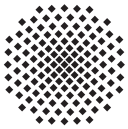
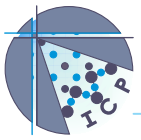


Universität Stuttgart



INSTITUTE FOR
COMPUTATIONAL
PHYSICS

Physik auf dem Computer

Mitschriften zur Vorlesung
Sommersemester 2017

Universität Stuttgart
Fakultät 8: Fachbereich Physik
B. Sc. Physik

10. Mai 2017

Dr. Jens Smiatek
Institut für Computerphysik
Universität Stuttgart
Allmandring 3
D-70569 Stuttgart
Germany

Email: smiatek@icp.uni-stuttgart.de

Inhaltsverzeichnis

Weiterführende Literatur

- Bronstein, Ilja N., et al. *Taschenbuch der Mathematik.* Eds. Eberhard Zeidler und Wolfgang Hackbusch. Springer-Verlag, Berlin 2012.
- Faires, J. Douglas und Burden, Richard L. *Numerische Methoden.* Spektrum Akademischer Verlag, Heidelberg 1994.
- Press, William H., et al. *Numerical Recipes.* Cambridge University Press, Cambridge 1989.

Allgemeine Hinweise zur Notation

Kurznotationen:

- $i = 1(1)n$ steht für $i = 1, 2, \ldots, n$
- $i = n(-a)1$ steht für $i = n, n - a, \ldots, 1 + a, 1$

1 Motivation: Physik auf dem Computer

1.1 Warum Physik auf dem Computer?

Numerische Methoden werden in vielen Bereichen der modernen Physik eingesetzt. Als Paradebeispiel hierfür gilt die neue *Dreieinigkeit der Physik (New Trinity of Physics)* (Abb. 1.1). Neben den klassischen Disziplinen Theorie und Experiment wird seit Mitte

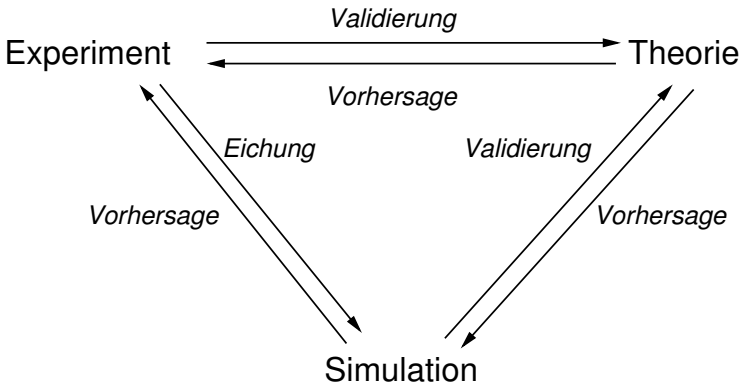


Abbildung 1.1: Die neue Dreieinigkeit der Physik mit den Säulen Experiment, Theorie und Simulation.

des letzten Jahrhunders auch der Simulationsansatz als dritte Säule in der modernen Physik angesehen. Theorien können nun durch Experiment **UND** Simulation verifiziert werden! Dies ist insbesondere wichtig für Wissenschaftstheorien wie dem Kritischen Rationalismus [1], welcher durch Karl R. Popper (1902 - 1994) begründet wurde und als wesentliche Kernaussage darauf abzielt, dass eine Theorie falsifizierbar sein muss, da sie ansonsten keine Theorie darstellt.

Einsatz von Rechnern und numerischen Methoden in der Theorie:

- numerische Lösungen von Gleichungen
- symbolische Mathematik (Mathematica©, ...)
- ...

Einsatz von Rechnern und numerischen Methoden im Experiment:

- Steuerung von Experimenten
- Auswertung von Daten
- ...

Fazit:
Computer und numerische Methoden werden in der modernen Physik in nahezu allen Bereichen eingesetzt.

1.2 Anwendung einer numerischen Methode zur Lösung der Bewegungsgleichung eines Fadenpendels (harmonischer Oszillator)

Im Folgendem soll nun ein einfaches Beispiel zur numerischen Lösung einer Bewegungsgleichung vorgestellt werden. Als Modellsystem dient das Fadenpendel (harmonischer

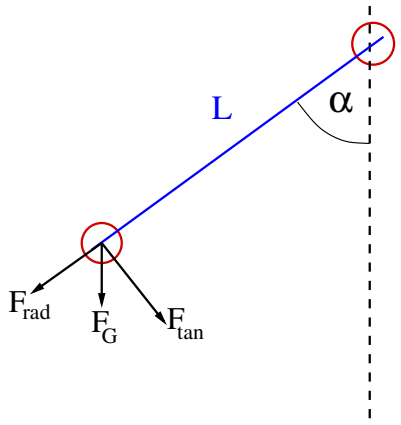


Abbildung 1.2: Schematische Darstellung eines Fadenependels.

Oszillator) in Abb. 1.2.

Annahmen:

- Steifer und masseloser Faden mit Längle L
- Kugel mit Masse m
- Es exisitert keine Reibung im System (keine Energiedissipation)

Vorherrschende Kräfte:

- radiale Komponente $\mathrm{F}_{\mathrm{rad}}$ (spielt keine Rolle für Bewegungsgleichung)
- Gravitation $\mathrm{F}_{\mathrm{G}} = m \cdot g$ mit Erdbeschleunigung $g = 9.81\ \mathrm{m/s^2}$
- tangentiale Kraft $\mathrm{F}_{\mathrm{tan}} = m \cdot g \sin\alpha = m \cdot a$

Mittels Newton lässt sich die Rückstellkraft berechnen,

$$\mathrm{F}_R = -\mathrm{F}_{\mathrm{tan}} = -m \cdot g \sin\alpha \tag{1.1}$$

welches

$$-\, m \cdot g \sin\alpha = m \cdot a \tag{1.2}$$

ergibt, wobei die Beschleunigung durch

$$a = -g \sin\alpha \tag{1.3}$$

gegeben ist. Die Tangentialbeschleunigung lässt sich auch mittels der Winkelbeschleunigung ausdrücken,

$$a = L\ddot{\alpha} \tag{1.4}$$

wodurch sich die allgemeine Bewegungsgleichung ergibt.

Allgemeine Bewegungsgleichung:

$$\ddot{\alpha}(t) = -\frac{g}{L} \sin\alpha(t) = -\omega^2 \sin\alpha(t) \tag{1.5}$$

1.2.1 Analytische Lösung mittels Näherung von kleinen Auslenkungen

Eine analytische Lösung für $\alpha(t)$ in Gleichung (1.5) kann für kleine Auslenkungen gefunden werden. Dann gilt $\sin\alpha \approx \alpha$ und Gleichung (1.5) vereinfacht sich zu

$$\ddot{\alpha}(t) = -\frac{g}{L}\alpha(t) \tag{1.6}$$

wobei eine Lösung von Gleichung (1.6) die folgende Grundform

$$\alpha(t) = A\cos\omega t + B\sin\omega t \tag{1.7}$$

mit Eigenfrequenz

$$\omega = \sqrt{\frac{g}{L}} \tag{1.8}$$

darstellt. Die Vorfaktoren A und B können nun durch

1. Ruhelage: Auslenkung $\alpha(t_0) = 0$ bei $t_0 = 0 \rightarrow A = \alpha(t_0)$
2. Geschwindigkeit: $\dot{\alpha}(t_0) = -\omega A \sin\omega t_0 + \omega B \cos\omega t_0 \rightarrow B = \frac{\dot{\alpha}(t_0)}{\omega}$

bestimmt werden. Nach Einsetzen von A und B in Gleichung (1.7) ergibt sich

$$\alpha(t) = \alpha(t_0)\cos\omega t + \frac{\dot{\alpha}(t_0)}{\omega}\sin\omega t \tag{1.9}$$

als allgemeine Lösung der Bewegungsgleichung (1.6).

