Remarks on Floating Points

Prof. Dr. Jian-Jia Chen

Department of Computer Science, Chair 12
TU Dortmund University, Germany

(based on the slides from Sedgewick and Wayne from University of Princeton)

Floating Point

Hinweis: Die meisten reellen Zahlen können nicht exakt dargestellt werden, z.B. π oder $\frac{1}{10}$.
$\rightarrow$ Rundungsfehler entstehen aus nicht intuitivem Verhalten.

```
if (0.1 + 0.2 == 0.3) {  /* false */  }
if (0.1 + 0.3 == 0.4) {  /* true  */  }
```

Beispiel aus der Finanzwelt:
Die Gebühren eines Telefonanrufes (50 Cent) sollen um 9% erhöht werden.
$\rightarrow$ Kaufmännisches Runden

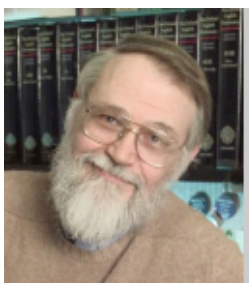
```
double a1 = 1.14 * 75;        // 85.49999999999999
double a2 = Math.round(a1); // 85 ← you lost 1¢

double b1 = 1.09 * 50;        // 54.50000000000001
double b2 = Math.round(b1); // 55 ← SEC violation (!)
```

Floating Point

Hinweis: Die meisten reellen Zahlen können nicht exakt dargestellt werden, z.B. π oder $\frac{1}{10}$.
$\rightarrow$ Rundungsfehler entstehen aus nicht intuitivem Verhalten.

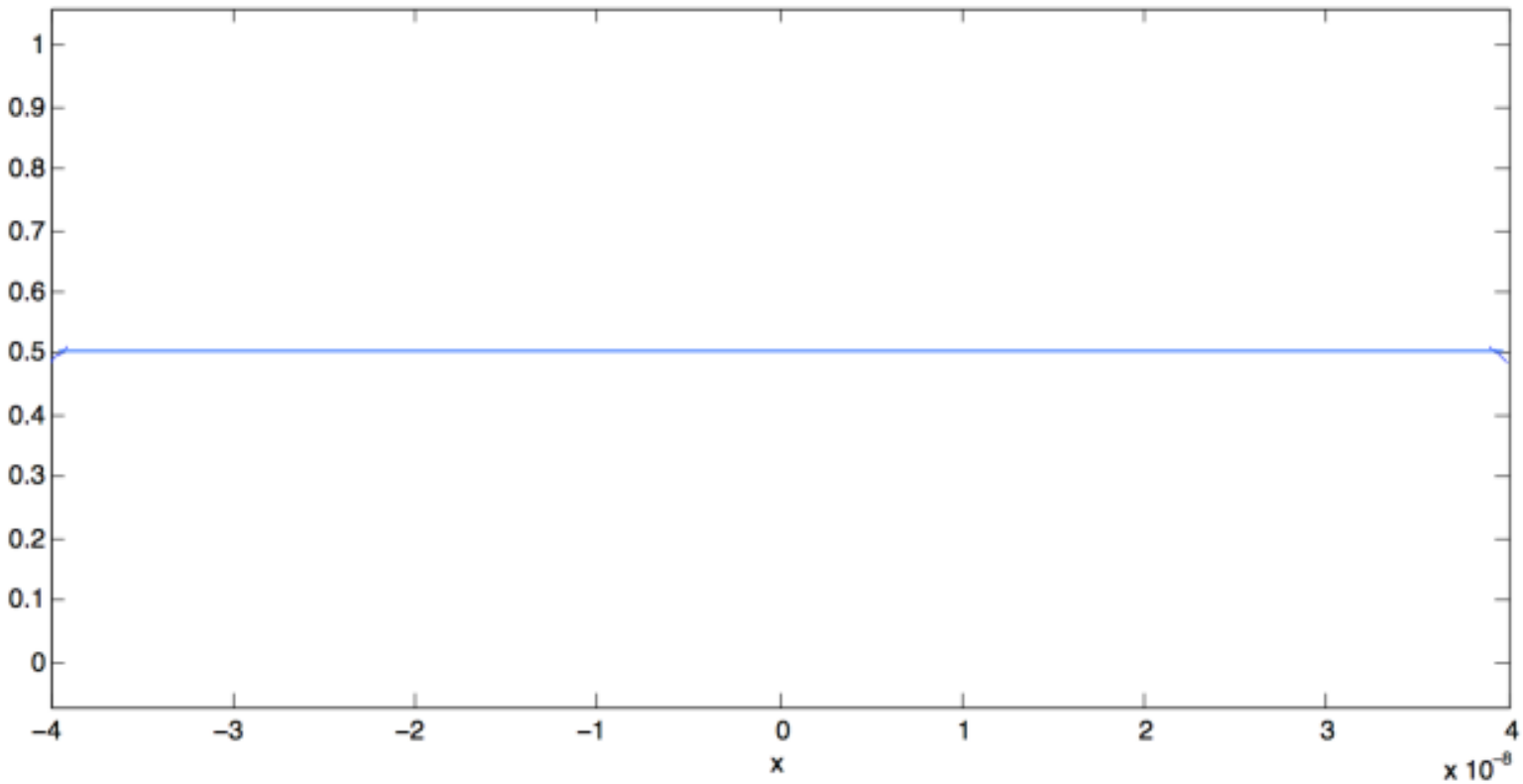
```
if (0.1 + 0.2 == 0.3) {  /* false */  }
if (0.1 + 0.3 == 0.4) {  /* true  */  }
```



