Lineare Gleichungssysteme
Gauss-Elimination

Olaf Schenk

Departement Informatik, Universität Basel

http://informatik.unibas.ch

Lineare Gleichungssysteme

Direkte Verfahren zur Lösung linearer Gleichungssysteme

- Vorbemerkungen
- Das Lösen von Dreieckssystemen
- Gauss-Elimination / Block Gauss-Elimination
- Pivotisierung
- Cholesky und LDL^T Zerlegung

Lineare Gleichungssysteme

Vorbemerkungen

- The simplest model in applied mathematics is a system of linear equations. It is also by far the most important one (Gilbert Strang, Introduction to Applied Mathematics, CA, 1992).
- Spötter behaupten, dass numerische Berechnungen ausschliesslich aus dem Lösen linearer Systeme bestehen. Richtig ist, dass sehr viele Aufgaben im Bereich Computational Science and Engineering durch Diskretisierung und Linearisierung auf das Lösen solcher Systeme reduziert werden.
- Bei der impliziten Lösung der Wärmeleitungsgleichung wird z.B. in jeder Iteration ein lineares Gleichungssystem gelöst.
- Typisch ist hier, dass diese Systeme nur Hilfsmittel sind, um ganz andere Probleme zu lösen.
- Typisch ist aber auch, dass die Rechenzeiten, die zur Berechnung der Simulationen erforderlich sind, überwiegend zur Lösung linearer Systeme eingesetzt werden.

Vorbemerkungen

$$\begin{array}{ccccccccc} a_{1,1}x_1 & + & a_{1,2}x_1 & + & \cdots & + & a_{1,n}x_n & = & b_1 \\ a_{2,1}x_1 & + & a_{2,2}x_1 & + & \cdots & + & a_{2,n}x_n & = & b_2 \\ & & & & \vdots & & & & \\ a_{m,1}x_1 & + & a_{m,2}x_1 & + & \cdots & + & a_{m,n}x_n & = & b_m \end{array}$$

ist ein **System von m linearen Gleichungen in den n Unbekannten** $x_1, x_2, \ldots, x_n$**.** Kurzschreibweise

$$Ax = b$$

mit

$$A = [a_{i,j}]_{1 \le i \le m, 1 \le j \le n} \in \mathbb{R}$$

$$x \in [x_1, x_2, \ldots, x_n]^T \in \mathbb{R}, \quad b \in [b_1, b_2, \ldots, b_m]^T \in \mathbb{R}.$$

A ist die **Koeffizientenmatrix**, x ist die **Lösung** und b die **rechte Seite** des linearen Gleichungssystemes.

Vorbemerkungen

Geometrische Deutung

Jede der m Gleichungen repräsentiert eine Hyperebene im $\mathbf{R}^n$:

$$A(i,:) \cdot \mathbf{x} = b_i \quad \text{mit} \quad A(i,:) = [a_{i,1}, a_{i,2}, \ldots, a_{i,n}] \quad (i = 1, 2, 3, \ldots, m).$$

Gesucht ist der Durchschnitt dieser Hyperebenen.

Analytische Deutung

Mit $A(:,j) = [a_{1,j}, a_{2,j}, \ldots, a_{m,j}]^T$ lässt sich $Ax = b$ auch als

$$x_1 A(:,1) + x_2 A(:,2) + \ldots + x_n A(:,n) = b$$

schreiben. Gesucht sind also Koeffizienten x_j, mit deren Hilfe man die rechte Seite b als Linearkombination der Spalten von A darstellen kann.

Vorbemerkungen

Definiert man das **Bild** von A durch

$$\mathbf{B}(A) := \left\{ y \in \mathbb{R}^m : \exists\, x = [x_1, x_2, \ldots, x_n]^T \in \mathbb{R}^n \quad \text{mit} \quad y = Ax = \sum_{j=1}^{n} x_j A(:,j) \right\},$$

so besitzt $Ax = b$ genau dann (mindestens) eine Lösung, wenn $b \in \mathbf{B}(\mathbf{A})$ gilt.

Der **Kern** oder **Nullraum** von A ist durch

$$\mathbf{N}(A) := \{ z \in \mathbb{R}^n : Az = 0 \}$$

definiert. Offensichtlich besitzt $Ax = b$ höchstens eine Lösung, wenn $\mathbf{N}(A) = \{0\}$ gilt.