1.2.2 Numerische Lösung mittels einer Simulation

Wie bereits angemerkt wurde, ist Gleichung (1.9) für kleine Auslenkungen gültig. Wie löst man aber die volle Bewegungsgleichung (1.5)? Hierfür ist keine einfache Näherung möglich und keine einfache analytische Lösung gegeben.
Als möglicher Ausweg kann eine **numerische Lösung** mittels einer Simulation gefunden werden.
Um die Simulation numerisch stabil zu halten, ist es sinnvoll, Parameterwerte für g und L innerhalb der gleichen Grössenordnung festzulegen. Da in obigem Beispiel die Erdbeschleunigung mit $g = 9.81\ \mathrm{m/s^2}$ festgelegt ist, kann $L = 1\ \mathrm{m}$ gewählt werden, so dass sich die Eigenfrequenz $\omega = \sqrt{g/L} \approx 3.13\ \mathrm{s^{-1}}$ ergibt. Dies sollte die numerische Stabilität der Simulation gewährleisten. Allerdings tritt noch ein weiterer wichtiger Punkt bei Simulationen auf.

Kernproblem von Simulationen:
Es existiert kein kontinuierlicher Zeitverlauf bei Simulationen und eine Diskretisierung der Zeit in diskrete Zeitintervalle ist unabdingbar.

Die Abfolge der Zeitintervalle ist durch

$$t_n = n \cdot \delta t \tag{1.10}$$

gegeben, wobei $n = 0, \ldots, N$ und δt die Breite des Zeitintervalls angibt. Das Ziel der Simulation ist nun, die allgemeine Bewegungsgleichung (1.5) numerisch zu lösen und als Funktion von $\alpha(t_n)$ darzustellen. Als erster Schritt für die Anwendung einer Simulation bedarf es einer willkürlichen, aber vernünftigen Wahl der Ausgangsposition und der Ausgangsgeschwindigkeit in Gleichung (1.5).
Hierzu werden

$$\dot{\alpha}(t_0) = v(t_0) \tag{1.11}$$

als Ausgangsgeschwindigkeit und $\alpha(t_0)$ als Ausgangsposition angenommen.
Als allgemeinen Ansatz wird eine Integration von Gleichung (1.5) durchgeführt, welche

$$\dot{\alpha}(t + \delta t) \equiv v(t_0 + \delta t) = v(t_0) + \int_{t_0}^{t_0+\delta t} [-\omega^2 \sin \alpha(\tau)]\, d\tau \tag{1.12}$$

ergibt. Die analytische Auswertung des Integrals ist jedoch in der Simulation nicht möglich, da nur die Stütz- und Auswertestellen bei ganzzahligen Vielfachen von δt bekannt sind oder berechnet werden können. Um dies zu ermöglichen und um die obige Gleichung numerisch handhabbar zu machen, wird folgende Näherung eingesetzt

$$f'(t)\, dt \approx f'(t) \lim_{\delta t \to 0} \delta t \tag{1.13}$$

welche zu

$$v(t_0 + \delta t) \approx v(t_0) - \omega^2 \sin \alpha(t_0)\, \delta t \tag{1.14}$$

führt, wobei $v(t_0 + \delta t)$ natürlich umso exakter ist, je kleiner der Wert für δt gewählt wurde. Allerdings gibt es bei physikalischen Simulationen auch eine untere Schranke, bei der die Gesetze der Quantenmechanik gelten und der Begriff von klassischen Trajektorien nicht mehr gültig ist.
Um die neue Position zu berechnen, wird dann nochmals "integriert", welches

$$\alpha(t_0 + \delta t) \approx \alpha(t_0) + v(t_0)\,\delta t \tag{1.15}$$

ergibt. Durch sukzessive und iterative Abfolge der Gleichungen (1.14) und (1.15) kann nun die Trajektorie des System näherungsweise berechnet werden.
Um die Qualität der numerischen Lösung einschätzen zu können, bedarf es der Überprüfung einer unabhängigen ud wohlbekannten Kenngrösse des Systems. Eine solche Kenngrösse stellt die Gesamtenergie dar, welche durch kinetische und potentielle Anteile festgelegt ist und welche per Definition konstant ist (s. Abschnitt 1.2).
Weitere Beispiele für den Einsatz von numerische Lösungen von Bewegungsgleichungen sind z. B. auch

- die Berechnung der Flugbahn eines Balls oder einer Rakete
- die Bewegung von Planeten im Sonnensystem
- die Kollision von harten Kugeln

Um die Bewegungsgleichungen noch realitätsnäher zu gestalten, können auch weitere Effekte wie Reibungskräfte in die Bewegungsgleichungen aufgenommen werden. Ebenso gibt es auch bessere Simulationsalgorithmen als die oben vorgestellte Variante. Ein vielfach in Simulationen eingesetzter Algorithmus ist der Velocity Verlet Algorithmus [3, 2] oder Varianten davon, welche folgende Grundabfolge aufzeigen:
1. Schritt: Berechnung der Geschwindigkeiten $v(t + \delta t/2)$

$$v(t + \delta t/2) = v(t) + \frac{\delta t}{2m} F(t) \tag{1.16}$$

2. Schritt: Berechnung der neuen Positionen $\alpha(t + \delta t)$

$$\alpha(t + \delta t) = \alpha(t) + \delta t\, v(t + \delta t/2) \tag{1.17}$$

3. Schritt: Berechnung der neuen Geschwindigkeiten $v(t + \delta t)$

$$v(t + \delta t) = v(t + \delta t/2) + \frac{\delta t}{2m} F(t + \delta t) \tag{1.18}$$

und nach Schritt 3 der Algorithmus wieder bei Schritt 1 startet.
Trotz der universellen Anwendbarkeit von numerischen Lösungen gibt es immer aber einen Punkt zu beachten.

2 Lineare Algebra: Lineare Gleichungssysteme

Eines der Grundlagen und der Hauptprobleme der numerischen Mathematik

- lineare Gleichungssysteme

Lineare Gleichungssysteme werden benutzt zum numerischen Lösen von Differentialgleichungen

- Rückführung auf Satz von linearen Gleichungssystemen

Durch die vorhandene Rechnerarchitektur ist es heutzutage möglich auch ein System von vielen Millionen Variablen zu lösen. In diesem Kapitel geht es hauptsächlich um grundlegende Methoden zum Lösen von linearen Gleichungssystemen wie z. B. die Gaußelimination, die LR- als auch die Choleskyzerlegung.