“ Floating point numbers are like piles of sand; every time you move them around, you lose a little sand and pick up a little dirt. ” — Brian Kernighan and P. J. Plauger

Auslöschung

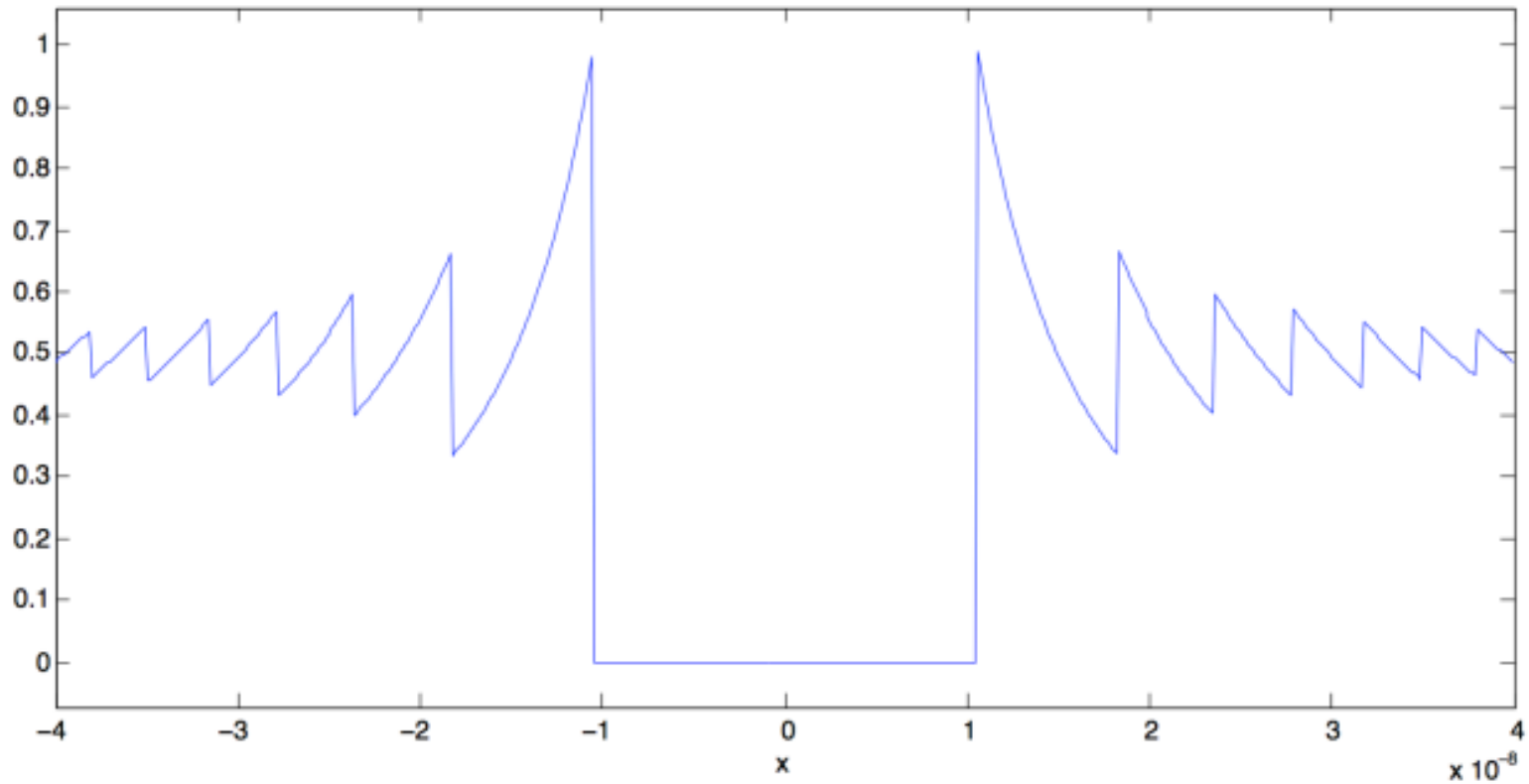
Gegeben: Sei die Funktion $f(x) = \frac{1-\cos x}{x^2}$
Gesucht: Die grafische Darstellung von $f(x)$ für die Werte $-4 \cdot 10^{-8} \leq x \leq 4 \cdot 10^{-8}$.