Vorbemerkungen

Für quadratische Matrizen $A \in \mathbb{R}^{n \times n}$ (in dieser Vorlesung werden wir uns ausschließlich mit solchen Matrizen befassen) heißt das:

$Ax = b$ **ist genau dann eindeutig lösbar, wenn** A **invertierbar ist.**

Satz. Für eine quadratische Matrix $A \in \mathbf{R}^{n \times n}$ sind die folgenden fünf Aussagen äquivalent:

1. A ist invertierbar.
2. $\mathbf{N}(\mathbf{A}) = \{\mathbf{0}\}$
3. $\mathbf{B}(A) = \mathbb{R}^n$
4. Alle Eigenwerte von A sind ungleich 0.
5. det(A) $\neq 0$

Dreieckssysteme

Grundidee

Die Grundidee vieler Algorithmen der numerischen linearen Algebra besteht darin, die zu lösende Aufgabe in eine einfache Form zu transformieren, bei der die Lösung „abgelesen“ werden kann. Sehr einfach zu lösende Gleichungssysteme sind z.B. **Dreiecksysteme**.

Ein lineares Gleichungssystem der Form

$$\begin{array}{ccccccccc} u_{1,1}x_1 & + & u_{1,2}x_2 & + & \cdots & + & u_{1,n}x_n & = & c_1 \\ & & u_{2,2}x_2 & + & \cdots & + & u_{2,n}x_n & = & c_2 \\ & & & & \ddots & & \vdots & & \vdots \\ & & & & & & u_{m,n}x_n & = & c_m \end{array}$$

oder kurz $Ux = c$ mit einer oberen Dreiecksmatrix U heisst oberes Dreieckssystem.

Dreieckssysteme

Ist $\det(U) \neq 0$, d.h. $u_{i,i} \neq 0$, so besitzt das obere Dreieckssystem $Ux = c$ eine eindeutige Lösung, die man durch **Rückwärtssubstitution**

$$x_j = \frac{1}{u_{j,j}} \left(c_j - u_{j,j+1}x_{j+1} - \ldots - u_{j,n}x_n\right) \quad \text{mit} \quad j = n, n-1, \ldots, 1$$

mit n Divisionen, $n(n-1)/2$ Multiplikationen und $n(n-1)/2$ Additionen, also insgesamt n^2 Gleitkommaoperationen berechnen kann.

Untere Dreieckssysteme werden —falls $\det(L) \neq 0$ —analog durch **Vorwärtssubstitution**,

$$x_j = \frac{1}{l_{j,j}} \left(c_j - u_{j,1}x_1 - \ldots - u_{j,j-1}x_{j-1}\right) \quad \text{mit} \quad j = 1, 2, \ldots, n.$$

mit n^2 Gleitkommaoperationen gelöst.

Gauss-Elimination GE

Idee:

Transformiere $Ax = b$ mit $\det(A) \neq 0$ auf ein oberes Dreiecksystem $Ux = c$, ohne die Lösung x zu verändern. (Löse dann $Ux = c$ durch Rückwärtssubstitution.)

Die Lösung eines Gleichungssystemes verändert sich nicht, wenn man es einer der drei folgenden Umformungen unterzieht:

- Subtrahieren des λ-fachen einer Gleichung von einer anderen.
- Vertauschen zweier Gleichungen.
- Vertauschen zweier Variablen.

Man muss sich allerdings merken, welche Gleichungen und Unbekannten vertauscht wurden.

Beispiel Gauss-Elimination GE

$$\begin{pmatrix} 2 & -1 & -3 & 3 \\ 4 & 0 & -3 & 1 \\ 6 & 1 & -1 & 6 \\ -2 & -5 & 4 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 \\ -8 \\ -16 \\ -12 \end{pmatrix}$$

1. Schritt: Eliminiere x_1 aus der zweiten, dritten und vierten Gleichung. Um das zu erreichen, subtrahiert man das $l_{i,1} = \frac{a_{i,1}}{a_{1,1}}$-fache der ersten Gleichung von der i-ten $(i = 2, 3, 4)$. Hier: $l_{2,1} = 2, l_{3,1} = 3, l_{4,1} = -1$. Es ergibt sich:

$$L_1[A|b] = \left(\begin{array}{cccc|c} 2 & -1 & -3 & 3 & 1 \\ 0 & 2 & 3 & -5 & -10 \\ 0 & 4 & 8 & -3 & -19 \\ 0 & -6 & 1 & 4 & -11 \end{array}\right)$$

