Günter Bärwolff

Numerik
für Ingenieure, Physiker und Informatiker

- Aufgabenlösungen
- Appendix
- Errata

März 2020

Aufgabenlösungen

Nachfolgend sind die Lösungen der in der Monographie "Numerik für Naturwissenschaftler und Ingenieure" gestellten Aufgaben kapitelweise zu finden. Um die Lösungsschritte nachvollziehbar zu machen, wurden die einzelnen Aufgabenlösungen möglichst umfangreich behandelt.

Stand März 2020 G. Bärwolff

Kapitel 1

1) Mit dem Gauß'schen Algorithmus findet man die Lösung

$$x = 1 - \frac{0{,}71}{\sqrt{2} - 1{,}42} = \frac{\sqrt{2} - 2{,}13}{\sqrt{2} - 1{,}42} = 2\frac{71 - 25\sqrt{2}}{71 - 50\sqrt{2}}, \quad y = \frac{2}{\sqrt{2} - 1{,}42} = -\frac{100}{71 - 50\sqrt{2}} \; .$$

Für die Berechnung von x bzw. y in Abhängigkeit von der Genauigkeit von $t = \sqrt{2}$ sind die Fehlerverstärkungsfaktoren zu bestimmen. Für

$$x(t) = 2\frac{71 - 25t}{71 - 50t} \quad \text{und} \quad y(t) = -\frac{100}{71 - 50t}$$

erhält man die absoluten Fehlerverstärkungsfaktoren

$$x'(t) = \frac{3550}{(71 - 50t)^2}, \; x'(t)|_{t \approx \sqrt{2}} \approx 42410, \quad y'(t) = -\frac{5000}{(71 - 50t)^2}, \; y'(t)|_{t \approx \sqrt{2}} \approx 59732 \; .$$

Die relativen Fehlerverstärkungsfaktoren ergeben sich zu

$$\frac{tx'(t)}{x(t)} = \frac{t3550}{(71 - 50t)(71 - 25t)}, \; \frac{tx'(t)}{x(t)}|_{t \approx \sqrt{2}} \approx 487, \quad \frac{ty'(t)}{y(t)} = \frac{t50}{71 - 50t}, \; \frac{ty'(t)}{y(t)}|_{t \approx \sqrt{2}} \approx 244 \; .$$

Die konkrete Berechnung von x bzw. y ergibt für unterschiedliche Näherungen der Zahl $\sqrt{2} = 1{,}4142135\ldots$

$$\begin{array}{lll} \sqrt{2} \approx 1{,}41 & x = 143{,}000\ldots \; , & y = -200{,}000\ldots \\ \sqrt{2} \approx 1{,}414 & x = 237{,}666\ldots \; , & y = -333{,}333\ldots \\ \sqrt{2} \approx 1{,}4142 & x = 245{,}827\ldots \; , & y = -344{,}827\ldots \; . \end{array}$$

Im Vergleich mit den nahezu exakten Lösungen

$$x = 2\frac{71 - 25\sqrt{2}}{71 - 50\sqrt{2}} \approx 246{,}4014182\ldots, \quad y = -\frac{100}{71 - 50\sqrt{2}} \approx -345{,}6358003\ldots$$

erkennt man die fatalen Auswirkungen der zu groben Näherungen für die Zahl $\sqrt{2}$ auf die Berechnung von x und y.
Die Erweiterung der Beziehungen für $x(t)$ und $y(t)$ mit dem Ausdruck $71 + 50t$ und die Berücksichtigung von $t^2 = 2$ ergibt die Berechnungsformeln

$$x(t) = \frac{5082}{41} + \frac{3550}{41}t \; , \quad y(t) = -\frac{7100}{41} - \frac{5000}{41}t \; ,$$

die wesentlich kleinere absolute und relative Fehlerverstärkungsfaktoren besitzen und für die obigen Näherungen von $\sqrt{2}$ mit

$$\begin{array}{lll} \sqrt{2} \approx 1{,}41 & x = 246{,}036\ldots \; , & y = -345{,}121\ldots \\ \sqrt{2} \approx 1{,}414 & x = 246{,}382\ldots \; , & y = -345{,}609\ldots \\ \sqrt{2} \approx 1{,}4142 & x = 246{,}400\ldots \; , & y = -345{,}634 \; . \end{array}$$

wesentlich stabilere Ergebnisse ergeben.

2) Man unterteilt die Berechnung von $x = b - \sqrt{b^2 - c}$ in die vier Schritte
a) $t_1(b) = b^2$

b) $t_2(t_1) = t_1 - c$

c) $t(t_2) = \sqrt{t_2}$

d) $x(t) = b - t$.

In den Schritten 1 bis 3 erhält man unter Berücksichtigung von $b^2 \gg c > 0$ die Konditionszahlen

$$\begin{aligned} k_1 &= |\frac{\partial t_1}{\partial t}\frac{b}{t_1}| = |\frac{2b \cdot b}{b^2}| = 2 \\ k_2 &= |\frac{\partial t_2}{\partial t_1}\frac{t_1}{t_2}| = |\frac{t_1}{t_1 - c}| = |\frac{b^2}{b^2 - c}| \approx 1 \\ k_3 &= |\frac{\partial t}{\partial t_2}\frac{t_2}{t}| = |\frac{t_2}{2\sqrt{t_2}\sqrt{t_2}}| = \frac{1}{2} \ , \end{aligned}$$

die allesamt harmlos sind. Die Kondition des 4. Schrittes ist

$$k_4 = |\frac{\partial x}{\partial t}\frac{t}{x}| = |\frac{-t \cdot t}{x(t)}| = |\frac{-\sqrt{b^2 - c}}{b - \sqrt{b^2 - c}}| = |\frac{(b + \sqrt{b^2 - c})\sqrt{b^2 - c}}{c}| \approx \frac{2b^2}{c} \gg 1 \ ,$$

führt also insgesamt auf die Konditionszahl

$$\kappa = k_1 k_2 k_3 k_4 \approx \frac{2b^2}{c} \gg 1 \ ,$$

also ergibt sich die Instabilität der Berechnungsformel.
Im Fall $b^2 \approx c$ erkennt man, dass die Konditionzahl des 2. Schrittes mit

$$|\frac{\partial t_2}{\partial t_1}\frac{t_1}{t_2}| = |\frac{b^2}{b^2 - c}|$$

sehr groß wird, so dass auch in diesem Fall die Berechnungsformel instabil ist.
Die Verwendung der äquivalenten Berechnungsformel $x = \frac{c}{b + \sqrt{b^2 - c}}$ führt auf die Berechnungsschritte

a) $t_1(b) = b^2$

b) $t_2(t_1) = t_1 - c$

c) $t_3(t_2) = \sqrt{t_2}$

d) $t(t_3) = b + t_3$

e) $x(t) = \frac{c}{t}$.

Als Konditionszahlen für die 5 Schritte erhält man

$$k_1 = 2, \ k_2 = 1, \ k_3 = \frac{1}{2}, \ k_4 = |\frac{b\sqrt{b^2 - c}}{b + \sqrt{b^2 - c}}|, \ k_5 = 1 \ ,$$

so dass die Konditionszahl

$$\kappa = k_1 k_2 k_3 k_4 k_5 = |\frac{b\sqrt{b^2 - c}}{b + \sqrt{b^2 - c}}| \approx \begin{cases} 0 \ , & b^2 \approx c \\ \frac{b}{2} \ , & b^2 \gg c > 0 \end{cases}$$

klein bleibt und die Berechnungsformel damit stabil ist.

3) Die Berechnung wird mit den Schritten

a) $z_1(x) = x^2$

b) $z_2(z_1) = z_1 + 1$
c) $z_3(z_2) = \sqrt{z_2}$
d) $z_4(z_3, x) = z_3 + x$
e) $z(z_4) = \frac{1}{z_4}$
durchgeführt. Es ergeben sich die Fehler

$$\begin{aligned}
\epsilon_{z_1} &= \frac{\partial z_1}{\partial x}\frac{x}{z_1}\epsilon_x + \tau_{z_1} = 2\epsilon_x + \tau_{z_1} \\
\epsilon_{z_2} &= \frac{\partial z_2}{\partial z_1}\frac{z_1}{z_2}\epsilon_{z_1} + \tau_{z_2} = \frac{x^2}{x^2+1}\epsilon_{z_1} + \tau_{z_2} \\
\epsilon_{z_3} &= \frac{\partial z_3}{\partial z_2}\frac{z_2}{z_3}\epsilon_{z_2} + \tau_{z_3} = \frac{1}{2}\epsilon_{z_2} + \tau_{z_3} \\
\epsilon_{z_4} &= \frac{\partial z_4}{\partial z_3}\frac{z_3}{z_4}\epsilon_{z_3} + \frac{\partial z_4}{\partial x}\frac{x}{z_4}\epsilon_x + \tau_{z_4} = \frac{\sqrt{x^2+1}}{\sqrt{x^2+1}+x}\epsilon_{z_3} + \frac{x}{\sqrt{x^2+1}+x}\epsilon_x + \tau_{z_4} \\
\epsilon_z &= \frac{\partial z}{\partial z_4}\frac{z_4}{z}\epsilon_{z_4} + \tau_z = -\epsilon_{z_4} + \tau_z \ .
\end{aligned}$$

Damit ergibt sich der Gesamtfehler

$$\begin{aligned}
\epsilon_z &= -\left(\frac{\sqrt{x^2+1}}{\sqrt{x^2+1}+x}\frac{x^2}{x^2+1} + \frac{x}{\sqrt{x^2+1}+x}\right)\epsilon_x \\
&\quad -\frac{\sqrt{x^2+1}}{\sqrt{x^2+1}+x}\left(\frac{1}{2}\tau_{z_1} + \frac{1}{2}\tau_{z_2} + \tau_{z_3}\right) + \tau_{z_3} - \tau_{z_4} + \tau_z \ .
\end{aligned}$$

Für den Betrag des Gesamtfehlers erhält man im Fall von $x^2 \gg 1$ mit $|\tau_{z_1}|, |\tau_{z_2}|, |\tau_{z_3}|, |\tau_{z_4}|, |\tau_z| \leq \tau$ die Abschätzung

$$|\epsilon_z| \leq |\epsilon_x| + 4\tau \ .$$

Im Unterschied dazu ergibt die äquivalente Berechnungsformel $z = \sqrt{x^2+1} - x$ im Fall $x^2 \gg 1$ einen wesentlich größeren Gesamtfehler. Probieren Sie es aus.

4) Die Berechnung von $w = (\sqrt{x})^3$ bedeutet
a) $w_1(x) = \sqrt{x}$
b) $w(w_1) = w_1^3$.
Mit $\frac{\partial w_1}{\partial x}\frac{x}{w_1} = \frac{1}{2}$ und $\frac{\partial w}{\partial w_1}\frac{w_1}{w} = 3$ erhält man

$$\epsilon_{w_1} = \frac{1}{2}\epsilon_x + \tau_{w_1}, \quad \epsilon_w = 3\epsilon_{w_1} + \tau_w = \frac{3}{2}\epsilon_x + \frac{3}{2}\tau_{w_1} + \tau_w \ .$$

Die Berechnung von $w = \sqrt{x^3}$ bedeutet
a) $w_1(x) = x^3$
b) $w(w_1) = \sqrt{w_1}$.
Mit $\frac{\partial w_1}{\partial x}\frac{x}{w_1} = 3$ und $\frac{\partial w}{\partial w_1}\frac{w_1}{w} = \frac{1}{2}$ erhält man

$$\epsilon_{w_1} = 3\epsilon_x + \tau_{w_1}, \quad \epsilon_w = 3\epsilon_{w_1} + \tau_w = \frac{3}{2}\epsilon_x + \frac{1}{2}\tau_{w_1} + \tau_w \ .$$

Damit ist der Gesamtfehler bei der Berechnung von w mit der Formel $w = \sqrt{x^3}$ geringer als mit der Formel $w = (\sqrt{x})^3$.

5) Es gilt für reelle Zahlen b offensichtlich

$$0 \leq b - [b] < 1 \ .$$

Es ist

$$\begin{aligned} a_n &= [(a - a_1E^{-1} - \cdots - a_{n-2}E^{-(n-2)} - a_{n-1}E^{-(n-1)})E^n] \\ &= [(a - a_1E^{-1} - \cdots - a_{n-2}E^{-(n-2)})E^n - [(a - a_1E^{-1} - \cdots - a_{n-2}E^{-(n-2)})E^{n-1}]E^{-(n-1)}E^n] \\ &= [\{(a - a_1E^{-1} - \cdots - a_{n-2}E^{-(n-2)})E^{n-1} - [(a - a_1E^{-1} - \cdots - a_{n-2}E^{-(n-2)})E^{(n-1)}]\}E] \\ &=: [\{\beta - [\beta]\}E] =: [\alpha E] \end{aligned}$$

mit $0 \leq \alpha < 1$. Damit ist $a_n \geq 0$ und $a_n = [\alpha E] < E$ und die erste Behauptung ist bewiesen.
Mit $a_{n+1} = [(a - \sum_{j=1}^{n} a_j E^{-j})E^{n+1}] =: [\alpha] < E$ ist offensichtlich auch

$$(a - \sum_{j=1}^{n} a_j E^{-j})E^{n+1} < E \ ,$$

da ja der Fall $\alpha = [\alpha]$ möglich ist. Daraus folgt

$$(a - \sum_{j=1}^{n} a_j E^{-j})E^{n+1} < E \iff a - \sum_{j=1}^{n} a_j E^{-j} < E^{-n} \ .$$

Aus $0 \leq a_{n+1} = [\alpha] \leq \alpha$ folgt insgesamt

$$0 \leq a - \sum_{j=1}^{n} a_j E^{-j} < E^{-n}$$

und damit die zweite Behauptung.

6) Für die inversen Matrizen errechnet man

$$A^{-1} = \begin{pmatrix} \frac{3}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{3}{4} \end{pmatrix} \quad \text{bzw.} \quad B^{-1} \approx \begin{pmatrix} 10^{-4} & 1 \\ 10^{-4} & -10^{-8} \end{pmatrix} \ .$$

Mit der Spaltensummennorm ergeben sich die Konditionszahlen

$$\text{cond}(A) = ||A||_1||A^{-1}||_1 = 4{\cdot}2 = 8 \quad \text{bzw.} \quad \text{cond}(B) = ||B||_1||B^{-1}||_1 \approx 10001{\cdot}1 = 10001 \ .$$

Für die Zeilensummennorm ergeben sich die Konditionszahlen

$$\text{cond}(A) = ||A||_\infty||A^{-1}||_\infty = 4{\cdot}2 = 8 \quad \text{bzw.} \quad \text{cond}(B) = ||B||_\infty||B^{-1}||_\infty \approx 10000{\cdot}1 = 10000 \ .$$

Kapitel 2

1) Gauß'scher Algorithmus:

$$\left(\begin{array}{cccc|c} 1 & -1 & 2 & -1 & 2 \\ -2 & 4 & 1 & 3 & -5 \\ 1 & -3 & -3 & -2 & 3 \end{array}\right) \Longrightarrow \left(\begin{array}{cccc|c} 1 & -1 & 2 & -1 & 2 \\ 0 & 2 & 5 & 1 & -1 \\ 0 & -2 & -5 & -1 & 1 \end{array}\right) \Longrightarrow \left(\begin{array}{cccc|c} 1 & -1 & 2 & -1 & 2 \\ 0 & 2 & 5 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right)$$

Damit stimmen der Rang der Matrix A und der Rang der erweiterten Matrix $A|b$ überein (ist gleich 2), so dass das Gleichungssystem lösbar ist. Als Lösung findet man mit den Parametern $x_3 = s$ und $x_4 = t$

$$x_2 = -1 - 5s - t\,, \quad x_1 = 3 - 2s + t + x_2 = 3 - 2s + t - 1 - 5s - t = 2 - 7s\,, \quad s,t \in \mathbb{R}\,.$$

2) Als Nebenprodukt des Gauß'schen Algorithmus

$$A = \begin{pmatrix} 4 & 2 & 1 \\ 1 & 4 & 2 \\ 2 & 2 & 4 \end{pmatrix} \Longrightarrow \begin{pmatrix} 4 & 2 & 1 \\ 0 & \frac{7}{2} & \frac{7}{4} \\ 0 & 1 & \frac{7}{2} \end{pmatrix} \Longrightarrow \begin{pmatrix} 4 & 2 & 1 \\ 0 & \frac{7}{2} & \frac{3}{4} \\ 0 & 0 & 3 \end{pmatrix}$$

erhält man durch die Berücksichtung der Faktoren zur Ezeugung von Nullspalten unter den Kopfelementen

$$L = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{4} & 1 & 0 \\ \frac{1}{2} & \frac{2}{7} & 1 \end{pmatrix} \quad \text{und} \quad R = \begin{pmatrix} 4 & 2 & 1 \\ 0 & \frac{7}{2} & \frac{7}{4} \\ 0 & 0 & 3 \end{pmatrix}$$

Für die Matrix B findet man

$$B = \begin{pmatrix} 4 & 2 & 4 \\ 2 & 4 & 2 \\ 4 & 2 & 4 \end{pmatrix} \Longrightarrow \begin{pmatrix} 4 & 2 & 4 \\ 0 & 3 & 0 \\ 0 & 0 & 0 \end{pmatrix}\,,$$

also für L und R

$$L = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \quad \text{und} \quad R = \begin{pmatrix} 4 & 2 & 4 \\ 0 & 3 & 0 \\ 0 & 0 & 0 \end{pmatrix}\,.$$

3) Ausgehend vom Programm **lr_gb.m** erhält man das nachfolgende Programm, das die Aufgabe löst.

```
# Programm aufg23 zur LR–Zerlegung einer quadratischen Matrix A
# input:  Matrix a
# output: untere Dreiecksmatrix ll mit Einsen auf der Diagonale,
#         obere Dreiecksmatrix rr
#         z Permutationsmatrix entsprechend der Zeilenvertauschungen
#         es gilt  z*a = ll*rr
# aufruf: [ll,rr,z] = aufg23(a);
function [ll,rr,z] = aufg23(a);
n = length(a(:,1));
z = eye(n);
ll = eye(n);
rr = zeros(n,n);
# Elimination
for j=1:n-1
# Pivotsuche
  apivot=abs(a(j,j));
  p=j;
  for l=j+1:n
```

```
    if (abs(a(l,j)) > abs(apivot))
      apivot=a(l,j);
      p=l;
    endif
  endfor
  if (apivot ~= 0)
# Vertauschung der Zeilen j und p
    for i=1:n
      c=a(j,i);
      a(j,i)=a(p,i);
      a(p,i)=c;
    endfor
    for i=1:n
      pp=z(j,i);
      z(j,i)=z(p,i);
      z(p,i)=pp;
    endfor
# Elimination
    for i=j+1:n
      a(i,j)=a(i,j)/a(j,j);
      for k=j+1:n
        a(i,k)=a(i,k)-a(i,j)*a(j,k);
      endfor
    endfor
  endif
endfor
#
idet = 0;
amin = 10000;
for j=1:n
  if (a(n,j) ~= 0)
    idet = idet + 1;
  endif
  if (abs(a(n,j)) < amin)
    amin = abs(a(n,j));
  endif
endfor
if (idet == 0 || amin < 1.0e-10)
  disp('Matrix singulaer');
endif
#
for i=1:n
  for j=1:n
    if (i <= j)
      rr(i,j) = a(i,j);
    endif
    if (i > j)
      ll(i,j) = a(i,j);
    endif
  endfor
endfor
endfunction
```

4) Als Verallgemeinerung des Programms **aufg23.m** erhält man das nachfolgende Programm, das die Aufgabe löst.

```
# Programm aufg24 zur LR-Zerlegung und Rangberechnung einer Matrix A
# input:  Matrix a(n,m), m>n
# output: untere Dreiecksmatrix ll(n,n) mit Einsen auf der Diagonale,
#         obere Dreiecks- bzw. Echolonmatrix rr(n,m)
#         z Permutationsmatrix entsprechend der Zeilenvertauschungen
#         es gilt  z*a = ll*rr
#         ra Rang der Matrix a
# aufruf: [ll,rr,z,ra] = aufg24(a);
function [ll,rr,z,ra] = aufg24(a);
n = length(a(:,1));
m = length(a(1,:));
z = eye(n);
ll = eye(n);
rr = zeros(n,m);
# Elimination
for j=1:m-1
# Pivotsuche
  apivot=abs(a(j,j));
```

```
  p=j;
  for l=j+1:n
    if (abs(a(l,j)) > abs(apivot))
      apivot=a(l,j);
      p=l;
    endif
  endfor
  if (apivot ~= 0)
# Vertauschung der Zeilen j und p
    for i=1:m
      c=a(j,i);
      a(j,i)=a(p,i);
      a(p,i)=c;
    endfor
    for i=1:n
      pp=z(j,i);
      z(j,i)=z(p,i);
      z(p,i)=pp;
    endfor
# Elimination
    for i=j+1:n
      a(i,j)=a(i,j)/a(j,j);
      for k=j+1:m
        a(i,k)=a(i,k)-a(i,j)*a(j,k);
      endfor
    endfor
  endif
endfor
#
if (n == m)
  idet = 0;
  amin = 10000;
  for j=1:n
    if (a(n,j) ~= 0)
      idet = idet + 1;
    endif
    if (abs(a(n,j)) < amin)
      amin = abs(a(n,j));
    endif
  endfor
  if (idet == 0 || amin < 1.0e-10)
    disp('Matrix singulaer');
  endif
endif
#
for i=1:n
  for j=1:m
    if (i <= j)
      rr(i,j) = a(i,j);
    endif
  endfor
endfor
for i=1:n
  for j=1:n
    if (i > j)
      ll(i,j) = a(i,j);
    endif
  endfor
endfor
#
ra = 0;
for i=1:n
  if (abs(rr(i,m)) > 1.0e-10)
    ra = ra + 1;
  endif
endfor
endfunction
```

5) Die Entwicklung der Determinante der Permutationsmatrix P_{ij} gemäß dem Determinantenentwicklungssatz nach der i-ten Zeile oder j-ten Spalte ergibt

$$\det(P_{ij}) = (-1)^{i+j} \det(E_{n-1}) \,,$$

wobei E_{n-1} eine Einheitsmatrix vom Typ $(n-1) \times (n-1)$ ist, also die Determinante 1 hat.

