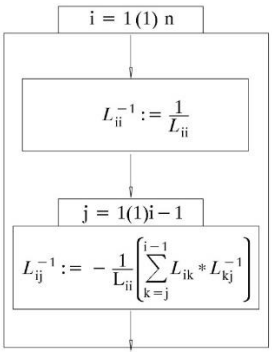
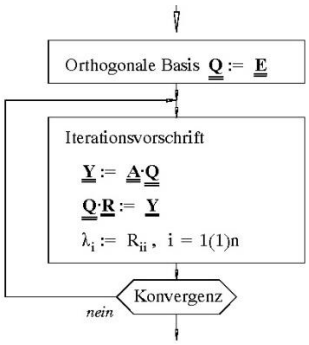
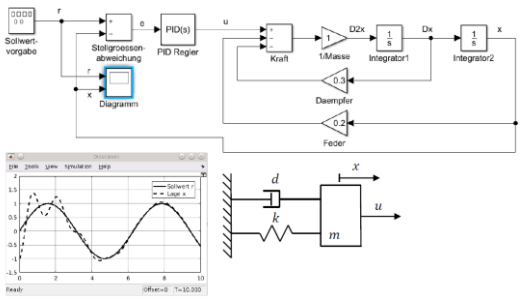
Numerische Methoden der Dynamik

Dr.-Ing. Pascal Ziegler
Andreas Baumann, M.Sc.
Institut für Technische und Numerische Mechanik
Pfaffenwaldring 9, 70569 Stuttgart

Einführung in die numerischen Methoden zur Behandlung mechanischer Systeme. Grundlagen der numerischen Mathematik: Numerische Prinzipe, Maschinenzahlen, Fehleranalyse. Lineare Gleichungssysteme: Cholesky-Zerlegung, Gauß-Elimination, LR-Zerlegung, QR-Zerlegung, iterative Methoden bei quadratischer Koeffizientenmatrix, Lineares Ausgleichsproblem. Eigenwertproblem: Grundlagen, Normalformen, Vektoriteration, Berechnung von Eigenwerten mit dem QR-Verfahren, Berechnung von Eigenvektoren. Anfangswertproblem bei gewöhnlichen Differentialgleichungen: Grundlagen, Einschrittverfahren (Runge-Kutta Verfahren), Extrapolationsverfahren, Mehrschrittverfahren. Werkzeuge und numerische Bibliotheken für lineare Gleichungssysteme, Eigenwertprobleme und Anfangswertprobleme. Theorie und Numerik in der Anwendung - ein Vergleich.







- **Umfang:** 6 LP (4 SWS)
- **Zeit:** Montag 11:30 - 13:00 Uhr, V 55.01
  Donnerstag 11:30 – 13:00 Uhr, V47.04 (erste Vorlesung am 8. April 2024)
- **Prüfung**: Im Sommersemester: schriftlich
  Im Wintersemester: in der Regel mündlich
- **Homepage**: http://www.itm.uni-stuttgart.de/courses/numerik
- **Ansprechpartner**: Dr.-Ing. Pascal Ziegler,
  Institut für Technische und Numerische Mechanik,
  Pfaffenwaldring 9, 70569 Stuttgart. 4. Stock, Zimmer 4.118,
  0711/685 - 68041, pascal.ziegler@itm.uni-stuttgart.de.

  Andreas Baumann, M.Sc.,
  Institut für Technische und Numerische Mechanik,
  Pfaffenwaldring 9, 70569 Stuttgart. 4. Stock, Zimmer 4.153,
  0711/685 - 66490, andreas.baumann@itm.uni-stuttgart.de.

Vorlesung

1 Einleitung

2 Grundlagen der Numerischen Mathematik
2.1 Begriffe
2.2 Numerische Prinzipe
2.3 Maschinenzahlen
2.4 Fehleranalyse

3 Lineare Gleichungssysteme
3.1 Problemstellung
3.2 Direkte Verfahren bei quadratischer Koeffizientenmatrix
3.3 Cholesky–Zerlegung
3.4 Gauß–Elimination
3.5 LR–Zerlegung
3.6 QR–Zerlegung
3.7 Iterative Methoden bei quadratischer Koeffizientenmatrix
3.8 Mehrgitterverfahren
3.9 Determinante einer Matrix
3.10 Inverse einer Matrix
3.11 Überbestimmte Gleichungssysteme

4 Eigenwertprobleme
4.1 Grundlagen
4.2 Normalformen
4.3 Vektoriteration
4.4 Berechnung von Eigenwerten mit dem QR–Verfahren
4.5 Berechnung von Eigenvektoren
4.6 Praktische Lösung des Eigenwertproblems
4.7 Werkzeuge und numerische Bibliotheken für Eigenwertprobleme

5 Anfangswertprobleme bei gewöhnlichen Differentialgleichungen
5.1 Problemstellung
5.2 Grundlagen
5.3 Einschrittverfahren
5.4 Extrapolationsverfahren
5.5 Mehrschrittverfahren
5.7 Werkzeuge und numerische Bibliotheken für Anfangswertprobleme

Literatur

E. Anderson et al.: *LAPACK User's Guide*.
Philadelphia: SIAM, 1992.

G.H. Golub und C.F. van Loan: *Matrix Computations*.
London: North Oxford Academic Publ., 1986.

E. Hairer, S.P. Nørsett, G. Wanner: *Solving Ordinary Differential Equations I. Nonstiff Problems*. Berlin: Springer, 2009.

E. Hairer und G. Wanner: *Solving Ordinary Differential Equations II. Stiff and Differential–Algebraic Problems*. Berlin: Springer, 2010.

P.C. Müller und W. Schiehlen: *Lineare Schwingungen*.
Wiesbaden: Akademische Verlagsgesellschaft, 1976 (vergriffen).

N.N.: *Die Programmiersprache C. Ein Nachschlagewerk*. Hannover: Regionales Rechenzentrum für Niedersachsen / Universität Hannover, 2001.

H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery: *Numerical Recipes in C*.
Cambridge: Cambridge University Press, 1993.

H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery: *Numerical Recipes 3rd Edition: The Art of Scientific Computing*,
Cambridge: Cambridge University Press, 2007.

W. Schiehlen und P. Eberhard: *Technische Dynamik*.
Berlin: Springer Vieweg, 2014

H.–R. Schwarz und N. Köckler: *Numerische Mathematik*.
Stuttgart: Teubner, 2006.

J. Stoer: *Numerische Mathematik 1*.
Berlin: Springer, 1993.

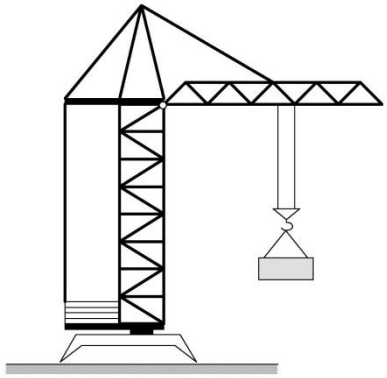
J. Stoer und R. Bulirsch: *Numerische Mathematik 2*.
Berlin: Springer, 2005.

R. Zurmühl und S. Falk: *Matrizen und ihre Anwendungen. Teil 2: Numerische Methoden*.
Berlin: Springer, 1986.

R. Schaback und H. Wendland: *Numerische Mathematik*.
Berlin: Springer, 2005.

Analyse technischer Systeme

Beispiel 1: Gleichgewicht eines Ladekranes

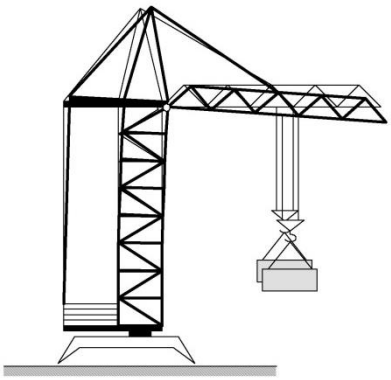


FEM

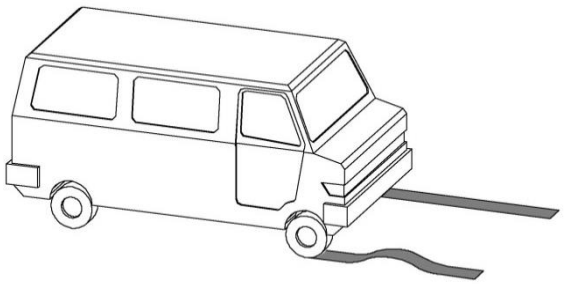
Bewegungsgleichungen $\boldsymbol{M} \cdot \ddot{\boldsymbol{y}} + \boldsymbol{K} \cdot \boldsymbol{y} = \boldsymbol{h}$

lineares Gleichungssystem $\boldsymbol{K} \cdot \boldsymbol{y}_0 = \boldsymbol{h}$

statisches Gleichgewicht $\boldsymbol{y}_0$



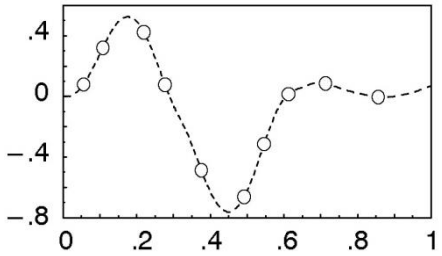
Beispiel 2: **Durch Fahrbahnunebenheiten hervorgerufene Fahrzeugschwingungen**

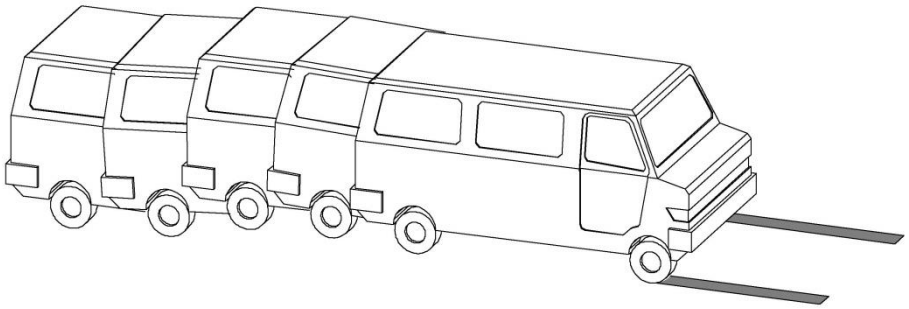


MKS

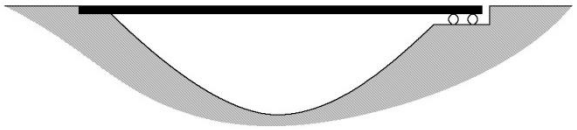
Bewegungsgleichungen: $\boldsymbol{M}(\boldsymbol{y},t)\cdot\ddot{\boldsymbol{y}}+\boldsymbol{k}(\boldsymbol{y},\dot{\boldsymbol{y}},t)=\boldsymbol{q}(\boldsymbol{y},\dot{\boldsymbol{y}},t)$

