Lineare Gleichungssysteme: direkte und iterative Löser

4. Vorlesung

170.021 Numerische Methoden 1

(170.026 Numerische Methoden I)

Alexander Steinicke

Montanuniversität Leoben

9. November 2023

Lineare Gleichungssysteme: direkte und iterative Löser

1. Lösbarkeit, Fehlerempfindlichkeit
   - Auffrischung: Matrixalgebra
   - Lösbarkeit: Fallunterscheidungen
   - MATLAB und lineare Systeme
   - Fehlerempfindlichkeit, Konditionszahl
2. Große Gleichungssysteme, spärlich besetzte Matrizen
3. Die einfachsten iterativen Verfahren
   - Jacobi-, Gauß-Seidel-, SOR-Verfahren
4. Direkte Verfahren
   - Dreiecksmatrizen
   - Gauß-Elimination, Pivotisierung
   - Stufenform
5. Überbestimmte Systeme
   - Kleinste Quadrate, Normalengleichungen
   - Beispiel: Lineare Datenmodelle

Gliederung 4. Vorlesung

Lineares Gleichungssystem in n Gleichungen und Unbekannten

$$\begin{array}{rcl} a_{11}\,x_1 + a_{12}\,x_2 + \ldots + a_{1n}\,x_n &=& b_1 \\ a_{21}\,x_1 + a_{22}\,x_2 + \ldots + a_{2n}\,x_n &=& b_2 \\ \vdots \qquad\qquad\qquad\qquad \vdots &\quad& \vdots \\ a_{n1}\,x_1 + a_{n2}\,x_2 + \ldots + a_{nn}\,x_n &=& b_n \end{array}$$

In Matrixschreibweise: $A\mathbf{x} = \mathbf{b}$.

mit $n \times n$-Matrix A und Vektoren $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$.

Matrix-Vektor-Algebra ist für das Thema „Gleichungssysteme“ unumgänglich. Frischen Sie Ihre Kenntnisse auf!

Einheitsmatrix, Inverse und Transponierte

Diese Begriffe sollten Sie kennen...

Die 3×3-Einheitsmatrix. Wichtige Eigenschaft: $A \cdot I = I \cdot A = A$

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Ein recht simples Beispiel:
Das Magische Quadrat der Ordnung 3

$$M = \begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix}$$

Die Inverse von M. Wichtige Eigenschaft: $M \cdot M^{-1} = M^{-1} \cdot M = I$

$$M^{-1} = \frac{1}{360} \begin{bmatrix} 53 & -52 & 23 \\ -22 & 8 & 38 \\ -7 & 68 & -37 \end{bmatrix}$$

Die transponierte Matrix M^T

$$M^T = \begin{bmatrix} 8 & 3 & 4 \\ 1 & 5 & 9 \\ 6 & 7 & 2 \end{bmatrix}$$

MATLAB-Befehle: `eye(3)`, `M=magic(3)`, `inv(M)`, `M'`

$A \cdot B = C$

Element c_{ij} ist Skalarprodukt der i-ten Zeile von A mit der j-ten Spalte von B.

$A : (\ell \times m)$-*Matrix*
$B : (m \times n)$-*Matrix*
$C : (\ell \times m) \cdot (m \times n) \rightarrow (\ell \times n)$-*Matrix*

- Spaltenanzahl von A und Zeilenanzahl von B müssen übereinstimmen!
- Im Allgemeinen nicht kommutativ: $A \cdot B \neq B \cdot A$!

Matrizenrechnung in MATLAB

`A'`	Matrixtransposition A^T
`A+B, A-B`	Matrixaddition, -subtraktion
`A.*B, A./B`	elementweise Multiplikation, Division
`A*B`	Matrixmultiplikation
`A/B`	schräge Schreibung: „Division“ $A \cdot B^{-1}$
`A\B`	noch schräger: „Division“ $A^{-1} \cdot B$

Wichtiger Befehl:

`x=A\b` berechnet Lösung des Gleichungssystems

$$A \cdot \mathbf{x} = \mathbf{b}$$

(MATLAB berechnet `A\b` mit Gauß-Elimination und nicht in der Form $\mathbf{x} = A^{-1} \cdot \mathbf{b}$.)

Gleichungssysteme: Aufgabenstellung

Eine der wichtigsten Aufgaben bei technisch-wissenschaftlichen Berechnungen ist das Lösen linearer Gleichungssysteme

$A\mathbf{x} = \mathbf{b}$	gegeben: $A, \mathbf{b}$. gesucht: Vektor $\mathbf{x}$
$AX = B$	B und X können auch Matrizen sein
$XA = B$	kommt selten vor

- Oft ist A quadratisch, man sucht eine eindeutige Lösung.
- Ist A eine $m \times n$-Matrix mit $m > n$ (mehr Gleichungen als Unbekannte): überbestimmtes System, man sucht die „am wenigsten falsche Lösung“ (Methode der kleinsten Fehlerquadrate).
- Ist $m < n$: unterbestimmtes System. Gesucht: ein- oder mehrparametrige Lösungsschar.

Fallunterscheidungen bei Gleichungssystemen

Vorübung. $a, x, b \in \mathbb{R}$: eine Gleichung, eine Unbekannte

Die lineare Gleichung

$$ax = b \quad a, b \text{ gegeben}, \quad x \text{ gesucht}$$

- hat eine eindeutige Lösung genau dann, wenn $a \neq 0$
- unendlich viele Lösungen genau dann, wenn $a = b = 0$
- keine Lösung genau dann, wenn $a = 0, b \neq 0$

Auch bei Systemen linearer Gleichungen treten genau diese drei Fälle auf!

Fallunterscheidungen bei Gleichungssystemen

Vorübung. $a, x, b \in \mathbb{R}$: eine Gleichung, eine Unbekannte

Die lineare Gleichung

$$ax = b \quad a, b \text{ gegeben}, \quad x \text{ gesucht}$$

- hat eine eindeutige Lösung genau dann, wenn $a \neq 0$
- unendlich viele Lösungen genau dann, wenn $a = b = 0$
- keine Lösung genau dann, wenn $a = 0, b \neq 0$

Auch bei Systemen linearer Gleichungen treten genau diese drei Fälle auf!

Fallunterscheidungen bei Gleichungssystemen

Vorübung. $a, x, b \in \mathbb{R}$: eine Gleichung, eine Unbekannte

Die lineare Gleichung

$$ax = b \quad a, b \text{ gegeben}, \quad x \text{ gesucht}$$

- hat eine eindeutige Lösung genau dann, wenn $a \neq 0$
- unendlich viele Lösungen genau dann, wenn $a = b = 0$
- keine Lösung genau dann, wenn $a = 0, b \neq 0$

Auch bei Systemen linearer Gleichungen treten genau diese drei Fälle auf!

Fallunterscheidungen bei Gleichungssystemen

Vorübung. $a, x, b \in \mathbb{R}$: eine Gleichung, eine Unbekannte

Die lineare Gleichung

$$ax = b \quad a, b \text{ gegeben}, \quad x \text{ gesucht}$$

- hat eine eindeutige Lösung genau dann, wenn $a \neq 0$
- unendlich viele Lösungen genau dann, wenn $a = b = 0$
- keine Lösung genau dann, wenn $a = 0, b \neq 0$

Auch bei Systemen linearer Gleichungen treten genau diese drei Fälle auf!

Zwei Gleichungen, zwei Unbekannte

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\cdot\begin{bmatrix}x\\ y\end{bmatrix}=\begin{bmatrix}1\\ 2\end{bmatrix}$$ eindeutige Lösung $\begin{array}{rcl} x & = & 0\\ y & = & 1/2\end{array}$

$$\begin{bmatrix}1 & 2\\ 3 & 6\end{bmatrix}\cdot\begin{bmatrix}x\\ y\end{bmatrix}=\begin{bmatrix}1\\ 3\end{bmatrix}$$ ∞ viele Lösungen

$$\begin{array}{rcccccc} x & = & 0, & 1, & 2, & 3, & \dots\\ y & = & 1/2, & 0 & -1/2, & -1, & \dots\end{array}$$

$$\begin{bmatrix}1 & 2\\ 3 & 6\end{bmatrix}\cdot\begin{bmatrix}x\\ y\end{bmatrix}=\begin{bmatrix}1\\ 2\end{bmatrix}$$ keine Lösung

Die Determinante determiniert

- Im ersten Fall ist $\det A \neq 0$,
- in den anderen beiden Fällen ist $\det A = 0$

Zwei Gleichungen, zwei Unbekannte

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$ eindeutige Lösung $\begin{array}{rcl} x & = & 0 \\ y & = & 1/2 \end{array}$

$$\begin{bmatrix} 1 & 2 \\ 3 & 6 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$ ∞ viele Lösungen

$\begin{array}{rcccccc} x & = & 0, & 1, & 2, & 3, & \dots \\ y & = & 1/2, & 0 & -1/2, & -1, & \dots \end{array}$

$$\begin{bmatrix} 1 & 2 \\ 3 & 6 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$ keine Lösung

Die Determinante determiniert

- Im ersten Fall ist $\det A \neq 0$,
- in den anderen beiden Fällen ist $\det A = 0$

Systeme $A\mathbf{x} = \mathbf{b}$ mit quadratischer Matrix

Ein lineares Gleichungssystem mit quadratischer $n \times n$-Matrix ist

- genau dann eindeutig lösbar, wenn rang $A = n$ ($\det(A) \neq 0$).
  Andernfalls gibt es
- keine eindeutige (wenn rang $A =$ rang$[A, \mathbf{b}]$), oder
- überhaupt keine Lösung (wenn rang $A \neq$ rang$[A, \mathbf{b}]$).

Falls das Konzept von Matrixrang und Rang der erweiterten Matrix zu schwere Kost ist: Die Folie „Stufenform und Lösbarkeit“ erklärt später noch einmal die Fallunterscheidungen zur Lösbarkeit.

Eindeutig lösbare Systeme mit quadratischer Matrix

Der wichtige Standard-Fall. Dazu gibt es verschiedene Aussagen.

Der Fall $\det(A) \neq 0$ ist (algebraisch) gleichbedeutend mit

- Alle n Zeilen von A sind linear unabhängig.
- Alle n Spalten von A sind linear unabhängig.
- $\operatorname{rang}(A) = n$
- A hat vollen Rang. Wissen Sie, was der Rang einer Matrix ist?
- A ist nichtsingulär
- A ist regulär

MATLAB: `A\b` findet die eindeutige Lösung, falls $\det A \neq 0$

Wie MATLAB Systeme $A\mathbf{x} = \mathbf{b}$ löst

Befehle \ und pinv bei quadratischen Matrizen

`x=A\b` berechnet mit einem Eliminationsverfahren

- für nichtsinguläre Matrizen die eindeutige Lösung.
- für singuläre Matrizen, wenn eine Lösungsschar existiert, manchmal eine Lösung mit möglichst vielen Null-Komponenten,
- für singuläre Matrizen, wenn keine Lösung existiert, meistens Unsinn.

`x=pinv(A)*b` berechnet mit Hilfe der Pseudoinversen

- für nichtsinguläre Matrizen mit unnötig hohem Aufwand die eindeutige Lösung.
- für singuläre Matrizen, wenn eine Lösungsschar existiert, eine Lösung mit minimaler 2-Norm.
- für singuläre Matrizen, wenn keine Lösung existiert, die „am wenigsten falsche Lösung“: jene, für die der Rest-Vektor $\mathbf{r} = A\mathbf{x} - \mathbf{b}$ am kleinsten ist. Kleinste-Quadrate-Lösung $\|A\mathbf{x} - \mathbf{b}\|_2 \to \min!$

Quadratische Systeme $A\mathbf{x} = \mathbf{b}$

MATLAB-Befehle `rank` und `rref`

`rank(A)` und `rank([A,b])`

melden MATLABs Meinung zum Rang von Matrix und erweiterter Matrix.

Rang-Bestimmung ist numerisch heikel

Linear abhängige Matrixzeilen können durch beliebig kleine Änderungen in den Koeffizienten (z. B. Rundungsfehler) unabhängig werden.
Rangbestimmung ist daher numerisch heikel und von einer Fehlertoleranz abhängig. MATLABs `rank`-Funktion verwendet Singulärwertzerlegung, ein aufwändiges, aber verlässliches Verfahren.

