Kapitel 2

Lineare und nichtlineare Gleichungen

In diesem Kapitel wollen wir lineare und nichtlineare Gleichungen bzw. genauer entsprechende Gleichungssysteme untersuchen und damit gleichzeitig einen kleinen Einblick in ein Teilgebiet der numerischen Mathematik geben.

2.1 Lineare Gleichungssysteme

2.1.1 Beispiele

Beispiel: Ein Betrieb kann 3 verschiedene Produkte A, B und C herstellen. Für A wird eine Arbeitsstunde, für B zwei und für C drei pro erzeugten Stück benötigt. Die Produkte werden zum Preis von 30, 50 und 70 Euro verkauft. Der Energieverbrauch pro Stück liegt bei 5, 3 und 2 kWh pro erzeugten Stück. Es werden in 40 Arbeitsstunden Produkte im Wert von 1040 Euro produziert bei einen Energieverbrauch von 93 kWh. Wieviel Stück der jeweiligen Produkte wurden erzeugt?

Wir nennen die jeweiligen Stückzahlen x, y und z. Die Aufgabe führt offenbar auf das lineare Gleichungssystem

$$\begin{pmatrix} 1 & 2 & 3 \\ 30 & 50 & 70 \\ 5 & 3 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 40 \\ 1040 \\ 93 \end{pmatrix}.$$

Eine Lösung mit Hilfe von Maple ist einfach:

```
> with(LinearAlgebra):
> A:=Matrix([[1,2,3],[30,50,70],[5,3,2]]):
> b:=<40,1040,93>:
> loes:=LinearSolve(A,b);
```

$$loes := \begin{bmatrix} 13 \\ 6 \\ 5 \end{bmatrix}$$

□

Beispiel: Unter `http://www.hta-bu.bfh.ch/e/mat/` haben wir die folgenden Aufgaben gefunden. Sie gehören zu einem Computing-Wettbewerb.

Moritz hat ein elektronisches Spiel zum Geburtstag bekommen: Es besteht aus einer 4×5-Matrix von Lämpchen (Leuchtknöpfen). Wenn man es einschaltet, leuchten einige davon auf, und wenn man auf einen der Knöpfe drückt, wechseln die unmittelbar benachbarten Knöpfe (rechts, links, oben, unten, falls vorhanden) ihren Zustand. Das Ziel ist, alle Lämpchen zum Erlöschen zu bringen.

- Welche Knöpfe muss Moritz drücken, wenn am Anfang alle Knöpfe aufleuchten?
- Schreiben Sie ein Programm, welches zu beliebiger Anfangskonfiguration diejenigen Lämpchen bestimmt, die gedrückt werden müssen.

Wir lösen die zweite Aufgabe, denn dann wird auch die erste gelöst sein. Hierbei untersuchen wir die Aufgabe, mit welcher Zugfolge man von lauter gelöschten Lämpchen zur Ausgangskonfiguration kommt. Die gleiche Zugfolge löst dann auch unser Problem. Wir gehen allgemeiner von einer $m \times n$-Matrix aus, also von insgesamt mn Lämpchen. Diese denken wir uns in einer festen Reihenfolge nummeriert, etwa zeilenweise von links nach rechts und oben beginnend. Ein *Lämpchen-Vektor* ist ein Vektor $x = (x_1, \ldots, x_{mn})^T$, wobei

$$x_i := \begin{cases} 1, & \text{das } i\text{-te Lämpchen leuchtet,} \\ 0, & \text{sonst,} \end{cases} \qquad i = 1, \ldots, mn.$$

Dae Anfangszustand wird in einem Vektor $b \in \mathbb{R}^{mn}$ abgelegt, naheliegenderweise ist

$$b_i := \begin{cases} 1, & \text{das } i\text{-te Lämpchen leuchtet zu Beginn,} \\ 0, & \text{sonst,} \end{cases} \qquad i = 1, \ldots, mn.$$

Weiter definieren wir die Matrix $A \in \mathbb{R}^{mn \times mn}$ durch

$$a_{ij} := \begin{cases} 1, & \text{Lämpchen } i \text{ ist Nachbar von Lämpchen } j, \\ 0, & \text{sonst,} \end{cases} \qquad i, j = 1, \ldots, mn.$$

Dann kommt es darauf an, einen Lämpchen-Vektor x zu bestimmen derart, dass $Ax \equiv b \mod 2$. Die Matrix A stellt sich als eine Block-Tridiagonalmatrix heraus, sie hat die Form

$$A = \begin{pmatrix} B & I & & \\ I & B & \ddots & \\ & \ddots & \ddots & I \\ & & I & B \end{pmatrix} \in \mathbb{R}^{mn \times mn},$$

wobei I die $n \times n$-Einheitsmatrix bedeutet und B selbst eine Tridiagonalmatrix ist, nämlich

$$B := \begin{pmatrix} 0 & 1 & & \\ 1 & 0 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & 0 \end{pmatrix} \in \mathbb{R}^{n \times n}.$$

Jetzt wäre es interessant, eine Antwort auf die folgende Frage zu kennen:

- Für welche (m, n) ist $\det A = \pm 1$?

Ist nämlich $\det A = \pm 1$, so ist A^{-1} eine ganzzahlige Matrix (Beweis?) und daher $A^{-1}b$ ein ganzzahliger Vektor, der nach Reduktion modulo 2 zu dem gesuchten Lämpchen-Vektor wird. Für kleine m, n geben wir $\det A$ in der folgenden Tabelle an:

$m \setminus n$	1	2	3	4	5	6
1	0	−1	0	1	0	−1
2	0	0	−1	1	0	1
3	0	−1	0	1	0	−1
4	0	1	1	0	1	1
5	0	0	0	1	0	−1
6	0	1	−1	1	−1	0

Wer kann eine allgemeine Beziehung beweisen? Für $(m, n) = (4, 5)$ ist die Determinante von A jedenfalls 1, daher kann aus jeder Anfangskonfiguration heraus das Löschen aller Lämpchen erreicht werden. Ist z. B. $b = (1, \ldots, 1)^T \in \mathbb{R}^{20}$ bzw. alle Lämpchen zu Beginn an, so ist

$$x = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix},$$

wobei wir den Lämpchen-Vektor gleich schon als Lämpchen-Matrix schreiben. □

Beispiel: In diesem Beispiel wollen wir uns überlegen, dass die Berechnung der Temperaturverteilung in einem Metallstab bzw. einer quadratischen Metallplatte bei gegebener Randtemperatur auf das Lösen eines linearen Gleichungssystems zurückgeführt werden kann.

Zunächst betrachten wir das eindimensionale Problem. Gegeben sei ein Stab, den man sich in $n + 1$ äquidistante Teile zerlegt denkt, die jeweils konstante Temperatur annehmen. Es gibt also n innere Punkte. In diesen sei die Temperatur jeweils der Mittelwert der Temperaturen beider Nachbarn. Mit u_j, $j = 1, \ldots, n$, bezeichnen wir die Temperatur im j-ten inneren Knoten, u_0 und u_{n+1} sind die vorgegebenen Randtemperaturen im linken bzw. rechten Stabende. Für die unbekannten $u_1, \ldots, u_n$ erhält man (nach Multiplikation mit 2) das lineare Gleichungssystem

$$\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \\ u_n \end{pmatrix} = \begin{pmatrix} u_0 \\ 0 \\ \vdots \\ 0 \\ u_{n+1} \end{pmatrix}.$$

Jetzt betrachten wir eine zweidimensionale Temperaturverteilung auf einer quadratischen Metallplatte. Diese sei durch ein quadratisches Gitter zerlegt, wobei man zeilen- und spaltenweise jeweils n innere Knoten habe. Wir beschränken uns hier auf den Fall $n = 3$. Die gegebenen Randtemperaturen in den 12 Randknoten seien $t_1, \ldots, t_{12}$, die gesuchten Temperaturen in den inneren Knoten seien mit $u_{i,j}$, $1 \le i, j \le 3$, bezeichnet.

Man kann dies folgendermaßen darstellen:

	t_1	t_2	t_3	
t_{12}	$u_{1,1}$	$u_{1,2}$	$u_{1,3}$	t_4
t_{11}	$u_{2,1}$	$u_{2,2}$	$u_{2,3}$	t_5
t_{10}	$u_{3,1}$	$u_{3,2}$	$u_{3,3}$	t_6
	t_9	t_8	t_7	

Man erhält $n^2 = 9$ Gleichungen in den inneren Knoten. Das lineare Gleichungssystem lautet (nach Multiplikation mit 4)

$$\left(\begin{array}{rrr|rrr|rrr} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ \hline -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ \hline 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{array}\right) \left(\begin{array}{c} u_{1,1} \\ u_{1,2} \\ u_{1,3} \\ \hline u_{2,1} \\ u_{2,2} \\ u_{2,3} \\ \hline u_{3,1} \\ u_{3,2} \\ u_{3,3} \end{array}\right) = \left(\begin{array}{c} t_1 + t_{12} \\ t_2 \\ t_3 + t_4 \\ \hline t_{11} \\ 0 \\ t_5 \\ \hline t_9 + t_{10} \\ t_8 \\ t_6 + t_7 \end{array}\right).$$

Es ist nun einfach, die Gestalt der Koeffizientenmatrix für allgemeines n zu raten. Man stellt fest, dass man wie im vorigen Beispiel zu einer Block-Tridiagonalmatrix gelangt ist. □

Beispiel: Wir übernehmen (wörtlich) ein Beispiel bei N. KÖCKLER (1990, S. 50 ff.)[1]. Ein neu entdeckter Himmelskörper, der sich auf einer Umlaufbahn um die Sonne bewegt, wurde an 10 Positionen beobachtet. Die kartesischen Koordinaten (x_i, y_i), $i = 1, \ldots, 10$, dieser Positionen, dargestellt in einem angepassten Koordinatensystem in der Bahnebene, sind in der folgenden Tabelle wiedergegeben:

Beobachtung Nr.	x_i	y_i
1	−1.024940	−0.389269
2	−0.949898	−0.322894
3	−0.866114	−0.265256
4	−0.773392	−0.216557
5	−0.671372	−0.177152
6	−0.559524	−0.147582
7	−0.437067	−0.128618
8	−0.302909	−0.121353
9	−0.155493	−0.127348
10	−0.007464	−0.148885

Die Bahn des Himmelskörpers ist eine Ellipse mit der Sonne in einem der beiden Brennpunkte (erstes Keplersches Gesetz):

$$x^2 = ay^2 + bxy + cx + dy + e.$$

[1] N. KÖCKLER (1990) *Numerische Algorithmen in Softwaresystemen.* B. G. Teubner, Stuttgart.

Setzen wir in diese Darstellung die beobachteten Positionen ein, so bekommen wir 10 lineare Gleichungen für die fünf unbekannten Koeffizienten a, b, c, d, e, die in Matrixform folgendermaßen aussehen:

$$\begin{pmatrix} y_1^2 & x_1y_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ y_{10}^2 & x_{10}y_{10} & x_{10} & y_{10} & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ e \end{pmatrix} = \begin{pmatrix} x_1^2 \\ \vdots \\ x_{10}^2 \end{pmatrix}.$$

Es handelt sich hier um ein *überbestimmtes* lineares Gleichungssystem, welches wegen der Beobachtungsfehler nicht exakt lösbar ist. Daher wendet man i. Allg. die *Methode der kleinsten Quadrate* an (wir kommen hierauf später ausführlich zurück) und bestimmt eine Lösung, für die der Defekt im Sinne der euklidischen Norm minimal ist. Bei Maple gibt es im `LinearAlgebra`-Package die Funktion `LeastSquares`, mit der eine solche Lösung bestimmt werden kann. Wir wollen hier eine MATLAB-Funktion `Kepler` angeben. Hierzu schreibt man in ein File `Kepler.m` z. B. den folgenden Inhalt.

```
function z=Kepler(x,y);
%*****************************************************
%Zu vorgegebenen Koordinaten (x,y) wird nach der
%Methode der kleinsten Quadrate die Darstellung einer
%Ellipse ausgegeben.
%*****************************************************
%Input-Parameter
%        x,y      zwei Vektoren, deren Laenge deutlich
%                 groesser als 5 ist.
%Output-Parameter
%          z      Die gesuchte Ellipse hat die
%                 Darstellung
%         x^2=z(1)*y^2+z(2)*xy+z(3)*x+z(4)*y+z(5)
%*****************************************************
A=[y.^2,x.*y,x,y,ones(length(x),1)];
rhs=x.^2; z=A\rhs;
```

Besetzen wir die Vektoren und rufen die Funktion `Kepler` auf, etwa durch

```
>> format long;
>>x=[-1.024940;-0.949898; -0.866114;-0.773392;-0.671372;-0.559524;-0.437067;...
     -0.302909;-0.155493;-0.007464];
>>y=[-0.389269;-0.322894;-0.265256;-0.216557;-0.177152; -0.147582;-0.128618;...
     -0.121353;-0.127348;-0.148885];
>>z=Kepler(x,y)
```

so erhalten wir

$$z = \begin{pmatrix} -1.38334886512014 \\ -0.66464965048688 \\ -0.67112854539521 \\ -3.37090756374251 \\ -0.47504214706864 \end{pmatrix}.$$

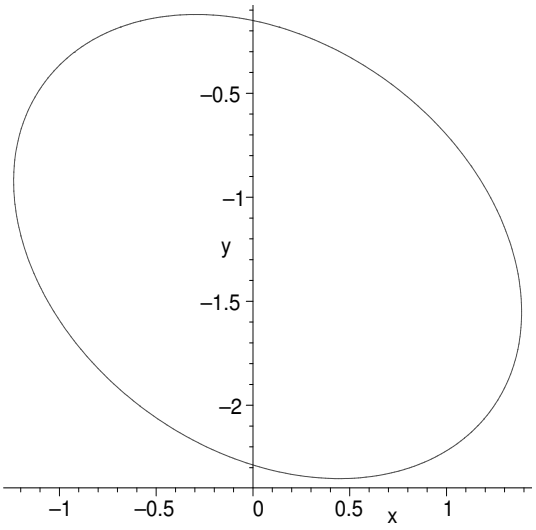

Abbildung 2.1: Bahn eines Himmelskörpers

Das Zeichnen der Ellipse geschieht am einfachsten mit Hilfe des `implicitplot`-Befehl von Maple. Die in Abbildung 2.1 gezeichnete Ellipse haben wir mit den folgenden Befehlen gewonnen:

```
with(plots):
a:=-1.383: b:=-0.665: c:=-0.671: d:=-3.371: e:=-0.475:
implicitplot(x^2=a*y^2+b*x*y+c*x+d*y+e,x=-1.3..1.4,y=-2.4..0,
    grid=[80,80],scaling=constrained);
```

Eine entsprechende Abbildung mit MATLAB herzustellen wäre wohl komplizierter. □

Beispiel: Interpolationsprobleme führen sehr oft auf die Aufgabe, ein lineares Gleichungssystem zu lösen. Gegeben seien z. B. paarweise verschiedene sogenannte Stützstellen $x_0, \ldots, x_n \in \mathbb{R}$ sowie zugehörige Stützwerte $f_0, \ldots, f_n \in \mathbb{R}$. Mit $\mathcal{P}_n$ bezeichnen wir den $(n+1)$-dimensionalen linearen Raum der Polynome vom Grad $\leq n$. Die Interpolationsaufgabe bestehe darin, ein $p \in \mathcal{P}_n$ mit $p(x_i) = f_i$, $i = 0, \ldots, n$, zu bestimmen. Diese Aufgabe ist eindeutig lösbar. Um dies einzusehen wähle man eine Basis von $\mathcal{P}_n$, es sei also $\mathcal{P}_n = \operatorname{span}\{v_0, \ldots, v_n\}$. Z. B. ist $v_j(x) := x^j$, $j = 0, \ldots, n$ (Monom-Basis). Der Ansatz $p(x) = \sum_{j=0}^{n} \alpha_j v_j(x)$ genügt den geforderten Interpolationsbedingungen genau dann, wenn $\sum_{j=0}^{n} \alpha_j v_j(x_i) = f_i$, $i = 0, \ldots, n$, bzw. $\alpha = (\alpha_0, \ldots, \alpha_n)^T$ Lösung des linearen Gleichungssystems

$$\begin{pmatrix} v_0(x_0) & \cdots & v_n(x_0) \\ \vdots & & \vdots \\ v_0(x_n) & \cdots & v_n(x_n) \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \vdots \\ \alpha_n \end{pmatrix} = \begin{pmatrix} f_0 \\ \vdots \\ f_n \end{pmatrix}$$

ist. Die Koeffizientenmatrix dieses linearen Gleichungssystems ist nichtsingulär. Denn ist

$$\begin{pmatrix} v_0(x_0) & \cdots & v_n(x_0) \\ \vdots & & \vdots \\ v_0(x_n) & \cdots & v_n(x_n) \end{pmatrix} \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_n \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix},$$

so ist $q(x) := \sum_{j=0}^{n} \beta_j v_j(x)$ ein Polynom aus $\mathcal{P}_n$ mit den $n+1$ paarweise verschiedenen Nullstellen $x_0, \ldots, x_n$, also notwendigerweise das Nullpolynom. Daher ist das oben angegebene Interpolationsproblem stets eindeutig lösbar.

Bei der praktischen Anwendung nutzt man aus, dass man die Wahl einer Basis von $\mathcal{P}_n$ in der Hand hat. Wählt man die Monombasis, so erhält man als Koeffizientenmatrix eine sogenannte Vandermondesche Matrix

$$V := \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix},$$

also eine voll besetzte Matrix, die i. Allg. auch noch ungünstige Stabilitätseigenschaften hat (was hier nicht näher erläutert werden soll). Das andere Extrem besteht darin, dass man eine sogenannte *Lagrange-Basis* wählt. Hier ist das j-te Basiselement gegeben durch

$$L_j(x) := \prod_{\substack{k=0 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k}, \qquad j = 0, \ldots, n.$$

Wegen $L_j(x_i) = \delta_{ij}$ (Kronecker-Symbol), $0 \leq i, j \leq n$, ist hier die Koeffizientenmatrix im linearen Gleichungssystem die Identität und

$$p(x) = \sum_{j=0}^{n} f_j L_j(x)$$

die sogenannte *Lagrange-Darstellung* des Interpolationspolynoms. Bei der *Newton-Darstellung* ist das j-te Basiselement durch

$$N_j(x) := \prod_{k=0}^{j-1} (x - x_k), \qquad j = 0, \ldots, n$$

gegeben (wobei natürlich $N_0(x) = 1$). Wegen $N_j(x_i) = 0$, $i = 0, \ldots, j-1$, ist nun die Koeffizientenmatrix im linearen Gleichungssystem eine untere Dreiecksmatrix (mit nichtverschwindenden Diagonalelementen). Dies erlaubt das sukzessive Lösen des linearen Gleichungssystems durch sogenannte Vorwärtssubstitution. Der Newton-Ansatz hat gegenüber dem von Lagrange den Vorteil, dass die Hinzunahme einer neuen Stützstelle nicht die gesamte bisherige Arbeit überflüssig macht. Anmerkungen zur Interpolation von Polynomen auf relativ elementarem Niveau findet man auch bei T. Sonar (2001, S. 59 ff.)[2].

Eine Interpolation mit Polynomen hohen Grades ist nicht sinnvoll, weil die entstehenden Polynome zu starken Oszillationen neigen. Dies wollen wir durch ein berühmtes

[2] T. Sonar (2001) *Angewandte Mathematik, Modellbildung und Informatik. Eine Einführung für Lehamtsstudenten, Lehrer und Schüler.* Viewg, Braunschweig-Wiesbaden.

Beispiel von C. Runge (1901) demonstrieren. Es sei

$$f(x) := \frac{1}{1+x^2}, \qquad x \in [-5,5].$$

Man wähle eine äquidistante Zerlegung des Intervalls $[-5,5]$ und nehme als Stützwerte die entsprechenden Funktionswerte von f, setze also

$$x_i := -5 + \frac{i10}{n}, \qquad f_i := f(x_i) \qquad (i = 0, \ldots, n).$$

In Abbildung 2.2 links geben wir f (gestrichelt) und das Interpolationspolynom für $n = 5$ an, rechts entsprechend für $n = 10$. Statt Polynome hoher Ordnung benutzt man

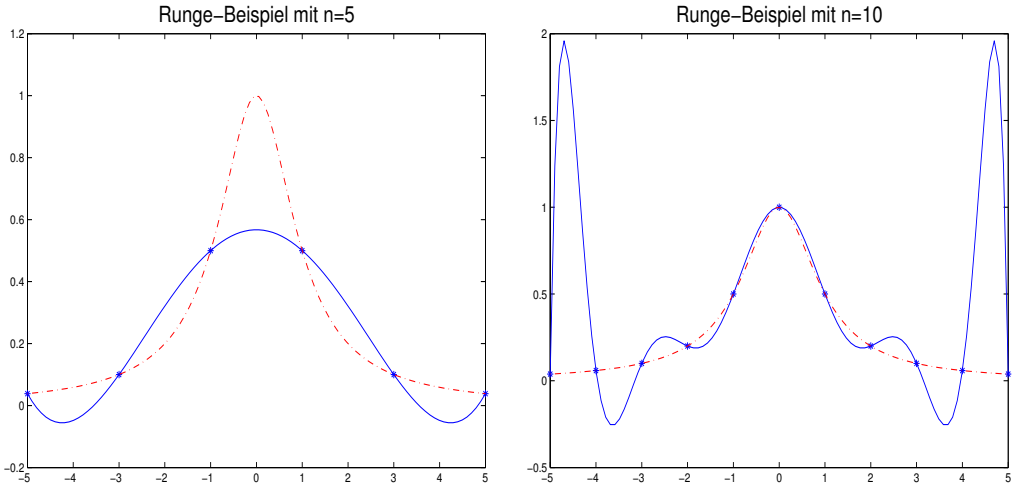

Abbildung 2.2: Das Runge-Beispiel mit $n = 5$ und $n = 10$

bei der Interpolation lieber stückweise Polynome niedriger Ordnung. Bezeichnet man mit Δ_n die Zerlegung $x_0 < \cdots < x_n$ (bisher mussten die Stützstellen nicht ansteigend geordnet sein), so nennt man ein Element aus

$$S_3(\Delta_n) := \{s \in C^2[x_0, x_n] : s|_{[x_j, x_{j+1}]} \in \mathcal{P}_3 \; j = 0, \ldots, n-1\}$$

einen *kubischen Spline* zur vorgegebenen Zerlegung Δ_n. Dies ist also eine auf dem Gesamtintervall $[x_0, x_n]$ zweimal stetig differenzierbare Funktion, welche restringiert auf jedes der Teilintervalle $[x_j, x_{j+1}]$, $j = 0, \ldots, n-1$, ein kubisches Polynom ist. Offensichtlich ist $S_3(\Delta_n)$ ein linearer Raum. Wir wollen es plausibel machen (das ist etwas anderes als ein Beweis!), dass $\dim S_3(\Delta_n) = n+3$. Auf jedem der n Teilintervalle ist ein kubischer Spline ein Polynom vom Grad ≤ 3, so dass man höchstens $4n$ Freiheitgrade hat (es ist $\dim \mathcal{P}_3 = 4$). Durch die Forderung nach zweimaliger stetiger Differenzierbarkeit in den $n-1$ inneren Knoten hat man aber noch $3(n-1)$ Bedingungen und damit insgesamt $4n - 3(n-1) = n+3$ Freiheitsgrade bei der Bestimmung eines kubischen Splines. Interpolationsbedingungen $s(x_j) = f_j$, $j = 0, \ldots, n$, in den $n+1$ Stützstellen $x_0, \ldots, x_n$ lassen noch zwei Bedingungen frei. Die bei weitem wichtigsten Zusatzbedingungen sind:

- Hermitesche Randbedingung: Die Steigungen bzw. Ableitungen des Splines an den beiden Intervallenden werden vorgegeben. Es seien also zusätzlich f_0' und f_n' gegeben, die Zusatzbedingungen sind $s'(x_0) = f_0'$ und $s'(x_n) = f_n'$.