Anfangswertproblem $\dot{\boldsymbol{x}}=\boldsymbol{f}(\boldsymbol{x},t),\quad \boldsymbol{x}(0)=\boldsymbol{x}_0$





Beispiel 3: Eigenfrequenzen einer Brücke



kontinuierliches System

Differentialgleichungen der Biegeschwingungen $\rho A \frac{\partial^2 w}{\partial t^2} + \frac{EI}{L^4}\frac{\partial^4 w}{\partial x^4} = 0$

Randbedingungen $w(0) = w(1) = w'(0) = w'(1) = 0$

Hilfsgleichung

$\det \boldsymbol{A}_1 = 2\cos(\gamma)\cosh(\gamma) - 2 = 0$

Eigenwertproblem

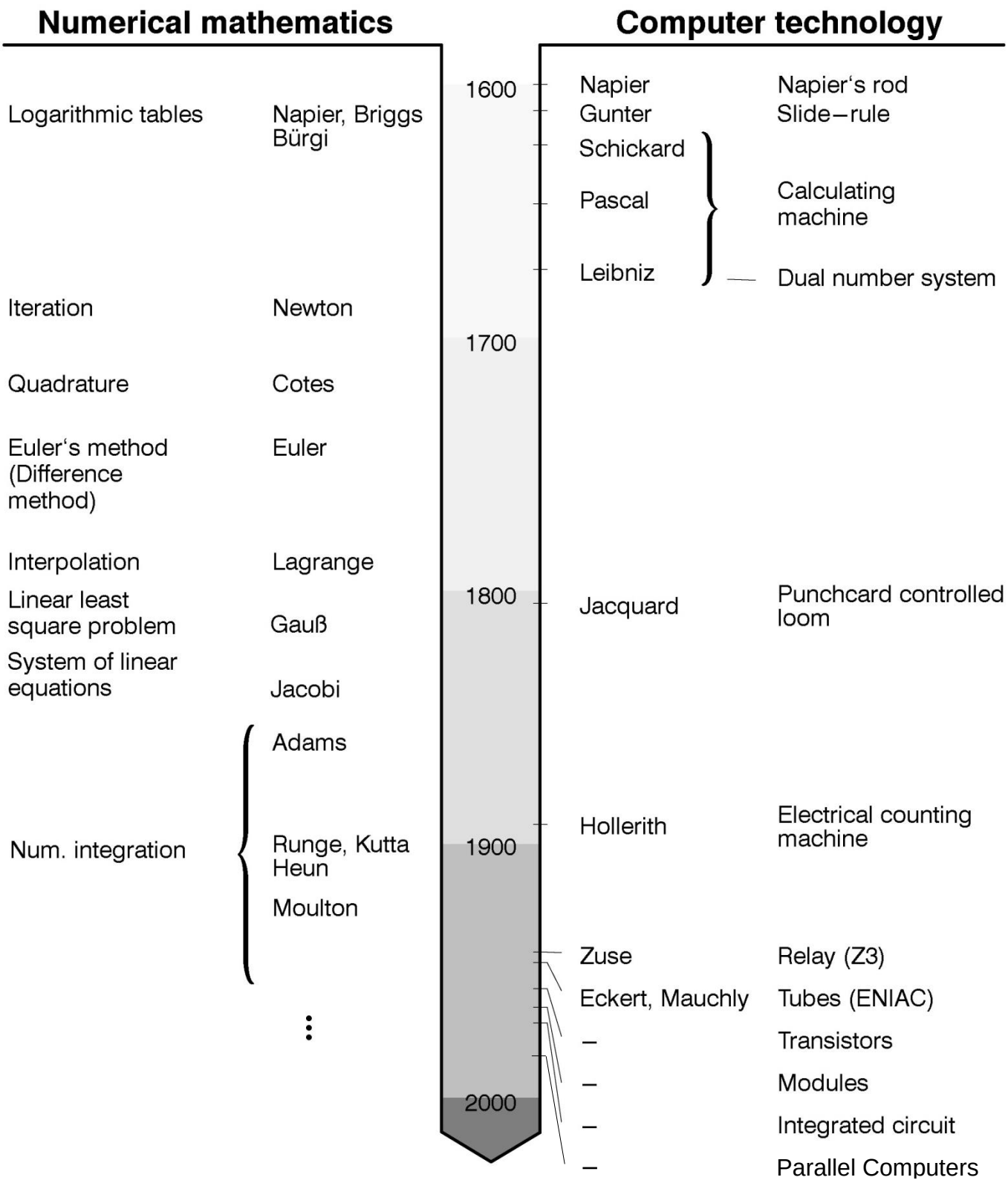
$\boldsymbol{A}_2 \cdot \hat{\boldsymbol{w}} = \lambda \hat{\boldsymbol{w}}$

Nullstellen γ_i

Eigenwerte λ_i

Eigenfrequenzen ω_i

Historische Entwicklung der numerischen Mathematik und der Computertechnologie



Zahldarstellung in Digitalrechnern

Die kleinste Informationseinheit in Digitalrechnern ist **1 Bit** (**b**inary digi**t**) mit den Werten 0 oder 1. Üblicherweise werden 8 Bit zu einem **Byte** zusammengefasst und Information nur in Vielfachen davon gespeichert.

Die Zahldarstellung ist auf verschiedenen Rechnern unterschiedlich. Die folgenden Angaben beziehen sich auf die x86-64-Architektur und richten sich nach der C-Notation:

Integer–Zahlen

Mit INTEGER–Zahlen können ganze Zahlen des erlaubten Zahlenbereichs **exakt** dargestellt werden.

Formate

- Char (int8_t): 1 Byte = 8 Bits
- Short Integer (int16_t): 2 Bytes = 16 Bits
- Integer (int32_t): 4 Bytes = 32 Bits
- Long Integer (int32_t): 8 Bytes = 64 Bits

Darstellung

- positive Zahlen

VZ	2^{n-2}						…	2^2	2^1	2^0
0	I	0	I	I	0	I	…	0	I	I
n–1	*n*–2						…	2	1	0

$0 < x \leq 2^{n-1} - 1$

- Null

VZ	2^{n-2}						…	2^2	2^1	2^0
0	0	0	0	0	0	0	…	0	0	0
n–1	*n*–2						…	2	1	0

$x = 0$

- negative Zahlen
  Darstellung im Allgemeinen im Zweierkomplement, d.h. Invertieren der Darstellung von $(-x)$ $(1 \rightarrow 0, 0 \rightarrow 1$, Einserkomplement) und Addition von 1.

VZ	2^{n-2}						…	2^2	2^1	2^0
I	I	0	I	I	0	I	…	0	I	I
n–1	*n*–2						…	2	1	0

$-2^{n-1} \leq x < 0$

Beispiel: Darstellung von $\pm 1234_{10}$ als Short Integer

$$
\begin{array}{rcccccccccccccccc}
1234_{10} = & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & {}_2 \\
\text{Einserkomplement} & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & {}_2 \\
+ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & {}_2 \\
\text{Zweierkomplement} & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & {}_2
\end{array} = -1234_{10}
$$

Gleitpunktzahlen (ANSI/IEEE 754 Standard)

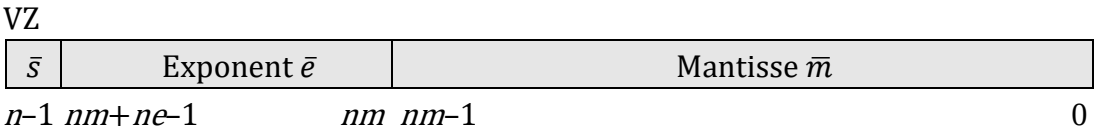
Mit Gleitkommazahlen lassen sich reelle Zahlen im Allgemeinen nur **fehlerbehaftet** darstellen, da sie zu diesem Zweck gerundet werden müssen. Die auf $(nm + 1)$ Stellen gerundete Dualzahl sei wie folgt normiert (Eindeutigkeit der Darstellung):

$$x = \pm \underbrace{1.z_{-1}z_{-2} \dots z_{-nm}}_{m} \cdot 2^e, \quad z_{-i} \in \{0,1\}.$$

Formate

- einfache Genauigkeit (float): 4 Bytes = 32 Bits
- doppelte Genauigkeit (double): 8 Bytes = 64 Bits

Darstellung



Vorzeichen

Das Vorzeichen wird in einem Vorzeichenbit festgehalten. Ist das Vorzeichenbit gesetzt ($\bar{s} = 1$), so ist die Zahl negativ.

Mantisse

Bei der Speicherung der Mantisse wird die Ziffer vor dem Dualpunkt als gesetzt angenommen und nur die Ziffern hinter dem Dualpunkt gespeichert (1 Bit zusätzliche Genauigkeit):

$$\bar{m} = m - 1.$$

Exponent

Der Exponent wird um einen Offset (Bias $= 2^{ne-1} - 1$) verschoben, so dass nur positive Werte auftreten:

$$\bar{e} = e + (2^{ne-1} - 1),$$
$$e = \bar{e} - (2^{ne-1} - 1), \quad -2^{ne-1} + 1 \leq e \leq 2^{ne-1}.$$

Dabei sind die extremen Exponenten allerdings Sonderfällen vorbehalten:

$\bar{e} = 0 \ldots 0_2$:	$\bar{m} = 0$	Darstellung von $x = 0$
	$\bar{m} \neq 0$	Darstellung von nicht–normierten Zahlen
$\bar{e} = 1 \ldots 1_2$:	$\bar{m} = 0$	Darstellung von $x = \pm\infty$
	$\bar{m} \neq 0$	Darstellung einer Nicht–Zahl (NaN) zur Fehlererkennung

Der Wert einer dargestellten Zahl ergibt sich damit im Normalfall zu

$$x = (-1)^{\bar{s}} \cdot 1.\bar{m} \cdot 2^{\bar{e}-(2^{ne-1}-1)}.$$

	float	double
Vorzeichen		
Bitposition	31	63
Exponent		
Anzahl der Bits ne	8	11
Bitpositionen	30 ÷ 23	62 ÷ 52
Bias	127	1023
Mantisse		
Anzahl der Bits nm	23	52
Bitpositionen	22 ÷ 0	51 ÷ 0
Zahlenbereich (normierte Darstellung)		
Betragsmaximum	3.402823 e+38	1.79769 e+308
Betragsminimum	1.175494 e-38	2.22507 e-308
Genauigkeit	5.960464 e-08	1.11022 e-16