`rref([A,b])`

berechnet mit einem speziellen Eliminationsverfahren (Gauß-Jordan) eine reduzierte Treppenform des Systems (reduced row echelon form), aus der alle Lösungsfälle abgelesen werden können.

Beispiele zu `rref`

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 6 \end{bmatrix}, b = \begin{bmatrix} 23 \\ 61 \\ 114 \end{bmatrix}, \quad \texttt{rref([A,b])} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 8 \\ 0 & 0 & 1 & 15 \end{bmatrix}$$

Eindeutige Lösung; links die Einheitsmatrix, rechts der Lösungsvektor.

$$A = \begin{bmatrix} 2 & 3 & 4 & 5 \\ 3 & 5 & 7 & 9 \\ 4 & 7 & 10 & 13 \\ 5 & 9 & 13 & 17 \end{bmatrix}, b = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \quad \texttt{rref([A,b])} = \begin{bmatrix} 1 & 0 & -1 & -2 & 2 \\ 0 & 1 & 2 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Zwei Nullzeilen in Matrix und erweiterter Matrix: Rang = n-2, zweiparametrige Lösungsschar: wähle x_3 und x_4 beliebig, bestimme x_1 und x_2 aus den ersten beiden Gleichungen.

$$x_1 = 2 + x_3 + 2x_4, \quad x_2 = -1 - 2x_3 + 3x_4$$

Beispiele zu `rref`

Mit derselben Matrix wie vorhin, aber anderer rechter Seite

$$A = \begin{bmatrix} 2 & 3 & 4 & 5 \\ 3 & 5 & 7 & 9 \\ 4 & 7 & 10 & 13 \\ 5 & 9 & 13 & 17 \end{bmatrix}, b = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}, \quad \texttt{rref([A,b])} = \begin{bmatrix} 1 & 0 & -1 & -2 & 0 \\ 0 & 1 & 2 & 3 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Zwei Nullzeilen in Matrix, aber nur eine in erweiterter Matrix: keine Lösung.

Gleichungen *nicht* durch Multiplikation mit Inverser lösen!

Die Gleichung $7x = 13$ lösen Sie ja auch nicht, indem sie zuerst $7^{-1} = 0,1429$ berechnen und dann $x = 0,1429 * 13$ auswerten.

Sie formen die Gleichung natürlich um und rechnen $x = \frac{13}{7}$.

MATLABs Befehl für Gleichungslösen ist `\` und *nicht* `inv`

Berechnung der Inversen und anschließende Multiplikation ist sehr viel rechenaufwändiger und anfälliger gegen Rundungsfehler als direktes Gleichungslösen!

Fehlerempfindlichkeit, Konditionszahl

Konditionszahl

Die Konditionszahl $\kappa(A)$ einer Matrix A misst für das Gleichungssystem $A\mathbf{x} = \mathbf{b}$, wie empfindlich der relative Fehler von $\mathbf{x}$ von kleinen relativen Änderungen in A und $\mathbf{b}$ abhängt.

Zusammenhang Konditionszahl – relative Fehler

$$\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \kappa(A)\left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|}\right) \quad \text{mit} \quad \kappa(A) = \|A\|\|A^{-1}\|$$

Der relative Fehler in $\mathbf{x}$ kann also $\kappa(A)$ mal größer sein als der relative Fehler in A und $\mathbf{b}$.

Determinante und Rang

$\det A \neq 0$ ist numerisch unbrauchbar

Das Kriterium $\det A \neq 0$ für die Lösbarkeit eines Gleichungssystems ist nur bei Matrizen mit wenigen Zeilen und kleinen ganzzahligen Elementen anwendbar. Ansonsten können Rundungsfehler das Ergebnis verfälschen.

MATLAB-Beispiele

`R=rosser` liefert 8×8-Matrix, `det(R)= -10611` (korrekt wäre 0!).
`H=hilb(6)` liefert 6×6-Matrix, `det(H)=5.3673e-18`. Das sieht aus wie $\det H = 0$, trotzdem ist die Matrix nicht singulär.

Rang einer Matrix wird numerisch verlässlicher berechnet

Das Kriterium $\operatorname{rang} A = n$ ist besser geeignet, um Lösbarkeit festzustellen.
Im obigen Beispiel rechnet MATLAB (korrekt): `rank(R)=7, rank(H)=6`

Lösbarkeit und Konditionszahl

Die Konditionszahl

lässt besser abschätzen, ob eine Matrix singulär oder nahezu singulär ist.
MATLAB liefert für die rosser-Matrix `cond(R) =1.4945e+16`
Rundungsfehler werden 10^{16}-fach verstärkt. Bei 16-stelliger Genauigkeit bedeutet das: Ergebnis völlig falsch!

A mit Konditionszahl $10^{16} \longrightarrow$ Gleichungssystem praktisch unlösbar

Man sagt, A ist numerisch singulär.

Der MATLAB-Operator \ zum Gleichungslösen macht automatisch eine Schätzung der *reziproken Konditionszahl* und liefert Warnung.

```
» A\b
Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate. RCOND = 7.624224e-17.
```

Konditionszahl, Beispiel 4×4 Hilbertmatrix H

$$H = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \end{bmatrix} \quad , \quad H^{-1} = \begin{bmatrix} 16 & -120 & 240 & -140 \\ -120 & 1200 & -2700 & 1680 \\ 240 & -2700 & 6480 & -4200 \\ -140 & 1680 & -4200 & 2800 \end{bmatrix}$$

Zeilensummennorm von Matrix und Inverser:

$$\|H\|_\infty = 2{,}08; \quad \|H^{-1}\|_\infty = 13\,620; \quad \kappa(H) = 28\,375$$

Störungen in den Daten werden $\approx 28\,000$-fach verstärkt.

4×4 Hilbertmatrix H, rechte Seite $\mathbf{b} = (1, 1, 1, 1)^T$

Exakte Lösung ist

$$\mathbf{x} = \begin{bmatrix} -4 \\ 60 \\ -180 \\ 140 \end{bmatrix}$$

Ändere das Element h_{44}: addiere $1/1000$. Lösung ändert sich auf

$$\mathbf{x} = \begin{bmatrix} 1,15789 \\ -1,89474 \\ -25,2632 \\ 36,8421 \end{bmatrix}$$

Gliederung 4. Vorlesung

Lösungsverfahren für lineare Gleichungssysteme

Grundlegende Unterscheidung: direkte und iterative Verfahren

Direkte Verfahren sind Varianten des Gaußschen Eliminationsverfahrens (LR-Zerlegung). Üblich bis $n \approx 10.000$ Unbekannten.
Direkte Verfahren sind allgemeiner anwendbar und rechnen zumeist schneller, sofern die Matrix im schnell zugänglichen Speicher des Rechners Platz hat.

Iterative Verfahren finden schrittweise verbesserte Näherungslösungen.
Üblich für $n \gg 10.000$.
Iterative Methoden sind nur für spezielle Matrixtypen anwendbar, die beispielsweise bei partiellen Differentialgleichungen auftreten.

Wo treten große lineare Gleichungssysteme auf?

„Groß“ in diesem Zusammenhang meint: Millionen von Gleichungen und Unbekannten

Modellierung und Simulation physikalischer Prozesse am Computer

- Numerische Strömungssimulation
- Finite-Elemente-Berechnungen in Festkörpermechanik
- Elektrisches Feld, Schwerefeld
- Wärmeleitung, konvektiver und diffusiver Transport
- Wellenausbreitung (Schall, elektromagnetisch, Wasser. . .)
- und, und und . . .

Die nächsten Folien zeigen, wie ein einfaches Wärmeleitungs-Problem zu Systemen mit vielen tausend Gleichungen führt und wie sich diese lösen lassen. (War schon in den Folien der letzen Woche, ist sich damals nicht mehr ausgegangen.)

Stationäre Wärmeleitungs-Aufgabe

Bilanzgleichungen für Temperaturen in neun Zellen eines Finite-Volumen-Rechengitters

Randtemperaturen sind vorgegeben, gesucht sind $T_1, \ldots, T_9$.

	0	0	0	
100	T_7	T_8	T_9	0
100	T_4	T_5	T_6	0
100	T_1	T_2	T_3	0
	0	0	0	

Die Finite-Volums-Diskretisierung der Wärmeleitungsgleichung liefert neun Bilanzgleichungen.

$$
\begin{aligned}
4T_1 - T_2 - T_4 &= 100 \\
-T_1 + 4T_2 - T_3 - T_5 &= 0 \\
-T_2 + 4T_3 - T_6 &= 0 \\
-T_1 + 4T_4 - T_5 - T_7 &= 100 \\
-T_2 - T_4 + 4T_5 - T_6 - T_8 &= 0 \\
-T_3 - T_5 + 4T_6 - T_9 &= 0 \\
-T_4 + 4T_7 - T_6 &= 100 \\
-T_5 - T_7 + 4T_8 - T_6 &= 0 \\
-T_6 - T_8 + 4T_9 &= 0
\end{aligned}
$$

Jede Gleichung verknüpft höchstens 5 Unbekannte; Muster „4-mal der Wert im Punkt minus Werte im Süden, Westen, Norden, Osten“

Stationäre Wärmeleitung: Matrix-Struktur

So eine Matrix-Struktur tritt in vielen Modellproblemen auf

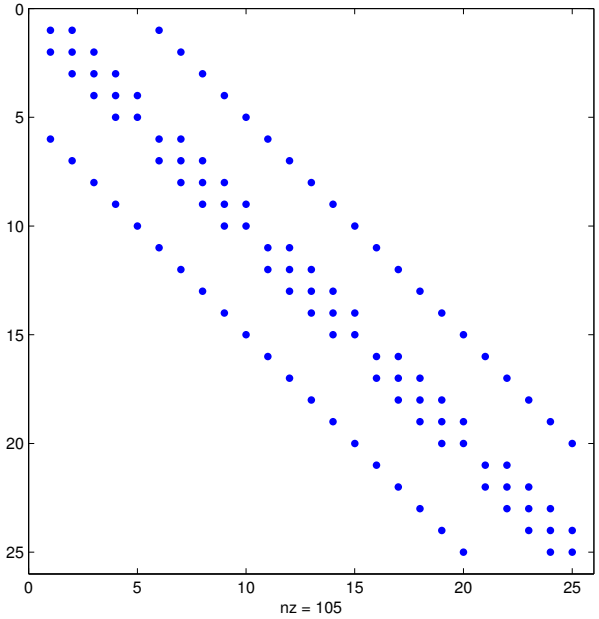
$$\begin{bmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & & & 4 & -1 & 0 & -1 & & \\ & -1 & & -1 & 4 & -1 & & -1 & \\ & & -1 & 0 & -1 & 4 & & & -1 \\ & & & -1 & & & 4 & -1 & 0 \\ & & & & -1 & & -1 & 4 & -1 \\ & & & & & -1 & 0 & -1 & 4 \end{bmatrix} \cdot \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \\ T_8 \\ T_9 \end{bmatrix} = \begin{bmatrix} 100 \\ 0 \\ 0 \\ 100 \\ 0 \\ 0 \\ 100 \\ 0 \\ 0 \end{bmatrix}$$

(Matrix-Elemente 0 sind großteils gar nicht mehr aufgeschrieben).