Exakte Darstellung

Auslöschung

Gegeben: Sei die Funktion $f(x) = \frac{1-\cos x}{x^2}$
Gesucht: Die grafische Darstellung von $f(x)$ für die Werte $-4 \cdot 10^{-8} \leq x \leq 4 \cdot 10^{-8}$.



Darstellung nach Standard IEEE 754

Auslöschung

```
public static double fl(double x) {
    return (1.0 - Math.cos(x)) / (x* x);
}
```

Beispiel: Evaluierung von $fl(x)$ für $x = 1.1 \cdot 10^{-8}$.

- $Math.cos(x)$ $= \mathbf{0.99999999999999}88897697537484$
- $1.0 - Math.cos(x)$ $= 1.1102 \cdot 10^{-16}$
- $\frac{1.0 - Math.cos(x)}{x \cdot x}$ $= 0.9175$

Unter *Auslöschung* (engl. cancellation) versteht man in der Numerik den Verlust an Genauigkeit bei der Subtraktion fast gleich großer Gleitkommazahlen.

Quelle: [Wolfgang Dahmen, Arnold Reusken: Numerik für Ingenieure und Naturwissenschaftler. 2. Auflage. Springer-Verlag, Berlin 2008, ISBN 978-3-540-76492-2, S. 41]

Berühmt-berüchtigte Softwarefehler

Ariane 5 Rakete, 04. Juni 1996

- Ein Projekt der europäische Weltraumorganisation ESA (10 Jahre Entwicklung, ca. 7.000.000.000$) explodiert direkt nach dem Start,
- wegen einer 64-bit float zu 16-bit signed int Konvertierung.

Aktien Handel in Vancouver (Kanada), November 1983

- Im Januar 1982 wurde der Index mit einem Wert von 1000 initialisiert,
- und bei jedem Handel (ca. 3000 mal pro Tag) bis auf drei Stellen hinter dem Komma abgeschnitten.
- Nach 22 Monaten entstand ein *truncation error* von mehr als 44%.

Lineare Gleichungssysteme (LGS)

$$\begin{cases} a_1x + b_1y + c_1z = d_1 \\ a_2x + b_2y + c_2z = d_2 \\ a_3x + b_3y + c_3z = d_3 \end{cases}$$

$$A = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix}, b = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix}$$

Matrixform: Finde x, sodass $Ax = b$

Grundlegende Probleme der Natur- und Ingenieurswissenschaften:

- Chemisches Gleichgewicht
- Lineare und nicht lineare Optimierung
- Kirchhoffsche Regeln
- Hookesches Gesetz
- Numerische Lösungen für Differenzialgleichungen
- ...