Beispiel: Darstellung von $x = -21.3$ in einfacher Genauigkeit (single precision):

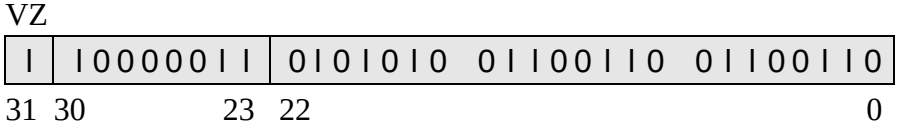
$$-21.3_{10} = -1\ 0\ 1\ 0\ 1.0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0 \ldots_2$$

normiert $-1.0\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0 \ldots \cdot 2^4$

gerundet $-1.\underbrace{0\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0}_{\bar{m}} \cdot 2^4$

$\bar{s} = 1$

$\bar{e} = 4_{10} + 127_{10} = 10000011_2$



Matrizenalgebra

Matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}: \boldsymbol{A} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}, \quad a_{ij} \in \mathbb{R},$

Vektor $\boldsymbol{x} \in \mathbb{R}^{n}: \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad x_i \in \mathbb{R}.$

Elementare Operationen

Operation	Schreibweise	Koordinaten	Abbildung
Addition	$\boldsymbol{C} = \boldsymbol{A} + \boldsymbol{B}$	$c_{ij} = a_{ij} + b_{ij}$	$\mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$
Multiplikation mit Skalar	$\boldsymbol{C} = \alpha\boldsymbol{A}$	$c_{ij} = \alpha a_{ij}$	$\mathbb{R} \times \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$
Transponieren	$\boldsymbol{C} = \boldsymbol{A}^{\mathrm{T}}$	$c_{ij} = a_{ji}$	$\mathbb{R}^{m\times n} \to \mathbb{R}^{n\times m}$
Differentiation	$\boldsymbol{C} = \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{A}$	$c = \frac{\mathrm{d}}{\mathrm{d}t}a_{ij}$	$\mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$
Matrizenmultiplikation	$\boldsymbol{y} = \boldsymbol{A}\cdot\boldsymbol{x}$ $\boldsymbol{C} = \boldsymbol{A}\cdot\boldsymbol{B}$	$y_i = \sum_k a_{ik}x_k$ $C_{ij} = \sum_k a_{ik}b_{kj}$	$\mathbb{R}^{m\times n} \times \mathbb{R}^{n} \to \mathbb{R}^{m}$ $\mathbb{R}^{m\times n} \times \mathbb{R}^{n\times p} \to \mathbb{R}^{m\times p}$
Inneres Produkt (Skalarprodukt)	$\alpha = \boldsymbol{x}\cdot\boldsymbol{y}$	$\alpha = \sum_k x_k y_k$	$\mathbb{R}^{n} \times \mathbb{R}^{n} \to \mathbb{R}$
Äußeres Produkt (Dyadisches Produkt)	$\boldsymbol{A} = \boldsymbol{x}\boldsymbol{y}$	$a_{ij} = x_i y_j$	$\mathbb{R}^{m} \times \mathbb{R}^{n} \to \mathbb{R}^{m\times n}$

Rechenregeln

Addition:
$$\boldsymbol{A}+(\boldsymbol{B}+\boldsymbol{C})=(\boldsymbol{A}+\boldsymbol{B})+\boldsymbol{C}$$
$$\boldsymbol{A}+\boldsymbol{B}=\boldsymbol{B}+\boldsymbol{A}$$

Multiplikation mit Skalar:
$$\alpha(\boldsymbol{A}\cdot\boldsymbol{B})=(\alpha\boldsymbol{A})\cdot\boldsymbol{B}=\boldsymbol{A}\cdot(\alpha\boldsymbol{B})$$
$$\alpha(\boldsymbol{A}+\boldsymbol{B})=\alpha\boldsymbol{A}+\alpha\boldsymbol{B}$$
$$(\alpha+\beta)\boldsymbol{A}=\alpha\boldsymbol{A}+\beta\boldsymbol{A}$$

Transposition:
$$(\boldsymbol{A}^T)^T=\boldsymbol{A}$$
$$(\boldsymbol{A}+\boldsymbol{B})^T=\boldsymbol{A}^T+\boldsymbol{B}^T$$
$$(\alpha\boldsymbol{A})^T=\alpha\boldsymbol{A}^T$$
$$(\boldsymbol{A}\cdot\boldsymbol{B})^T=\boldsymbol{B}^T\cdot\boldsymbol{A}^T$$

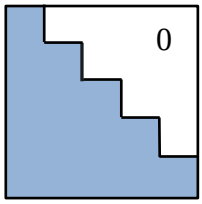
Differentiation:
$$\frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{A}+\boldsymbol{B})=\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{A}+\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{B}$$
$$\frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{A}\cdot\boldsymbol{B})=\left(\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{A}\right)\cdot\boldsymbol{B}+\boldsymbol{A}\cdot\left(\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{B}\right)$$

Matrizenmultiplikation:
$$\boldsymbol{A}\cdot(\boldsymbol{B}+\boldsymbol{C})=\boldsymbol{A}\cdot\boldsymbol{B}+\boldsymbol{A}\cdot\boldsymbol{C}$$
$$\boldsymbol{A}\cdot(\boldsymbol{B}\cdot\boldsymbol{C})=(\boldsymbol{A}\cdot\boldsymbol{B})\cdot\boldsymbol{C}$$
$$\boldsymbol{A}\cdot\boldsymbol{B}\neq\boldsymbol{B}\cdot\boldsymbol{A}$$ im Allgemeinen

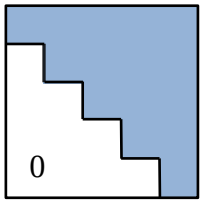
Skalarprodukt:
$$\boldsymbol{x}\cdot\boldsymbol{y}=\boldsymbol{y}\cdot\boldsymbol{x}$$
$$\boldsymbol{x}\cdot\boldsymbol{x}\geq 0\quad\forall\boldsymbol{x},\quad\boldsymbol{x}\cdot\boldsymbol{x}=0\Leftrightarrow\boldsymbol{x}=\boldsymbol{0}$$
$$\boldsymbol{x}\cdot\boldsymbol{y}=0\Leftrightarrow\boldsymbol{x},\boldsymbol{y}\text{ orthogonal}$$

Spezielle quadratische Matrizen

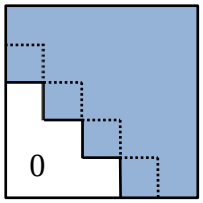
Linksdreiecksmatrix, untere Dreiecksmatrix (lower triangular $\boldsymbol{L}$)

$\boldsymbol{L}=$ 

Rechtsdreiecksmatrix, obere Dreiecksmatrix (upper triangular $\boldsymbol{R}$)

$\boldsymbol{R}=$ 

(obere) Hessenberg–Matrix

$\boldsymbol{H}=$ 

Einheitsmatrix
$$\boldsymbol{E}=\begin{bmatrix}1 & & 0\\ & \ddots & \\ 0 & & 1\end{bmatrix}$$

Diagonalmatrix

$$\boldsymbol{D}\cdot\boldsymbol{A}=\boldsymbol{D}\cdot\begin{bmatrix}\bar{\boldsymbol{a}}_1\\ \vdots\\ \bar{\boldsymbol{a}}_n\end{bmatrix}=\begin{bmatrix}d_1\bar{\boldsymbol{a}}_1\\ \vdots\\ d_n\bar{\boldsymbol{a}}_n\end{bmatrix},\quad \boldsymbol{D}^{-1}=\operatorname{diag}\left\{\frac{1}{d_i}\right\}\text{ für } d_i\neq 0$$

$$\boldsymbol{A}\cdot\boldsymbol{D}=[\boldsymbol{a}_1\dots\boldsymbol{a}_n]\cdot\boldsymbol{D}=[d_1\boldsymbol{a}_1\dots d_n\boldsymbol{a}_n]$$

Permutationsmatrix

$$\boldsymbol{P}_{rs}\cdot\boldsymbol{A}=\boldsymbol{P}_{rs}\cdot\begin{bmatrix}\vdots\\ \bar{\boldsymbol{a}}_r\\ \vdots\\ \bar{\boldsymbol{a}}_s\\ \vdots\end{bmatrix}=\begin{bmatrix}\vdots\\ \bar{\boldsymbol{a}}_s\\ \vdots\\ \bar{\boldsymbol{a}}_r\\ \vdots\end{bmatrix},\quad \boldsymbol{P}_{rs}=\begin{array}{c}\left[\begin{array}{ccccccccccc}1&&&&&&&&&&\\&\ddots&&&&&&&&&\\&&1&&&&&&&&\\&&&0&&&&1&&&\\&&&&1&&&&&&\\&&&&&\ddots&&&&&\\&&&&&&1&&&&\\&&&1&&&&0&&&\\&&&&&&&&1&&\\&&&&&&&&&\ddots&\\&&&&&&&&&&1\end{array}\right]\begin{array}{c}\\ \\ \\ r\\ \\ \\ \\ s\\ \\ \\ \\ \end{array}\\ \begin{array}{ccccccccccc}&&&r&&&&s&&&\end{array}\end{array}$$

$$\boldsymbol{A}\cdot\boldsymbol{P}_{rs}=[\dots\boldsymbol{a}_r\dots\boldsymbol{a}_s\dots]\cdot\boldsymbol{P}_{rs}=[\dots\boldsymbol{a}_s\dots\boldsymbol{a}_r\dots]$$

$\boldsymbol{P}_{rs}^T=\boldsymbol{P}_{rs}$	symmetrisch
$\boldsymbol{P}_{rs}^{-1}=\boldsymbol{P}_{rs}^T$	orthogonal
$\Rightarrow \boldsymbol{P}_{rs}^2=\boldsymbol{E}$	involutorisch
Allgemein:	$\boldsymbol{P}=[\boldsymbol{e}_{s_1}\dots\boldsymbol{e}_{s_n}]$ mit
	$\boldsymbol{e}_i$ Einsvektoren
	$(s_1,\dots,s_n)$: Permutation von $(1,\dots,n)$

Inverse Matrix $\boldsymbol{A}^{-1}$	$\boldsymbol{A}^{-1}\cdot\boldsymbol{A}=\boldsymbol{A}\cdot\boldsymbol{A}^{-1}=\boldsymbol{E}$
	$(\boldsymbol{A}\cdot\boldsymbol{B})^{-1}=\boldsymbol{B}^{-1}\cdot\boldsymbol{A}^{-1}$
Symmetrische Matrix	$\boldsymbol{A}=\boldsymbol{A}^T$
Schiefsymmetrische Matrix	$\boldsymbol{A}=-\boldsymbol{A}^T$
Positiv definite Matrix	$\boldsymbol{x}\cdot\boldsymbol{A}\cdot\boldsymbol{x}>0\quad\forall\boldsymbol{x}\neq\boldsymbol{0}$
Orthogonale Matrix $(\boldsymbol{A}\cdot\boldsymbol{A}^T=\boldsymbol{E})$	$\boldsymbol{A}^{-1}=\boldsymbol{A}^T$
Involutorische Matrix	$\boldsymbol{A}^2=\boldsymbol{E}$
Diagonaldominante Matrix	$\lvert a_{ii}\rvert>\sum_{j\neq i}\lvert a_{ij}\rvert\quad\forall i$

Normen

Normen sind Maße für Vektoren und Matrizen.