- Natürliche Randbedingung[3]: Es werden die zweiten Ableitungen an den Intervallenden vorgegeben. Gegeben seien also f_0'' und f_n'' (sehr oft ist $f_0'' = f_n'' = 0$), die beiden Zusatzbedingungen sind $s''(x_0) = f_0''$ und $s''(x_n) = f_n''$.

- Not-a-knot Bedingung: Es wird gefordert, dass der kubische Spline in dem ersten und dem letzten inneren Knoten, also in x_1 und x_{n-1}, dreimal stetig differenzierbar ist. Die Zusatzbedingungen sind also $s'''(x_1 - 0) = s'''(x_1 + 0)$ und $s'''(x_{n-1}-0) = s'''(x_{n-1}+0)$. Diese Bedingung ist angebracht, wenn keine Informationen über erste oder zweite Ableitungen an den Intervallenden zur Verfügung stehen.

Auf die Aufstellung der entsprechenden Gleichungssysteme gehen wir in den Aufgaben 3, 4 und 5 näher ein. □

2.1.2 *LU*-, *QR*- und Cholesky-Zerlegung: Motivation

Gegeben sei ein lineares Gleichungssystem $Ax = b$ mit einer nichtsingulären Koeffizientenmatrix $A \in \mathbb{R}^{n\times n}$ und einer rechten Seite $b \in \mathbb{R}^n$. *Direkte Verfahren* zur Lösung linearer Gleichungssysteme beruhen darauf, eine gewisse Zerlegung der Koeffizientenmatrix zu berechnen und das gegebene lineare Gleichungssystem in ein äquivalentes lineares Gleichungssystem (*äquivalent* soll hierbei heißen, dass die Lösungen übereinstimmen) mit einer oberen Dreecksmatrix als Koeffizientenmatrix zu transformieren. Ein solches lineares Gleichungssystem kann dann durch sogenanntes Rückwärtseinsetzen (backward substitution) mit $\frac{1}{2}n(n+1)$ "flops" gelöst werden[4]. Genauer werden wir konstruktiv, also durch Angabe eines Algorithmus, zeigen:

- *LU*-Zerlegung: Sei $A \in \mathbb{R}^{n\times n}$ nichtsingulär. Dann existiert eine Permutationsmatrix $P \in \mathbb{R}^{n\times n}$ (das ist eine quadratische Matrix, die in jeder Zeile und jeder

[3] In Maple gibt es den Befehl `spline`. Z. B. erhält man durch `spline([0,1,2,3],[0,1,4,3],x,3);` den kubischen Spline

$$s(x) = \begin{cases} \frac{1}{5}x + \frac{4}{5}x^3, & 0 \le x \le 1, \\ \frac{14}{5} - \frac{41}{5}x + \frac{42}{5}x^2 - 2x^3, & 1 \le x \le 2, \\ -\frac{114}{5} + \frac{151}{5}x - \frac{54}{5}x^2 + \frac{6}{5}x^3, & 2 \le x \le 3. \end{cases}$$

Dies ist ein natürlicher Spline in dem (eingeschränkten) Sinne, dass die zweiten Ableitungen an den Intervallenden verschwinden.

[4] Auf diese Anzahl ist schon der 10-jährige C. F. Gauß gekommen. Um die Schüler "ruhig zu stellen", gab der Lehrer ihnen die Aufgabe, die Zahlen von 1 bis 100 zu addieren. Gauß gab die richtige Antwort 5050 in Sekunden, sein Trick ist inzwischen wohlbekannt: Er addierte zunächst 1 und 99, dann 2 und 98 usw., erhält also als Summe der Zahlen von 1 bis 100 ohne 50 und 100 genau $49 \cdot 100 = 4900$. Hinzufügen der beiden fehlenden Zahlen ergibt die gesuchte Zahl.

Spalte genau eine 1 als Eintrag besitzt und sonst nur Nullen), eine untere Dreiecksmatrix $L \in \mathbb{R}^{n\times n}$ (also alle Einträge unterhalb der Diagonalen verschwinden bzw. $l_{ij} = 0$ für $i < j$) mit Einsen in der Diagonalen und eine obere Dreiecksmatrix $U \in \mathbb{R}^{n\times n}$ (also alle Einträge oberhalb der Diagonalen verschwinden bzw. $u_{ij} = 0$ für $i > j$) mit

$$PA = LU.$$

Die Zerlegung wird mit Hilfe des *Gaußschen Eliminationsverfahrens* berechnet.

- QR-Zerlegung: Sei $A \in \mathbb{R}^{m\times n}$ mit $m \geq n$ und Rang $(A) = n$ gegeben[5]. Dann gibt es eine orthogonale Matrix $Q \in \mathbb{R}^{n\times n}$ (d. h. es ist $Q^TQ = I$) und eine obere Dreiecksmatrix $R \in \mathbb{R}^{m\times n}$ mit

$$A = QR.$$

Diese Zerlegung kann man mit sogenannten Householder-Spiegelungen oder mit sogenannten Givens-Rotationen berechnen.

- Cholesky-Zerlegung: Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch (d. h. $a_{ij} = a_{ji}$ für $1 \leq i, j \leq n$) und positiv definit. Dann existiert (genau) eine untere Dreiecksmatrix $L \in \mathbb{R}^{n\times n}$ mit positiven Diagonalelementen und

$$A = LL^T.$$

Wir werden nun zeigen, weshalb es für die numerische Lösung eines linearen Gleichungssystems (bzw. eines linearen Least Square Problems) wünschenswert ist, eine der eben angegebenen Zerlegungen zu berechnen.

- LU-Zerlegung: Gegeben sei ein lineares Gleichungssystem $Ax = b$ mit nichtsingulärem $A \in \mathbb{R}^{n\times n}$. Bekannt sei eine LU-Zerlegung $PA = LU$, also quadratische Matrizen P, L, U mit den oben angegebenen Eigenschaften. Eine Permutationsmatrix $P \in \mathbb{R}^{n\times n}$ ist durch eine Permutation $\{\pi(1), \ldots, \pi(n)\}$ der Zahlen $\{1, \ldots, n\}$ gegeben: In der i-ten Zeile steht in der $\pi(i)$-ten Spalte eine 1, sonst nur Nullen. Mit Hilfe des Kronecker-Symbols kann man dies auch durch $p_{ij} = \delta_{\pi(i)j}$, $1 \leq i, j \leq n$, ausdrücken. Eine Permutationsmatrix ist orthogonal[6] und folglich nichtsingulär. Daher ist das lineare Gleichungssystem $Ax = b$ äquivalent zu $PAx = Pb$. Die neue rechte Seite Pb entsteht aus b durch eine Permutation der Komponenten, genauer ist

$$(Pb)_i = \sum_{j=1}^{n} p_{ij}b_j = \sum_{j=1}^{n} \delta_{\pi(i)j}b_j = b_{\pi(i)}, \qquad i = 1, \ldots, n.$$

[5] Z. B. ist $m = n$ und A nichtsingulär. Der allgemeinere Fall wird zugelassen, um auch lineare Least Square Probleme zu erfassen.

[6] Denn es ist

$$(P^TP)_{ij} = \sum_{k=1}^{n} (P^T)_{ik}(P)_{kj} = \sum_{k=1}^{n} p_{ki}p_{kj} = \sum_{k=1}^{n} \delta_{\pi(k)i}\delta_{\pi(k)j} = \delta_{ij}.$$

Das lineare Gleichungssystem $PAx = LUx = Pb$ wird in zwei Schritten gelöst. Zunächst bestimmt man $y = Ux$ als Lösung von $Ly = Pb$ durch Vorwärtseinsetzen (forward substitution), danach berechnet man die gesuchte Lösung x durch Rückwärtseinsetzen aus $Ux = y$.

- QR-Zerlegung: Gegeben seien eine Matrix $A \in \mathbb{R}^{m\times n}$ mit $m \geq n$ und Rang $(A) = n$, ein Vektor $b \in \mathbb{R}^m$ und hiermit das lineare Least Square Problem

$$\text{Minimiere} \quad \|Ax - b\|_2, \quad x \in \mathbb{R}^n.$$

Hierbei sei $\|\cdot\|_2$ die *euklidische Norm* auf dem $\mathbb{R}^m$. Für $y \in \mathbb{R}^m$ ist also

$$\|y\|_2 := \sqrt{y^T y} = \Big(\sum_{i=1}^{m} y_i^2\Big)^{1/2}.$$

Angenommen, es sei eine QR-Zerlegung von A bekannt, also eine orthogonale Matrix $Q \in \mathbb{R}^{m\times m}$ und eine obere Dreiecksmatrix R. Da wir Rang $(A) = n$ bzw. die lineare Unabhängigkeit der n Spalten von A voraussetzen und Q als orthogonale Matrix nichtsingulär ist, sind auch die n Spalten von R linear unabhängig. Als obere $m \times n$-Dreiecksmatrix hat R die Form

$$R = \begin{pmatrix} \hat{R} \\ 0 \end{pmatrix}$$

mit einer oberen $n \times n$-Dreiecksmatrix $\hat{R}$, welche wegen des eben gesagten nichtsingulär ist. Zur Abkürzung setzen wir

$$\begin{pmatrix} c \\ d \end{pmatrix} := Q^T b \qquad \text{mit} \quad c \in \mathbb{R}^n,\ d \in \mathbb{R}^{m-n}.$$

Für ein beliebiges $x \in \mathbb{R}^n$ ist dann

$$\begin{aligned} \|Ax - b\|_2^2 &= \|QRx - b\|_2^2 \\ &= \|Rx - Q^T b\|_2^2 \\ &= \left\| \begin{pmatrix} \hat{R} \\ 0 \end{pmatrix} x - \begin{pmatrix} c \\ d \end{pmatrix} \right\|_2^2 \\ &= \|\hat{R}x - c\|_2^2 + \|d\|_2^2. \end{aligned}$$

Hierbei haben wir ausgenutzt, dass die euklidische Norm invariant unter der Transformation mit einer orthogonalen Matrix ist. Genauer: Ist $y \in \mathbb{R}^m$ und $Q \in \mathbb{R}^{m\times m}$ orthogonal, so ist

$$\|Qy\|_2 = \sqrt{(Qy)^T (Qy)} = \sqrt{y^T Q^T Q y} = \sqrt{y^T y} = \|y\|_2.$$

Daher wird $\|Ax - b\|_2$ minimal, wenn $\hat{R}x = c$. Dieses lineare Gleichungssystem mit einer (nichtsingulären) oberen Dreiecksmatrix kann man durch Rückwärtseinsetzen lösen.

- Cholesky-Zerlegung: Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch und positiv definit. Wie wir zeigen werden, besitzt A dann eine Cholesky-Zerlegung, es existiert also eine untere Dreiecksmatrix $L \in \mathbb{R}^{n\times n}$ mit positiven Diagonalelementen derart, dass $A = LL^T$. Umgekehrt gilt natürlich: Ist $A = LL^T$ mit einer nichtsingulären Matrix $L \in \mathbb{R}^{n\times n}$, so ist A symmetrisch und positiv definit. Ist eine Cholesky-Zerlegung $A = LL^T$ der symmetrischen, positiv definiten Matrix A bekannt, so kann ein lineares Gleichungssystem $Ax = b$ wieder durch Vorwärts- und Rückwärtseinsetzen gelöst werden: Zunächst man $y = L^T x$ aus $Ly = b$ durch Vorwärtseinsetzen, danach die gesuchte Lösung x aus $L^T x = y$ durch Rückwärtseinsetzen.

Jetzt wollen wir kurz auf die Möglichkeiten von Maple und MATLAB zur Berechnung dieser Zerlegungen eingehen.

- LU-Zerlegung: Im Package `LinearAlgebra` von Maple gibt es zur LU-Zerlegung die Funktion `LUDecomposition`, in MATLAB die Funktion `lu`. Hilfen zu diesen Funktionen erhält man durch `?LUDecomposition` bzw. `help lu`. Es ist wichtig sich zu informieren, weil der Gebrauch des Begiffes LU-Zerlegung nicht ganz einheitlich ist. In Maple werden z. B. durch

  ```
  (P,L,U)=LinearAlgebra[LUDecomposition](A);
  ```

  eine Permutationsmatrix P, eine untere Dreiecksmatrix L mit Einsen in der Diagonalen und eine obere Dreiecksmatrix U mit $PLU = A$ berechnet, in MATLAB erhält man durch

  ```
  [L,U,P]=lu(A)
  ```

  eine Zerlegung $PA = LU$ mit Matrizen P, L, U, die die üblichen Eigenschaften haben.

  Beispiel: Sei

  $$A := \begin{pmatrix} 0 & 1 & 1 & -3 \\ -2 & 3 & 1 & 4 \\ 0 & 0 & 0 & 1 \\ 3 & 1 & 0 & 0 \end{pmatrix}.$$

  Zu berechnen sei eine LU-Zerlegung. Zunächst verwenden wir Maple:

  ```
  > with(LinearAlgebra):
  > A:=Matrix([[0,1,1,-3],[-2,3,1,4],[0,0,0,1],[3,1,0,0]]):
  > (P,L,U):=LUDecomposition(A);
  ```

  $$P,\, L,\, U := \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{3}{2} & \frac{11}{2} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} -2 & 3 & 1 & 4 \\ 0 & 1 & 1 & -3 \\ 0 & 0 & -4 & \frac{45}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  ```
  > P:=Transpose(P):
  ```

```
> Defect:=P.A-L.U;
```

$$Defect := \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Und nun noch MATLAB:

```
A=[0,1,1,-3;-2,3,1,4;0,0,0,1;3,1,0,0];
format long;
[L,U,P]=lu(A)
```

ergibt

$$L = \begin{pmatrix} 1.00000000000000 & 0 & 0 & 0 \\ -0.66666666666667 & 1.00000000000000 & 0 & 0 \\ 0 & 0.27272727272727 & 1.00000000000000 & 0 \\ 0 & 0 & 0 & 1.00000000000000 \end{pmatrix},$$

ferner

$$U = \begin{pmatrix} 3.00000000000000 & 1.00000000000000 & 0 & 0 \\ 0 & 3.66666666666667 & 1.00000000000000 & 4.00000000000000 \\ 0 & 0 & 0.72727272727273 & -4.09090909090909 \\ 0 & 0 & 0 & 1.00000000000000 \end{pmatrix}$$

und

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

ehält man durch `P*A-L*U` die Nullmatrix. An diesem Beispiel erkennt man auch, dass P, L, U keineswegs eindeutig bestimmt sind. □

- QR-Zerlegung: Im Package `LinearAlgebra` von Maple gibt es zur QR-Zerlegung die Funktion `QRDecomposition`, in MATLAB die Funktion `qr`. Wieder sollte man sich durch die entsprechenden Fragen über den Gebrauch und die Optionen der Funktionen informieren.

  Beispiel: Sei

$$A := \begin{pmatrix} 1 & -8 \\ 2 & -1 \\ 2 & 14 \end{pmatrix}.$$

  Zu berechnen sei eine QR-Zerlegung. Zunächst verwenden wir Maple:

```
> with(LinearAlgebra):
> A:=Matrix([[1,-8],[2,-1],[2,14]]):
> (Qhat,Rhat):=QRDecomposition(A);
```

$$Qhat,\ Rhat := \begin{bmatrix} \frac{1}{3} & \frac{-2}{3} \\ \frac{2}{3} & \frac{-1}{3} \\ \frac{2}{3} & \frac{2}{3} \end{bmatrix}, \begin{bmatrix} 3 & 6 \\ 0 & 15 \end{bmatrix}$$

```
> Equal(Qhat.Rhat,A);
```

$$true$$

```
> (Q,R):=QRDecomposition(A,fullspan);
```

$$Q,\ R := \begin{bmatrix} \frac{1}{3} & \frac{-2}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{-1}{3} & \frac{-2}{3} \\ \frac{2}{3} & \frac{2}{3} & \frac{1}{3} \end{bmatrix}, \begin{bmatrix} 3 & 6 \\ 0 & 15 \\ 0 & 0 \end{bmatrix}$$

```
> Equal(Q.R,A);
```

$$true$$

Im ersten Fall wird die sogenannte *reduzierte* QR-Zerlegung berechnet, also eine Matrix $\hat{Q} \in \mathbb{R}^{m\times n}$ (in unserem Beispiel ist $m = 3$ und $n = 2$), deren Spalten ein Orthonormalsystem im $\mathbb{R}^m$ bilden, für die also $\hat{Q}^T\hat{Q} = I$, sowie eine obere Dreiecksmatrix $\hat{R} \in \mathbb{R}^{n\times n}$ mit $A = \hat{Q}\hat{R}$. Im zweiten Fall wird die oben beschriebene *volle* QR-Zerlegung berechnet. Allerdings ist dies nicht möglich, wenn A Gleitkommazahlen enthält.

Und nun noch die entsprechenden Resultate mit MATLAB:

```
A=[1,-8;2,-1;2,14];
format long;
[Q,R]=qr(A)
```

liefert eine volle QR-Zerlegung:

$$Q = \begin{pmatrix} -0.33333333333333 & 0.66666666666667 & 0.66666666666667 \\ -0.66666666666667 & 0.33333333333333 & -0.66666666666667 \\ -0.66666666666667 & -0.66666666666667 & 0.33333333333333 \end{pmatrix}$$

und

$$R = \begin{pmatrix} -3 & -6 \\ 0 & -15 \\ 0 & 0 \end{pmatrix}.$$

Eine reduzierte (economy size) QR-Zerlegung erhält man durch

```
[Qhat,Rhat]=qr(A,0)
```

nämlich

$$\hat{Q} = \begin{pmatrix} -0.33333333333333 & 0.66666666666667 \\ -0.66666666666667 & 0.33333333333333 \\ -0.66666666666667 & -0.66666666666667 \end{pmatrix}, \quad \hat{R} = \begin{pmatrix} -3 & -6 \\ 0 & -15 \end{pmatrix}.$$

Es gibt diverse andere Optionen, auf die aber nicht eingegangen werden soll. Übrigens erkennt man, dass auch eine QR-Zerlegung nicht eindeutig bestimmt ist. Dies ist erst dann der Fall, wenn man die Vorzeichen der Diagonalelemente von R vorschreibt. □

- Cholesky-Zerlegung: In Maple kann man mit der Funktion `LUDecomposition` und einer speziellen Option die Cholesky-Zerlegung einer symmetrischen, positiv definiten Matrix bestimmen. In MATLAB gibt es die Funktion `chol`. Wir geben ein Beispiel.

  Beispiel: Sei
  $$A := \begin{pmatrix} 4 & -10 & 2 \\ -10 & 34 & -17 \\ 2 & -17 & 18 \end{pmatrix}.$$
  Zunächst gehen wir auf Maple ein:

  ```
  > with(LinearAlgebra):
  > A:=Matrix([[4,-10,2],[-10,34,-17],[2,-17,18]]):
  > L:=LUDecomposition(A,method=Cholesky);
  ```
  $$L := \begin{bmatrix} 2 & 0 & 0 \\ -5 & 3 & 0 \\ 1 & -4 & 1 \end{bmatrix}$$
  ```
  > Equal(L.Transpose(L),A);
  ```
  $$true$$

  Bei der Anwendung der Funktion `chol` in MATLAB muss man beachten, dass `R=chol(A)` eine *obere* Dreiecksmatrix mit $R^T R = A$ liefert. Durch