Beispiel Gauss-Elimination GE

$$L_1[A|b] = \left(\begin{array}{cccc|c} 2 & -1 & -3 & 3 & 1 \\ 0 & 2 & 3 & -5 & -10 \\ 0 & 4 & 8 & -3 & -19 \\ 0 & -6 & 1 & 4 & -11 \end{array}\right)$$

2. Schritt: Eliminiere x_2 aus der dritten und vierten Gleichung. Um das zu erreichen, subtrahiert man das $l_{i,2} = \frac{a_{i,2}}{a_{2,2}}$-fache der zweiten Gleichung von der i-ten $(i = 3, 4)$. Hier: $l_{3,2} = 2, l_{4,2} = -3$. Es ergibt sich:

$$L_2[A|b] = \left(\begin{array}{cccc|c} 2 & -1 & -3 & 3 & 1 \\ 0 & 2 & 3 & -5 & -10 \\ 0 & 0 & 2 & 7 & 1 \\ 0 & 0 & 10 & -11 & -41 \end{array}\right)$$

Beispiel Gauss-Elimination GE

$$L_2[A|b] = \left(\begin{array}{cccc|c} 2 & -1 & -3 & 3 & 1 \\ 0 & 2 & 3 & -5 & -10 \\ 0 & 0 & 2 & 7 & 1 \\ 0 & 0 & 10 & -11 & -41 \end{array}\right)$$

3. Schritt: Eliminiere x_3 aus der vierten Gleichung, d.h. subtrahiere das $l_{i,3} = \frac{a_{i,3}}{a_{3,3}}$-fache der dritten Gleichung von der i-ten $(i = 4)$. Hier: $l_{4,3} = 5$.

$$L_3[A|b] = \left(\begin{array}{cccc|c} 2 & -1 & -3 & 3 & 1 \\ 0 & 2 & 3 & -5 & -10 \\ 0 & 0 & 2 & 7 & 1 \\ 0 & 0 & 0 & -46 & -46 \end{array}\right)$$

Beispiel Gauss-Elimination GE

$$L_3[A|b] = \left(\begin{array}{cccc|c} 2 & -1 & -3 & 3 & 1 \\ 0 & 2 & 3 & -5 & -10 \\ 0 & 0 & 2 & 7 & 1 \\ 0 & 0 & 0 & -46 & -46 \end{array}\right)$$

Die Eliminationsphase ist jetzt abgeschlossen und man kann $x_4, x_3, x_2.x_1$ durch eine Rückwärtssubstitution bestimmen:

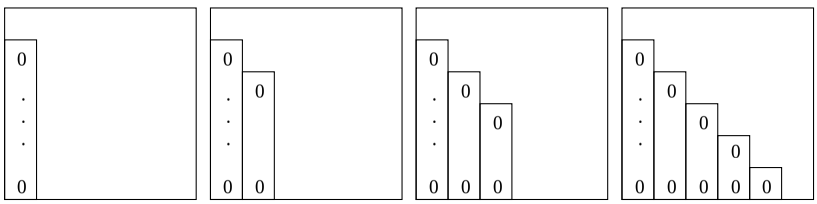
$$x_4 = 1, \quad x_3 = \frac{1}{2}[1 - 7x_4] = -3, \quad x_2 = \frac{1}{2}[-10 - 3x_3 + 5x_4] = 2$$

$$x_1 = \frac{1}{2}[1 + x_2 + 3x_3 - 3x_4] = -\frac{9}{2}$$

Algorithmus der Gauss-Elimination

[1] Addiere das λ-fache der Zeile $j-$ mit der i–ten Zeile ($i > j$),
[2] Löse das resultierende obere Dreieckssystem $Ux = c$.

```
1  ... Für jede Spalte i
2  ... Ändere alle Elemente unterhalb der Diagonalen zu Null
3  ... durch Addition eines λ-fachen der i-ten Zeile
4  for i ⟵ 1 to n-1
5    ... Für jede Zeile j unterhalb der Zeile i
6    for j ⟵ i+1 to n
7      ... Addiere das λ-fachen der i-ten Zeile zu Zeile j
8      for k ⟵ i to n
9        a_{j,k} ⟵ a_{j,k} − (a_{j,i}/a_{i,i}) · a_{i,k}
```



*V*erschiedene Phasen der Gauss-Elimination.