Vektornormen

Eine Vektornorm $\|\boldsymbol{x}\|$ ist eine Abbildung des $\mathbb{R}^n$ auf den $\mathbb{R}$ mit folgenden Eigenschaften:

1) $\|\boldsymbol{x}\| \geq 0 \quad \forall \boldsymbol{x} \in \mathbb{R}^n$ und $\|\boldsymbol{x}\| = 0 \Leftrightarrow \boldsymbol{x} = \boldsymbol{0}$
2) $\|\alpha \boldsymbol{x}\| = |\alpha| \|\boldsymbol{x}\| \quad \forall \alpha \in \mathbb{R}, \quad \boldsymbol{x} \in \mathbb{R}^n$
3) $\|\boldsymbol{x} + \boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\| \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$

Matrixnormen

Eine Matrixnorm $\|\boldsymbol{A}\|$ ist eine Abbildung des $\mathbb{R}^{m \times n}$ auf den $\mathbb{R}$ mit folgenden Eigenschaften:

1) $\|\boldsymbol{A}\| \geq 0 \quad \forall \boldsymbol{A} \in \mathbb{R}^{m \times n}$ und $\|\boldsymbol{A}\| = 0 \Leftrightarrow \boldsymbol{A} = \boldsymbol{0}$
2) $\|\alpha \boldsymbol{A}\| = |\alpha| \|\boldsymbol{A}\| \quad \forall \alpha \in \mathbb{R}, \quad \boldsymbol{A} \in \mathbb{R}^{m \times n}$
3) $\|\boldsymbol{A} + \boldsymbol{B}\| \leq \|\boldsymbol{A}\| + \|\boldsymbol{B}\| \quad \forall \boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{m \times n}$

Zusammenhänge zwischen Vektor– und Matrixnormen

- Eine Matrixnorm heißt **submultiplikativ**, wenn gilt
  $\|\boldsymbol{A} \cdot \boldsymbol{B}\| \leq \|\boldsymbol{A}\| \|\boldsymbol{B}\| \quad \forall \boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{m \times n}$
- Eine Matrixnorm $\|\boldsymbol{A}\|$ heißt **verträglich** mit einer Vektornorm $\|\boldsymbol{x}\|$, wenn gilt
  $\|\boldsymbol{A} \cdot \boldsymbol{x}\| \leq \|\boldsymbol{A}\| \|\boldsymbol{x}\| \quad \forall \boldsymbol{A} \in \mathbb{R}^{m \times n}, \quad \boldsymbol{x} \in \mathbb{R}^n$
- Die kleinste aller mit einer Vektornorm $\|\boldsymbol{x}\|$ verträglichen Matrixnormen $\|\boldsymbol{A}\|$ heißt **Grenzennorm** $\mathrm{lub}(\boldsymbol{A})$ (least upper bound):

  $$\mathrm{lub}(\boldsymbol{A}) := \max_{\boldsymbol{x} \neq \boldsymbol{0}} \frac{\|\boldsymbol{A} \cdot \boldsymbol{x}\|}{\|\boldsymbol{x}\|}, \quad \boldsymbol{A} \in \mathbb{R}^{m \times n}, \quad \boldsymbol{x} \in \mathbb{R}^n$$

  Anmerkung: Jede Grenzennorm ist submultiplikativ und es gilt $\mathrm{lub}(\boldsymbol{E}) = 1$

- Gängige Vektornormen und deren zugehörige Grenzennormen:

Vektornorm $\|\boldsymbol{x}\|$	zugehörige Grenzennorm $\mathrm{lub}(\boldsymbol{A})$
$\|\boldsymbol{x}\|_2 = \sqrt{\sum_i x_i^2} = \sqrt{\boldsymbol{x}\cdot\boldsymbol{x}}$ *Euklidische Norm*	$\|\boldsymbol{A}\|_2 = \max_{\boldsymbol{x}\neq\boldsymbol{0}} \sqrt{\frac{\boldsymbol{x}\cdot\boldsymbol{A}^T\cdot\boldsymbol{A}\cdot\boldsymbol{x}}{\boldsymbol{x}\cdot\boldsymbol{x}}} = \sqrt{\lambda_{max}(\boldsymbol{A}^T\cdot\boldsymbol{A})}$ *Euklidische Norm*
$\|\boldsymbol{x}\|_\infty = \max_{\boldsymbol{i}} \|x_i\|$ *Maximumsnorm*	$\|\boldsymbol{A}\|_\infty = \max_i \sum_j \|A_{ij}\|$ *Zeilensummennorm*

- Alle Vektornormen des $\mathbb{R}^n$ sind **äquivalent**, d.h. für jedes Paar $\|\boldsymbol{x}\|_a, \|\boldsymbol{x}\|_b$ von Normen gibt es Konstanten $m, M > 0$, so dass gilt:
  $m\|\boldsymbol{x}\|_a \leq \|\boldsymbol{x}\|_b \leq M\|\boldsymbol{x}\|_a \quad \forall \boldsymbol{x} \in \mathbb{R}^n$

Beispiel: $\|\boldsymbol{x}\|_\infty \leq \|\boldsymbol{x}\|_2 \leq \sqrt{n}\|\boldsymbol{x}\|_\infty$
$\frac{1}{\sqrt{n}}\|\boldsymbol{x}\|_2 \leq \|\boldsymbol{x}\|_\infty \leq \|\boldsymbol{x}\|_2$

Verfahrensvergleich zur Behandlung von linearen Gleichungssystemen

Name	Typ	Struktur-ausnutzung	Aufwand	Vorteile	Nachteile
Gauß ohne Pivotisierung	$\boldsymbol{L \cdot R}$		$\frac{n^3}{3}$		Kondition
Gauß mit Pivotisierung	$\boldsymbol{L \cdot R}$		$\frac{n^3}{3}$	Robustheit	
Gauß mit Speicherung	$\boldsymbol{L \cdot R}$		$\frac{n^3}{3}$		
Cholesky	$\boldsymbol{L \cdot L^T}$	$\boldsymbol{A = A^T > 0}$	$\frac{n^3}{6}$	Kondition Effizienz	Strukturaus-nutzung
Crout	$\boldsymbol{L \cdot R}$		$\frac{n^3}{3}$	Vektorisier-barkeit	
Bunch – Kaufman	$\boldsymbol{L \cdot D \cdot L^T}$	$\boldsymbol{A = A^T}$	$\frac{n^3}{6}$	Effizienz	Strukturaus-nutzung
Aasen	$\boldsymbol{L \cdot D \cdot L^T}$	$\boldsymbol{A = A^T}$	$\frac{n^3}{6}$	Effizienz	Strukturaus-nutzung
Householder	$\boldsymbol{Q \cdot R}$		$2\frac{n^3}{3}$	Kondition Ausgleichs-problem EWP	Aufwand
Givens	$\boldsymbol{Q \cdot R}$		$2\frac{n^3}{3}$	Kondition Ausgleichs-problem	Aufwand
Jacobi	Gesamt-schritt iterativ	$\boldsymbol{A = A^T > 0}$		große Systeme dünn besetzt	Struktur Konvergenz
Gauß–Seidel	Einzelschritt iterativ	$\boldsymbol{A = A^T > 0}$		große Systeme dünn besetzt	Struktur Konvergenz
SOR	Mittelung iterativ	$\boldsymbol{A = A^T > 0}$		große Systeme dünn besetzt	Struktur Konvergenz

E. Anderson et al.: *LAPACK User's Guide*.
Philadelphia: SIAM, 1992.
http://www.netlib.org/lapack/lug/lapack_lug.html

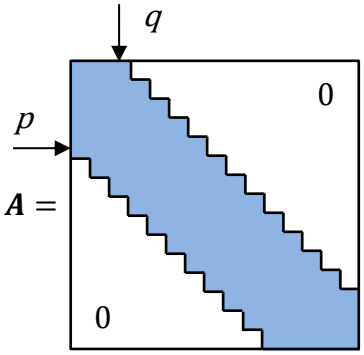
N.N.: LINPACK Index–Dokumentation. Über NETLIB abrufbar unter:
http://www.netlib.no/netlib/linpack/index.html

Matrix–Zerlegung bei Bandstrukturen

Bei der Berechnung von FEM (schwach besetzte Matrizen), MKS (kartesische Koordinaten) und der Lösung von PDE (leicht überlappende Gebietsdiskretisierung) treten oft schwach besetzte Matrizen mit Bandstruktur auf.

