



„fast jeden Abend mache ich eine neue Auflage des Tableau, wo immer leicht nachzuhelfen ist. Bei der Einformigkeit des Messungsgeschäfts gibt dies immer eine angenehme Unterhaltung; man sieht daran auch immer gleich, ob etwas Zweifelhaftes eingeschlichen ist, was noch wünschenswert bleibt usw. Ich empfehle Ihnen diesen Modus zur Nachahmung. Schwerlich werden Sie je wieder direct eliminiren, wenigstens nicht, wenn Sie mehr als zwei Unbekannte haben. Das indirecte Verfahren läßt sich halb im Schlafe ausführen oder man kann während desselben an andere Dinge denken.“

C. F. GAUSS in [2]

3.2. Iterative Verfahren

■ Idee:

- Geg. $x^{(0)} \in \mathbb{R}^n$ - Startnäherung
- Erzeugen (wie?) sukzessiv Folge von Näherungen

$$x^{(1)}, x^{(2)}, \dots, x^{(k)} \xrightarrow[k \rightarrow \infty]{\heartsuit} x \in \mathbb{R}^n : Ax = b \quad (1)$$

■ Fragen:

- Konstruktionsprinzipien
- Konvergenzanalyse: $x^{(k)} \xrightarrow[k \rightarrow \infty]{} x$! ?
- Konvergenzgeschwindigkeit (KG) und Fehlerabschätzungen:

z. B. q -lineare KG, d. h. $\exists q \in (0, 1)$ - Konvergenzrate:

$$\|x - x^{(k)}\| \leq q \|x - x^{(k-1)}\| \leq \dots \leq q^k \|x - x^{(0)}\|$$

r -lineare KG, d. h. $\exists q \in (0, 1)$ und $c = \text{const} > 0$:

$$\|x - x^{(k)}\| \leq c q^k$$

- Praktisch: Konvergenztest, z. B. Defekttest:
Man stoppt die Iteration, falls

$$(5) \quad \|d^{(k)}\| \leq \varepsilon \|d^{(0)}\| \stackrel{\text{z. B.}}{:=} \varepsilon \sqrt{\sum_{i=1}^n (d_i^{(0)})^2}$$

Euklidische Norm

mit dem Defekt $d^{(k)} = b - Ax^{(k)}$ und vorgeg. rel. Genauigkeit $\varepsilon = 10^{-t} \in (0, 1)$.

- In welcher Norm $\|\cdot\|$ soll man den Fehler $z^{(k)} = x - x^{(k)}$ messen?

z. B. gilt für den Fehler $z^{(k)} = x - x^{(k)}$

$$\|z^{(k)}\|_2^2 := \|z^{(k)}\|_{ATA}^2 := (A^T A z^{(k)}, z^{(k)}) = \|A(x - x^{(k)})\|_2^2 = \|d^{(k)}\|_2^2$$

wobei $\|\cdot\| := (\cdot, \cdot)^{0.5}$ - Euklidische Norm

3.2.1. Klassische Iterationverfahren

■ Jacobi-Verfahren (GSV = Gesamtschrittverfahren):

- Idee: $a_{i1}x_1 + \dots + \underline{a_{ii}x_i} + \dots + a_{in}x_n = b_i$

$$\Rightarrow x_i = \frac{1}{a_{ii}} \left[b_i - \sum_{\substack{j=1 \\ j \neq i}}^n a_{ij}x_j \right] \text{ (Exakt!)}!$$

- Algorithmus:

Startnäherung: $x^{(0)} = (x_1^{(0)}, \dots, x_n^{(0)})^T \in \mathbb{R}^n$ geg. ..

Iteration: $k=0, 1, \dots, k_{\text{stop}}$ (Defekttest (5)!) ..

$$x^{(k+1)} = (x_1^{(k+1)}, \dots, x_n^{(k+1)})^T \in \mathbb{R}^n:$$

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^n a_{ij}x_j^{(k)} \right), \quad i=1, 2, \dots, n$$

(6)

- Nachteil: bei FE-Systemen $K_n u_n = f_n$:

Langsame Konvergenz (?), aber in gedämpfter Version (6) gute Glättungseigenschaften \rightarrow **Multigrid-Methoden!**

■ Gauß-Seidel-Verfahren (ESV = Einzelschrittverfahren):

- Idee: Verwende die bereits berechneten neuen Komp.!

- Algorithmus:

Startnäherung: $x^{(0)} \in \mathbb{R}^n$ geg.