Damit gilt $\det(P_{ij}) = (-1)^{i+j}$.

6) Die Erzeugung einer Nullspalte unter einem Diagonalelement einer Bandmatrix A mit dem Gauß'schen Algorithmus, z.B. in der j-ten Spalte wird durch die sukkzessive Multiplikation einer Matrix $\tilde{A}$ von links mit Elementarmatrizen der Form

$$L_{ij}(\lambda) = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & & \ddots & & \\ & & \lambda & & \ddots & \\ & & & & & 1 \end{pmatrix} =: (\lambda_{l\nu}) \tag{1}$$

erreicht, wobei für j die Bedingung $i < j < i + p$ gilt, d.h. L ist ebenfalls eine Bandmatrix mit der Bandbreite $2p - 1$. Von der Matrix $\tilde{A}$ weiss man, dass unter den Diagonalelementen $\tilde{a}_{11}, \ldots, \tilde{a}_{j-1\,j-1}$ bereits Nullen stehen. Die Multiplikation ergibt

$$\hat{A} = L_{ij}\tilde{A} = (\hat{a}_{lk})$$

und für das Element $\hat{a}_{lk}$ gilt

$$\hat{a}_{lk} = \sum_{\nu=1}^{n} \lambda_{l\nu}\tilde{a}_{\nu k} \; .$$

Wegen $\tilde{a}_{\nu k} = 0$ für $k < \nu$ sowie $\lambda_{l\nu} = 0$ für $l < \nu < l + p$ gilt

$$\hat{a}_{lk} = 0 \quad \text{für } k \geq l + p \text{ oder } k \leq l - p \; .$$

Kapitel 3

1) Die Aussage iv) ergibt sich wegen

$$QQ^T = QQ^{-1} = E$$

und $\det(Q) = \det(Q^T)$ sowie $\det(E) = 1$ aus dem Determinantenmultiplikationssatz. Es gilt

$$1 = \det(E) = \det(QQ^T) = \det(Q)\det(Q^T) = (\det(Q))^2 \implies \det(Q) = \pm 1 \ .$$

Wenn Q_1 und Q_2 orthogonal sind, dann ist $Q_1^{-1} = Q_1^T$ bzw. $Q_2^{-1} = Q_2^T$ und es folgt wegen der Eigenschaft $(AB)^T = B^T A^T$ für das Matrixprodukt

$$(Q_1Q_2)^T = Q_2^T Q_1^T = Q_2^{-1}Q_1^{-1} \implies (Q_1Q_2)^T Q_1Q_2 = Q_2^{-1}Q_1^{-1}Q_1Q_2 = Q_2^{-1}Q_2 = E \ ,$$

also die Aussage v) des Satzes 3.1.

2) Sei $\vec{u}$ ein Spaltenvektor aus dem $\mathbb{R}^n$ mit der Länge 1. Für die Householdermatrix $H = E - 2\vec{u}\vec{u}^T$ gilt

$$H^T = (E - 2\vec{u}\vec{u}^T)^T = E^T - 2(\vec{u}\vec{u}^T)^T = E - 2(\vec{u}^T)^T\vec{u}^T = E - 2\vec{u}\vec{u}^T = H \ .$$

Weiterhin gilt wegen $\vec{u}^T\vec{u} = 1$

$$\begin{aligned} HH^T &= (E - 2\vec{u}\vec{u}^T)(E - 2\vec{u}\vec{u}^T) \\ &= E - 2\vec{u}\vec{u}^T - 2\vec{u}\vec{u}^T + 4\vec{u}\vec{u}^T\vec{u}\vec{u}^T \\ &= E - 4\vec{u}\vec{u}^T + 4\vec{u}\vec{u}^T = E \ . \end{aligned}$$

3) Für die Ausgleichsgerade $\hat{y} = b_0 + b_1 x$ auf der Basis der Wertetabelle (x_k, y_k), $k = 1, \ldots, m$ entsteht für b_0, b_1 durch die notwendige Bedingung

$$\frac{\partial F}{\partial b_0} = \frac{\partial F}{\partial b_1} = 0$$

mit $F(b_0, b_1) = \sum_{j=1}^m (y_k - b_0 - b_1 x)^2$ das Gleichungssystem

$$\begin{pmatrix} \sum_{j=1}^m 1 & \sum_{j=1}^m x_j \\ \sum_{j=1}^m x_j & \sum_{j=1}^m x_j^2 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^m y_j \\ \sum_{j=1}^m x_j y_j \end{pmatrix} \ .$$

Als Lösung findet man

$$b_1 = \frac{\sum_{j=1}^m x_j(y_j - \bar{y})}{\sum_{j=1}^m x_j(x_j - \bar{y})} \ , \quad b_0 = \bar{y} - b_1\bar{x} \ .$$

Die Gleichheit der Ausdrücke

$$\frac{\sum_{j=1}^m x_j(y_j - \bar{y})}{\sum_{j=1}^m x_j(x_j - \bar{x})} \quad \text{und} \quad \frac{\sum_{j=1}^m (x_j - \bar{x})(y_j - \bar{y})}{\sum_{j=1}^m (x_j - \bar{x})(x_j - \bar{x})}$$

zeigt man leicht durch die jeweilige Erweiterung zum gemeinsamen Hauptnenner.

4) Entsprechend den Ausführungen im Abschnitt 3.3 kann man die Aufgabenstellung mit Hilfe der Programme **qr_gb.m** und **bestapprqr_gb.m** lösen. Im nachfolgenden Programm ist die wahlweise Regression implementiert.

```
# Programm aufg34 zur linearen bzw. logarithmisch lin. Regression
# input:  Tabelle x_11 x_12 ... x_1n  y_1
#                 ...                    , d.h. Matrix x(m,n), Vektor y(m,1)
#                x_m1 x_m2 ... x_mn  y_m
#         Wahlparameter rtyp (0 --> lin. Regr., .ne. 0 -->  log.lin. Regr.)
# output: Koeffizienten r_0, r_1,...,r_n, Residuenvektor d
# verwendete Programme: qr_gb.m, bestapprqr_gb
# aufruf: [r,d] = aufg34(x,y,rtyp);
function [r,d] = aufg34(x,y,rtyp);
m = length(x(:,1));
n = length(x(1,:)); n1 = n+1;
if (m < n || m == n)
  printf('zu wenig Messungen/Daten fuer die Regression');
  return
endif
c = zeros(m,n1);
if (rtyp == 0)
# lineare Regression
  for i=1:m
    c0(i,1)=1;
    for j=1:n
      c0(i,j+1)=x(i,j);
    endfor
  endfor
  [c,t] = qr_gb(c0);
  [r,d] = bestapprqr_gb(c,t,y);
else
  itest = 0;
  for i=1:m
    if (y(i) < 0)
      itest = 1;
    endif
    for j=1:n
      if (x(i,j) < 0)
        itest = 1;
      endif
    endfor
  endfor
  if (itest == 1)
    printf('Daten nicht vollst. positiv, Abbruch');
    return
  endif
  for i=1:m
    c0(i,1)=1;
    b(i)=log(y(i));
    for j=1:n
      c0(i,j+1)=log(x(i,j));
    endfor
  endfor
  [c,t] = qr_gb(c0);
  [r,d] = bestapprqr_gb(c,t,b);
  r(1) = exp(r(1));
endif
endfunction
```

Zur Ermittlung eines funktionalen Zusammenhangs $y = y(x)$ auf der Basis der Daten

x	y
1	2
2	8
3	20
4	35
5	80

ergeben die Octave-Kommandozeilen

```
x=[1;2;3;4;5];
y=[2;8;20;35;80];
[r,d] = aufg34(x,y,1);
```

das Ergebnis $r = (1{,}8459\,,\ 2{,}2227)$, so dass die logarithmisch lineare Regression den funk-

tionalen Zusammenhang

$$y(x) = 1{,}8459 \cdot x^{2{,}2227}$$

liefert. Die Komponenten d_k des Residuenvektors

$$\vec{d} = (0{,}080203\,,\ -0{,}074134\,,\ 0{,}059055\,,\ -0{,}138859\,,\ 0{,}191847)^T$$

geben Auskunft über die Fehler, und zwar gilt

$$d_k = ln(y_k) - ln(1{,}8459) - 2{,}2227 \cdot ln(x_k) \Longleftrightarrow y_k = e^{d_k}\, 1{,}8459 \cdot x_k^{2{,}2227}\ ,$$

wobei k für die entsprechende Zeile der Messwerttabelle steht. Bei der logarithmisch linearen Regression ergeben damit die Werte e^{d_k} eine Information über einen Störfaktor in der angenommenen Beziehung, so dass Werte nahe bei 1 auf einen sinnvollen angenommenen Zusammenhang hinweisen. Im Beispiel gilt

$$(e^{d_1}, e^{d_2}, e^{d_3}, e^{d_4}, e^{d_5}) = (1{,}08351\,,\ 0{,}92855\,,\ 0{,}94265\,,\ 0{,}87035\,,\ 1{,}21148)\ ,$$

so dass der Ansatz vernünftig ist. Im Unterschied dazu ergibt die lineare Regression, also ein Ansatz der Form

$$y(x) = r_0 + r_1 x\ ,$$

realisert durch die Kommandosequenz

```
x=[1;2;3;4;5];
y=[2;8;20;35;80];
[r,d] = aufg34(x,y,0);
```

das Resultat $r = (-25{,}9\,,\ 18{,}3)$ bzw. $y(x) = -25{,}9\ + 18{,}3\, x$ mit dem Residuenvektor

$$\vec{d} = (d_k) = (9{,}6\,,\ -2{,}7\,,\ -9\,,\ -12{,}3\,,\ 14{,}4)^T\ ,$$

also den Fehlern

$$d_k = y_k + 25{,}9 - 18{,}3\, x_k\ .$$

Daraus erkennt man, dass der lineare Ansatz $y(x) = r_0 + r_1 x$ keine akzeptable Beschreibung der Messergebnisse liefert.

5) Die Lösbarkeit des Minimumproblems folgt aus der Stetigkeit des Funktionals $F(r_0, r_1, \ldots, r_n)$ und der Beschränktheit von F nach unten durch 0. Notwendig für eine Lösung von (3.8) ist die Erfüllung des Gleichungssystems (3.9). Andererseits ist eine Lösung von (3.9) auch eine Lösung des Minimumproblems (3.8), da das Funktional konvex ist.
Zum Nachweis von Aussage c) des Satzes 3.4 ist anzumerken, dass man durch einen Messpunkt (x_1, y_1) unendlich viele Geraden legen kann, d.h. es existieren unendlich viele Lösungen ($m = 1$, $n = 1$). Hat man nur zwei Messpunkte gegeben, kann man durch diese Messpunkte unendlich viele Ebenen ($m = 2$, $n = 2$) legen etc., d.h. die jeweiligen Matrizen A des Normalgleichungssystems müssen notwendigerweise singulär sein.

Kapitel 4

1) Ist A eine Dreiecksmatrix mit den Diagonalelementen a_{jj}, $j = 1, \ldots, n$, dann ist

$$\det(A - \lambda E) = (a_{11} - \lambda)(a_{22} - \lambda) \ldots (a_{nn} - \lambda) \; ,$$

so dass die Diagonalelemente gerade die Eigenwerte sind.
Gilt $\tilde{A} = C^{-1}AC$ mit der regulären Matrix C, dann gilt

$$\begin{aligned} \det(\tilde{A} - \lambda E) &= \det(C^{-1}AC - \lambda C^{-1}C) \\ &= \det(C^{-1}(A - \lambda E)C) = \det(C^{-1}) \det(A - \lambda E) \det(C) = \det(A - \lambda E) \; , \end{aligned}$$

da $\det(C^{-1}) \det(C) = 1$ ist. Also haben $\tilde{A}$ und A die gleichen Eigenwerte.
Für die Eigenwerte von $A_\epsilon = A + \epsilon E$ gilt

$$\det(A_\epsilon - \lambda E) = \det(A + \epsilon E - \lambda E) = \det(A - (\lambda - \epsilon)E) \; ,$$

d.h. A hat die Eigenwerte μ_k, dann gilt

$$\mu_k = \lambda_k - \epsilon \iff \lambda_k = \mu_k + \epsilon \; .$$

Alle Eigenwerte einer regulären Matrix sind von null verschieden (warum ist das so, begründen Sie das). Ist A regulär und hat die Inverse A^{-1}, dann gilt für die Eigenwerte λ und die dazugehörenden Eigenvektoren $\vec{v}$

$$A\vec{v} = \lambda\vec{v} \iff A^{-1}A\vec{v} = \lambda A^{-1}\vec{v} \iff \frac{1}{\lambda}\vec{v} = A^{-1}\vec{v} \; .$$

Da die Determinante einer Matrix mit der Determinante ihrer Transponierten übereinstimmt gilt

$$\det(A - \lambda E) = \det((A - \lambda E)^T) = \det(A^T - \lambda E^T) = \det(A^T - \lambda E) \; ,$$

also haben A und A^T die gleichen Eigenwerte.

2) Die Gerschgorin-Kreise der Matrix A sind

$$K_1 = \{z \,|\, |z - 10| \leq 3\} \; , \quad K_2 = \{z \,|\, |z - 5| \leq 2\} \; ,$$

d.h. die Eigenwerte liegen in Kreisen mit dem Mittelpunkt $(10{,}0)$ und dem Radius 3 bzw. dem Mittelpunkt $(5{,}0)$ und dem Radius 2.
Die Gerschgorin-Kreise der Matrix B sind

$$K_1 = \{z \,|\, |z - 3| \leq 1\} \; , \quad K_2 = \{z \,|\, |z - 12| \leq 2\} \; , \quad K_3 = \{z \,|\, |z - 20| \leq 1 \; .$$

Da B symmetrisch ist, sind die Eigenwerte reell, d.h. für die Eigenwerte gilt:

$$\lambda_1 \in [2{,}4], \quad \lambda_2 \in [10{,}14], \quad \lambda_3 \in [19{,}21] \; .$$

Die Gerschgorin-Kreise der Matrix C sind

$$K_1 = K_3 = \{z \,|\, |z - 2| \leq 1\} \; , \quad K_2 = \{z \,|\, |z - 2| \leq 2\} \; .$$

Da A symmetrisch ist, sind die Eigenwerte reell, d.h. für die Eigenwerte gilt:

$$\lambda_{1,2,3} \in [0{,}4] \; .$$

Als Eigenwerte von B errechnet man (numerisch) die Nullstellen $\lambda_1 = 2{,}8895\ldots$, $\lambda_2 = 11{,}9865\ldots$, $\lambda_3 = 20{,}1240\ldots$ des Polynoms

$$\det(B - \lambda E) = -\lambda^3 + 35\lambda^2 - 335\lambda + 717\,.$$

Als Eigenwerte von A findet man

$$\lambda_1 = 2 - \sqrt{2}\,, \quad \lambda_2 = 2\,, \quad \lambda_3 = 2 + \sqrt{2}\,.$$

3) Mit der zulässigen Wahl von $\vec{z} = (0{,}0{,}1)^T$ erhält man mit

$$\vec{v}_2\vec{z}^T = \begin{pmatrix} 0 & 0 & \frac{1}{2} \\ 0 & 0 & \frac{\sqrt{2}}{2} \\ 0 & 0 & \frac{1}{2} \end{pmatrix} \quad \text{und} \quad \langle \vec{v}_2, \vec{z} \rangle = \frac{1}{2}$$

für $\tilde{A}$

$$\begin{aligned} \tilde{A} &= (E - \frac{\vec{v}_2\vec{z}^T}{\langle \vec{v}_2, \vec{z} \rangle})A = (E - \begin{pmatrix} 0 & 0 & 2 \\ 0 & 0 & \sqrt{2} \\ 0 & 0 & 1 \end{pmatrix}) \begin{pmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -\sqrt{2} \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{pmatrix} = \begin{pmatrix} 2 & 0 & -2 \\ 1 & 2 - \sqrt{2} & 1 - 2\sqrt{2} \\ 0 & 0 & 0 \end{pmatrix}\,. \end{aligned}$$

Als Eigenwerte von $\tilde{A}$ findet man

$$\lambda_1 = 0\,, \quad \lambda_2 = 2\,, \quad \lambda_3 = 2 - \sqrt{2}\,,$$

während A die Eigenwerte

$$\lambda_1 = 2 + \sqrt{2}\,, \quad \lambda_2 = 2\,, \quad \lambda_3 = 2 - \sqrt{2}$$

hat.

4) Nach Satz 4.5 erhält man den betragsgrößten Eigenwert und den dazugehörenden Eigenvektor einer diagonalisierbaren Matrix (eine positiv definite Matrix ist bekanntlich diagonalisierbar) mit der von-Mises-Iteration mit dem Programm

```
# Programm aufg44 von-Mises-Vektoriteration
# input:  pos. definite symmertische Matrix a, Toleranz epsilon
# output: groesster Eigenwert und der dazugehoerende Eigenvektor
# aufruf: [lambda,v,iter] = aufg44(a,epsilon);
function [lambda,v,iter] = aufg44(a,epsilon);
n = length(a(:,1));
z = ones(n,1); z(1) = z(1)+0.5;
v = ones(n,1); v = v/norm(v); v(1)=v(1)+0.5; va=v; zeta=1;
lambda = -1; lambdaa = -2;
iter = 0;
while (abs(lambda-lambdaa) > epsilon && iter < 50)
  v = a*va;
  if (iter > 0)
    lambdaa = lambda;
  endif
  if (va'*z == 0)
    z(1) = z(1) + 0.5;
  endif
  lambda = (v'*z)/(va'*z);
  va = v;
  zeta = 0;
```

```
  for i=1:n
    if (v(i) ~= 0 && zeta == 0)
      if (v(i) < 0)
        zeta = -1;
      endif
      if (v(i) > 0)
        zeta = 1;
      endif
    endif
  endfor
  v = zeta/(norm(v))*v;
  iter = iter+1;
endwhile
endfunction
```

Mit der Deflationstechnik (Satz 4.7) kann man einen Eigenwert λ_1 der Matrix A mit dem Eigenvektor $\vec{x}_1$ auf den Eigenwert 0 der Matrix

$$\tilde{A} = (E - \frac{\vec{x}_1 \vec{x}_1'}{\langle \vec{x}_1, \vec{x}_1 \rangle})A$$

abbilden. Nun kann man im Fall einer positiv definiten symmetrischen Matrix (alle Eigenwerte sind positiv) mit dem Programm **aufg44.m** den zweitgrößten Eigenwert λ_2 und den dazugehörenden Eigenvektor $\vec{x}_2$ bestimmen. Die nochmalige Deflation

$$\tilde{\tilde{A}} = (E - \frac{\vec{x}_2 \vec{x}_2'}{\langle \vec{x}_2, \vec{x}_2 \rangle})\tilde{A}$$

führt mit der Matrix $\tilde{\tilde{A}}$ auf eine Matrix, die den doppelten Eigenwert 0 besitzt. Jetzt kann man den drittgrößten Eigenwert bestimmen usw.
Mit dem folgenden Programm bestimmt man alle Eigenwerte und Eigenvektoren durch die sukkzessive Deflation und die mehrfache Anwendung der Von-Mises-Vektoriteration.

```
# Programm aufg44a von-Mises-Vektoriteration
# input:  pos. definite symmertische Matrix a, Toleranz epsilon
# output: Eigenwerte und dazugehoerende Eigenvektoren
# benoetigtes Programm: aufg44.m
# aufruf: [lambda,v,iter] = aufg44a(a,epsilon);
function [lambda,v,iter] = aufg44a(a,epsilon);
n = length(a(:,1));
a1 = a;
for i=1:n
  [lambda(i),vv,iter(i)] = aufg44(a1,epsilon);
  a1 = a1 - (vv*vv')/(vv'*vv)*a1;
  v(:,i) = vv;
endfor
endfunction
```

Testen Sie das Programm **aufg44a.m** für die Matrizen

$$A_1 = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix} \quad \text{und} \quad A_2 = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix} .$$

Alle Eigenwerte und Eigenvektoren einer positiv definiten symmetrischen Matrix kann man natürlich auch mit Hilfe der inversen Von-Mises-Vektoriteration bzw. der QR-Iteration bestimmen (s. dazu nachfolgendes Programm zur Lösung der Aufgabe 5).