Modifikation der Gauss-Elimination

Original Version:

```
1  for i ⟵ 1 to n − 1
2    ... Für jede Zeile j unterhalb der Zeile i
3    for j ⟵ i+1 to n
4      ... Addiere das λ-fachen der i-ten Zeile zu Zeile j
5      for k ⟵ i to n
6        a_{j,k} ⟵ a_{j,k} − (a_{j,i}/a_{i,i}) · a_{i,k}
```

Entferne die Konstanten $(a_{j,i}/a_{i,i})$ aus der inneren Schleife.

```
1  for i ⟵ 1 to n − 1
2    for j ⟵ i+1 to n
3      m = a_{j,i}/a_{i,i}
4      for k ⟵ i to n
5        a_{j,k} ⟵ a_{j,k} − m · a_{i,k}
```

Modifikation der Gauss-Elimination

Vorherige Version

```
1  for i ⟵ 1 to n − 1
2    for j ⟵ i + 1 to n
3      m = a_{j,i}/a_{i,i}
4      for k ⟵ i to n
5        a_{j,k} ⟵ a_{j,k} − m · a_{i,k}
```

Benutze die Information, dass alle Elemente unterhalb der Diagonalen in der Spalte i null werden

```
1  for i ⟵ 1 to n − 1
2    for j ⟵ i + 1 to n
3      m = a_{j,i}/a_{i,i}
4      for k ⟵ i + 1 to n
5        a_{j,k} ⟵ a_{j,k} − m · a_{i,k}
```


Modifikation der Gauss-Elimination

Vorherige Version

```
1  for i ⟵ 1 to n − 1
2    for j ⟵ i + 1 to n
3      m = a_{j,i}/a_{i,i}
4      for k ⟵ i + 1 to n
5        a_{j,k} ⟵ a_{j,k} − m · a_{i,k}
```

Speichere die Faktoren m unterhalb der Diagonalen an den Stellen, die Null werden. Damit können die Faktoren wieder benutzt werden.

```
1  for i ⟵ 1 to n − 1
2    for j ⟵ i + 1 to n
3      a_{j,i} = a_{j,i}/a_{i,i}
4      for k ⟵ i + 1 to n
5        a_{j,k} ⟵ a_{j,k} − a_{j,i} · a_{i,k}
```

Modifikation der Gauss-Elimination

Vorherige Version

```
1  for i ⟵ 1 to n − 1
2    for j ⟵ i + 1 to n
3      a_{j,i} = a_{j,i}/a_{i,i}
4      for k ⟵ i + 1 to n
5        a_{j,k} ⟵ a_{j,k} − a_{j,i} · a_{i,k}
```

Trennung der Schleifen/Loops.

```
1    for i ⟵ 1 to n − 1
2      for j ⟵ i + 1 to n
3        a_{j,i} = a_{j,i}/a_{i,i}
4      for j ⟵ i + 1 to n
5        for k ⟵ i + 1 to n
6          a_{j,k} ⟵ a_{j,k} − a_{j,i} · a_{i,k}
```


Modifikation der Gauss-Elimination

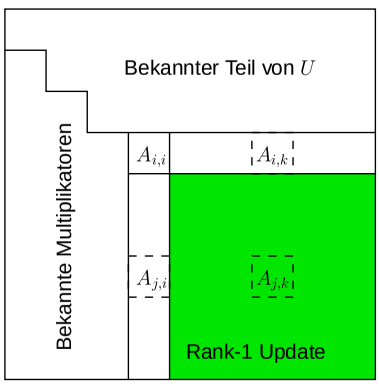
BLAS Matrix Notation

Skalierung in Zeile 2

$$\mathbf{y} = a \cdot \mathbf{y}$$

Rank-1 Update in Zeile 3,4

$$\mathbf{A} = \mathbf{A} + \mathbf{x} \cdot \mathbf{y}^T$$



Matlab-Notation

```
1  for i ⟵ 1 to n − 1
2    A(i+1:n, i) = A(j, i)/A(i, i);
3    A(i+1:n, i+1:n) = A(i+1:n, i+1:n)
4       - A(i+1:n, i) · A(i,i+1:n);
```