Obere Bandbreite: $A_{ij} = 0 \quad \forall j > i + q$

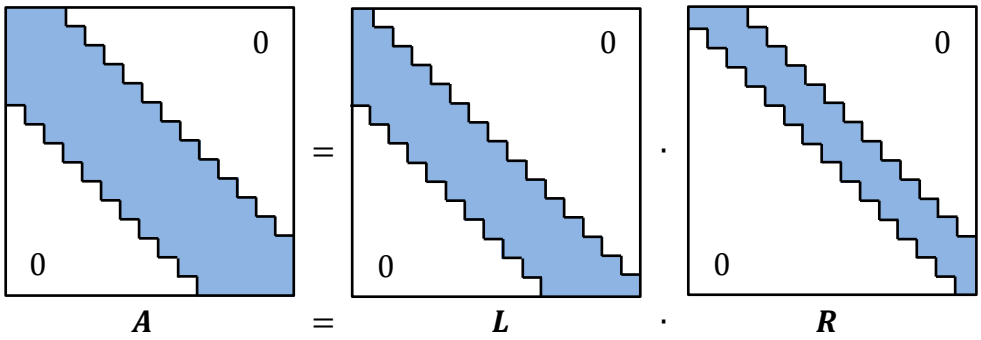
Untere Bandbreite: $A_{ij} = 0 \quad \forall i > j + p$



Bandstrukturen pflanzen sich in den Zerlegungsmatrizen fort: $\boldsymbol{A} = \boldsymbol{L} \cdot \boldsymbol{R}$

Bandbreite von $\boldsymbol{A} = (p, q) \Leftrightarrow$ Bandbreite von $\boldsymbol{L} = (p, 1)$

Bandbreite von $\boldsymbol{R} = (1, q)$



Aufwand von Algorithmen bei Bandstrukturen:

Gauß ohne Pivotisierung:	$\begin{cases} npq - \frac{pq^2}{2} - \frac{p^3}{6} + pn & \forall p \le q \\ npq - \frac{pq^2}{2} - \frac{q^3}{6} + qn & \forall p > q \end{cases}$
Cholesky:	$\frac{np^2}{2} - \frac{p^3}{3} + \frac{3(np - p^2)}{2} + n$
Vorwärtseinsetzen:	$np - \frac{p^2}{2}$
Rückwärtseinsetzen:	$n(q+1) - \frac{q^2}{2}$

Determinanten

Die Determinante einer quadratischen, reellen Matrix $A = [\boldsymbol{a}_1 \dots \boldsymbol{a}_n] \in \mathbb{R}^{n\times n}$ ist durch folgende Grundgesetze definiert:

(1) $\det[\boldsymbol{a}_1 \dots \boldsymbol{a}_i + \boldsymbol{b}_i \dots \boldsymbol{a}_n] = \det[\boldsymbol{a}_1 \dots \boldsymbol{a}_i \dots \boldsymbol{a}_n] + \det[\boldsymbol{a}_1 \dots \boldsymbol{b}_i \dots \boldsymbol{a}_n]$

(2) $\det[\boldsymbol{a}_1 \dots \lambda\boldsymbol{a}_i \dots \boldsymbol{a}_n] = \lambda\det[\boldsymbol{a}_1 \dots \boldsymbol{a}_i \dots \boldsymbol{a}_n]$

(3) $\det\big[\dots \boldsymbol{a}_i \dots \boldsymbol{a}_j \dots\big] = -\det[\dots \boldsymbol{a}_j \dots \boldsymbol{a}_i \dots]$

(4) $\det \boldsymbol{E} = 1$

Determinantenformel:

$$\begin{aligned}\det \boldsymbol{A} &= \det[\boldsymbol{a}_1 \dots \boldsymbol{a}_n] \\ &= \det[a_{11}\boldsymbol{e}_1 + \dots + a_{n1}\boldsymbol{e}_n, \dots, a_{1n}\boldsymbol{e}_1 + \dots + a_{nn}\boldsymbol{e}_n] \\ &= \sum_{i_1,\dots,i_n\in\{1,\dots,n\}} a_{i_1 1}a_{i_2 2}\dots a_{i_n n}\det[e_{i_1}\dots e_{i_n}] \\ &= \sum_{\substack{(i_1\dots i_n)\\ \text{Permutation von } (1\dots n)}} \operatorname{sign}(i_1\dots i_n)a_{i_1 1}a_{i_2 2}\dots a_{i_n n}\end{aligned}$$

Rechenregeln:

- $\det[\dots \boldsymbol{a} \dots \boldsymbol{a} \dots] = 0$
- $\det[\boldsymbol{a}_1 \dots \boldsymbol{0} \dots \boldsymbol{a}_n] = 0$
- $\det\big[\dots \boldsymbol{a}_i \dots \boldsymbol{a}_j + \lambda\boldsymbol{a}_i \dots\big] = \det[\dots \boldsymbol{a}_i \dots \boldsymbol{a}_j \dots]$
- $\det \boldsymbol{A}^T = \det \boldsymbol{A}$
- $\det \boldsymbol{A}^{-1} = \frac{1}{\det \boldsymbol{A}}$
- $\det \boldsymbol{L} = \prod_{i=1}^{n} L_{ii}, \quad \det \boldsymbol{R} = \prod_{i=1}^{n} R_{ii}$
- $\det \boldsymbol{A} \neq 0 \Leftrightarrow \boldsymbol{A}$ regulär
- $\det(\boldsymbol{A}\cdot\boldsymbol{B}) = \det(\boldsymbol{A})\det(\boldsymbol{B})$

Werkzeuge zur Behandlung von linearen Gleichungssystemen

	Bibliotheken	Abgeschlossene Programme
symbolisch	muParser (C++, C#)	Maple Mathematica MuPad ...
numerisch	IMSL NAG LAPACK / BLAS LINPACK Numerical Recipes matplotlib (Python) ...	Matlab Scilab Octave ...

Matlab

Mit Hilfe von Matlab können viele Analysemethoden einfach angewandt werden. Matlab basiert auf effizienten Berechnungsmethoden der linearen Algebra.

```
A = [ 1 2 4; 2 13 23; 4 23 77 ]
b = [ 3 -4 5 ]'
A * b
A \ b
inv(A)
inv(A) * b
L = chol(A)'
[ L, U ] = lu(A)
L * U
[ Q, R ] = qr(A)
Q * R
pinv([ 1 2 4 7; 2 13 23 30; 4 23 77 100 ])
pinv([ 1 2 4; 2 13 23; 4 23 77; 8 33 140 ])
```

Maple

Maple ermöglicht das symbolische Rechnen mit dem Computer.

```
with(linalg);
A := matrix(3, 3, [ [ 1, 2, 4 ], [ 2, 13, 23 ], [ 4, 23, 77 ] ]);
b := vector(3, [ 3, -4, 5 ]);
inverse(A);
multiply(inverse(A), b);
linsolve(A, b);
A := matrix(3, 3, [ [ a11, a21, a31 ], [ a12, a22, a32 ],
                    [ a13, a23, a33 ] ]);
b := vector(3, [ b1, b2, b3 ]);
inverse(A);
multiply(inverse(A), b);
linsolve(A, b);
```

LAPACK/BLAS:

Die numerische Bibliothek LAPACK stellt eine umfassende Sammlung hochentwickelter Numerik–Unterprogramme für die Analyse linearer Probleme dar. Sie greift für die Elementarrechnungen auf die maschinenabhängig optimierte Bibliothek BLAS zurück.

```
NAME
      DGETRF  -  compute an LU factorization of a general M-by-N
      matrix A using partial pivoting with row interchanges
SYNOPSIS
      SUBROUTINE DGETRF( M, N, A, LDA, IPIV, INFO )
      INTEGER           INFO, LDA, M, N
      INTEGER           IPIV( * )
      DOUBLE PRECISION A( LDA, * )
PURPOSE
      DGETRF computes an LU factorization of a general M-by-N
      matrix A using partial pivoting with row interchanges.
      The factorization has the form
          A = P * L * U
      where P is a permutation matrix, L is lower triangular
      with unit diagonal elements (lower trapezoidal if m > n),
      and U is upper triangular (upper trapezoidal if m < n).
      This is the right-looking Level 3 BLAS version of the algorithm.
ARGUMENTS
      M       (input) INTEGER
               The number of rows of the matrix A.  M >= 0.
      N       (input) INTEGER
               The number of columns of the matrix A.  N >= 0.
      A       (input/output) DOUBLE PRECISION array, dimension (LDA,N)
               On entry, the M-by-N matrix to be factored. On
               exit, the factors L and U from the factorization
               A = P*L*U; the unit diagonal elements of L are not stored.
      LDA     (input) INTEGER
               The leading dimension of the array A.  LDA >= max(1,M).
      IPIV    (output) INTEGER array, dimension (min(M,N))
               The pivot indices; for 1 <= i <= min(M,N), row i
               of the matrix was interchanged with row IPIV(i).
      INFO    (output) INTEGER
               = 0:  successful exit
               < 0:  if INFO = -i, the i-th argument had an illegal value
               > 0:  if INFO = i, U(i,i) is exactly zero. The
               factorization has been completed, but the factor U
               is exactly singular, and division by zero will
               occur if it is used to solve a system of equations.
NAME
      DGETRS  - solve a system of linear equations  A * X = B or
      A' * X = B with a general N-by-N matrix  A  using the  LU
      factorization computed by DGETRF
SYNOPSIS
      SUBROUTINE DGETRS( TRANS,  N,  NRHS, A, LDA, IPIV, B, LDB, INFO )
      CHARACTER         TRANS
      INTEGER           INFO, LDA, LDB, N, NRHS
      INTEGER           IPIV( * )
      DOUBLE            PRECISION A( LDA, * ), B( LDB, * )
```

Informationsbeschaffung zur Numerischen Mathematik im Internet:

Diskussionsforen in News–Gruppen:

`news://news.uni-stuttgart.de/sci.math.num-analysis`

Software–Bibliotheken im Internet:

`http://www.netlib.no/`
`http://www.netlib.org/`

Eigenwertproblem

Das Eigenwertproblem
$\boldsymbol{A} \cdot \boldsymbol{x} = \lambda \boldsymbol{x}$ oder $(\lambda \boldsymbol{E} - \boldsymbol{A}) \cdot \boldsymbol{x} = \boldsymbol{0}$
mit $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ hat nichttriviale Lösungen, falls

$$(\lambda \boldsymbol{E} - \boldsymbol{A}) singulär \Leftrightarrow p(\lambda) \coloneqq \det(\lambda \boldsymbol{E} - \boldsymbol{A}) = \lambda^n + a_{n-1}\lambda^{n-1} + \cdots + a_0 = 0$$

charakteristische Gleichung

Lösungen:

Eigenwerte	$\lambda_i \in \mathbb{C}$,	$i = 1(1)n$,
(Rechts–)Eigenvektoren	$\boldsymbol{x}_i \in \mathbb{C}^n$,	$i = 1(1)n$,
	$\boldsymbol{A} \cdot \boldsymbol{x}_i = \lambda_i \boldsymbol{x}_i$,	$i = 1(1)n$.