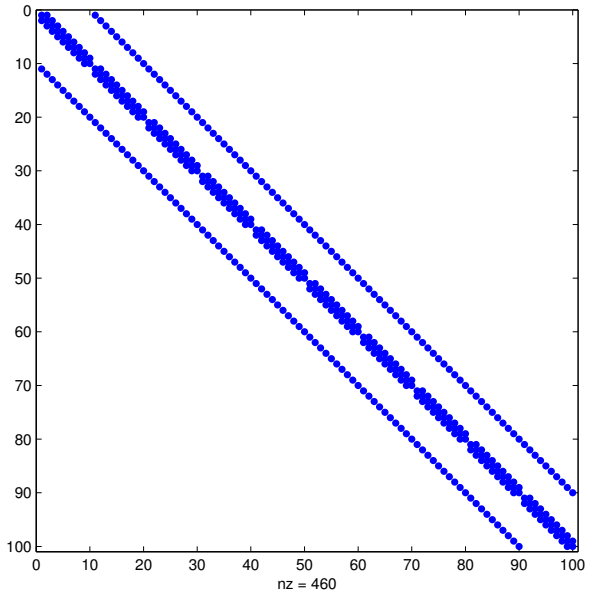
Matrix-Struktur bei größeren Gittern

Die typische Fünf-Band-Struktur eines diskreten Poisson-Problems

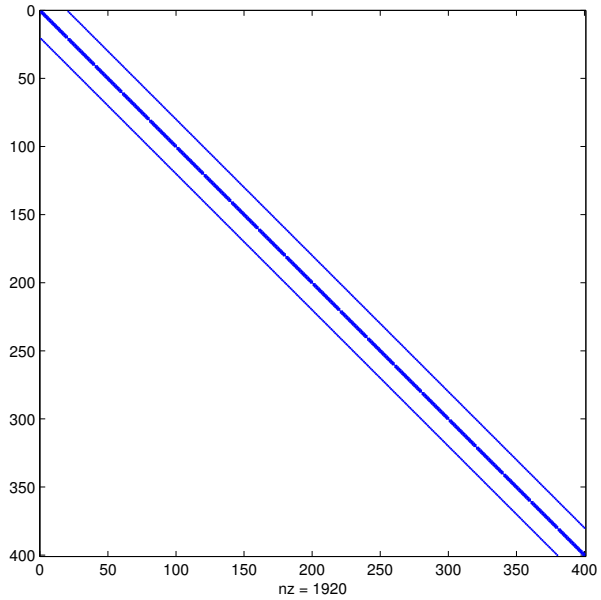
5×5



10×10



20×20



Spärlich besetzte Matrix

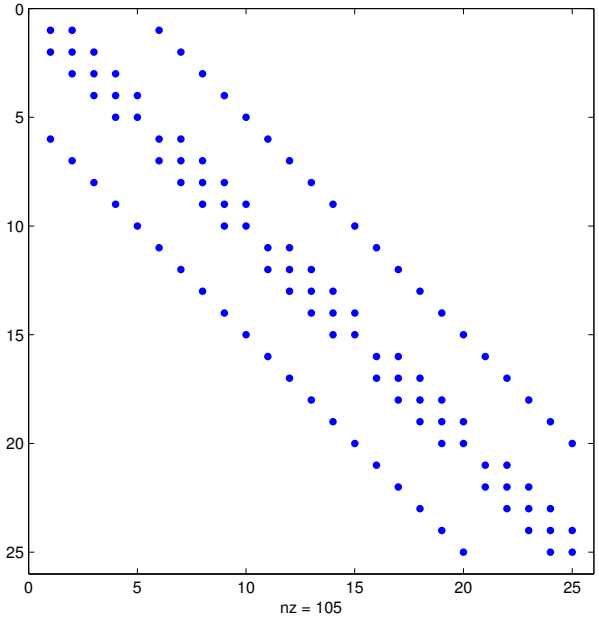
Die meisten Positionen sind mit Nullen besetzt

Für spärlich besetzte Matrizen lassen sich auch Systeme mit Millionen von Unbekannten lösen. Bei voll besetzten Matrizen dieser Größe wäre Gauß-Elimination völlig unmöglich.

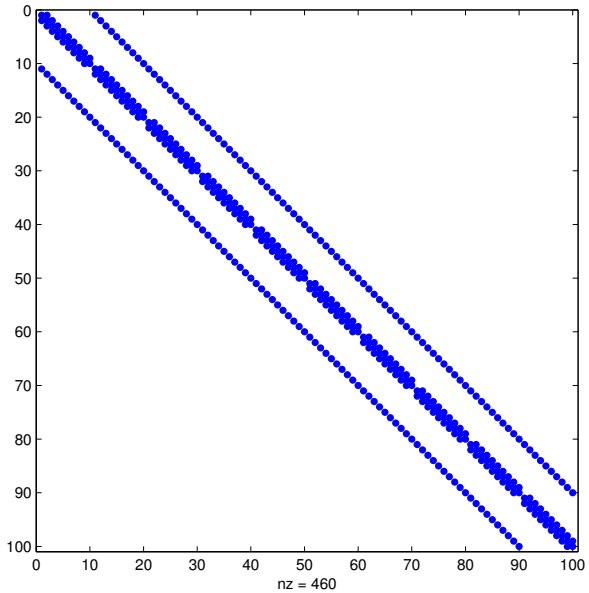
Matrix-Struktur bei größeren Gittern

Die typische Fünf-Band-Struktur eines diskreten Poisson-Problems

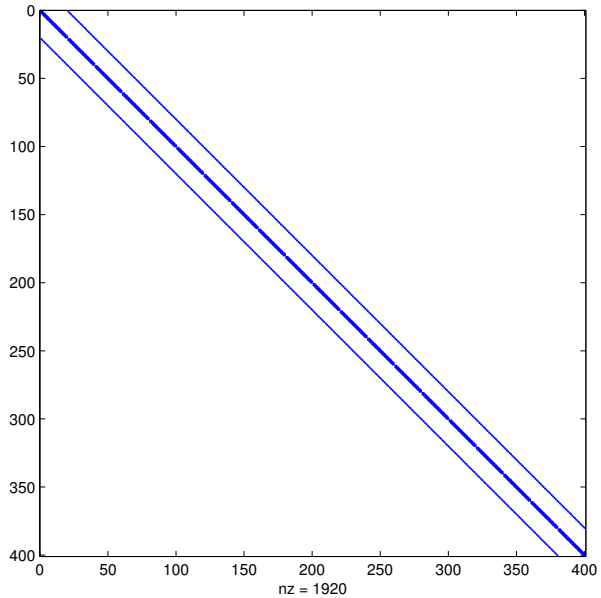
5×5



10×10



20×20



Spärlich besetzte Matrix

Die meisten Positionen sind mit Nullen besetzt

Für spärlich besetzte Matrizen lassen sich auch Systeme mit Millionen von Unbekannten lösen. Bei voll besetzten Matrizen dieser Größe wäre Gauß-Elimination völlig unmöglich.

Iterative Lösungsverfahren

Wir lösen die Gleichungen nach den Diagonal-Termen auf

$$
\begin{aligned}
4T_1 - T_2 - T_4 &= 100 \\
-T_1 + 4T_2 - T_3 - T_5 &= 0 \\
-T_2 + 4T_3 - T_6 &= 0 \\
-T_1 + 4T_4 - T_5 - T_7 &= 100 \\
-T_2 - T_4 + 4T_5 - T_6 - T_8 &= 0 \\
-T_3 - T_5 + 4T_6 - T_9 &= 0 \\
-T_4 + 4T_7 - T_6 &= 100 \\
-T_5 - T_7 + 4T_8 - T_6 &= 0 \\
-T_6 - T_8 + 4T_9 &= 0
\end{aligned}
$$

Iterative Lösungsverfahren

Wir lösen die Gleichungen nach den Diagonal-Termen auf

$$
\begin{aligned}
4T_1 - T_2 - T_4 &= 100 & 4T_1 &= 100 + T_2 + T_4 \\
-T_1 + 4T_2 - T_3 - T_5 &= 0 & 4T_2 &= 0 + T_1 + T_3 + T_5 \\
-T_2 + 4T_3 - T_6 &= 0 & 4T_3 &= 0 + T_2 + T_6 \\
-T_1 + 4T_4 - T_5 - T_7 &= 100 & 4T_4 &= 100 + T_1 + T_5 + T_7 \\
-T_2 - T_4 + 4T_5 - T_6 - T_8 &= 0 & 4T_5 &= 0 + T_2 + T_4 + T_6 + T_8 \\
-T_3 - T_5 + 4T_6 - T_9 &= 0 & 4T_6 &= 0 + T_3 + T_5 + T_9 \\
-T_4 + 4T_7 - T_6 &= 100 & 4T_7 &= 100 + T_4 T_6 \\
-T_5 - T_7 + 4T_8 - T_6 &= 0 & 4T_8 &= 0 + T_5 + T_7 + T_6 \\
-T_6 - T_8 + 4T_9 &= 0 & 4T_9 &= 0 + T_6 + T_8
\end{aligned}
$$

Iterative Lösungsverfahren

Wir lösen die Gleichungen nach den Diagonal-Termen auf

$$
\begin{aligned}
4T_1 - T_2 - T_4 &= 100 & T_1 &= (100 + T_2 + T_4)/4 \\
-T_1 + 4T_2 - T_3 - T_5 &= 0 & T_2 &= (T_1 + T_3 + T_5)/4 \\
-T_2 + 4T_3 - T_6 &= 0 & T_3 &= (T_2 + T_6)/4 \\
-T_1 + 4T_4 - T_5 - T_7 &= 100 & T_4 &= (100 + T_1 + T_5 + T_7)/4 \\
-T_2 - T_4 + 4T_5 - T_6 - T_8 &= 0 & T_5 &= (T_2 + T_4 + T_6 + T_8)/4 \\
-T_3 - T_5 + 4T_6 - T_9 &= 0 & T_6 &= (T_3 + T_5 + T_9)/4 \\
-T_4 + 4T_7 - T_6 &= 100 & T_7 &= (100 + T_4 + T_6)/4 \\
-T_5 - T_7 + 4T_8 - T_6 &= 0 & T_8 &= (T_5 + T_7 + T_6)/4 \\
-T_6 - T_8 + 4T_9 &= 0 & T_9 &= (T_6 + T_8)/4
\end{aligned}
$$

Iterative Lösungsverfahren

Wir lösen die Gleichungen nach den Diagonal-Termen auf

$$
\begin{aligned}
4T_1 - T_2 - T_4 &= 100 & T_1 &= (100 + T_2 + T_4)/4 \\
-T_1 + 4T_2 - T_3 - T_5 &= 0 & T_2 &= (T_1 + T_3 + T_5)/4 \\
-T_2 + 4T_3 - T_6 &= 0 & T_3 &= (T_2 + T_6)/4 \\
-T_1 + 4T_4 - T_5 - T_7 &= 100 & T_4 &= (100 + T_1 + T_5 + T_7)/4 \\
-T_2 - T_4 + 4T_5 - T_6 - T_8 &= 0 & T_5 &= (T_2 + T_4 + T_6 + T_8)/4 \\
-T_3 - T_5 + 4T_6 - T_9 &= 0 & T_6 &= (T_3 + T_5 + T_9)/4 \\
-T_4 + 4T_7 - T_6 &= 100 & T_7 &= (100 + T_4 + T_6)/4 \\
-T_5 - T_7 + 4T_8 - T_6 &= 0 & T_8 &= (T_5 + T_7 + T_6)/4 \\
-T_6 - T_8 + 4T_9 &= 0 & T_9 &= (T_6 + T_8)/4
\end{aligned}
$$

Einsetzen von Startwerten rechts liefert verbesserte Werte links $\rightarrow$ iteriere!

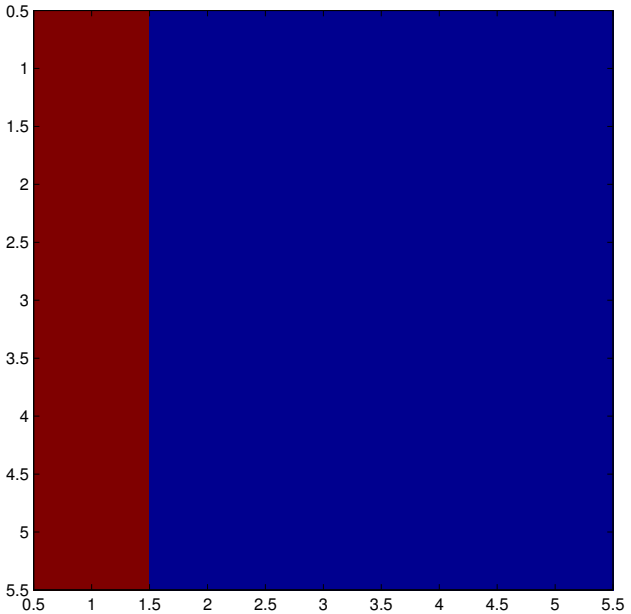
Ablauf der Iteration

bei 3×3 inneren Zellen

In dieser Form ein Jacobi-Verfahren: alle Gitter-Werte werden gleichzeitig aktualisiert. Noch langsamer als das Gauß-Seidel-Verfahren, aber für Parallel-Rechner geeignet.