Iteration: $k=0, 1, \dots, k_{\text{stop}}$ (Defekttest (5)!) ..

$$x^{(k+1)} = (x_1^{(k+1)}, x_2^{(k+1)}, \dots, x_n^{(k+1)}):$$

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[b_i - \sum_{s=1}^{i-1} a_{is}x_s^{(k+1)} - \sum_{s=i+1}^n a_{is}x_s^{(k)} \right]$$

$$i=1, 2, \dots, n$$

(7)

$$\sum_{i=1}^n a_{ii} = \sum_{i=1}^n a_{ii} = 0$$

- Nachteile bei FE-GS: lang. Konv., aber gute Glättung \rightarrow **MGM!**

■ SOR-Verfahren (= ω -Gauß-Seidel-Verfahren):

● **SOR** = **S**uccessive **O**ver**R**elaxation

● Algorithmus:

Startnäherung: $x^{(0)} \in \mathbb{R}^n$ geg.

Iteration: $k = 0, 1, \dots, k_{\text{stop}}$ (Defekttest (5)!))

(8)

$$x^{(k+1)} = (x_1^{(k+1)}, \dots, x_n^{(k+1)})^T \in \mathbb{R}^n:$$

$i = 1, 2, \dots, n:$

$$\bullet \tilde{x}_i^{(k+1)} = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij} x_j^{(k)} \right]$$

$$\bullet x_i^{(k+1)} = x_i^{(k)} + \omega (\tilde{x}_i^{(k+1)} - x_i^{(k)})$$

● Bemerkung:

- 1) $\omega = 1 \rightarrow \text{SOR} = \text{ESV}$, d.h. (8) = (7)
- 2) Durch "geschickte" Wahl des Relaxationsparameter $\omega \in (1, 2)$, z. B. $\omega \approx 1,5$ (Over) kann oft schneller Konvergenz als beim Jacobi- bzw. Gauß-Seidel-Verfahren erreicht werden !!!

■ Richardson-Verfahren ($C=I$): $x^{(0)}$ geg.

$$(9)_{C=I} \quad \frac{x^{(k+1)} - x^{(k)}}{\tau} + Ax^{(k)} = b, \quad k=0,1,\dots$$

■ $Ax = b \iff C^{-1}Ax = C^{-1}b$

■ Präkonditioniertes Richardson-Verfahren:

- Iterationsvorschrift: $x^{(0)}$ geg.

$$(9) \quad C \frac{x^{(k+1)} - x^{(k)}}{\tau} + Ax^{(k)} = b, \quad k=0,1,\dots,$$

wobei der Präkonditionierer C (reg.):

1. $C \approx A$, z.B. $C=A, \tau=1 \Rightarrow x^{(1)} \approx x$ - Lsg. (1)

Genauer: $\text{Konditionszahl}(C^{-1}A) \ll \kappa(A)$!

2. GS $Cw = d$ soll "schnell" (d.h. für FE-GS mit $O(n)$ ops) auflösbar sein !

- Algorithmus:

Startnäherung: $x^{(0)} \in \mathbb{R}^n$ geg.

Iteration: $k=0,1,\dots, k_{\text{stop}}$ (Konvergenztest)

$$d^{(k)} = b - Ax^{(k)} \quad (\text{Defekt berechnen})$$

$$C w^{(k)} = d^{(k)} \quad (\text{Präkond.-System lösen})$$

$$x^{(k+1)} = x^{(k)} + \tau w^{(k)} \quad (\text{Korrektur})$$

- Konvergenztest z.B. Defekttest

$$\|d^{(k)}\| \leq \varepsilon \|d^{(0)}\| \quad \text{mit } \varepsilon = 10^{-5} < 1.$$

■ Beispiele: $C = ?$

1. $C = I$: klassisches Richardson-Verfahren (9) $C=I$
2. $C = D := \text{diag } A = \begin{bmatrix} \diagdown \end{bmatrix}$: τ -Jacobi-Verfahren
 $\Downarrow \tau = 1$: Jacobi-Verfahren (6) !
3. $C = (L + \frac{1}{\omega} D)$ - SOR-Präkonditionierer,
 wobei $A = L + D + U = \begin{bmatrix} \blacktriangleright \end{bmatrix} + \begin{bmatrix} \diagdown \end{bmatrix} + \begin{bmatrix} \blacktriangleleft \end{bmatrix}$;
 Für $\tau = 1 \Downarrow$ SOR (8) bzw. ESV (7) für $\omega = 1$!
4. $C = (L + \frac{1}{\omega} D) D^{-1} (U + \frac{1}{\omega} D)$ - SSOR-Präkond.,
 $\tau = \frac{2}{\omega} - 1 \Downarrow$ SSOR falls $A = A^T$ ($\Downarrow U = L^T$)
5. $C = \tilde{L} \tilde{U}$ = ILU-Zerlegung von A :
 siehe Folie 32 g
6. Moderne Präkonditioner
 $C = \text{MG, BPX, AMG, AMLI}$
 $\text{DD, FETI, FETI-DP, BCCD}$
 etc.