Pseudo-Code

```
1  for i ⟵ 1 to n − 1
2    for j ⟵ i + 1 to n
3      a_{j,i} = a_{j,i}/a_{i,i}
4      for j ⟵ i + 1 to n
5        for k ⟵ i + 1 to n
6          a_{j,k} ⟵ a_{j,k} − a_{j,i} · a_{i,k}
```

Zerlegung in zwei Faktoren $A = LU$

```
1  for i ⟵ 1 to n − 1
2     A(i+1:n, i) = A(j, i)/A(i, i);
3     A(i+1:n, i+1:n) = A(i+1:n, i+1:n) - A(i+1:n, i) · A(i,i+1:n);
```

- Fasse die Multiplikatoren m zu einer strikt unteren Dreiecksmatrix M zusammen. Dann ist $L := I + M$ (mit I als Einheitsmatrix)
- Nenne die obere Dreiecksmatrix der resultierenden Matrix U.
- **Satz (LU Faktorisierung)**: Falls der oben angegebene Algorithmus terminiert (keine Division durch Null), dann gilt $A = LU$.

Zerlegung in zwei Faktoren $A = LU$

- Zerlege lineares Gleichungssystem in $Ax = (LU)x = L(Ux) = Ly = b$ mit $y := Ux$.
- Löse $Ax = b$ mit Gauss-Elimination durch
  1. Faktorisiere in zwei Dreiecksmatrizen $A = LU$ Aufwand $O(\frac{2}{3}n^3)$
  2. Vorwärtssubstitution $Ly = b$ für y Aufwand $O(n^2)$
  3. Rückwartssubstitution $Ux = y$ für x Aufwand $O(n^2)$
- Für die Substitutionen benötigt man nur die Faktoren und nicht die Matrix.
- Insbesondere dann vorteilhaft, wenn $Ax = b_i$ mit einer Matrix A und mehreren rechten Seiten b_i zu lösen ist.

Komplexität Gauss-Elimination

```
1   for i ⟵ 1 to n − 1
2     for j ⟵ i + 1 to n
3       a_{j,i} = a_{j,i}/a_{i,i}   ... Skalierung
4     for j ⟵ i + 1 to n
5       for k ⟵ i + 1 to n
6         a_{j,k} ⟵ a_{j,k} − a_{j,i} · a_{i,k}   ... Rang 1-Update
```

Skalierung : $\sum_{i=1}^{n-1}(n-i) = O(n^2)$

Rang-1 Update: : $2 \times \sum_{i=1}^{n-1}(n-i)^2 = O(2/3n^3)$

Satz: Es sei $Ax = b$ ein lineares Gleichungssystem mit einer $n \times n$ Matrix. Dann benötigt die LU Faktorisierung die Gleitkommaoperationen:

$A = LU$ Faktorisierung $\longleftrightarrow$ $O(2/3n^3)$

$Ly = b$ und $Ux = y$ Vorwärts-/Rückwärtssubstitution $\longleftrightarrow$ $O(n^2)$

Level-3 BLAS Leistung

- $q = f/m$ #Anzahl Operationen für jeden langsamen Speicherzugriff (siehe Lektion 2)
- Skalierung: $f = n^2, m = 2 \cdot n^2 \longrightarrow q = 1/2$ (Kein Level-3 BLAS!)
- Rank-1 Update: $f = 2/3 \cdot n^3, m = 2n^3 \longrightarrow q = 1/3$ (Kein Level-3 BLAS!)

Block Gauss-Elimination

Um einen Level-3 BLAS Algorithmus zu erreichen, muss die Matrix A in Blöcke A_b der Grösse $b \cdot b$ zerlegt werden. Dann sieht die Block Gauss-Elimination folgendermassen aus:

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 & 0 & 0 \\ L_{21} & L_{22} & 0 & 0 \\ L_{31} & L_{32} & L_{33} & 0 \\ L_{41} & L_{42} & L_{43} & L_{44} \end{pmatrix} \times \begin{pmatrix} U_{11} & U_{12} & U_{13} & U_{14} \\ 0 & U_{22} & U_{23} & U_{24} \\ 0 & 0 & U_{33} & U_{34} \\ 0 & 0 & 0 & U_{44} \end{pmatrix}$$