	0	0	0	
100	0	0	0	0
100	0	0	0	0
100	0	0	0	0
	0	0	0	

Anfangsbedingung



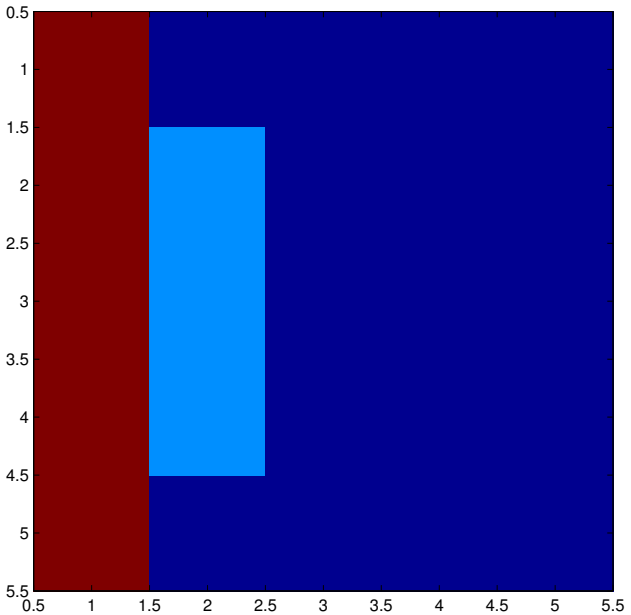
Ablauf der Iteration

bei 3×3 inneren Zellen

In dieser Form ein Jacobi-Verfahren: alle Gitter-Werte werden gleichzeitig aktualisiert. Noch langsamer als das Gauß-Seidel-Verfahren, aber für Parallel-Rechner geeignet.

	0	0	0	
100	25	0	0	0
100	25	0	0	0
100	25	0	0	0
	0	0	0	

Schritt 1



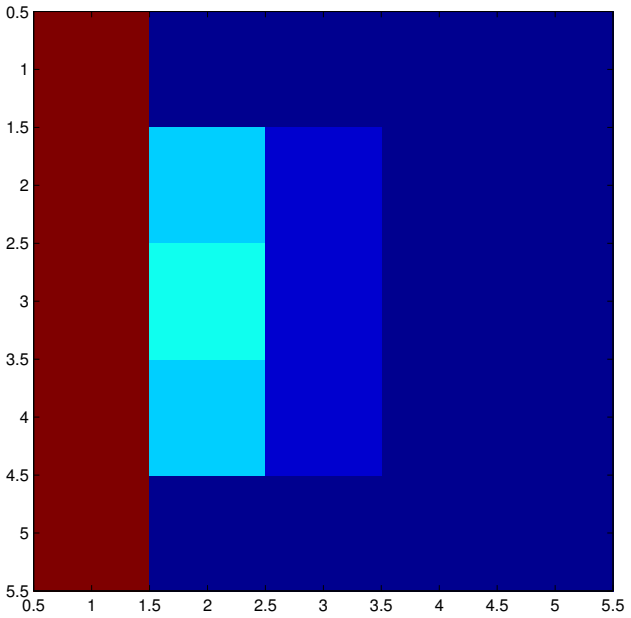
Ablauf der Iteration

bei 3×3 inneren Zellen

In dieser Form ein Jacobi-Verfahren: alle Gitter-Werte werden gleichzeitig aktualisiert. Noch langsamer als das Gauß-Seidel-Verfahren, aber für Parallel-Rechner geeignet.

	0	0	0	
100	31	6	0	0
100	38	6	0	0
100	31	6	0	0
	0	0	0	

Schritt 2



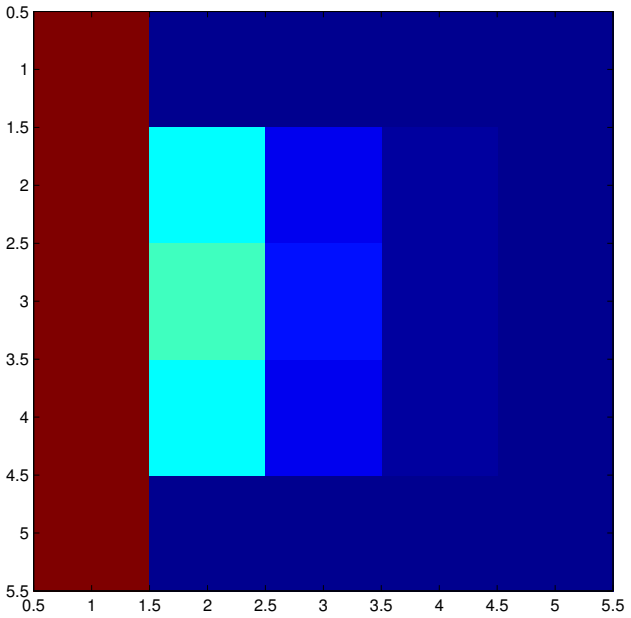
Ablauf der Iteration

bei 3×3 inneren Zellen

In dieser Form ein Jacobi-Verfahren: alle Gitter-Werte werden gleichzeitig aktualisiert. Noch langsamer als das Gauß-Seidel-Verfahren, aber für Parallel-Rechner geeignet.

	0	0	0	
100	36	9	2	0
100	42	13	2	0
100	36	9	2	0
	0	0	0	

Schritt 3



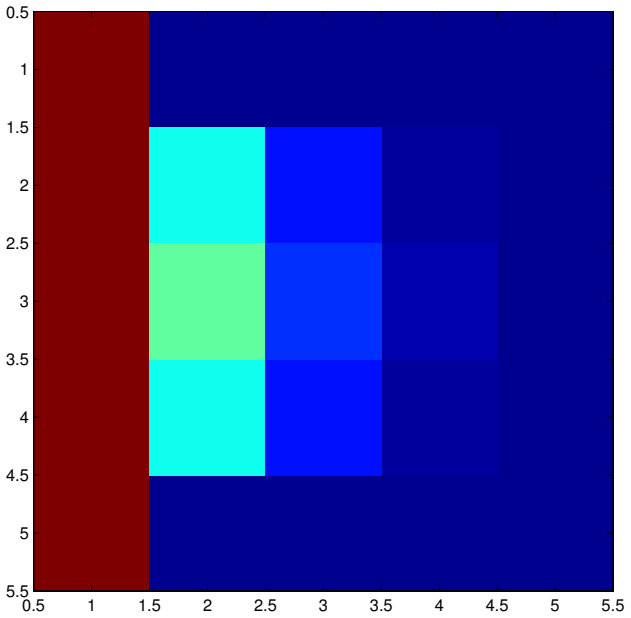
Ablauf der Iteration

bei 3×3 inneren Zellen

In dieser Form ein Jacobi-Verfahren: alle Gitter-Werte werden gleichzeitig aktualisiert. Noch langsamer als das Gauß-Seidel-Verfahren, aber für Parallel-Rechner geeignet.

	0	0	0	
100	38	13	3	0
100	46	16	4	0
100	38	13	3	0
	0	0	0	

Schritt 4



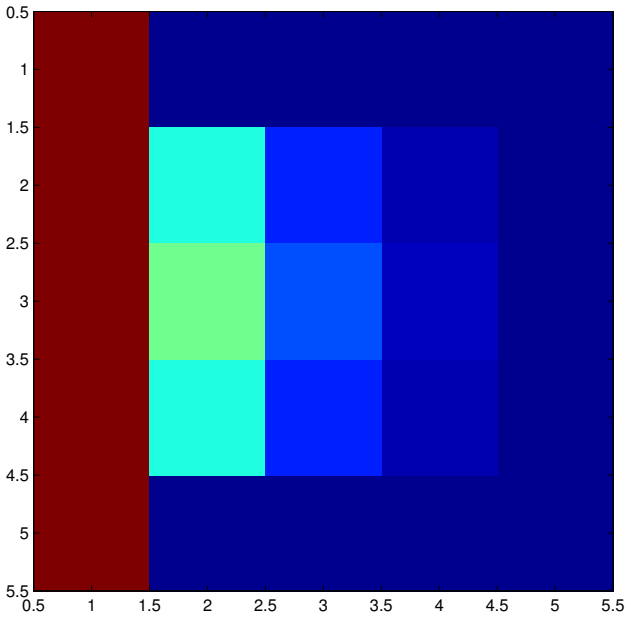
Ablauf der Iteration

bei 3×3 inneren Zellen

In dieser Form ein Jacobi-Verfahren: alle Gitter-Werte werden gleichzeitig aktualisiert. Noch langsamer als das Gauß-Seidel-Verfahren, aber für Parallel-Rechner geeignet.

	0	0	0	
100	40	14	4	0
100	48	19	5	0
100	40	14	4	0
	0	0	0	

Schritt 5



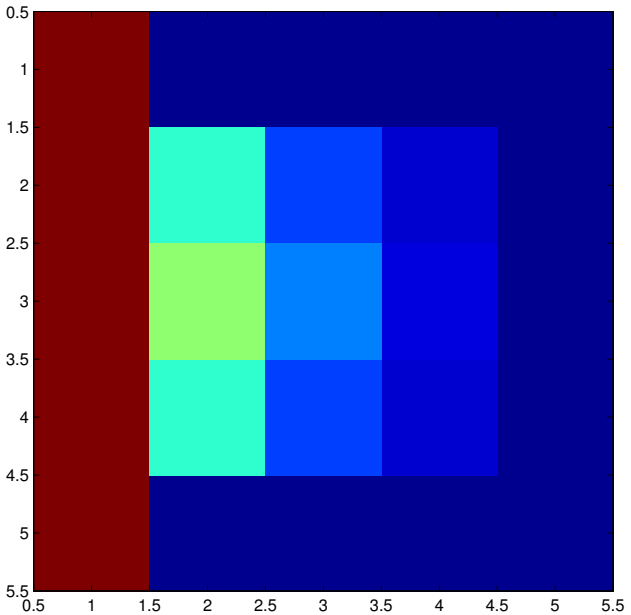
Ablauf der Iteration

bei 3×3 inneren Zellen

In dieser Form ein Jacobi-Verfahren: alle Gitter-Werte werden gleichzeitig aktualisiert. Noch langsamer als das Gauß-Seidel-Verfahren, aber für Parallel-Rechner geeignet.

