3.3 Gauss-Elimination

3.3.1. Beispiel:

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

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:

- **Multiplizieren einer Zeile (Gleichung) mit einer Zahl verschieden von Null.**
- **Addieren eines Vielfachen einer Zeile zu einer anderen Zeile (Gleichung).**
- **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

→ Dreieckssystem.

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

Zu Eliminieren: -3

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

Ergibt:

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

Danach soll 5 zu Null werden:

Dritte Zeile - 5/10 * Erste Zeile

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

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 $a_{n-1,n-1}$.

Immer lösbar?

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

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

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

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

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: **<u>Zu vermeiden sind große Zwischenwerte!</u>**

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

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

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:

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

Suche in Untermatrix ‚großen' Eintrag und vertausche entsprechend Zeilen und Spalten, so dass diese große Zahl an die Diagonal-Position 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**.**

Keine Vertauschungen in b nötig!

***Totalpivotsuche*:**

Durchsuche die gesamte n-k+1 x n-k+1 – Untermatrix und vertausche sowohl Spalten, als auch Zeilen, um das betrags-größte 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

Idee: Faktorisierung von A in „einfache“ Matrizen.

3.4.1. Definition:

Sei $\mathbf{A_k}$ die Matrix, die im k-ten Schritt der Gauss-Elimination bearbeitet wird, also $\mathbf{A_1 = A}$ die Ausgangsmatrix und $\mathbf{A_n = U}$ die Endmatrix in oberer Dreiecksgestalt.

Die Gewichte $\mathbf{l(s,k)=l_{s,k}}$ 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

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

P sei die Permutationsmatrix zu den Pivotvertauschungen.

Beispiel: Vertauschung 1←→ 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

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

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

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

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

Insgesamt:

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

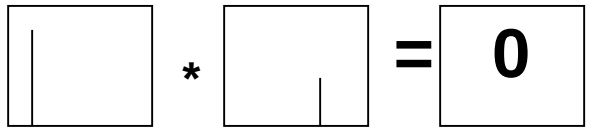
mit der Matrix

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

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: $\mathbf{L_i L_j = 0}$ für **i<=j;**



Also:

$$\tilde{L}^{-1} = \left((I - L_{n-1}) \cdots (I - L_1)\right)^{-1} =$$

$$= (I - L_1)^{-1} \cdots (I - L_{n-1})^{-1} =$$

$$= (I + L_1) \ldots (I + L_{n-1})$$

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

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

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:
Sammle 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.
Sammle die Pivotvertauschungen in Permutationsmatrix **P**.

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:

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

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 ≠ 0

||a*A|| = |a|*||A|| für a∈R

||A+B|| <= ||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 vorgegeben Vektornorm, kann leicht definiert werden durch

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

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

$$\frac{\|Ay\|}{\|y\|} \leq \sup_x \frac{\|Ax\|}{\|x\|} = \|A\| \Rightarrow \|Ay\| \leq \|A\| \cdot \|y\| \qquad \forall y \neq 0$$

Für $Bx \neq 0$:

$$\|A \cdot B\| = \sup \frac{\|ABx\|}{\|x\|} = \sup_x \left(\frac{\|ABx\|}{\|Bx\|} \cdot \frac{\|Bx\|}{\|x\|} \right) \leq$$

$$\leq \sup_{y=Bx} \frac{\|Ay\|}{\|y\|} \cdot \sup_x \frac{\|Bx\|}{\|x\|} \leq \|A\| \cdot \|B\|$$

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

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

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): $\|A\|_2 = \sup_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2}$ als maximale Längenänderung

mit der Eigenschaft

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

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

3.5.7. Anmerkung:

Speziell wichtige Klasse von Matrizen mit Norm 1:

Q ist orthogonale Matrix ←→ $Q^{-1} = Q^T$

oder $Q\,Q^T = I = Q^T Q$,

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

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

Basistransformation auf “ideales” Koordinatensystem:

$$\|A\|_2 = \|U^T \Lambda U\|_2 = \|\Lambda\|_2 = \max_{i=1,\ldots,n} |\lambda_i|$$

