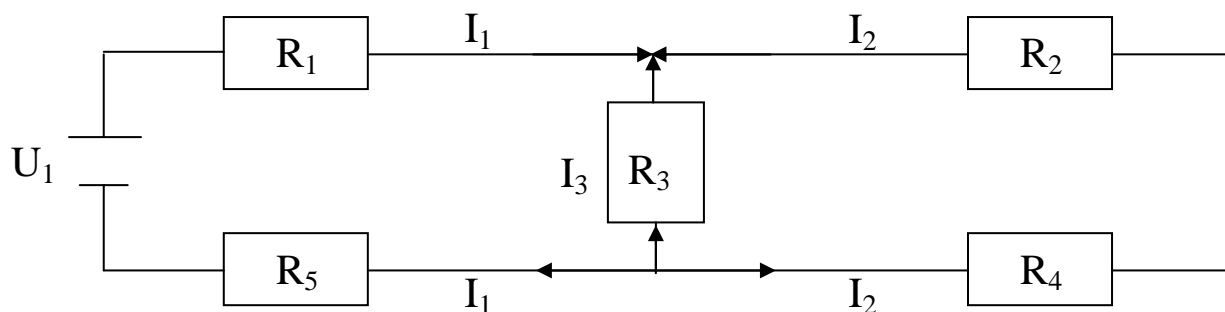


III. Lineare Gleichungssysteme

Beispiel: Elektrische Schaltkreise



Ohmsche Widerstände in Reihen- und Parallelschaltung.

Kirchhoff'sche Regeln:

- In jedem Knoten ist die Summe der zufließenden elektrischen Ströme gleich der Summe der abfließenden Ströme.
- In jeder Masche ist die Summe der elektrischen Spannungsabfälle gleich der Summe der in dieser Masche vorhandenen Quellenspannungen.

Mittels $\mathbf{U} = \mathbf{R} * \mathbf{I}$ ergibt dies drei lineare Gleichungen

$$I_1 + I_2 + I_3 = 0$$

$$R_1 I_1 - R_5 I_1 - R_3 I_3 = U_1$$

$$R_1 I_1 + R_5 I_1 - R_2 I_2 - R_4 I_2 = U_1$$

zur Berechnung der auftretenden Ströme:

$$\begin{pmatrix} 1 & 1 & 1 \\ R_1 - R_5 & 0 & -R_3 \\ R_1 + R_5 & -R_2 & -R_4 \end{pmatrix} \cdot \begin{pmatrix} I_1 \\ I_2 \\ I_3 \end{pmatrix} = \begin{pmatrix} 0 \\ U_1 \\ U_1 \end{pmatrix}$$

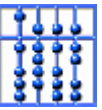
3.1 Dreiecksgleichungssysteme

Beispiel:

$$\begin{aligned} 10x_1 - 7x_2 + 0x_3 &= 7 \\ & 2.5x_2 + 5x_3 = 2.5 \\ & 6.2x_3 = 6.2 \end{aligned}$$

In Matrixschreibweise:

$$\begin{pmatrix} 10 & -7 & 0 \\ 0 & 2.5 & 5 \\ 0 & 0 & 6.2 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 2.5 \\ 6.2 \end{pmatrix}$$



Wegen der Dreiecksform lässt sich das System leicht von unten her auflösen:

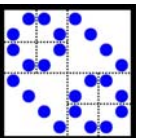
$$x_3 = 6.2 / 6.2 = 1; \quad \text{also } x_3 = 1;$$

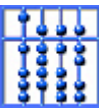
$$2.5x_2 = 2.5 - 5x_3 = -2.5, \quad \text{also } x_2 = -1;$$

$$10x_1 = 7 + 7x_2 = 7 - 7 = 0, \quad \text{also } x_1 = 0;$$

Lösungsvektor:
$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix};$$

Probe durch Einsetzen!





Allgemein:

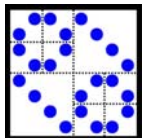
$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ & a_{22} & & a_{2n} \\ & & \ddots & \vdots \\ & & & a_{nn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

wird gelöst mittels **Programm 3.1.1.:**

$$x_n = b_n / a_{nn};$$

für $i = n - 1, n - 2, \dots, 1:$

$$x_i = \frac{b_i - \sum_{j=i+1}^n a_{ij} x_j}{a_{ii}}$$



Genauso wird das untere Dreieckssystem

$$\begin{pmatrix} a_{11} & & & \\ a_{21} & a_{22} & & \\ \vdots & & \ddots & \\ a_{n1} & \dots & \dots & a_{nn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

von oben her gelöst mit dem Programm:

$$x_1 = b_1 / a_{11};$$

für $i = 2, 3, \dots, n$:

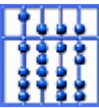
$$x_i = \frac{b_i - \sum_{j=1}^{i-1} a_{ij} x_j}{a_{ii}}$$

Wichtig:

Alle $a_{ii} \neq 0$, da sonst System nicht eindeutig lösbar!

$$\det(A) = a_{11} \cdot a_{22} \cdot \dots \cdot a_{nn} \neq 0$$

Determinante, Rang, linear abhängig, Basis,...?