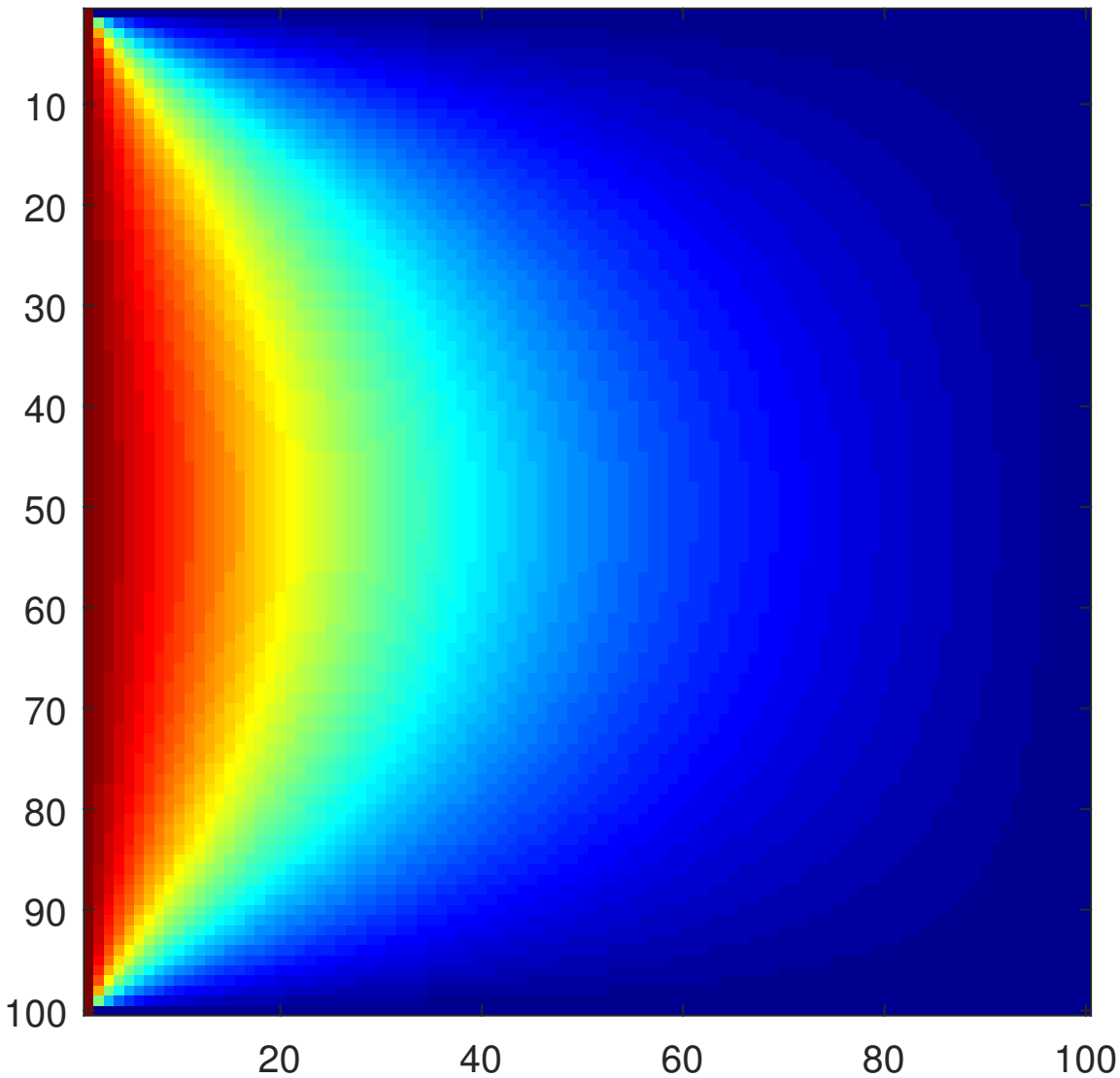
	0	0	0	
100	42	18	7	0
100	52	24	9	0
100	42	18	7	0
	0	0	0	

Schritt 10



Stationäre Temperaturverteilung

100×100-Gitter, insgesamt 10.000 Jacobi-Iterationen



Gliederung 4. Vorlesung

Die einfachsten iterativen Verfahren

- Jacobi
- Gauß-Seidel
- SOR (successive overrelaxation)

Dazu gibt es Beispiele: `JACGSSOR.m`, und animierte GIF-Dateien im Zusatzmaterial!

Jacobi-Verfahren

Idee: Löse jede Gleichung nach ihrem Diagonal-Term auf.

Standard-Form, 3×3-Beispiel

$$\begin{aligned} a_{11}\,x_1 + a_{12}\,x_2 + a_{13}\,x_3 &= b_1 \\ a_{21}\,x_1 + a_{22}\,x_2 + a_{23}\,x_3 &= b_2 \\ a_{31}\,x_1 + a_{32}\,x_2 + a_{33}\,x_3 &= b_3 \end{aligned}$$

Auflösen nach Diagonal-Term $\rightarrow$ Fixpunkt-Form

$$\begin{aligned} a_{11}\,x_1 &= b_1 - a_{12}x_2 - a_{13}x_3 \\ a_{22}\,x_2 &= b_2 - a_{21}x_1 - a_{23}x_3 \\ a_{33}\,x_3 &= b_3 - a_{31}x_1 - a_{32}x_2 \end{aligned}$$

Jacobi-Verfahren

Idee: Löse jede Gleichung nach ihrem Diagonal-Term auf.

Standard-Form, 3×3-Beispiel

$$\begin{aligned} a_{11}\,x_1 + a_{12}\,x_2 + a_{13}\,x_3 &= b_1 \\ a_{21}\,x_1 + a_{22}\,x_2 + a_{23}\,x_3 &= b_2 \\ a_{31}\,x_1 + a_{32}\,x_2 + a_{33}\,x_3 &= b_3 \end{aligned}$$

Auflösen nach Diagonal-Term $\rightarrow$ Fixpunkt-Form

$$\begin{aligned} x_1 &= (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11} \\ x_2 &= (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22} \\ x_3 &= (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33} \end{aligned}$$

Jacobi-Verfahren

Idee: Löse jede Gleichung nach ihrem Diagonal-Term auf.

Standard-Form, 3×3-Beispiel

$$\begin{aligned} a_{11}\,x_1 + a_{12}\,x_2 + a_{13}\,x_3 &= b_1 \\ a_{21}\,x_1 + a_{22}\,x_2 + a_{23}\,x_3 &= b_2 \\ a_{31}\,x_1 + a_{32}\,x_2 + a_{33}\,x_3 &= b_3 \end{aligned}$$

Auflösen nach Diagonal-Term $\rightarrow$ Fixpunkt-Form

$$\begin{aligned} x_1 &= (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11} \\ x_2 &= (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22} \\ x_3 &= (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33} \end{aligned}$$

setze Startwerte ein, iteriere

Jacobi-Verfahren

3×3-Beispiel

$$\begin{aligned} 2x_1 - x_2 &= 1 \\ -x_1 + 2x_2 - x_3 &= 0 \\ -x_2 + 2x_3 &= 0 \end{aligned}$$

Auflösen nach Diagonal-Term $\rightarrow$ Fixpunkt-Form

Jacobi-Verfahren

3×3-Beispiel

$$\begin{aligned} 2x_1 - x_2 &= 1 \\ -x_1 + 2x_2 - x_3 &= 0 \\ -x_2 + 2x_3 &= 0 \end{aligned}$$

Auflösen nach Diagonal-Term $\rightarrow$ Fixpunkt-Form

$$\begin{aligned} x_1 &= (1 + x_2)/2 \\ x_2 &= (x_1 + x_3)/2 \\ x_3 &= x_2/2 \end{aligned}$$

Jacobi-Verfahren

3×3-Beispiel

$$\begin{aligned} 2x_1 - x_2 &= 1 \\ -x_1 + 2x_2 - x_3 &= 0 \\ -x_2 + 2x_3 &= 0 \end{aligned}$$

Auflösen nach Diagonal-Term $\rightarrow$ Fixpunkt-Form

$$\begin{aligned} x_1 &= (1 + x_2)/2 \\ x_2 &= (x_1 + x_3)/2 \\ x_3 &= x_2/2 \end{aligned}$$

setze Startwerte ein, iteriere

Gauß-Seidel-Verfahren

Idee: Im Prinzip wie das Jacobi-Verfahren, nur: setze neue Werte, soweit verfügbar, schon im aktuellen Schritt ein

Fixpunkt-Form, 3×3-Beispiel, Ablauf der Iteration
Startwerte x_1, x_2, x_3, neu berechnete Werte x_1, x_2, x_3

$$\begin{aligned} x_1 &= (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11} \\ x_2 &= (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22} \\ x_3 &= (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33} \end{aligned}$$

Gauß-Seidel-Verfahren

Idee: Im Prinzip wie das Jacobi-Verfahren, nur: setze neue Werte, soweit verfügbar, schon im aktuellen Schritt ein

Fixpunkt-Form, 3×3-Beispiel, Ablauf der Iteration
Startwerte x_1, x_2, x_3, neu berechnete Werte x_1, x_2, x_3

$$
\begin{aligned}
x_1 &= (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11} \\
x_2 &= (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22} \\
x_3 &= (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}
\end{aligned}
$$

Gauß-Seidel-Verfahren

Idee: Im Prinzip wie das Jacobi-Verfahren, nur: setze neue Werte, soweit verfügbar, schon im aktuellen Schritt ein

Fixpunkt-Form, 3×3-Beispiel, Ablauf der Iteration
Startwerte x_1, x_2, x_3, neu berechnete Werte x_1, x_2, x_3

$$
\begin{aligned}
x_1 &= (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11} \\
x_2 &= (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22} \\
x_3 &= (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}
\end{aligned}
$$

Gauß-Seidel-Verfahren

Idee: Im Prinzip wie das Jacobi-Verfahren, nur: setze neue Werte, soweit verfügbar, schon im aktuellen Schritt ein

Fixpunkt-Form, 3×3-Beispiel, Ablauf der Iteration
Startwerte x_1, x_2, x_3, neu berechnete Werte x_1, x_2, x_3

$$
\begin{aligned}
x_1 &= (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11} \\
x_2 &= (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22} \\
x_3 &= (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}
\end{aligned}
$$

SOR-Verfahren

SOR bedeutet *successive over-relaxation*

Idee

Jeweils neuer Näherungswert zuerst als Zwischenresultat aus Gauß-Seidel-Schritt; endgültiger Näherungswert durch Extrapolation (Überrelaxation) aus alter Näherung und Zwischenresultat.

Extrapolationsfaktor ω

Der neue Wert liegt (hoffentlich) näher am Ziel als der alte. Extrapoliere den Schritt von alt auf neu um den Faktor ω.

Schwierig ist, den Faktor ω im Bereich $1 < \omega < 2$ gut zu wählen. Zu klein: bringt wenig Verbesserung. Zu groß: schießt übers Ziel hinaus.

SOR-Verfahren

3×3-Beispiel, Ablauf der Iteration

Alte Werte x_1, x_2, x_3
Zwischenresultate x_1, x_2, x_3 nach Gauß-Seidel-Schritt
Extrapolierte Werte x_1, x_2, x_3

GS-Schritt	$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11}$
Extrapolation	$x_1 = \omega x_1 + (1-\omega)x_1$
GS-Schritt	$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22}$
Extrapolation	$x_2 = \omega x_2 + (1-\omega)x_2$
GS-Schritt	$x_3 = (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}$
Extrapolation	$x_3 = \omega x_3 + (1-\omega)x_3$

SOR-Verfahren

3×3-Beispiel, Ablauf der Iteration

Alte Werte x_1, x_2, x_3
Zwischenresultate x_1, x_2, x_3 nach Gauß-Seidel-Schritt
Extrapolierte Werte x_1, x_2, x_3

GS-Schritt	$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11}$
Extrapolation	$x_1 = \omega x_1 + (1-\omega)x_1$
GS-Schritt	$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22}$
Extrapolation	$x_2 = \omega x_2 + (1-\omega)x_2$
GS-Schritt	$x_3 = (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}$
Extrapolation	$x_3 = \omega x_3 + (1-\omega)x_3$

SOR-Verfahren

3×3-Beispiel, Ablauf der Iteration

Alte Werte x_1, x_2, x_3
Zwischenresultate x_1, x_2, x_3 nach Gauß-Seidel-Schritt
Extrapolierte Werte x_1, x_2, x_3

GS-Schritt	$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11}$
Extrapolation	$x_1 = \omega x_1 + (1-\omega)x_1$
GS-Schritt	$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22}$
Extrapolation	$x_2 = \omega x_2 + (1-\omega)x_2$
GS-Schritt	$x_3 = (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}$
Extrapolation	$x_3 = \omega x_3 + (1-\omega)x_3$

SOR-Verfahren

3×3-Beispiel, Ablauf der Iteration

Alte Werte x_1, x_2, x_3
Zwischenresultate x_1, x_2, x_3 nach Gauß-Seidel-Schritt
Extrapolierte Werte x_1, x_2, x_3

GS-Schritt	$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11}$
Extrapolation	$x_1 = \omega x_1 + (1-\omega)x_1$
GS-Schritt	$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22}$
Extrapolation	$x_2 = \omega x_2 + (1-\omega)x_2$
GS-Schritt	$x_3 = (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}$
Extrapolation	$x_3 = \omega x_3 + (1-\omega)x_3$

SOR-Verfahren

3×3-Beispiel, Ablauf der Iteration

Alte Werte x_1, x_2, x_3
Zwischenresultate x_1, x_2, x_3 nach Gauß-Seidel-Schritt
Extrapolierte Werte x_1, x_2, x_3

GS-Schritt	$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11}$
Extrapolation	$x_1 = \omega x_1 + (1-\omega)x_1$
GS-Schritt	$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22}$
Extrapolation	$x_2 = \omega x_2 + (1-\omega)x_2$
GS-Schritt	$x_3 = (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}$
Extrapolation	$x_3 = \omega x_3 + (1-\omega)x_3$

SOR-Verfahren

3×3-Beispiel, Ablauf der Iteration

Alte Werte x_1, x_2, x_3
Zwischenresultate x_1, x_2, x_3 nach Gauß-Seidel-Schritt
Extrapolierte Werte x_1, x_2, x_3

GS-Schritt	$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11}$
Extrapolation	$x_1 = \omega x_1 + (1-\omega)x_1$
GS-Schritt	$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22}$
Extrapolation	$x_2 = \omega x_2 + (1-\omega)x_2$
GS-Schritt	$x_3 = (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}$
Extrapolation	$x_3 = \omega x_3 + (1-\omega)x_3$

Konvergenz

Iterative Verfahren konvergieren nicht für beliebige Matrizen!