Block Gauss-Elimination - Level 3 BLAS

Beispiel der Gauss-Elimination für zwei Blöcke

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix}$$
$$= \begin{pmatrix} L_{11}U_{11} & L_{11}U_{12} \\ L_{21}U_{11} & L_{21}U_{12} + L_{22}U_{22} \end{pmatrix}$$

Algorithmus

1. Gauss-Elimination des Diagonalblockes $A_{11} = L_{11} \cdot U_{11}$
2. Skalierung der Blöcke ausserhalb der Diagonalen:

   $L_{11} \cdot U_{12} = A_{12}$ oder $(L_{11})^{-1} \cdot A_{12} = U_{12}$

   $L_{21} \cdot U_{11} = A_{21}$ oder $A_{21} \cdot (U_{11})^{-1} = L_{21}$

   Die Matrizen L_{11} und U_{11} werden nicht invertiert, da sie untere bzw. obere Dreieckssysteme sind.
3. Update der restlichen Matrix:
   $\tilde{A}_{22} = A_{22} - L_{21} \cdot U_{12}$ und Faktorisierung von $\tilde{A}_{22} = L_{22} \cdot U_{22}$

Komplexität Gauss-Elimination

Operationszahl f der Block Gauss-Elimination

Es sei A eine $n \times n$ Matrix, welche in Blöcke der Grösse b geteilt wird. Dann werden bei der Block Gauss-Elimination die folgenden Operationen durchgeführt

Operation	Aufwand
$A_{11} = L_{11} \cdot U_{11}$	$\sum_{i=1}^{n/b} \frac{2}{3} \cdot b^3$
$L_{11} \cdot U_{12} = A_{12}$	$\sum_{i=1}^{(n/b)-1} (i \cdot b) \cdot b^2$
$L_{21} \cdot U_{11} = A_{21}$	$\sum_{i=1}^{(n/b)-1} (i \cdot b) \cdot b^2$
$\tilde{A}_{22} = \tilde{A}_{22} - L_{21}U_{12}$	$\sum_{i=1}^{(n/b)-1} 2 \cdot b \cdot (i \cdot b)^2$
	$O(\frac{2}{3}n^3) + O(n^2)$

Die Block Gauss-Elimination benötigt genauso viele Gleitkommaoperationen wie die ungeblockte Gauss-Elimination.

Komplexität Gauss-Elimination

Speicherzugriffe m der Block Gauss-Elimination

Es sei angenommen, dass man $2 \cdot b^2$ Operanden im Cache halten kann,

Operation	Aufwand (Speicherzugriffe)	Bemerkung
$A_{11} = L_{11} \cdot U_{11}$	$\sum_{i=1}^{n/b} b^2$	
$L_{11} \cdot U_{12} = A_{12}$	$\sum_{i=1}^{(n/b)-1} 2 \cdot (i \cdot b) \cdot b$	L_{11} ist im Cache
$L_{21} \cdot U_{11} = A_{21}$	$\sum_{i=1}^{(n/b)-1} 2 \cdot (i \cdot b) \cdot b$	U_{11} ist im Cache
$\tilde{A}_{22} = \tilde{A}_{22} - L_{21}U_{12}$	$\sum_{i=1}^{(n/b)-1} 4(i \cdot b)^2$	
	$\simeq 4/3 \cdot 1/b[O(n^3) + O(n^2)]$	

Verhältnis Berechnungen / Speicherzugriffe $q = f/m$:

$f = 2/3 \cdot n^3 \quad m = 4/3 \cdot 1/b \cdot n^3 \quad \longrightarrow \quad q = \frac{1}{2}b$ (Level-3 BLAS)

Dabei b ist natürlich durch die Grösse des Caches beschränkt.

Gauss-Elimination Rekorde für grosse vollbesetzte Matrizen

Liste der weltweit schnellsten 500 Rechner: www.top500.org

Jahr	Systemgrösse n	Maschine	# Procs	Gflops	(Peak)
1950	O(100)				
1995	128,600	Intel Paragon	6768	281	(338)
1996	215,000	Intel ASCI Red	7264	1068	(1453)
1998	148,000	Cray T3E	1488	1127	(1786)
1998	235,000	Intel ASCI Red	9152	1338	(1830)
1999	374,000	SGI ASCI Blue	5040	1608	(2520)
2000	362,000	Intel ASCI Red	9632	2380	(3207)
2001	518,000	IBM ASCI White	8000	7226	(12000)
2002 ($*$)	1,075,200	NEC Earth Simulator	5104	35000	(40832)

Zeit zum Lösen von ($*$) mit Gauss-Elimination: $\frac{2}{3}n^3/Gflops \approx 5.2$ Stunden

Pivotisierung

Das Eliminationsverfahren bricht zusammen, wenn es auf die Matrix $A = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$, obwohl $\det(A) \neq 0$.