3.2 Einschub: Rechnen mit Matrizen

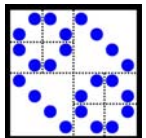
Matrix $A = \begin{pmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{pmatrix} \in \mathfrak{R}^{n,m}$ beschreibt Abbildung

$$\mathbf{x} \rightarrow \mathbf{f}(\mathbf{x}) = \mathbf{Ax} = \mathbf{b} .$$

Matrizen bilden kommutative Gruppe bzgl. +, bzw.

invertierbare $n \times n$ -Matrizen bilden sogar Gruppe bzgl. *
(aber nicht kommutativ!)

$$\mathbf{A}^{-1} * \mathbf{A} = \mathbf{I} = \mathbf{Einheitsmatrix} = \mathbf{Identitat} = \mathbf{1}$$

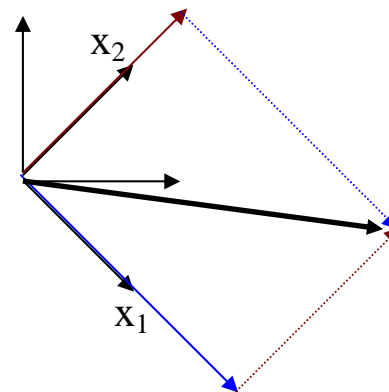
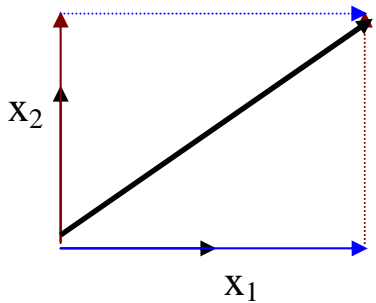


Beispiel:

Abbildung

$$f(x) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} * \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$$

Daher: $\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} * \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix} / \sqrt{2}$ und $\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} * \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} / \sqrt{2}$



Drehung um 45°

3.2.1. Matrix-Multiplikation: $A * B = C$

$$\begin{pmatrix} a_{11} & \cdots & a_{1k} \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ a_{n1} & \cdots & a_{nk} \end{pmatrix} * \begin{pmatrix} b_{11} & \overset{j}{\cdot} & b_{1m} \\ \vdots & \cdot & \vdots \\ b_{k1} & \cdot & b_{km} \end{pmatrix} = \begin{pmatrix} c_{11} & \cdots & c_{1m} \\ \vdots & \cdot & \vdots \\ c_{n1} & \cdots & c_{nm} \end{pmatrix} \quad i$$

mit

$$c_{ij} = \sum_{r=1}^k a_{ir} \cdot b_{rj}$$

Spezialfall:

$$a^T \cdot b = (a_1 \quad \cdots \quad a_n) \cdot \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix} = \sum_{j=1}^n a_j \cdot b_j \in \mathfrak{R}$$

ist Skalarprodukt (Inneres Produkt) der Vektoren a und b,



oder

$$a \cdot b^T = \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} \cdot (b_1 \quad \dots \quad b_m) = \begin{pmatrix} a_1 b_1 & \dots & a_1 b_m \\ \vdots & \ddots & \vdots \\ a_n b_1 & \dots & a_n b_m \end{pmatrix}$$

als Äußeres Produkt von a und b.

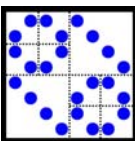
$A = a b^T$ ist Rang-1-Matrix, d.h. die durch A beschriebene Abbildung $f(x)$ hat eindimensionalen Bildraum:

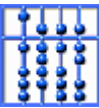
$$f(x) = A x = (a b^T) x = (b^T x) a = r(x) \cdot a$$

bildet die 2-dim. Ebene auf die Gerade durch den Vektor a ab.

Eine invertierbare $n \times n$ – Matrix A hat vollen Rang n und $\det(A) \neq 0$.

$$A^T = \left((a_{i,j})_{i,j=1}^{n,m} \right)^T = (a_{j,i})_{j,i=1}^{m,n} \leftrightarrow \text{Spiegelung an der Hauptdiagonalen}$$





3.2.2. Orthonormalbasis:

Vektoren u_j , $j=1, \dots, n$ mit $u_j^T u_k = 0$ für $j \neq k$
 $u_j^T u_k = 1$ für $j = k$

Dies sind n *linear unabhängige* Vektoren, also eine **Basis** des \mathbb{R}^n .

Norm: $\|x\|_2 = \sqrt{x^T x}$ ist die euklid'sche Länge.

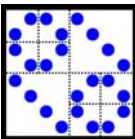
Eigenschaften: $\|x\| > 0$ für $x \neq 0$

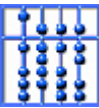
$$\|a \cdot x\| = |a| \cdot \|x\| \text{ für } a \in \mathbb{R}$$

$$\|x+y\| \leq \|x\| + \|y\|$$

und

$$|x^T y| \leq \|x\| \cdot \|y\|$$





Q ist orthogonale Matrix, wenn stets

$$\|Qx\|_2 = \|x\|_2 \quad \text{oder} \quad x^T Q^T Q x = x^T x, \text{ d.h.}$$

$$Q^T Q = I \quad \text{oder} \quad Q^{-1} = Q^T$$

Die Spalten (Zeilen) von Q bilden eine Orthonormalbasis!

Ein lineares Gleichungssystem **$Ax = b$** ist lösbar, falls

$$\text{Rang}(A) = \text{Rang}(A | b), \quad \text{d.h.}$$

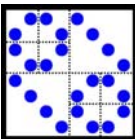
b ist durch Spalten von A darstellbar.