2.1 Lineare Gleichungssysteme

Bei vielen praktischen Aufgaben werden für n unbekannte Größen x_i (mit $i = 1, \ , 2, \ldots n$) m Bedingungen in Gleichungsform gestellt:

$$\begin{aligned} F_1(x_1, x_2, \ldots x_n) &= 0 \\ F_2(x_1, x_2, \ldots x_n) &= 0 \\ &\vdots \\ F_m(x_1, x_2, \ldots x_n) &= 0 \end{aligned} \tag{2.1}$$

Aufgabe:
Die Unbekannten x_i sind so zu bestimmen, dass sie eine Lösung des Gleichungssystems (2.1) darstellen.
Anmerkung:
In der Regel gilt $m = n$, d. h. die Anzahl der Unbekannten stimmt mit Anzahl der Gleichungen überein
Aber auch:

- $m > n$: überbestimmtes System, mehr Gleichungen als Unbekannte
- $m < n$: unterbestimmes System, weniger Gleichungen als Unbekannte

Überbestimmte Systeme:
In der Regel existiert keine Lösung!
Ersatz: Quadratmittelaufgabe:

$$\sum_{i=1}^{m} F_i^2(x_1, x_2, \dots x_n) = \min$$

Unterbestimmte Systeme:
$n - m$ Unbekannte können frei gewählt werden, d. h. die Lösung von Glg. (2.1) hängt von $n - m$ Parametern ab: $(n - m)$ - *dimensionale Lösungsmannigfaltigkeit*

Anmerkung: Im Gegensatz zu linearen Gleichungssystemen können in nicht-linearen Gleichungssystemen auch nicht-lineare Unbekannte auftreten ($\rightarrow$ Lösung typischerweise durch Iterationsverfahren)

2.1.1 Allgemeine Matrixschreibweise eines linearen Gleichungssystems

Gegeben sei das lineare Gleichungssystem für $m = n$

$$\begin{pmatrix} a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \dots & + & a_{2n}x_n & = & b_2 \\ \vdots & & \vdots & & \dots & & \vdots & & \vdots \\ a_{n1}x_1 & + & a_{n2}x_2 & + & \dots & + & a_{nn}x_n & = & b_n \end{pmatrix} \tag{2.2}$$

mit der Bedingung $A = (a_{ik}) \in \mathbb{R}^{n \times n}$ für $i, k = 1, 2, \dots, n$ oder

$$\overleftrightarrow{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \dots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$$

mit $b \in \mathbb{R}^n$ und $x \in \mathbb{R}^n$. Das obige Gleichungssystem (Glg. (2.2)) kann auch in der kompakten Form

Allgemeine Matrizengleichung

$$\overleftrightarrow{A} \cdot \vec{x} = \vec{b} \tag{2.3}$$

oder schlicht $A \cdot x = b$ geschrieben werden. Im Allgemeinen wird nach den Lösungen $x \in \mathbb{R}^n$ gesucht, wobei nicht klar ist, ob es eine Lösung gibt oder ob diese eindeutig ist.

2.2 Dreiecksmatrizen

Im nun folgenden Abschnitt beschäftigen wir uns mit Dreiecksmatrizen. Unter einer Dreiecksmatrix versteht man eine quadratische Matrix ($m = n$), welche sich dadurch

auszeichnet, dass alle Eintärge unterhalb (obere Dreiecksmatrix) bzw. oberhalb (untere Dreiecksmatrix) der Hauptdiagonalen null sind.

Rechte obere Dreiecksmatrix R (im Englischen U)
Quadratische Matrix $A \in \mathbb{R}^{n \times n}$ mit $a_{ij} = 0$ für $i > j$.

Linke untere Dreiecksmatrix L (im Englischen L)
Quadratische Matrix $A \in \mathbb{R}^{n \times n}$ mit $a_{ij} = 0$ für $i < j$.

Dreiecksmatrizen werden zum Lösen von linearen Gleichungssystemen benötigt, wie wir in den späteren Abschnitten noch sehen werden.

2.3 Dreieckszerlegung einer Matrix

Durch elementare Umformungen wie dem Vertauschen von Zeilen oder Spalten, der Multiplikation einer Zeile mit einer von 0 verschiedenen Zahl oder der Addition eines Vielfachen einer Zeile zu einer anderen Zeile kann das System $\overleftrightarrow{A} \cdot \vec{x} = \vec{b}$ in ein gestaffeltes Gleichungssystem $\overleftrightarrow{R} \cdot \vec{x} = \vec{c}$ mit

$$\overleftrightarrow{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & \dots & r_{1n} \\ & r_{22} & r_{23} & \dots & r_{2n} \\ & & r_{33} & & r_{3n} \\ & \varnothing & & \ddots & \vdots \\ & & & & r_{nn} \end{pmatrix}$$

überführt werden, welches eine Lösung besitzt, falls das Gleichungssystem (2.2) **regulär** mittels

$$\det A \neq 0 \tag{2.4}$$

ist. Die Lösung kann dann eindeutig durch **Rückwärtssubstitution**

Rückwärtssubstitutionsgleichung

$$x_i = \frac{1}{r_{ii}} \left(c_i - \sum_{k=i+1}^{n} r_{ik} x_k \right) \tag{2.5}$$

mit $i = n-1, n-2, \dots, 1$ und $x_n = \frac{c_n}{r_{nn}}$

bestimmt werden.

Für reguläre linke untere Dreiecksmatrizen kann die **Vorwärtssubstitution** angewendet werden. Entsprechend $\overleftrightarrow{A} \cdot \vec{x} = \vec{b} \rightarrow \overleftrightarrow{L} \cdot \vec{x} = \vec{c}$ mit

$$\overleftrightarrow{L} = \begin{pmatrix} \ell_{11} & & & & \\ \ell_{21} & \ell_{22} & & \varnothing & \\ \ell_{31} & l_{32} & \ell_{33} & & \\ \vdots & \vdots & & \ddots & \\ \ell_{n1} & l_{n2} & \dots & \ell_{n,n-1} & \ell_{nn} \end{pmatrix}$$

gilt

Vorwärtssubstitutionsgleichung

$$x_i = \frac{1}{\ell_{ii}} \left(c_i - \sum_{k=1}^{i-1} \ell_{ik} x_k \right) \tag{2.6}$$

mit $i = 2, 3, \dots, n$ und $x_1 = \frac{c_1}{\ell_{11}}$

Des Weiteren gilt für Diagonalmatrizen

Diagonalmatrizensubstitution

$$x_i = \frac{c_i}{r_{ii}} \text{ bzw. } x_i = \frac{c_i}{\ell_{ii}} \text{ mit } i = 1, 2, \dots, n$$

Löseroutine in Python (SciPy):

- scipy.linalg.solve_triangular (...)

Frage: Wie gelingt nun aber die Transformation von $\overleftrightarrow{A}$ zu $\overleftrightarrow{R}$?

2.3.1 Prinzip des Gaußschen Eliminationsverfahrens

Das Gaußsche Eliminationsverfahren ist ein Verfahren, um den Übergang von $\overleftrightarrow{A}$ zu $\overleftrightarrow{R}$ in $n-1$ Eliminationsschritten durchzuführen.
Hauptprinzip:
Mithilfe einer Gleichung soll eine Unbekannte aus den restlichen Gleichungen entfernt werden $\rightarrow m-1$ Gleichungen mit $n-1$ Unbekannten (1. Schritt) usw.
Elementaroperationen:
Für das System (Glg. (2.3)) $\overleftrightarrow{A} \cdot \vec{x} = \vec{b}$ (im Folgenden als Koeffizientenmatrix $(A|b)$ dargestellt), können folgende Operationen vorgenommen werden:

Elementaroperationen

1. Vertauschung zweier Gleichungen (Zeilentausch in $(A|b)$)
2. Vertauschung zweier Spalten in $\vec{x}$ und $\overleftrightarrow{A}$ (Variablentausch)
3. Addition eines Vielfachen einer Zeile zu einer anderen Zeile
4. Multiplikation einer Zeile mit einer Konstanten $d \neq 0$