5) Mit den Programmen zur Erzeugung der Hessenbergform einer Matrix und der QR-Zerlegung kann man die QR-Iteration zur Berechnung der Eigenwerte und Eigenvektoren mit dem nachfolgenden Programm realisieren.

```
# Programm aufg45 zur Berechnung der Eigenwerte und Eigenvektoren
# einer reellen symmetrischen Matrix a
# input:  Matrix a vom Typ (n x n), toleranz epsilon, kappa Wilkinson-shift
# output: Matrix lambda mit den Eigenwerten auf der Hauptdiagonalen
#         Matrix qh mit den Eigenvektoren als Spalten
#         Residuum resi, Iterationszahl iter
# aufruf: [lambda,qh,resi,iter] = aufg45(a,kappa,epsilon);
# benoetigte Programme: hessenberg_gb.m, qriteration_gb.m
function [lambda,qh,resi,iter] = aufg45(a,kappa,epsilon);
n = length(a(1,:));
# Transformation von a auf Hessenbergform
[h,qt] = hessenberg_gb(a);
p = eye(n);
resi = epsilon + 1;
iter = 1;
# Ermittlung der Maschinengenauigkeit
tau = 1;
while ((tau+1)>1)
  tau = tau/2;
endwhile
# qr-Iteration
while (resi > epsilon && iter < 200)
  [h,p] = qriteration_gb(h,p,kappa,tau);
  resi = 0;
  for i=1:n
    for j=1:n
      if (i > j)
        resi = resi + abs(h(i,j));
      endif
    endfor
  endfor
  iter = iter + 1;
endwhile
if (resi > epsilon)
  printf('keine Konvergenz');
  return
endif
printf('Konvergenz, Residuum: ');
resi
printf('Iterationszahl: ');
iter
lambda = h;
qh = qt'*p;
endfunction
```

Bei der Anwendung des Programm kann man den beträchtlichen Einfluss des Shift-Parameters κ feststellen. Sucht man z.B. die Eigenwerte und Eigenvektoren der Matrix

$$A = \begin{pmatrix} 1 & -2 & 0 \\ -2 & 1 & -2 \\ 0 & -2 & 1 \end{pmatrix},$$

dann benötigt man mit $\kappa = 1{,}2$ insgesamt 95 Iterationen, um die Summe der Absolutbeträge des unteren Dreiecks der Matrix Λ (in der sich die Eigenwerte auf der Hauptdiagonalen ergeben) kleiner als 10^{-5} zu machen, während man mit $\kappa = 1{,}95$ nur 21 Iterationen braucht.

Kapitel 5

1) Zur Berechnung des Polynomwertes mit der Formel (5.3) benötigt man zur Berechnung eines Basispolynoms $2(n-1)+1$ Multiplikationen, also für die $n+1$ Basispolynome insgesamt $(2n-1)(n+1)$ Multiplikationen. Hinzu kommen noch $n+1$ Multiplikationen gemäß Formel (5.3), also insgesamt $2n(n+1)$ **Multiplikationen** und $2(n-1)+n$ Additionen.
Bei der baryzentrischen Formel (5.9) benötigt man die Koeffizienten μ_i (und λ_i). Zur Berechnung der λ_i, $i=0,\ldots,n$ nach Formel (5.5) benötigt man $(n+1)(n+1)$ Multiplikationen. Zur Berechnung de μ_i, $i=0,\ldots,n$ nach Formel (5.6) benötigt man $n+1$ Multiplikationen. In der baryzentrischen Formel (5.9) sind $n+2$ Multiplikationen durchzuführen. Insgesamt sind also $(n+1)^2+2n+3$ **Multiplikationen** und $(n+1)(n+2)$ Additionen bei der Benutzung der baryzentrischen Formel (5.9) erforderlich.

2) Für die Wertetabelle

x_i	0	1	2	3
y_i	0	3	2	1

,

also die Ausgangsposition für ein Polynom 2. Grades. Dafür kann man das Schema

$$
\begin{array}{lllllllll}
[x_0]=[0]=0 & & & & & & & & \\
 & \searrow & & & & & & & \\
[x_1]=[1]=3 & \to & [x_0x_1]=[01]=3 & & & & & & \\
 & \searrow & & & \searrow & & & & \\
[x_2]=[2]=2 & \to & [x_1x_2]=[12]=-1 & \to & [x_0x_1x_2]=[012]=-2 & & & & \\
 & & & & & \searrow & & & \\
[x_3]=[3]=1 & \to & [x_2x_3]=[23]=-1 & \to & [x_1x_2x_3]=[123]=0 & \to & [x_0x_1x_2x_3]=[0123]=\frac{3}{3} & &
\end{array}
$$

Als Newton'sches Interpolationspolynom ergibt sich

$$p_3(x)=0+3(x-0)-2(x-0)(x-1)+\frac{2}{3}(x-0)(x-1)(x-2)=3x-2x(x-1)+\frac{2}{3}x(x-1)(x-2)\ .$$

3) Das Steigungsschema lautet

$$
\begin{array}{lllllllll}
[1]=2 & & & & & & & & \\
 & \searrow & & & & & & & \\
[1]=2 & \to & [11]=f'(1)=1 & & & & & & \\
 & \searrow & & \searrow & & & & & \\
[4]=3 & \to & [14]=\frac{2-3}{1-4}=\frac{1}{3} & \to & [114]=\frac{1-1/3}{1-4}=-\frac{2}{9} & & & & \\
 & \searrow & & \searrow & & \searrow & & & \\
[4]=3 & \to & [44]=f'(4)=2 & \to & [144]=\frac{1/3-2}{1-4}=\frac{5}{9} & \to & [1144]=\frac{-2/9-5/9}{1-4}=\frac{7}{27} & & \\
 & \searrow & & \searrow & & \searrow & & & \\
[4]=3 & \to & [44]=f'(4)=2 & \to & [444]=\frac{f''(4)}{2}=\frac{1}{2} & \to & [1444]=\frac{5/9-1/2}{1-4}=-\frac{1}{54} & &
\end{array}
$$

und schließlich

$$
\begin{array}{lllll}
[1144]=\frac{7}{27} & & & & \\
 & \searrow & & & \\
[1444]=-\frac{1}{54} & \to & [11444]=\frac{7/27+1/54}{1-4}=-\frac{5}{54} & &
\end{array}
\ .
$$

Aus dem Steigungsschema liest man mit

$$\begin{aligned} h_4(x) &= [1] + [11](x-1) + [114](x-1)^2 + [1144](x-1)^2(x-4) + [11444](x-1)^2(x-4)^2 \\ &= 2 + 1(x-1) - \frac{2}{9}(x-1)^2 + \frac{7}{27}(x-1)^2(x-4) - \frac{5}{54}(x-1)^2(x-4)^2 \end{aligned}$$

das interpolierende Hermite'sche Polynom ab.

4) Für eine dreimal stetig differenzierbare Funktion gilt mit $x_1 = x_0 + h,\ x_2 = x_0 + 2h$ nach dem Satz von Taylor

$$\begin{aligned} f(x_1) &= f(x_0) + hf'(x_0) + \frac{h^2}{2}f''(x_0) + O(h^3) \\ f(x_2) &= f(x_0) + 2hf'(x_0) + \frac{4h^2}{2}f''(x_0) + O(h^3)\ . \end{aligned}$$

Subtrahiert man die zweite Gleichung vom vierfachen der ersten Gleichung, erhält man

$$4f(x_1) - f(x_2) = 3f(x_0) + 2hf'(x_0) + O(h^2)\ .$$

Die Auflösung nach $f'(x_0)$ ergibt

$$f'(x_0) = \frac{-f(x_2) + 4f(x_1) - 3f(x_0)}{2h} + O(h^2)\ .$$

Völlig analog zeigt man

$$f'(x_2) = \frac{3f(x_2) - 4f(x_1) + f(x_0)}{2h} + O(h^2)\ .$$

Weiter gilt nach dem Satz von Taylor

$$\begin{aligned} f(x_2) &= f(x_1) + hf'(x_1) + \frac{h^2}{2}f''(x_1) + O(h^3) \\ f(x_0) &= f(x_1) - hf'(x_0) + \frac{h^2}{2}f''(x_1) + O(h^3)\ . \end{aligned}$$

Die Subtraktion der Gleichungen ergibt

$$f(x_2) - f(x_0) = 2hf'(x_1) + O(h^3) \implies f'(x_1) = \frac{f(x_2) - f(x_0)}{2h} + O(h^2)\ .$$

5) Für 2 nichtnegative Zahlen a, b gilt

$$|a - b| \le |a + b| = a + b\ .$$

Diese Ungleichung wird weiter unten benutzt und sollte zur Übung nachgewiesen werden.
Zum Nachweis der Lipschitz-Stetigkeit unterscheiden wir die Fälle

a) $x, y > 0$ bzw. $x, y < 0$,

b) $x > 0,\ y < 0$ bzw. $x < 0,\ y > 0$,

c) $x \neq 0,\ y = 0$ bzw. $x = 0,\ y \neq 0$.

Im Fall a) gilt gilt aufgrund des Mittelwertsatzes

$$|f(x) - f(y)| = |\arctan x - \arctan y| = |\frac{1}{1+\xi^2}|\,|x - y| \iff |f(x) - f(y)| \le |x - y|$$

bzw.

$$\begin{aligned} |f(x)-f(y)| &= |\arctan(-x)-\arctan(-y)| = |-\arctan x+\arctan y| \\ &= |\frac{1}{1+\xi^2}|\,|x-y| \iff |f(x)-f(y)| \le |x-y| \,. \end{aligned}$$

Im Fall b) sei $x>0,\ y<0$. Dann ist wegen der oben angegebenen Ungleichung

$$|f(x)-f(y)| = |\arctan x-\arctan(-y)| := |a-b| \le |a+b| = |\arctan x+\arctan(-y)|$$

bzw. weil $\arctan x$ eine ungerade Funktion ist

$$|f(x)-f(y)| \le |\arctan x-\arctan y| \le |\frac{1}{1+\xi^2}||x-y| \,.$$

Für $x<0,\ y>0$ erhält man das analoge Resultat.
Im Fall c) ist für $x\neq 0$ und $y=0$

$$|f(x)-f(y)| = |f(x)| = |\arctan x| = |\arctan x-\arctan 0| = |\frac{1}{1+\xi^2}|\,|x-0| \le |x-y| \,.$$

Für $x=0$ und $y\neq 0$ erhält man auf die gleiche Weise

$$|f(x)-f(y)| \le |x-y| \,.$$

Insgesamt ist damit die Funktion Lipschitz-stetig mit der Lipschitz-Konstanten $K=1$.
Um die Funktion im Intervall $[-1,1]$ mit der Bernstein-Formel mit einer Genauigkeit von $\epsilon = 10^{-3}$ zu approximieren, muss der Grad n der Ungleichung

$$n > \frac{K^2}{\epsilon^2} = \frac{1}{10^{-6}} = 10^6$$

genügen. Es müssen also Bernstein-Polynome bis zum Grad $n=10^6$ verwendet werden.

6) Wie Abschnitt 5.5 beschrieben, ist das Gleichungssystem (5.42) für $i=0,1,\ldots,n-1$ aufzustellen. Mit den periodischen Bedingungen

$$y_0=y_n,\ y_{-1}=y_{n-1},\ y_0''=y_n'',\ y_{-1}''=y_{n-1}'' \Longleftrightarrow m_0=m_n,\ m_{-1}=m_{n-1},\ \ h_{-1}=h_{n-1}$$

kann eliminiert man $y_{-1}, m_{-1}, m_n, h_{-1}$ und erhält dann ein Gleichungssystem zur Berechnung von $m_0,\ldots,m_{n-1}$ und kann damit den Spline berechnen. Die entstehende Koeffizientenmatrix ist eine zyklische Matrix, d.h. sie enhält neben der Hauptdiagonale, der darunter liegenden Diagonalen und der darüberliegenden Diagonalen mit von Null verschiedenen Elementen noch die von Null verschiedenen Elemente $a(1,n)$ und $a(n,1)$. Mit den folgenden Programmen können einzelne Spline-Werte berechnet werden bzw. der Spline geplottet werden.

```
# Programm aufg56_wert zur Berechnung eines kubischen Splines (periodisch)
# Berechnung des Wertes s(xx) eines kubischen Splines
# fuer die Stuetzpunkte (x_0,y_0),...,(x_n,y_n)
# input:  Vektoren x(1:n+1), y(1:n+1), Stelle xx
# output: Wert des Splines s(xx)
# berechnet wird ein interpolierender Spline
# benutzte Funktionen: splabl_per, splber_per
# aufruf: fwert = aufg56_wert(xx,x,y);
function [fwert] = aufg56_wert(xx,x,y);
n=length(x)-1;
ys = splabl_per(x,y,n);
fwert = splber_per(xx,x,y,ys,n);
endfunction
```

```
# Programm aufg56_plot Plot des Graphs eines kubischen Splines (periodisch)
# fuer die Stuetzpunkte (x_0,y_0),...,(x_n,y_n)
# input:  Vektoren x(1:n+1), y(1:n+1)
# benutzte Funktionen: splabl_per, splber_per
# aufruf:  aufg56_plot(x,y);
function aufg56_plot(x,y);
n=length(x)-1;
xmin=min(x);
xmax=max(x);
ys = splabl_per(x,y,n);
for i=1:300
   xx(i)=xmin+(i-1)*(xmax-xmin)/299;
   yy(i) = splber_per(xx(i),x,y,ys,n);
endfor
xxyy=[xx yy];
# Sicherung der berechneten Spline-Werte unter dem Namen "xxyy"
save 'xxyy' xxyy;
plot(x,y,'+',xx,yy)
endfunction
```

```
# Programm zur Bestimmung der 2. Ableitungen (periodisch)
function [xs] = splabl_per(x,y,n);
# Interpolation mit echten Splines/Bestimmung der Ableitungen ***
for i=0:n-1
   ys(1+i)=x(2+i)-x(1+i);
endfor
xs = zeros(n,1);
b = zeros(n,1);
# Aufbau der Koeffizientenmatrix aa(n,n)
aa = zeros(n,n);
aa(1,1) = 2*(ys(n)+ys(1)); aa(1,2) = ys(1); aa(1,n) = ys(n);
aa(n,1) = ys(n); aa(n,n-1) = ys(n-1); aa(n,n) = 2*(ys(n-1)+ys(n));
for i=2:n-1
   aa(i,i-1) = ys(i);
   aa(i,i+1) = ys(i+1);
   aa(i,i) = 2*(ys(i+1)+ys(i));
endfor
b(1) = 6*((y(2)-y(1))/ys(1)-(y(1)-y(n))/ys(n));
b(n) = 6*((y(1)-y(n))/ys(n)-(y(n)-y(n-1))/ys(n-1));
for i=2:n-1
   b(i)=6*((y(i+1)-y(i))/ys(i)-(y(i)-y(i-1))/ys(i-1));
endfor
# Bestimmung der Ableitungen xs(1)...xs(n+1)
xs = aa\b;
xs(n+1)=x(1);
endfunction
```

```
# Programm zur Berechnung
# eines Funktionswertes eines kubischen Splines (periodisch)
function [splinew] = splber_per(z,x,y,ys,n)
# wobei x(1) <= z <= x(n+1) vorausgesetzt wird
i=1;
while (x(i) > z | z > x(i+1))
   i=i+1;
endwhile
iz=i;
h=x(iz+1)-x(iz);
z1=z-x(iz);
a0=y(iz);
a1=(y(iz+1)-y(iz))/h - h*(2*ys(iz)+ys(iz+1))/6;
a2=ys(iz)/2;
a3=(ys(iz+1)-ys(iz))/(6*h);
splinew = a0+(a1+(a2+a3*z1)*z1)*z1;
endfunction
```

7) Für die Aufgabe werden für einen Test die Werte

$$u(0) = 0,\ u(k) = k^2,\ u(2k) = (2k)^2,\ u(3k) = (3k)^2,\ \ldots, u(nk) = (nk)^2, \ldots$$

vorgegeben, also das Profil $u(y) = y^2$. Die Aufgabe löst man mit dem rechtsseitigen Differenzenquotienten 2. Ordnung

$$\frac{-f(x_2) + 4f(x_1) - 3f(x_0)}{2h} \approx f'(x_0)\ .$$

Statt dem Programm 5.4a arbeitet man mit dem Programm

```
# Programm Aufgabe 5.7a
function [y1] = f1(a,h);
  urr=(a+2*h)^2;ur=(a+h)^2;u=a^2;
  y1 = (-urr + 4*ur - 3*u)/(2*h);
endfunction
```

In der Praxis muss man natürlich mit numerisch berechneten oder experimentell bestimmten Werten von u an den Stellen $y = a + 2h,\ a + h,\ a$ statt mit dem Test-Profil $u(y) = y^2$ arbeiten. Mit dem Programm

```
# Programm Aufgabe 5.7
function [fwert] = extrap_f1_gb(a,h);
n = length(h)-1;
for k=1:n
  y(k+1) = f1(a,h(k+1));
endfor
fwert = lagrangebaryip_gb(0,h,y);
endfunction
```

erhält man das Resultat $\tau_w = 0$, also den exakten Wert von $u'(y) = 2y$ an der Stelle $y = 0$.

8) Mit der Substitution $t = x\pi$ erhält man mit

$$g(t) := f(\frac{t}{\pi}) = |\frac{t}{\pi} - 2k|,\ t \in [(2k-1)\pi, (2k+1)\pi],\ k \in \mathbb{Z},$$

eine 2π-periodische Funktion.

Kapitel 6

1) $E_2[x^3] = 0$ wurde bereits gezeigt. Für $E_2[x^4]$ erhält man

$$\begin{aligned} E_2[x^4] &= \int_a^b x^4\,dx - \frac{b-a}{6}[a^4 + 4(\frac{a+b}{2})^4 + b^4] \\ &= \frac{1}{5}(b^5 - a^5) - \frac{b-a}{6}[a^4 + 4(\frac{a+b}{2})^4 + b^4]\ . \end{aligned}$$

Wählt man speziell $a = 0 < b$, dann ergibt sich

$$E_2[x^4] = \frac{1}{5}b^5 - \frac{b}{6}[\frac{b^4}{4} + b^4] = (\frac{1}{5} - \frac{5}{24})b^5 \neq 0\ .$$

Damit ist gezeigt, dass der Genauigkeitsgrad der Kepler'schen Fassregel gleich 3 ist.

2) Für $E_{S_1}[f]$ erhält analog zur Berechnung von $E_{S_2}[f]$

$$E_{S_1}[f] = -\frac{1}{12}(\beta - \alpha)h^2\ .$$

Für die zweite und vierte Ableitung der $\arctan$-Funktion ergibt sich

$$\arctan''(x) = -\frac{2x}{(1+x^2)^2}\ , \quad \arctan^{(4)}(x) = -\frac{48x^3}{(1+x^2)^4} + \frac{24x}{(1+x^2)^3}\ .$$

Mit $\alpha = 0$ und $\beta = 10$ ergeben sich die Abschätzungen

$$|E_{S_1}[f]| \leq \frac{10}{12}h^2|\arctan''(\eta)| \leq \frac{10}{12}h^2 20 = \frac{50}{3}h^2$$

bzw.

$$|E_{S_2}[f]| \leq \frac{1}{18}h^4|\arctan^{(4)}(\eta)| \leq \frac{1}{18}h^4(48 \cdot 1000 + 240] = \frac{24120}{9}h^4$$

für die Fehler der summierten Trapezregel bzw. der Kepler'schen Fassregel.