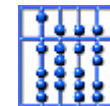
Denn mit der Lösung x (falls sie existiert) ist $b = Ax$ eine Linearkombination von Spalten von A:

$$b = A_{\cdot,1} x_1 + \cdots + A_{\cdot,n} x_n$$

Dann ist

$$\mathbf{x} = \text{inv}(A) * \mathbf{b} = \mathbf{A}^{-1} * \mathbf{b}$$





3.3 Gauss-Elimination

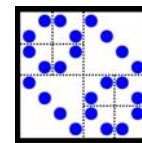
3.3.1. Beispiel:

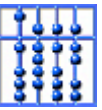
$$\begin{array}{rclcl} 10x_1 & - & 7x_2 & + & 0x_3 & = & 7 \\ -3x_1 & + & 2x_2 & + & 6x_3 & = & 4 \\ 5x_1 & - & x_2 & + & 5x_3 & = & 6 \end{array}$$

In Matrixschreibweise:

$$\begin{pmatrix} 10 & -7 & 0 \\ -3 & 2 & 6 \\ 5 & -1 & 5 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 4 \\ 6 \end{pmatrix}$$

Grundidee: Zurückführung auf den bekannten Fall eines Dreiecksgleichungssystems.



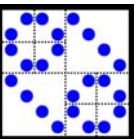


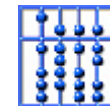
3.3.2. Erlaubte Transformationen:

1. **Multiplizieren einer Zeile (Gleichung) mit einer Zahl verschieden von Null.**
2. **Addieren eines Vielfachen einer Zeile zu einer anderen Zeile (Gleichung).**
3. **Vertauschen von Zeilen (Gleichungen), bzw. Spalten (Unbekannten) (entspricht Umnummerierung).**

Operationen sind dabei nicht nur an der Matrix durchzuführen, sondern ev. auch an der rechten Seite (Vektor b) und dem Lösungsvektor x .

Benutze diese Regeln, um in der Matrix die Subdiagonal-elemente der Reihe nach von oben nach unten, bzw. von links nach rechts, zu Null zu machen \rightarrow Dreieckssystem.





$$\begin{pmatrix} 10 & -7 & 0 \\ -3 & 2 & 6 \\ 5 & -1 & 5 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 4 \\ 6 \end{pmatrix}, \quad \begin{array}{l} \cdot (3/10) \\ \downarrow \\ + \end{array}$$

Zu Eliminieren: -3

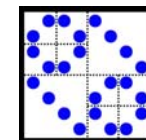
Addiere dazu zur zweiten Zeile die erste Zeile, multipliziert mit 3/10.

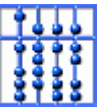
Ergibt:

$$\begin{pmatrix} 10 & -7 & 0 \\ 0 & -0.1 & 6 \\ 5 & -1 & 5 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 6.1 \\ 6 \end{pmatrix}, \quad \begin{array}{l} \cdot (-5/10) \\ \downarrow \\ \downarrow \\ + \end{array}$$

Danach soll 5 zu Null werden:

Dritte Zeile - 5/10 * Erste Zeile





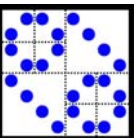
$$\begin{pmatrix} 10 & -7 & 0 \\ 0 & -0.1 & 6 \\ 0 & 2.5 & 5 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 6.1 \\ 2.5 \end{pmatrix}, \quad \begin{matrix} \cdot (2.5/0.1) \\ \downarrow + \end{matrix}$$

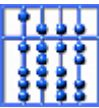
Eliminiere 2.5 in letzter Zeile.

Resultat:

$$\begin{pmatrix} 10 & -7 & 0 \\ 0 & -0.1 & 6 \\ 0 & 0 & 155 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 6.1 \\ 155 \end{pmatrix},$$

Dieses System kann nun von unten her aufgelöst werden, wie in Abschnitt 3.1. beschrieben.

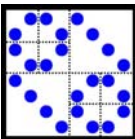


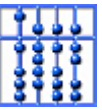


Benutze also jeweils das Diagonalelement, um die darunter liegenden Einträge Spalte für Spalte zu eliminieren, und zwar von a_{11} , a_{22} , bis a_{nn} .

**Problem: Es könnte irgendwann eine Null auf der Diagonalen auftreten: $a_{ii}=0$.
Was dann?**

Im Beispiel: Ersetze $a_{22} = 2$ durch $a_{22} = 2.1$





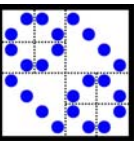
$$\begin{pmatrix} 10 & -7 & 0 \\ -3 & 2.1 & 6 \\ 5 & -1 & 5 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 4 \\ 6 \end{pmatrix}, \quad \begin{matrix} 3/10 & -5/10 \\ \downarrow & \downarrow \\ & \downarrow \end{matrix}$$

ergibt

$$\begin{pmatrix} 10 & -7 & 0 \\ 0 & 0 & 6 \\ 0 & 2.5 & 5 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 6.1 \\ 2.5 \end{pmatrix}.$$

Um Fortfahren zu können, ist eine Vertauschung notwendig, z.B. vertausche zweite Zeile mit dritter Zeile:

$$\begin{pmatrix} 10 & -7 & 0 \\ 0 & 2.5 & 5 \\ 0 & 0 & 6 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 2.5 \\ 6.1 \end{pmatrix} \quad \updownarrow$$



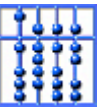
Problem gelöst!

Betrachten wir das ursprüngliche System, so sehen wir, dass wir zwar ohne Vertauschung durchkommen, aber der Wert -0.1 auf der Diagonalen a_{22} führt zu der großen Zahl 155 in der letzten Zeile.

Erinnerung: Zu vermeiden sind große Zwischenwerte!