Ziel der Elementaroperationen:
Bringe Matrix $(A|b)$

$$\left(\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \vdots & \vdots & & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} & b_n \end{array}\right)$$

spaltenweise auf Dreiecksform.
1. Schritt: Es sei $a_{11} \neq 0$ (Pivotelement, vgl. Abschnitt 2.3.3), ansonsten Vertauschung der Gleichungen
2. Schritt: Überführung der Matrix $\overleftrightarrow{A}$ nach Matrix $\overline{A}$; Vertausche die 1. und die r-te Zeile von $\overleftrightarrow{A}$
3. Schritt: Überführung der Matrix $\overline{A}$ nach $\overleftrightarrow{A}_1$ $((\overline{A}|b) \to (A_1|b_1)$; Subtrahiere für $i = 2, 3, \dots n$ das ℓ_{i1}-fache der 1. Zeile von der i-ten Zeile der Matrix $\overline{A}$

Ergebnis:

$$\overleftrightarrow{A}_1 = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & a_{1n}^{(1)} \\ 0 & a_{22}^{(1)} & \dots & a_{2n}^{(1)} \\ \vdots & a_{32}^{(1)} & \dots & a_{3n}^{(1)} \\ \vdots & \vdots & & \\ 0 & a_{n2}^{(1)} & \dots & a_{nn}^{(1)} \end{pmatrix} \qquad \vec{b}_1 = \begin{pmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \\ \vdots \\ b_n^{(1)} \end{pmatrix}$$

Farbige Teilmatrix ist vom Typ $(n-1,\ n-1)$ und wird analog zu $\overleftrightarrow{A}$ im nächsten Schritt zu $\overleftrightarrow{A}_2$ überführt.
Die Elemente in $\overleftrightarrow{A}_1$ und $\vec{b}_1$ lassen sich mit:

$$a_{ik}^{(1)} = \overline{a}_{ik} - \ell_{i1} \cdot \overline{a}_{1k} \text{ mit } \ell_{i1} = \frac{\overline{a}_{i1}}{\overline{a}_{11}}$$

$$b_i^{(1)} = \overline{b}_i - \ell_{i1} \cdot \overline{b}_1$$

für $i, k = 2, 3, \dots n$ berechnen.
Durch sukzessives Wiederholen kann schlüssendlich die gewünschte Form $\overleftrightarrow{R} \cdot \vec{x} = \vec{c}$ ermittelt werden, d. h. das Verfahren wird beendet, wenn alle $a_{ik}^{(r-1)} = 0$ für $i, k \geq r$ sind. Dann gilt $(A_{r-1}|b_{r-1}) = (R|c)$.

Lösungsverhalten:
1. Fall: Das System ist unlösbar für $a_{ik}^{(r-1)} \neq 0$ für $i, k \geq r$.
2. Fall: Das System ist lösbar für $a_{ik}^{(r-1)} = 0$ für $i, k \geq r$.
Im Speziellen gilt dann:

i) $r = n$: Die Lösung ist eindeutig.

ii) $r < n$: Die Lösung ist nicht eindeutig, $n - r$ Unbekannte sind frei wählbar.

2.3.2 Ein Beispiel zur Gauß-Elimination

Gegeben sei folgendes Gleichungssystem

$$\begin{aligned} 5x_1 - x_2 + 2x_3 &= 3 \\ 7x_2 + x_3 &= 4 \\ 10x_1 + x_2 + x_3 &= 1 \end{aligned}$$

welches folgende Koeffizientenmatrix liefert:

$$\left(\begin{array}{ccc|c} 5 & -1 & 2 & 3 \\ 0 & 7 & 1 & 4 \\ 10 & 1 & 1 & 1 \end{array}\right) \begin{matrix} \mathrm{E}_1 \\ \mathrm{E}_2 \\ \mathrm{E}_3 \end{matrix}$$

mit dem kanonischen Pivotelement $a_{11} = 5$ (vgl. Abschnitt 2.3.3). Die jeweiligen E_1, E_2 und E_3 kennzeichnen die jeweiligen Zeilen in der Koeffizientenmatrix, wobei die veränderten Zeilen nach p Elementaroperationen mit $\mathrm{E}_i^{(p)}$ für $i = 1, 2, 3$ notiert werden.

1. Schritt: Vertausche E_3 mit E_2:

$$\begin{pmatrix} \mathrm{E}_1 \\ \mathrm{E}_2 \\ \mathrm{E}_3 \end{pmatrix} \rightarrow \begin{pmatrix} \mathrm{E}_1 \\ \mathrm{E}_3 \\ \mathrm{E}_2 \end{pmatrix} \equiv \begin{pmatrix} \mathrm{E}'_1 \\ \mathrm{E}'_2 \\ \mathrm{E}'_3 \end{pmatrix}$$

2. Schritt: Multipliziere E'_1 mit (-2) und addiere zu E'_2:

$$\left(\begin{array}{ccc|c} 5 & -1 & 2 & 3 \\ 0 & 3 & -3 & -5 \\ 0 & 7 & 1 & 4 \end{array}\right) \begin{matrix} \mathrm{E}''_1 \\ \mathrm{E}''_2 \\ \mathrm{E}''_3 \end{matrix}$$

3. Schritt: Multipliziere E_2'' mit $\left(-\frac{7}{3}\right)$ und addiere zu E_3'':

$$\left(\begin{array}{ccc|c} 5 & -1 & 2 & 3 \\ 0 & 3 & -3 & -5 \\ 0 & 0 & 8 & \frac{47}{3} \end{array}\right)$$

Obige Matrix hat die gewünschte rechte obere Dreiecksform und kann nun eindeutig durch Rückwärtssubstitution (Glg. (2.5)) oder durch Einsetzen der Werte gelöst werden.

Lösung durch Einsetzen:

i) $8x_3 = \frac{47}{3} \rightarrow x_3 = \frac{47}{24}$

ii) $3x_2 - 3 \cdot \frac{47}{24} = -5 \rightarrow x_2 = \frac{7}{24}$

iii) $5x_1 - \frac{7}{24} + 2 \cdot \frac{47}{24} = 3 \rightarrow x_1 = -\frac{1}{8}$

Lösung durch Rückwartssubstitution (Glg. (2.5)):

i) $x_3 = \frac{\frac{47}{3}}{8}$ mit $i = n = 3$.

ii) $x_2 = \frac{1}{r_{22}}(c_2 - r_{23} \cdot x_3) = \frac{1}{3}\left(-5 + 3 \cdot \frac{47}{24}\right) = \frac{7}{24}$ mit $i = n - 1$.

iii) $x_1 = \ldots$

2.3.3 Zur Bestimmung des Pivotelements

Das bereits erwähnte Element a_{r1} heißt Pivotelement (franz./engl. "Drehpunkt"), da es den
Dreh- und Angelpunkt des iterativen Verfahrens der Gaussös-Elimination darstellt.
$\Rightarrow$ Numerisch günstig für Wahl des Pivotelements: $a_{r1} \gg 1$

Wahlmöglichkeiten für Pivotelement

1. **Kanonische Pivotwahl:** keine Vertauschungen!
   Einsatz nur bei Sicherstellung $a_{r1} \gg 1$
   Anmerkung: Verfahren scheitert schon bei $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
2. **Spaltenpivotwahl:** Pivotelement ist das betragsmäßig größte Element in der Spalte
   (Vertauschung der Zeilen, aber Variablenreihenfolge bleibt unverändert.
3. **Totalpivotwahl:** Bestimme als Pivotelement das betragsgrößte Element in der Restmatrix
   (Spalten- und Zeilentausch nötig!)
   Abfolge:
   a) Vertauschung der Zeilen
   b) Vertauschung der Spalten