Dreiecksmatrix

Für eine obere Dreiecksmatrix gilt: $a_{ij} = 0$ for $i > j$

$$\begin{cases} 2x_0 + 4x_1 - 2x_2 = 2 \\ 0x_0 + 1x_1 + 1x_2 = 4 \\ 0x_0 + 0x_1 + 12x_2 = 24 \end{cases}$$

Rückwärtseinsetzen:

- Gleichung 2: $x_2 = \frac{24}{2} = 2$
- Gleichung 1: $x_1 = 4 - x_2 = 2$
- Gleichung 0: $x_0 = \frac{2-4x_1+2x_2}{2} = 1$

```
for (int i = N-1; i >= 0; i--) {
    double sum = 0.0;
    for (int j = i+1; j < N; j++)
        sum += A[i][j] * x[j];
    x[i] = (b[i] - sum) / A[i][i];
}
```

Gaußsches Eliminationsverfahren

$$\begin{cases} 0x_0 + 1x_1 + 1x_2 = 4 \\ 2x_0 + 4x_2 - 2x_2 = 2 \\ 0x_0 + 3x_2 + 15x_2 = 36 \end{cases}$$

$\downarrow$ Tausche Reihe0 mit Reihe1 $\downarrow$

$$\begin{cases} 2x_0 + 4x_2 - 2x_2 = 2 \\ 0x_0 + 1x_1 + 1x_2 = 4 \\ 0x_0 + 3x_2 + 15x_2 = 36 \end{cases}$$

$\downarrow$ Reihe3 $-$ (Reihe2 $\cdot$ 3) $\downarrow$

$$\begin{cases} 2x_0 + 4x_2 - 2x_2 = 2 \\ 0x_0 + 1x_1 + 1x_2 = 4 \\ 0x_0 + 0x_2 + 12x_2 = 24 \end{cases}$$

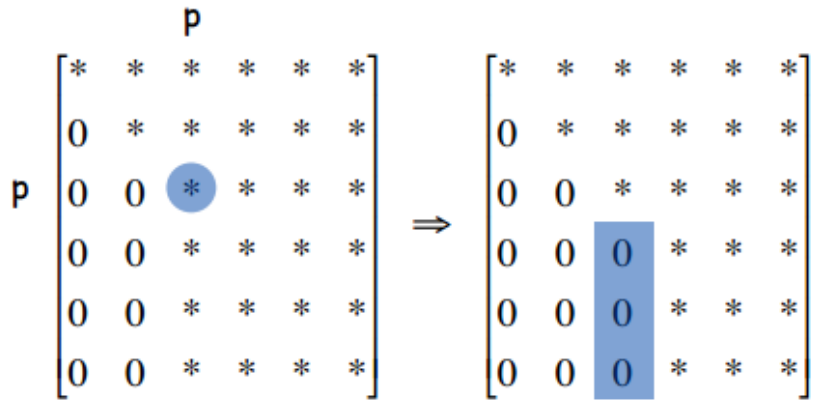
Gaußsches Eliminationsverfahren: Vorwärtseinsetzen

Arithmetischer Operationen nutzen, um eine obere Dreiecksmatrix zu erzeugen.

Pivotisierung: Einträge unter dem Pivotelement a_{pp} Nullen.

$$a_{ij} = a_{ij} - \frac{a_{ip}}{a_{pp}} a_{pj}$$

$$b_i = b_i - \frac{a_{ip}}{a_{pp}} b_p$$

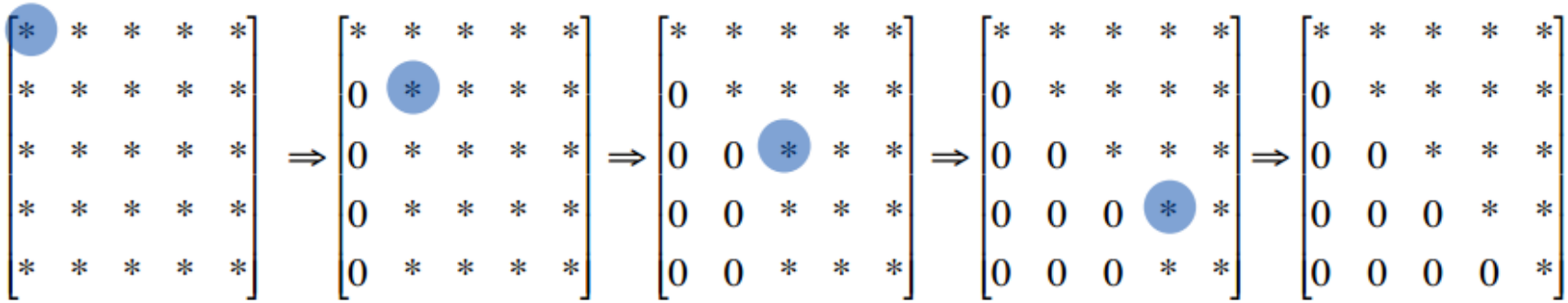



```
for (int i = p + 1; i < N; i++) {
   double alpha = A[i][p] / A[p][p];
   b[i] -= alpha * b[p];
   for (int j = p; j < N; j++)
      A[i][j] -= alpha * A[p][j];
}
```