Daher ist es auch im ursprünglichen System besser, die Vertauschung von zweiter und dritter Zeile vorzunehmen.

$$\begin{pmatrix} 10 & -7 & 0 \\ 0 & 2.5 & 5 \\ 0 & -0.1 & 6 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 2.5 \\ 6.1 \end{pmatrix} \quad \begin{array}{c} \updownarrow \\ \cdot 0.1/2.5 \\ \downarrow \\ + \end{array}$$



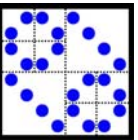
Dann lautet der letzte Eliminationsschritt

$$\begin{pmatrix} 10 & -7 & 0 \\ 0 & 2.5 & 5 \\ 0 & 0 & 6.2 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 2.5 \\ 6.2 \end{pmatrix}$$

Es treten keine großen Zwischenwerte mehr auf.

Allgemeines Vorgehen:

Pivotsuche

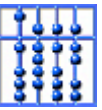


Sieht im k-ten Eliminationsschritt so aus:

$$\begin{pmatrix} a_{11} & \cdots & \cdots & \cdots & \cdots & a_{1n} \\ 0 & \ddots & & & & \vdots \\ \vdots & \ddots & & & & \vdots \\ & & a_{k-1,k-1} & & & \\ 0 & \cdots & 0 & \boxed{a_{kk}} & \cdots & a_{k,n} \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & a_{nk} & \cdots & a_{nn} \end{pmatrix}$$

Suche in **Untermatrix** ‚großen‘ Eintrag und vertausche entsprechend Zeilen und Spalten, so dass diese große Zahl an die Diagonal-Position $\boxed{a_{kk}}$ kommt.

Dieses Element heißt *Pivotelement*.



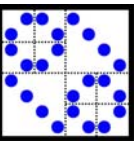
Gebräuchlichste Variante:

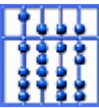
3.3.3. *Spaltenpivotsuche:*

Durchsuche nur die Spalte von a_{kk} bis a_{nk} nach betragsgrößtem Element a_{jk} und vertausche dann die gefundene Zeile j mit der k -ten Zeile.

Der Zusatzaufwand ist gering, da nur jeweils eine Spalte durchsucht werden muss, und zwei Zeilen (Gleichungen) vertauscht werden müssen.

Vertausche also zwei Zeilen in der Matrix und entsprechend in der rechten Seite b .





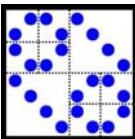
Weniger üblich *Zeilenpivotsuche:*

Durchsuche die k -te Zeile nach betragsgrößtem Element und vertausche zwei Spalten (= Umnummerierung in Vektor x), so dass das betragsgrößte Element der k -ten Zeile an die Diagonalposition kommt.

Totalpivotsuche:

Durchsuche die gesamte $n-k+1 \times n-k+1$ – Untermatrix und vertausche sowohl Spalten, als auch Zeilen, um das betragsgrößte Element an die Diagonalposition zu versetzen.

Aufwändig! Nur sinnvoll, wenn das Gleichungssystem sehr schlecht konditioniert ist! (Mehr dazu später)



3.3.4. Umwandlung auf Dreiecksform mit Spaltenpivotsuche: Algorithmus der Gauss-Elimination.

1. Teil des Programms: Pivotsuche und –vertauschung

```

FOR k=1,...,n-1 DO
  alpha = |a(k,k)|; j=k;
  FOR s=k+1,...,n DO
    IF |a(s,k)| > alpha THEN
      alpha = |a(s,k)|; j=s;
    ENDIF
  ENDFOR
# Pivotelement ist a(j,k) und Pivotzeile ist j
  FOR i=k,...,n DO
    alpha = a(k,i); a(k,i) = a(j,i); a(j,i) = alpha;
  ENDFOR
  alpha = b(j); b(j) = b(k); b(k) = alpha;

```

2. Teil: Eigentliche Elimination

```

# Eliminationsschritt
  FOR s=k+1,...,n DO
    l(s,k) = a(s,k)/a(k,k);
    b(s) = b(s) - l(s,k)b(k);
    FOR i=k+1,...,n DO
      a(s,i) = a(s,i) - l(s,k)a(k,i);
    ENDFOR
  ENDFOR
ENDFOR

```

Dadurch ist das System auf obere Dreiecksform gebracht, und kann wie in 3.1 beschrieben einfach von unten her gelöst werden.

3. 4 Die LU-Zerlegung

3.4.1. Definition:

Sei \mathbf{A}_k die Matrix, die im k-ten Schritt der Gauss-Elimination bearbeitet wird, also $\mathbf{A}_1 = \mathbf{A}$ die Ausgangsmatrix und $\mathbf{A}_n = \mathbf{U}$ die Endmatrix in oberer Dreiecksgestalt.

Die Gewichte $\mathbf{l}(\mathbf{s}, \mathbf{k}) = l_{s,k}$ aus obigem GE-Algorithmus schreiben wir in eine neue untere Dreiecksmatrix:

$$L := \begin{pmatrix} 1 & & & & & & \\ l_{2,1} & \ddots & & & & & \\ \vdots & \ddots & 1 & & & & \\ \vdots & & l_{k+1,k} & 1 & & & \\ \vdots & & \vdots & \ddots & \ddots & & \\ l_{n,1} & \cdots & l_{n,k} & \cdots & l_{n,n-1} & 1 & \end{pmatrix}$$

Für jede Spaltenelimination sammeln wir die Gewichte in

$$L_k := \begin{pmatrix} 0 & & & & & & \\ \vdots & \ddots & & & & & \\ \vdots & & 0 & & & & \\ \vdots & & l_{k+1,k} & 0 & & & \\ \vdots & & \vdots & \vdots & \ddots & & \\ \vdots & & l_{n,k} & 0 & \cdots & 0 & \\ 0 & & & & & & \end{pmatrix}$$

P sei die Permutationsmatrix zu den Pivotvertauschungen.