3) Mit der Transformation $t = -2 + x$ bzw. $x = t + 2$ ist

$$\int_1^3 \frac{\sin x}{x}\,dx = \int_{-1}^1 \frac{\sin(t+2)}{t+2}\,dt\ .$$

Weiter ist

$$\int_{-1}^1 \frac{\sin(t+2)}{t+2}\,dt = \int_{-1}^1 \sqrt{1-t^2}\frac{\sin(t+2)}{t+2}\,\frac{1}{\sqrt{1-t^2}}\,dt\ ,$$

so dass sich die Gauß-Tschebyscheff-Quadratur zu

$$\int_1^3 \frac{\sin x}{x}\,dx \approx \frac{\pi}{n}\sum_{j=1}^n \sqrt{1-t_j^2}\,\frac{\sin(t_j+2)}{t_j+2} \quad (t_j = \cos(\frac{(2j-1)\pi}{2n}),\ j = n,\ldots,1)$$

ergibt.

4) Startet man mit $p_0(x) = 1$ dann erhält man mit $\langle f, g\rangle_\rho = \int_{-1}^1 fg\,dx$

$$\beta_0 = \frac{\langle x,1\rangle_\rho}{\langle 1,1\rangle_\rho} = 0\ , p_1(x) = x - \beta_0 = x\ , \gamma_1^2 = \frac{\langle x,x\rangle_\rho}{\langle 1,1\rangle_\rho} = \frac{1}{3}\ .$$

Weiter ergibt sich

$$\beta_1 = \frac{\langle x^2, x\rangle_\rho}{\langle x, x\rangle_\rho} = 0\ , p_2(x) = (x-\beta_1)p_1(x) - \gamma_1 p_0(x) = x^2 - \frac{1}{3}\ , \gamma_2^2 = \frac{\langle x^2-\frac{1}{3}, x^3-\frac{1}{3}\rangle_\rho}{\langle x, x\rangle_\rho} = \frac{4}{15}$$
$$\beta_2 = \frac{\langle x(x^2-\frac{1}{3}), x(x^2-\frac{1}{3})\rangle_\rho}{\langle x^2-\frac{1}{3}, x^2-\frac{1}{3}\rangle_\rho} = 0\ , p_3(x) = (x-\beta_2)p_2(x) - \gamma_2^2 p_1(x) = x(x^2-\frac{1}{3}) - \frac{4}{15}x = x^3 - \frac{3}{5}x\ .$$

Als Nullstellen von $p_3(x)$ findet man $x_1 = -\sqrt{\frac{3}{5}}$, $x_2 = 0$, $x_3 = \sqrt{\frac{3}{5}}$. Mit der Substitution $t = -3 + x$, $dt = dx$ gilt

$$\int_2^4 \sin(x^2)\, dx = \int_{-1}^1 \sin((t+3)^2)\, dt\ .$$

Für die Gewichte der Gauß-Legendre-Quadratur folgt mit den Nullstellen von $p_3(x)$

$$\begin{aligned}
\sigma_1 &= \langle L_1,1\rangle_\rho = \int_{-1}^1 \frac{5}{6}(x^2 - x\sqrt{\frac{3}{5}})\, dx = \frac{5}{9}\ ,\\
\sigma_2 &= \langle L_2,1\rangle_\rho = \int_{-1}^1 (1 - x^2\frac{5}{3})\, dx = \frac{8}{9}\ ,\\
\sigma_3 &= \langle L_3,1\rangle_\rho = \int_{-1}^1 \frac{5}{6}(x^2 + x\sqrt{\frac{3}{5}})\, dx = \frac{5}{9}\ ,
\end{aligned}$$

so dass sich die Gauß-Legendre-Quadratur

$$\int_2^4 \sin(x^2)\, dx \approx \frac{5}{9}\sin((-\sqrt{\frac{3}{5}}+3)^2) + \frac{8}{9}\sin 9 + \frac{5}{9}\sin((\sqrt{\frac{3}{5}}+3)^2)$$

ergibt.

5) Mit der Substitution $\xi = -x + 2$, $d\xi = -dx$ erhält man

$$\begin{aligned}
\int_{-\infty}^2 e^{-x^2}\, dx &= -\int_\infty^0 e^{-(2-\xi)^2}\, d\xi = \int_0^\infty e^{-(2-\xi)^2}\, d\xi\\
&= \int_0^\infty e^{-(2-\xi)^2} e^{\xi} e^{-\xi}\, d\xi = \int_0^\infty e^{\xi-(2-\xi)^2} e^{-\xi}\, d\xi\ ,
\end{aligned}$$

also $g(\xi) = e^{\xi-(2-\xi)^2}$.

6) Für den Fall $n = 2$ erhält man für die Gewichte

$$\begin{aligned}
\sigma_0 &= \int_0^1 q_0(x)\, dx = \int_0^1 \binom{2}{0} x^0 (1-x)^2\, dx = [-\frac{(1-x)^3}{3}]_0^1 = \frac{1}{3}\ ,\\
\sigma_1 &= \int_0^1 q_1(x)\, dx = \int_0^1 \binom{2}{1} x^1 (1-x)\, dx = 2[\frac{x^2}{2} - \frac{x^3}{3}]_0^1 = \frac{1}{3}\ ,\\
\sigma_2 &= \int_0^1 q_2(x)\, dx = \int_0^1 \binom{2}{2} x^2\, dx = \frac{1}{3}\ .
\end{aligned}$$

Für $n = 3$ erhält man durch analoge Rechnungen die Gewichte $\sigma_j = \frac{1}{4}$ und für $n = 4$ erhält man die konstanten Gewichte $\sigma_j = \frac{1}{5}$.
Es ergibt sich also für $n = 2,3,4$ die Quadraturformel

$$A_n[f] = \sum_{k=1}^n \frac{1}{n+1} f(\frac{k}{n})\ .$$

Der Genauigeitsgrad ist 1, denn für $f(x) = x^0 = 1$ und $f(x) = x^1$ ist die Quadraturformel exakt. Für $f(x) = x^2$ ergibt sich

$$\int_0^1 x^2\,dx = \frac{1}{3}\ ,$$

aber

$$A_n[x^2] = \sum_{k=0}^{n} \frac{1}{n+1}(\frac{k}{n})^2 = \frac{1}{(n+1)n^2}\sum_{k=0} k^2 = \frac{1}{(n+1)n^2}\frac{n(n+1)(2n+1)}{6} = \frac{2n+1}{6n} = \frac{1}{3}+\frac{1}{6n}\ ,$$

also ist $A_n[x^2] \neq \int_0^1 x^2\,dx$ so dass der Genauigkeitsgrad 1 ist.
Wenn man $A_n[f]$ in der Form

$$A_n[f] = \frac{n}{n+1}[\frac{f(0)}{n} + \sum_{j=1}^{n}\frac{1}{n}f(\frac{j}{n})] =: \frac{n}{n+1}[\frac{f(0)}{n} + R_n(f)]$$

aufschreibt, erkennt man mit $R_n(f) = \sum_{j=1}^{n}\frac{1}{n}f(\frac{j}{n})$ eine Riemann'sche Summe. Es gilt offensichtlich

$$\lim_{n\to\infty} A_n[f] = \lim_{n\to\infty}\frac{n}{n+1}[\frac{f(0)}{n} + R_n(f)] = \lim_{n\to\infty} R_n(f) = \int_0^1 f(x)\,dx\ .$$

Kapitel 7

1) Mit einem Plot der Funktionen $\Psi_3(x)$ und $y = x$ für $x < 0$ sowie der Ableitung $\Psi_3'(x) = \frac{e^x}{\sqrt{1-e^{2x}}}$ erkennt man, dass in der Nähe sämtlicher Fixpunkte die Werte von $\Psi_3'(x)$ außerhalb des Intervall $[-1{,}1]$ liegen, so dass die Fixpunkte abstoßend sind und durch eine Fixpunktiteration nicht erreicht werden.
Das Gleiche trifft auf die Fixpunktabbildung $\Psi_4(x)$ zu.

2) Mit etwas Geschick findet man heraus, dass

$$\cos([0{,}7\,,\,0{,}77]) \subset [0{,}7\,,\,0{,}77]$$

gilt. Damit hat man mit K

$$1 > K = 0{,}7 \geq \max_{x\in[0{,}7\,,\,0{,}77]} |\cos'(x)| = 0{,}69614\ldots$$

eine Kontraktionskonstante. Mit $x_0 = 0{,}7$ folgt $x_1 = 0{,}76484\ldots$, also $|x_1 - x_0| = 0{,}06484\cdots \leq 0{,}0649$. Genügt n der Ungleichung

$$\frac{0{,}7^n}{0\;3}|x_1-x_0| < \frac{0{,}7^n}{0{,}3}0{,}0649 < 10^{-4} \iff n\ln 0{,}7 \leq \ln\frac{0{,}3\cdot 10^{-4}}{0{,}0649} \iff n \geq \frac{\ln 0{,}00046225}{\ln 0{,}7} = 21{,}531\,,$$

dann wird aufgrund der A-priori-Abschätzung (7.6) die geforderte Genauigkeit erreicht.

3) Aus $K^n = e^{-\gamma n}$ folgt

$$\ln K^n = -\gamma n \iff -\ln K = \gamma$$

und damit aus der A-priori-Abschätzung (7.6) die behauptete Ungleichung mit dem Konvergenzexponenten $\gamma = -\ln K > 0$.

4) Die Auflösung der Gleichung nach x ergibt

$$x = \frac{3}{4\arcsin\frac{1}{2x}}\,.$$

Ein Plot der Graphen der Funktionen $y = x$ und $\Psi_1(x) = \frac{3}{4\arcsin\frac{1}{2x}}$ deutet auf Lösungen $x \approx \pm 0{,}5$ hin. Der Versuch einer Fixpunktiteration

$$x_{k+1} = \Psi_1(x_k)\,, \quad k = 0{,}1{,}2{,}\ldots$$

mit dem Startwert $x_0 = 0{,}51$ erweist sich als nicht erfolgreich (x_0 muss größer als $0{,}5$ sein, da der $\arcsin\frac{1}{2x}$ nur für $|x| > 0{,}5$ definiert ist). Wendet man auf $\arcsin\frac{1}{2x} = \frac{3}{4x}$ die Sinus-Funktion an, so erhält man die Fixpunktgleichung

$$x = \frac{1}{2\sin\frac{3}{4x}} =: \Psi_2(x) \quad (x \neq k\pi,\, k \in \mathbb{Z})\,.$$

Die Fixpunktiteration

$$x_{k+1} = \Psi_2(x_k)\,, \quad k = 0{,}1{,}2{,}\ldots$$

mit dem Startwert $x_0 = 0{,}5$ ist erfolgreich und man erhält mit

$$\bar{x} \approx 0{,}5014101095599521$$

eine auf 15 Stellen genaue Näherung des Fixpunktes nach 20 Iterationen. Als 2. Fixpunkt erhält man $\bar{x} \approx -0{,}5014101095599521$ ausgehend von $x_0 = -0{,}5$.
Man stellt auch fest, dass die Ableitung von $\Psi_2(x)$

$$\Psi_2'(x) = \frac{3\cos(\frac{3}{4x})}{8x^2\sin^2(\frac{3}{4x})}$$

für $x = 0{,}5$ den Wert $0{,}10664\ldots$ hat, also der Fixpunkt von Ψ_2 nahe $0{,}5$ ein anziehender Fixpunkt zu sein scheint. Die Ableitung von $\Psi_1(x)$

$$\Psi_1'(x) = \frac{3}{8x\arcsin(\frac{1}{2x})\sqrt{1-\frac{1}{4x^2}}}$$

ist für $x = 0{,}51$ mit $2{,}7732\ldots$ größer als 1, so dass der Fixpunkt der Abbildung Ψ_1 nahe $0{,}51$ offensichtlich ein abstoßender Fixpunkt ist.

5) Die Lösung des Gleichungssystems ist gleichbedeutend mit der Bestimmung einer Nullstelle der Abbildung

$$\vec{f}: \mathbb{R}^4 \to \mathbb{R}^4\,, \quad \vec{f}(\vec{x}) = \vec{f}(x,y,z,\lambda) = \begin{pmatrix} \sin(zy) + 2x\lambda \\ xz\cos(zy) + 2y\lambda \\ xy\cos(zy) + 2z\lambda \\ x^2 + y^2 + z^2 - 4 \end{pmatrix}\,.$$

Mit der Ableitungsmatrix

$$\vec{f}\,'(x,y,z,\lambda) = \begin{pmatrix} 2\lambda & z\cos(zy) & y\cos(zy) & 2x \\ z\cos(zy) & -xz^2\sin(zy) + 2\lambda & x\cos(zy) - xyz\sin(zy) & 2y \\ y\cos(zy) & x\cos(zy) - xyz\sin((zy) & -xy^2\sin(zy) + 2\lambda & 2z \\ 2x & 2y & 2z & 0 \end{pmatrix}$$

erhält man das Newton-Verfahren

$$\vec{x}^{(k+1)} = \vec{x}^{(k)} - [\vec{f}\,'(\vec{x}^{(k)})]^{-1}\vec{f}(\vec{x}^{(k)})\,, \quad k = 0{,}1{,}2{,}\ldots$$

bzw.

$$\vec{f}\,'(\vec{x}^{(k)})\,\vec{z}^{(k+1)} = -\vec{f}(\vec{x}^{(k)})\,, \quad \vec{x}^{(k+1)} = \vec{z}^{(k+1)} + \vec{x}^{(k)}\,, \quad k = 0{,}1{,}2{,}\ldots\,.$$

Mit der ziemlich willkürlich gewählten Startiteration $\vec{x}^{(0)} = (1{,}1{,}1,\ 0{,}5)^T$ (hier muss man nur sichern, dass $\vec{f}\,'(\vec{x}^{(0)}$ nicht singulär ist) erhält man nach 8 Newton-Iterationen mit

$$x \approx 1{,}36002\,, \quad y \approx -1{,}03690\,, \quad z = -1{,}03690\,, \quad \lambda \approx -0\ 3234$$

eine Lösung mit $||\vec{f}(x,y,z,\lambda)|| < 10^{-13}$.
Mit den nachfolgenden Programmen werden der Wert der Abbildung und die Jacobi-Matrix als Grundbausteine für das Newton-Verfahren berechnet.

```
# Programm aufg7.5a
function [f] = funktion75(x,y,z,w);
f = zeros(4,1);
f(1) = sin(z*y) + 2*x*w;
f(2) = x*z*cos(z*y) + 2*y*w;
f(3) = x*y*cos(z*y) + 2*z*w;
f(4) = x^2 + y^2 + z^2 - 4;
endfunction
```

```
# Programm aufg7.5b
function [a] = jacobi75(x,y,z,w);
czy = cos(z*y); szy = sin(z*y);
a(1,1) = 2*w; a(1,2)=z*czy; a(1,3)=y*czy; a(1,4)=2*x;
a(2,1) = z*czy; a(2,2)=-x*z^2*szy+2*w; a(2,3)=x*czy-x*y*z*szy; a(2,4)=2*y;
a(3,1) = y*czy; a(3,2)= x*czy-x*y*z*szy; a(3,3)= -x*y^2*szy +2*w; a(3,4) = 2*z;
a(4,1) = 2*x; a(4,2) = 2*y; a(4,3)=2*z; a(4,4)=0;
endfunction
```

6) Das Gauß-Newton-Verfahren ist im folgenden Programm implementiert.

```
# Programm Gauss-Newton (Kap. 7)
# nichtlineares Ausgleichsproblem
# f(x) = b*sin(x - a)
#
# x_k   y_k
# -1,5  -0,4
# 0     0,1
# 1     0,45
# 2     0,35
# 3     0
# 4     -0,4
# 4,5   -0,5
#
# r(a,b,c) = ( b*sin(a*x1 + c) - y1;
#              b*sin(a*x2 + c) - y2;
#              ...
#              b*sin(a*xn + c) - yn )
# r'(a,b,c) = ...
#
# Aufruf: [a,b,c,it] = gaussnewton3(a0,b0,c0,alpha);
#
function [a,b,c,it] = gaussnewton3(a0,b0,c0,alpha);
xx = [ -1.5 0 1 2 3 4 4.5];
yy = [ -0.4 0.1 0.45 0.35 0 -0.4 -0.5];
a = a0; b = b0; c = c0;
M = Rs(a,b,c,xx,yy);
abc = [a; b; c];
ys = -R(a,b,c,xx,yy);
s = 10;
epsilon = 0.000001;
it = 0;
while (norm(s) > epsilon && it < 40)
   it = it + 1;
   s = M\ys;
   as = s + abc;
   abc = alpha*as + (1 - alpha)*abc;
   M = Rs(abc(1),abc(2),abc(3),xx,yy);
   ys = -R(abc(1),abc(2),abc(3),xx,yy);
   0.5*norm(R(abc(1),abc(2),abc(3),xx,yy))^2
endwhile
s
a = abc(1);
b = abc(2);
c = abc(3);
x = linspace(-2,5,20);
for j=1:20
   y(j) = b*sin(a*x(j) + c);
endfor
#
plot(x,y,'b-');
xlabel('x');
ylabel('y=f(x)');
hold on
plot(xx,yy,'*');
hold off
endfunction
#
function rabc = R(a,b,c,xx,yy);
r = zeros(7,1);
for j=1:7
   r(j) = b*sin(a*xx(j) + c) - yy(j);
endfor
rabc = r;
endfunction
```

```
#
function rsabc = Rs(a,b,c,xx,yy);
r = zeros(7,3);
for j=1:7
   r(j,1) = b*cos(a*xx(j)+c)*xx(j);
   r(j,2) = sin(a*xx(j)+c);
   r(j,3) = b*cos(a*xx(j) + c);
endfor
rsabc = r;
endfunction
```

7) Die Aufgabe kann mit dem folgenden Programm gelöst werden.

```
# Programm Uebungsaufgabe 7.6 zum SQP-Verfahren (sqp2_gb.m)
# Min F(x,y,z) , u.d.N. G(x,y,z) = 0
#
# F, G, gradF und gradG sind als Programme (Function)
# Spaltenvektoren, bereitzustellen
# input:  Startwert (x0,y0,z0,lambda0), Lagrange-Multiplikator
# output: Loesung (x*,y*,z*,lambda*), F(x*,y*,z*) und G(x*,y*,z*)
# Aufruf: [xs,ys,zs,lambda,f,g] = sqp2_gb(x0,y0,z0,lambda0)
#
function [xs,ys,zs,lambdas,f,g] = sqp2_gb(x0,y0,z0,lambda0)
M = zeros(4,4);
x = x0; y= y0; z=z0; lambda = lambda0;
B = [2+2*lambda , 0 , 0 ; 0 , 2 + 2*lambda , 0 ; 0 , 0 , 1+2*lambda];
M = [ B , gradG(x,y,z); gradG(x,y,z)', 0];
rhs = [-gradF(x,y,z)-lambda*gradG(x,y,z) ; -G(x,y,z)];
#
epsilon = 0.000001;
k = 0;
while abs(G(x,y,z)) > epsilon  &&  k < 30
   k = k + 1
   X = M\rhs;
   dx = X(1); dy = X(2); dz = X(3); dlambda = X(4);
   x = x + dx; y = y + dy; z = z + dz; lambda = lambda + dlambda;
   B = [2+2*lambda , 0 , 0 ; 0 , 2 + 2*lambda , 0 ; 0 , 0 , 1+2*lambda];
   M = [ B , gradG(x,y,z); gradG(x,y,z)' , 0];
   rhs = [-gradF(x,y,z)-lambda*gradG(x,y,z) ; -G(x,y,z)];
endwhile
xs = x;
ys = y;
zs = z;
lambdas = lambda;
f= F(x,y,z);
g = G(x,y,z);
endfunction
#
function grad = gradF(x,y,z);
grad = [-2*(3-x) ; -2*(4-y) ; -(5-z)];
endfunction
#
function grad = gradG(x,y,z);
grad = [2*(x-1) ; 2*(y-2) ; 2*z ];
endfunction
#
function f = F(x,y,z);
f = (3-x)^2 + (4-y)^2 + (5-z)^2 ;
endfunction
#
function g = G(x,y,z);
g = (x-1)^2- (y-2)^2 + z^2 -1;
endfunction
```