Konvergenzaussagen für Jacobi- und Gauß-Seidel existieren für Matrizen, bei denen in jeder Zeile die Beträge in der Hauptdiagonalen im Vergleich zur Summe der restlichen Element-Beträge dominieren.

Stark diagonaldominante Matrix $A = [a_{ij}]$

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}| \quad \forall j = 1, \ldots, n$$

Es reicht auch $\geq$ (*schwach diagonaldominante Matrix*) für Konvergenz.

(Mit ein paar Zusatzannahmen, die bei praktisch relevanten Anwendungen in der Regel erfüllt sind.)

Zum Glück erfüllen Matrizen, die bei praktischen Anwendungen auftreten, oft dieses Kriterium (vergleiche Wärmeleitungs-Beispiel oben!)

Prüfungsfrage

Gegeben sei das System $A\mathbf{x} = \mathbf{b}$ mit

$$A = \begin{bmatrix} 4 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} -40 \\ 62 \\ 18 \end{bmatrix}, \quad \mathbf{x}^{(0)} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

1. Was liefern die ersten beiden Schritte des iterativen Jacobi-Verfahrens?
2. Was liefern die ersten beiden Schritte des iterativen Gauß-Seidel-Verfahrens?

Gliederung 4. Vorlesung

Dreiecksmatrizen

Gleichungssysteme mit Dreiecksmatrizen sind direkt auflösbar

Beispiel für $n = 4$: Linke untere und rechte obere Dreiecksmatrix

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \ell_{21} & 1 & 0 & 0 \\ \ell_{31} & \ell_{32} & 1 & 0 \\ \ell_{41} & \ell_{42} & \ell_{43} & 1 \end{bmatrix}, \qquad R = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ 0 & r_{22} & r_{23} & r_{24} \\ 0 & 0 & r_{33} & r_{34} \\ 0 & 0 & 0 & r_{44} \end{bmatrix}$$

Vorwärts- bzw. Rückwärts-Substitution löst Gleichungssysteme mit Dreiecksform.

Rechenaufwand

bei einem $n \times n$-System beträgt jeweils $n^2/2 + O(n)$ Punktoperationen.

Klassische Gauß-Elimination

transformiert ein Gleichungssystem auf obere Dreiecksform

(sofern bei der Pivot-Berechnung immer $a_{kk} \neq 0$!)

Für alle Spalten $k = 1, \ldots n-1$

in Spalte k: für Zeilen unterhalb des Diagonalelements

Zeilenindex $i = k+1, \ldots, n$

setze $p = a_{ik}/a_{kk}$ (Pivot-Koeffizient)

subtrahiere das p-fache derZeile k von Zeile i:

Für die Spalten $j = k, \ldots n$ von Zeile i

$$a_{ij} = a_{ij} - p a_{kj}$$

Für rechte Seite: $b_i = b_i - p b_k$

Rechenaufwand

beträgt $n^3/3 + O(n^2)$ Punktoperationen

Beispiel

Lösung eines Gleichungssystems durch Elimination

Das Gaußsche Eliminationsverfahren transformiert

—sofern bei der Pivot-Berechnung immer $a_{kk} \neq 0$!—

die erweiterte Koeffizientenmatrix auf Dreiecksgestalt.

$$[A\,\mathbf{b}] = \begin{bmatrix} 5 & 6 & 7 & 6 \\ 10 & 20 & 23 & 6 \\ 15 & 50 & 67 & 14 \end{bmatrix} \longrightarrow \begin{bmatrix} 5 & 6 & 7 & 6 \\ 0 & 8 & 9 & -6 \\ 0 & 0 & 10 & 20 \end{bmatrix}$$

Rücksubstitution liefert Lösung.

Warnhinweis

If anything can go wrong, it will!

Das Eliminationsverfahren in der oben angegebenen einfachen Form bricht zusammen, wenn in seinem Verlauf $a_{kk} = 0$ auftritt.

Division durch Null

bei der Berechnung des Pivot-Koeffizienten.

Abhilfe

Pivotisierung!
Pivot = Dreh-, Angelpunkt.

Pivotisierung

Systematisches Vertauschen von Gleichungen und Unbekannten verhindert vorzeitige Division durch $a_{kk} = 0$ im Eliminationsverfahren.

vollständige Pivotisierung Sucht betragsgrößtes Element unter allen Einträgen in den Positionen von a_{kk} bis a_{nn} und bringt es an die kk-Position. Vertauscht Gleichungen und Unbekannte.

Spalten-Pivotisierung Sucht nur in der k-ten Spalte, vom Element a_{kk} an abwärts. Vertauscht nur Geichungen, behält Reihenfolge der Unbekannten. Standardverfahren!.

Zusatz-Nutzen Pivotisierung verringert Rundungsfehler bei der Elimination.

Stufenform und Lösbarkeit

Mögliche Fälle nach Abschluss des Eliminationsverfahrens

Gauß-Elimination mit vollständiger oder Zeilen-Pivotsuche transformiert die Originalmatrix A und rechte Seite $\mathbf{b}$ auf ein System in Stufenform: In jeder Zeile verringert sich die Zahl der Unbekannten um mindestens eine, die dann auch in den darauffolgenden Zeilen nicht mehr vorkommt.

Nach Transformation auf Stufenform

- Es treten Nullzeilen in A auf und alle entsprechenden Einträge in b sind ebenfals Null: unendlich viele Lösungen
- Es treten Nullzeilen in A auf, aber zumindest ein entsprechender Eintrage in b ist nicht Null: keine Lösung
- Es treten keine Nullzeilen in A auf: eindeutige Lösung

Gauß-Jordan-Verfahren

Eine erweiterte Variante der Standard-Gauß-Elimination

Das Gauß-Jordan-Verfahren transformiert die erweiterte Koeffizientenmatrix auf reduzierte Stufenform (reduced row echelon form). Die Lösung ist direkt ablesbar.

$$[A\,\mathbf{b}] = \begin{bmatrix} 5 & 6 & 7 & 6 \\ 10 & 20 & 23 & 6 \\ 15 & 50 & 67 & 14 \end{bmatrix}, \; \texttt{rref}([A, \mathbf{b}]) \longrightarrow \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 2 \end{bmatrix}$$

$$[A\,\mathbf{b}] = \begin{bmatrix} 5 & 6 & 7 & 6 \\ 10 & 20 & 23 & 6 \\ 15 & 50 & 57 & -6 \end{bmatrix}, \; \texttt{rref}([A, \mathbf{b}]) \longrightarrow \begin{bmatrix} 1 & 0 & 1/20 & 21/10 \\ 0 & 1 & 9/8 & -3/4 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$[A\,\mathbf{b}] = \begin{bmatrix} 5 & 6 & 7 & 6 \\ 10 & 20 & 23 & 6 \\ 15 & 50 & 57 & 14 \end{bmatrix}, \; \texttt{rref}([A, \mathbf{b}]) \longrightarrow \begin{bmatrix} 1 & 0 & 1/20 & 0 \\ 0 & 1 & 9/8 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Gauß-Jordan-Verfahren

Eine erweiterte Variante der Standard-Gauß-Elimination

Das Gauß-Jordan-Verfahren transformiert die erweiterte Koeffizientenmatrix auf reduzierte Stufenform (reduced row echelon form). Die Lösung ist direkt ablesbar.

$$[A\,\mathbf{b}] = \begin{bmatrix} 5 & 6 & 7 & 6 \\ 10 & 20 & 23 & 6 \\ 15 & 50 & 67 & 14 \end{bmatrix}, \quad \texttt{rref}([A, \mathbf{b}]) \longrightarrow \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 2 \end{bmatrix}$$

$$[A\,\mathbf{b}] = \begin{bmatrix} 5 & 6 & 7 & 6 \\ 10 & 20 & 23 & 6 \\ 15 & 50 & 57 & -6 \end{bmatrix}, \quad \texttt{rref}([A, \mathbf{b}]) \longrightarrow \begin{bmatrix} 1 & 0 & 1/20 & 21/10 \\ 0 & 1 & 9/8 & -3/4 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$[A\,\mathbf{b}] = \begin{bmatrix} 5 & 6 & 7 & 6 \\ 10 & 20 & 23 & 6 \\ 15 & 50 & 57 & 14 \end{bmatrix}, \quad \texttt{rref}([A, \mathbf{b}]) \longrightarrow \begin{bmatrix} 1 & 0 & 1/20 & 0 \\ 0 & 1 & 9/8 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Gauß-Jordan-Verfahren

Eine erweiterte Variante der Standard-Gauß-Elimination

Das Gauß-Jordan-Verfahren transformiert die erweiterte Koeffizientenmatrix auf reduzierte Stufenform (reduced row echelon form). Die Lösung ist direkt ablesbar.

$$[A\,\mathbf{b}] = \begin{bmatrix} 5 & 6 & 7 & 6 \\ 10 & 20 & 23 & 6 \\ 15 & 50 & 67 & 14 \end{bmatrix}, \quad \texttt{rref}([A, \mathbf{b}]) \longrightarrow \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 2 \end{bmatrix}$$

$$[A\,\mathbf{b}] = \begin{bmatrix} 5 & 6 & 7 & 6 \\ 10 & 20 & 23 & 6 \\ 15 & 50 & 57 & -6 \end{bmatrix}, \quad \texttt{rref}([A, \mathbf{b}]) \longrightarrow \begin{bmatrix} 1 & 0 & 1/20 & 21/10 \\ 0 & 1 & 9/8 & -3/4 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$[A\,\mathbf{b}] = \begin{bmatrix} 5 & 6 & 7 & 6 \\ 10 & 20 & 23 & 6 \\ 15 & 50 & 57 & 14 \end{bmatrix}, \quad \texttt{rref}([A, \mathbf{b}]) \longrightarrow \begin{bmatrix} 1 & 0 & 1/20 & 0 \\ 0 & 1 & 9/8 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Gliederung 4. Vorlesung

Überbestimmte Systeme

Ein lineares Gleichungssystem mit mehr Gleichungen als Unbekannten heißt überbestimmt

$$\begin{array}{rcrcl} x & & & = & 1 \\ & & y & = & 2 \\ x & + & y & = & 4 \end{array} \qquad \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}$$

Allgemein: ein überbestimmtes lineares System

$$A\mathbf{x} = \mathbf{b} \quad \text{mit einer } m \times n\text{-Matrix } A, \text{ wobei } m > n$$

- In der Regel hat ein solches System keine (exakte) Lösung.
- (Kompromiss-)Lösung sucht möglichst kleinen Residuenvektor

$$\mathbf{r} = \mathbf{b} - A\mathbf{x}$$

- Klassische kleinste-Quadrate-Lösung mit Normalengleichungen

$$A^T A\mathbf{x} = A^T \mathbf{b}$$

Überbestimmte Systeme

Ein lineares Gleichungssystem mit mehr Gleichungen als Unbekannten heißt überbestimmt

$$\begin{array}{rcrcl} x & & & = & 1 \\ & & y & = & 2 \\ x & + & y & = & 4 \end{array} \qquad \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}$$

Allgemein: ein überbestimmtes lineares System

$$A\mathbf{x} = \mathbf{b} \quad \text{mit einer } m \times n\text{-Matrix } A, \text{ wobei } m > n$$