  ```
  A=[4,-10,2;-10,34,-17;2,-17,18];format long;
  L=chol(A)'
  ```

  erhält man dieselbe Matrix L wie oben. Das ist auch kein Wunder, denn die Cholesky-Zerlegung ist eindeutig bestimmt. □

2.1.3 LU-, QR- und Cholesky-Zerlegung: Existenz und numerische Berechnung

Sei $A \in \mathbb{R}^{n\times n}$ nichtsingulär. Die Idee des Gaußschen Eliminationsverfahrens ohne Spaltenpivotsuche (welches nicht immer durchführbar ist!) zur Berechnung einer Zerlegung $A = LU$ mit einer unteren Dreiecksmatrix $L \in \mathbb{R}^{n\times n}$ mit Einsen in der Diagonalen und einer oberen Dreiecksmatrix $U \in \mathbb{R}^{n\times n}$ besteht einfach darin, dass A in $n-1$ Schritten durch Multiplikation mit sogenannten Gaußschen Matrizen in eine obere Dreiecksmatrix transformiert wird. Diese wohlbekannte Idee wollen wir an einem Beispiel demonstrieren.

Beispiel: Sei

$$A := \begin{pmatrix} 2 & -1 & 3 \\ -4 & 6 & -5 \\ 6 & 13 & 16 \end{pmatrix}.$$

Multipliziert man A von links mit

$$M_1 := \begin{pmatrix} 1 & 0 & 0 \\ \frac{4}{2} & 1 & 0 \\ -\frac{6}{2} & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -3 & 0 & 1 \end{pmatrix},$$

so erhält man

$$M_1 A = \begin{pmatrix} 2 & -1 & 3 \\ 0 & 4 & 1 \\ 0 & 16 & 7 \end{pmatrix}.$$

Eine Multiplikation von $M_1 A$ von links mit

$$M_2 := \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{16}{4} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -4 & 1 \end{pmatrix}$$

liefert

$$M_2 M_1 A = \begin{pmatrix} 2 & -1 & 3 \\ 0 & 4 & 1 \\ 0 & 0 & 3 \end{pmatrix} =: U,$$

also eine obere Dreiecksmatrix. Daher ist

$$A = M_1^{-1} M_2^{-1} U = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 3 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 4 & 1 \end{pmatrix} U = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 3 & 4 & 1 \end{pmatrix}}_{=L} \underbrace{\begin{pmatrix} 2 & -1 & 3 \\ 0 & 4 & 1 \\ 0 & 0 & 3 \end{pmatrix}}_{=U}$$

die gesuchte LU-Zerlegung von A. □

Nun gehen wir vom speziellen Beispiel zum allgemeinen Fall über und geben im folgenden Satz notwendige und hinreichende Bedingungen dafür an, dass das Gaußsche Eliminationsverfahren ohne Spaltenpivotsuche durchführbar ist, und dass es, wenn es durchführbar ist, eine LU-Zerlegung liefert.

Satz 1.1 *Sei $A = (a_{ij}) \in \mathbb{R}^{n \times n}$. Man betrachte das folgende Verfahren:*

- *Setze $A^{(1)} := A$, $A^{(1)} = (a_{ij}^{(1)})$.*
- *Für $k = 1, \ldots, n-1$:*
  - *Bestimme*

$$l_k := \frac{1}{a_{kk}^{(k)}} (\underbrace{0, \ldots, 0}_{k}, a_{k+1,k}^{(k)}, \ldots, a_{nk}^{(k)})^T$$

und anschließend

$$M_k := I - l_k e_k^T.$$

Hierbei bedeute e_k den k-ten Einheitsvektor im $\mathbb{R}^n$. Die Matrix M_k stimmt außer in der k-ten Spalte, und hier auch nur in den Einträgen unterhalb des Diagonalelements, mit der Einheitsmatrix überein. Danach berechne man

$$A^{(k+1)} := M_k A^{(k)}.$$

Hierbei werden die ersten k Zeilen und ersten $k-1$ Spalten von $A^{(k)}$ nicht verändert. In der k-ten Spalte unterhalb des Diagonalelements werden Nullen erzeugt. Den unteren $(n-k)\times(n-k)$-Block in $A^{(k+1)}$ berechnet man aus

$$a_{ij}^{(k+1)} := a_{ij}^{(k)} - \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} a_{kj}^{(k)}, \qquad i,j = k+1,\ldots,n.$$

Dann gilt:

1. *Das obige Verfahren ist genau dann durchführbar, d. h. es ist $a_{kk}^{(k)} \neq 0$, $k = 1,\ldots,n-1$, wenn die ersten $n-1$ Hauptabschnittsdeterminanten von A nicht verschwinden, wenn also*

$$\det \begin{pmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & & \vdots \\ a_{k1} & \cdots & a_{kk} \end{pmatrix} \neq 0, \qquad k = 1,\ldots,n-1.$$

2. *Ist das Verfahren durchführbar, so ist $U := A^{(n)}$ eine obere Dreiecksmatrix und $A = LU$ mit*

$$L := I + \sum_{k=1}^{n-1} l_k e_k^T,$$

einer unteren Dreiecksmatrix mit Einsen in der Diagonalen.

Beweis: Wegen $A = A^{(1)}$ und $\det(M_k) = 1$ ist $\det(A) = \det(A^{(k)})$, $k = 1,\ldots,n$. Das entsprechende gilt auch für den linken oberen $k\times k$-Block in $A^{(k)}$, d. h. es ist

$$\det \begin{pmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & & \vdots \\ a_{k1} & \cdots & a_{kk} \end{pmatrix} = a_{11}^{(1)} \cdots a_{kk}^{(k)}, \qquad k = 1,\ldots,n.$$

Das Verfahren ist genau dann durchführbar, wenn $a_{kk}^{(k)} \neq 0$, $k = 1,\ldots,n-1$, bzw. die ersten $n-1$ Hauptabschnittsdeterminanten von Null verschieden sind. Damit ist der erste Teil des Satzes bewiesen.

Für den zweiten Teil beachte man, dass die Matrizen M_k nichtsingulär sind und

$$M_k(I + l_k e_k^T) = (I - l_k e_k^T)(I + l_k e_k^T) = I + l_k e_k^T - l_k e_k^T - \underbrace{e_k^T l_k}_{=0} e_k e_k^T = I,$$

also $M_k^{-1} = I + l_k e_k^T$, $k = 1, \ldots, n-1$, gilt. Nach Konstruktion ist $U := A^{(n)}$ eine obere Dreiecksmatrix und

$$A = M_1^{-1} \cdots M_{n-1}^{-1} U = (I + l_1 e_1^T) \cdots (I + l_{n-1} e_{n-1}^T) U = \Big(I + \sum_{k=1}^{n-1} l_k e_k^T\Big) U.$$

Die Gültigkeit von

$$(I + l_1 e_1^T) \cdots (I + l_{n-1} e_{n-1}^T) = \Big(I + \sum_{k=1}^{n-1} l_k e_k^T\Big)$$

erkennt man hierbei folgendermaßen: Beide Matrizen haben dieselbe letzte Spalte, nämlich e_n. Die ersten $n-1$ Spalten stimmen aber auch überein, wie man für $j = 1, \ldots, n-1$ aus

$$\begin{aligned}
(I + l_1 e_1^T) \cdots (I + l_{n-1} e_{n-1}^T) e_j &= (I + l_1 e_1^T) \cdots (I + l_j e_j^T) e_j \\
&= (I + l_1 e_1^T) \cdots (I + l_{j-1} e_{j-1}^T)(e_j + l_j) \\
&= e_j + l_j \\
&= \Big(I + \sum_{k=1}^{n-1} l_k e_k^T\Big) e_j
\end{aligned}$$

erkennt. Offensichtlich ist

$$L := I + \sum_{k=1}^{n-1} l_k e_k^T$$

eine untere Dreiecksmatrix mit Einsen in der Diagonalen. Der Satz ist bewiesen. □

Natürlich gibt es nichtsinguläre Matrizen, bei denen nicht alle Hauptabschnittsdeterminanten von Null verschieden sind. Z. B. kann schon der $(1,1)$-Eintrag verschwinden. Dann ist man auf eine sogenannte Pivotsuche angewiesen, welche aus Stabilitätsgründen sowieso angebracht ist. Die übliche Strategie ist eine *Spaltenpivotsuche*. Im k-ten Schritt bestimmt man $r = r(k) \in \{k, \ldots, n\}$ mit

$$|a_{rk}^{(k)}| = \max_{i=k,\ldots,n} |a_{ik}^{(k)}|$$

und vertauscht in $A^{(k)}$ die k-te und die r-te Zeile. Anschließend multipliziert man die durch (eventuelle) Vertauschung zweier Zeilen gewonnene Matrix mit der Gauß-Matrix $M_k = I - l_k e_k^T$, um Einträge in der k-ten Spalte unterhalb des Diagonalelementes zu annullieren. Die neue Matrix wird $A^{(k+1)}$ genannt. Hier ist also $A^{(k+1)} := M_k P_k A^{(k)}$, wobei P_k eine spezielle Permutationsmatrix ist, nämlich eine Vertauschungsmatrix. Diese gewinnt man aus der Einheitsmatrix, indem man die k-te und die r-te Zeile (oder auch Spalte, denn sie ist symmetrisch) miteinander vertauscht. Es ist klar, dass dieses Verfahren bei nichtsingulärem $A = A^{(1)}$ durchführbar ist und $U := A^{(n)}$ eine obere Dreiecksmatrix ist. Es ist also

$$M_{n-1} P_{n-1} \cdots M_1 P_1 A = U.$$

Da die Vertauschungsmatrizen $P_1, \ldots, P_{n-1}$ symmetrisch und orthogonal sind, kann man die letzte Gleichung auch in der Form

$$M'_{n-1} \cdots M'_1 P_{n-1} \cdots P_1 A = U$$

schreiben, wobei

$$M'_k := P_{n-1} \cdots P_{k+1} M_k P_{k+1} \cdots P_{n-1}, \qquad k = 1, \ldots, n-1.$$

Dann ist aber

$$\begin{aligned} M'_k &= P_{n-1} \cdots P_{k+1}(I - l_k e_k^T) P_{k+1} \cdots P_{n-1} \\ &= I - P_{n-1} \cdots P_{k+1} l_k \underbrace{e_k^T P_{k+1} \cdots P_{n-1}}_{=e_k^T} \\ &= I - l'_k e_k^T \end{aligned}$$

ebenfalls eine Gauß-Matrix mit

$$l'_k := P_{n-1} \cdots P_{k+1} l_k, \qquad k = 1, \ldots, n-1.$$

Mit

$$P := P_{n-1} \cdots P_1$$

ist daher

$$PA = (I + l'_1 e_1^T) \cdots (I + l'_{n-1} e_{n-1}^T) U = \Big(I + \sum_{k=1}^{n-1} l'_k e_k^T\Big) U = LU,$$

wobei

$$L := I + \sum_{k=1}^{n-1} l'_k e_k^T.$$

Dies bedeutet: Macht man im k-ten Schritt bei der Berechnung der Gauß-Matrix M_k die Zuweisung $a_{ik} := a_{ik}/a_{kk}$, $i = k+1, \ldots, n$, schreibt man also die relevanten Daten von M_k in die gerade frei gewordenen Positionen in der k-ten Spalte von A unterhalb des Diagonalelements, so steht nach Abschluss in der unteren Hälfte von A die untere Hälfte von L, in der oberen Hälfte von A (einschließlich der Diagonalen) steht die obere Dreiecksmatrix U. Wir fassen das bisher zum Gaußschen Eliminationsverfahren mit Spaltenpivotsuche gesagte in einer MATLAB-Funktion `GEpiv` zusammen[7].

```
function  [L,U,piv]=GEpiv(A);
%****************************************************************
%Input-Parameter:
%  A    n x n-Matrix
%Output-Parameter:
```

[7] Siehe

C. F. van Loan (1997, S. 214) *Introduction to Scientific Computing. A Matrix-Vector Approach using MATLAB*. Prentice Hall, Upper Saddle River.

```
%  L    n x n-untere Dreiecksmatrix mit Einsen in der
%       Diagonalen
%  U    n x n-obere Dreiecksmatrix
%  piv  ganzzahliger n-Vektor, der eine Permutation
%       von 1:n ist. Es ist A(piv,:)=LU. Ist also I
%       die n x n-Einheitsmatrix und P=I(piv,:), so
%       ist PA=LU.
%************************************************************
   [n,n]=size(A);
   piv=1:n;
   for k=1:n-1
       [maxv,q]=max(abs(A(k:n,k)));
       r=q+k-1;
       piv([k r])=piv([r k]);
       A([k r],:)=A([r k],:);
       if A(k,k)~=0      %Andernfalls ist A singulaer
          A(k+1:n,k)=A(k+1:n,k)/A(k,k);
          A(k+1:n,k+1:n)=A(k+1:n,k+1:n)-A(k+1:n,k)*A(k,k+1:n);
       end
   end
   L=eye(n)+tril(A,-1);
   U=triu(A);
%************************************************************
```

In den Aufgaben 6, 7, 8 und 9 gehen wir auf einige weitere Aussagen zum Gaußschen Eliminationsverfahren ein.

Nun kommen wir zur Berechnung einer QR-Zerlegung einer Matrix $A \in \mathbb{R}^{m\times n}$ mit $m \geq n$. Die Existenz einer (reduzierten) QR-Zerlegung ist aus der linearen Algebra wohlbekannt, auch wenn man das vielleicht selber nicht weiß. Der folgende Satz erinnert an diese (eventuell) verschütteten Vorkenntnisse.

Satz 1.2 *Sei* $A = (\ a_1\ \cdots\ a_n\ ) \in \mathbb{R}^{m\times n}$ *mit* $m \geq n$ *und* $\mathrm{Rang}\,(A) = n$ *gegeben. Die* n *Spalten* $a_1, \ldots, a_n \in \mathbb{R}^m$ *von* A *seien also linear unabhängig. Man betrachte das folgende klassische Gram-Schmidt-Verfahren zur sukzessiven Orthonormierung von* $\{a_1, \ldots, a_n\}$*:*

- *Für* $k = 1, \ldots, n$*:*

  Setze $q'_k := a_k$.

  Für $i = 1, \ldots, k-1$*:*

  Berechne $r_{ik} := q_i^T a_k$.

  Berechne $q'_k := q'_k - r_{ik} q_i$.

  % *Es ist* $q'_k = a_k - \sum_{i=1}^{k-1} r_{ik} q_i$.

  Berechne $r_{kk} := \|q'_k\|_2$ *und anschließend* $q_k := q'_k / r_{kk}$.

Dann gilt:

1. *Es ist* $\{q_1, \ldots, q_n\}$ *ein Orthonormalsystem im* $\mathbb{R}^m$ *mit*

$$\mathrm{span}\,\{q_1, \ldots, q_k\} = \mathrm{span}\,\{a_1, \ldots, a_k\}, \qquad k = 1, \ldots, n.$$

und

$$a_k = \sum_{i=1}^{k} r_{ik} q_i, \qquad k = 1, \ldots, n.$$

2. *Definiert man* $\hat{Q} := (\ q_1 \ \cdots \ q_n\ ) \in \mathbb{R}^{m \times n}$ *und die obere Dreiecksmatrix* $\hat{R} \in \mathbb{R}^{n \times n}$ *mit* r_{ik} *für* $i \leq k$ *als* (i,k)*-Eintrag, so ist* $\hat{Q}^T \hat{Q} = I$ *und* $A = \hat{Q}\hat{R}$*, d. h. durch das Gram-Schmidt-Verfahren wird eine reduzierte* QR*-Zerlegung von* A *berechnet.*

Beweis: Durch vollständige Induktion nach k kann man leicht zeigen, dass $\{q_1, \ldots, q_k\}$ ein Orthonormalsystem mit $\operatorname{span}\{q_1, \ldots, q_k\} = \operatorname{span}\{a_1, \ldots, a_k\}$ und $a_k = \sum_{i=1}^{k} r_{ik} q_i$ ist. Hieraus folgt sofort der zweite Teil des Satzes. Dies erkennt man sehr einfach, wenn man auf beiden Seiten der behaupteten Gleichung $A = \hat{Q}\hat{R}$ die k-te Spalte betrachtet. Denn es ist

$$\hat{Q}\hat{R}e_k = (\ q_1 \ \cdots \ q_k \ \ q_{k+1} \ \cdots \ q_n\ ) \begin{pmatrix} r_{1k} \\ \vdots \\ r_{kk} \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \sum_{i=1}^{k} r_{ik} q_i = a_k = A e_k.$$

Damit ist der Satz bewiesen. □

Das klassische Gram-Schmidt-Verfahren ist für die Praxis unbrauchbar, da die Orthonormalität der Spalten von $\hat{Q}$ wegen Rundungsfehlern sukzessive verloren geht. Außerdem erhält man nur eine reduzierte QR-Zerlegung und nicht eine volle. Dies ist theoretisch zwar keine Einschränkung (man ergänze $\{q_1, \ldots, q_n\}$ zu einer Orthonormalbasis $\{q_1, \ldots, q_n, q_{n+1}, \ldots, q_m\}$ des $\mathbb{R}^m$), für die Praxis aber schon. In der Praxis wird A durch sukzessive Multiplikation von links mit gewissen orthogonalen Matrizen $Q_1, \ldots, Q_p$ in eine obere Dreiecksmatrix R überführt:

$$Q_p \cdots Q_1 A = R.$$

Mit $Q := Q_1^T \cdots Q_p^T$ ist dann die gesuchte (volle) QR-Zerlegung gefunden. Im wesentlichen spielen bei diesen Transformationen zwei Klassen spezieller orthogonaler Matrizen eine Rolle, nämlich *Householder-Spiegelungen* und *Givens-Rotationen*. Wir beschränken uns auf die Beschreibung eines Verfahrens, welches Givens-Rotationen benutzt. Dies sind Matrizen der Form

$$G_{ik} = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix} \begin{matrix} \\ \\ i \\ \\ k \\ \\ \\ \end{matrix}$$

$$\begin{matrix} & i & & k \end{matrix}$$

wobei $1 \leq i < k \leq m$ und $c^2 + s^2 = 1$. Es ist leicht nachzuweisen, dass eine Givens-Rotation orthogonal ist. Denn offensichtlich bilden die Spalten einer Givens-Rotation ein Orthonormalsystem, d. h. sie haben die euklidische Länge 1 und stehen paarweise aufeinander senkrecht. Eine 2×2-Givens-Rotation hat die Form

$$G = \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \qquad \text{mit} \quad c^2 + s^2 = 1.$$

Bei gegebenem $x = (x_1, x_2)^T \in \mathbb{R}^2 \setminus \{0\}$ können (c, s) mit $c^2 + s^2 = 1$ so bestimmt werden, dass die zweite Komponente von $y := Gx$ verschwindet. Dies führt nämlich auf die Forderung

$$y_2 := -sx_1 + cx_2 = 0,$$

so dass

$$c := \frac{x_1}{\sqrt{x_1^2 + x_2^2}}, \qquad s := \frac{x_2}{\sqrt{x_1^2 + s^2}}$$

das Verlangte tut. In MATLAB gibt es für diese Aufgabe die Funktion `planerot`. Nach `help planerot` erfährt man:

```
PLANEROT Givens plane rotation.
   [G,Y] = PLANEROT(X), where X is a 2-component column vector,
   returns a 2-by-2 orthogonal matrix G so that Y = G*X has Y(2) = 0.
```

Bei der sukzessiven Annullierung der Einträge von A unterhalb der Diagonalen kann man z. B. spaltenweise (von links nach rechts) und in jeder Spalte von unten nach oben vorgehen. Hierbei beachte man, dass sich bei der Multiplikation einer Matrix von links mit einer Givens-Rotation G_{ik} nur die i-te und die k-te Zeile verändern. Die neuen Zeilen sind Linearkombinationen der alten. Wir machen uns die Vorgehensweise bei einer 4×3-Matrix klar. Bei einer Transformation festbleibende Elemente werden mit $\bullet$, sich verändernde mit $*$ bezeichnet. Nullen sind durch ein blank gekennzeichnet:

$$\begin{pmatrix} \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{pmatrix} \xrightarrow{G_{34}} \begin{pmatrix} \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ * & * & * \\ & * & * \end{pmatrix} \xrightarrow{G_{23}} \begin{pmatrix} \bullet & \bullet & \bullet \\ * & * & * \\ & * & * \\ & \bullet & \bullet \end{pmatrix} \xrightarrow{G_{12}} \begin{pmatrix} * & * & * \\ & * & * \\ & \bullet & \bullet \\ & \bullet & \bullet \end{pmatrix} \xrightarrow{G_{34}} \begin{pmatrix} \bullet & \bullet & \bullet \\ & \bullet & \bullet \\ & * & * \\ & & * \end{pmatrix}$$

und anschließend

$$\xrightarrow{G_{23}} \begin{pmatrix} \bullet & \bullet & \bullet \\ & * & * \\ & & * \\ & & \bullet \end{pmatrix} \xrightarrow{G_{34}} \begin{pmatrix} \bullet & \bullet & \bullet \\ & \bullet & \bullet \\ & & * \\ & & \end{pmatrix}.$$

Wir geben nun eine MATLAB-Funktion `QRGivens` an, die mit diesen Ideen eine volle QR-Zerlegung einer Matrix $A \in \mathbb{R}^{m \times n}$ mit $m \geq n$ berechnet. Man beachte, dass die Matrix A nicht vollen Rang zu haben braucht. Der obere $n \times n$-Block in der oberen Dreiecksmatrix R ist allerdings genau dann nichtsingulär, wenn A vollen Rang besitzt.