Beispiel: Vertauschung $1 \leftrightarrow 2$ entspricht $P = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

Im Folgenden betrachten wir zur Vereinfachung die Gauss-Elimination ohne Pivotsuche und ohne Permutation **P**!

Der k-te Schritt der GE kann als Matrixprodukt beschrieben werden in der Form

$$A_{k+1} = (I - L_k)A_k = A_k - L_k A_k \quad \text{mit Einheitsmatrix } I.$$

(Ziehe von den unteren Zeilen von A_k jeweils die entsprechend gewichtete k-te Zeile ab.)

$$L_k \cdot A_k = \begin{pmatrix} 0 & \dots & 0 \\ \cdot & 0 & \cdot \\ \cdot & l_{k+1,k} & \cdot \\ \vdots & \vdots & \vdots \\ 0 & l_{n,k} & 0 \end{pmatrix} * \begin{pmatrix} * \\ A_{k,.} \\ * \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ l_{k+1,k} * A_{k,.} \\ \vdots \\ l_{n,k} * A_{k,.} \end{pmatrix}$$

Dabei bezeichnet $A_{k,.}$ die k-te Zeile von A_k .

Insgesamt:

$$U = A_n = (I - L_{n-1})A_{n-1} = \dots = \overbrace{(I - L_{n-1}) \dots (I - L_1)}^{\tilde{L}} A_1 = \tilde{L}A$$

mit der Matrix

$$\tilde{L} := (I - L_{n-1}) \dots (I - L_2)(I - L_1)$$

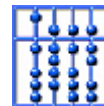
Wie sieht die Inverse dieser Matrix aus? Es gilt

$$(I + L_j) \cdot (I - L_j) = I - L_j + L_j - L_j^2 = I$$

Dabei verwenden wir: $L_i L_j = \mathbf{0}$ für $i \leq j$; $\begin{array}{|c|} \hline \\ \hline \end{array} * \begin{array}{|c|} \hline \\ \hline \end{array} = \mathbf{0}$

i j

Also:



$$\begin{aligned}\tilde{L}^{-1} &= \left((I - L_{n-1}) \cdots (I - L_1) \right)^{-1} = \\ &= (I - L_1)^{-1} \cdots (I - L_{n-1})^{-1} = \\ &= (I + L_1) \cdots (I + L_{n-1})\end{aligned}$$

und wegen

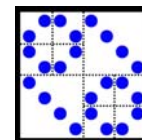
$$(I + L_1) \cdot (I + L_2) = I + L_1 + L_2 + L_1 L_2 = I + L_1 + L_2$$

dann auch

$$\tilde{L}^{-1} = (I + L_1) \cdots (I + L_{n-1}) = I + L_1 + L_2 + \cdots + L_{n-1} = L.$$

Damit ergibt sich:

$$U = \tilde{L}A \quad \leftrightarrow \quad A = LU$$





Allgemeiner mit Pivotsuche muss noch die Permutation **P** mitberücksichtigt werden, die durch das Auswählen der Pivotelemente hervorgerufen wird. Dann gilt:

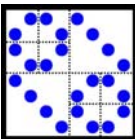
$$PA = LU$$

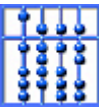
Allgemeines Vorgehen:

Sammele aus dem Algorithmus 3.3.4 der Gauss-Elimination die Gewichte $l_{j,k}$ in Matrix L und bezeichne mit U die obere Dreiecksmatrix, die sich als Endresultat der GE ergibt.

Sammele die Pivotvertauschungen in Permutationsmatrix P .

Wichtig: Eine Permutation im Rahmen der Pivotsuche muss auf ganz L angewendet werden! Danach erst nächste Elimination.





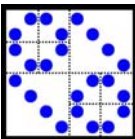
In dem betrachteten Beispiel:

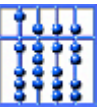
$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 10 & -7 & 0 \\ -3 & 2 & 6 \\ 5 & -1 & 5 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ -0.3 & -0.04 & 1 \end{pmatrix} \cdot \begin{pmatrix} 10 & -7 & 0 \\ 0 & 2.5 & 5 \\ 0 & 0 & 6.2 \end{pmatrix}$$

Vorteil: Nach einmaliger Durchführung und Speicherung von L und U kann jedes weitere Gleichungssystem in A schnell gelöst werden:

$$Ax = P^T L(Ux) = b \quad \rightarrow \quad Ly = Pb \quad \text{und} \quad Ux = y$$

Bem.: Speichere ev. L im Array von A an Stelle der entstehenden Nullen.





3.5 Die Kondition eines linearen Gleichungssystems (einer Matrix)

Neu benötigt: Matrixnorm und ihre Eigenschaften.

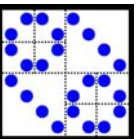
|| · || Matrixnorm: $||A|| > 0$ für $A \neq 0$
 $||a \cdot A|| = |a| \cdot ||A||$ für $a \in \mathbb{R}$
 $||A+B|| \leq ||A|| + ||B||$

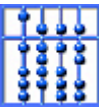
Besonders wichtig sind Matrixnormen, die zu einer Vektornorm „passen“ :

3.5.1. Eine Matrixnorm ist mit einer Vektornorm verträglich,

wenn $||Ax|| \leq ||A|| \cdot ||x||$.