(Algebraische) Vielfachheit v_k: Sind λ_k, $k = 1(1)m \leq n$, verschiedene Nullstellen und ist λ_k jeweils eine v_k–fache Nullstelle von $\boldsymbol{p}(\lambda)$, dann gilt:

$$\boldsymbol{p}(\lambda) = (\lambda - \lambda_1)^{v_1}(\lambda - \lambda_2)^{v_2} \ldots (\lambda - \lambda_m)^{v_m} = 0, \quad \sum_{k=1}^{m} v_k = n\,.$$

Defekt (geometrische Vielfachheit) d_k: Der Defekt einer Nullstelle λ_k ist definiert als Rangabfall der Matrix $(\lambda_k \underline{\underline{\mathbf{E}}} - \underline{\underline{\mathbf{A}}})$:

$$d_k \coloneqq n - \mathrm{Rg}(\lambda_k \boldsymbol{E} - \boldsymbol{A}), \quad 1 \leq d_k \leq v_k, \quad k = 1(1)m.$$

Damit existieren zu jedem Eigenwert λ_k genau d_k linear unabhängige Eigenvektoren, die einen d_k–dimensionalen Eigenraum aufspannen:

$$\mathrm{L}(\lambda_k) \coloneqq \{\boldsymbol{x} \in \mathbb{C}^n | (\lambda_k \boldsymbol{E} - \boldsymbol{A}) \cdot \boldsymbol{x} = \boldsymbol{0}\}, \quad k = 1(1)m.$$

Eigenschaften:

- Eigenwerte sind invariant unter Ähnlichkeitstransformationen:
  $\lambda(\boldsymbol{T}^{-1} \cdot \boldsymbol{A} \cdot \boldsymbol{T}) = \lambda(\boldsymbol{A}), \quad \boldsymbol{T}$ regulär

- Eigenwerte einer Dreiecksmatrix $\boldsymbol{L}$ bzw. $\boldsymbol{R}$ entsprechen den Diagonalelementen:
  $\lambda_i = L_{ii}$ bzw. $\lambda_i = R_{ii}, \quad i = 1(1)n.$

- Spektralverschiebung: Die Matrix $(\boldsymbol{A} + s\boldsymbol{E})$ hat die Eigenwerte $\mu_k = (\lambda_k + s)$, $k = 1(1)m$, wobei λ_k die Eigenwerte von $\boldsymbol{A}$ sind.

Analogien zwischen reellen und komplexen Matrizen:

$\boldsymbol{A} \in \mathbb{R}^{n\times n}$	$\boldsymbol{C} = \boldsymbol{A} + i\boldsymbol{B} \in \mathbb{C}^{n\times n}, \quad \boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n\times n}$
Transponierte $\boldsymbol{A}^T$	Konjugiert Transponierte $\boldsymbol{C}^* = \overline{\boldsymbol{C}}^T = \boldsymbol{A}^T - i\boldsymbol{B}^T$
Symmetrische Matrix $\boldsymbol{A}^T = \boldsymbol{A}$	Hermitesche Matrix $\boldsymbol{C}^* = \boldsymbol{C}$
Schiefsymmetrische Matrix $\boldsymbol{A}^T = -\boldsymbol{A}$	Schiefhermitesche Matrix $\boldsymbol{C}^* = -\boldsymbol{C}$
Orthogonale Matrix $\boldsymbol{A}^T \cdot \boldsymbol{A} = \boldsymbol{A} \cdot \boldsymbol{A}^T = \boldsymbol{E}$	Unitäre Matrix $\boldsymbol{C}^* \cdot \boldsymbol{C} = \boldsymbol{C} \cdot \boldsymbol{C}^* = \boldsymbol{E}$

Schursche Normalform

Jede Matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ lässt sich mit einer unitären Transformation $\boldsymbol{T}^* \cdot \boldsymbol{T} = \boldsymbol{E}, \quad \boldsymbol{T} \in \mathbb{C}^{n\times n}$ auf Schursche Normalform (Rechtsdreiecksform) transformieren:

$$\boldsymbol{T}^* \cdot \boldsymbol{A} \cdot \boldsymbol{T} = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix} = \boldsymbol{R}, \quad \boldsymbol{R} \in \mathbb{C}^{n\times n}$$

Speziell:

- Für symmetrische reelle Matrizen führt die Schursche Normalform auf Diagonalform mit reellen Eigenwerten

- Hat eine Matrix nur reelle Eigenwerte, dann gibt es eine Kongruenztransformation $\boldsymbol{Q} \in \mathbb{R}^{n\times n}, \quad \boldsymbol{Q}^T \cdot \boldsymbol{Q} = \boldsymbol{E}$, so dass $\boldsymbol{Q}^T \cdot \boldsymbol{A} \cdot \boldsymbol{Q} = \boldsymbol{R} \quad \boldsymbol{R} \in \mathbb{R}^{n\times n}$.

Praktische Lösung des Eigenwertproblems

Unterteilung der Eigenwertlösung in elementare Grundaufgaben

Steigerung von Effizienz und Kondition durch Umformulierung des Eigenwertproblems mit Ähnlichkeitstransformationen:

```
      subroutine rg(nm,n,a,wr,wi,z,iv1,fv1,ierr)
c
c     this subroutine calls the recommended sequence of
c     subroutines from the eigensystem subroutine package (eispack)
c     to find the eigenvalues and eigenvectors
c     of a real general matrix.
c
c     on input
c        nm  must be set to the row dimension of the two-dimensional
c        array parameters as declared in the calling program
c        dimension statement.
c        n  is the order of the matrix  a.
c        a  contains the real general matrix.
c     on output
c        wr  and  wi  contain the real and imaginary parts,
c        respectively, of the eigenvalues.  complex conjugate
c        pairs of eigenvalues appear consecutively with the
c        eigenvalue having the positive imaginary part first.
c        z  contains the real and imaginary parts of the
c        eigenvectors. if the j-th eigenvalue is real, the
c        j-th column of  z  contains its eigenvector.  if the j-th
c        eigenvalue is complex with positive imaginary part, the
c        j-th and (j+1)-th columns of  z  contain the real and
c        imaginary parts of its eigenvector.  the conjugate of this
c        vector is the eigenvector for the conjugate eigenvalue.
c        ierr  is an integer output variable set equal to an error
c           completion code described in the documentation for hqr
c           and hqr2.  the normal completion code is zero.
c        iv1  and  fv1  are temporary storage arrays.
c
      call balanc(nm,n,a,is1,is2,fv1)
      call elmhes(nm,n,is1,is2,a,iv1)
      call eltran(nm,n,is1,is2,a,iv1,z)
      call hqr2(nm,n,is1,is2,a,wr,wi,z,ierr)
      call balbak(nm,n,is1,is2,fv1,n,z)
      return
      end
```

Balancieren

Skalieren des Eigenwertproblems zur Konditionsverbesserung durch Diagonalmatrizen:

```
      subroutine balanc(nm,n,a,low,igh,scale)
c
c     this subroutine balances a real matrix and isolates
c     eigenvalues whenever possible.
c
c     on input
c        n is the order of the matrix.
c        a contains the input matrix to be balanced.
c     on output
c        a contains the balanced matrix.
```

Rücktransformation der Skalierung auf Ursprungsdarstellung nach der Lösung des Eigenwertproblems:

```
      subroutine balbak(nm,n,low,igh,scale,m,z)
c
c     this subroutine forms the eigenvectors of a real general
c     matrix by back transforming those of the corresponding
c     balanced matrix determined by  balanc.
c
c     on input
c        n is the order of the matrix.
c        z  contains the real and imaginary parts of the eigen-
c          vectors to be back transformed in its first m columns.
c     on output
c        z contains the real and imaginary parts of the
c          transformed eigenvectors in its first m columns.
```

Reduktion auf obere Hessenberg–Form

Transformation auf obere Hessenberg–Form zur Reduktion des Rechenaufwands durch Ähnlichkeitstransformationen:

1) Permutationen
2) Eliminationen
3) Spiegelungen
4) Drehungen

```
      subroutine elmhes(nm,n,low,igh,a,int)
c
c     given a real general matrix, this subroutine
c     reduces a submatrix situated in rows and columns
c     low through igh to upper hessenberg form by
c     stabilized elementary similarity transformations.
c
c     on input
c        n is the order of the matrix.
c        a contains the input matrix.
c     on output
c        a contains the hessenberg matrix.  the multipliers
c          which were used in the reduction are stored in the
c          remaining triangle under the hessenberg matrix.
```

```
      subroutine eltran(nm,n,low,igh,a,int,z)
c
c     this subroutine accumulates the stabilized elementary
c     similarity transformations used in the reduction of a
c     real general matrix to upper hessenberg form by  elmhes.
c
c     on input
c        n is the order of the matrix.
c        a contains the multipliers which were used in the
c          reduction by  elmhes  in its lower triangle
c          below the subdiagonal.
c     on output
c        z contains the transformation matrix produced in the
c          reduction by  elmhes.
```

Iterative Eigenwertlösung

Überführung des Eigenwertproblems in Rechtsdreiecksform durch

1) Jacobi–Verfahren
2) LR–Verfahren
3) QR–Verfahren

```
      subroutine hqr2(nm,n,low,igh,h,wr,wi,z,ierr)
c
c     this subroutine finds the eigenvalues and eigenvectors
c     of a real upper hessenberg matrix by the qr method.  the
c     eigenvectors of a real general matrix can also be found
c     if  elmhes  and  eltran  or  orthes  and  ortran  have
c     been used to reduce this general matrix to hessenberg form
c     and to accumulate the similarity transformations.
c
c     on input
c        n is the order of the matrix.
c        h contains the upper hessenberg matrix.
c        z contains the transformation matrix produced by  eltran
c          after the reduction by  elmhes, or by  ortran  after the
c          reduction by  orthes, if performed.  if the eigenvectors
c          of the hessenberg matrix are desired, z must contain the
c          identity matrix.
c     on output
c        wr and wi contain the real and imaginary parts,
c          respectively, of the eigenvalues.  the eigenvalues
c          are unordered except that complex conjugate pairs
c          of values appear consecutively with the eigenvalue
c          having the positive imaginary part first.  if an
c          error exit is made, the eigenvalues should be correct
c          for indices ierr+1,...,n.
c        z contains the real and imaginary parts of the eigenvectors.
c          if the i-th eigenvalue is real, the i-th column of z
c          contains its eigenvector.
c          if the i-th eigenvalue is complex
c          with positive imaginary part, the i-th and (i+1)-th
c          columns of z contain the real and imaginary parts of its
c          eigenvector.  the eigenvectors are unnormalized.  if an
c          error exit is made, none of the eigenvectors
c          has been found.
```

Werkzeuge zur numerischen Behandlung von Eigenwertproblemen

	Bibliotheken	Abgeschlossene Programme
symbolisch		Maple Mathematica MuPad .....
numerisch	IMSL NAG LAPACK / BLAS EISPACK Numerical Recipes	Matlab Scilab Octave .....