```
function [Q,R]=QRGivens(A);
%************************************************
%Input-Parameter:
%   A    m x n-Matrix mit m>=n.
%Output-Parameter:
%   Q    m x m orthogonale Matrix
%   R    m x n obere Dreiecksmatrix
%        A = QR
%************************************************
  [m,n]=size(A);
  Q=eye(m);
  for j=1:n
     for i=m:-1:j+1
         %Annulliere A(i,j)
         [G,A(i-1:i,j)]=planerot(A(i-1:i,j));
         A(i-1:i,j+1:n)=G*A(i-1:i,j+1:n);
         Q(:,i-1:i)=Q(:,i-1:i)*G';
     end
  end
  R=triu(A);
%************************************************
```

Beispiel: Wie in einem früheren Beispiel sei

$$A := \begin{pmatrix} 1 & -8 \\ 2 & -1 \\ 2 & 14 \end{pmatrix}.$$

Mit der eben angegebenen Funktion `QRGivens` erhalten wir in MATLAB nach

```
A=[1,-8;2,-1;2,14];format long;
[Q,R]=QRGivens(A)
```

das Ergebnis

$$Q = \begin{pmatrix} 0.33333333333333 & -0.66666666666667 & 0.66666666666667 \\ 0.66666666666667 & -0.33333333333333 & -0.66666666666667 \\ 0.66666666666667 & 0.66666666666667 & 0.33333333333333 \end{pmatrix}, \qquad R = \begin{pmatrix} 3 & 6 \\ 0 & 15 \\ 0 & 0 \end{pmatrix}.$$

Bis auf eine unterschiedliche Vorzeichenverteilung ist das genau das durch die MATLAB-Funktion `qr` erhaltene Ergebnis. □

Jetzt kommen wir noch zur Cholesky-Zerlegung einer symmetrischen, positiv definiten Matrix $A \in \mathbb{R}^{n\times n}$, also einer Darstellung $A = LL^T$ mit einer unteren Dreiecksmatrix $L \in \mathbb{R}^{n\times n}$, die nur positive Diagonalelemente besitzt. Der folgende Satz sagt aus, dass jede symmetrische, positiv definite Matrix eine eindeutige Cholesky-Zerlegung besitzt.

Satz 1.3 *Ist $A \in \mathbb{R}^{n\times n}$ symmetrisch und positiv definit, so existiert genau eine untere Dreiecksmatrix $L \in \mathbb{R}^{n\times n}$ mit positiven Diagonalelementen derart, dass $A = LL^T$.*

Beweis: Die Behauptung wird durch vollständige Induktion nach n bewiesen. Für $n = 1$ ist die Aussage offenbar richtig. Wir nehmen an, dass jede symmetrische, positiv

definite $(n-1)\times(n-1)$-Matrix eine eindeutige Cholesky-Zerlegung besitzt. Die $n\times n$-Matrix denke man sich zerlegt:

$$A = \begin{pmatrix} A_{n-1} & a \\ a^T & \alpha \end{pmatrix}.$$

Hierbei ist $A_{n-1} \in \mathbb{R}^{(n-1)\times(n-1)}$ symmetrisch und positiv definit (Beweis?). Für die untere Dreiecksmatrix L mache man den Ansatz

$$L = \begin{pmatrix} L_{n-1} & 0 \\ l^T & \beta \end{pmatrix}$$

mit einer unteren Dreiecksmatrix $L_{n-1} \in \mathbb{R}^{(n-1)\times(n-1)}$ und einem Vektor $l \in \mathbb{R}^{n-1}$. Dann ist

$$LL^T = \begin{pmatrix} L_{n-1} & 0 \\ l^T & \beta \end{pmatrix} \begin{pmatrix} L_{n-1}^T & l \\ 0^T & \beta \end{pmatrix} = \begin{pmatrix} L_{n-1}L_{n-1}^T & L_{n-1}l \\ (L_{n-1}l)^T & l^Tl+\beta^2 \end{pmatrix} = \begin{pmatrix} A_{n-1} & a \\ a^T & \alpha \end{pmatrix} = A$$

genau dann, wenn

$$L_{n-1}L_{n-1}^T = A_{n-1}, \qquad L_{n-1}l = a, \qquad l^Tl+\beta^2 = \alpha.$$

Also ist L_{n-1} notwendigerweise der nach Induktionsannahme eindeutig existierende Cholesky-Faktor (untere Dreiecksmatrix mit positiven Diagonalelementen) von A_{n-1} und es bleibt die eindeutige Existenz eines Vektors $l \in \mathbb{R}^{n-1}$ und einer positiven Zahl β mit

$$L_{n-1}l = a, \qquad l^Tl+\beta^2 = \alpha$$

zu zeigen. Da L_{n-1} nichtsingulär ist, ist l durch $L_{n-1}l = a$ eindeutig festgelegt. Zu zeigen bleibt, dass $\alpha - l^Tl$ positiv ist, da dann genau ein positives β mit $l^Tl+\beta^2 = \alpha$ existiert. Nun ist aber

$$\begin{aligned} \alpha - l^Tl &= \alpha - a^TL_{n-1}^{-T}L_{n-1}^{-1}a \\ &= \alpha - a^TA_{n-1}^{-1}a \\ &= \begin{pmatrix} -A_{n-1}^{-1}a \\ 1 \end{pmatrix}^T \begin{pmatrix} A_{n-1} & a \\ a^T & \alpha \end{pmatrix} \begin{pmatrix} -A_{n-1}^{-1}a \\ 1 \end{pmatrix} \\ &> 0 \quad , \end{aligned}$$

da A positiv definit ist, womit alles gezeigt ist. □

Ein Verfahren zur Berechnung des Cholesky-Faktors L erhält man sehr leicht aus der Bestimmungsgleichung $A = LL^T$ durch Koeffizientenvergleich. Wegen der Symmetrie genügt es, die unteren Hälften zu vergleichen. Für $i \geq j$ ist

$$a_{ij} = (A)_{ij} = (LL^T)_{ij} = \sum_{k=1}^{j} l_{ik}l_{jk} = \sum_{k=1}^{j-1} l_{ik}l_{jk} + l_{ij}l_{jj}.$$

Dies ergibt:

$$i = j: \qquad a_{jj} = l_{jj}^2 + \sum_{k=1}^{j-1} l_{jk}^2 \qquad \text{bzw.} \qquad l_{jj} := \Big(a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2\Big)^{1/2},$$

$$i > j: \qquad a_{ij} = l_{ij} l_{jj} + \sum_{k=1}^{j-1} l_{ik} l_{jk} \qquad \text{bzw.} \qquad l_{ij} := \Big(a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}\Big) \Big/ l_{jj}.$$

Wegen dieser Gleichungen kann man sukzessive spalten- oder zeilenweise die Matrix L berechnen. So lautet z. B. eine Spaltenversion des resultierenden *Cholesky-Verfahrens*:

- Input: Gegeben sei die symmetrische, positiv definite Matrix $A = (a_{ij}) \in \mathbb{R}^{n \times n}$. Benutzt wird nur die untere Hälfte von A, d. h. Elemente a_{ij} mit $i \geq j$.

- Für $j = 1, \ldots, n$:

  $l_{jj} := (a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2)^{1/2}$.

  Für $i = j+1, \ldots, n$:

  $l_{ij} := (a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}) / l_{jj}$.

- Output: Nur die untere Hälfte von L (einschließlich der Diagonalen) ist definiert.

Bevor das Pivotelement durch Wurzelziehen gebildet wird, prüft man natürlich, ob $a_{kk} - \sum_{j=1}^{k-1} a_{kj}^2 \geq \epsilon$ ist, wobei $\epsilon > 0$ eine vorgegebene kleine Zahl ist. Eine Umsetzung[8] in eine MATLAB-Funktion ist die folgende (wobei wir auf den oben angegebenen Test verzichten):

```
function L=Chol(A);
%*****************************************************
%Input-Parameter:
%     A    n x n symmerische, positiv definite Matrix
%Output-Parameter:
%     L    n x n untere Dreiecksmatrix mit positiven
%          Diagonalelementen und A=L*L'.
%*****************************************************
  [n,n]=size(A);
  L=zeros(n); s=zeros(n,1);
  for j=1:n
     s(j:n)=A(j:n,j);
     for k=1:j-1
         s(j:n)=s(j:n)-L(j:n,k)*L(j,k);
     end
     % s(i)=A(i,j)-sum_1^(j-1)L(i,k)L(j,k), i=j,...,n
     L(j:n,j)=s(j:n)/sqrt(s(j));
  end
```

[8] Siehe C, F. VAN LOAN (1997, S. 245).

2.1.4 Vorwärts- und Rückwärteinsetzen

Zum Schluss des Abschnitts über lineare Gleichungssysteme wollen wir noch ganz kurz auf lineare Gleichungssysteme eingehen, bei denen die Koeffizientenmatrix eine untere bzw. obere Dreiecksmatrix ist, also auf das sogenannte Vorwärts- bzw. Rückwärtseinsetzen. Wegen der Einfachheit des Stoffes können wir uns ganz kurz fassen.

Gegeben sei ein lineares Gleichungssystem $Lx = b$ mit der nichtsingulären unteren Dreiecksmatrix $L \in \mathbb{R}^{n\times n}$ und der rechten Seite $b \in \mathbb{R}^n$. Die Lösung erhält man natürlich durch:

- $x_1 := b_1/l_{11}$.
- Für $i = 2, \ldots, n$:
  $x_i := (b_i - \sum_{j=1}^{i-1} l_{ij}x_j)/l_{ii}$.

Eine Umsetzung in eine MATLAB-Funktion ist einfach:

```
function x=Lsol(L,b);
%*****************************************************
%Input-Parameter:
%     L    n x n nichtsingulaere untere Dreiecksmatrix
%     b    n-Spaltenvektor
%Output-Parameter:
%     x    n-Spaltenvektor mit Lx=b
%*****************************************************
  n=length(b);  x=zeros(n,1);
  x(1)=b(1)/L(1,1);
  for i=2:n
      x(i)=(b(i)-L(i,1:i-1)*x(1:i-1))/L(i,i);
  end
```

Ist ein lineares Gleichungssystem $Ux = b$ mit der nichtsingulären oberen Dreiecksmatrix $U \in \mathbb{R}^{n\times n}$ und der rechten Seite $b \in \mathbb{R}^n$ gegeben, so erhält man die Lösung natürlich aus

- $x_n := b_n/u_{nn}$.
- Für $i = n-1, \ldots, 1$:
  $x_i := (b_i - \sum_{j=i+1}^{n} u_{ij}x_j)/u_{ii}$.

Eine entsprechende MATLAB-Funktion ist

```
function x=Usol(U,b);
%*****************************************************
%Input-Parameter:
%     U    n x n nichtsingulaere obere Dreiecksmatrix
%     b    n-Spaltenvektor
%Output-Parameter:
%     x    n-Spaltenvektor mit Ux=b
%*****************************************************
  n=length(b);  x=zeros(n,1);
```

```
  x(n)=b(n)/U(n,n);
  for i=n-1:-1:1
      x(i)=(b(i)-U(i,i+1:n)*x(i+1:n))/U(i,i);
  end
```

Jetzt fügen wir noch alles zusammen, um eine Funktion zur Lösung eines linearen Gleichungssystems bzw. eines linearen Least Squares Problems zu erhalten.

```
function x=LinGl(A,b);
%************************************************************
%Input-Parameter:
%    A    n x n nichtsingulaere Matrix
%    b    n-Spaltenvektor
%Output-Parameter:
%    x    Ax=b
%Es werden GEpiv, Lsol und Usol benutzt.
%************************************************************
   [L,U,piv]=GEpiv(A);
   y=Lsol(L,b(piv));
   x=Usol(U,y);
%************************************************************
```

Jetzt noch eine Funktion zur Lösung eines linearen Least Squares Problems. Es handelt sich hier also um die Aufgabe, bei gegebenen $A \in \mathbb{R}^{m \times n}$ mit $m \geq n$ und $b \in \mathbb{R}^m$ den Defekt $\|Ax - b\|_2$ bezüglich der euklidischen Norm zu minimieren. Wenn A vollen Rang hat, ist diese Aufgabe eindeutig lösbar.

```
function x=LSq(A,b);
%************************************************************
%Input-Parameter:
%    A    m x n-Matrix mit m>=n und rang(A)=n
%    b    m-Spaltenvektor
%Output-Parameter:
%    x    Die Loesung des linearen Least Aquares Problems
%         Minimiere ||Ax-b||_2.
%Es werden QRGivens und Usol benutzt.
%************************************************************
   [m,n]=size(A);
   [Q,R]=QRGivens(A);
   b=Q'*b; c=b(1:n); Rhat=R(1:n,:);
   x=Usol(Rhat,c);
%************************************************************
```

Beispiel: In einem See wird der Sauerstoffgehalt s (gemessen in mg/l) in Abhängigkeit von der Tiefe t (gemessen in m) festgestellt. Gemessen werden die Werte

i	t_i	s_i
1	15.0	6.50
2	20.0	5.60
3	30.0	5.40
4	40.0	6.00
5	50.0	4.60
6	60.0	1.40
7	70.0	0.10

Es wird ein linearer Zusammenhang $s = x_1 + x_2 t$ zwischen Tiefe und Sauerstoffgehalt vermutet und nach den "besten" Parametern x_1, x_2 gefragt. Diese werden nach der Methode der kleinsten Quadrate als Lösung der Aufgabe

$$\text{Minimiere} \quad \Big\{ \sum_{i=1}^{7} [(x_1 + x_2 t_i) - s_i]^2 \Big\}^{1/2}$$

bestimmt. Dies ist ein lineares Least Squares Problem mit den Daten

$$A := \begin{pmatrix} 1 & 15 \\ 1 & 20 \\ 1 & 30 \\ 1 & 40 \\ 1 & 50 \\ 1 & 60 \\ 1 & 70 \end{pmatrix}, \qquad b := \begin{pmatrix} 6.50 \\ 5.60 \\ 5.40 \\ 6.00 \\ 4.60 \\ 1.40 \\ 0.10 \end{pmatrix}.$$

Anwendung der obigen Funktion `LSq` liefert (nach `format long`) das Ergebnis

$$x = \begin{pmatrix} 8.63101983002833 \\ -0.10813031161473 \end{pmatrix}.$$

Natürlich können wir diese Aufgabe auch mit Maple lösen. Man erhält