8) Hier bietet es sich an, die Nebenbedingung nach x aufzulösen und die Funktion $\hat{F}(y) = (2+y)^2 + 3y^2 + 2$ mit Schulwissen zu lösen. Im folgenden Programm ist ein SQP-Verfahren zur Lösung implementiert.

```
# Aufgabe 7.7 zum SQP-Verfahren (sqp1_gb.m)
# Min F(x,y) , u.d.N. G(x,y) = 0
#
# F, G, gradF und gradG sind als Programme (Function)
# Spaltenvektoren, bereitzustellen
# input:  Startwert (x0,y0,lambda0), Lagrange-Multiplikator
```

```
# output: Loesung (x*,y*,lambda*), F(x*,y*) und G(x*,y*)
# Aufruf: [xs,ys,lambda,f,g] = sqp1_gb(x0,y0,lambda0)
#
function [xs,ys,lambdas,f,g] = sqp1_gb(x0,y0,lambda0)
M = zeros(3,3);
x = x0; y= y0; lambda = lambda0;
B = [2 + 2*lambda , 0 ; 0 , 6];
M = [ B , gradG(x,y); gradG(x,y)', 0];
rhs = [-gradF(x,y)-lambda*gradG(x,y) ; -G(x,y)];
#
epsilon = 0.000001;
k = 0;
while abs(G(x,y)) > epsilon  &&  k < 20
   k = k + 1
   X = M\rhs;
   dx = X(1); dy = X(2); dlambda = X(3);
   x = x + dx; y = y + dy; lambda = lambda + dlambda;
   B = [2 + 2*lambda , 0 ; 0 , 6];
   M = [ B , gradG(x,y); gradG(x,y)' , 0];
   rhs = [-gradF(x,y)-lambda*gradG(x,y) ; -G(x,y)];
endwhile
xs = x;
ys = y;
lambdas = lambda;
f= F(x,y);
g = G(x,y);
endfunction
#
function grad = gradF(x,y);
grad = [2*x ; 6*y];
endfunction
#
function grad = gradG(x,y);
grad = [2*x ; -1];
endfunction
#
function f = F(x,y);
f = x^2 + 3*y^2 + 4;
endfunction
#
function g = G(x,y);
g = x^2- y -2;
endfunction
```

9) Die Aufgabe kann mit dem folgenden Programm gelöst werden.

```
# Programm Uebungsaufgabe 7.8 zum SQP-Verfahren (sqp3_gb.m)
# Min F(x,y,z) , u.d.N. G(x,y,z) = 0
#
# F, G, gradF und gradG sind als Programme (Function)
# Spaltenvektoren, bereitzustellen
# input:  Startwert (x0,y0,z0,lambda0), Lagrange-Multiplikator
# output: Loesung (x*,y*,z*,lambda*), F(x*,y*,z*) und G(x*,y*,z*)
# Aufruf: [xs,ys,zs,lambda,f,g] = sqp3_gb(x0,y0,z0,lambda0)
#
function [xs,ys,zs,lambdas,f,g] = sqp2_gb(x0,y0,z0,lambda0)
M = zeros(4,4);
x = x0; y= y0; z=z0; lambda = lambda0;
B = [2*lambda , z , y ; z , 2*lambda , x ; y , x , 2*lambda];
M = [ B , gradG(x,y,z); gradG(x,y,z)', 0];
rhs = [-gradF(x,y,z)-lambda*gradG(x,y,z) ; -G(x,y,z)];
#
epsilon = 0.000001;
k = 0;
while abs(G(x,y,z)) > epsilon  &&  k < 30
   k = k + 1
   X = M\rhs;
   dx = X(1); dy = X(2); dz = X(3); dlambda = X(4);
   x = x + dx; y = y + dy; z = z + dz; lambda = lambda + dlambda;
   B = [2*lambda , z , y ; z , 2*lambda , x ; y , x , 2*lambda];
   M = [ B , gradG(x,y,z); gradG(x,y,z)' , 0];
   rhs = [-gradF(x,y,z)-lambda*gradG(x,y,z) ; -G(x,y,z)];
endwhile
xs = x;
ys = y;
```

```
zs = z;
lambdas = lambda;
f= F(x,y,z);
g = G(x,y,z);
endfunction
#
function grad = gradF(x,y,z);
grad = [y*z ; x*z ; x*y];
endfunction
#
function grad = gradG(x,y,z);
grad = [2*x ; 2*y ; 2*z ];
endfunction
#
function f = F(x,y,z);
f = x*y*z ;
endfunction
#
function g = G(x,y,z);
g = x^2 + y^2 +  z^2 -1;
endfunction
```

10) Zur Untersuchung auf Irreduzibilität werden die entsprechenden Graphen $G(A)$, $G(B)$, $G(C)$ gezeichnet:

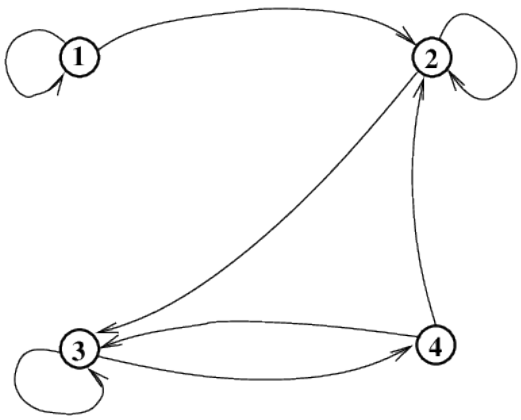

Abbildung 1:
Graph $G(A)$

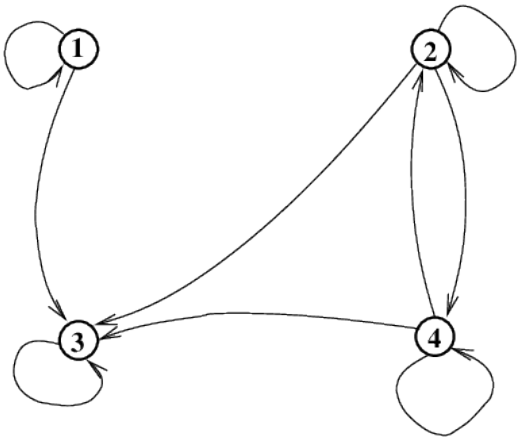

Abbildung 2:
Graph $G(B)$

Während man im Graphen $G(C)$ jeden Knoten von jedem Knoten durch einen gerichteten Weg erreichen kann, ist dies in den Graphen $G(A)$ und $G(B)$ nicht möglich. Deshalb ist die Matrix C irreduzibel und die Matrizen A und B sind nicht irreduzibel.

11) Durch den Tausch der Spalten 2 und 3 erhält man mit

$$\begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_3 \\ x_2 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

ein Gleichungssystem mit einer schwach diagonal dominanten, irreduziblen Koeffizientenmatrix. Mit dem Programm 7.3 erhält man nach 54 Iterationen die Lösung $\vec{x} = (x_1, x_3, x_2, x_4)^T = (4, 7, 8, 6)^T$.

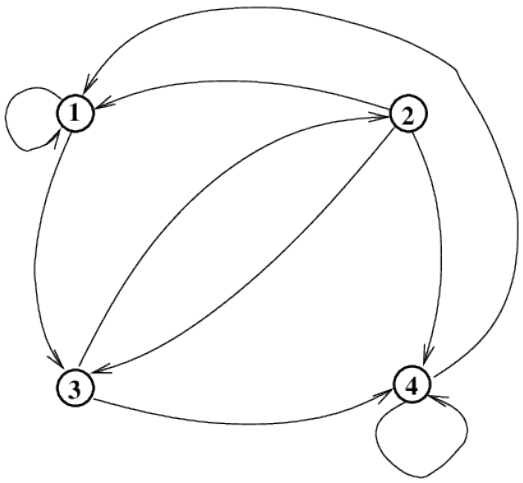

Abbildung 3:
Graph $G(C)$

12) Zu zeigen ist $A\vec{v}_k = \lambda_k \vec{v}_k$ für $k = 1, \ldots, n$, d.h.

$$\begin{aligned}
a(\frac{c}{b})^{1/2} \sin\frac{k\pi}{n+1} + b(\frac{c}{b})^{2/2} \sin\frac{2k\pi}{n+1} &= (a + 2\sqrt{cb}\cos\frac{k\pi}{n+1})(\frac{c}{b})^{1/2} \sin\frac{k\pi}{n+1} \\
c(\frac{c}{b})^{(j-1)/2} \sin\frac{(j-1)k\pi}{n+1} + a(\frac{c}{b})^{j/2} \sin\frac{jk\pi}{n+1} + b(\frac{c}{b})^{(j+1)/2} \sin\frac{(j+1)k\pi}{n+1} & \\
= (a + 2\sqrt{cb}\cos\frac{k\pi}{n+1})(\frac{c}{b})^{j/2} \sin\frac{jk\pi}{n+1} \quad (j = 2, \ldots, n-1) & \\
c(\frac{c}{b})^{(n-1)/2} \sin\frac{(n-1)k\pi}{n+1} + a(\frac{c}{b})^{n/2} \sin\frac{nk\pi}{n+1} = (a + 2\sqrt{cb}\cos\frac{k\pi}{n+1})(\frac{c}{b})^{n/2} \sin\frac{nk\pi}{n+1} \; . &
\end{aligned}$$

Die Gültigkeit der ersten Gleichung ergibt sich durch Anwendung des Additionstheorems $\sin(2x) = 2\sin x \cos x$. Die Gleichungen 2 bis $n-1$ gelten aufgrund der Identität

$$\sin\alpha + \sin\beta = 2\sin\frac{\alpha+\beta}{2}\cos\frac{\alpha-\beta}{2} \tag{2}$$

mit $\alpha = \frac{(j+1)k\pi}{n+1}$ und $\beta = \frac{(j-1)k\pi}{n+1}$. Die n-te Gleichung gilt wegen

$$\sin\frac{(n-1)k\pi}{n+1} = \sin\frac{(n+1)k\pi}{n+1} + \sin\frac{(n-1)k\pi}{n+1} = 2\sin\frac{nk\pi}{n+1}\cos\frac{k\pi}{n+1} \; ,$$

wobei $\sin\frac{(n+1)k\pi}{n+1} = \sin k\pi = 0$ und die Identität (2) benutzt wurde.

13) Für $a = 2$ und $c = b = -1$ hat die Matrix A die Eigenwerte $\lambda_k = 2 + 2\cos\frac{k\pi}{n+1}$, $k = 1, \ldots, n$. Für den Spektralradius folgt

$$\rho(A) = \max_{1\le k\le n} |\lambda_k| = 2 + 2\cos\frac{\pi}{n+1} \; .$$

Für die Eigenwerte μ_k von A^{-1} folgt

$$\mu_k = \frac{1}{\lambda_k} = \frac{1}{2 + 2\cos\frac{k\pi}{n+1}} \; , \quad k = 1, \ldots, n \; .$$

Mit

$$\min_{1\leq k\leq n} |\lambda_k| = 2 + 2\cos\frac{([\frac{n}{2}]+1)\pi}{n+1}$$

ergibt sich für den Spektralradius von A^{-1}

$$\rho(A^{-1}) = \frac{1}{2 + 2\cos\frac{([\frac{n}{2}]+1)\pi}{n+1}} \ .$$

Mit der Spektralnorm ergibt sich die Konditionszahl der Matrix A zu

$$\text{cond}(A) = ||A||_2||A^{-1}||_2 = \frac{2 + 2\cos\frac{\pi}{n+1}}{2 + 2\cos\frac{([\frac{n}{2}]+1)\pi}{n+1}} \ .$$

Für $n = 101$ ist $[\frac{n}{2}] = 50$ und man erhält die Konditionszahl

$$\text{cond}(A) = 1{,}9995 \ .$$

Für größere Dimensionen steigt die Konditionszahl an, ist aber durch 2 beschränkt.

Kapitel 8

1a) Bei Vorgabe von $a_1 = a_2 = 1$ und unter Berücksichtigung der generellen Fixierung von $a_3 = 1$ erhält man zur Bestimmung der Koeffizienten das Gleichungssystem

$$\begin{aligned}
a_0 &= -a_1 - a_2 - a_3 = -3 \\
-b_0 - b_1 - b_2 - b_3 &= -a_1 - 2a_2 - 3a_3 = -6 \\
-b_1 - 2b_2 - 3b_3 &= -\frac{1}{2}a_1 - 2a_2 - \frac{9}{2}a_3 = -7 \\
-\frac{1}{2}b_1 - 2b_2 - \frac{9}{2}b_3 &= -\frac{1}{6}a_1 - \frac{4}{3}a_2 - \frac{9}{2}a_3 = -6 \\
-\frac{1}{24}b_1 - \frac{2}{3}b_2 - \frac{27}{8}b_3 &= -\frac{1}{120}a_1 - \frac{32}{120}a_2 - \frac{243}{120}a_3 = -\frac{23}{10}\ ,
\end{aligned}$$

also 5 Gleichungen für die 5 Unbekannten $a_0, b_0, \ldots, b_3$. Mit der Lösung

$$a_0 = -3,\ b_0 = 1{,}1333\ldots,\ b_1 = 3{,}1,\ b_2 = 1{,}4,\ b_3 = 0{,}3666\ldots$$

erhält man das 3-Schritt-Verfahren

$$y_{k+1} + y_k + y_{k-1} - 3y_{k-2} = h[0{,}3666 f_{k+1} + 1{,}4 f_k + 3{,}1 f_{k-1} + 1{,}1333 f_{k-2}]$$

mit der maximalen Ordnung 4. Für das erste bzw. zweite charakteristische Polynom erhält man

$$\rho(z) = -3 + z + z^2 + z^3 \quad \text{bzw.} \quad \sigma(z) = 1{,}1333 + 3{,}1z + 1{,}4z^2 + 0{,}3666z^3\ .$$

1b) Bei Vorgabe von $a_0 = a_1 = 0$ und unter Berücksichtigung der generellen Fixierung von $a_3 = 1$ erhält man zur Bestimmung der Koeffizienten das Gleichungssystem

$$\begin{aligned}
a_2 &= -1 \\
2a_2 - b_0 - b_1 - b_2 - b_3 &= -3 \\
2a_2 - b_1 - 2b_2 - 3b_3 &= -4{,}5 \\
\frac{4}{3}a_2 - \frac{1}{2}b_1 - 2b_2 - \frac{9}{2}b_3 &= -4{,}5 \\
\frac{32}{120}a_2 - \frac{1}{24}b_1 - \frac{2}{3}b_2 - \frac{27}{8}b_3 &= -\frac{243}{120}
\end{aligned}$$

mit der Lösung

$$a_2 = -1,\ b_0 = \frac{296}{4995},\ b_1 = -\frac{47}{180},\ b_2 = \frac{38}{45},\ b_3 = \frac{7141}{19980}\ .$$

Damit ergibt sich das 3-Schritt-Verfahren

$$y_{k+1} - y_k = h[\frac{7141}{19980} f_{k+1} + \frac{38}{45} f_k - \frac{47}{180} f_{k-1} + \frac{296}{4995} f_{k-2}]$$

mit der Ordnung 3. Für das erste bzw. zweite charakteristische Polynom erhält man

$$\rho(z) = -z^2 + z^3 \quad \text{bzw.} \quad \sigma(z) = \frac{296}{4995} - \frac{47}{180}z + \frac{38}{45}z^2 + \frac{7141}{19980}z^3\ .$$

2) Mit den charakteristischen Polynomen des Verfahrens 1a) erhält man mit

$$\mu(\varphi) = \frac{-3 + e^{i\,\varphi} + e^{i\,2\varphi} + e^{i\,3\varphi}}{1{,}1333 + 3{,}1e^{i\,\varphi} + 1{,}4e^{i\,2\varphi} + 0{,}3666e^{i\,3\varphi}} \quad (\varphi \in [0, 2\pi])$$

die Randkurve des Gebietes der absoluten Stabilität des in 1a) konstruierten Verfahrens. Ein Plot der Randkurve zeigt, dass das Gebiet der absoluten Stabilität die gesamte linke Halbebene umfasst.
Mit den charakteristischen Polynomen des Verfahrens 1b) erhält man mit

$$\mu(\varphi) = \frac{-e^{i\,2\varphi} + e^{i\,3\varphi}}{\frac{296}{4995} - \frac{47}{180}e^{i\,\varphi} + \frac{38}{45}e^{i\,2\varphi} + \frac{7141}{19980}e^{i\,3\varphi}} \quad (\varphi \in [0, 2\pi])$$

die Randkurve des Gebietes der absoluten Stabilität des in 1b) konstruierten Verfahrens.

3) Es sollen hier nur das explizite Eulerverfahren, die verbesserte Polygonzugmethode und das Heun-Verfahren 3. Ordnung skizziert werden. Mit den Festlegungen $\vec{y}(t) = (x(t), y(t))^T$ und

$$\vec{y}\,' = \vec{F}(t, \vec{y}) = \vec{F}(\vec{y}) \iff \begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \begin{pmatrix} f_1(\vec{y}) \\ f_2(\vec{y}) \end{pmatrix} := \begin{pmatrix} \frac{y(t)}{x(t)^2+y(t)^2} \\ \frac{-x(t)}{x(t)^2+y(t)^2} \end{pmatrix}$$

realisiert das folgende Programm die Verfahren

```
# Programm aufg8.3
# Aufruf: [xx,yy] = aufg83(h,iwahl);
# iwahl: 1 Euler explizit, 2 verbesserte Polygonzugmethode, 3 Heun 3. Ordn.
function [xx,yy] = aufg83(h,iwahl);
t = h;
k = 1;
x = 1;
y = 0;
while (t < 2*pi)
if (iwahl == 1)
# Euler explizit
  r2 = x**2+y**2;
  x = x + h*y/r2;
  y = y - h*x/r2;
elseif (iwahl == 2)
# verbesserte Polygonzugmethode
  r2 = x**2+y**2;
  k11 = y/r2;k12 = -x/r2;
  r2 = (x+0.5*h*k11)**2 + (y+0.5*h*k12)**2;
  k21 = (y+0.5*h*k12)/r2; k22 = -(x+0.5*h*k11)/r2;
  x = x + h*k21;
  y = y + h*k22;
elseif (iwahl == 3)
# Heun-Verfahren 3. Ordnung
  r2 = x**2+y**2;
  k11 = y/r2;
  k12 = -x/r2;
  r2 = (x+h*k11/3)**2 + (y+h*k12/3)**2;
  k21 = (y+h*k12/3)/r2;
  k22 = -(x+h*k11/3)/r2;
  r2 = (x+2*h*k21/3)**2 + (y+2*h*k22/3)**2;
  k31 = (y+2*h*k22/3)/r2;
  k32 = -(x+2*h*k21/3)/r2;
  x = x + 0.25*h*(k11 + 3*k31);
  y = y + 0.25*h*(k12 + 3*k32);
endif
t = t + h;
xx(k) = x;
yy(k) = y;
k = k + 1;
endwhile
endfunction
```

Mit der Vorgabe der Schrittweite h kann man nun Tests durchführen.

4) Die Fixpunktiteration für das implizite Runge-Kutta-Verfahren kann man im Berech-

nungsschritt von y_k zu y_{k+1} in der Form

$$\begin{aligned} k_1^{(s+1)} &= f(x_k + \frac{3-\sqrt{3}}{6}h, y_k + \frac{1}{4}hk_1^{(s)} + \frac{3-2\sqrt{3}}{12}hk_2^{(s)}) \\ k_2^{(s+1)} &= f(x_k + \frac{3+\sqrt{3}}{6}h, y_k + \frac{3+2\sqrt{3}}{12}hk_1^{(s+1)} + \frac{1}{4}hk_2^{(s)}) \end{aligned}$$

ansetzen, also einer Iteration vergleichbar mit der Gauß-Seidel-Iteration, die sehr einfach zu implementieren ist. Ist $|k_j^{(s+1)} - k_1^{(s)}| < \epsilon$, $j = 1{,}2$ für eine geforderte Genauigkeit $\epsilon > 0$, dann erfolgt der Schritt

$$y_{k+1} = y_k + \frac{h}{2}[k_1^{(s+1)} + k_2^{(s+1)}] \, .$$

Das folgende Programm realisiert die Methode.

```
# Programm aufg8.4
# zur Loesung von gew. DGL 1. Ordnung y'=f(x,y), y(x0)=y0
# implizites Runge-Kutta-Verfahren 4. Ordnung
# input:  Schrittweite h, Intervallgrenzen x0,x1, Anfangswert y0=y(x0)
# output: Loesung y(x) an den Punkten x
# benutztes Programm: Nutzerfunktion f (rechte Seite der Dgl.)
# Aufruf: [y,x] = aufg84(x0,x1,y0,h,epsilon)
function [y,x] = aufg84(x0,x1,y0,h,epsilon);
n = (x1 - x0)/h;
yn = y0; y(1)=y0;
xn = x0; x(1)=x0;
for k=1:n
   k1 = f(xn,yn);k2=f(xn,yn);k1a = k1+epsilon;k2a = k2+epsilon;
   while (abs(k1-k1a)+abs(k2-k2a) > epsilon)
     k1a = k1;k2a = k2;
     k1 = f(xn + h*(3-sqrt(3))/6,yn + h*k1/4 + h*k2*(3-2*sqrt(3))/12);
     k2 = f(xn + h*(3+sqrt(3))/6,yn + h*k1*(3-2*sqrt(3))/12 + h*k2/4);
   endwhile
   yn = yn + 0.5*h*(k1+k2);
   xn = xn+h; x(k+1)=xn; y(k+1)=yn;
endfor
endfunction
```

Zusammen mit dem Programm

```
# Programm zur Bereitstellung
# der rechten Differentialgleichungsseite f(x,y)
function [y] = f(x,y);
   y = y/(1.+x)+(x+1)*cos(x);
endfunction
```

wird das Anfangswertproblem

$$y'(x) = \frac{y}{1+x} + (x+1)\cos x \, , \quad 0 < x < 4, \; y(0) = 1$$

durch die Octave-Kommandozeilen

```
x0=0;x1=4;y0=1;h=0.01;eps=0.00001;
[y,x] = aufg84(x0,x1,y0,h,eps);
```

gelöst. Dabei waren pro Zeitintegrationsschritt nur maximal 3 Fixpunktiterationen erforderlich, um die geforderte Genauigkeit $\epsilon = 10^{-5}$ zu erreichen.