■ Fehleranalysis für präkond. RICHARDSON:

- Aus (9) $x^{(k+1)} = x^{(k)} - \tau C^{-1} A x^{(k)} + \tau C^{-1} b$
 und (1) $Ax = b$
 folgt sofort das Fehlerschema
- $$(10) \quad z^{(k+1)} = x - x^{(k+1)} = x - x^{(k)} - \tau C^{-1} A (x - x^{(k)})$$
- $$= (I - \tau C^{-1} A) z^{(k)}$$

- Damit gilt die Fehlerabschätzung

$$(11) \quad \|z^{(k+1)}\| \leq \underbrace{\|I - \tau G^{-1}A\|}_{=: q < 1} \|z^{(k)}\|$$

? Norm?

!! hängt von τ und G ab!!
Konvergenzfaktor

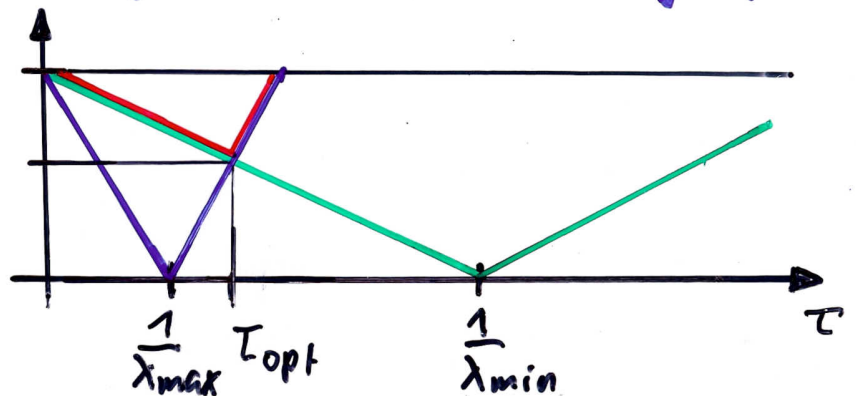
- Falls $A = A^T > 0$ (SPD), dann gilt für die Normen: $\|\cdot\| = \|\cdot\|_{\mathbb{R}^n}$, $\|\cdot\|_A$, $\|\cdot\|_C$, $\|\cdot\|_{A^T C^{-1}A}$:

$$\|I - \tau G^{-1}A\| = \max\{|1 - \tau \lambda_{\max}|, |1 - \tau \lambda_{\min}|\} =: q < 1$$

EWP: $A\varphi = \lambda G\varphi$

$$q_{\text{opt}} = \frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}} \approx \frac{\kappa - 1}{\kappa + 1}$$

mit



$$\kappa(C^{-1}A) = \text{cond}_2(C^{-1}A) = \frac{\lambda_{\max}(C^{-1}A)}{\lambda_{\min}(C^{-1}A)}$$

$\lambda_{\min}/\lambda_{\max} = \text{min/max EW des verallg. EWP } A\varphi = \lambda G\varphi$

Bem.: $\|z^{(k)}\|_{A^T C^{-1}A}^2 = (G^{-1}A z^{(k)}, A z^{(k)}) = (C^{-1}d^{(k)}, d^{(k)})$

$$= (w^{(k)}, d^{(k)}) - \text{berechenbar!}$$

Konvergenztest: $(w^{(k)}, d^{(k)}) \leq \varepsilon^2 (w^{(0)}, d^{(0)})$

- In der Praxis nimmt man anstelle von präkond. Richardson-Verfahren präkond. Krylov-Raum-Verf. z. B. PCG-Verfahren für SPD GS (siehe Abs. 4.2.2!).