Eigenwertzerlegung von $A=A^T$. U ist orthogonale Matrix

Sei A nun eine allgemeine Matrix → SVD
Dann kann man die Singuläre Werte Zerlegung $A=U^T\Sigma V$ benutzen

Basis zu A^TA und AA^T → zwei „ideale“ Koordinatensysteme

$A = U^T \Sigma V$ Wobei U die Eigenvektoren von AA^T sind,
V die Eigenvektoren von A^TA, und
Σ, die sog. Singulären Werte von A
sind die Wurzeln aus den Eigenwerten
von A^TA und auch von AA^T.

$$\|A\|_2 = \|U^T \Sigma V\|_2 = \|\Sigma\|_2 = \max_{i=1,\ldots,n} \sqrt{|\lambda_i (A^T A)|} = \sigma_{\max}$$

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!

3.5.8. 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 → $\|.\|_F$

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

3.6 Gestörte Eingabedaten

Für ein lineares Gleichungssystem **A x = b** mit Matrix **A,** Vektor der rechten Seite **b** und gesuchtem Lösungsvektor **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 **Ax = b** und $A\tilde{x} = A(x + \Delta x) = b + \Delta b = \tilde{b}$

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

Die Matrix **A** wird hier als exakt angenommen!

Damit erhält man die Ungleichungen

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

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

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

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

3.6.1.

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

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

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

(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.

cond(A) groß → kleine Störungen in b bewirken große Fehler in x

Wieder: Numerisch stabil, wenn alle Zwischenmatrizen gut konditioniert sind!

$$cond_2(Q) = \| Q^{-1} \|_2 \| Q \|_2 = \| Q^T \|_2 \| Q \|_2 = 1$$

Außerdem gilt

$$cond_2(QA) = cond_2(A)$$

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!

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

Als Vektor betrachtet ist $\begin{pmatrix} 10^{-5} \\ 1 \end{pmatrix} \approx \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ 

Aber komponentenweise ist natürlich $10^{-5} \not\approx 0$

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: $\|A\|_2 \approx \sqrt{2}\,,\ \|A^{-1}\|_2 \approx \sqrt{2}\cdot 10^9\,,\ \ cond_2(A) \approx 2\cdot 10^9$

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

Dann folgt $|\Delta x_1| = 10^9 \cdot |\Delta b_1|, \Delta x_2 = 0 \quad \|x\| = 1, \quad \|b\| = \sqrt{2}$

Damit ergibt sich $\dfrac{\|\Delta x\|}{\|x\|} = \dfrac{10^9 \cdot |\Delta b_1|}{1} = \sqrt{2}\cdot 10^9 \cdot \dfrac{\|\Delta b\|}{\|b\|}$

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

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

3.7 Kosten der Gauss-Elimination

Zunächst Dreieckssystem:

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

$$für \quad i = n-1, n-2, \ldots, 1:$$

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

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
```

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

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 + , - , * , /). Die Kosten eines Algorithmus werden üblicherweise in flop angegeben. Dazu gibt man in erster Näherung nur den Term höchster Ordnung an.

In unserem Fall: $$2\cdot\left(\frac{n^2}{2}-\frac{n}{2}\right)+n=n^2 \quad\rightarrow\quad n^2 \qquad flop$$

oder $$O(n^2) \qquad flop$$

3.7.3. Exkurs: Landau'sche Symbole:

1. Betrachte Funktion in n für n → ∞ :

$$f(n) = \mathrm{O}(g(n)), \quad \textit{falls} \quad \left|\frac{f(n)}{g(n)}\right| \leq M < \infty \quad \textit{für} \quad n \geq N$$

Beispiel: $n^2 + n = \mathrm{O}(n^2)$; denn n^2 ist der am stärksten wachsende Term → f(n) / g(n) = 1 + 1/n <= M

2. Betrachte Funktion in h für h → 0 :

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

Beispiel: $h^2 + h = \mathrm{O}(h)$; denn h ist der am langsamsten schrumpfende Term → f(h) / g(h) = h + 1 <= c

O(..) bezeichnet also jeweils den dominanten Term!