5) Das Verfahren 1a) ist nicht nullstabil, da das erste charakteristische Polynom Nullstellen besitzt, die dem Betrage nach größer als 1 sind. Probieren Sie das Verfahren mit dem Programm **aufg85a.m** aus.
Das Verfahren 1b) wird mit dem folgenden Programm realisiert.

```
# Programm aufg8.5b
# 3-Schritt-Verfahren dritter Ordnung
# zur Loesung von gew. DGL 1. Ordnung y'=f(x,y), y(x0)=y0
# input:  Schrittweite h, Intervallgrenzen x0,x1, Anfangswert y0=y(x0)
# output: Loesung y(x) an den Punkten x
# benutztes Programm: Nutzerfunktion f (rechte Seite der Dgl.)
#      und Heun-Verfahren dritter Ordnung (heun3_gb.m) zur Berechnung
#      von y1, y2 als Startwerte fuer das 3-Schritt-Verfahren
# Aufruf: [y,x] = aufg85b(x0,x1,y0,h,epsilon)
function [y,x] = aufg85b(x0,x1,y0,h,epsilon);
n = (x1 - x0)/h;
yn = y0; y(1)=y0;
xn = x0; x(1)=x0;
# Berechnung von y1 und y2
[y,x] = heun3_gb(x0,x0+2*h,y0,h)
yn = y(3); xn = x0+2*h;
# Berechnung von y3, y4,...
for k=3:n
   f3=f(xn+h,yn);
   f3a=f3+epsilon;
   f2=f(x(k),y(k));f1=f(x(k-1),y(k-1));f0=f(x(k-2),y(k-2));
   b = h*(38/45*f2-47/180*f1+296/4995*f0)+y(k);
   itfix = 0;
   while (abs(f3-f3a) > epsilon)
     f3a = f(xn+h,yn);
     yn = h*7141/19980*f3a + b;
     f3 = f(xn+h,yn);
     itfix = itfix+1
     f3-f3a
   endwhile
   xn = xn+h; x(k+1)=xn; y(k+1)=yn;
endfor
endfunction
```

Da es sich bei dem 3-Schritt-Verfahren um ein implizites Verfahren handelt, ist pro Zeitintegrationsschritt eine Fixpunktiteration im Programm **aufg85b.m** implementiert. In der Regel sind jedoch nur 2 bis 3 Fixpunktiterationen pro Zeitschritt erforderlich.

6) Zur Lösung der Aufgabe kann man das Programm 9.3 (**rwp2nl_gb.m**) nutzen. Die Octave-Kommandozeile

```
[y,x] = rwp2nl_gb(1.,6.79e-12,0,10,3000,105.4,100);
```

erzeugt mit dem Vektor y an den Stützstellen x die numerische Lösung. Dabei waren 7 Newton-Iterationen nötig, um den Defekt des bei der Diskretisierung entstandenen nichtlinearen Gleichungssystems von 10^3 auf 10^{-10} zu verringern.

7) Grundlage für die Lösung der Zweipunkt-Randwertaufgabe

$$y'' = \sigma y^4 \ , \ 0 < x < 10, \ y(0) = y_a, \ y(10) = y_b$$

ist das Programm 8.5 (**schiessv_gb.m**). Es sind letztlich die beiden Anfangswertprobleme

$$y'' = \sigma y^4 \ , \ 0 < x \leq 10, \ y(0) = y_a, \ y'(0) = \zeta,$$

bzw.

$$y''_\zeta = 4\sigma y^3 y_\zeta \ , \ 0 < x \leq 10, \ y_\zeta(0) = 0, \ y'_\zeta(0) = 1$$

pro Newton-Iterationsschritt zu lösen. Für die konkrete rechte Seite $f(x, y) = \sigma y^4$ müssen die Programme 8.5d, 8.5e angepasst werden. Es ergibt sich

```
# Programm aufg8.7a
function [y] = fs2(x,y1,y2);
   sigma = 6.79e-12;
   y = sigma*y1**4;
```

```
endfunction
# Programm aufg8.7b
function [y] = fsy2(x,yk,y1,y2);
   sigma = 6.79e-12;
   y = sigma*4*yk**3*y1;
endfunction
```

Mit den Octave-Kommandozeilen

```
a=0;b=10;eta0=3000;eta1=770;n=100;
[y1,x] = schiessv_gb(a,b,eta0,eta1,n,-765);
```

erhält man für ζ und $g(\zeta)$ die Resultate der nachfolgenden Tabelle.

Newton-Iteration	ζ	$g(\zeta)$
1	$-773{,}62$	$8764{,}6$
2	$-788{,}89$	$4020{,}0$
3	$-807{,}13$	$1676{,}1$
4	$-816{,}97$	$486{,}87$
5	$-818{,}46$	$59{,}747$
7	$-818{,}52$	$2{,}2364$
8	$-818{,}52$	$0{,}056316$
9	$-818{,}52$	$3{,}4381e-05$
10	$-818{,}52$	$8{,}4929e-07$

Hierzu ist allerdings anzumerken, dass das konvergente Newton-Verfahren erst nach verschiedenen Fehlversuchen mit ungeeigneten Starwerten für ζ schließlich mit dem Startwert $\zeta = -765$ erhalten wurde.
Die Wahl dieses Startwerts basierte auf einer genaueren Betrachtung der Lösung der Aufgabe 6).

Kapitel 9

1) Sei ω_{ij} eine FV-Kontrollzelle mit dem Stützwert u_{ij} im Zentrum. Δx und Δy seien die Länge und Breite von ω_{ij}. Das Gebiet Ω wird richtungsäquidistant in $N \times M$ Kontrollzellen unterteilt. Als Randbedingungen werden etwas allgemeiner

$$u = u_r \quad \text{auf } \Gamma_1, \qquad \text{grad}\, u \cdot \vec{n} = q \quad \text{auf } \Gamma_2$$

vorgegeben. Außerdem wird in der Gleichung ein Quell-Senken-Glied f auf der rechten Seite berücksichtigt. Die Geschwindigkeit $\vec{v} = (v, w)^T$ ist auf $\Omega \cup \partial\Omega$ gegeben. Für den konvektiven Term ergibt sich die Bilanz

$$\begin{aligned}\int_{\omega_{ij}} \text{div}\,(u\vec{v})\,dF &= \int_{\partial\omega_{ij}} u\vec{v}\cdot\vec{n}\,ds \\ &\approx [v_{i+1/2j}\frac{u_{ij}+u_{i+1j}}{2} - v_{i-1/2j}\frac{u_{ij}+u_{i-1j}}{2}]\Delta y \\ &\quad +[w_{ij+1/2}\frac{u_{ij}+u_{ij+1}}{2} - w_{ij-1/2}\frac{u_{ij}+u_{ij-1}}{2}]\Delta x\ .\end{aligned}$$

Für den Diffusionsterm erhält man

$$\begin{aligned}\int_{\omega_{ij}} \text{div}\,(\lambda\text{grad}\,u)\,dF &= \int_{\partial\omega_{ij}} \lambda\text{grad}\,u\cdot\vec{n}\,ds \\ &\approx \lambda[\frac{u_{i+1j}-u_{ij}}{\Delta x} - \frac{u_{ij}-u_{i-1j}}{\Delta x}]\Delta y + \lambda[\frac{u_{ij+1}-u_{ij}}{\Delta y} - \frac{u_{ij}-u_{ij-1}}{\Delta y}]\Delta x\ .\end{aligned}$$

Nach Division durch $\Delta x \Delta y$ ergibt sich insgesamt die Gleichung

$$\begin{aligned}[v_{i+1/2j}\frac{u_{ij}+u_{i+1j}}{2} - v_{i-1/2j}\frac{u_{ij}+u_{i-1j}}{2}]/\Delta x + [w_{ij+1/2}\frac{u_{ij}+u_{ij+1}}{2} - w_{ij-1/2}\frac{u_{ij}+u_{ij-1}}{2}]/\Delta y \\ +\lambda[\frac{2u_{ij}-u_{i+1j}-u_{i-1j}}{\Delta x^2} + \frac{2u_{ij}-u_{ij+1}-u_{ij-1}}{\Delta y^2}] = f_{ij}\end{aligned}$$

für innere Zellen ω_{ij}. Für Randzellen an Γ_1, d.h. $i = 1, 1 < j < M$, ergibt sich mit

$$\frac{u_{ij}+u_{i-1j}}{2} = u_r \iff u_{i-1j} = 2u_r - u_{ij}$$

die Randgleichung

$$\begin{aligned}[v_{i+1/2j}\frac{u_{ij}+u_{i+1j}}{2} - v_{i-1/2j}u_r]/\Delta x + [w_{ij+1/2}\frac{u_{ij}+u_{ij+1}}{2} - w_{ij-1/2}\frac{u_{ij}+u_{ij-1}}{2}]/\Delta y \\ +\lambda[\frac{3u_{ij}-u_{i+1j}-2u_r}{\Delta x^2} + \frac{2u_{ij}-u_{ij+1}-u_{ij-1}}{\Delta y^2}] = f_{ij}\ .\end{aligned}$$

Für Randzellen an Γ_2 ergibt sich für $1 < i < N, j = 1$ mit

$$(u_{ij} - u_{ij-1})/\Delta y = -q \iff u_{ij-1} = u_{ij} + q\Delta y$$

die Randgleichung

$$\begin{aligned}[v_{i+1/2j}\frac{u_{ij}+u_{i+1j}}{2} - v_{i-1/2j}\frac{u_{ij}+u_{i-1j}}{2}]/\Delta x + [w_{ij+1/2}\frac{u_{ij}+u_{ij+1}}{2} - w_{ij-1/2}(u_{ij}+\frac{q\Delta y}{2})]/\Delta y \\ +\lambda[\frac{2u_{ij}-u_{i+1j}-u_{i-1j}}{\Delta x^2} + \frac{u_{ij}-u_{ij+1}-q\Delta y}{\Delta y^2}] = f_{ij}\ ,\end{aligned}$$

für $i = N, 1 < j < M$ mit

$$(u_{i+1j} - u_{ij})/\Delta x = q \iff u_{i+1j} = u_{ij} + q\Delta x$$

die Randgleichung

$$[v_{i+1/2j}(u_{ij} + \frac{q\Delta x}{2}) - v_{i-1/2j}\frac{u_{ij} + u_{i-1j}}{2}]/\Delta x + [w_{ij+1/2}\frac{u_{ij} + u_{ij+1}}{2} - w_{ij-1/2}\frac{u_{ij} + u_{ij-1}}{2}]/\Delta y$$
$$+\lambda[\frac{u_{ij} - u_{i-1j} - q\Delta x}{\Delta x^2} + \frac{2u_{ij} - u_{ij+1} - u_{ij-1}}{\Delta y^2}] = f_{ij}$$

und für $1 < i < N, j = M$ mit

$$(u_{ij+1} - u_{ij})/\Delta y = q \iff u_{ij+1} = u_{ij} + q\Delta y$$

die Randgleichung

$$[v_{i+1/2j}\frac{u_{ij} + u_{i+1j}}{2} - v_{i-1/2j}\frac{u_{ij} + u_{i-1j}}{2}]/\Delta x + [w_{ij+1/2}(u_{ij} + \frac{q\Delta y}{2}) - w_{ij-1/2}\frac{u_{ij} + u_{ij-1}}{2}]/\Delta y$$
$$+\lambda[\frac{2u_{ij} - u_{i+1j} - u_{i-1j}}{\Delta x^2} + \frac{u_{ij} - u_{ij-1} - q\Delta y}{\Delta y^2}] = f_{ij}\ .$$

An den Eckpunkten erhält man für $i = 1, j = 1$ die Gleichung

$$[v_{i+1/2j}\frac{u_{ij} + u_{i+1j}}{2} - v_{i-1/2j}u_r]/\Delta x + [w_{ij+1/2}\frac{u_{ij} + u_{ij+1}}{2} - w_{ij-1/2}(u_{ij} + \frac{q\Delta y}{2})]/\Delta y$$
$$+\lambda[\frac{3u_{ij} - u_{i+1j} - 2u_r}{\Delta x^2} + \frac{u_{ij} - u_{ij+1} - q\Delta y}{\Delta y^2}] = f_{ij}\ ,$$

für $i = 1, j = M$ die Gleichung

$$[v_{i+1/2j}\frac{u_{ij} + u_{i+1j}}{2} - v_{i-1/2j}u_r]/\Delta x + [w_{ij+1/2}(u_{ij} + \frac{q\Delta y}{2}) - w_{ij-1/2}\frac{u_{ij} + u_{ij-1}}{2}]/\Delta y$$
$$+\lambda[\frac{3u_{ij} - u_{i+1j} - 2u_r}{\Delta x^2} + \frac{u_{ij} - u_{ij-1} - q\Delta y}{\Delta y^2}] = f_{ij}\ ,$$

für $i = N, j = 1$ die Gleichung

$$[v_{i+1/2j}(u_{ij} + \frac{q\Delta x}{2}) - v_{i-1/2j}\frac{u_{ij} + u_{i-1j}}{2}]/\Delta x + [w_{ij+1/2}\frac{u_{ij} + u_{ij+1}}{2} - w_{ij-1/2}(u_{ij} + \frac{q\Delta y}{2})]/\Delta y$$
$$+\lambda[\frac{u_{ij} - u_{i-1j} - q\Delta x}{\Delta x^2} + \frac{u_{ij} - u_{ij+1} - q\Delta y}{\Delta y^2}] = f_{ij}$$

und schließlich für $i = N, j = M$ die Gleichung

$$[v_{i+1/2j}(u_{ij} + \frac{q\Delta x}{2}) - v_{i-1/2j}\frac{u_{ij} + u_{i-1j}}{2}]/\Delta x + [w_{ij+1/2}(u_{ij} + \frac{q\Delta y}{2}) - w_{ij-1/2}\frac{u_{ij} + u_{ij-1}}{2}]/\Delta y$$
$$+\lambda[\frac{u_{ij} - u_{i-1j} - q\Delta x}{\Delta x^2} + \frac{u_{ij} - u_{ij-1} - q\Delta y}{\Delta y^2}] = f_{ij}\ .$$

Die Lösung des Gleichungssystems für die Unbekannten u_{ij}, $i = 1, \ldots, N, j = 1, \ldots, M$ kann direkt oder iterativ erfolgen. Als Orientierung für die Implementierung der iterativen Lösung kann man das Programm 9.1 verwenden. Im nachfolgenden Programm wurde ein SOR-Verfahren realisiert. Allerdings wurde aufgrund der Rechteckgeometrie ohne Gebietsidentifikationsmatrix gearbeitet.

```
# Programm zur Loesung der Aufgabe 9.1
# mit Dirichlet- und Neumann-RB, iterative Loesung
# input: n,m Stuetzpunkte in x- und y-Richtung
# output: Loesung u(x,y) auf dem Gitter x,y,
#         udiff Abstand zweier Loesungen, it Iterationszahl
# Aufruf: [u,x,y,udiff,it] = aufg91i(n,m,omega,epsilon,itmax);
function [u,x,y,udiff,it] = aufg91i(n,m,omega,epsilon,itmax);
# Materialdaten, Quellen
la = 3.0; ff=0;
# Gebeitslaenge, Gebietshoehe
ll = 2; hh = 1;
# Dimensionierung des Gitters
dx = ll/n; dy = hh/m;
h = dx; k = dy;
# Gitterinitialisierung
x(1) = dx/2; y(1) = dy/2;
for i=1:n
  x(i) = (i-1)*dx+x(1);
endfor
for j=1:m
  y(j) = (j-1)*dy+y(1);
endfor
# Startfeld
for i=1:n
  for j=1:m
    u(i,j) = 1.0;
  endfor
endfor
# Konstanten
h2=h^2; k2=k^2; dx2=2*dx; dy2=2*dy;
udiff = 1; unorm = 1; it = 0;
# Randwerte
uw=1; q1=0; q2=0; q3=0;
# Aufbau der Matrix und der rechten Seite
for i=1:n
    for j=1:m
      ind = i+(j-1)*n;
      vm = (x(i)-0.5*dx)^2; vp = (x(i)+0.5*dx)^2;
      wm = (1-y(j)+0.5*dy)^2; wp = (1-y(j)-0.5*dy)^2;
      if (1 < i && i < n && 1 < j && j < m )
        a(ind,ind) = 2*la/h2+2*la/k2 + (vp-vm)/dx2 + (wp-wm)/dy2;
        a(ind,ind+1) = -(la/h2 - vp/dx2);
        a(ind,ind-1) = -(la/h2 + vm/dx2);
        a(ind,ind+n) = -(la/k2 - wp/dy2);
        a(ind,ind-n) = -(la/k2 + wm/dy2);
        ff1(ind) = ff;
      endif
      if (i == 1 && 1 < j && j < m )
        a(ind,ind) = 3*la/h2+2*la/k2 + vp/dx2 + (wp-wm)/dy2;
        a(ind,ind+1) = -(la/h2 - vp/dx2);
        a(ind,ind+n) = -(la/k2 - wp/dy2);
        a(ind,ind-n) = -(la/k2 + wm/dy2);
        ff1(ind) = ff+vm*uw/h+la*2*uw/h2;
      endif
      if (1 < i && i < n && j == 1 )
        a(ind,ind) = 2*la/h2+la/k2 + (vp-vm)/dx2 + wp/dy2 -wm/dy;
        a(ind,ind+1) = -(la/h2 - vp/dx2);
        a(ind,ind-1) = -(la/h2 + vm/dx2);
        a(ind,ind+n) = -(la/k2 - wp/dy2);
        ff1(ind) = ff+wm*q1*k/dy2+la*q1/k;
      endif
      if (i == n && 1 < j && j < m )
        a(ind,ind) = la/h2+2*la/k2 + vp/dx -vm/dx2 + (wp-wm)/dy2;
        a(ind,ind-1) = -(la/h2 + vm/dx2);
        a(ind,ind+n) = -(la/k2 - wp/dy2);
        a(ind,ind-n) = -(la/k2 + wm/dy2);
        ff1(ind) = ff-vp*q2/2+la*q2/h;
      endif
      if (1 < i && i < n && j == m )
        a(ind,ind) = 2*la/h2+la/k2 + (vp-vm)/dx2 + wp/dy - wm/dy2;
        a(ind,ind+1) = -(la/h2 - vp/dx2);
        a(ind,ind-1) = -(la/h2 + vm/dx2);
        a(ind,ind-n) = -(la/k2 + wm/dy2);
        ff1(ind) = ff-wp*q3/2+la*q3/k;
      endif
```

```
      if (i == 1 && j == 1 )
        a(ind,ind) = 3*la/h2+la/k2 + vp/dx2 + wp/dy2 - wm/dy;
        a(ind,ind+1) = -(la/h2 - vp/dx2);
        a(ind,ind+n) = -(la/k2 - wp/dy2);
        ff1(ind) = ff+vm*uw/h+wm*q1/2+la*2*uw/h2+la*q1/k;
      endif
      if (i == 1 && j == m )
        a(ind,ind) = 3*la/h2+la/k2 + vp/dx2 + wp/dy - wm/dy2;
        a(ind,ind+1) = -(la/h2 - vp/dx2);
        a(ind,ind-n) = -(la/k2 + wm/dy2);
        ff1(ind) = ff + vm*uw/h - wp*q3/2 + la*2*uw/h2 +la*q3/k;
      endif
      if (i == n && j == 1 )
        a(ind,ind) = la/h2+la/k2 + vp/dx - vm/dx2 + wp/dy2 - wm/dy;
        a(ind,ind-1) = -(la/h2 + vm/dx2);
        a(ind,ind+n) = -(la/k2 - wp/dy2);
        ff1(ind) = ff-vp*q2/2+wm*q1/2+la*q2/h+la*q1/k;
      endif
      if (i == n && j == m )
        a(ind,ind) = la/h2+la/k2 + vp/dx - vm/dx2 + wp/dy - wm/dy2;
        a(ind,ind-1) = -(la/h2 + vm/dx2);
        a(ind,ind-n) = -(la/k2 + wm/dy2);
        ff1(ind) = ff-vp*q2/2-wp*q3/2+la*q2/h+la*q3/k;
      endif
    endfor
endfor
# SOR-Iteration
while (sqrt(udiff/unorm) > epsilon && it < itmax)
  ualt = u;
  udiff
  it
#
  for i=1:n
    for j=1:m
      ind = i+(j-1)*n;
      if (1 < i && i < n && 1 < j && j < m )
        u(i,j)=(ff1(ind)-a(ind,ind+1)*u(i+1,j)-a(ind,ind-1)*u(i-1,j) ...
                -a(ind,ind+n)*u(i,j+1)-a(ind,ind-n)*u(i,j-1))/a(ind,ind);
      endif
      if (i == 1 && 1 < j && j < m )
        u(i,j)=(ff1(ind)-a(ind,ind+1)*u(i+1,j)-a(ind,ind+n)*u(i,j+1) ...
                -a(ind,ind-n)*u(i,j-1))/a(ind,ind);
      endif
      if (1 < i && i < n && j == 1 )
        u(i,j)=(ff1(ind)-a(ind,ind+1)*u(i+1,j)-a(ind,ind-1)*u(i-1,j) ...
                -a(ind,ind+n)*u(i,j+1))/a(ind,ind);
      endif
      if (i == n && 1 < j && j < m )
        u(i,j)=(ff1(ind)-a(ind,ind-1)*u(i-1,j)-a(ind,ind+n)*u(i,j+1) ...
                -a(ind,ind-n)*u(i,j-1))/a(ind,ind);
      endif
      if (1 < i && i < n && j == m )
        u(i,j)=(ff1(ind)-a(ind,ind+1)*u(i+1,j)-a(ind,ind-1)*u(i-1,j) ...
                -a(ind,ind-n)*u(i,j-1))/a(ind,ind);
      endif
      if (i == 1 && j == 1 )
        u(i,j)=(ff1(ind)-a(ind,ind+1)*u(i+1,j)-a(ind,ind+n)*u(i,j+1))/a(ind,ind);
      endif
      if (i == 1 && j == m )
        u(i,j)=(ff1(ind)-a(ind,ind+1)*u(i+1,j)-a(ind,ind-n)*u(i,j-1))/a(ind,ind);
      endif
      if (i == n && j == 1 )
        u(i,j)=(ff1(ind)-a(ind,ind-1)*u(i-1,j)-a(ind,ind+n)*u(i,j+1))/a(ind,ind);
      endif
      if (i == n && j == m )
        u(i,j)=(ff1(ind)-a(ind,ind-1)*u(i-1,j)-a(ind,ind-n)*u(i,j-1))/a(ind,ind);
      endif
    endfor
  endfor
  udiff = 0.0; unorm = 0.0;
  for i=1:n
    for j=1:m
      u(i,j) = omega*u(i,j)+(1-omega)*ualt(i,j);
      udiff = udiff + (u(i,j)-ualt(i,j))^2;
      unorm = unorm + ualt(i,j)^2;
```

```
    endfor
  endfor
  if (unorm == 0.0)
    unorm = 1;
  endif
  it = it + 1;
endwhile
endfunction
```