```
> with(LinearAlgebra):
> A:=Matrix([[1,15],[1,20],[1,30],[1,40],[1,50],[1,60],[1,70]]):
> b:=<6.5,5.6,5.4,6.0,4.6,1.4,0.1>:
> x:=LeastSquares(A,b);
```

$$x := \begin{bmatrix} 8.63101983002832895 \\ -.108130311614730870 \end{bmatrix}$$

Wir wollen uns das Ergebnis veranschaulichen. In Abbildung 2.3 links haben wir die Least Squares Approximation der Daten durch eine Gerade, rechts durch eine Parabel dargestellt. Man beachte, dass es wegen notwendiger Beobachtungsfehler keinen Sinn machen würde, durch die (sieben) Datenpaare (t, s) ein Polynom sechsten Grades zu legen, auch wenn man hierdurch eine wesentlich bessere Übereinstimmung mit den Beobachtungen erhalten würde. Bei einem "richtigen" Modell haben die Parameter, die aus den Beobachtungen und durch die Methode der kleinsten Quadrate zu bestimmen sind, einen "interpretierbaren" Sinn, und das wäre bei einem Polynom sechsten Grades in der Monomdarstellung i. Allg. nicht der Fall. Trotzdem geben wir eine Darstellung des interpolierenden Polynoms in Abbildung 2.4 an. □

2.1.5 Aufgaben

1. Jeannot[9] spielt mit einer Waage, deren zwei Schalen sich im Gleichgewicht befinden, wenn man auf die eine ein Gewicht von 100 g und auf die andere zwei gleiche Schlüs-

[9] Diese Aufgabe haben wir wörtlich

J. C. Baillif (1985, S. 11) *Denkpirouetten. Spiele aus Logik und Mathematik*. Hugendubel, München entnommen.

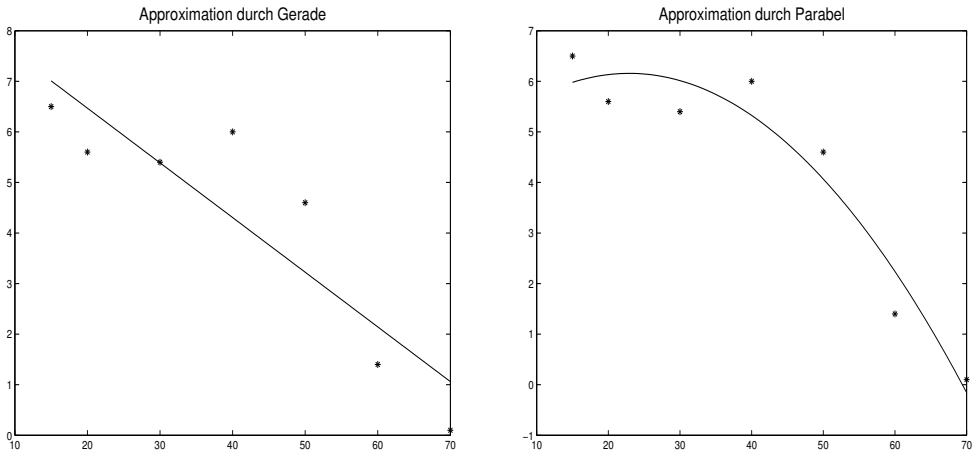

Abbildung 2.3: Least Squares Approximation

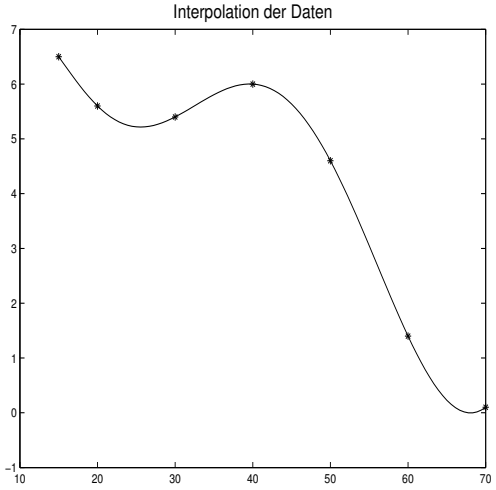

Abbildung 2.4: Interpolation der Daten

sel, zwei gleiche Münzen und drei gleiche Spielsoldaten, oder aber einen Apfel, einen Soldaten und eine Aprikose legt.

Eine Münze, ein Schlüssel, ein Soldat und eine Pflaume wiegen zusammen 50 g.

Die Aprikose, der Apfel und die Pflaume wiegen genausoviel wie eine Münze, ein Schlüssel, ein Soldat und die Feder des Weckers.

Wieviel wiegt die Feder des Weckers?

2. Vor dem Pokerspiel entspricht die Summe, über die Paul verfügt, plus zweimal das, was André und Bernard besitzen, dem Zweifachen dessen, was Claude hat, plus dem Dreifachen dessen, über das Jean verfügt, und nochmals 300 Francs.

   Wenn André 1500 Francs mehr hätte, hätte er genausoviel wie Jean und Bernard, zuzüglich dem Zweifachen dessen, worüber Paul verfügt.

   Wenn Claude 1100 Francs mehr hätte, würde er soviel besitzen wie alle anderen vier Spieler zusammen.

Das Dreifache dessen, was André hat, das Vierfache von dem, über das Jean verfügt, und weitere 1200 Francs machen das Dreifache von dem aus, was Bernard und Paul besitzen.

Wieviel haben Jean und Paul zusammen?

3. Seien reelle Zahlen α, β gegeben. Man zeige, dass sich jedes $p \in \mathcal{P}_3$, also jedes kubische Polynom, eindeutig darstellen lässt in der Form

$$p(x) := a + b(x-\alpha) + c(x-\alpha)^2 + d(x-\alpha)^2(x-\beta),$$

dass also durch $\{v_0, v_1, v_2, v_3\}$ mit

$$v_0(x) := 1, \quad v_1(x) := x-\alpha, \quad v_2(x) := (x-\alpha)^2, \quad v_3(x) := (x-\alpha)^2(x-\beta),$$

eine Basis von $\mathcal{P}_3$ gegeben ist.

4. Sei mit Δ_n die Zerlegung $x_0 < \cdots < x_n$ von $[x_0, x_n]$ in n Teilintervalle $[x_j, x_{j+1}]$, $j = 0, \ldots, n-1$, bezeichnet. Gegeben seien $f_0, \ldots, f_n$ und $f'_0, \ldots, f'_n$. Man zeige, dass es genau ein $s \in C^1[x_0, x_n]$ mit $s|_{[x_j, x_{j+1}]} \in \mathcal{P}_3$, $j = 0, \ldots, n-1$, und $s(x_j) = f_j$, $s'(x_j) = f'_j$, $j = 0, \ldots, n$, gibt und bestimme es.

   Hinweis: Man berücksichtige Aufgabe 3 und mache für die Restriktion von s auf das Intervall $[x_j, x_{j+1}]$ den Ansatz

$$s|_{[x_j, x_{j+1}]}(x) = a_j + b_j(x-x_j) + c_j(x-x_j)^2 + d_j(x-x_j)^2(x-x_{j+1})$$

   mit noch zu bestimmenden a_j, b_j, c_j, d_j, $j = 0, \ldots, n-1$.

5. In Aufgabe 4 konnte gezeigt werden, dass es zu einer gegebenen Zerlegung Δ_n des Intervalls $[x_0, x_n]$ in n Teilintervalle $[x_j, x_{j+1}]$, $j = 0, \ldots, n-1$, und gegebenen reellen Zahlen $f_0, \ldots, f_n$ sowie $f'_0, \ldots, f'_n$ genau ein $s \in C^1[x_0, x_n]$ mit $s|_{[x_j, x_{j+1}]} \in \mathcal{P}_3$, $j = 0, \ldots, n-1$, und $s(x_j) = f_j$, $s'(x_j) = f'_j$, $j = 0, \ldots, n$, gibt.

   (a) Welchen Bedingungen müssen $f'_0, \ldots, f'_n$ genügen, damit s sogar zweimal stetig differenzierbar ist, also $s \in S_3(\Delta_n)$ ein kubischer Spline zur Zerlegung Δ_n ist?

   (b) Man zeige, dass $f'_0, \ldots, f'_n$ eindeutig dadurch bestimmt sind, dass $s \in S_3(\Delta_n)$ der

   - Hermiteschen Randbedingung (hier sind f'_0 und f'_n gegeben),
   - natürlichen Randbedingung (hier sind f''_0 und f''_n gegeben, die Forderung ist $s''(x_0) = f''_0$ und $s''(x_n) = f''_n$),
   - not-a-knot Bedingung (die Zusatzforderung ist, dass der kubische Spline s in x_1 und x_{n-1} dreimal stetig differenzierbar ist)

   genügt.

   Hinweis: Für den letzten Teil stelle man jeweils zur Bestimmung von $f'_1, \ldots, f'_{n-1}$ ein lineares Gleichungssystem mit einer symmetrischen $(n-1)\times(n-1)$-Koeffizientenmatrix auf. Die Nichtsingularität der Koeffizientenmatrix folgt dann aus dem folgenden Resultat (siehe z. B. J. Werner (1992, S. 170)[10]:

[10] J. Werner (1992) *Numerische Mathematik 1*. Vieweg, Braunschweig-Wiesbaden.

- *Sei $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ eine symmetrische Matrix mit positiven Diagonalelementen. Ist A strikt diagonal dominant, d. h. gilt*

$$\sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}| < a_{ii}, \qquad i = 1, \ldots, n,$$

so ist A positiv definit. Genauer gilt: Ist λ ein Eigenwert von A, so ist

$$\min_{i=1,\ldots,n}\left(a_{ii} - \sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}|\right) \leq \lambda \leq \max_{i=1,\ldots,n}\left(a_{ii} + \sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}|\right).$$

Man sollte dieses Ergebnis wenigstens anwenden können. Wer (mit Anstrengung) versucht, es zu beweisen, ist gut, wer dabei erfolgreich ist, sehr gut.

6. Sei $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ strikt zeilenweise diagonal dominant, d. h. es ist

$$\sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}| < |a_{ii}|, \qquad i = 1, \ldots, n.$$

Man zeige:

(a) A ist nichtsingulär.

(b) Alle Hauptabschnittsdeterminanten von A sind von Null verschieden, so dass nach Satz 1.1 das Gaußsche Eliminationsverfahren ohne Spaltenpivotsuche durchführbar ist und eine Darstellung $A = LU$ mit einer unteren Dreiecksmatrix L mit Einsen in der Diagonalen und einer oberen Dreiecksmatrix U liefert.

7. Sei $A \in \mathbb{R}^{n\times n}$ strikt spaltenweise diagonal dominant, d. h. es sei

$$\sum_{\substack{i=1\\ i\neq j}}^{n} |a_{ij}| < |a_{jj}|, \qquad j = 1, \ldots, n.$$

Man zeige:

(a) Die Matrix A ist nichtsingulär.

(b) Die Matrix A besitzt nicht nur eine LU-Zerlegung der Form $A = LU$, sondern mehr noch: Das Gaußsche Eliminationsverfahren mit Spaltenpivotsuche benutzt keine Vertauschungen.

8. Die nichtsinguläre Matrix $A \in \mathbb{R}^{n\times n}$ besitze eine Darstellung $A = LU$ mit einer unteren Dreiecksmatrix $L \in \mathbb{R}^{n\times n}$ mit Einsen in der Diagonalen und einer oberen Dreiecksmatrix $U \in \mathbb{R}^{n\times n}$. Man zeige, dass L und U hierdurch eindeutig bestimmt sind.

9. Wieviele Multiplikationen/Divisionen werden bei der Anwendung des Gaußschen Eliminationsverfahrens auf eine Matrix $A \in \mathbb{R}^{n\times n}$ gemacht?

10. Gegeben sei die Matrix

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 3 & 6 & 10 & 15 & 21 \\ 1 & 4 & 10 & 20 & 35 & 56 \\ 1 & 5 & 15 & 35 & 70 & 126 \\ 1 & 6 & 21 & 56 & 126 & 252 \end{pmatrix}.$$

    (a) Welche Matrix würden Sie erhalten, wenn sie die 6×6-Matrix A um eine Zeile und eine Spalte zu einer 7×7-Matrix erweitern müssten?

    (b) Mit Hilfe des Gaußschen Eliminationsverfahrens mit Spaltenpivotsuche berechne man eine LU-Zerlegung von A. Hierzu sollte man Funktionen aus Maple, MATLAB und ein selbst (z. B. in MATLAB) geschriebenes Programm benutzen.

11. Sogenannte Householder-Matrizen bilden die Grundlage des Verfahrens von Householder zur Berechnung einer QR-Zerlegung einer Matrix $A \in \mathbb{R}^{m\times n}$ mit $m \geq n$. Hierbei heißt eine Matrix der Form

$$Q := I - \frac{2}{u^T u} u u^T$$

mit $u \in \mathbb{R}^m \setminus \{0\}$ eine *Householder-Matrix*. Man zeige:

    (a) Eine Householder-Matrix ist symmetrisch und orthogonal.

    (b) Ist $a \in \mathbb{R}^m \setminus \{0\}$ und definiert man $u := a + \operatorname{sign}(a_1)\|a\|_2\, e_1$, so ist durch

$$Q := I - \frac{2}{u^T u} u u^T = I - \beta u u^T \qquad \text{mit} \quad \beta := \frac{2}{u^T u} = \frac{1}{\|a\|_2(\|a\|_2 + |a_1|)}$$

    eine Householder-Matrix gegeben, die a in ein Vielfaches des ersten Einheitsvektors überführt. Genauer ist $Qa = -\operatorname{sign}(a_1)\|a\|_2\, e_1$. Hierbei sei $\operatorname{sign}(0) := 1$.

12. Sei $A \in \mathbb{R}^{m\times n}$ mit $m \geq n$ und $\operatorname{Rang}(A) = n$. Sei $A = \hat{Q}\hat{R} = Q^* R^*$ jeweils eine reduzierte QR-Zerlegung von A (die Spalten der $m \times n$-Matrizen $\hat{Q}, Q^*$ bilden also ein Orthonormalsystem, die $n \times n$-Matrizen sind obere Dreiecksmatrizen). Man beachte, dass $\hat{R}, R^*$ wegen der Rangvoraussetzung an A nichtsingulär sind, die Diagonalelemente also von Null verschieden sind. Man zeige: Ist $\operatorname{sign}(\hat{r}_{jj}) = \operatorname{sign}(r^*_{jj})$, $j = 1, \ldots, n$, so ist $\hat{Q} = Q^*$ und $\hat{R} = R^*$.

13. Sei $A = \begin{pmatrix} a_1 & \cdots & a_n \end{pmatrix} \in \mathbb{R}^{n\times n}$. Man beweise die Hadamardsche Determinantenungleichung

$$|\det(A)| \leq \prod_{i=1}^{n} \|a_i\|_2.$$

    Hinweis: Man benutze eine QR-Zerlegung von A und zeige, dass $\|a_i\|_2^2 \geq r_{ii}^2$, $i = 1, \ldots, n$.

14. Sei

$$A := \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 2 & 3 & 4 \end{pmatrix}.$$

Man benutze Maple, MATLAB und eine (z. B. in MATLAB) selbst geschriebene Funktion zur Berechnung einer (vollen) QR-Zerlegung von A.

15. Ist $A \in \mathbb{R}^{n\times n}$ symmetrisch, so ist

$$\lambda_{\min}(A)\,\|x\|_2^2 \le x^T A x \le \lambda_{\max}(A)\,\|x\|_2^2 \qquad \text{für alle } x \in \mathbb{R}^n,$$

wobei $\lambda_{\min}(A)$ den kleinsten und $\lambda_{\max}(A)$ den größten Eigenwert von A bedeutet.

Hinweis: Man kann benutzen, dass eine symmetrische Matrix durch eine Ähnlichkeitstransformation mit einer orthogonalen Matrix auf Diagonalgestalt transformiert werden kann bzw., äquivalent dazu, ein vollständiges Orthonormalystem von Eigenvektoren existiert.

16. Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch und positiv semidefinit. Man zeige, dass es positive Konstanten c_0, C_0 mit

$$c_0\,\|Ax\|_2^2 \le x^T A x \le C_0\,\|Ax\|_2^2 \qquad \text{für alle } x \in \mathbb{R}^n$$

gibt. Insbesondere gilt: Ist $A \in \mathbb{R}^{n\times n}$ symmetrisch und positiv semidefinit, so folgt aus $x^T Ax = 0$, dass $Ax = 0$.

2.2 Nichtlineare Gleichungen und Gleichungssysteme

2.2.1 Beispiele

Beispiel: In G. Schwabs Sagen des klassischen Altertums kann man nachlesen:

"So gelangten sie (d. h. Dido und ihre Gefährten, die nach der Ermordung ihres Gatten Sychäus durch ihren Bruder Pygmalion Tyrus verlassen mussten) an die Küste Afrikas und an den Ort, wo du (d. h. Äneas) jetzt bald die gewaltigen Mauern der neuen Stadt Karthago und ihre himmelansteigende Burg erblicken wirst. Hier erkaufte sie (d. h. Dido) anfangs nur ein Stück Landes, welches Byrsa oder Stierhaut genannt wurde; mit diesem Namen aber verhielt es sich so: Dido, in Afrika angekommen, verlangte nur so viel Feldes, als sie mit einer Stierhaut zu umspannen vermochte. Diese Haut aber schnitt sie in so dünne Riemen, dass dieselbe den ganzen Raum einschloss, den jetzt Byrsa, die Burg Karthagos, einnimmt. Von dort aus erwarb sie mit ihren Schätzen immer größeres Gebiet, und ihr königlicher Geist gründete das mächtige Reich, das sie jetzt beherrscht".

Wir betrachten ein etwas einfacheres Problem als das der Dido. Angenommen man sucht bei vorgegebenen $0 < a < l$ diejenige Kurve in der oberen Halbebene, die die Punkte $A := (-a, 0)$ und $B := (a, 0)$ verbindet, die Länge $2l$ besitzt und zusammen mit dem Segment AB maximalen Flächeninhalt umschließt. Anschaulich ist völlig klar (trotzdem muss es bewiesen werden!), dass man als Lösung ein Kreissegment erhält, jedenfalls dann, wenn $2l \le \pi a$, also $2l$ nicht größer als der halbe Umfang eines Kreises mit dem Radius a ist. Bezeichnet man mit r den Radius des gesuchten Kreises, so ist $r\alpha = l$ und $r\sin\alpha = a$. Eliminieren von r führt auf eine Gleichung für den Winkel α, nämlich

$$f(\alpha) := \sin\alpha - \frac{a}{l}\alpha = 0,$$

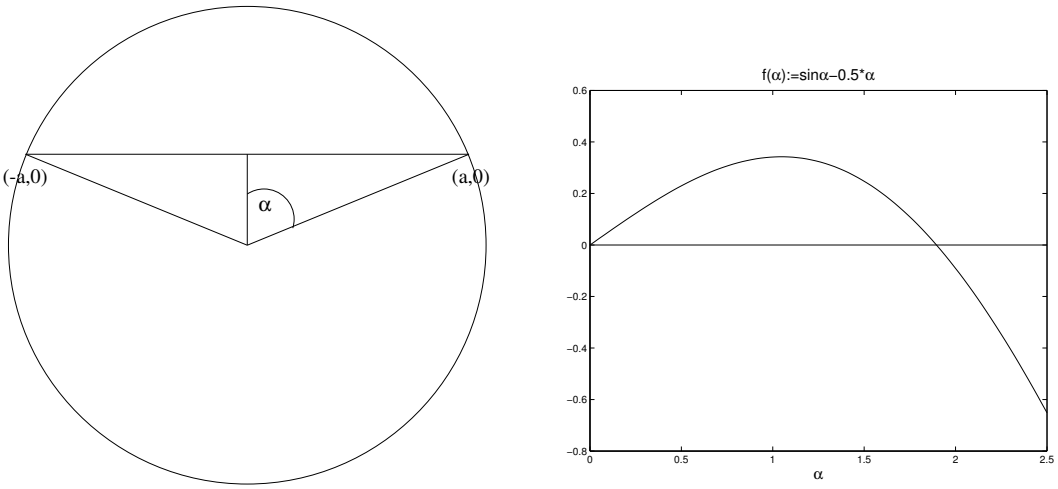

Abbildung 2.5: Das Problem der Dido

von der eine positive Lösung zu bestimmen ist. Es ist $f(0) = 0$, $f'(0) = 1 - a/l > 0$ und $f(\pi) = -(a/l)\pi < 0$. Daher besitzt f im Intervall $(0, \pi)$ eine Nullstelle, die nach obiger Zeichnung für $a/l = \frac{1}{2}$ etwa bei 1.9 liegt. Diese kann mit Hilfe des Newton-Verfahrens

$$\alpha_{k+1} := \alpha_k - \frac{f(\alpha_k)}{f'(\alpha_k)}, \qquad k = 0, 1, \ldots$$

mit einer geeigneten Startnäherung α_0 berechnet werden. Für $a/l = \frac{1}{2}$ und dem Startwert $\alpha_0 := 1.5$ erhalten wir z. B. die folgenden Werte:

α	$f(\alpha)$
1.50000000000000	$2.474949866040544e-00$
2.07655820063043	$-1.634734800818400e-01$
1.91050661565908	$-1.240206694219914e-02$
1.89562200298785	$-1.046263551932602e-04$
1.89549427647277	$-7.730581397247249e-09$
1.89549426703398	$-2.220446049250313e-16$
1.89549426703398	0

In Maple steht die Funktion `fsolve` zur Verfügung. Man erkennt, dass man den Bereich, in dem eine Lösung zu suchen ist, eventuell einschränken muss, um nicht eine triviale Lösung zu erhalten.

```
> fsolve(sin(alpha)-0.5*alpha,alpha);
                              0.
> fsolve(sin(alpha)-0.5*alpha,alpha,1.5..2);
                          1.895494267
```

In MATLAB steht die Funktion `fzero` zur Verfügung. Hier muss allerdings ein Startwert angegeben werden. Die Funktion, von der eine Nullstelle zu bestimmen ist, wird als inline object oder über ein function file übergeben. Man hat also hier zwei Möglichkeiten.

```
alpha_stern=fzero(inline('sin(alpha)-0.5*alpha'),1.9)
```

ist die erste Möglichkeit und liefert

$$\alpha^* = 1.89549426703398.$$

Mit der zweiten Möglichkeit schreiben wir ein File `fcn.m` mit dem Inhalt

```
function beta=fcn(alpha);
beta=sin(alpha)-0.5*alpha;
```

Der Aufruf

```
alpha_stern=fzero(@fcn,1.9)
```

liefert natürlich dasselbe Ergebnis. Diese zweite Möglichkeit ist bei komplizierten Funktionen sicher vorzuziehen. □

Beispiel: Zur Zeit t sei eine bestimmte Konzentration einer chemischen Substanz durch $g(t) := 10e^{-3t} + 2e^{-5t}$ gegeben. Es soll der Zeitpunkt bestimmt werden, in dem die Konzentration nur noch halb so groß wie zur Zeit $t = 0$ ist. Wegen $g(0) = 12$ führt dies auf eine Nullstellenaufgabe für

$$f(t) := 10e^{-3t} + 2e^{-5t} - 6.$$

Da f monoton fallend ist, $f(0) = 6$ und $\lim_{t\to\infty} f(t) = -6$, besitzt f genau eine Nullstelle auf $\mathbb{R}_+$. Diese liegt etwa bei 0.211, wie man z. B. mit Hilfe von `fsolve` (Maple) erhält. □

Beispiel: Die Gleichung $f(z) := e^z - z = 0$ besitzt keine reelle Lösung, da $f(x) \geq 1$ für alle $x \in \mathbb{R}$. Wir stellen die Frage, ob es komplexe $z \in \mathbb{C}$ mit $f(z) = 0$ gibt. Wir spalten in Real- und Imaginärteil auf: $z = x + iy$ und erhalten wegen $e^z = e^x(\cos y + i \sin y)$ das nichtlineare Gleichungssystem

$$g(x, y) := \begin{pmatrix} e^x \cos y - x \\ e^x \sin y - y \end{pmatrix} = 0.$$

Das Newton-Verfahren kann auch auf nichtlineare Gleichungssysteme übertragen werden, wie wir später sehen werden. Man beachte, dass viele mathematische Anwendersysteme auch komplex rechnen können und daher das Newton-Verfahren $z_{k+1} := z_k - f(z_k)/f'(z_k)$ auch ohne den Umweg über den $\mathbb{R}^2$ durchgeführt werden kann. In MATLAB könnte dies z. B. folgendermaßen aussehen:

```
z=complex(1,1); Z=z;
for k=1:6
   z=z-(exp(z)-z)/(exp(z)-1);
   Z=[Z;z];
end;
Z
```

Hier erhalten wir

k	z_k
0	$1.00000000000000 + 1.00000000000000i$
1	$0.41956978951242 + 1.08597257226218i$
2	$0.27943162439556 + 1.33130774424201i$
3	$0.31877394181938 + 1.33694557803917i$
4	$0.31813150923617 + 1.33723547391984i$
5	$0.31813150520475 + 1.33723570143070i$
6	$0.31813150520476 + 1.33723570143069i$

Entsprechend kann man auch in Maple vorgehen, weil auch in Maple reelle Zahlen nicht vor komplexen ausgezeichnet sind. Am einfachsten ist hier die Anwendung von `fsolve`, allerding sollte man Maple sagen, dass man im Komplexen nach einer Lösung sucht:

```
> fsolve(exp(z)=z,z);
```

$$\text{fsolve}(e^z = z,\ z)$$

```
> fsolve(exp(z)=z,z,complex);
```

$$.3181315052 - 1.337235701\,I$$

Die MATLAB-Funktion `fzero` geht von einer Nullstellenaufgabe für eine *reellwertige* Funktion aus. □

2.2.2 Nichtlineare Gleichungen

In diesem Unterabschnitt betrachten wir den eindimensionalen Spezialfall, genauer eine nichtlineare Gleichung

$$f(x) = 0$$

in einer unabhängigen reellen Variablen x. Da nichtlineare Gleichungen in der Regel nicht geschlossen gelöst werden können, ihre Lösungen also nicht in endlich vielen Schritten berechenbar sind, ist man auf Iterationsverfahren zur Approximation der Lösung angewiesen. Dies gilt natürlich erst Rrecht für den mehrdimensionalen Fall. Wir werden auf drei Verfahren etwas genauer eingehen.

Sind $a < b$ zwei Punkte, in denen f unterschiedliches Vorzeichen har, ist also $f(a)f(b) < 0$, ist ferner : $[a, b] \longrightarrow \mathbb{R}$ stetig, so liefert der Zwischenwertsatz die Existenz einer Nullstelle x^* von f in (a, b). Diese kann mit Hilfe des *Bisektionsverfahrens* berechnet werden:

- Input: Reelle Zahlen $a < b$, eine stetige Funktion $f : [a, b] \longrightarrow \mathbb{R}$ mit $f(a)f(b) < 0$, eine (kleine) Zahl $\delta > 0$, die die gewünschte Genauigkeit steuert. Ferner sei $\epsilon > 0$ die sogenannte *Maschinengenauigkeit*. In MATLAB ist `eps` der Abstand von 1.0 zur nächst größeren Gleitkommazahl, es ist `eps=2.220446049250313e-16`.
- Berechne $f_a := f(a)$, $f_b := f(b)$.
- Solange $b - a > \delta + \epsilon \max(|a|, |b|)$:

  - $c := \frac{1}{2}(a+b)$, berechne $f_c := f(c)$.
  - Falls $f_a f_c \leq 0$ (es ist Nullstelle in $[a, c]$)
    * $b := c$, $f_b := f_c$
  - Andernfalls (es ist Nullstelle in $[c, b]$)
    * $a := c$, $f_a := f_c$

- Output: $\frac{1}{2}(a+b)$.

In jedem Schritt wird die Länge des Intervalls, in dem eine Nullstelle gesucht wird, halbiert. Bezeichnet man mit $[a_0, b_0]$ das Ausgangsintervall und mit $[a_k, b_k]$ das Intervall im k-ten Schritt, so ist $0 \leq b_k - a_k = (b_0 - a_0)/2^k$, $k = 0, 1, \ldots$. Nach Konstruktion enthält das Intervall $[a_k, b_k]$ eine Nullstelle x^* von f. Bezeichnet man mit $x_k := \frac{1}{2}(a_k + b_k)$ den Mittelpunkt dieses Intervalls, so ist also

$$|x_k - x^*| \leq \frac{b_k - a_k}{2} = \frac{b_0 - a_0}{2^{k+1}}.$$

Wir schreiben[11] eine einfache MATLAB-Funktion und wenden diese anschließend auf ein Beispiel an.