WHY

3.2.2. GG für GS mit spd Matrizen

- Btr. nun GS (1) $Ax = b$ mit spd Matrix A ,
d.h. **S**ymmetrisch und **P**ositiv **D**efinit

$$(12) \quad A = A^T \quad \text{und} \quad (Ax, x) > 0 \quad \forall x \in \mathbb{R}^n: x \neq 0.$$

Dann sind offenbar folgende Formulierungen
äquivalent (vgl. auch Satz 3.1)

$$(1) \quad \text{Ges. } x \in \mathbb{R}^n:$$

$$\bullet \quad Ax = b \quad \times$$

$$\bullet \quad (Ax, y) = (b, y) \quad \forall y \in \mathbb{R}^n$$

$$(13) \quad \bullet \quad E(x) = \min_{y \in \mathbb{R}^n} E(y) \quad \times$$

mit dem Energiefunktional

$$E(y) = \frac{1}{2} (Ay, y) - (b, y).$$

wobei $(\cdot, \cdot) := (\cdot, \cdot)_{\mathbb{R}^n}$ das Euklidische
Skalarprodukt in \mathbb{R}^n ist, d.h.

$$(x, y) = x^T y = \sum_{i=1}^n x_i y_i$$

- Richtung des steilsten Abstiegs im Pkt. $y \in \mathbb{R}^n$:

$$(14) \quad -\nabla E(y) := - \left[\frac{\partial E(y)}{\partial y_i} \right]_{i=1, \dots, n} = b - Ay =: d$$

• Idee für Verfahren des steilsten Abstiegs (= Gradientenverfahren)

- $x^{(0)}$ geg. Startnäherung;

$$d^{(0)} := b - Ax^{(0)}$$

$$s^{(0)} := d^{(0)}$$

$$x^{(1)} := x^{(0)} + \alpha^{(1)} s^{(0)}$$

$$\alpha^{(1)} : E(x^{(0)} + \alpha^{(1)} s^{(0)}) = \min_{\alpha} E(x^{(0)} + \alpha s^{(0)})$$

$$\frac{d}{d\alpha} E(x^{(0)} + \alpha s^{(0)}) = (Ax^{(0)}, s^{(0)}) - (b, s^{(0)}) + \alpha (As^{(0)}, s^{(0)})$$

$$\stackrel{!}{=} 0$$

$$\alpha^{(1)} = \frac{(d^{(0)}, s^{(0)})}{(As^{(0)}, s^{(0)})}$$

- Die Richtung des steilsten Abstiegs von $E(\cdot)$ im Punkt $x^{(1)}$ läßt sich nun leicht rekursiv berechnen

$$d^{(1)} = b - Ax^{(1)}$$

$$= b - A(x^{(0)} + \alpha^{(1)} s^{(0)})$$

$$= b - Ax^{(0)} - \alpha^{(1)} As^{(0)}$$

$$= d^{(0)} - \alpha^{(1)} As^{(0)}$$

- Damit erhalten wir den folgenden Algorithmus für das Gradientenverfahren:

Startschritt:

$$x^{(0)} \in \mathbb{R}^n \text{ geg. Startnäherung}$$

$$d^{(0)} := b - Ax^{(0)}$$

$$s^{(0)} := d^{(0)}$$

Iteration: $k=0, 1, \dots, k_{\text{stop}}$

Konvergenztest

$$\|d^{(k)}\| \leq \varepsilon \|d^{(0)}\|$$

$$\alpha^{(k+1)} := \frac{(d^{(k)}, s^{(k)})}{(As^{(k)}, s^{(k)})}$$

$$x^{(k+1)} := x^{(k)} + \alpha^{(k+1)} s^{(k)} \quad (\text{Iterierende})$$

$$d^{(k+1)} := d^{(k)} - \alpha^{(k+1)} As^{(k)} \quad (\text{Defekt})$$

$$s^{(k+1)} := d^{(k+1)} \quad (\text{Suchrichtung})$$

- Mögliche Verbesserungen:

1) Präkonditionierung: $Ax = b \mapsto G^{-1}Ax = G^{-1}b$

$$w^{(k+1)} = G^{-1} * d^{(k+1)} \quad (\text{präkonditionierter Defekt})$$

\Rightarrow Präkonditionierte Gradientenverfahren

2) Verwendung konjugierter Suchrichtungen:

$$(As^{(k+1)}, s^{(k)}) \stackrel{!}{=} 0$$

$$s^{(k+1)} = d^{(k+1)} + \beta^{(k+1)} s^{(k)}$$

\rightarrow **PGG-Verfahren!**

Preconditioned Conjugate Gradient (PCG) Method

PCG - Algorithms:

Startschritte:

$$x^{(0)} \in \mathbb{R}^n - \text{geg. Startnäher., z.B. } x^{(0)} = C^{-1}b$$

$$d^{(0)} := b - Ax^{(0)}$$

$$w^{(0)} := C^{-1} \cdot d^{(0)}$$

$$s^{(0)} := w^{(0)}$$

Iteration: $k=0, 1, \dots$ (Konvergenztest)

$$\text{Test: } (w^{(k)}, d^{(k)}) \leq \varepsilon^2 (w^{(0)}, d^{(0)})$$

$$\|z^{(k)}\|_{AC^{-1}A} \leq \varepsilon \|z^{(0)}\|_{AC^{-1}A} \quad \downarrow \text{(norm)}$$

$$\alpha^{(k+1)} := \frac{(d^{(k)}, s^{(k)})}{(As^{(k)}, s^{(k)})} \stackrel{\uparrow \text{(norm)}}{=} \frac{(d^{(k)}, w^{(k)})}{(As^{(k)}, s^{(k)})}$$

$$x^{(k+1)} := x^{(k)} + \alpha^{(k+1)} s^{(k)}$$

$$d^{(k+1)} := d^{(k)} - \alpha^{(k+1)} As^{(k)}$$

$$w^{(k+1)} := C^{-1} \cdot d^{(k+1)}$$

$$\beta^{(k+1)} := \frac{(Aw^{(k+1)}, s^{(k)})}{(As^{(k)}, s^{(k)})} \stackrel{\downarrow \text{(norm)}}{=} \frac{(w^{(k+1)}, d^{(k+1)})}{(w^{(k)}, d^{(k)})}$$

$$s^{(k+1)} := w^{(k+1)} + \beta^{(k+1)} s^{(k)} \perp s^{(k)}$$

• Fehlerabschätzung:

$$z^{(k)} = x - x^{(k)} = \text{Fehler}$$

$$(14) \|z^{(k)}\|_A \leq \frac{2q^k}{1+q^{2k}} \|z^{(0)}\|_A$$

mit

$$\|z\|_A^2 := (Az, z)$$

$$q = \frac{\sqrt{\kappa(C^{-1}A)} - 1}{\sqrt{\kappa(C^{-1}A)} + 1} < 1.$$

$$\kappa(C^{-1}A) := \frac{\lambda_{\max}(C^{-1}A)}{\lambda_{\min}(C^{-1}A)} = \frac{\text{max. EW}}{\text{min EW}}$$