Gaußsches Eliminationsverfahren: Vorwärtseinsetzen

Arithmetischer Operationen nutzen, um eine obere Dreiecksmatrix zu erzeugen.

Pivotisierung: Einträge unter dem Pivotelement a_{pp} Nullen.



```
for (int p = 0; p < N; p++) {
   for (int i = p + 1; i < N; i++) {
      double alpha = A[i][p] / A[p][p];
      b[i] -= alpha * b[p];
      for (int j = p; j < N; j++)
         A[i][j] -= alpha * A[p][j];
   }
}
```

Gaußsches Eliminationsverfahren: Beispiel

$$
\begin{array}{rcrcrcrcr}
1\,x_0 & + & 0\,x_1 & + & 1\,x_2 & + & 4\,x_3 & = & 1 \\
2\,x_0 & + & -1\,x_1 & + & 1\,x_2 & + & 7\,x_3 & = & 2 \\
-2\,x_0 & + & 1\,x_1 & + & 0\,x_2 & + & -6\,x_3 & = & 3 \\
1\,x_0 & + & 1\,x_1 & + & 1\,x_2 & + & 9\,x_3 & = & 4
\end{array}
$$

Gaußsches Eliminationsverfahren: Beispiel

$1\,x_0$	+	$0\,x_1$	+	$1\,x_2$	+	$4\,x_3$	=	1
$0\,x_0$	+	$-1\,x_1$	+	$-1\,x_2$	+	$-1\,x_3$	=	0
$0\,x_0$	+	$1\,x_1$	+	$2\,x_2$	+	$2\,x_3$	=	5
$0\,x_0$	+	$1\,x_1$	+	$0\,x_2$	+	$5\,x_3$	=	3

Gaußsches Eliminationsverfahren: Beispiel

$$
\begin{array}{rcrcrcrcr}
1\,x_0 & + & 0\,x_1 & + & 1\,x_2 & + & 4\,x_3 & = & 1 \\
0\,x_0 & + & -1\,x_1 & + & -1\,x_2 & + & -1\,x_3 & = & 0 \\
0\,x_0 & + & 0\,x_1 & + & 1\,x_2 & + & 1\,x_3 & = & 5 \\
0\,x_0 & + & 0\,x_1 & + & -1\,x_2 & + & 4\,x_3 & = & 3
\end{array}
$$

Gaußsches Eliminationsverfahren: Beispiel

$$
\begin{array}{rcrcrcrcr}
1\,x_0 & + & 0\,x_1 & + & 1\,x_2 & + & 4\,x_3 & = & 1 \\
0\,x_0 & + & -1\,x_1 & + & -1\,x_2 & + & -1\,x_3 & = & 0 \\
0\,x_0 & + & 0\,x_1 & + & 1\,x_2 & + & 1\,x_3 & = & 5 \\
0\,x_0 & + & 0\,x_1 & + & 0\,x_2 & + & 5\,x_3 & = & 8
\end{array}
$$

Gaußsches Eliminationsverfahren: Beispiel

$$
\begin{array}{rcrcrcrcr}
1\,x_0 & + & 0\,x_1 & + & 1\,x_2 & + & 4\,x_3 & = & 1 \\
0\,x_0 & + & -1\,x_1 & + & -1\,x_2 & + & -1\,x_3 & = & 0 \\
0\,x_0 & + & 0\,x_1 & + & 1\,x_2 & + & 1\,x_3 & = & 5 \\
0\,x_0 & + & 0\,x_1 & + & 0\,x_2 & + & 5\,x_3 & = & 8
\end{array}
$$

$$
\begin{array}{lll}
x_3 & = & 8/5 \\
x_2 = 5 - x_3 & = & 17/5 \\
x_1 = 0 - x_2 - x_3 & = & -25/5 \\
x_0 = 1 - x_2 - 4x_3 & = & -44/5
\end{array}
$$