Was nun? Pivotisierung! $\rightarrow$ Zeilen-/Spaltenpermutationen.

Ist Π eine Permutation der Indexmenge $\{1, 2, 3, \ldots, n\}$, dann ist die zugehörige Permutationsmatrix $P = P_\Pi = [p_{i,j}] \in \mathbb{R}^{n \times n}$ durch

$$p_{i,j} = \begin{cases} 1, & \text{falls} \quad j = \Pi(i) \\ 0, & \text{falls} \quad j \neq \Pi(i) \end{cases}$$

definiert.
Ist $A = [a_{i,j}] \in \mathbb{R}^{n \times n}$ mit den Zeilen $z_1^T, \ldots, z_n^T$ und Spalten $s_1, \ldots, s_n$, so besitzt PA die permutierten Zeilen $z_{\Pi(1)}^T, \ldots, z_{\Pi(n)}^T$ und AP^T die Spalten $s_{\Pi(1)}, \ldots, s_{\Pi(n)}$. Insbesondere ist $PAP^T = [a_{\Pi(i),\Pi(j)}]$.

Beispiel Permutationen

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad A = \begin{pmatrix} 2 & -1 & -3 & 3 \\ 4 & 0 & -3 & 1 \\ 6 & 1 & -1 & 6 \\ -2 & -5 & 4 & 1 \end{pmatrix} \quad P^T = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

$$PA = \begin{pmatrix} 4 & 0 & -3 & 1 \\ -2 & -5 & 4 & 1 \\ 2 & -1 & -3 & 3 \\ 6 & 1 & -1 & 6 \end{pmatrix} \quad AP^T = \begin{pmatrix} -1 & 3 & 2 & -3 \\ 0 & 1 & 4 & -3 \\ 1 & 6 & 6 & -1 \\ -5 & 1 & -2 & 4 \end{pmatrix}$$

$$PAP^T = \begin{pmatrix} 0 & 1 & 4 & -3 \\ -5 & 1 & -2 & 4 \\ -1 & 3 & 2 & -3 \\ 1 & 6 & 6 & -1 \end{pmatrix}$$

Spaltenpivotsuche

```
1  for i ⟵ 1 to n − 1
2    Bestimme p so, dass |a_{p,j}| = max_{i=j,...n} |A_{i,j}|   ... Spaltenpivotsuche
3    Vertausche Zeilen A(j,:) und A(p,:), rechte Seite b(j) und b(p)
4    for j ⟵ i + 1 to n
5      a_{j,i} = a_{j,i}/a_{i,i}   ... Skalierung
6    for j ⟵ i + 1 to n
7      for k ⟵ i + 1 to n
8        a_{j,k} ⟵ a_{j,k} − a_{j,i} · a_{i,k}   ... Rang 1-Update
```

Satz: Es sei A invertierbar. Dann gibt es eine Permutationsmatrix durch Spaltenpivotsuche**Block Gauss-Elimination** $P \in \mathbb{R}^{n\times n}$, so dass

$PA = LU$ existiert und

(a) Bestimme P, L, U, so dass $PA = LU$.

(b) Löse $Ly = Pb$ durch Vorwärtssubstitution.

(c) Löse $Ux = y$ durch Rückwärtssubstitution.

Spaltenpivotsuche

Spaltenpivotsuche muss immer durchgeführt werden, um **kleine Pivotelemente** zu vermeiden, die zu **ungenauen Ergebnissen** führen können:

$$\begin{pmatrix} 10^{-20} & 1 \\ 1 & 1 \end{pmatrix} \mathbf{x} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{Lösung } \mathbf{x} = \frac{1}{1-10^{-20}} \begin{pmatrix} -1 \\ 1 \end{pmatrix} \approx \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

Gauss-Elimination mit IEEE 64-BIT Rundungseinheit $u = 2^{-53} \approx 10^{-16}$:

$10^{-20}x_1 + x_2 = 1$

$(1 - 10^{20})x_2 = -10^{20}$, gerundet $-10^{20}\tilde{x}_2 = -10^{20}$, d.h. $\tilde{x}_2 = 1$ und $\tilde{x}_1 = 0$

Gauss Elimination mit Spaltenpivotsuche würde das richtige Ergebnis liefern $\tilde{x}_1 = -1$ und $\tilde{x}_2 = 1$.