Matlab

Mit Hilfe von Matlab können Eigenwertaufgaben sehr einfach gelöst werden. Dabei wird sowohl das spezielle als auch das allgemeine Eigenwertproblem unterstützt. Die Lösung wird durch die in Abschnitt 4.6 dargestellten Methoden generiert.

```
A = [ 4 -1 1; 9 -8 9; 11 -11 12 ]
[ ev, ew ] = eig(A)

B = [ 1 0 0; 0 1 0; 0 0 1]
[ ev, ew ] = eig(A, B)

m1 = 600
m2 = 40
d1 = 2400
k1 = 15000
k2 = 160000
A = [ -d1/m1 d1/m1 -k1/m1 k1/m1; d1/m2 -d1/m2 k1/m2 -(k1+k2)/m2; ...
       1 0 0 0; 0 1 0 0 ]
[ ev, ew ] = eig(A)
```

Maple

Mit Maple können Eigenwerte und Eigenvektoren symbolisch und numerisch bestimmt werden. Dabei wird sowohl das spezielle als auch das allgemeine Eigenwertproblem unterstützt.

```
with(linalg);
A := matrix(3, 3, [ [ 4, -1, 1 ], [ 9, -8, 9 ], [ 11, -11, 12 ] ]);
eigenvals(A);

eigenvects(A);

charpoly(A, 'x');
det(evalm(A - x));
fsolve(");

B := matrix(3, 3, [ [ 1, 0, 0 ], [ 0, 1, 0 ], [ 0, 0, 1 ] ]);
eigenvals(A, B);

m1 := 600.0;
m2 := 40.0;
d1 := 2400.0;
k1 := 15000.0;
k2 := 160000.0;
A := matrix(4,4,
           [[ -d1/m1, d1/m1, -k1/m1, k1/m1 ],
            [ d1/m2, -d1/m2, k1/m2, -(k1+k2)/m2 ],
            [ 1.0, 0.0, 0.0, 0.0 ],
            [ 0.0, 1.0, 0.0, 0.0 ]]);
eigenvals(A);
eigenvects(A);

A := matrix(3, 3, [ [ a11, a21, a31 ], [ a12, a22, a32 ],
                    [ a13, a23, a33 ] ]);
eigenvals(A);
eigenvects(A);
```

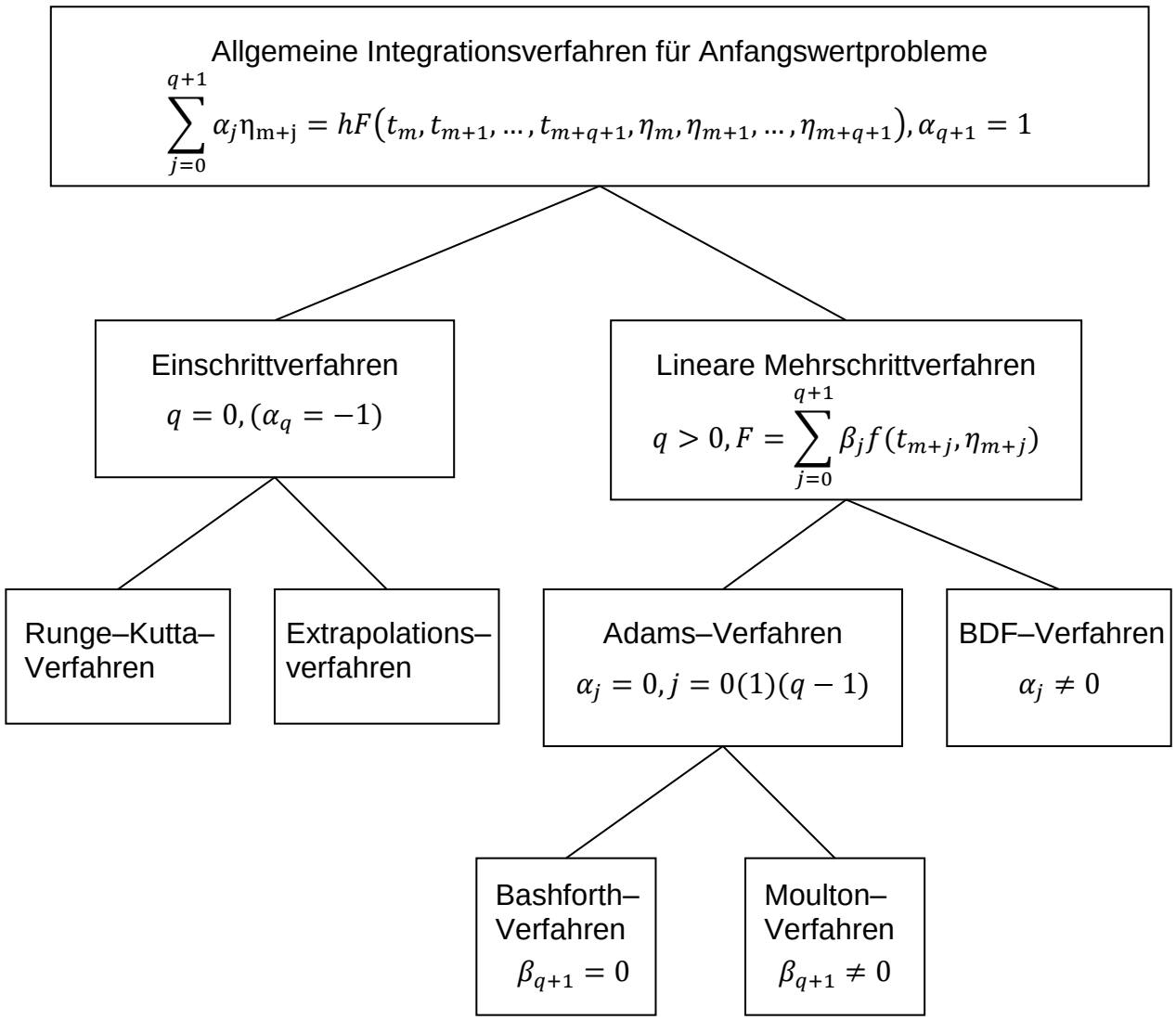
LAPACK/BLAS

Als Teil der Methoden zur Behandlung von Linearen Algebra Problemen enthält die Bibliothek LAPACK auch leistungsfähige Routinen zur Behandlung von Eigenwertproblemen. Dabei werden symmetrische/hermitische und nichtsymmetrsiche sowie speziell vorkonditionierte (Bsp. Schursche Form) Matrizen reell und komplex unterstützt.

```
NAME
    DGEEVX - compute for an N-by-N real nonsymmetric matrix A,
     the eigenvalues and, optionally,  the  left  and/or  right
     eigenvectors

SYNOPSIS
     SUBROUTINE DGEEVX( BALANC, JOBVL, JOBVR, SENSE, N, A, LDA,
                        WR, WI, VL, LDVL, VR, LDVR,  ILO,  IHI,
                        SCALE,  ABNRM,  RCONDE,  RCONDV,  WORK,
                        LWORK, IWORK, INFO )

PURPOSE
     DGEEVX computes for an N-by-N real nonsymmetric matrix  A,
     the  eigenvalues  and,  optionally,  the left and/or right
     eigenvectors.

     Optionally also, it computes a balancing transformation to
     improve  the conditioning of the eigenvalues and eigenvec-
     tors (ILO, IHI, SCALE, and  ABNRM),  reciprocal  condition
     numbers  for the eigenvalues (RCONDE), and reciprocal con-
     dition numbers for the right eigenvectors (RCONDV).

     The left eigenvectors of A  are  the  same  as  the  right
     eigenvectors  of  A**T.  If u(j) and v(j) are the left and
     right eigenvectors,  respectively,  corresponding  to  the
     eigenvalue lambda(j), then (u(j)**T)*A = lambda(j)*(u(j)**T)
     and A*v(j) = lambda(j) * v(j).

     The computed eigenvectors are normalized to have Euclidean
     norm equal to 1 and largest component real.

     Balancing a matrix means permuting the rows and columns to
     make it more nearly upper triangular, and applying a diag-
     onal similarity transformation D * A * D**(-1), where D is
     a diagonal matrix, to make its rows and columns closer  in
     norm  and  the  condition  numbers  of its eigenvalues and
     eigenvectors smaller.  The computed  reciprocal  condition
     numbers correspond to the balanced matrix.  Permuting rows
     and columns will not  change  the  condition  numbers  (in
     exact  arithmetic) but diagonal scaling will.  For further
     explanation of balancing, see section 4.10.2 of the LAPACK
     Users' Guide.
```

Vergleich der Integrationsverfahren für Anfangswertprobleme

Einordnung der behandelten Integrationsverfahren



Einschrittverfahren

- meist für einfache Systeme und mittlere bis geringe Genauigkeitsanforderungen
- inzwischen leistungsfähige Verfahren hoher Ordnung verfügbar
- einfacher Code
- Schrittweitenänderung relativ einfach möglich
- robust gegen Unstetigkeiten
  (Coulombsche Reibung, Lagerspiele, Anschläge, etc.)