Anmerkungen zur Gauß-Elimination:
Benötigte Anzahl von Rechenoperationen bei Gauß-Elimination: $O(n^3)$
In Python: Keine Gauß-Elimination (zu langsam)
Stattdessen in Python für eine eindeutig lösbare Matrix: scipy.linalg.solve()
Basiert auf LR-Zerlegung (vgl. Abschnitt 2.3.6)

2.3.4 Matrixinversion

Es sei $\overleftrightarrow{A} \in \mathbb{R}^{n \times n}$ regulär , d.h. eine quadratische Matrix mit existierender Inverser $\overleftrightarrow{A}^{-1}$.
Es gilt weiterhin das algemeine Gleichungssystem (Glg. (2.3)) $\overleftrightarrow{A} \cdot \vec{x} = \vec{b}$.
Man kann dann zeigen, dass zu einer regulären Matrix $\overleftrightarrow{A}$ immer eine inverse Matrix $\overleftrightarrow{A}^{-1}$ existiert mit

$$\overleftrightarrow{A} \cdot \overleftrightarrow{A}^{-1} = \overleftrightarrow{A}^{-1} \cdot \overleftrightarrow{A} = \overleftrightarrow{E} \rightarrow \vec{x} = \frac{\vec{b}}{\overleftrightarrow{A}} = \overleftrightarrow{A}^{-1} \cdot \vec{b}$$

wobei $\overleftrightarrow{E}$ die Einheitsmatrix ist.
Anmerkung zum Abschnitt 2.3.2:
Rücksubstitutionsmethode (Glg. (2.3.1)) liefert implizit die Inverse von $\overleftrightarrow{A}$

Weitere Möglichkeit zur Lösung des Gleichungssystems (Glg. (2.3)**):**
Berechnung der inversen Matrix durch Gauß-Elimination
Dazu Transformation: $A|B \rightarrow E|A^{-1}$ mit n × n Einheitsmatrix E

Ablauf der Matrizeninversion

1. Gauß-Elimination mit Spaltenpivotwahl
   Elimination der Elemente unter- **und** oberhalb der Diagonalen
   (Erzeugung der Diagonalmatrix $\overleftrightarrow{A} \rightarrow \overleftrightarrow{D}$)
2. Multiplikation der Pivotzeile mit $\frac{1}{a_{ii}}$ für $\overleftrightarrow{D} \rightarrow \overleftrightarrow{E}$
   (Bedingung für Diagonalelemente)

2.3.5 Beispiel zur Matrizeninversion

Aufgabe:
Berechnung der Inversen von

$$\overleftrightarrow{A} = \begin{pmatrix} 3 & 5 & 1 \\ 2 & 4 & 5 \\ 1 & 2 & 2 \end{pmatrix} \begin{matrix} \mathrm{E}_1 \\ \mathrm{E}_2 \\ \mathrm{E}_3 \end{matrix}$$

unter Berücksichtigung der Benutzung der Notation von Abschnitt 2.3.2.
Genereller Ablauf:

1. Vertauschung von E_1 und E_3:

$$\left(\begin{array}{ccc|ccc} 3 & 5 & 1 & 1 & 0 & 0 \\ 2 & 4 & 5 & 0 & 1 & 0 \\ 1 & 2 & 2 & 0 & 0 & 1 \end{array}\right) \rightarrow \left(\begin{array}{ccc|ccc} 1 & 2 & 2 & 0 & 0 & 1 \\ 2 & 4 & 5 & 0 & 1 & 0 \\ 3 & 5 & 1 & 1 & 0 & 0 \end{array}\right) \begin{matrix} \mathrm{E}'_1 \\ \mathrm{E}'_2 \\ \mathrm{E}'_3 \end{matrix}$$

2. $\mathrm{E}'_2 - (\mathrm{E}'_1 \cdot 2)$:

$$\left(\begin{array}{ccc|ccc} 1 & 2 & 2 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & -2 \\ 3 & 5 & 1 & 1 & 0 & 0 \end{array}\right)$$

3. $\mathrm{E}''_3 - (\mathrm{E}''_1 \cdot 3)$:

$$\left(\begin{array}{ccc|ccc} 1 & 2 & 2 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & -2 \\ 0 & -1 & -5 & 1 & 0 & -3 \end{array}\right)$$

4. $\mathrm{E}_2''' \leftrightarrow (\mathrm{E}_3''' \cdot (-1))$:

$$\left(\begin{array}{ccc|ccc} 1 & 2 & 2 & 0 & 0 & 1 \\ 0 & 1 & 5 & -1 & 0 & 3 \\ 0 & 0 & 1 & 0 & 1 & -2 \end{array}\right)$$

5. $\mathrm{E}_1'''' - (\mathrm{E}_2'''' \cdot 2)$:

$$\left(\begin{array}{ccc|ccc} 1 & 0 & -8 & 2 & 0 & -5 \\ 0 & 1 & 5 & -1 & 0 & 3 \\ 0 & 0 & 1 & 0 & 1 & -2 \end{array}\right)$$

6. $\mathrm{E}_1^{\mathrm{V}} + (\mathrm{E}_3^{\mathrm{V}} \cdot 8)$:

$$\left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 2 & 8 & -21 \\ 0 & 1 & 5 & -1 & 0 & 3 \\ 0 & 0 & 1 & 0 & 1 & -2 \end{array}\right)$$

7. $\mathrm{E}_2^{\mathrm{VI}} - (\mathrm{E}_3^{\mathrm{VI}} \cdot 5)$:

$$\underbrace{\left(\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right.}_{\mathrm{E}} \underbrace{\left.\begin{array}{ccc} 2 & 8 & -21 \\ -1 & -5 & 13 \\ 0 & 1 & -2 \end{array}\right)}_{\mathrm{A}^{-1}}$$

Aber:

- Verfahren ist numerisch nicht sehr günstig
  - Skalierung wie bei der Gauß-Elimination: $O(n^3)$ Rechenoperationen
- Verfahren ist häufig numerisch nicht stabil

In Python: Lösungsroutine scipy.linalg.inv()

2.3.6 LR-Zerlegung

Wie schon erwähnt:
"Python"benutzt die LR-Zerlegung, anstatt der Gauß-Elimination um ein Gleichungssystem (Glg. (2.3)) zu lösen, da die LR-Zerlegung numerisch effizienter ist.
Anmerkung:
Im Englischen wird die LR-Zerlegung auch als LU-Decomposition bezeichnet.
Die grundlegende Idee der LR-Zerlegung basiert auf der **Matrixfaktorisierung**.

Matrixfaktorisierung in der LR-Zerlegung
Die quadratische Matrix $A \in \mathbb{R}^{n \times n}$ wird in die linke untere Dreiecksmatrix L und in die rechte obere Dreiecksmatrix R zerlegt:

$$\overleftrightarrow{A} = \overleftrightarrow{L} \cdot \overleftrightarrow{R}$$

Nebenbedingung:
Definition der Diagonalelemente $\ell_{ii} = 1$ für $i = 1, 2, \ldots n$
(Zuordnung zur L-Matrix)

Durch Anwendung der Dreiecksform:
Lösung des Gleichungssystems (Glg. (2.3)) mittels

$$\overleftrightarrow{A} \cdot \vec{x} = \overleftrightarrow{L} \cdot \overleftrightarrow{R} \cdot \vec{x} = \vec{b}$$

durch Vorwärts (L)- und der Rückwärtssubstitution (R) entsprechend den Gleichungen (2.6) und (2.5)!