Im Programm wurde die Koeffizientenmatrix **a** und die rechte Seite **ff1** des linearen Gleichungssystems aufgebaut, um auch die Möglichkeit der direkten Lösung zu haben. Die Koeffizientenmatrix des zu lösenden linearen Gleichungssystems ist aufgrund der konvektiven Glieder der Differentialgleichung nicht symmetrisch und positiv definit. Deshalb gibt es keinen Bereich für den Relaxationsparameter ω, wo die Konvergenz des SOR-Verfahrens gesichert ist. Für $\omega = 1{,}2$ lag bei einer Diskretisierung von 40×20 finiten Volumenzellen Konvergenz vor, während das SOR-Verfahren für $\omega = 1{,}6$ nicht konvergierte. Allerdings treten größere Probleme bei der Lösung erst dann auf, wenn die konvektiven Glieder die diffusiven dominieren.

2) Die Upwind-Approximation ergibt mit den Vereinbarungen

$$\begin{aligned}
vp_+ &= \frac{v_{i+1/2j} + |v_{i+1/2j}|}{2}, \quad vp_- = \frac{v_{i+1/2j} - |v_{i+1/2j}|}{2}, \\
vm_+ &= \frac{v_{i-1/2j} + |v_{i-1/2j}|}{2}, \quad vm_- = \frac{v_{i-1/2j} - |v_{i-1/2j}|}{2}, \\
wp_+ &= \frac{w_{ij+1/2} + |w_{ij+1/2}|}{2}, \quad wp_- = \frac{w_{ij+1/2} - |w_{ij+1/2}|}{2}, \\
wm_+ &= \frac{w_{ij-1/2} + |w_{ij-1/2}|}{2}, \quad wm_- = \frac{w_{ij-1/2} - |w_{ij-1/2}|}{2}
\end{aligned}$$

gemäß dem Vorgehen in Abschnitt 9.2.3 für den konvektiven Term die FV-Approximation

$$\begin{aligned}
&\int_{\omega_{ij}} \operatorname{div}(u\vec{v})\, dF = \int_{\partial\omega_{ij}} u\vec{v} \cdot \vec{n}\, ds \\
&\approx [vp_+ u_{ij} + vp_- u_{i+1j} - vm_+ u_{i-1j} - vm_- u_{ij}]\Delta y \\
&\quad + [wp_+ u_{ij} + wp_- u_{ij+1} - wm_+ u_{ij-1} - wm_- u_{ij}]\Delta x\,.
\end{aligned}$$

Mit der Bilanz des Diffusionsterms über ω_{ij} und nach Division durch $\Delta x \Delta y$ ergibt sich insgesamt die Gleichung

$$\begin{aligned}
&\frac{vp_+ - vm_-}{\Delta x} u_{ij} + \frac{vp_-}{\Delta x} u_{i+1j} - \frac{vm_+}{\Delta x} u_{i-1j} + \frac{wp_+ - wm_-}{\Delta y} u_{ij} + \frac{wp_-}{\Delta y} u_{ij+1} - \frac{wm_+}{\Delta y} u_{ij-1} \\
&\qquad + \lambda\left[\frac{2u_{ij} - u_{i+1j} - u_{i-1j}}{\Delta x^2} + \frac{2u_{ij} - u_{ij+1} - u_{ij-1}}{\Delta y^2}\right] = f_{ij}
\end{aligned}$$

für innere Zellen ω_{ij}. Analog zum Vorgehen bei der Aufgabe 1 werden durch Berücksichtigung der Randbedingungen in den FV-Zellen am Rand die jeweiligen Ghostwerte eliminiert und Rand- bzw. Eckgleichungen aufgestellt. Im Einzelnen erhält man:
Für Randzellen an Γ_1, d.h. $i = 1, 1 < j < M$, ergibt sich mit

$$\frac{u_{ij} + u_{i-1j}}{2} = u_r \iff u_{i-1j} = 2u_r - u_{ij}$$

die Randgleichung

$$\begin{aligned}
&\frac{vp_+ - vm_- + vm_+}{\Delta x} u_{ij} + \frac{vp_-}{\Delta x} u_{i+1j} - 2\frac{vm_+}{\Delta x} u_r + \frac{wp_+ - wm_-}{\Delta y} u_{ij} + \frac{wp_-}{\Delta y} u_{ij+1} - \frac{wm_-}{\Delta y} u_{ij-1} \\
&\qquad + \lambda\left[\frac{3u_{ij} - u_{i+1j} - 2u_r}{\Delta x^2} + \frac{2u_{ij} - u_{ij+1} - u_{ij-1}}{\Delta y^2}\right] = f_{ij}\,.
\end{aligned}$$

Für Randzellen an Γ_2 ergibt sich für $1 < i < N, j = 1$ mit

$$(u_{ij} - u_{ij-1})/\Delta y = -q \iff u_{ij-1} = u_{ij} + q\Delta y$$

die Randgleichung

$$\frac{vp_+ - vm_-}{\Delta x}u_{ij} + \frac{vp_-}{\Delta x}u_{i+1j} - \frac{vm_+}{\Delta x}u_{i-1j} + \frac{wp_+ - wm_- - wm_+}{\Delta y}u_{ij} + \frac{wp_-}{\Delta y}u_{ij+1} - wm_+q$$
$$+\lambda[\frac{2u_{ij} - u_{i+1j} - u_{i-1j}}{\Delta x^2} + \frac{u_{ij} - u_{ij+1} - q\Delta y}{\Delta y^2}] = f_{ij} \ ,$$

für $i = N, 1 < j < M$ mit

$$(u_{i+1j} - u_{ij})/\Delta x = q \iff u_{i+1j} = u_{ij} + q\Delta x$$

die Randgleichung

$$\frac{vp_+ - vm_- + vp_-}{\Delta x}u_{ij} + vp_-q - \frac{vm_+}{\Delta x}u_{i-1j} + \frac{wp_+ - wm_-}{\Delta y}u_{ij} + \frac{wp_-}{\Delta y}u_{ij+1} - \frac{wm_+}{\Delta y}u_{ij-1}$$
$$+\lambda[\frac{u_{ij} - u_{i-1j} - q\Delta x}{\Delta x^2} + \frac{2u_{ij} - u_{ij+1} - u_{ij-1}}{\Delta y^2}] = f_{ij}$$

und für $1 < i < N, j = M$ mit

$$(u_{ij+1} - u_{ij})/\Delta y = q \iff u_{ij+1} = u_{ij} + q\Delta y$$

die Randgleichung

$$\frac{vp_+ - vm_-}{\Delta x}u_{ij} + \frac{vp_-}{\Delta x}u_{i+1j} - \frac{vm_+}{\Delta x}u_{i-1j} + \frac{wp_+ - wm_- + wp_-}{\Delta y}u_{ij} + wp_-q - \frac{wm_+}{\Delta y}u_{ij-1}$$
$$+\lambda[\frac{2u_{ij} - u_{i+1j} - u_{i-1j}}{\Delta x^2} + \frac{u_{ij} - u_{ij-1} - q\Delta y}{\Delta y^2}] = f_{ij} \ .$$

An den Eckpunkten erhält man für $i = 1, j = 1$ die Gleichung

$$\frac{vp_+ - vm_- + vm_+}{\Delta x}u_{ij} + \frac{vp_-}{\Delta x}u_{i+1j} - 2\frac{vm_+}{\Delta x}u_r + \frac{wp_+ - wm_- - wm_+}{\Delta y}u_{ij} + \frac{wp_-}{\Delta y}u_{ij+1} - wm_+q$$
$$+\lambda[\frac{3u_{ij} - u_{i+1j} - 2u_r}{\Delta x^2} + \frac{u_{ij} - u_{ij+1} - q\Delta y}{\Delta y^2}] = f_{ij} \ ,$$

für $i = 1, j = M$ die Gleichung

$$\frac{vp_+ - vm_- + vm_+}{\Delta x}u_{ij} + \frac{vp_-}{\Delta x}u_{i+1j} - 2\frac{vm_+}{\Delta x}u_r + \frac{wp_+ - wm_- + wp_-}{\Delta y}u_{ij} + wp_-q - \frac{wm_+}{\Delta y}u_{ij-1}$$
$$+\lambda[\frac{3u_{ij} - u_{i+1j} - 2u_r}{\Delta x^2} + \frac{u_{ij} - u_{ij-1} - q\Delta y}{\Delta y^2}] = f_{ij} \ ,$$

für $i = N, j = 1$ die Gleichung

$$\frac{vp_+ - vm_- + vp_-}{\Delta x}u_{ij} + vp_-q - \frac{vm_+}{\Delta x}u_{i-1j} + \frac{wp_+ - wm_- - wm_+}{\Delta y}u_{ij} + \frac{wp_-}{\Delta y}u_{ij+1} - wm_+q$$
$$+\lambda[\frac{u_{ij} - u_{i-1j} - q\Delta x}{\Delta x^2} + \frac{u_{ij} - u_{ij+1} - q\Delta y}{\Delta y^2}] = f_{ij}$$

und schließlich für $i = N, j = M$ die Gleichung

$$\frac{vp_+ - vm_- + vp_-}{\Delta x}u_{ij} + vp_-q - \frac{vm_+}{\Delta x}u_{i-1j} + \frac{wp_+ - wm_- + wp_-}{\Delta y}u_{ij} + wp_-q - \frac{wm_+}{\Delta y}u_{ij-1}$$
$$+\lambda[\frac{u_{ij} - u_{i-1j} - q\Delta x}{\Delta x^2} + \frac{u_{ij} - u_{ij-1} - q\Delta y}{\Delta y^2}] = f_{ij} \ .$$

Aufgrund der Eigenschaft

$$vp_+ \geq 0,\ vm_+ \geq 0,\ vp_- \leq 0,\ vm_- \leq 0,\ wp_+ \geq 0,\ wm_+ \geq 0,\ wp_- \leq 0,\ wm_- \leq 0$$

ergibt sich mit der gewählten Upwind-Approximation ein Gleichungssystem mit einer stabilisierten Diagonale zur Bestimmung der gesuchten Werte u_{ij}, $i = 1, \ldots, N, j = 1, \ldots, M$. Im nachfolgenden Programm ist die Upwind-Approximation implementiert.

```
# Programm zur Loesung der Aufgabe 9.2
# mit Dirichlet- und Neumann-RB, iterative Loesung
# input: n,m Stuetzpunkte in x- und y-Richtung,
#        omega Relaxationsparameter, epsilon Genauigkeit
# output: Loesung u(x,y) auf dem Gitter x,y,
#         udiff Abstand zweier Loesungen, it Iterationszahl
# Aufruf: [u,x,y,udiff,it] = aufg92i(n,m,omega,epsilon,itmax)
function [u,x,y,udiff,it] = aufg92i(n,m,omega,epsilon,itmax);
# Materialdaten, Quellen
la = 3.0; ff=0;
# Gebeitslaenge, Gebietshoehe
ll = 2; hh = 1;
# Dimensionierung des Gitters
dx = ll/n; dy = hh/m;
h = dx; k = dy;
# Gitterinitialisierung
x(1) = dx/2; y(1) = dy/2;
for i=1:n
  x(i) = (i-1)*dx+x(1);
endfor
for j=1:m
  y(j) = (j-1)*dy+y(1);
endfor
# Startfeld
for i=1:n
  for j=1:m
    u(i,j) = 1.0;
  endfor
endfor
# Konstanten
h2=h^2; k2=k^2; dx2=2*dx; dy2=2*dy;
udiff = 1; unorm = 1; it = 0;
# Randwerte
uw=1; q1=0; q2=0; q3=0;
# Aufbau der Matrix und der rechten Seite
for i=1:n
    for j=1:m
      ind = i + (j-1)*n;
      vm = (x(i)-0.5*dx)^2; vp = (x(i)+0.5*dx)^2;
      wm = (1-y(j)+0.5*dy)^2; wp = (1-y(j)-0.5*dy)^2;
      avm = abs(vm); avp = abs(vp);
      awm = abs(wm); awp = abs(wp);
      vpp = 0.5*(vp+avp); vpm = 0.5*(vp-avp);
      vmp = 0.5*(vm+avm); vmm = 0.5*(vm-avm);
      wpp = 0.5*(wp+awp); wpm = 0.5*(wp-awp);
      wmp = 0.5*(wm+awm); wmm = 0.5*(wm-awm);
      if (1 < i && i < n && 1 < j && j < m )
        a(ind,ind) = 2*la/h2+2*la/k2 + (vpp-vmm)/dx + (wpp-wmm)/dy;
        a(ind,ind+1) = -(la/h2 - vpm/dx);
        a(ind,ind-1) = -(la/h2 + vmp/dx);
        a(ind,ind+n) = -(la/k2 - wpm/dy);
        a(ind,ind-n) = -(la/k2 + wmp/dy);
        ff1(ind) = ff;
      endif
      if (i == 1 && 1 < j && j < m )
        a(ind,ind) = 3*la/h2+2*la/k2 + (vpp-vmm+vmp)/dx + (wpp-wmm)/dy;
        a(ind,ind+1) = -(la/h2 - vpm/dx);
        a(ind,ind+n) = -(la/k2 - wpm/dy);
        a(ind,ind-n) = -(la/k2 + wmp/dy);
        ff1(ind) = ff-2*vpm*uw/h+la*2*uw/h2;
      endif
      if (1 < i && i < n && j == 1 )
        a(ind,ind) = 2*la/h2+la/k2 + (vpp-vmm)/dx + (wpp - wmm - wmp)/dy;
        a(ind,ind+1) = -(la/h2 - vpm/dx);
        a(ind,ind-1) = -(la/h2 + vmp/dx);
```

```
        a(ind,ind+n) = -(la/k2 - wpm/dy);
        ff1(ind) = ff+wmp*q1+la*q1/k;
      endif
      if (i == n && 1 < j && j < m )
        a(ind,ind) = la/h2+2*la/k2 + (vpp - vmm + vpm)/dx + (wpp-wmm)/dy;
        a(ind,ind-1) = -(la/h2 + vmp/dx);
        a(ind,ind+n) = -(la/k2 - wpm/dy);
        a(ind,ind-n) = -(la/k2 + wmp/dy);
        ff1(ind) = ff-vpm*q2+la*q2/h;
      endif
      if (1 < i && i < n && j == m )
        a(ind,ind) = 2*la/h2+la/k2 + (vpp-vmm)/dx + (wpp - wmm + wpm)/dy;
        a(ind,ind+1) = -(la/h2 - vpm/dx);
        a(ind,ind-1) = -(la/h2 + vmp/dx);
        a(ind,ind-n) = -(la/k2 + wmp/dy);
        ff1(ind) = ff-wpm*q3+la*q3/k;
      endif
      if (i == 1 && j == 1 )
        a(ind,ind) = 3*la/h2+la/k2 + (vpp - vmm + vmp)/dx +(wpp - wmm- wmp)/dy;
        a(ind,ind+1) = -(la/h2 - vpm/dx);
        a(ind,ind+n) = -(la/k2 - wpm/dy);
        ff1(ind) = ff+2*vmp*uw/h+wmp*q1+la*2*uw/h2+la*q1/k;
      endif
      if (i == 1 && j == m )
        a(ind,ind) = 3*la/h2+la/k2 + (vpp - vmm + vmp)/dx + (wpp - wmm + wpm)/dy;
        a(ind,ind+1) = -(la/h2 - vpm/dx);
        a(ind,ind-n) = -(la/k2 + wmp/dy);
        ff1(ind) = ff + 2*vmp*uw/h - wpm*q3 + la*2*uw/h2 +la*q3/k;
      endif
      if (i == n && j == 1 )
        a(ind,ind) = la/h2+la/k2 + (vpp - vmm + vpm)/dx + (wpp - wmm - wmp)/dy;
        a(ind,ind-1) = -(la/h2 + vmp/dx);
        a(ind,ind+n) = -(la/k2 - wpm/dy);
        ff1(ind) = ff-vpm*q2+wmp*q1+la*q2/h+la*q1/k;
      endif
      if (i == n && j == m )
        a(ind,ind) = la/h2+la/k2 + (vpp - vmm + vpm)/dx + (wpp - wmm + wpm)/dy;
        a(ind,ind-1) = -(la/h2 + vmp/dx);
        a(ind,ind-n) = -(la/k2 + wmp/dy);
        ff1(ind) = ff-vmp*q2-wmp*q3+la*q2/h+la*q3/k;
      endif
    endfor
endfor
# sor-Iteration
while (sqrt(udiff/unorm) > epsilon && it < itmax)
  ualt = u;
  udiff
  it
  for i=1:n
    for j=1:m
      ind = i + (j-1)*n;
      if (1 < i && i < n && 1 < j && j < m )
        u(i,j)=(ff1(ind)-a(ind,ind+1)*u(i+1,j)-a(ind,ind-1)*u(i-1,j) ...
                -a(ind,ind+n)*u(i,j+1)-a(ind,ind-n)*u(i,j-1))/a(ind,ind);
      endif
      if (i == 1 && 1 < j && j < m )
        u(i,j)=(ff1(ind)-a(ind,ind+1)*u(i+1,j)-a(ind,ind+n)*u(i,j+1) ...
                -a(ind,ind-n)*u(i,j-1))/a(ind,ind);
      endif
      if (1 < i && i < n && j == 1 )
        u(i,j)=(ff1(ind)-a(ind,ind+1)*u(i+1,j)-a(ind,ind-1)*u(i-1,j) ...
                -a(ind,ind+n)*u(i,j+1))/a(ind,ind);
      endif
      if (i == n && 1 < j && j < m )
        u(i,j)=(ff1(ind)-a(ind,ind-1)*u(i-1,j)-a(ind,ind+n)*u(i,j+1) ...
                -a(ind,ind-n)*u(i,j-1))/a(ind,ind);
      endif
      if (1 < i && i < n && j == m )
        u(i,j)=(ff1(ind)-a(ind,ind+1)*u(i+1,j)-a(ind,ind-1)*u(i-1,j) ...
                -a(ind,ind-n)*u(i,j-1))/a(ind,ind);
      endif
      if (i == 1 && j == 1 )
        u(i,j)=(ff1(ind)-a(ind,ind+1)*u(i+1,j)-a(ind,ind+n)*u(i,j+1))/a(ind,ind);
      endif
      if (i == 1 && j == m )
```

```
        u(i,j)=(ff1(ind)-a(ind,ind+1)*u(i+1,j)-a(ind,ind-n)*u(i,j-1))/a(ind,ind);
      endif
      if (i == n && j == 1 )
        u(i,j)=(ff1(ind)-a(ind,ind-1)*u(i-1,j)-a(ind,ind+n)*u(i,j+1))/a(ind,ind);
      endif
      if (i == n && j == m )
        u(i,j)=(ff1(ind)-a(ind,ind-1)*u(i-1,j)-a(ind,ind-n)*u(i,j-1))/a(ind,ind);
      endif
    endfor
  endfor
  udiff = 0.0; unorm = 0.0;
  for i=1:n
    for j=1:m
      u(i,j) = omega*u(i,j)+(1-omega)*ualt(i,j);
      udiff = udiff + (u(i,j)-ualt(i,j))^2;
      unorm = unorm + ualt(i,j)^2;
    endfor
  endfor
  if (unorm == 0.0)
    unorm = 1;
  endif
  it = it + 1;
endwhile
endfunction
```