Extrapolationsverfahren

- nur bei sehr hohen Genauigkeitsanforderungen effizient
- geringer Overhead
- Schrittweitenänderung einfach möglich
- große Grundschrittweite ungünstig für spezielle Anwendungen

Mehrschrittverfahren

- sehr effizient bei komplexer rechter Seite (mechanische Systeme)
  für mittlere und hohe Genauigkeiten
- großer Overhead
- benötigt Anlaufstrecke
- Schrittweitenänderung aufwendig
- empfindlich gegen Unstetigkeiten
- implizite BDF–Verfahren sehr leistungsfähig bei (numerisch) steifen Systemen

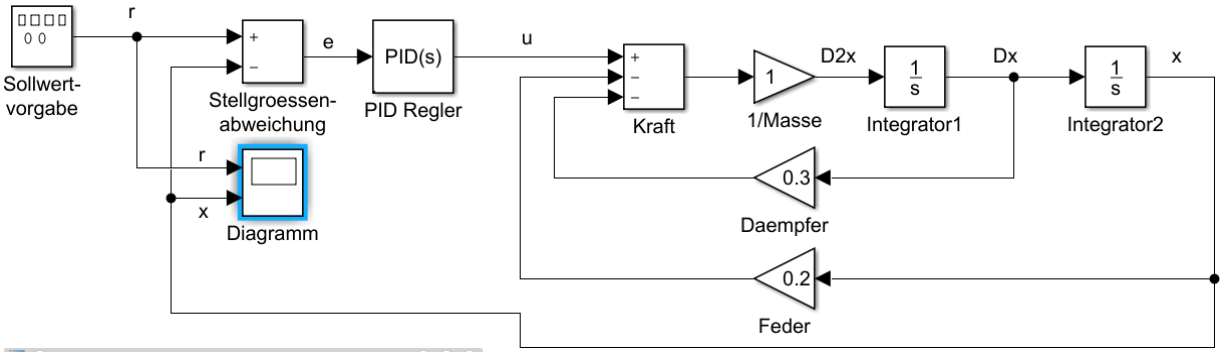
Werkzeuge zur Behandlung von Anfangswertproblemen

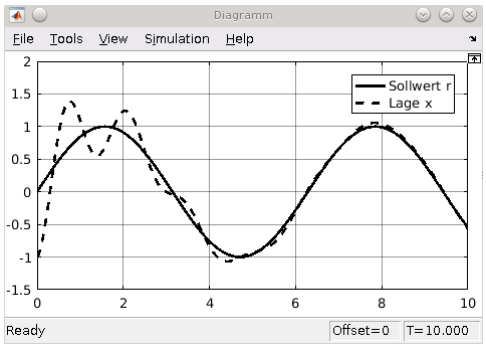
		Bibliotheken	Abgeschlossene Programme
symbolisch			Maple Mathematica MuPad ...
numerisch		IMSL NAG ODEPACK Shampine & Gordon MBSPACK MBSSIM Numerical Recipes ...	Matlab/Simulink Scilab NEWMOS ...

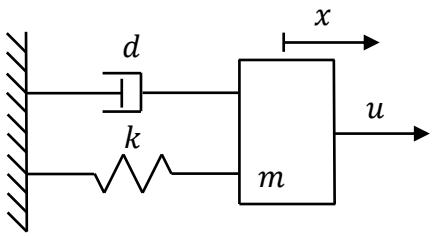
Simulink

Simulink ist Bestandteil der Matlab–Umgebung und eignet sich zur blockorientierten Modellierung und Simulation dynamischer Systeme. Über eine graphische Oberfläche wird der Rechenplan des Modells erstellt, wobei auf Standardblöcke zurückgegriffen werden kann. Für die numerische Integration stehen in Simulink die Einschrittverfahren Euler, RK3 und RK5 zur Verfügung. Ferner werden die Mehrschrittverfahren Adams (PECE) und Gear (ein BDF–Code) angeboten.

Beispiel: Lageregelung eines Einmassenschwingers durch PID–Regelung







Maple

Mit Maple können Differentialgleichungen symbolisch (oder falls alle Zahlenwerte gegeben sind auch numerisch) integriert werden. Dieser Ansatz ist auf einfache Probleme beschränkt.

```
int(x^2+x^3, x);
int(x^2+x^3, x=1..100);
```

NEWMOS

NEWMOS ist ein Simulator für modulare Systeme, der am Institut B für Mechanik entwickelt wurde. Er erlaubt die Simulation von differential–algebraischen Systemen und die Durchführung von Multirate–Simulationen.

Beispiel: Multirate–Simulation des Viertelfahrzeugs mit aktivem Dämpfer

```
define mbs_system (viertel)
   with parameters (m1, m2, k2) inputs (F, u) outputs (yap, yrp)
   by positions (ya, yr) velocities (yaps, yrps)
{
   M[0][0] = m1;
   M[1][1] = m2;
   qe[0] = F;
   qe[1] = -F - k2 * (yr - u);
   K[0][0] = 1;
   K[1][1] = 1;
   y[0] = yaps;
   y[1] = yrps;
};
viertel.m1 = 600;
viertel.m2 = 40;
viertel.k2 = 160000;

define ss_system (daempfer)
   with parameters (Td, D) inputs (yap, yrp) outputs (F) by states (Fs)
{
   xp[0] = (D * (yrp - yap) - Fs) / Td;
   y[0] = Fs;
};
daempfer.Td = 1.0e-5;
daempfer.D = 2400;

define ss_system (anregung)
   with outputs (u) by { y[0] = 0.1; };

connect(viertel.F, daempfer.F);
connect(viertel.u, anregung.u);
connect(daempfer.yap, viertel.yap);
connect(daempfer.yrp, viertel.yrp);
```

```
load integrator (int1) of type (ruku4) from file (ruku4.so);
int1.h = 1.0e-3;
load buffer (buf1) of type (extrapolation) from file (extrapol.so);
buf1.order = 3;

load integrator (int2) of type (ruku4) from file (ruku4.so);
int2.h = 15e-6;
load buffer (buf2) of type (extrapolation) from file (extrapol.so);
buf1.order = 3;

write (t, all states, all derivatives, all outputs) plain to
   file (viertel.out) with stepsize (0.001);

setup (set1) by (viertel, anregung, int1, buf1);
setup (set2) by (daempfer, int2, buf2);

simulate (set1, set2) from (0) to (1) with timeslice (0.001);
```

ODEPACK

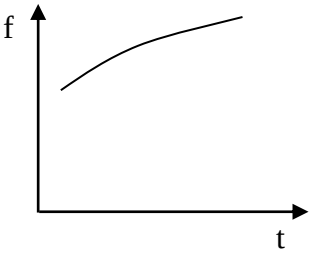
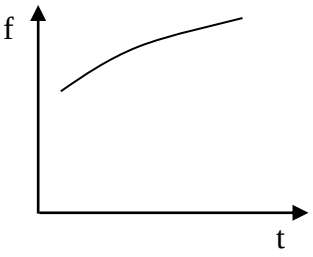
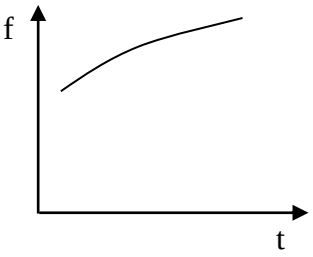
Die numerische Bibliothek ODEPACK stellt eine umfassende Sammlung hochentwickelter Numerik–Unterprogramme für die Integration von Anfangswertproblemen dar. Dabei werden insbesondere Mehrschrittverfahren unterstützt. Es existieren auch Methoden zur automatischen Umschaltung zwischen steifen (BDF) und nichtsteifen (PECE) Bereichen. Ferner werden differential–algebraische Systeme in der Index–1–Form unterstützt.

```
      subroutine lsode (f, neq, y, t, tout, itol, rtol, atol, itask,
     1                  istate, iopt, rwork, lrw, iwork, liw, jac, mf)
c-----------------------------------------------------------------------
c this is the march 30, 1987 version of
c lsode.. livermore solver for ordinary differential equations.
c this version is in double precision.
c
c lsode solves the initial value problem for stiff or nonstiff
c systems of first order ode-s,
c       dy/dt = f(t,y) ,  or, in component form,
c       dy(i)/dt = f(i) = f(i,t,y(1),y(2),...,y(neq)) (i = 1,...,neq).
c lsode is a package based on the gear and gearb packages, and on the
c october 23, 1978 version of the tentative odepack user interface
c standard, with minor modifications.
c-----------------------------------------------------------------------
c reference..
c       alan c. hindmarsh,  odepack, a systematized collection of ode
c       solvers, in scientific computing, r. s. stepleman et al. (eds.),
c       north-holland, amsterdam, 1983, pp. 55-64.
c-----------------------------------------------------------------------
c author and contact.. alan c. hindmarsh,
c                      computing and mathematics research div., l-316
c                      lawrence livermore national laboratory
c                      livermore, ca 94550.
c-----------------------------------------------------------------------
c summary of usage.
c
```

```
c first provide a subroutine of the form..
c                subroutine f (neq, t, y, ydot)
c                   dimension y(neq), ydot(neq)
c which supplies the vector function f by loading ydot(i) with f(i).
c
c f       = name of subroutine for right-hand side vector f.
c           this name must be declared external in calling program.
c neq     = number of first order ode-s.
c y       = array of initial values, of length neq.
c t       = the initial value of the independent variable.
c tout    = first point where output is desired (.ne. t).
c itol    = 1 or 2 according as atol (below) is a scalar or array.
c rtol    = relative tolerance parameter (scalar).
c atol    = absolute tolerance parameter (scalar or array).
c itask   = 1 for normal computation of output values of y at t = tout.
c istate  = integer flag (input and output).  set istate = 1.
c iopt    = 0 to indicate no optional inputs used.
c rwork   = real work array of length at least..
c                  20 + 16*neq                    for mf = 10,
c                  22 +  9*neq + neq**2           for mf = 22,
c lrw     = declared length of rwork (in user-s dimension).
c iwork   = integer work array of length at least..
c                  20         for mf = 10,
c                  20 + neq   for mf = 21 or 22
c liw     = declared length of iwork (in user-s dimension).
c jac     = name of subroutine for jacobian matrix.
c mf      = method flag.  standard values are..
c             10 for nonstiff (adams) method, no jacobian used.
c             22 for stiff method, internally generated full jacobian.
c note that the main program must declare arrays y, rwork, iwork,
c and possibly atol.
c
c the output from the first call (or any call) is..
c      y  = array of computed values of y(t) vector.
c      t  = corresponding value of independent variable (normally tout).
c istate  = 2  if lsode was successful, negative otherwise.
c             -1 means excess work done on this call, perhaps wrong mf.
c             -2 means excess accuracy requested (tolerances too small).
c             -3 means illegal input detected (see printed message).
c             -4 means repeated error test failures (check all inputs).
c             -5 means repeated convergence failures, perhaps bad jac.
c                supplied or wrong choice of mf or tolerances).
c             -6 means error weight became zero during problem (solution
c                component i vanished, and atol or atol(i) = 0.)
```

Numerische Quadratur

$$\dot{x} = f(t) \rightarrow A \coloneqq x_1 - x_0 = \int_a^b f(t)\mathrm{d}t$$

Ordnung	numerische Integration	numerische Quadratur
1	explizites Euler-Verfahren	Rechtecksnäherung $\eta_{i+1} = \eta_i + hf(t_i)$ 
2	Heun-Verfahren	Trapezregel $\eta_{i+1} = \eta_i + \frac{h}{2}[f(t_i) + f(t_{i+1})]$ 
2	Collatz-Verfahren	Mittelpunktsregel $\eta_{i+1} = \eta_i + hf(t_{i+0.5})$ 
4	Runge-Kutta 4. Ordnung	Simpson-Regel $\eta_{i+1} = \eta_i + \frac{h}{6}[f(t_i) + 4f(t_{i+0.5}) + f(t_i + 1)]$
allgemein	Einschrittverfahren	Newton-Cotes Formeln