- In der Regel hat ein solches System keine (exakte) Lösung.
- (Kompromiss-)Lösung sucht möglichst kleinen Residuenvektor

$$\mathbf{r} = \mathbf{b} - A\mathbf{x}$$

- Klassische kleinste-Quadrate-Lösung mit Normalengleichungen

$$A^T A\mathbf{x} = A^T \mathbf{b}$$

Überbestimmte Systeme

Ein lineares Gleichungssystem mit mehr Gleichungen als Unbekannten heißt überbestimmt

$$\begin{array}{rcrcl} x & & & = & 1 \\ & & y & = & 2 \\ x & + & y & = & 4 \end{array} \qquad \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}$$

Allgemein: ein überbestimmtes lineares System

$$A\mathbf{x} = \mathbf{b} \quad \text{mit einer } m \times n\text{-Matrix } A, \text{ wobei } m > n$$

- In der Regel hat ein solches System keine (exakte) Lösung.
- (Kompromiss-)Lösung sucht möglichst kleinen Residuenvektor

$$\mathbf{r} = \mathbf{b} - A\mathbf{x}$$

- Klassische kleinste-Quadrate-Lösung mit Normalengleichungen

$$A^T A\mathbf{x} = A^T \mathbf{b}$$

Überbestimmte Systeme

Ein lineares Gleichungssystem mit mehr Gleichungen als Unbekannten heißt überbestimmt

$$\begin{array}{rcrcl} x & & & = & 1 \\ & & y & = & 2 \\ x & + & y & = & 4 \end{array} \qquad \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}$$

Allgemein: ein überbestimmtes lineares System

$$A\mathbf{x} = \mathbf{b} \quad \text{mit einer } m \times n\text{-Matrix } A, \text{ wobei } m > n$$

- In der Regel hat ein solches System keine (exakte) Lösung.
- (Kompromiss-)Lösung sucht möglichst kleinen Residuenvektor

$$\mathbf{r} = \mathbf{b} - A\mathbf{x}$$

- Klassische kleinste-Quadrate-Lösung mit Normalengleichungen

$$A^T A\mathbf{x} = A^T \mathbf{b}$$

Überbestimmte Systeme

Ein lineares Gleichungssystem mit mehr Gleichungen als Unbekannten heißt überbestimmt

$$\begin{array}{rcrcl} x & & & = & 1 \\ & & y & = & 2 \\ x & + & y & = & 4 \end{array} \qquad \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}$$

Allgemein: ein überbestimmtes lineares System

$$A\mathbf{x} = \mathbf{b} \quad \text{mit einer } m \times n\text{-Matrix } A, \text{ wobei } m > n$$

- In der Regel hat ein solches System keine (exakte) Lösung.
- (Kompromiss-)Lösung sucht möglichst kleinen Residuenvektor

$$\mathbf{r} = \mathbf{b} - A\mathbf{x}$$

- Klassische kleinste-Quadrate-Lösung mit Normalengleichungen

$$A^T A\mathbf{x} = A^T \mathbf{b}$$

Beispiel: Wägung zweier Massen

Zwei Massen m_1, m_2 werden zuerst einzeln, dann gemeinsam abgewogen. Die Messwerte sind:

$$\begin{aligned} m_1 &= 1 \\ m_2 &= 2 \\ m_1 + m_2 &= 4 \end{aligned}$$

Lösungsvorschläge?

- Letzte Gleichung weglassen: Fehler $\mathbf{r} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$
- Fehler „aufteilen“, etwa $m_1 = 1{,}5; m_2 = 2{,}5 \rightarrow \mathbf{r} = \begin{bmatrix} -0{,}5 \\ -0{,}5 \\ 0 \end{bmatrix}$
- oder noch fairer auf die drei Komponenten aufteilen...

Beispiel: Wägung zweier Massen

Zwei Massen m_1, m_2 werden zuerst einzeln, dann gemeinsam abgewogen. Die Messwerte sind:

$$\begin{aligned} m_1 &= 1 \\ m_2 &= 2 \\ m_1 + m_2 &= 4 \end{aligned}$$

Lösungsvorschläge?

- Letzte Gleichung weglassen: Fehler $\mathbf{r} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$
- Fehler „aufteilen“, etwa $m_1 = 1{,}5; m_2 = 2{,}5 \rightarrow \mathbf{r} = \begin{bmatrix} -0{,}5 \\ -0{,}5 \\ 0 \end{bmatrix}$
- oder noch fairer auf die drei Komponenten aufteilen...

Beispiel: Wägung zweier Massen

Zwei Massen m_1, m_2 werden zuerst einzeln, dann gemeinsam abgewogen. Die Messwerte sind:

$$\begin{aligned} m_1 &= 1 \\ m_2 &= 2 \\ m_1 + m_2 &= 4 \end{aligned}$$

Lösungsvorschläge?

- Letzte Gleichung weglassen: Fehler $\mathbf{r} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$
- Fehler „aufteilen“, etwa $m_1 = 1{,}5; m_2 = 2{,}5 \rightarrow \mathbf{r} = \begin{bmatrix} -0{,}5 \\ -0{,}5 \\ 0 \end{bmatrix}$
- oder noch fairer auf die drei Komponenten aufteilen...

Beispiel: Wägung zweier Massen

Zwei Massen m_1, m_2 werden zuerst einzeln, dann gemeinsam abgewogen. Die Messwerte sind:

$$\begin{aligned} m_1 &= 1 \\ m_2 &= 2 \\ m_1 + m_2 &= 4 \end{aligned}$$

Lösungsvorschläge?

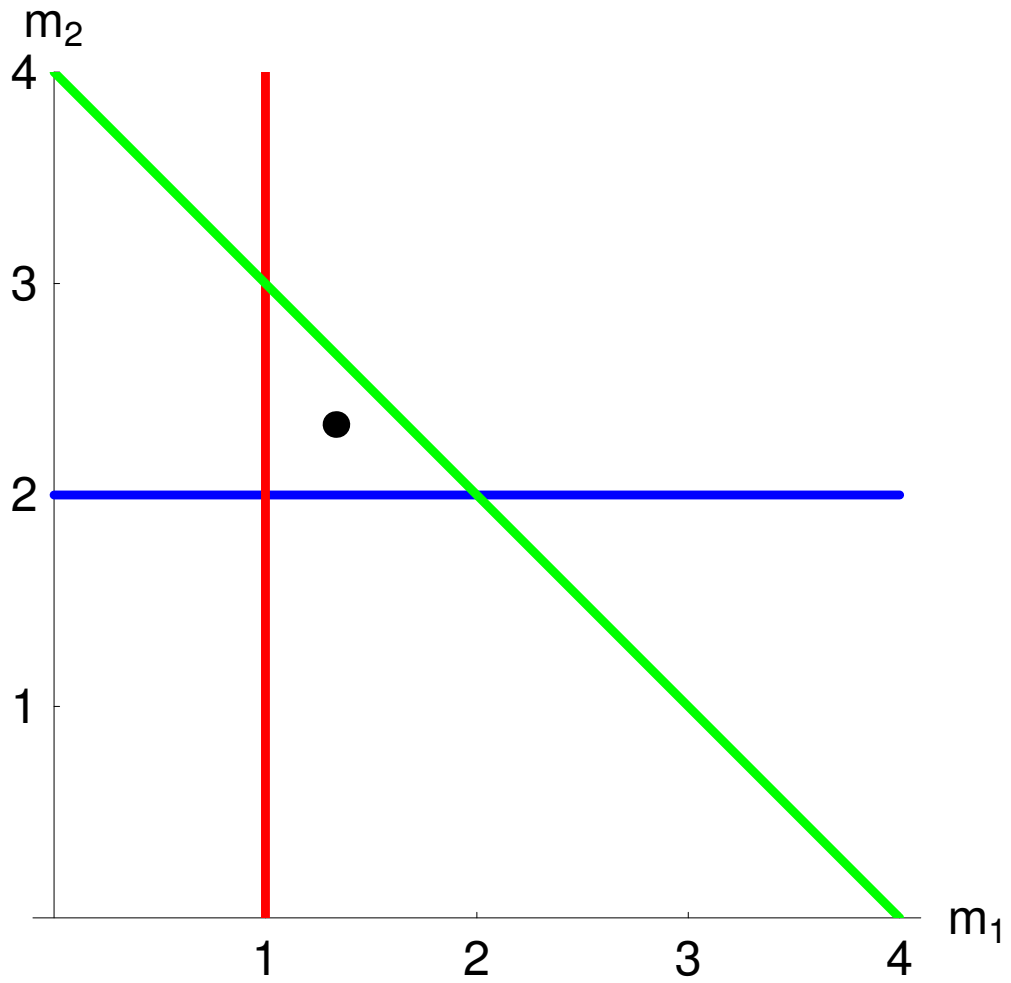
- Letzte Gleichung weglassen: Fehler $\mathbf{r} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$
- Fehler „aufteilen“, etwa $m_1 = 1{,}5; m_2 = 2{,}5 \rightarrow \mathbf{r} = \begin{bmatrix} -0{,}5 \\ -0{,}5 \\ 0 \end{bmatrix}$
- oder noch fairer auf die drei Komponenten aufteilen...

Geometrische Interpretation

Zeilen des Gleichungssystems sind Geradengleichungen

$$\begin{aligned} m_1 &= 1 \\ m_2 &= 2 \\ m_1 + m_2 &= 4 \end{aligned}$$

Den drei Gleichungen entsprechen drei Gerade im $\mathbb{R}^2$. Kein gemeinsamer Schnittpunkt!
„In der Mitte des Fehlerdreiecks“, im Punkt $(4/3; 7/3)$ ist die Summe der Fehlerquadrate minimal.



$$(m_1 - 1)^2 + (m_2 - 2)^2 + (m_1 + m_2 - 4)^2 \to \min!$$

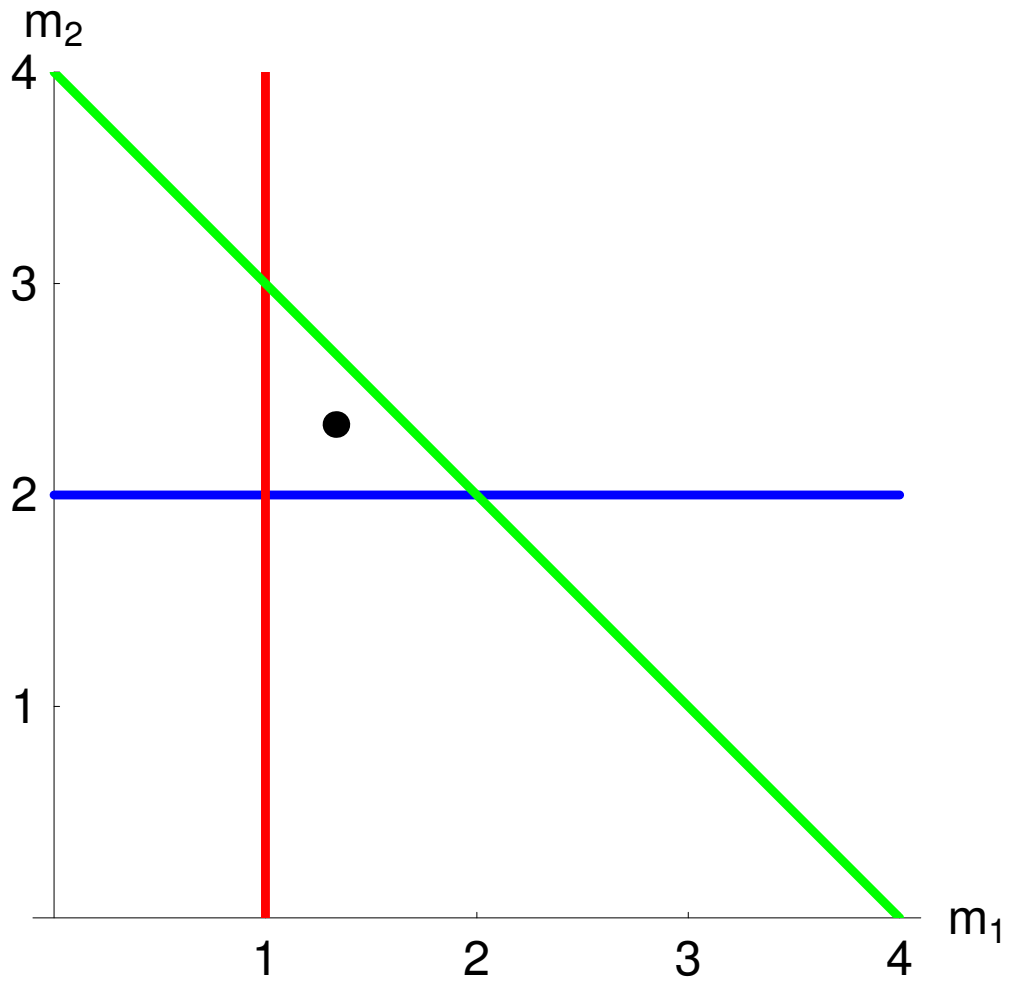
Geometrische Interpretation

Zeilen des Gleichungssystems sind Geradengleichungen

$$\begin{aligned} m_1 &= 1 \\ m_2 &= 2 \\ m_1 + m_2 &= 4 \end{aligned}$$

Den drei Gleichungen entsprechen drei Gerade im $\mathbb{R}^2$. Kein gemeinsamer Schnittpunkt!