Vektor
Matrix
Vektornorm





3.5.2. Eine Matrixnorm heißt submultiplikativ, wenn stets gilt

$$\|A \cdot B\| \leq \|A\| \cdot \|B\|$$

3.5.3. Eine submultiplikative Matrixnorm, verträglich mit einer vorgegebenen Vektornorm, kann leicht definiert werden

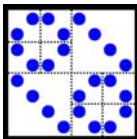
durch

$$\|A\| = \sup_{x \neq 0} \frac{\|Ax\|}{\|x\|} ,$$

die sog. Grenzen-Norm oder lub-Norm (least upper bound):

$$\frac{\|Ay\|}{\|y\|} \leq \sup_x \frac{\|Ax\|}{\|x\|} = \|A\| \Rightarrow \|Ay\| \leq \|A\| \cdot \|y\| \quad \forall y$$

und



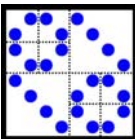


$$\begin{aligned} \|A \cdot B\| &= \sup \frac{\|ABx\|}{\|x\|} = \sup_x \left(\frac{\|ABx\|}{\|Bx\|} \cdot \frac{\|Bx\|}{\|x\|} \right) \leq \\ &\leq \sup_{y=Bx} \frac{\|Ay\|}{\|y\|} \cdot \sup_x \frac{\|Bx\|}{\|x\|} \leq \|A\| \cdot \|B\| \end{aligned}$$

So erhält man zu den Vektornormen $\|x\|_1 = \sum_{i=1}^n |x_i|$

und $\|x\|_\infty = \max_{i=1, \dots, n} |x_i|$

dazugehörige Matrix-Grenzen-Normen $\|\cdot\|_1$ und $\|\cdot\|_\infty$.





3.5.4. Das innere Produkt $(x, y) = x^T y = \sum x_i \cdot y_i$
führt auf die

3.5.5. Euklid'sche Norm: $\|x\|_2^2 = (x, x) = x^T \cdot x = \sum x_i^2$

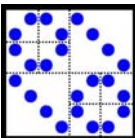
und die damit verträgliche **Matrixnorm**

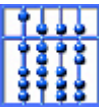
3.5.6. (2-Norm): $\|A\|_2 = \sup_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2}$

mit der Eigenschaft

$$\|Ax\|_2^2 = (Ax, Ax) = (Ax)^T (Ax) = x^T A^T Ax = x^T (A^T A)x$$

$$\|A\|_2^2 = \sup_{x \neq 0} \frac{\|Ax\|_2^2}{\|x\|_2^2} = \sup_{x \neq 0} \frac{x^T (A^T A)x}{x^T x} = \lambda_{\max}(A^T A)$$





Also
$$\|A\|_2 = \sqrt{\lambda_{\max}(A^T A)} = \sigma_{\max}(A)$$

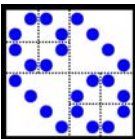
wobei $\lambda_{\max}(A^T A)$ der größte Eigenwert von $A^T A$ ist, und $\sigma_{\max}(A)$ der größte Singulärwert von A .

Diese Matrixnorm ist i.A. nicht einfach berechenbar!

3.5.7. Def. der Frobeniusnorm:
$$\|A\|_F := \sqrt{\sum |a_{j,k}|^2}$$

Fasse die Matrix A als Vektor auf und benutze für diesen langen Vektor die euklid'sche Vektornorm $\rightarrow \|\cdot\|_F$

Die Frobeniusnorm ist mit der euklid'scher Vektornorm verträglich und submultiplikativ.



3.6 Gestörte Eingabedaten

Für ein lineares Gleichungssystem $\mathbf{A} \mathbf{x} = \mathbf{b}$ mit Matrix \mathbf{A} , Vektor der rechten Seite \mathbf{b} und gesuchtem Lösungsvektor \mathbf{x} , untersuchen wir wieder den Einfluss von Eingabefehlern bei sonst exakter Rechnung

→ Kondition der Matrix

Eingabedaten mit Fehler: $b \rightarrow b + \Delta b = \tilde{b}$

Damit ergibt sich an Stelle der exakten Lösung x die Näherung

$$x \rightarrow x + \Delta x = \tilde{x}$$

Mit $\mathbf{A} \mathbf{x} = \mathbf{b}$ und $\mathbf{A} \tilde{\mathbf{x}} = \mathbf{A}(\mathbf{x} + \Delta \mathbf{x}) = \mathbf{b} + \Delta \mathbf{b} = \tilde{\mathbf{b}}$

gilt: $\mathbf{A} \Delta \mathbf{x} = \Delta \mathbf{b}$ **oder** $\Delta \mathbf{x} = \mathbf{A}^{-1} \Delta \mathbf{b}$

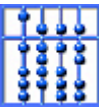
Die Matrix **A** wird hier als exakt angenommen!
 Damit erhält man die Ungleichungen

$$\|\Delta x\|_2 = \|A^{-1} \Delta b\|_2 \leq \|A^{-1}\|_2 \|\Delta b\|_2$$

$$\|b\|_2 = \|Ax\|_2 \leq \|A\|_2 \|x\|_2$$

wegen der Verträglichkeit der euklid'schen Vektor- und Matrixnorm.