Beweis der Äquivalenz der LR-Zerlegung zum Ausgangssystem:
Das Gleichungssystem

$$\overleftrightarrow{A} \cdot \vec{x} = \vec{b} \tag{2.7}$$

kann mittels

$$\overleftrightarrow{A} = \overleftrightarrow{L} \cdot \overleftrightarrow{R} \tag{2.8}$$

mit

$$\overleftrightarrow{L} \cdot \vec{y} = \vec{b} \text{ und } \overleftrightarrow{R} \cdot \vec{x} = \vec{y}. \tag{2.9}$$

dargestellt werden. Einsetzen von Gleichung (2.8) in obige allgemeine Matrizengleichung (vgl. Glg. (2.3)) ergibt

$$\overleftrightarrow{A} \cdot \vec{x} = (\overleftrightarrow{L} \cdot \overleftrightarrow{R}) \cdot \vec{x} = \overleftrightarrow{L} \cdot (\overleftrightarrow{R} \cdot \vec{x}) = \overleftrightarrow{L} \cdot (\vec{y}) = \vec{b}$$

welches genau mit den obigen Forderungen übereinstimmt.

Lösung eines Gleichungssystems mittels LR-Zerlegung

1. Berechnung des Hilfsvektors $\vec{y}$ durch Vorwärtssubstitution (Glg.(2.6)):

$$\overleftrightarrow{L} \cdot \vec{y} = \vec{b} \tag{2.10}$$

2. Berechnung der Lösung $\vec{x}$ durch Rückwärtssubstitution (Glg. (2.5)):

$$\overleftrightarrow{R} \cdot \vec{x} = \vec{y} \tag{2.11}$$

Ein Beispiel für eine Matrixfaktorisierung mittels LR-Zerlegung:

$$\overleftrightarrow{A} = \begin{pmatrix} 3 & 1 & 6 \\ 2 & 1 & 3 \\ 1 & 1 & 1 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 1/3 & 1 & 0 \\ 2/3 & 1/2 & 1 \end{pmatrix}}_{\text{L}} \cdot \underbrace{\begin{pmatrix} 3 & 1 & 6 \\ 0 & 2/3 & -1 \\ 0 & 0 & -1/2 \end{pmatrix}}_{\text{R}} \tag{2.12}$$

Die Bestimmung der einzelnen Koeffizienten in den Dreicksmatrizen erfolgt nach:

$$\ell_{11} \cdot r_{11} = a_{11}$$

$$r_{1i} = \frac{a_{1i}}{\ell_{11}} \quad \text{und} \quad \ell_{i1} = \frac{a_{i1}}{r_{11}} \quad \text{für } i = 2, 3, \ldots n$$

$$\ell_{ii} \cdot r_{ii} = a_{ii} - \sum_{k=1}^{i-1} \ell_{ik} \cdot r_{ki} \quad \text{für } i = 2, 3, \ldots n-1$$

$$r_{ij} = \frac{1}{l_{ii}} \left(a_{ij} \sum_{k=1}^{i-1} l_{ik} \cdot u_{kj} \right) \quad \text{und} \quad \ell_{ji} = \frac{1}{r_{ii}} \left(a_{ji} \sum_{k=1}^{i-1} \ell_{jk} \cdot r_{ki} \right) \quad \text{für alle } j = i+1, i+2, \ldots n$$

$$\ell_{nn} \cdot r_{nn} = a_{nn} - \sum_{k=1}^{n-1} \ell_{nk} \cdot r_{kn}$$

Form der L und R - Matrizen:

$$\overleftrightarrow{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & \ldots & r_{1n} \\ & r_{22} & r_{23} & \ldots & r_{2n} \\ & & r_{33} & & r_{3n} \\ & \varnothing & & \ddots & \vdots \\ & & & & r_{nn} \end{pmatrix}, \quad \overleftrightarrow{L} = \begin{pmatrix} 1 & & & & \\ \ell_{21} & 1 & & \varnothing & \\ \ell_{31} & \ell_{32} & 1 & & \\ \vdots & \vdots & & \ddots & \\ \ell_{n1} & \ell_{n2} & \ldots & \ell_{n,n-1} & 1 \end{pmatrix}$$

Anwendung der LR-Zerlegung

1. Durchführung der Dreieckszerlegung: $\overleftrightarrow{A} = \overleftrightarrow{L} \cdot \overleftrightarrow{R}$
2. $\overleftrightarrow{L} \cdot \vec{y} = \vec{b}$ (Bestimmung des Hilfsvektors $\vec{y}$ durch Vorwärtssubstitution (Glg. (2.6)))
3. $\overleftrightarrow{R} \cdot \vec{x} = \vec{y}$ (Bestimmung der Lösung $\vec{x}$ durch Rückwärtssubstitution (Glg. (2.5)))

Rechenaufwand:
$O(n^2)$ Rechenoperationen, falls $\overleftrightarrow{L}$ und $\overleftrightarrow{R}$ bekannt sind.
Daher: Bei gegebener Matrix und vorhandener Faktorisierung werden $O(n^2)$ Rechenoperationen benötigt, um Gleichungssysteme zu lösen, welche Matrix $\overleftrightarrow{A} = \overleftrightarrow{L} \cdot \overleftrightarrow{R}$ beinhalten.

Amerkungen:

- Keine LR-Zerlegung für einfache Matrizen (z. B. 2×2 Matrizen)
- Falls Zeilentausch für Bestimmung von $\overleftrightarrow{L}$ und $\overleftrightarrow{R}$ notwendig ist
  - Einführung der Permutationsmatrix $\overleftrightarrow{P}$

2.3.7 Permutationsmatrix

Die Permutationsmatrix kann eingesetzt werden, um den expliziten Zeilentausch bei der Gauß-Elimination im Rahmen der LU-Zerlegung zu vermeiden.

Definition der Permutationsmatrix

Eine Permutationsmatrix $\overleftrightarrow{P}$ $(n \times n)$ ist eine Matrix mit genau einem Element mit dem Wert 1 in jeder Spalte und jeder Zeile, alle anderen Elemente sind null.

Beispiel:

$$\overleftrightarrow{P} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Beispielhafter Einsatz der Permutationsmatrix:

i) $\overleftrightarrow{P} \cdot \overleftrightarrow{A} = \overleftrightarrow{L} \cdot \overleftrightarrow{R}$

ii) $\overleftrightarrow{L} \cdot \vec{y} = \overleftrightarrow{P} \cdot \vec{b}$

iii) $\overleftrightarrow{R} \cdot \vec{x} = \vec{y}$

2.3.8 Diagonal dominante und positiv definite Matrizen

Eine $n \times n$ Matrix ist diagonal-dominant, wenn

Definition von diagonal dominanten Matrizen

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}| \text{ für alle } i = 1, 2, \ldots n$$

gilt.