Spaltenpivotsuche

Gauss-Elimination ohne Pivotsuche liefert also bei gerundeter Rechnung die folgenden Faktoren:

$$\tilde{L} = \begin{pmatrix} 1 & 0 \\ 10^{20} & 1 \end{pmatrix} \text{ und } \tilde{U} = \begin{pmatrix} 10^{-20} & 1 \\ 0 & -10^{20} \end{pmatrix}$$

anstelle der exakten Faktoren

$$L = \begin{pmatrix} 1 & 0 \\ 10^{20} & 1 \end{pmatrix} \text{ und } U = \begin{pmatrix} 10^{-20} & 1 \\ 0 & 1 - 10^{20} \end{pmatrix}$$

Beim Gleichungslösen wird also verwendet:

$$\tilde{L}\tilde{U} = \begin{pmatrix} 10^{-20} & 1 \\ 1 & 0 \end{pmatrix} \text{ statt } LU = \begin{pmatrix} 10^{-20} & 1 \\ 0 & 1 \end{pmatrix} = A$$

Pivotisierung ist wichtig für die Numerische Stabilität der Gauss-Elimination.

Equilibrierung der Gleichungen

Die Spaltenpivotsuche kann sehr leicht ad absurdum geführt werden, wenn man die verschiedenden Gleichungen unterschiedlich gewichtet:

Beispiel

$$\begin{pmatrix} 1 & 10^{20} \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 10^{20} \\ 0 \end{pmatrix}$$

führt bei Rechnung mit Spaltenpivotsuche auf $\tilde{x}_2 = 1$ und $\tilde{x}_1 = 0$. Die Matrix A heisst zeilenequilibriert, wenn jeder Zeile die Beitragssumme der Einträge gleich ist mit

$$\sum_{j=1}^{n} |a_{i,j}| = 1 \qquad (i = 1, 2, \ldots, n)$$

Mann kann das leicht dadurch erreichen, indem man die $i - te$ Zeile vorab mit $1/\sum_{j=1}^{n} |a_{i,j}|$ multipliziert (skaliert).

Cholesky und LDL^T Zerlegung

Satz: Es sei $Ax = b$ ein lineares Gleichungssystem mit einer $n \times n$ symmetrischen positiv definiten Matrix A_{SPD}[a] oder einer symmetrisch indefiniten Matrix A_{IND} [b]. Dann benötigt man nur die untere Dreieicksmatrix L und es können beide Matrizen in

$$\begin{array}{lcll} A_{SPD} & = & LL^T & \text{Cholesky-Faktorisierung} \\ A_{IND} & = & L_1 D L_1^T & LDL^T\text{-Faktorisierung} \end{array}$$

zerlegt werden. Der numerische Aufwand ist bei beiden Systemen $O(\frac{1}{3}n^3)$. Die Matrix D ist eine Diagonalmatrix mit positiven und negativen Diagonalelementen und L_1 ist eine normierte untere Dreiecksmatrix.

[a] Matrix A ist symmetrisch positiv defi nite $:= x^T Ax > 0 \; \forall x \in \mathbb{R}^{n \times n} \setminus \{0\}$

[b] Matrix kann negative Daigonaleinträge oder negative Eigenwerte besitzen.

Software für Gauss-Elimination vollbesetzter Matrizen

Software

- **Linear Algebra Package:** The LAPACK library provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems.
  `http://www.netlib.org/lapack`
- **ATLAS:** The ATLAS (Automatically Tuned Linear Algebra Software) project is an ongoing research effort focusing on applying empirical techniques in order to provide portable performance.
  `http://math-atlas.sourceforge.net`
- **SCALAPACK:** The ScaLAPACK (or Scalable LAPACK) library includes a subset of LAPACK routines redesigned for distributed memory MIMD parallel computers.
  `http://www.netlib.org/scalapack/slug/index.html`