```
function root=Bisection(fname,a,b,delta);
%*****************************************************************
%Input-Parameter:
%   fname     string, Name einer stetigen Funktion f einer Variablen
%     a,b     a<b definieren Intervall, auf dem f stetig ist mit
%             f(a)f(b)<=0
%   delta     positive reelle Zahl
%Output-Parameter:
%    root     Mittelpunkt eines Intervalls [alpha,beta] mit
%             f(alpha)f(beta)<=0 und
%             |beta-alpha|<=delta+eps max(|a|,|b|)
%*****************************************************************
  f_a=feval(fname,a); f_b=feval(fname,b);
  while b-a>delta+eps*max(abs(a),abs(b))
     c=0.5*(a+b); f_c=feval(fname,c);
     if f_a*f_c<=0
         b=c;  f_b=f_c;
     else
         a=c;  f_a=f_c;
     end;
  end
  root=0.5*(a+b);
%*****************************************************************
```

[11] Siehe C. F. van Loan (1997, S. 264).

Beispiel: Als Resultat von `x=Bisection(inline('tan(x/4)-1'),2,4,1e-12);` erhalten wir (mit `format long`) $x = 3.14159265358967$. □

Nachdem wir kurz die einfache Idee des Bisektionsverfahrens geschildert haben, kommen wir nun zum *Newton-Verfahren*, einem der wichtigsten Verfahren der Numerischen Mathematik. Auf die naheliegende Idee, die dem Newton-Verfahren zu grunde liegt, sind wir schon in der Einführung eingegangen. Ist x_k eine aktuelle Näherung für eine Nullstelle von f, so approximiere man f in der Nähe von x_k durch die lineare Funktion $f_k(x) := f(x_k)+f'(x_k)(x-x_{k+1})$, also die ersten beiden Terme einer Taylor-Entwicklung von f in x_k, und gewinne die neue Näherung x_{k+1} (für $f'(x_k) \neq 0$) als Nullstelle von f_k. Dies führt auf die Iterationsvorschrift des (eindimensionalen) Newton-Verfahrens:

$$x_{k+1} := x_k - \frac{f(x_k)}{f'(x_k)}, \qquad k = 0, 1, \ldots .$$

Nun interessieren natürlich Konvergenzaussagen zu diesem Verfahren. Der folgende Satz ist ein *lokaler Konvergenzsatz*.

Satz 2.1 *Die Funktion $f : \mathbb{R} \longrightarrow \mathbb{R}$ besitze in $x^* \in \mathbb{R}$ eine Nullstelle, es sei also $f(x^*) = 0$. Es gelte:*

1. *Die Funktion f ist auf einem Intervall U^*, welches x^* im Inneren enthält, stetig differenzierbar und f' dort lipschitzstetig, d. h. es existiert eine Konstante $L > 0$ mit $|f'(x) - f'(y)| \leq L\,|x - y|$ für alle $x, y \in U^*$.*
2. *Es ist $f'(x^*) \neq 0$, d. h. x^* ist eine einfache Nullstelle von f.*

Dann existiert ein $\delta > 0$ bzw. ein Intervall $I^* = [x^*-\delta, x^*+\delta] \subset U^*$ mit der Eigenschaft, dass für jeden Startwert $x_0 \in I^*$ die durch die Newton-Vorschrift

$$x_{k+1} := x_k - \frac{f(x_k)}{f'(x_k)}, \qquad k = 0, 1, \ldots ,$$

gewonnene Folge $\{x_k\}$ definiert ist (d. h. $f'(x_k)$ existiert und ist von Null verschieden) und gegen x^* konvergiert. Ferner existiert eine Konstante $C > 0$ mit

$$|x_{k+1} - x^*| \leq C\,|x_k - x^*|^2, \qquad k = 0, 1, \ldots ,$$

d. h. die Folge $\{x_k\}$ konvergiert quadratisch gegen x^*.

Beweis: Man wähle $\delta > 0$ so klein, dass $I^* := [x^* - \delta, x^* + \delta] \subset U^*$, $|f'(x)| \geq \frac{1}{2}|f'(x^*)|$ für alle $x \in I^*$ und $\delta \leq |f'(x^*)|/(2L)$. Sei $x_0 \in I^*$ beliebig. Wir werden die folgende Implikationskette nachweisen:

$$x_k \in I^* \Longrightarrow |x_{k+1} - x^*| \leq \frac{L}{|f'(x^*)|}|x_k - x^*|^2 \leq \frac{1}{2}|x_k - x^*| \Longrightarrow x_{k+1} \in I^*.$$

Dies wird dann zeigen, dass die gesamte Folge $\{x_k\}$ im Intervall I^* bleibt, die Folge $\{x_k\}$ wegen

$$|x_{k+1} - x^*| \leq \frac{1}{2}|x_k - x^*| \leq \frac{1}{2^{k+1}}|x_0 - x^*| \leq \frac{\delta}{2^{k+1}}$$

gegen x^* konvergiert und dies offensichtlich, wie man sagt, mit quadratischer Konvergenzgeschwindigkeit.

Sei also $x_k \in I^*$. Dann ist

$$\begin{aligned}
|x_{k+1} - x^*| &= \Big|x_k - x^* - \frac{f(x_k)}{f'(x_k)}\Big| \\
&= \frac{1}{|f'(x_k)|}\Big|\int_{x_k}^{x^*} [f'(x) - f'(x_k)]\,dx\Big| \\
&\qquad \text{(hier benutzen wir } f(x^*) = 0) \\
&\le \frac{2}{|f'(x^*)|}\Big|\int_{x_k}^{x^*} [f'(x) - f'(x_k)]\,dx\Big| \\
&\le \frac{2L}{|f'(x^*)|}\operatorname{sign}(x^* - x_k)\int_{x_k}^{x^*} |x - x^*|\,dx \\
&= \frac{L}{|f'(x^*)|}|x_k - x^*|^2 \\
&\le \frac{L\delta}{|f'(x^*)|}|x_k - x^*| \\
&\le \frac{1}{2}|x_k - x^*|.
\end{aligned}$$

Damit ist der Satz bewiesen. □

Bemerkung: Man sollte sich ganz deutlich machen, dass es sich bei dem letzten Satz um einen lokalen Konvergenzsatz handelt: Ist x^* eine einfache Nullstelle von f und ist f hinreichend glatt auf einer Umgebung von x^* (z. B. zweimal stetig differenzierbar, denn dies impliziert die Lipschitzstetigkeit), so konvergiert das Newton-Verfahren für jeden Startwert aus einer hinreichend kleinen Umgebung von x^* quadratisch gegen x^*. Die quadratische Konvergenzgeschwindigkeit ist deshalb eine so schöne Eigenschaft, weil sie grob gesagt bedeutet, dass sich in jedem Schritt die Anzahl gültiger Stellen verdoppelt. Man beachte aber, dass die Linearisierungs-Idee des Newton-Verfahrens i. Allg. auch nur lokal gerechtfertigt ist, so dass man eventuell erst mit einem anderen Verfahren in den Bereich kommen muss, in dem die vorzüglichen Konvergenzeigenschaften des Newton-Verfahrens gelten. Nur unter Konvexitäts- und Monotonievoraussetzungen an f kann man ein globales Konvergenzergebnis erwarten, siehe Satz 2.2. □

Beispiel: Sei $a > 0$ gegeben und $f(x) := x^2 - a$. Die eindeutige positive Nullstelle von f ist also die positive Quadratwurzel aus a. Das Newton-Verfahren lautet

$$x_{k+1} := x_k - \frac{x_k^2 - a}{2x_k} = \frac{1}{2}\Big(x_k + \frac{a}{x_k}\Big), \qquad k = 0, 1, \ldots.$$

Der lokale Konvergenzsatz 2.1 ist natürlich anwendbar. Es stellt sich aber heraus, dass Konvergenz sogar für jeden Startwert $x_0 > 0$ vorliegt[12]. □

Beispiel: Sei $f(x) := x - \cos x$. In Abbildung 2.6 veranschaulichen wir links die Glei-

[12] Näheres findet man z. B. bei

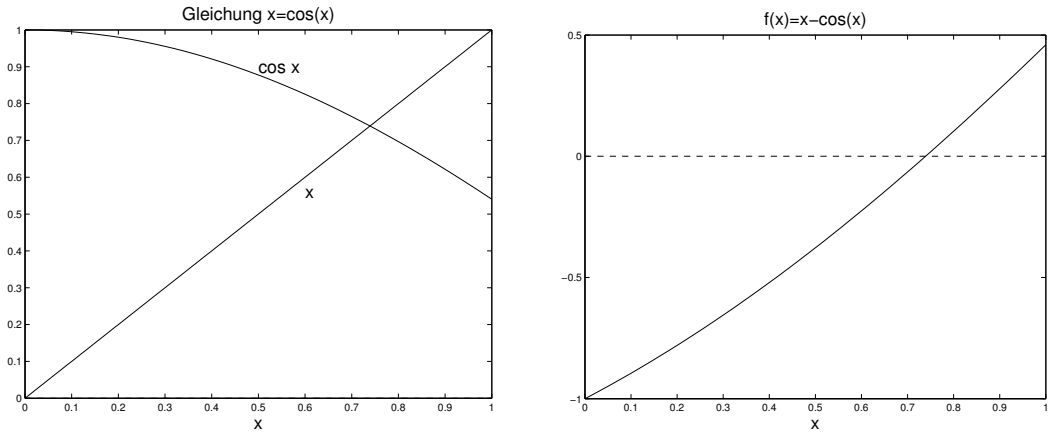

Abbildung 2.6: Die Gleichung $x = \cos x$ und die Funktion $f(x) := x - \cos x$

chung $x = \cos x$ und zeichnen rechts die Funktion $f(x) := x - \cos x$. Das auf f angewandte Newton-Verfahren lautet

$$x_{k+1} := x_k - \frac{x_k - \cos x_k}{1 + \sin x_k}.$$

Wir erhalten z. B. die folgenden Werte:

k	x_k
0	1.00000000000000
1	0.75036386784024
2	0.73911289091136
3	0.73908513338528
4	0.73908513321516
5	0.73908513321516

Man erkennt die vorzügliche Konvergenz. □

Globale Konvergenzaussagen für das Newton-Verfahren, bei denen also der Startwert x_0 aus einer hinreichend kleinen Umgebung einer Nullstelle x^* von f zu sein hat, gelten i. Allg. nur unter Monotonie- und Konvexitätsvoraussetzungen an f. Der folgende Satz ist ein solcher globaler Konvergenzsatz.

Satz 2.2 *Sei $f \in C^2[a,b]$ mit $f'(x) > 0$ und $f''(x) \geq 0$ für alle $x \in [a,b]$, d. h. die auf $[a,b]$ zweimal stetig differenzierbare Funktion f sei auf $[a,b]$ streng monoton wachsend und konvex. Ferner sei $f(a) < 0 < f(b)$ und $a - f(a)/f'(a) \leq b$. Dann liefert das Newton-Verfahren*

$$x_{k+1} := x_k - \frac{f(x_k)}{f'(x_k)}, \qquad k = 0, 1, \ldots$$

J. Werner (1992, S. 1 ff.) *Numerische Mathematik 1*. Vieweg, Braunschweig-Wiesbaden
und

C. F. van Loan (1997, S. 259 ff) *Introduction to Scientific Computing. A Matrix-Vector Approach using MATLAB*. Prentice-Hall, Upper Saddle River.

für jedes $x_0 \in [a,b]$ eine Folge $\{x_k\}$ mit der Eigenschaft, dass[13] *$\{x_k\}_{k\in\mathbb{N}}$ monoton fallend und von mindestens zweiter Ordnung gegen die einzige Nullstelle x^* von f in $[a,b]$ konvergiert, wobei natürlich $x_k \neq x^*$ für alle k angenommen wird.*

Beweis: Die Folge $\{x_k\}$ wird durch die Iterationsfunktion

$$F(x) := x - \frac{f(x)}{f'(x)}$$

erzeugt, d. h. es ist $x_{k+1} = F(x_k)$, $k = 0, \ldots,$. Dann ist

$$F'(x) = \frac{f(x)f''(x)}{f'(x)^2} \begin{cases} \leq 0, & x \in [a, x^*], \\ \geq 0, & x \in [x^*, b]. \end{cases}$$

Sei $x_0 \in [a,b]$ beliebig. Wir zeigen, dass $b \geq x_k \geq x_{k+1} \geq x^*$, $k = 1, \ldots$. Nach spätestens einem Schritt sind also alle Iterierten des Newton-Verfahrens rechts der einzigen Nullstelle x^* von f in $[a,b]$, ferner ist die Folge $\{x_k\}_{k\in\mathbb{N}}$ monoton fallend. Denn ist $a \leq x_0 \leq x^*$, so ist

$$x^* = F(x^*) \leq F(x_0) = x_1 \leq F(a) \leq b,$$

d. h. nach einem Schritt ist man rechts von x^*, aber immer noch links von b. Ist aber $x_k \in [x^*, b]$, so ist

$$x^* = F(x^*) \leq F(x_k) = x_{k+1} \leq x_k \leq b.$$

Als monoton fallende, nach unten durch x^* beschränkte Folge ist $\{x_k\}_{k\in\mathbb{N}}$ konvergent. Der Limes ist notwendigerweise eine Nullstelle von f, stimmt also mit x^* überein. Die quadratische Konvergenz folgt aus dem lokalen Konvergenzsatz. □

Beispiel: Sei $f(x) := x - \cos x$. Die Voraussetzungen von Satz 2.2 sind offenbar mit $[a,b] := [0,1]$ erfüllt, so dass für jeden Startwert aus diesem Intervall Konvergenz des Newton-Verfahrens vorliegt. □

Wir geben nun eine MATLAB-Funktion an, durch die das Newton-Verfahren implementiert wird. Hierbei wird mit einer aktuellen Näherung x_k abgebrochen, wenn $|f(x_k)|$ kleiner einer vorgegebenen Toleranz ist oder k größer einer maximalen Anzahl von Iterationsschritten ist.

```
function [x,iter]=Newton(fun,x_0,tol,max_iter);
%*****************************************************************
%Input-Parameter:
%     fname       Name einer Funktion fun. [f,f_strich]=fun(x)
%                 ergibt Funktionswert und Ableitung der Funktion in x
%     x_0         Startwert
%     tol         positiver Wert, abbruch wenn |f|<tol
%     max_iter    maximale Anzahl von Iterationen
%Output-Parameter:
```

[13] Für uns ist, eventuell im Widerspruch zu gewissen DIN-Normen, 0 keine natürliche Zahl.

```
%     x           gefundene Nullstelle (bei Erfolg)
%     iter        Anzahl der durchgefuehrten Iterationen
%*********************************************************************
  iter=1;
  x=x_0;  [f,f_strich]=feval(fun,x);
  while (abs(f)>tol)&(iter<max_iter)
     iter=iter+1;
     x=x-f/f_strich;
     [f,f_strich]=feval(fun,x);
  end;
%*********************************************************************
```

Beispiel: Sei $f(x) := x - 0.5 \sin x - 0.85$ (Keplersche Gleichung). Durch einen Plot verschaffen wir uns zunächst einen Überblick über den Funktionsverlauf, siehe Abbildung 2.7. Zur Anwendung der obigen Funktion `Newton` schreiben wir ein function file

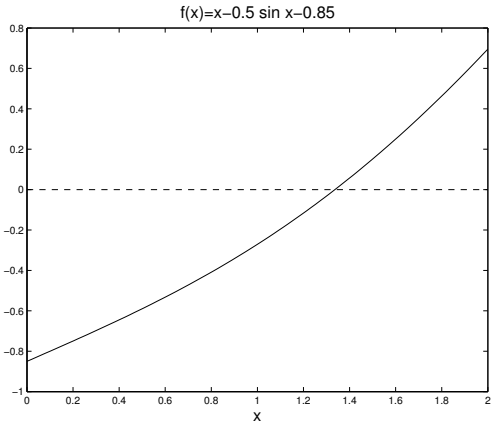

Abbildung 2.7: Die Funktion $f(x) := x - 0.5 \sin x - 0.85$

`Beisp1.m` mit dem Inhalt

```
function [f,f_strich]=Beisp1(x);
f=x-0.5*sin(x)-0.85;
if nargout>1
   f_strich=1-0.5*cos(x);
end;
```

Mit dem Aufruf `[x,iter]=Newton(@Beisp1,1.2,1e-12,20)` erhalten wir

$$x = 1.33631781724031, \qquad \text{iter} = 5.$$

Wendet man Satz 2.2 an, so erkennt man, dass Konvergenz für jeden Startwert aus $[0, 1.7]$ vorliegt. □

Auch auf das *Sekantenverfahren* waren wir in der Einführung schon kurz eingegangen, wir hatten sogar schon eine MATLAB-Funktion hierzu angegeben. Man gewinnt das

Sekantenverfahren aus dem Newton-Verfahren, in dem man den Differentalquotienten $f'(x_k)$ durch den Differenzenquotienten $(f(x_k) - f(x_{k-1}))/(x_k - x_{k-1})$ ersetzt. Daher lautet die Iterationsvorschrift des Sekantenverfahrens:

$$x_{k+1} := x_k - \frac{(x_k - x_{k-1})f(x_k)}{f(x_k) - f(x_{k-1})}, \qquad k = 1, 2, \ldots.$$

Für dieses Verfahren gilt ebenfalls ein lokaler Konvergenzsatz, ganz ähnlich Satz 2.1 für das Newton-Verfahren[14]. Im wesentlichen unter denselben Voraussetzungen wie in Satz 2.2 (also: x^* einfache Nullstelle von f, die Funktion f zweimal stetig differenzierbar auf einer Umgebung von x^*) kann gezeigt werden, dass es ein $\delta > 0$ gibt mit der Eigenschaft, dass für beliebige Startwerte $x_0, x_1 \in [x^* - \delta, x^* + \delta]$ mit $x_0 \neq x_1$ das Sekantenverfahren eine gegen x^* konvergente Folge $\{x_k\}$ liefert. Genauer ist $|x_k - x^*| \leq c_k$ mit einer Nullfolge $\{c_k\} \subset \mathbb{R}_+$, zu der eine Konstante $C > 0$ mit $c_{k+1} \leq C c_k^{(1+\sqrt{5})/2}$ für alle k existiert.

Bemerkung: Etwas zur Allgemeinbildung: Man sagt, eine Strecke der Länge $L > 0$ werde nach dem *goldenen Schnitt* in Strecken der Länge $l \geq L/2$ und $L - l$ geteilt (also ist l die längere Strecke), wenn das Verhältnis L/l der gesamten Strecke zur längeren gleich dem Verhältnis $l/(L - l)$ der längeren zur kürzeren Strecke ist. Es hat also

$$\frac{L}{l} = \frac{l}{L - l} = \frac{1}{L/l - 1}$$

bzw.

$$\left(\frac{L}{l}\right)^2 - \frac{L}{l} = 1$$

zu gelten. Die positive Lösung ist

$$\frac{L}{l} = \tau := \frac{1 + \sqrt{5}}{2} \approx 1.618033989.$$

Daher nennt man τ die Goldene Schnitt Zahl.

Die Goldene Schnitt Zahl hängt eng mit den Fibonacci-Zahlen zusammen. Fibonacci bzw. Leonardo von Pisa stellte im liber abbaci (1202) die folgende Aufgabe: Angenommen, Kaninchen brauchen einen Monat, um geschlechtsreif zu werden. Danach reproduzieren sie sich einmal jeden Monat, wobei sie einen Monat lang tragen. Im Null-ten Monat sei ein neugeborenes Paar von Kaninchen vorhanden. Wieviele Kaninchenpaare sind im k-ten Monat anzutreffen (wobei angenommen wird, dass Kaninchen unendlich lange leben)?

Bezeichnen wir ein neugeborenes Kaninchenpaar mit N und ein ausgewachsenes Paar mit A, so hat man also

Monat	0	1	2	3	4	5	6
N, A	$1N$	$1A$	$1A + 1N$	$2A + 1N$	$3A + 2N$	$5A + 3N$	$8A + 5N$
Σ	1	1	2	3	5	8	13

[14]Siehe z. B.

J. Werner (1992, S. 112) *Numerische Mathematik 1*. Vieweg, Braunschweig-Wiesbaden.

Bezeichnet f_k die Anzahl der Kaninchenpaare im k-ten Monat, so ist also

$$f_0 = f_1 = 1, \qquad f_{k+1} = f_k + f_{k-1} \quad (k = 1, 2, \ldots).$$

Die Zahlen f_k, $k = 0, 1, \ldots$, heißen *Fibonacci-Zahlen*. Die Binetsche Formel besagt, dass

$$f_k = \frac{1}{\sqrt{5}}[\tau^{k+1} - (-\tau)^{-(k+1)}], \qquad k = 0, 1, \ldots.$$

Diese beweist man leicht durch vollständige Induktion nach k. Hieraus folgt dann

$$\lim_{k\to\infty} \frac{f_{k+1}}{f_k} = \tau.$$

Das Verhältnis zweier aufeinanderfolgenden Fibonacci-Zahlen konvergiert also gegen die goldene Schnitt Zahl.

Ein weiteres wichtiges Verhältnis ist das DIN-Format. Ein Rechteck mit Seitenlängen L und $l \le L$ hat DIN-Format, wenn

$$\frac{L}{l} = \frac{l}{L/2} \qquad \text{bzw.} \qquad \frac{L}{l} = \sqrt{2}.$$

Gleichbedeutend ist, dass das Halbieren einer Seite Papier, welche DIN-Format hat, wieder ein DIN-Format ergibt. Ein Rechteck im DIN-Format mit einer Fläche von $1\,\mathrm{m}^2 = 10\,000\,\mathrm{cm}^2$, hat die Seitenlängen l (in cm) mit $\sqrt{2}l^2 = 100^2$, also $l = 100/2^{1/4} \approx 84.0896$ und $L = \sqrt{2}l \approx 118.9207$. Ein Rechteck mit diesen Maßen hat das Format DIN A0. Die ersten Formate sind daher

Format	hoch	breit
DIN A0	118.9	84.1
DIN A1	84.1	59.5
DIN A2	59.5	42.0
DIN A3	42.0	29.7
DIN A4	29.7	21.0

□

Beispiel: Wir betrachten noch einmal das letzte Beispiel. Nach