„In der Mitte des Fehlerdreiecks“, im Punkt $(4/3; 7/3)$ ist die Summe der Fehlerquadrate minimal.



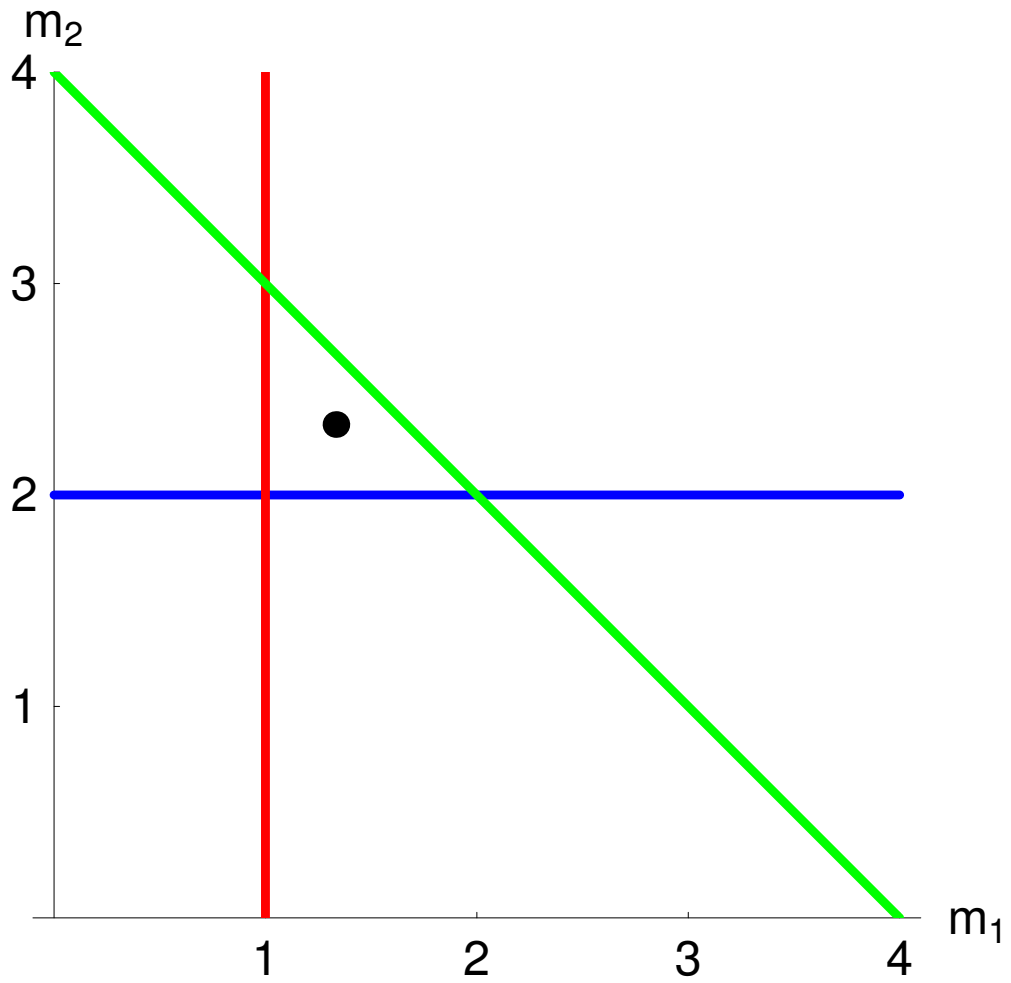
$$(m_1 - 1)^2 + (m_2 - 2)^2 + (m_1 + m_2 - 4)^2 \to \min!$$

Geometrische Interpretation

Zeilen des Gleichungssystems sind Geradengleichungen

$$\begin{aligned} m_1 &= 1 \\ m_2 &= 2 \\ m_1 + m_2 &= 4 \end{aligned}$$

Den drei Gleichungen entsprechen drei Gerade im $\mathbb{R}^2$. Kein gemeinsamer Schnittpunkt!
„In der Mitte des Fehlerdreiecks“, im Punkt $(4/3; 7/3)$ ist die Summe der Fehlerquadrate minimal.



$$(m_1 - 1)^2 + (m_2 - 2)^2 + (m_1 + m_2 - 4)^2 \to \min!$$

Methode der kleinsten Fehlerquadrate

Suche m_1, m_2 so dass

$$(m_1 - 1)^2 + (m_2 - 2)^2 + (m_1 + m_2 - 4)^2 \to \min!$$

Differenziere nach m_1 und m_2, setze Ableitungen gleich Null $\to$

$$\begin{array}{rcrl} 2m_1 & + & m_2 & = 5 \\ m_1 & + & 2m_2 & = 6 \end{array}$$

Führt man diese Rechnung allgemein für ein überbestimmtes System $A\mathbf{x} = \mathbf{b}$ durch, erhält man die Normalengleichungen $A^T A\mathbf{x} = A^T\mathbf{b}$.

hier:
$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}$$

Übersicht der Verfahren

für überbestimmte Systeme $A \cdot \mathbf{x} = \mathbf{b}$

Normalengleichungen $A^T \cdot A = A^T \mathbf{b}$ Klassischer Lösungsweg. Anfällig für Daten- und Rundungsfehler (schlechte Konditionszahl).

(noch nicht behandelt, kommt noch:)

QR-Zerlegung Standardverfahren zur numerischen Lösung. Algebraisch äquivalent zu Norm.gleichungen, aber numerisch weniger fehlerempfindlich.

MATLAB `x = A\b` verwendet bei überbestimmten Systemen automatisch QR-Zerlegung

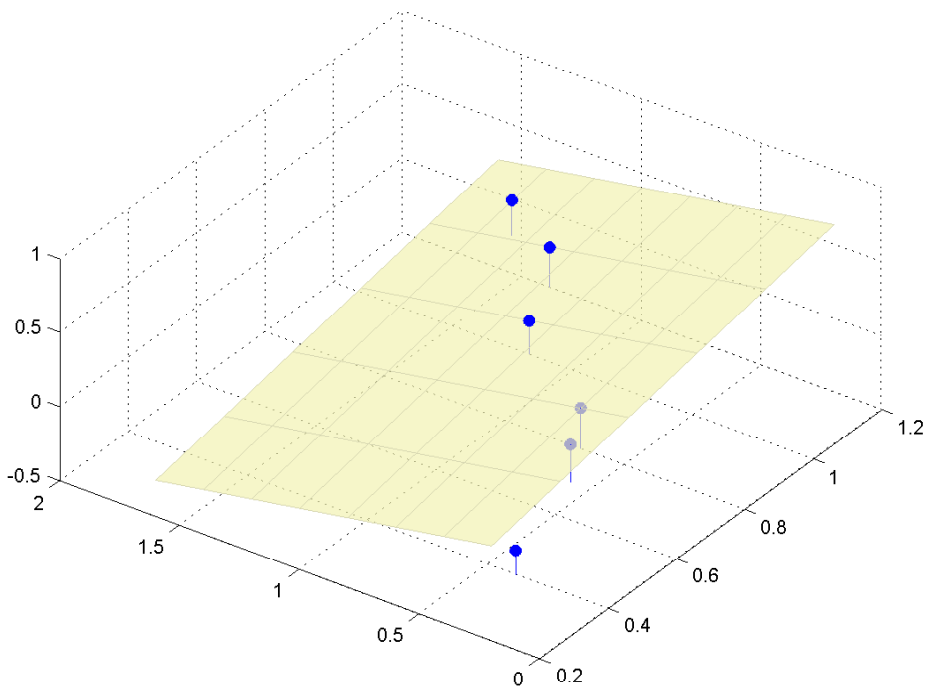
Sonderfälle ($\operatorname{rang} A \leq n$) Überbestimmte Systeme, die trotzdem exakt lösbar sind oder eine Schar von (ex. oder kl. Quadr.) Lösungen haben. Normalengleichungen können versagen. Methode: Singulärwert-Zerlegung.

Lineares Modell in zwei Variablen

Anpassen einer Ausgleichs-Ebene: Beispiel aus der Matlab-Hilfe

Angenommen, eine Größe y hängt von zwei Parametern x_1 und x_2 ab. Folgende Messwerte liegen vor:

x_1 :	0.2	0.5	0.6	0.8	1.0	1.1
x_2 :	0.1	0.3	0.4	0.9	1.1	1.4
y :	0.17	0.26	0.28	0.23	0.27	0.24



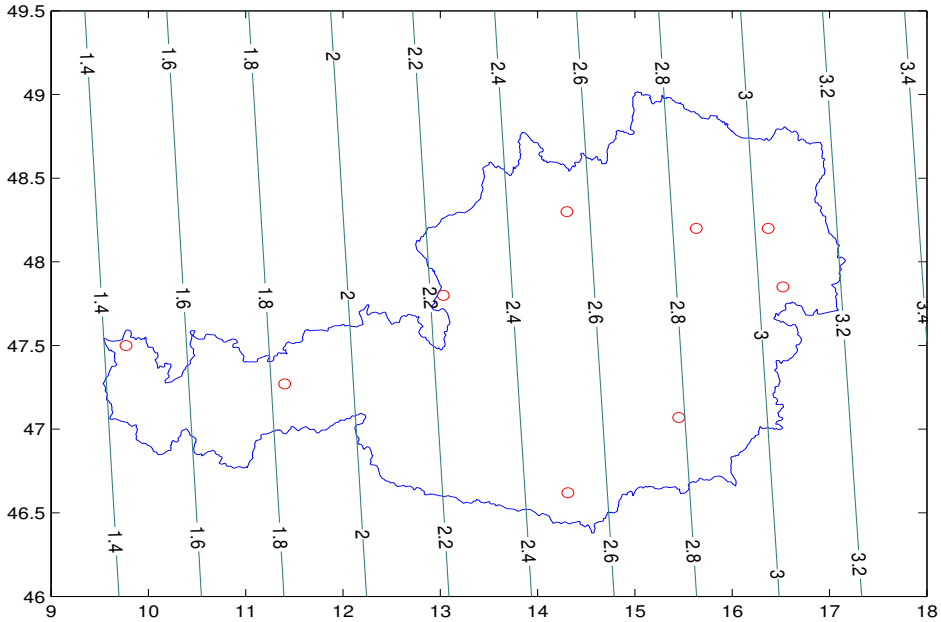
Wir nehmen ein lineares Modell $y = a_0 + a_1 x_1 + a_2 x_2$ an und setzen die gegebenen Datentripel ein

$\longrightarrow$ führt auf ein System von 6 linearen Gleichungen in den 3 unbekannten Koeffizienten a_0, a_1, a_2.

$\longrightarrow$ Kleinste-Quadrate-Lösung liefert Ebene mit „bestmöglicher“ Anpassung an Daten

Lineares Modell in zwei Variablen

Beispiel: Magnetische Deklinationswerte 2008.5 in Österreich



Wien	3°	00’
Eisenstadt	3°	02’
St.Pölten	2°	54’
Graz	2°	47’
Linz	2°	33’
Klagenfurt	2°	30’
Salzburg	2°	15’
Innsbruck	1°	53’
Bregenz	1°	24’

Daten: ZAMG

Kleinste-Quadrate-Anpassung

liefert Modell:

$$\delta = -2.0987 + 0.2365\lambda + 0.0261\phi$$