Gaußsches Eliminationsverfahren: Teilpivotisierung

Hinweis: Der Code schlägt fehl, wenn das Pivotelement $a_{pp} = 0$.

$$
\begin{array}{rcrcrcr}
1\,x_0 & + & 1\,x_1 & + & 0\,x_3 & = & 1 \\
2\,x_0 & + & 2\,x_1 & + & -2\,x_3 & = & -2 \\
0\,x_0 & + & 3\,x_1 & + & 15\,x_3 & = & 33
\end{array}
$$

$$
\begin{array}{rcrcrcr}
1\,x_0 & + & 1\,x_1 & + & 0\,x_3 & = & 1 \\
0\,x_0 & + & 0\,x_1 & + & -2\,x_3 & = & -4 \\
0\,x_0 & + & 3\,x_1 & + & 15\,x_3 & = & 33
\end{array}
$$

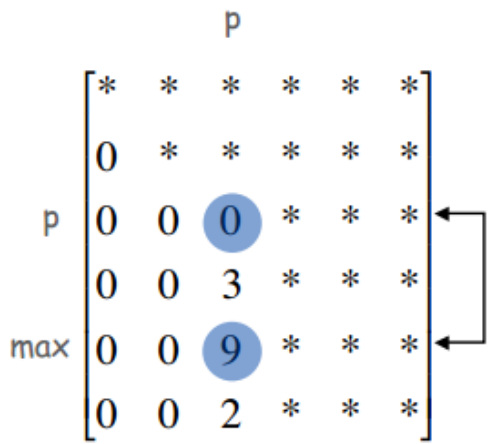
$$
\begin{array}{rcrcrcr}
1\,x_0 & + & 1\,x_1 & + & 0\,x_3 & = & 1 \\
0\,x_0 & + & 0\,x_1 & + & -2\,x_3 & = & -4 \\
0\,x_0 & + & \mathrm{Nan}\,x_1 & + & \mathrm{Inf}\,x_3 & = & \mathrm{Inf}
\end{array}
$$

Gaußsches Eliminationsverfahren: Teilpivotisierung

Teilpivotisierung: Tausche Reihe p mit der Reihe, welche den größten Eintrag in Spalte p entlang der Reihen i unter der Diagonalen hat.

```
// find pivot row
int max = p;
for (int i = p + 1; i < N; i++)
if (Math.abs(A[i][p]) > Math.abs(A[max][p]))
   max = i;

// swap rows p and max
double[] T = A[p]; A[p] = A[max]; A[max] = T;
double   t = b[p]; b[p] = b[max]; b[max] = t;
```



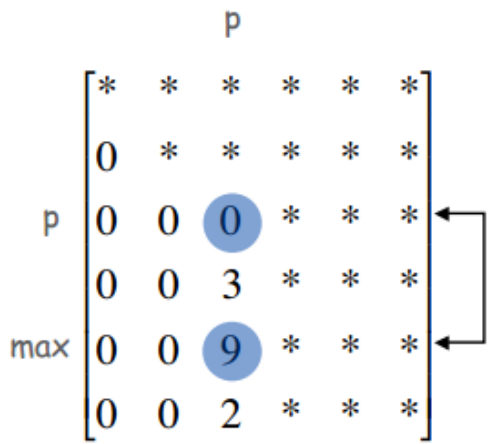
Frage: Was passiert, wenn Pivotelement $a_{pp} = 0$ während der Teilpivotisierung?

Gaußsches Eliminationsverfahren: Teilpivotisierung

Teilpivotisierung: Tausche Reihe p mit der Reihe, welche den größten Eintrag in Spalte p entlang der Reihen i unter der Diagonalen hat.

```
// find pivot row
int max = p;
for (int i = p + 1; i < N; i++)
if (Math.abs(A[i][p]) > Math.abs(A[max][p]))
   max = i;

// swap rows p and max
double[] T = A[p]; A[p] = A[max]; A[max] = T;
double   t = b[p]; b[p] = b[max]; b[max] = t;
```



Frage: Was passiert, wenn Pivotelement $a_{pp} = 0$ während der Teilpivotisierung?