```
[x,f_x,info]=Sekant(@Beisp1,1.0,1.2,1e-12,20)
```

erhalten wir

$$x = 1.33631781724031, \qquad f_x = 0, \qquad \text{info} = 6.$$

Es sind also nur 6 Iterationen durchgeführt worden. In jedem Schritt, außer dem ersten, erfolgt nur eine Funktionsauswertung. Dagegen müssen beim Newton-Verfahren in jedem Schritt zwei Funktionswerte berechnet werden. □

2.2.3 Der Banachsche Fixpunktsatz

Wir haben bisher nur nichtlineare Gleichungen in $\mathbb{R}$ betrachtet und hierzu einige Verfahren kennen gelernt. Jetzt wollen wir auch nichtlineare Gleichungssysteme untersuchen, etwa ein nichtlineares Gleichungssystem der Form

$$f(x_1,\ldots,x_n) = \begin{pmatrix} f_1(x_1,\ldots,x_n) \\ \vdots \\ f_n(x_1,\ldots,x_n) \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} = 0.$$

Hierbei ist $f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ eine gegebene Abbildung, von der i. Allg. vorausgesetzt wird, dass sie hinreichend glatt ist. Während ein Abstandsbegriff in $\mathbb{R}$ durch den Absolutbetrag in natürlicher Weise gegeben ist, kann man im $\mathbb{R}^n$ und erst recht in Funktionenräumen, wie sie etwa bei gewöhnlichen Differentialgleichungen oder Approximationsaufgaben vorkommen, auf verschiedene Weise einen Abstandsbegriff zwischen Punkten des $\mathbb{R}^n$ (oder gewissen Funktionen) einführen. Dies führt zum Begriff einer *Norm* auf dem $\mathbb{R}^n$ oder allgemeiner auf einem linearen Raum (bzw. einem Vektorraum).

Wir nehmen an, dass die folgenden Begriffe aus Analysis II bekannt sind:

- Linearer[15] normierter Raum, Norm, Konvergenz und Cauchyfolge in einem linearen normierten Raum , Banachraum,
- Abgeschlossene Teilmengen eines linearen normierten Raumes,
- Stetige Abbildungen zwischen linearen normierten Räumen,
- Lipschitzstetige Abbildungen.
- Äquivalenz der Normen auf dem $\mathbb{R}^n$ (bzw. einem endlichdimensionalen linearen Raum): Der Konvergenzbegriff auf dem $\mathbb{R}^n$ ist normunabhängig.

Beispiele: Der $\mathbb{R}^n$, versehen mit einer beliebigen Vektornorm $\|\cdot\|$, ist bekanntlich ein Banachraum. Die wichtigsten Normen im $\mathbb{R}^n$ sind die euklidische Vektornorm, die Betragssummennorm und die Maximumnorm, welche für einen Vektor $x = (x_j) \in \mathbb{R}^n$ durch

$$\|x\|_2 := \Big(\sum_{j=1}^{n} x_j^2\Big)^{1/2}, \qquad \|x\|_1 := \sum_{j=1}^{n} |x_j|, \qquad \|x\|_\infty := \max_{j=1,\ldots,n} |x_j|$$

gegeben sind. Da wir uns in diesem Kapitel mit Problemen im $\mathbb{R}^n$ beschäftigen, wollen wir noch nicht auf Beispiele unendlichdimensionaler linearer normierter Räume eingehen. □

Jetzt folgt der bekannte Banachsche Fixpunktsatz (auch Kontraktionssatz genannt), den wir nur der Vollständigkeit halber beweisen.

[15] Der zugrunde gelegte Skalarkörper ist bei uns grundsätzlich der Körper $\mathbb{R}$ der reellen Zahlen.

Satz 2.3 (Banach) *Sei $(X, \|\cdot\|)$ ein Banachraum, $D \subset X$ abgeschlossen und $F : X \longrightarrow X$ eine Abbildung. Es sei $F(D) \subset D$ und F auf D kontrahierend, d. h. es existiert eine Konstante $q \in (0,1)$ mit*

$$\|F(x) - F(y)\| \leq q\,\|x - y\| \qquad \text{für alle } x, y \in D.$$

Dann besitzt F genau einen Fixpunkt x^* in D, die Folge $\{x_k\}$ mit $x_{k+1} := F(x_k)$ konvergiert gegen x^* für jedes $x_0 \in D$ und es gilt die Fehlerabschätzung

$$\|x^* - F(x)\| \leq \frac{q}{1-q}\,\|x - F(x)\| \qquad \text{für alle } x \in D.$$

Beweis: Sei $x \in D$ beliebig. Man definiere eine Folge $\{x_k\} \subset D$ durch

$$x_0 := x, \qquad x_{k+1} := F(x_k) \qquad (k = 0, 1, \ldots).$$

Durch vollständige Induktion nach k zeigt man, dass $\|x_{k+1} - x_k\| \leq q^k\,\|x_1 - x_0\|$, $k = 0, 1, \ldots$, anschließend folgt mit Hilfe der Dreiecksungleichung

$$(*) \qquad \|x_{k+p} - x_k\| \leq \frac{q^k}{1-q}\,\|x_1 - x_0\| \qquad (k = 0, 1, \ldots,\ p \in \mathbb{N}).$$

Daher ist $\{x_k\}$ eine Cauchyfolge, also konvergent gegen ein $x^* \in X$, welches wegen der Abgeschlossenheit von D sogar in D liegt. Da F insbesondere stetig ist und $x_{k+1} = F(x_k)$ gilt, ist $x^* = F(x^*)$, also $x^* \in D$ ein Fixpunkt von F. Da F auf D kontrahiert, ist x^* einziger Fixpunkt von F in D. Mit $p \to \infty$ folgt aus $(*)$, dass

$$\|x^* - x_k\| \leq \frac{q^k}{1-q}\,\|x_1 - x_0\|, \qquad k = 0, 1, \ldots.$$

Mit $k = 1$ folgt hieraus die behauptete Fehlerabschätzung. □

Beispiel: Wir betrachten den eindimensionalen Fall. Sei also im Banachschen Fixpunktsatz speziell $(X, \|\cdot\|) := (\mathbb{R}, |\cdot|)$, ferner sei $F : D \longrightarrow \mathbb{R}$ auf D stetig differenzierbar. Wie können wir überprüfen, ob F auf D kontrahierend ist? Das geeignete Hilfsmittel hierzu ist der Mittelwertsatz der Differentialrechnung. Hiernach existiert zu beliebigen $x, y \in D$ ein $\theta \in (0,1)$ mit

$$F(y) - F(x) = F'(\underbrace{x + \theta(y - x)}_{\in D})(y - x).$$

Für beliebige $x, y \in D$ ist daher

$$|F(x) - F(y)| \leq \sup_{\xi \in D} |F'(\xi)|\,|x - y|.$$

Ist also $q := \sup_{\xi \in D} |F'(\xi)| < 1$ (insbesondere also endlich, was für nicht kompaktes D nicht selbstverständlich ist), so ist F auf D kontrahierend. □

Beispiele: Mit Hilfe des letzten Beispiels wollen wir den Fixpunktsatz auf zwei Beispiele von Fixpunktaufgaben in $(\mathbb{R}, |\cdot|)$ anwenden.

Gegeben sei die Fixpunktaufgabe $x = e^{-x}$, es sei also bei der Anwendung des Banachschen Fixpunktsatzes $F(x) := e^{-x}$. Die Abbildung F bildet offensichtlich $D := [e^{-1}, 1]$ in sich ab, ferner ist

$$q := \sup_{\xi \in D} |F'(\xi)| = \max_{\xi \in [1/e,1]} e^{-\xi} = e^{-1} \approx 0.3679,$$

also F auf dem abgeschlossenen Intervall $D := [e^{-1}, 1]$ kontrahierend. Der Banachsche Fixpunktsatz liefert, dass die durch $x_{k+1} := \exp(-x_k)$ gewonnene Folge $\{x_k\}$ für jeden Startwert $x_0 \in D$ gegen den einzigen Fixpunkt x^* von F in D konvergiert. Beachtet man nun noch, dass für beliebiges $x_0 \in \mathbb{R}$ die nächste Iterierte $x_1 := F(x_0)$ in $\mathbb{R}_+$ ist, danach $x_2 = F(x_1) \in (0, 1)$ und schließlich $x_3 = F(x_2) \in [e^{-1}, 1]$, man also nach spätestens drei Schritten in D "gelandet" ist, so erkennt man, dass die durch $x_{k+1} := \exp(-x_k)$ gewonnene Folge $\{x_k\}$ für jeden Startwert $x_0 \in \mathbb{R}$ gegen den einzigen Fixpunkt x^* von F konvergiert. In Abbildung 2.8 veranschaulichen veranschaulichen wir uns die Fixpunk-

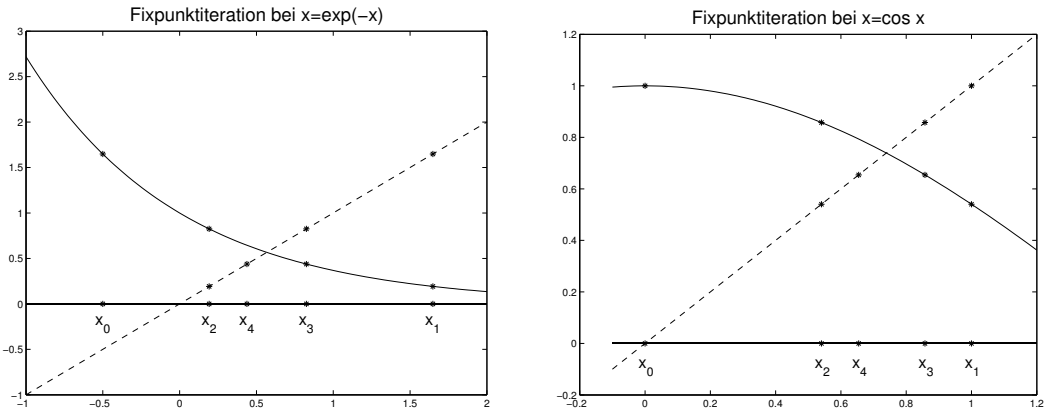

Abbildung 2.8: Veranschaulichung der Fixpunktiteration

titeration. Mit $x := 0.5$ liefert die Fehlerabschätzung im Banachschen Fixpunktsatz, dass $|x^* - \exp(-0.5)| \leq 0.0620$.

Als zweites Beispiel betrachten wir die Fixpunktaufgabe $x = \cos x$. Die Argumentation ist hier fast dieselbe wie im ersten Beispiel, diesmal ist $F(x) := \cos x$. Offensichtlich bildet F das Intervall $D := [0, 1]$ in sich ab. Wegen

$$q := \sup_{\xi \in D} |F'(\xi)| = \max_{\xi \in [0,1]} \sin \xi = \sin 1 \approx 0.8415$$

ist F auf D kontrahierend. Für beliebiges $x_0 \in \mathbb{R}$ ist spätestens die zweite Iterierte x_2 in $[0, 1]$ enthalten, so dass die durch $x_{k+1} := \cos x_k$ gewonnene Folge für jeden Startwert $x_0 \in \mathbb{R}$ gegen den einzigen Fixpunkt x^* konvergiert. □

Ganz kurz wollen wir nun auf *Matrixnormen* eingehen. Sei also $\|\cdot\|$ eine gegebene Norm auf dem $\mathbb{R}^n$. Ist $A \in \mathbb{R}^{n \times n}$ eine $n \times n$-Matrix (bzw. die durch $x \mapsto Ax$ definierte lineare Abbildung vom $\mathbb{R}^n$ in den $\mathbb{R}^n$), so heißt die durch

$$\|A\| := \sup_{x \neq 0} \frac{\|Ax\|}{\|x\|} = \max_{x : \|x\| = 1} \|Ax\|$$

definierte Abbildung $\|\cdot\| : \mathbb{R}^{n\times n} \longrightarrow \mathbb{R}$ die der Vektornorm $\|\cdot\|$ *zugeordnete Matrixnorm* auf $\mathbb{R}^{n\times n}$. Die Eigenschaften einer Norm weist man leicht nach. Ist sowohl auf dem $\mathbb{R}^n$ als auch auf dem $\mathbb{R}^m$ je eine Norm gegeben, so kann entsprechend natürlich auch eine diesen beiden Vektornorm zugeordnete Matrixnorm auf $\mathbb{R}^{m\times n}$ definiert werden. Offensichtlich ist $\|Ax\| \le \|A\|\,\|x\|$ für alle $x \in \mathbb{R}^n$.

Beispiele: Für die gängigen Vektornormen soll die zugeordnete Matrixnorm berechnet werden. Zunächst betrachten wir die Maximumnorm $\|\cdot\|_\infty$ auf dem $\mathbb{R}^n$. Sei $A = (a_{ij}) \in \mathbb{R}^{n\times n}$. Für beliebige $i \in \{1,\dots,n\}$ und $x \in \mathbb{R}^n$ ist dann

$$|(Ax)_i| = \Big|\sum_{j=1}^n a_{ij}x_j\Big| \le \sum_{j=1}^n |a_{ij}| \underbrace{|x_j|}_{\le\|x\|_\infty} \le \Big(\max_{i=1,\dots,n}\sum_{j=1}^n |a_{ij}|\Big)\|x\|_\infty$$

und folglich

$$\|Ax\|_\infty \le \Big(\max_{i=1,\dots,n}\sum_{j=1}^n |a_{ij}|\Big)\|x\|_\infty.$$

Daher ist

$$\|A\|_\infty := \sup_{x\neq 0}\frac{\|Ax\|_\infty}{\|x\|_\infty} \le \max_{i=1,\dots,n}\sum_{j=1}^n |a_{ij}|.$$

Wir behaupten, dass hier Gleichheit gilt. Sei $\max_{i=1,\dots,n}\sum_{j=1}^n |a_{ij}| = \sum_{j=1}^n |a_{kj}|$, die maximale Zeilenbetragssume von A trete also in der k-ten Zeile auf. Ist $a_{kj} = 0$, $j = 1,\dots,n$, so ist $A = 0$ die Nullmatrix, ein trivialer Fall, in dem die Behauptung richtig ist. Andernfalls definiere man $\hat{x} \in \mathbb{R}^n$ durch

$$\hat{x}_j := \begin{cases} \operatorname{sign}(a_{kj}), & a_{kj} \neq 0, \\ 0, & a_{kj} = 0, \end{cases} \qquad j = 1,\dots,n.$$

Dann ist $\|\hat{x}\|_\infty = 1$ und

$$\|A\|_\infty \le \max_{i=1,\dots,n}\sum_{j=1}^n |a_{ij}| = \sum_{j=1}^n a_{kj}\hat{x}_j = \Big|\sum_{j=1}^n a_{kj}\hat{x}_j\Big| \le \|A\hat{x}\|_\infty \le \|A\|_\infty \underbrace{\|\hat{x}\|_\infty}_{=1} = \|A\|_\infty,$$

womit die Behauptung bewiesen ist. Naheliegenderweise nennt man die Matrixnorm $\|\cdot\|_\infty$ die *maximale Zeilenbetragssummennorm*.

Entsprechend ist die der Betragssummennorm $\|\cdot\|_1$ zugeordnete Matrixnorm durch

$$\|A\|_1 = \max_{j=1,\dots,m}\sum_{i=1}^n |a_{ij}|$$

gegeben, die *maximale Spaltenbetragssummennorm*.

Jetzt soll die der euklidischen Vektornorm zugeordnete Matrixnorm berechnet werden. Hierbei benutzen wir (siehe auch Aufgabe 15 in Abschnitt 2.1):

- *Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch. Dann ist*

$$x^T Ax \le \lambda_{\max}(A)\,\|x\|_2^2 \qquad \textit{für alle } x \in \mathbb{R}^n.$$

Hierbei bedeutet $\lambda_{\max}(A)$ einen maximalen Eigenwert von A.

Denn seien $\lambda_1, \ldots, \lambda_n$ die (notwendig reellen) Eigenwerte von A und $\{u_1, \ldots, u_n\}$ ein zugehöriges Orthonormalsystem von Eigenvektoren. Ein beliebiges $x \in \mathbb{R}^n$ lässt sich eindeutig in der Form $x = \sum_{i=1}^n \alpha_i u_i$ darstellen. Dann ist

$$x^T A x = \Big(\sum_{i=1}^n \alpha_i u_i\Big)^T \Big(\sum_{i=1}^n \lambda_i \alpha_i u_i\Big) = \sum_{i=1}^n \lambda_i \alpha_i^2 \le \lambda_{\max}(A) \sum_{i=1}^n \alpha_i^2 = \lambda_{\max}(A)\, \|x\|_2^2,$$

womit die Behauptung bewiesen ist.

Sei nun $A \in \mathbb{R}^{n\times n}$ beliebig. Wendet man das letzte Resultat auf die symmetriche (und positiv semidefinite) Matrix $A^T A$ an, so erhält man

$$\|Ax\|_2 = \sqrt{(Ax)^T (Ax)} = \sqrt{x^T A^T A x} \le \sqrt{\lambda_{\max}(A^T A)\, \|x\|_2^2} = \sqrt{\lambda_{\max}(A^T A)}\, \|x\|_2,$$

so dass

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

Ist andererseits $u_{\max}$ ein zu $\lambda_{\max}(A^T A)$ gehörender, durch $\|u_{\max}\|_2 = 1$ normierter Eigenvektor von $A^T A$, so ist

$$\|A\|_2 \le \lambda_{\max}(A^T A) = \sqrt{u_{\max}^T A^T A u_{\max}} = \|A u_{\max}\|_2 \le \|A\|_2 \underbrace{\|u_{\max}\|_2}_{=1} = \|A\|_2,$$

womit insgesamt

$$\|A\|_2 = \sqrt{\lambda_{\max}(A^T A)}$$

bewiesen ist. Die der euklidischen Vektornorm zugeordnete Matrixnorm wird auch *Spektralnorm* genannt. Sind $\lambda_1, \ldots, \lambda_n$ die (i. Allg. komplexen) Eigenwerte einer $n \times n$-Matrix A, so heißt $\rho(A) := \max_{i=1,\ldots,n} |\lambda_i|$ der *Spektralradius* von A (der minimale Radius eines Kreises um den Nullpunkt in der komplexen Zahlenebene, der alle Eigenwerte von A enthält). Dann kann man auch

$$\|A\|_2 = \sqrt{\rho(A^T A)}$$

schreiben. □

Jetzt erinnern wir an einige Begriffe und Ergebnisse der Differentialrechnung im $\mathbb{R}^n$ (wir folgen ziemlich wörtlich J. Werner (1992, S 92 ff.)).

1. Sei $U \subset \mathbb{R}^n$ offen und $f : U \longrightarrow \mathbb{R}^m$ eine Abbildung. Diese heißt in $x \in U$ *(total) differenzierbar*, wenn es eine lineare Abbildung $A : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ bzw. eine Matrix $A \in \mathbb{R}^{m\times n}$ mit

   $$(*) \qquad \lim_{h\to 0} \frac{\|f(x+h) - f(x) - Ah\|}{\|h\|} = 0$$

   gibt. Hierbei ist $\|\cdot\|$ eine beliebige Norm auf dem $\mathbb{R}^m$ bzw. dem $\mathbb{R}^n$.

2. Sei $U \subset \mathbb{R}^n$ offen und $f : U \longrightarrow \mathbb{R}^m$ in $x \in U$ differenzierbar, d. h. es gelte $(*)$ mit einer Matrix $A = (a_{ij}) \in \mathbb{R}^{m\times n}$. Ist

$$f(x) = \begin{pmatrix} f_1(x) \\ \vdots \\ f_m(x) \end{pmatrix} = \begin{pmatrix} f_1(x_1, \ldots, x_n) \\ \vdots \\ f_n(x_1, \ldots, x_n) \end{pmatrix},$$

so gilt:

   (a) f ist in x stetig.

   (b) Alle Komponenten $f_i : U \longrightarrow \mathbb{R}$, $i = 1, \ldots, m$, von f sind in x partiell differenzierbar und es ist

$$a_{ij} = \frac{\partial f_i}{\partial x_j}(x) \qquad i = 1, \ldots, m,\ j = 1, \ldots, n.$$

3. Sei $U \subset \mathbb{R}^n$ offen.

   Ist $f : U \longrightarrow \mathbb{R}^m$ und sind alle Komponenten $f_i : U \longrightarrow \mathbb{R}$, $i = 1, \ldots, m$, von f in $x \in U$ partell differenzierbar, so heißt

$$f'(x) := \Big(\frac{\partial f_i}{\partial x_j}(x)\Big)_{\substack{i=1,\ldots,m \\ j=1,\ldots,n}} \in \mathbb{R}^{m\times n}$$

   die *Funktionalmatrix* (gelegentlich, vor allem in der englischsprachigen Literatur, *Jacobi-Matrix* bzw. *Jacobian* genannt) von f in x.

   Ist $f : U \longrightarrow \mathbb{R}$ in $x \in U$ partiell differenzierbar, so heißt

$$\nabla f(x) := \Big(\frac{\partial f}{\partial x_1}(x), \ldots, \frac{\partial f}{\partial x_n}(x)\Big)^T \in \mathbb{R}^n$$

   der *Gradient* von f in x.

4. Ist $U \subset \mathbb{R}^n$ offen und $f : U \longrightarrow \mathbb{R}^m$ eine Abbildung mit der Eigenschaft, dass sämtliche Komponenten f_i von f in $x \in U$ stetig partiell differenzierbar sind (d. h. alle partiellen Ableitungen existieren in einer Umgebung von x und sind in x stetig), so ist f in x differenzierbar.

5. Mittelwertsatz: Sei $D \subset \mathbb{R}^n$ nichtleer und *konvex* (d. h. mit zwei Punkten $x, y \in D$ ist $(1 - t)x + ty \in D$ für alle $t \in [0, 1]$) und $U \supset D$ eine offene Obermenge. Die Abbildung $f : U \longrightarrow \mathbb{R}^m$ (bzw. alle ihre Komponenten f_i) sei (bzw. seien) in jedem Punkt $x \in D$ stetig partiell differenzierbar. Für alle $x, y \in D$ ist dann

$$f(y) - f(x) = \int_0^1 f'(x + t(y - x))(y - x)\, dt = \Big(\int_0^1 f'(x + t(y - x))\, dt\Big)(y - x).$$

   Hierbei ist das Integral über eine m-Vektorfunktion bzw. eine $m \times n$-Matrixfunktion komponenten- bzw. koeffizientenweise zu verstehen. Für $i = 1, \ldots, m$

und alle $x, y \in D$ gilt daher

$$\begin{aligned} f_i(y) - f_i(x) &= \int_0^1 \nabla f_i(x + t(y-x))^T (y-x)\, dt \\ &= \sum_{j=1}^{n} \Big(\int_0^1 \frac{\partial f_i}{\partial x_j}(x + t(y-x))\, dt \Big)(y_j - x_j). \end{aligned}$$

Das folgende Lemma (siehe z. B. J. Werner (1992, S. 93)) geben wir ohne (seinen einfachen) Beweis an.

Lemma 2.4 *Sei $g : [a,b] \longrightarrow \mathbb{R}^n$ stetig und $\|\cdot\|$ eine Norm auf dem $\mathbb{R}^n$. Dann ist*

$$\Big\| \int_a^b g(t)\, dt \Big\| \le \int_a^b \|g(t)\|\, dt.$$

Nun ist es einfach, den folgenden Satz zu beweisen. Mit seiner Hilfe kann die Lipschitzkonstante einer glatten Abbildung des $\mathbb{R}^n$ in sich berechnet werden.

Satz 2.5 *Sei $D \subset \mathbb{R}^n$ nichtleer und konvex, $U \supset D$ eine offene Obermenge und $F : U \longrightarrow \mathbb{R}^n$ auf D (d. h. in jedem Punkt von D) stetig partiell differenzierbar. Für eine beliebige Norm $\|\cdot\|$ auf $\mathbb{R}^n$ bezeichne $\|\cdot\|$ auch die zugeordnete Matrixnorm auf $\mathbb{R}^{n\times n}$. Für alle $x, y \in D$ ist dann*

$$\|F(x) - F(y)\| \le \Big(\sup_{t\in[0,1]} \|F'(x + t(y-x))\| \Big) \|x - y\| \le \Big(\sup_{\xi \in D} \|F'(\xi)\| \Big) \|x - y\|.$$

Ist daher D kompakt oder sind die partiellen Ableitungen von F (bzw. ihrer Komponenten) betragsmäßig auf D nach oben beschränkt, so ist F auf D lipschitzstetig mit der Lipschitzkonstanten $L := \sup_{\xi\in D} \|F'(\xi)\|$.

Beweis: Der Beweis erfolgt offenbar sofort durch Anwendung des Mittelwertsatzes und des vorigen Lemma. □

Beispiel: Sei[16] $F : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$ definiert durch

$$F(x) := \frac{1}{2} \begin{pmatrix} \cos x_1 - \sin x_2 \\ \sin x_1 + \cos x_2 \end{pmatrix}.$$

Dann ist

$$F'(x) = \frac{1}{2} \begin{pmatrix} -\sin x_1 & -\cos x_2 \\ \cos x_1 & -\sin x_2 \end{pmatrix}.$$

Dann ist

$$F'(x)^T F'(x) = \frac{1}{4} \begin{pmatrix} 1 & \sin x_1 \cos x_2 - \cos x_1 \sin x_2 \\ \sin x_1 \cos x_2 - \cos x_1 \sin x_2 & 1 \end{pmatrix},$$

[16] Dieses Beispiel haben wir

R. Kress (1998, S. 101) *Numerical Analysis*. Springer, New York-Berlin-Heidelberg

entnommen.

die Eigenwerte dieser Matrix sind

$$\lambda_{1,2} = \frac{1}{4}[1 \pm (\sin x_1 \cos x_2 - \cos x_1 \sin x_2)] = \frac{1}{4}[1 \pm \sin(x_1 - x_2)].$$

Daher ist

$$\|F'(x)\|_2 = \sqrt{\rho(F'(x)^T F'(x))} \leq \frac{1}{2}\sqrt{1 + |\sin(x_1 - x_2)|} \leq \frac{1}{\sqrt{2}} \approx 0.7071.$$

Etwas einfacher hätten wir dies erhalten, wenn wir benutzt hätten, dass die Spektralnorm durch die sogenannte *Frobenius-Norm* majorisiert wird. Für jedes $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ gilt genauer (siehe Aufgabe 7)

$$\|A\|_2 = \sqrt{\rho(A^T A)} \leq \|A\|_F := \Big(\sum_{i,j=1}^{n} a_{ij}^2\Big)^{1/2}.$$

Jedenfalls wissen wir nun, dass die obige Abbildung auf $D := \mathbb{R}^2$ kontrahiert und daher wegen des Banachschen Fixpunktsatzes genau einen Fixpunkt besitzt. Mit Maple erhalten wir