Also auch $\frac{1}{\|x\|_2} \leq \frac{\|A\|_2}{\|b\|_2}$ und damit insgesamt:



3.6.1.

$$\frac{\|\Delta x\|_2}{\|x\|_2} \leq \left(\|A^{-1}\|_2 \cdot \|A\|_2 \right) \cdot \frac{\|\Delta b\|_2}{\|b\|_2}$$

**rel.Fehler
in x**

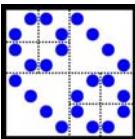
**Kondition
von A**

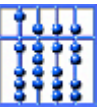
**rel. Fehler
in b**

3.6.2 Definition: Die Kondition der Matrix A bzgl. der euklid'schen Norm ist gegeben durch

$$cond_2(A) = \|A^{-1}\|_2 \cdot \|A\|_2$$

(Genauso kann man Konditionszahlen bzgl. anderer verträglicher Normen definieren.)





Die Konditionszahl beschreibt also wieder, wie sich ein relativer Eingabefehler in Vektor b auf das Resultat, den Lösungsvektor x , auswirkt.

$\text{cond}(A)$ groß \rightarrow kleine Störungen in b bewirken große Fehler in x

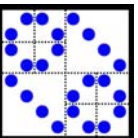
Man beachte:

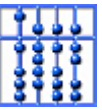
Der relative Fehler des Gesamtvektors wird abgeschätzt, nicht der Fehler einzelner Komponenten!

Als Vektor betrachtet ist $\begin{pmatrix} 10^{-16} \\ 1 \end{pmatrix} \approx \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, aber



komponentenweise ist natürlich $10^{-16} \neq 0$.





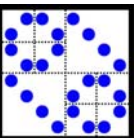
Beispiel: $A = \begin{pmatrix} 10^{-9} & 1 \\ 0 & 1 \end{pmatrix}$ und $A^{-1} = \begin{pmatrix} 10^9 & -10^9 \\ 0 & 1 \end{pmatrix}$

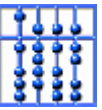
Normen: $\|A\|_2 \approx \sqrt{2}$ und $\|A^{-1}\|_2 \approx \sqrt{2} \cdot 10^9$

Also $cond_2(A) = 2 \cdot 10^9$, d.h. A hat große Konditionszahl.

Wähle speziell: $b = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, $x = A^{-1}b = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, $\Delta b = \begin{pmatrix} \Delta b_1 \\ 0 \end{pmatrix}$,

Dann folgt $\Delta x = 10^9 \cdot |\Delta b_1|$, $\|x\| = 1$, $\|b\| = \sqrt{2}$





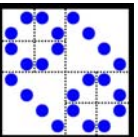
Damit ergibt sich

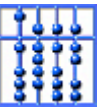
$$\frac{\|\Delta x\|}{\|x\|} = \frac{10^9 \cdot |\Delta b_1|}{1} = \sqrt{2} \cdot 10^9 \cdot \frac{\|\Delta b\|}{\|b\|}$$

und

$$\text{cond}(A) = \|A\| \cdot \|A^{-1}\| = \sqrt{2} \cdot \sqrt{2} \cdot 10^9 = 2 \cdot 10^9$$

Ein kleiner Fehler in der ersten Komponente von b wirkt sich verstärkt um Faktor 10^9 in der ersten Komponente von x aus.





3.6.3. Anmerkung:

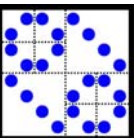
Speziell wichtige Klasse von Matrizen mit Konditionszahl 1:

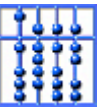
Q ist orthogonale Matrix $\leftrightarrow Q^{-1} = Q^T$
 oder $Q Q^T = I = Q^T Q$,

$$\|Qx\|_2^2 = x^T Q^T Qx = x^T x = \|x\|_2^2;$$

$$\|Qx\|_2 = \|x\|_2 \rightarrow \|Q\|_2 = \sup_{x \neq 0} \frac{\|Qx\|_2}{\|x\|_2} = 1$$

Und damit auch **cond₂(Q)=1**



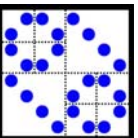


Außerdem gilt

$$\mathit{cond}_2(QA) = \mathit{cond}_2(A)$$

denn
$$\|QA\|_2 = \sup_{x \neq 0} \frac{\|QAx\|_2}{\|x\|_2} = \sup_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2} = \|A\|_2$$

Orthogonale Matrizen sind selbst gut konditioniert und lassen bei Multiplikation mit einer Matrix die Kondition der Ausgangsmatrix unverändert!



3.7 Kosten der Gauss-Elimination

3.7.1. Zunächst Dreieckssystem:

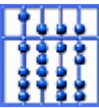
Programm:

```
FOR i = n, ..., 1 DO
  x(i) = b(i);
  FOR j=i+1, ..., n DO
    x(i) = x(i) - a(i,j)x(j);
  ENDFOR
  x(i) = x(i)/a(i,i);
ENDFOR
```

$$x_n = b_n / a_{nn};$$

für $i = n-1, n-2, \dots, 1$:

$$x_i = \frac{b_i - \sum_{j=i+1}^n a_{ij} x_j}{a_{ii}}$$



In jedem Schritt i fallen eine Division und jeweils $n-i$ Additionen und Multiplikationen an:

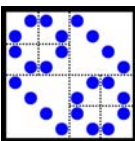
$$\text{Also } \sum_{i=1}^{n-1} (n-i) = \sum_{j=1}^{n-1} j = \frac{n(n-1)}{2} = \frac{n^2}{2} - \frac{n}{2}$$

Additionen und genauso viele Multiplikationen.
Dazu kommen n Divisionen.

3.7.2. Definition:

Unter flop verstehen wir eine elementare ‚floating point operation‘ (Gleitpunktoperation wie $+$, $-$, $*$, $/$).

Die Kosten eines Algorithmus werden üblicherweise in flop angegeben. Dazu gibt man in erster Näherung nur den Term höchster Ordnung an.