Vorteile von diagonal-dominanten Matrizen:

- $\overleftrightarrow{A}$ ist nicht-singulär (det A $\neq 0 \rightarrow$ eindeutige Lösbarkeit des Gleichungssystems (z. B. zur Interpolation von Funktionen))
- Anwendung der Gauß-Elimination auf jedes Linearsystem (Glg. (2.3)) $\overleftrightarrow{A} \cdot \vec{x} = \vec{b}$ ohne Zeilen- und Spaltentausch

Definition von positiv definiten Matrizen

Eine Matrix $\overleftrightarrow{A}$ ist positiv definit, falls sie symmetrisch ist ($a_{ij} = a_{ji}$) und falls $\vec{x}^{\mathrm{T}} \cdot \overleftrightarrow{A} \cdot \vec{x} > 0$ für jede n-dimensionale Spaltenmatrix $\vec{x} \neq 0$ gilt.

Anmerkung zur Notation:
Das Superscript T kennzeichnet die Transponierte eines Vektors oder einer Matrix, d. h. eine Matrix, in der die Koeffizienten entlang der Hauptdiagonalen der Ursprungsmatrix gespiegelt sind.

Eine positiv definite Matrixäquivalenz zu $\overleftrightarrow{A}$ mit den Vorteilen von diagonal dominanten Matrizen kann erzeugt werden durch:

- Faktorisierung von $\overleftrightarrow{A}$ in $\overleftrightarrow{L} \cdot \overleftrightarrow{L}^{\mathrm{T}}$, wobei $\overleftrightarrow{L}$ eine untere Dreiecksmatrix mit positiven Diagonalelementen darstellt ($\overleftrightarrow{A}$ muss symetrisch sein $\Rightarrow$ Nutzung des Cholesky-Verfahren (Abschnitt 2.3.9))
- Faktorisierung von $\overleftrightarrow{A}$ in $\overleftrightarrow{L} \cdot \overleftrightarrow{D} \cdot \overleftrightarrow{L}^{\mathrm{T}}$, wobei $\overleftrightarrow{L}$ eine untere Dreiecksmatrix mit Einsen auf ihrer Diagonalen und $\overleftrightarrow{D}$ eine Diagonalmatrix mit positiven Diagonalelementen darstellt

Falls $\overleftrightarrow{L}^{\mathrm{T}} = \overleftrightarrow{R}$: $\quad \overleftrightarrow{A} = \overleftrightarrow{L} \cdot \overleftrightarrow{D} \cdot \overleftrightarrow{L}^{\mathrm{T}} = \overleftrightarrow{L} \cdot \overleftrightarrow{D} \cdot \overleftrightarrow{R}$ (LDR-Zerlegung)

LR-Zerlegung in Python:
`scipy.linalg.lu()` oder `scipy.linalg.lu_solve()` (Lösen des linearen Gleichungssystems)

2.3.9 Cholesky-Zerlegung

Bedingung für Cholesky-Zerlegung:
Matrix $\overleftrightarrow{A}$ ist symmetrisch und positiv definit, d.h. für die zugehörige quadratische Form $\overleftrightarrow{Q}(\vec{x})$ gilt:

$$\overleftrightarrow{Q}(\vec{x}) = \vec{x}^{\mathrm{T}} \cdot \overleftrightarrow{A} \cdot \vec{x} = \sum_{i=1}^{n} \sum_{k=1}^{n} a_{ik} x_i x_k > 0 \text{ für alle } \vec{x} \in \mathbb{R},\ \vec{x} \neq 0$$

(Anm.: Erzeugung der transponierten Matrix: Spiegelung an der Hauptdiagonalen).

Zu jeder symmetrischen und positiv definiten Matrix $\overleftrightarrow{A}$ (vergleiche auch Abschnitt 2.3.8) existiert eine eindeutige Dreieckszerlegung $\overleftrightarrow{A} = \overleftrightarrow{L} \cdot \overleftrightarrow{L}^{\mathrm{T}}$ mit unterer linker Dreiecksmatrix $\overleftrightarrow{L}$:

$$\overleftrightarrow{L} = \begin{pmatrix} \ell_{11} & & & & \\ \ell_{21} & \ell_{22} & & \varnothing & \\ \ell_{31} & \ell_{32} & \ell_{33} & & \\ \vdots & \vdots & & \ddots & \\ \ell_{n1} & \ell_{n2} & \dots & \dots & \ell_{nn} \end{pmatrix}$$

mit $\ell_{11} = \sqrt{a_{11}}$

$$\ell_{j1} = \frac{a_{j1}}{\ell_{11}} \text{ für alle } j = 2, 3, \dots n$$

$$\ell_{ii} = \left(a_{ii} - \sum_{k=1}^{i-1} \ell_{ik}^2 \right)^{\frac{1}{2}} \text{ für alle } i = 2, 3, \dots n-1$$

$$\ell_{ji} = \frac{1}{\ell_{ii}} \left(a_{ji} - \sum_{k=1}^{i-1} \ell_{jk} \ell_{ik} \right) \text{ für alle } j = i+1, i+2, \dots n$$

$$\ell_{nn} = \left(a_{nn} - \sum_{k=1}^{n-1} \ell_{nk}^2 \right)^{\frac{1}{2}}$$

Ablauf des Cholesky-Verfahrens

1. $\overleftrightarrow{A} = \overleftrightarrow{L} \cdot \overleftrightarrow{L}^{\mathrm{T}}$ (Ermittlung der Cholesky-Zerlegung und Substitution $\overleftrightarrow{L}^{\mathrm{T}} \cdot \vec{x} = \vec{c}$ für Lösung $\overleftrightarrow{A} \cdot \vec{x} = \vec{b}$)
2. $\overleftrightarrow{L} \cdot \vec{c} = \vec{b}$ (Bestimmung des Hilfsvektors $\vec{c}$ durch Vorwärtssubstitution (Glg. (2.6)))
3. $\overleftrightarrow{L}^{\mathrm{T}} \cdot \vec{x} = \vec{c}$ (Bestimmung der Lösung $\vec{x}$ durch Rückwärtssubstitution (Glg. (2.5))

Anmerkung:
Für große Werte von n ist der Aufwand beim Cholesky-Verfahren etwa halb so groß wie bei der LR-Zerlegung.

2.4 Zusammenfassung der wichtigsten Punkte des Kapitels

Kurzzusammenfassung: Lösung von linearen Gleichungssystemen

- Hauptproblem: Lösung von $\overleftrightarrow{A} \cdot \vec{x} = \vec{b}$ (Bestimmung von $\vec{x}$)
- Standardverfahren: Gauß-Elimination (bringt Matrix $\overleftrightarrow{A}$ in $\overleftrightarrow{R}$ Form)
- Nachteile: langsam $O(n^3)$, numerisch instabil, keine Garantie für Lösung
- Ausweg:
  1. Matrixinversion
  2. LR-Zerlegung ($\overleftrightarrow{A} = \overleftrightarrow{L} \cdot \overleftrightarrow{R}$)
     Berechnung von $\vec{x}$ durch Vorwärts- und Rückwärtssubstitution nach Anwendung der Matrizen $\overleftrightarrow{L}$ und $\overleftrightarrow{R}$
  3. Cholesky-Verfahren (analog zur LR-Zerlegung, hier aber $\overleftrightarrow{A} = \overleftrightarrow{L} \cdot \overleftrightarrow{L}^{\mathrm{T}}$)
     Falls $\overleftrightarrow{L}$ ermittelt wurde, einfache Berechnung von $\overleftrightarrow{L}^{\mathrm{T}}$
  4. Orthogonalisierungsverfahren, z.B. Householder-Verfahren (nicht besprochen)
     $\overleftrightarrow{A} = \overleftrightarrow{Q} \cdot \overleftrightarrow{R}$ mit $\overleftrightarrow{Q} \in \mathbb{R}^{\mathrm{n}\times\mathrm{n}}$ und $\overleftrightarrow{R} \in \mathbb{R}^{\mathrm{n}\times\mathrm{n}}$

3 Analysis: Darstellung von Funktionen

Ausgangsfrage:
Wie können Computer Funktionen wie z.B. die Sinusfunktion darstellen oder berechnen?
→ Prozessoren können nur Grundrechenarten
Lösung:
Darstellung von Funktionen durch stückweise definierte Polynome!
Idee der Interpolation:
Rückführung eines Polynoms auf eine Ansammlung von Werten
Polynome besitzen Ableitungen und Integrale: Natürliche Auswahl von Polynomen zur Approximation von Funktionen!