```
> fsolve({x_1=0.5*(cos(x_1)-sin(x_2)),x_2=0.5*(sin(x_1)+cos(x_2))},{x_1,x_2});
```

$$\{x\_1 = .2290592672,\ x\_2 = .5418967160\}$$

Wie die verhältnismäßig große Kontraktionskonstante erwarten lässt, ist die Konvergenz nicht überragend:

k	x_k	
0	1.00000000000000	1.00000000000000
1	−0.15058433946988	0.69088664533802
2	0.17573141646014	0.31033272214856
3	0.33961172396776	0.56353017678320
4	0.20435511866608	0.58924783619601
5	0.21172809775056	0.51714732904876
6	0.24163334928146	0.53969140150891
7	0.22853857417878	0.54857807247071
8	0.22626202725898	0.53991061134828
9	0.23022622218652	0.54104551836410
10	0.22929116792681	0.54268422834854

□

Der folgende Satz ist ein lokaler Konvergenzsatz. Es werden hinreichende Bedingungen an einen Fixpunkt angegeben, um wenigstens lokale Konvergenz (bei Start in einer hinreichend kleinen Umgebung des Fixpunktes) zu sichern.

Satz 2.6 *Die Abbildung $F : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ besitze den Fixpunkt x^* und sei in einer Umgebung von x^* stetig partiell differenzierbar. Sei $\|\cdot\|$ eine Norm auf dem $\mathbb{R}^n$ bzw. die zugeordnete Matrixnorm. Ist dann $\|F'(x^*)\| < 1$, so existiert ein $\delta > 0$ derart, dass die Folge $\{x_k\}$ mit $x_{k+1} := F(x_k)$ für jedes $x_0 \in \mathbb{R}^n$ mit $\|x_0 - x^*\| \leq \delta$ gegen x^* konvergiert.*

Beweis: Man definiere die positive Zahl $\epsilon := (1-\|F'(x^*)\|)/2$. Da F in einer Umgebung von x^* stetig partiell differenzierbar ist, existiert ein $\delta > 0$ mit:

1. F ist auf einer offenenen Obermenge der Kugel

$$B[x^*;\delta] := \{x \in \mathbb{R}^n : \|x - x^*\| \leq \delta\}$$

stetig partiell differenzierbar.

2. Es ist $\|F'(x) - F'(x^*)\| \leq \epsilon$ für alle $x \in B[x^*;\delta]$.

Für $x \in B[x^*;\delta]$ erhält man aus dem Mittelwertsatz und Lemma 2.4, dass

$$\begin{aligned}
\|F(x) - F(x^*) - F'(x^*)(x-x^*)\| &= \left\| \int_0^1 [F'(x^* + t(x-x^*)) - F'(x^*)](x-x^*)\,dt \right\| \\
&\leq \int_0^1 \|F'(\underbrace{x^* + t(x-x^*)}_{\in B[x^*;\delta]}) - F'(x^*)\|\,dt\,\|x-x^*\| \\
&\leq \epsilon\,\|x-x^*\|.
\end{aligned}$$

Berücksichtigt man noch, dass x^* ein Fixpunkt von F ist, so folgt für alle $x \in B[x^*;\delta]$, dass

$$\begin{aligned}
\|F(x) - x^*\| &= \|F(x) - F(x^*)\| \\
&\leq \|F(x) - F(x^*) - F'(x^*)(x-x^*)\| + \|F'(x^*)\|\,\|x-x^*\| \\
&\leq [\epsilon + \|F'(x^*)\|]\,\|x-x^*\| \\
&= q\,\|x-x^*\|,
\end{aligned}$$

wobei $q := (1 + \|F'(x^*)\|)/2 \in (0,1)$. Insbesondere folgt hieraus: Ist $x_0 \in B[x^*;\delta]$, so ist $\|x_k - x^*\| \leq q^k\,\|x_0 - x^*\|$, $k = 0, 1, \ldots$. Hieraus folgt die Konvergenz der Folge $\{x_k\}$ gegen x^*, der Satz ist bewiesen. □

Bemerkung: Es würde im letzten Satz genügen, $\rho(F'(x^*)) < 1$ vorauszusetzen (statt $\|F'(x^*)\| < 1$ mit einer zugeordneten Matrixnorm). Denn es gilt die folgende Aussage (siehe z. B. J. Werner (1992, S. 23 ff.)):

- *Seien $A \in \mathbb{R}^{n\times n}$ und $\epsilon > 0$ gegeben. Dann existiert eine Vektornorm auf dem $\mathbb{R}^n$ derart, dass mit der zugeordneten Matrixnorm $\|A\| \leq \rho(A) + \epsilon$ gilt.*

Die Bedingung $\rho(F'(x^*)) < 1$ ist schwächer als $\|F'(x^*)\| < 1$ mit einer zugeordneten Matrixnorm (Beweis?). □

2.2.4 Das Newton-Verfahren für nichtlineare Gleichungssysteme

Gegeben sei nun ein nichtlineares Gleichungssystem $f(x) = 0$, wobei mit einer offenen Menge $U \subset \mathbb{R}^n$ die Funktion $f : U \longrightarrow \mathbb{R}^n$ auf U stetig partiell differenzierbar ist. Das Newton-Verfahren zur Lösung von $f(x) = 0$ lautet

$$x_{k+1} := x_k - f'(x_k)^{-1} f(x_k), \qquad k = 0, 1, \ldots.$$

Hierbei ist $x_0 \in U$ ein geeigneter Startvektor. Ohne weitere Voraussetzungen ist die Durchführbarkeit des Verfahrens natürlich nicht gesichert, also die Funktionalmatrix $f'(x_k)$ existiert und nichtsingulär ist. Die Motivation (Linearisiere die nichtlineare Abbildung f in aktueller Näherung) für das Verfahren ist im Prinzip genau dieselbe wie im eindimensionalen Fall. Man beachte, dass man bei der Durchführung des Newton-Verfahrens in jedem Schritt ein nichtlineares Gleichungssystem zu lösen hat:

- Sei x_0 Startvektor.
- Für $k = 0, 1, \ldots$:
  - Bestimme p_k mit $f'(x_k)p_k = -f(x_k)$.
  - Berechne neue Näherung $x_{k+1} := x_k + p_k$.

Beispiel: Als Beispiel zum Banachschen Fixpunktsatz hatten wir die Fixpunktaufgabe $x = F(x)$ mit

$$F(x) := \frac{1}{2} \begin{pmatrix} \cos x_1 - \sin x_2 \\ \sin x_1 + \cos x_2 \end{pmatrix}$$

betrachtet. Zu lösen ist also das nichtlineare Gleichungssystem

$$f(x) := \begin{pmatrix} x_1 - \frac{1}{2}(\cos x_1 - \sin x_2) \\ x_2 - \frac{1}{2}(\sin x_1 + \cos x_2) \end{pmatrix} = 0.$$

Mit dem selben Startvektor wie bei der Fixpunktiteration geben wir einige Schritte des Newton-Verfahrens an:

k	x_k	
0	1.00000000000000	1.00000000000000
1	0.26610581354771	0.60053910584450
2	0.22701533406014	0.54716433661596
3	0.22884672177495	0.54244444945610
4	0.22903704750276	0.54195442477264
5	0.22905692483687	0.54190280439516
6	0.22905902006463	0.54189735844629
7	0.22905924112547	0.54189678380833
8	0.22905926445122	0.54189672317348
9	0.22905926691252	0.54189671677538
10	0.22905926717223	0.54189671610027

Der Unterschied zur Fixpunktiteration ist deutlich. □

Das Newton-Verfahren zeichnet sich sehr oft durch seine vorzügliche "Konvergenzgeschwindigkeit" aus. Dies wollen wir jetzt genauer fassen:

Definition 2.7 Sei $\{x_k\} \subset \mathbb{R}^n$ eine Folge, die gegen ein $x^* \in \mathbb{R}^n$ konvergiert. Für alle k sei $x_k \neq x^*$. Ferner sei $\|\cdot\|$ eine Norm auf $\mathbb{R}^n$.

1. Für ein $p \geq 1$ konvergiert die Folge $\{x_k\}$ von *mindestens p-ter Ordnung* gegen x^*, wenn eine Konstante $c > 0$ (für $p = 1$ sei $c \in (0, 1)$) mit

$$\|x_{k+1} - x^*\| \leq c \|x_k - x^*\|^p$$

   für alle hinreichend großen k existiert.

2. Die Folge $\{x_k\}$ konvergiert *superlinear* gegen x^*, wenn

$$\lim_{k\to\infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} = 0.$$

Bemerkungen: Statt von Konvergenz von mindestens erster oder mindestens zweiter Ordnung spricht man auch von linearer bzw. quadratischer Konvergenz.

Sind die Voraussetzungen des Banachschen Fixpunktsatzes erfüllt, so liegt wegen

$$\|x_{k+1} - x^*\| = \|F(x_k) - F(x^*)\| \leq q \|x_k - x^*\|$$

(hierbei ist $q \in (0, 1)$ die Kontraktionskonstante) Konvergenz von mindestens erster Ordnung bzw. lineare Konvergenz vor.

Nur die lineare Konvergenz ist normabhängig, die anderen Begriffe sind wegen der Äquivalenz der Normen normunabhängig. □

Wir wollen uns mit einem einzigen Satz zur Konvergenz des Newton-Verfahrens begnügen und geben diesen auch noch ohne Beweis an (Beweis z. B. bei J. Werner (1992, S. 102 ff.)). Es ist ein lokaler Konvergenzsatz wie Satz 2.1 und verallgemeinert diesen auf den mehrdimensionaslen Fall.

Satz 2.8 *Die Abbildung $f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ besitze in $x^* \in \mathbb{R}^n$ eine Nullstelle, es sei also $f(x^*) = 0$. Es gelte:*

1. *Die Abbildung f ist auf einer Umgebung von x^* stetig differenzierbar.*

2. *Die Funktionalmatrix $f'(x^*) \in \mathbb{R}^{n\times n}$ ist nichtsingulär.*

Es bezeichne $\|\cdot\|$ eine beliebige Norm auf dem $\mathbb{R}^n$ bzw. die zugeordnete Matrixnorm. Dann existiert ein $\delta > 0$ derart, dass für jedes $x_0 \in \mathbb{R}^n$ mit $\|x_0 - x^*\| \leq \delta$ die durch das Newton-Verfahren

$$x_{k+1} := x_k - f'(x_k)^{-1} f(x_k), \qquad k = 0, 1, \ldots,$$

gewonnene Folge definiert ist (d. h. $f'(x_k)$ existiert und ist nichtsingulär für $k = 0, 1, \ldots$) und superlinear gegen x^* konvergiert. Gilt sogar

3. *$f'(\cdot)$ ist auf einer (hinreichend kleinen) Kugel um x^* in x^* lipschitzstetig, d. h. es existieren $\eta > 0$ und $L > 0$ mit*

$$\|f'(x) - f'(x^*)\| \leq L \|x - x^*\| \qquad \text{für alle } x \text{ mit } \|x - x^*\| \leq \eta,$$

so konvergiert $\{x_k\}$ *bei hinreichend kleinem* $\delta > 0$ *für jedes* x_0 *mit* $\|x_0 - x^*\| \leq \delta$ *von mindestens zweiter Ordnung gegen* x^*.

Wir geben nun eine sehr einfache MATLAB-Funktion `NewtonGLS` zur Lösung eines nichtlinearen Gleichungssystems an:

```
function [x,iter]=NewtonGLS(fun,x_0,tol,max_iter);
%*********************************************************************
%Input-Parameter:
%     fname       Name einer Funktion fun. [f,f_strich]=fun(x)
%                 ergibt Funktionswert und Funktionalmatrix der
%                 Funktion in x
%     x_0         Startwert
%     tol         positiver Wert, Abbruch wenn ||f||_2<tol
%     max_iter    maximale Anzahl von Iterationen
%Output-Parameter:
%     x           gefundene L"osung (bei Erfolg)
%     iter        Anzahl der durchgefuehrten Iterationen
%*********************************************************************
  iter=1;
  x=x_0;  [f,f_strich]=feval(fun,x);
  while (norm(f)>tol)&(iter<max_iter)
     iter=iter+1;
     p=-f_strich\f;  x=x+p;
     [f,f_strich]=feval(fun,x);
  end;
%*********************************************************************
```

Beispiel: Wir wollen das nichtlineare Gleichungssystem

$$f(x) := \begin{pmatrix} e^{x_1}\cos x_2 - x_1 \\ e^{x_1}\sin x_2 - x_2 \end{pmatrix} = 0$$

lösen. Dann ist $z = x_1 + ix_2$ eine komplexe Lösung von $e^z - z = 0$. Wir schreiben zunächst ein file Funexp.m, in welchem f und die Funktionalmatrix $f'(x)$ berechnet wird.

```
function [f,f_strich]=Funexp(x);
e_1=exp(x(1));c_1=cos(x(1));c_2=cos(x(2));s_1=sin(x(1));s_2=sin(x(2));
f=[e_1*c_2-x(1);e_1*s_2-x(2)];
f_strich=[e_1*c_2-1,-e_1*s_2;e_1*s_2,e_1*c_2-1];
```

Nach

```
[x,iter]=NewtonGLS(@Funexp,[1;1],1e-12,20);
```

ist

$$x = \begin{pmatrix} 0.31813150520475 \\ 1.33723570143070 \end{pmatrix}, \qquad \text{iter} = 6$$

das Resultat. □

2.2.5 Aufgaben

1. Gegeben[17] sei das nichtlineare Gleichungssystem

$$f(x_1, x_2) := \begin{pmatrix} x_2 \exp(x_1) - 2 \\ x_1^2 + x_2 - 4 \end{pmatrix} = 0.$$

   Durch Elimination von x_2 führe man diese Aufgabe auf die Bestimmung der Schnittpunkte zweier Funktionen in einer unabhängigen Variablen zurück. Mit Hilfe von Maple oder MATLAB bestimme man Näherungen für Lösungen des gegebenen nichtlinearen Gleichungssystems.

2. Als Variante zu Satz 2.2 zeige man: Sei $f \in C^2[a, b]$ mit $f'(x) > 0$ und $f''(x) \leq 0$ für alle $x \in [a, b]$, d. h. die auf $[a, b]$ zweimal stetig differenzierbare Funktion f sei auf $[a, b]$ streng monoton wachsend und konkav. Ferner sei $f(a) < 0 < f(b)$ und $b - f(b)/f'(b) \geq a$. Dann liefert das Newton-Verfahren

$$x_{k+1} := x_k - \frac{f(x_k)}{f'(x_k)}, \qquad k = 0, 1, \ldots$$

   für jedes $x_0 \in [a, b]$ eine Folge $\{x_k\}$ mit der Eigenschaft, dass $\{x_k\}_{k\in\mathbb{N}}$ monoton wachsend und von mindestens zweiter Ordnung gegen die einzige Nullstelle x^* von f in $[a, b]$ konvergiert, wobei natürlich $x_k \neq x^*$ für alle k angenommen wird.

3. Mit Hilfe von Aufgabe 2 überlege man sich, für welche Startwerte das Newton-Verfahren, angewandt auf die Nullstellenaufgabe $f(x) := x - e^{-x} = 0$, eine konvergente Folge bildet. Mit $x_0 := 1.0$ berechne man die ersten fünf Iterierten.

4. Die Funktion $f(x) := xe^{-x}$ hat $x^* = 0$ als einzige Nullstelle. Man untersuche, für welche Startwerte das Newton-Verfahren eine gegen x^* konvergente Folge liefert.

5. Durch $f_0 := 1$, $f_1 := 1$ und $f_{k+1} := f_k + f_{k-1}$, $k = 1, 2, \ldots$, sei die Fibonacci-Folge $\{f_k\}$ definiert. Ferner sei $\tau := (1 + \sqrt{5})/2$.

   (a) Man beweise die Binetsche Formel

$$f_k = \frac{1}{\sqrt{5}}[\tau^{k+1} - (-\tau)^{-(k+1)}], \qquad k = 0, 1, \ldots.$$

   (b) Man zeige, dass

$$\lim_{k\to\infty} \frac{f_{k+1}}{f_k} = \tau.$$

6. Gegeben sei die Nullstellenaufgabe $f(x) := x + \ln x = 0$. Durch einen Plot von f über dem Intervall $[0.3, 1]$ verschaffe man sich einen Überblick über eventuelle Nullstellen. Mit Hilfe der Maple-Funktion `fsolve` oder die MATLAB-Funktion `fzero` bestimme man Näherungslösungen.

7. Die Frobeniusnorm $\|\cdot\|_F$ auf $\mathbb{R}^{n\times n}$ ist für $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ definiert durch $\|A\|_F := (\sum_{i,j=1}^n a_{ij}^2)^{1/2}$. Man zeige:

[17] Siehe

N. Köckler (1990, S. 138) *Numerische Algorithmen in Softwaresystemen*. B. G. Teubner, Stuttgart.

(a) Es ist $\|A\|_F = \sqrt{\text{Spur}(A^T A)}$ für jedes $A \in \mathbb{R}^{n\times n}$. Hierbei ist die Spur einer Matrix die Summe ihrer Diagonalelemente.

(b) Die Frobeniusnorm $\|\cdot\|_F : \mathbb{R}^{n\times n} \longrightarrow \mathbb{R}$ hat die Eigenschaften einer Norm (Definitheit, Homogenität und Dreiecksungleichung) und ist darüberhinaus *submultiplikativ*, d. h. es ist $\|AB\| \le \|A\|\,\|B\|$ für alle $A, B \in \mathbb{R}^{n\times n}$.

(c) Es ist $\|A\|_2 \le \|A\|_F$ für alle $A \in \mathbb{R}^{n\times n}$.

8. Man beweise, dass

$$\lim_{k\to\infty} \underbrace{\sqrt{2+\sqrt{2+\cdots+\sqrt{2}}}}_{k \text{ Wurzeln}} = 2.$$

9. Gegeben sei das nichtlineare Gleichungssystem

$$f(x) := \begin{pmatrix} x_1 - 0.1x_1^2 - \sin x_2 \\ x_2 - \cos x_1 - 0.1x_2^2 \end{pmatrix} = 0.$$

(a) Aus irgendeinem Grund vermuten Sie, dass das nichtlineare Gleichungssystem $f(x) = 0$ eine Lösung in $[0,1] \times [0,1]$ besitzt. Verschaffen Sie sich durch den `implicitplot` Befehl im `plots`-package von Maple eine Näherung.

(b) Benutzen Sie `fsolve` in Maple, um das System $f(x) = 0$ zu lösen. Bestimmen Sie mehr als eine Lösung.

(c) Benutzen Sie eine selbst (z. B. in MATLAB) geschriebene Funktion, um das Gleichungssystem zu lösen.

10. Gegeben[18] sei die Abbildung $f : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$ mit

$$f(x) := \begin{pmatrix} \exp(x_1^2 + x_2^2) - 3 \\ x_1 + x_2 - \sin(3(x_1 + x_2)) \end{pmatrix}.$$

Man bestimme die Funktionalmatrix $f'(x)$. Für welche x ist $f'(x)$ singulär?

[18] Diese Aufgabe findet man bei

J. Stoer (1994, S. 356) *Numerische Mathematik 1*. Springer, Berlin-Heidelberg-New York.