Antwort: Das Gleichungssystem hat keine oder unendliche viele Lösungen.

Stabilität

Stabilität: Algorithmus $fl(x)$ um $f(x)$ zu berechnen ist numerisch stabil, wenn $fl(x) \approx f(x + \epsilon)$ für einige geringe Störungen ϵ.

Beispiel 1: Numerisch instabiler weg zur Berechnung von $f(x) = \frac{1-\cos x}{x^2}$.

```
public static double fl(double x) {
    return (1.0 - Math.cos(x)) / (x* x);
}
```

$fl(1.1 \cdot 10^{-8}) = 0.9175$ (eigentlich $\approx \frac{1}{2}$)

Hinweis: Numerisch stabile Gleichung von $f(x) = \frac{2\sin^2(\frac{x}{2})}{x^2}$

Stabilität

Stabilität: Algorithmus $fl(x)$ um $f(x)$ zu berechnen ist numerisch stabil, wenn $fl(x) \approx f(x + \epsilon)$ für einige geringe Störungen ϵ.

Beispiel 2: Gaußsches Eliminationsverfahren ohne Teilpivotierung kann fehlschlagen.

$a = 10^{-17}$

$$\begin{aligned} a\,x_0 &+ 1\,x_1 = 1 \\ 1\,x_0 &+ 2\,x_1 = 3 \end{aligned}$$

Algorithm	x_0	x_1
no pivoting	0.0	1.0
partial pivoting	1.0	1.0
exact	$\frac{1}{1-2a} \approx 1$	$\frac{1-3a}{1-2a} \approx 1$

Theorem: Teilpivotierung verbessert die numerische stabilität.

Schlecht konditionierte Probleme

Konditionierung: Ein Problem ist gut konditioniert, wenn $f(x) \approx f(x + \epsilon)$ für alle geringen Störeinflüsse ϵ.

Beispiel 1: arccos() and tan() Funktionen.
$\arccos(.99999991) \approx 0.000425$, $\tan(1.57078) \approx 6.12490 \cdot 10^5$
$\arccos(.99999992) \approx 0.000400$, $\tan(1.57079) \approx 6.12490 \cdot 10^4$

Folgerung: Die folgende Gleichung, um die Großkreisentfernung zwischen (x_1, y_1) and (x_2, y_2), ist ungenau für benachbarte Punkte!

$$d = 60 \arccos(\sin x_1 \sin x_2 + \cos x_1 \cos x_2 \cos(y_1 - y_2))$$

Schlecht konditionierte Probleme

Konditionierung: Ein Problem ist gut konditioniert, wenn $f(x) \approx f(x+\epsilon)$ für alle geringen Störeinflüsse ϵ.

Beispiel 2: Hilbert-Matrix.

- Kleine Störungen in H_n machen die Matrix singulär.
- $H_{12}x = b$ kann nicht durch die Benutzung von floating points gelöst werden.

$$H_n = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n+1} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \cdots & \frac{1}{n+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n} & \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{2n-1} \end{pmatrix}$$

Konditionszahl: Die Konditionszahl stellt ein Maß für die Abhängigkeit zwischen der Lösung eines Problems und der Störung der Eingangsdaten dar; sie beschreibt den Faktor, um den der Eingangsfehler im ungünstigsten Fall verstärkt wird.

Fazit

Präzision ist von Stabilität sowie Kondition abhängig!

- Gefährlich: Anwendung eines instabilen Algorithmus auf ein gut konditioniertes Problem.
- Gefährlich: Anwendung eines stabilen Algorithmus auf ein schlecht konditioniertes Problem.
- Sicher: Anwendung eines stabilen Algorithmus auf ein gut konditioniertes Problem.

Numerische Analyse ist die Kunst bzw. Wissenschaft einen numerisch stabilen Algorithmus für ein gut konditioniertes Problem zu entwerfen.

Fazit

Merke:

1. Manche Algorithmen sind nicht geeignet für floating-point Berechnungen.
2. Floating-point Berechnungen sind ungeeignet für manche Probleme.