In unserem Fall:

$$2 \cdot \left(\frac{n^2}{2} - \frac{n}{2} \right) + n = n^2 \rightarrow n^2 \text{ flop's}$$

oder

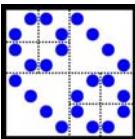
$$O(n^2) \text{ flop's}$$

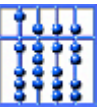
3.7.3. Exkurs: Landau'sche Symbole:

1. Betrachte Funktion in n für $n \rightarrow \infty$:

$$f(n) = O(g(n)), \quad \text{falls} \quad \frac{f(n)}{g(n)} \leq M < \infty \quad \text{für} \quad n \geq N$$

Beispiel: $n^2 + n = O(n^2)$; denn n^2 ist der am stärksten wachsende Term $\rightarrow f(n) / g(n) = 1 + 1/n \leq M$



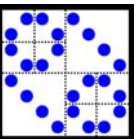


2. Betrachte Funktion in h für $h \rightarrow 0$:

$$f(h) = O(g(h)), \quad \text{falls} \quad \left| \frac{f(h)}{g(h)} \right| \leq c < \infty \quad \text{für} \quad |h| \leq \delta$$

Beispiel: $h^2 + h = O(h)$; denn h ist der am langsamsten schrumpfende Term $\rightarrow f(h) / g(h) = h + 1 \leq c$

$O(..)$ bezeichnet also jeweils den wichtigsten Term!



Andere, formale Möglichkeit zum Zählen von Operationen:

Schreibe Programm um als Zähler:

```

add = div = mult = 0;
FOR i=n,...,1 DO
    FOR j=i+1,...,n DO
        add = add + 1; mult = mult + 1;
    ENDFOR
    div = div + 1;
ENDFOR
    
```

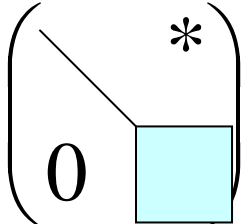
Damit ergibt sich mit $add := \text{Zahl der Additionen}$:

$$add = \sum_{i=n}^1 \sum_{j=i+1}^n 1 = \sum_{i=n}^1 (n-i) = \frac{n^2}{2} + O(n) \quad (\text{genauso mult und div})$$

3.7.4. Kosten der Gauss-Elimination:

Im k-ten Teilschritt arbeitet man in einer $(n-k) \times (n-k)$ Untermatrix A_k

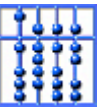
In dieser Matrix wird für $i=k+1, \dots, n$ neu berechnet :

$$a_{ij} = a_{ij} - l_{ik} a_{kj};$$


Das sind ca. $(n-k)^2$ Additionen und genauso viele Multiplikationen.

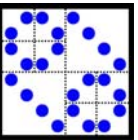
Insgesamt also

$$2 \sum_{k=1}^{n-1} (n-k)^2 = 2 \sum_{j=1}^{n-1} j^2 = \frac{2(n-1)(2n-1)n}{6} = \frac{2}{3} n^3 + O(n^2) \quad \text{flop's}$$



Dazu kommen $O(n^2)$ flop's für die Spaltenpivotsuche und $O(n^2)$ flop's für das Auflösen des Dreiecksgleichungssystems.

Diese Kosten fallen aber praktisch nicht ins Gewicht gegenüber den obigen $\frac{2}{3}n^3$.



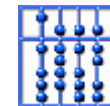
3.7.5 Beispiel zur Verdeutlichung der Kondition

$$A = \begin{pmatrix} 1 & 0 & \dots & 0 & 1 \\ -1 & 1 & \ddots & \vdots & 1 \\ -1 & -1 & \ddots & 0 & 1 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ -1 & -1 & \dots & -1 & 1 \end{pmatrix} \begin{matrix} \downarrow \\ \\ \\ \\ \end{matrix} \begin{matrix} + \\ \\ \vdots \\ \\ \downarrow \end{matrix}$$

Eliminiere die erste Spalte durch Addieren der ersten Zeile:

$$\begin{pmatrix} 1 & & & & 1 \\ 0 & 1 & & & 2 \\ 0 & -1 & 1 & & 2 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & -1 & \dots & -1 & 2 \end{pmatrix} \begin{matrix} \downarrow \\ \\ \\ \\ \end{matrix} \begin{matrix} + \\ \\ \vdots \\ \\ \downarrow \end{matrix}$$

Im nächsten Schritt

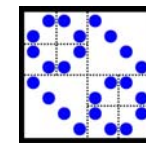


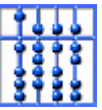
$$\begin{pmatrix} 1 & & & & 1 \\ 0 & 1 & & & 2 \\ 0 & 0 & 1 & & 4 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & -1 & 4 \end{pmatrix}$$

Und endlich

$$U = \begin{pmatrix} 1 & & & & 1 \\ & 1 & & & 2 \\ & & 1 & & 4 \\ & & & \ddots & \vdots \\ & & & & 1 & 2^{n-2} \\ & & & & & 2^{n-1} \end{pmatrix}$$

In jedem Schritt verdoppelt sich der größte Eintrag in der Matrix!
 Kondition von A selbst ist $\mathbf{O(n)}$, Kondition von U ist $\mathbf{O(2^{n-1})}$!





Im Verlauf der Gauss-Elimination kann die Kondition der Matrizen sehr stark anwachsen!

Aber: In der Praxis kommt das *so gut wie* nie vor!

Gauss-Elimination mit Pivotsuche *gilt als* numerisch stabil.