3.1 Effiziente Berechnung von Polynomen: Horner-Schema

Gegeben sei ein Polynom

$$P(x) = \sum_{i=0}^{n} c_i \cdot x^i$$

mit $(n+1)$ Termen (Polynom vom Grad n)

Rechenaufwand zur Auswertung:

- $O(n)$ Multiplikationen der Potenzen mit Koeffizienten
- $O(n-1)$ Mulitplikationen zur Bildung der Potenzen

Einfache Auswertung resultiert in $O(2n)$ Rechenoperationen!
Außerdem:
Benutzung des Zwischenspeichers für Werte der Potenzen bei gegebenen Wert von x.

Höhere Effizienz: Horner-Schema

$$P(x) = \sum_{i=0}^{n} c_i \cdot x^i = c_0 + x(c_1 + x(c_2 + x(...(c_{n-1} + c_n x)))...)$$

Rechenaufwand des Horner-Schemas:

- $O(n)$ Additionen (keine explizite Neuberechnung der Potenzen)
- Kein Speichern der Zwischenwerte nötig!

Fazit:
Das Horner-Schema ermöglicht eine schnellere Berechnung von Polynomen mit $O(n)$ Rechenoperationen und Werte müssen nicht zwischengespeichert werden!

Beispiel:
$1 + 7x - 5x^2 - 3x^3 + x^4 = P(x)$

Einfacher Ansatz: $7 \cdot x - (5 \cdot x) \cdot x - (3 \cdot x^2) \cdot x + (4 \cdot x^3) \cdot x$
$\Rightarrow$ Rechenoperationen: 7 Multiplikationen!

Horner: $x \cdot (7 + x \cdot (-5 + x \cdot (-3 + x \cdot 4)))$
$\Rightarrow$ Rechenoperationen: 4 Multiplikationen

In Python: `numpy.polyval()`
Anm.: Auswertung hier in umgekehrter Reihenfolge $\sum_{i=0}^{n} c_i \cdot x^{n-i}$

3.1.1 Polynomdivision zur Bestimmung von Nullstellen mittels des Horner-Schemas

Beweis für Nullstellenbestimmung mit Horner-Schema:

$$
\begin{aligned}
P(x) &= \sum_{i=0}^{n} c_i \cdot x^i = x \cdot \left(\sum_{i=0}^{n-1} c_{i+1} \cdot x^i \right) + c_0 \\
&= x \cdot \left(\sum_{i=0}^{n-2} d'_{i+1} \cdot x^i \cdot (x - x_0) + d'_0 \right) + c_0 \\
&\text{mit } d'_i = c_{i+1} + x_0 \cdot (c_{i+2} + x_0 \cdot (\dots \ (c_{n-1} + x_0 \cdot c_n)) \dots) \equiv d_{i+1} \\
P(x) &= \sum_{i=0}^{n-2} d_{i+2} \cdot x^{i+1} \cdot (x - x_0) + \underbrace{d_1 \cdot x + c_0}_{d_0} \\
&= \left(\sum_{i=0}^{n-1} d_{i+1} \cdot x^i \right) (x - x_0) + d_0
\end{aligned}
$$

$d_0 \dots$ Divisionsrest; bei Teilbarkeit von $P(x)$ durch $(x - x_0)$ ist $d_0 = 0$

3.2 Taylor-Polynome

Die Grundlage in der Anwendung von Polynomen zur Interpolation liegt im Weierstraßschen Approximationssatz

Weierstraßscher Approximationssatz

Gegeben sei eine auf $[a, b]$ definierte und stetige Funktion $f(x)$. Dann gibt es zu jedem $\epsilon > 0$ ein auf $[a, b]$ definiertes Polynom $P(x)$, so dass

$$|f(x) - P(x)| < \epsilon \text{ für alle } x \in [a, b]$$

gilt.

Eine einfache Ausnutzung des obigen Approximationssatzes liegt in der Anwendung von Taylor-Polynomen

Definition: Taylor-Reihe und Taylor-Polynome

Ist eine Funktion $f(x)$ um einen Punkt hinreichend gut differenzierbar, so lässt sie sich als Taylor-Reihe

$$f(x) = P_n(x) + R_n(x)$$

mit

$$P_n(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k$$

und Restglied

$$R_n(x) = \frac{f^{(n-1)}(\zeta(x))}{(n+1)!}(x - x_0)^{n+1}$$

mit Zahl $\zeta(x)$ zwischen x und x_0 darstellen.
Die Polynome $P_n(x)$ heißen Taylorpolynome mit

$$P_n(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + ...$$

$f^{(k)}$ sind die entsprechenden Ableitungen und Restglied $R_n(x)$ kennzeichnet den Abbruchfehler, falls $n \neq \infty$ (endliche Summation).

Bei Existenz der Ableitungen und $(x - x_0) \ll 1$: schnelle Konvergenz der Taylor-Reihe!
Bedeutung:
Endliche Anzahl von Taylor-Termen stellt eine gute polynomielle Näherung dar!
Problem:
Taylor-Polynome stimmen zwar mit einer gegebenen Funktion an einer festen Stützstelle sehr gut überein, aber Genauigkeit bezieht sich auf Stützstelle!
Ein gutes Interpolationspolynom muss aber über ein ganzes Intervall eine gute Approximation liefern!

Beispiel für Taylor-Polynome:

$$f(x) = e^x \text{ mit } x_0 = 0$$

Resultierende Taylor-Polynome:
$P_0(x) = 1,\ \ P_1(x) = 1 + x,\ \ P_2(x) = 1 + x + \frac{x^2}{2},\ \ P_3(x) = 1 + x + \frac{x^2}{2} + \frac{x^3}{6}$

x	$P_0(x)$	$P_1(x)$	$P_2(x)$	$P_3(x)$	e^x
0	1	1	1	1	1
0.5	1	1.5	1.625	1.646	1.649
1.0	1	2.0	2.5	2.667	2.718

- Grössere Abweichungen für größere Intervalle $(x - x_0)$

4 Literaturverzeichnis

[1] Böhm, Jan M. ; Hoock, Claudia ; Popper, Karl R.: *Karl Poppers kritischer Rationalismus heute: zur Aktualität kritisch-rationaler Wissenschaftstheorie.* Tübingen, Deutschland : Mohr Siebeck, 2002

[2] Frenkel, Daan ; Smit, Berend: *Understanding Molecular Simulation: From Algorithms to Applications.* Orlando, FL, USA : Academic Press, Inc., 1996

[3] Verlet, Loup: Computer experiments on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules. In: *Phys. Rev.* 159 (1967), Nr. 1, S. 98