$$A\varphi = \lambda C\varphi$$

■ A nicht spd \rightarrow CG-like-Methoden

\Downarrow

\rightarrow GMRES

CGS

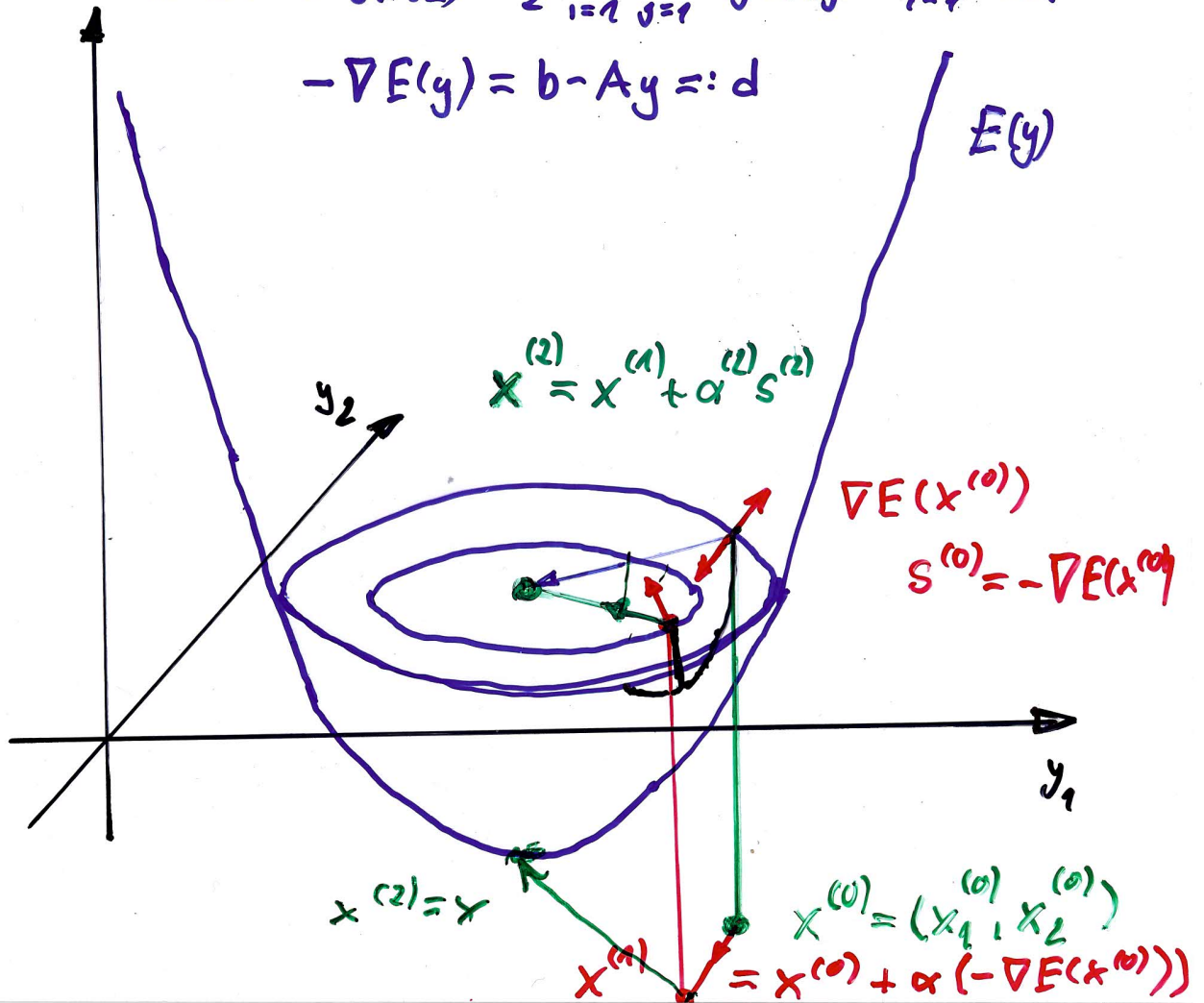
\rightarrow BiCG-Stab

(siehe Literatur)

$$E(x) = \min_{y \in \mathbb{R}^n} E(y) = \min_{y \in \mathbb{R}^n} \frac{1}{2} (Ay, y) - (b, y)$$

$$n=2: E(y_1, y_2) = \frac{1}{2} \sum_{i=1}^2 \sum_{j=1}^2 a_{ij} y_i y_j - \sum_{i=1}^2 b_i y_i$$

$$-\nabla E(y) = b - Ay =: d$$



3.2.3. Mehrgitterverfahren

MGM = Multi-Grid-Methods

zur Lösung von FE-GS $K_h u_h = f_h$

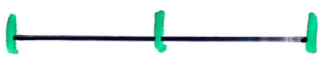
Idee: Klassische IV wie GS oder ω -Jacobi ($\omega < 1$)

glätten Fehler $u_h - u_h^{(k)}$ und folglich auch

Defekt $d_h^{(k)} = f_h - K_h u_h^{(k)} = K_h (u_h - u_h^{(k)})$

Löse Defektgleichung auf gröberen Gitter

$H = 2h$



$$K_H w_H^{(k)} = d_H^{(k)} := I_h^H d_h^{(k)}$$

Neue Näherung:

$$u_h^{new} = u_h^{(k)} + I_h^h w_H^{(k)}$$

TGM

Zweigitterverfahren:

$$u_h^{(old)} \longrightarrow u_h^{(new)}$$

$$u_h^{(0)} := u_h^{(old)}$$

1-5

FOR $j := 1$ STEP 1 UNTIL K DO

$$u_h^{(j)} := GS(u_h^{(j-1)}) := u_h^{(j-1)} + (L_h + D_h)^{-1} (f_h - K_h u_h^{(j-1)})$$

$$d_H^{(k)} := I_h^H d_h^{(k)} = I_h^H (f_h - K_h u_h^{(k)})$$

$$K_H w_H^{(k)} = d_H^{(k)}$$

← MGM = rekursiv!

$$w_H^{(k)} = I_H^h w_H^{(k)}$$

$$u_h^{(new)} = u_h^{(k)} + w_H^{(k)}$$

$$K_H = I_H^H K_h I_H^h$$

siehe Skriptum: S. 161-164

Buch : Abs. 5.2.4

S. 276-283