3) Die zentrale Differenzenapproximation ergibt mit den bekannten Beziehungen

$$v_{i+1/2j} = v_{ij} + \frac{h}{2}v_x + O(h^2),\ v_{i-1/2j} = v_{ij} - \frac{h}{2}v_x + O(h^2),\ \frac{u_{i+1j} - 2u_{ij} + u_{i-1j}}{h^2} = u_{xx} + O(h^2)$$

für die erste Komponente uv von $u\vec{v}$

$$\begin{aligned}
&\frac{u_{i+1j} + u_{ij}}{2}v_{i+1/2j} - \frac{u_{ij} + u_{i-1j}}{2}v_{i-1/2j} \\
&= \frac{u_{i+1j}}{2h}[v_{ij} + \frac{h}{2}v_x + O(h^2)] + \frac{u_{ij}}{2h}[v_{ij} + \frac{h}{2}v_x + O(h^2)] \\
&- \frac{u_{ij}}{2h}[v_{ij} - \frac{h}{2}v_x + O(h^2) - \frac{u_{i-1j}}{2h}[v_{ij} - \frac{h}{2}v_x + O(h^2)] \\
&= \frac{u_{i+1j} - u_{i-1j}}{2h}v_{ij} + \frac{u_{i+1j} + u_{i-1j}}{4}v_x + \frac{u_{ij}}{2}v_x + O(h^2) \\
&= u_x v + O(h^2) + \frac{u_{i+1j} + u_{i-1j} + 2u_{ij}}{4}v_x + O(h^2) \\
&= u_x v + \frac{u_{i+1j} + u_{i-1j} - 2u_{ij}}{4h^2}h^2 v_x + \frac{4u_{ij}}{4}v_x + O(h^2) \\
&= u_x v + \frac{1}{4}(u_{xx} + O(h^2))h^2 v_x + uv_x + O(h^2) \\
&= u_x v + uv_x + O(h^2) = (uv)_x + O(h^2)\ .
\end{aligned}$$

Der Nachweis für die zweite Komponente uw von $u\vec{v}$ erfolgt analog.

4) Mit den Bezeichungen

$$\begin{aligned}
vp_+ &= \frac{v_{i+1/2j} + |v_{i+1/2j}|}{2},\ vp_- = \frac{v_{i+1/2j} - |v_{i+1/2j}|}{2}, \\
vm_+ &= \frac{v_{i-1/2j} + |v_{i-1/2j}|}{2},\ vm_- = \frac{v_{i-1/2j} - |v_{i-1/2j}|}{2}
\end{aligned}$$

gilt im Fall $v_{i+1/2j}v_{i-1/2j} \geq 0$ für die erste Komponente die Upwind-Approximation

$$\frac{v_{i+1/2j}u_{ij} - v_{i-1/2j}u_{i-1j}}{h}$$

und durch Taylorreihenentwicklung erhält man

$$\frac{v_{i+1/2j}u_{ij} - v_{i-1/2j}u_{i-1j}}{h} = (vu)_x + O(h) \; .$$

Falls $v_{i+1/2j}v_{i-1/2j} \leq 0$ ist, erhält man für die erste Komponente die Upwind-Approximation

$$\frac{v_{i+1/2j}u_{ij} - v_{i-1/2j}u_{ij}}{h}$$

und mit geeigneten Taylorentwicklungen folgt

$$\frac{v_{i+1/2j}u_{ij} - v_{i-1/2j}u_{ij}}{h} = uv_x + O(h^2) \; .$$

Aufgrund der unterschiedlichen Vorzeichen von $v_{i+1/2j}$ und $v_{i-1/2j}$ im nichttrivialen Fall ergibt sich

$$\frac{v_{i+1/2j} + v_{i-1/2j}}{2} = O(h)$$

und damit auch $v_{ij} = O(h)$ bzw. $u_x v = O(h)$. Insgsamt gilt dann

$$\begin{aligned} \frac{v_{i+1/2j}u_{ij} - v_{i-1/2j}u_{ij}}{h} &= uv_x + O(h^2) = uv_x + u_x v - O(h) + O(h^2) \\ &= uv_x + u_x v + O(h) = (uv)_x + O(h) \; . \end{aligned}$$

Der Nachweis für die zweite Komponente uw von $u\vec{v}$ erfolgt analog.

5) Die Diskretisierung erfolgt in radialer, azimuthaler und z-Richtung, der Einfachheit halber mit den Diskretisierungsinkrementen $\Delta\rho$, $\Delta\varphi$ und Δz äquidistant. Für eine Volumenzelle ω_{ijk} gilt dann

$$\begin{aligned} \omega_{ijk} &= \{(\rho\cos\varphi, \rho\sin\varphi, z) \,|\, 1 + (i-1/2)\Delta\rho \leq \rho \leq 1 + (i+1/2)\Delta\rho, \\ &\qquad (j-1/2)\Delta\varphi \leq \varphi \leq (j+1/2)\Delta\varphi,\ (k-1/2)\Delta z \leq z \leq (k+1/2)\Delta z\} \; , \end{aligned}$$

für $i = 1, \ldots, N$, $j = 1, \ldots, M$, $k = 1, \ldots, P$, wobei die Punkte

$$(x_i, y_j, z_k) = (\rho_i \cos\varphi_j, (\rho_i \sin\varphi_j, z_k), \quad \rho_i = 1 + i\Delta\rho,\ \varphi_j = j\Delta\varphi,\ z_k = k\Delta z$$

jeweils die Mittelpunkte der Zellen ω_{ijk} sind, wo auch die Stützwerte u_{ijk} berechnet werden sollen.
Die Bilanz der rechten Seite der Differentialgleichung ergibt für konstantes λ

$$\begin{aligned} \int_{\omega_{ijk}} \lambda\Delta u \, dV &= \lambda \int_{z_{k-1/2}}^{z_{k+1/2}} \int_{\varphi_{j-1/2}}^{\varphi_{j+1/2}} \int_{\rho_{i-1/2}}^{\rho_{i+1/2}} [\frac{1}{\rho}[\frac{\partial}{\partial\rho}(\rho\frac{\partial u}{\partial\rho})] + \frac{1}{\rho^2}\frac{\partial^2 u}{\partial\varphi^2} + \frac{\partial^2 u}{\partial z^2}] \, \rho d\rho d\varphi dz \\ &= \lambda \int_{z_{k-1/2}}^{z_{k+1/2}} \int_{\varphi_{j-1/2}}^{\varphi_{j+1/2}} \int_{\rho_{i-1/2}}^{\rho_{i+1/2}} [\frac{\partial}{\partial\rho}(\rho\frac{\partial u}{\partial\rho}) + \frac{1}{\rho}\frac{\partial^2 u}{\partial\varphi^2} + \rho\frac{\partial^2 u}{\partial z^2}] \, d\rho d\varphi dz \; . \end{aligned}$$

Die iterierte Integration ergibt mit den östlichen und westlichen, nördlichen und südlichen

sowie hinteren und vorderen Randflächen $\partial\omega_o, \partial\omega_w, \partial\omega_n, \partial\omega_s, \partial\omega_h, \partial\omega_v$ von ω_{ijk}

$$\begin{aligned}\int_{\omega_{ijk}} \Delta u\, dV &\approx [\rho_{i+1/2}\frac{\partial u}{\partial \rho}|_{\partial\omega_o} - \rho_{i-1/2}\frac{\partial u}{\partial \rho}|_{\partial\omega_w}]\Delta\varphi\Delta z \\ &\quad + \frac{1}{\rho_i}[\frac{\partial u}{\partial \varphi}|_{\partial\omega_h} - \frac{\partial u}{\partial \varphi}|_{\partial\omega_v}]\Delta\rho\Delta z + \rho_i[\frac{\partial u}{\partial z}|_{\partial\omega_n} - \frac{\partial u}{\partial z}|_{\partial\omega_s}]\Delta\rho\Delta\varphi \\ &\approx [\rho_{i+1/2}\frac{u_{i+1jk}-u_{ijk}}{\Delta\rho} - \rho_{i-1/2}\frac{u_{ijk}-u_{i-1jk}}{\Delta\rho}]\Delta\varphi\Delta z \\ &\quad + \frac{1}{\rho_i}[\frac{u_{ij+1k}-u_{ijk}}{\Delta\varphi} - \frac{u_{ijk}-u_{ij-1k}}{\Delta\varphi}]\Delta\rho\Delta z \\ &\quad + \rho_i[\frac{u_{ijk+1}-u_{ijk}}{\Delta z} - \frac{u_{ijk}-u_{ijk-1}}{\Delta z}]\Delta\rho\Delta\varphi\ .\end{aligned}$$

Die Division durch $\rho_i\Delta\rho\Delta\varphi\Delta z$ und die Berücksichtung von $\frac{\partial u}{\partial \varphi} = 0$ ergibt die FV-Approximation

$$\frac{1}{\rho_i}[\rho_{i+1/2}\frac{u_{i+1jk}-u_{ijk}}{\Delta\rho} - \rho_{i-1/2}\frac{u_{ijk}-u_{i-1jk}}{\Delta\rho}]/\Delta\rho + [\frac{u_{ijk+1}-u_{ijk}}{\Delta z} - \frac{u_{ijk}-u_{ijk-1}}{\Delta z}]/\Delta z$$

für alle $j = 1, \ldots, M$. Für die linke Seite der Differentialgleichung bilanziert man

$$\int_{\omega_{ijk}} \frac{\partial u}{\partial t}\, dV \approx \frac{\partial u_{ijk}}{\partial t}\rho_i\Delta\rho\Delta\varphi\Delta z$$

und die Division durch $\rho_i\Delta\rho\Delta\varphi\Delta z$ ergibt die Approximation $\frac{\partial u_{ijk}}{\partial t}$. Da die Approximationen unabhängig von j sind, kann man auf den Index j verzichten und erhält für die Differentialgleichung letztendlich die FV-Approximation

$$\frac{\partial u_{ik}}{\partial t} = \lambda[\frac{\rho_{i+1/2}(u_{i+1k}-u_{ik}) - \rho_{i-1/2}(u_{ik}-u_{i-1k})}{\rho_i\Delta\rho^2} + \frac{u_{ik+1}-2u_{ik}+u_{ik-1}}{\Delta z^2}]\ .$$

Mit der Berücksichtigung der Anfangs- und Randbedingungen erhält man ein Anfangswertproblem zur Berechnung der Stützwerte u_{ik} im Zeitintervall]0,10].

6) Mit dem Hilbertraum

$$U = V = H_0^1([0,1]) = \{u \,|\, u, u_x \in L_2([0,1]),\, u(0) = 0\}$$

definiert man auf $U \times V$ die Bilinearform

$$A(u,v) = \int_0^1 [(1+x^2)u_x v_x + 3uv]\, dx + 4u(1)v(1)$$

und auf V das Funktional

$$F(v) = \int_0^1 e^{-x^2} v\, dx + 8v(1)\ ,$$

was nachfolgend gerechtfertigt werden soll. Testet man die linke Seite der Gleichung mit

einer Funktion $v \in U$, dann ergibt sich durch partielle Integration

$$\begin{aligned}\int_0^1 [-\frac{d}{dx}((1+x^2)\frac{du}{dx}) + 3u]v\,dx &= -(1+x^2)\frac{du}{dx}v|_0^1 + \int_0^1 (1+x^2)\frac{du}{dx}\frac{dv}{dx}\,dx + \int_0^1 3uv\,dx \\ &= \int_0^1 [(1+x^2)u_x v_x + 3uv]\,dx - 2\frac{du}{dx}v|_{x=1} \\ &= \int_0^1 [(1+x^2)u_x v_x + 3uv]\,dx - 2(4-2u(1))v(1) \\ &= \int_0^1 [(1+x^2)u_x v_x + 3uv]\,dx + 4u(1)v(1) - 8v(1)\ .\end{aligned}$$

Der Test der rechten Seite der Gleichung mit einer Funktion $v \in U$ ergibt

$$\int_0^1 e^{-x^2} v\,dx\ ,$$

so dass man insgesamt

$$\int_0^1 [(1+x^2)u_x v_x + 3uv]\,dx + 4u(1)v(1) - 8v(1) = \int_0^1 e^{-x^2} v\,dx \iff A(u,v) = F(v)$$

und damit die schwache Formulierung erhält.

7) Zu zeigen ist die Existenz von Konstanten $C, D, \gamma > 0$, so dass

a) $|A(u,v)| \le C||u||_V||v||_V$, $|F(v)| \le D||v||_V$ (Stetigkeit) und

b) $A(u,u) \ge \gamma||u||_V^2$ (Koerzitivität)

mit

$$||u||_V = ||u||_{H^1} = [\int_0^1 (u_x^2 + u^2)\,dx]^{1/2}$$

gilt. Für $x \in$ [0,1] gilt aufgrund der Cauchy-Schwarz'schen Ungleichung

$$|A(u,v)| \le 2||u_x||_{L_2}||v_x||_{L_2} + 3||u||_{L_2}||v||_{L_2} + 4||u||_V||v||_V \le 9||u||_V||v||_V\ ,$$

da

$$||u_x||_{L_2} = [\int_0^1 u_x^2\,dx]^{1/2} \le [\int_0^1 (u_x^2 + u^2)\,dx]^{1/2} = ||u||_V$$

und

$$||u||_{L_2} = [\int_0^1 u^2\,dx]^{1/2} \le [\int_0^1 (u_x^2 + u^2)\,dx]^{1/2} = ||u||_V$$

gilt. Außerdem gilt

$$\begin{aligned}|F(v)| &= |\int_0^1 e^{-x^2} v\,dx + 8v(1)| \le e^0 \int_0^1 |v|\,dx + 8|v(1)| \\ &\le 9\int_0^1 |v|\,dx \le 9[\int_0^1 v^2\,dx]^{1/2} \le 9[\int_0^1 v_x^2 + v^2\,dx]^{1/2} = 9||v||_V\ .\end{aligned}$$

Damit ist a) mit $C = D = 9$ bewiesen.
Da $(1 + x^2) \geq 1$ für alle $x \in [0,1]$ ist, folgt

$$A(u,u) \geq \int_0^1 u_x^2 \, dx + 3 \int_0^1 u^2 \, dx \geq \int_0^1 u_x^2 \, dx + \int_0^1 u^2 \, dx = ||u||_V^2 \ ,$$

also b) mit $\gamma = 1$.

8) Mit den quadratischen FE bzw. Basis-Funktionen

$$\varphi_i(x) = \begin{cases} 2(\frac{x-x_i}{h})^2 - 3\frac{x-x_i}{h} + 1 & x_i < x < x_{i+1} \\ 2(\frac{x-x_{i-1}}{h})^2 - \frac{x-x_{i-1}}{h} & x_{i-1} \leq x \leq x_i \\ 0 & \text{sonst} \end{cases}$$

für $i = 1, \ldots, n$, $x_i = i \cdot h$, $h = 1/n$, und

$$\varphi_{i+1/2}(x) = \begin{cases} 4\frac{x-x_i}{h}(1 - \frac{x-x_i}{h}) & x_i < x < x_{i+1} \\ 0 & \text{sonst} \end{cases}$$

für $i = 1, \ldots, n-1$ sind zur Berechnung der Steifigkeitsmatrix die Integrale

$$\int_{supp\,\varphi_i} \varphi_i^2(x) \, dx, \ \int_{supp\,\varphi_i} \varphi_i(x)\varphi_{i+1}(x) \, dx, \ \int_{supp\,\varphi_i} \varphi_i(x)\varphi_{i-1}(x) \, dx,$$
$$\int_{supp\,\varphi_{i+1/2}} \varphi_{i+1/2}^2(x) \, dx, \ \int_{supp\,\varphi_{i+1/2}} \varphi_{i+1/2}(x)\varphi_i(x) \, dx, \ \int_{supp\,\varphi_{i+1/2}} \varphi_{i+1/2}(x)\varphi_{i+1}(x) \, dx,$$

bzw.

$$\int_{supp\,\varphi_i} \varphi_i'^2(x) \, dx, \ \int_{supp\,\varphi_i} \varphi_i'(x)\varphi_{i+1}'(x) \, dx, \ \int_{supp\,\varphi_i} \varphi_i'(x)\varphi_{i-1}'(x) \, dx,$$
$$\int_{supp\,\varphi_{i+1/2}} \varphi_{i+1/2}'^2(x) \, dx, \ \int_{supp\,\varphi_{i+1/2}} \varphi_{i+1/2}'(x)\varphi_i'(x) \, dx, \ \int_{supp\,\varphi_{i+1/2}} \varphi_{i+1/2}'(x)\varphi_{i+1}'(x) \, dx,$$

und

$$\int_{supp\,\varphi_i} x^2\varphi_i'^2(x) \, dx, \ \int_{supp\,\varphi_i} x^2\varphi_i'(x)\varphi_{i+1}'(x) \, dx,$$
$$\int_{supp\,\varphi_i} x^2\varphi_i'(x)\varphi_{i-1}'(x) \, dx, \ \int_{supp\,\varphi_{i+1/2}} x^2\varphi_{i+1/2}'^2(x) \, dx,$$
$$\int_{supp\,\varphi_{i+1/2}} x^2\varphi_{i+1/2}'(x)\varphi_i'(x) \, dx, \ \int_{supp\,\varphi_{i+1/2}} x^2\varphi_{i+1/2}'(x)\varphi_{i+1}'(x) \, dx,$$

zu bestimmen. Man erhält mit den Ableitungen

$$\varphi_i'(x) = \begin{cases} (4\frac{x-x_i}{h} - 3)/h & x_i < x < x_{i+1} \\ (4\frac{x-x_{i-1}}{h} - 1)/h & x_{i-1} \leq x \leq x_i \\ 0 & \text{sonst} \end{cases}$$

und

$$\varphi_{i+1/2}'(x) = \begin{cases} 4(1 - 2\frac{x-x_i}{h})/h & x_i < x < x_{i+1} \\ 0 & \text{sonst} \end{cases}$$

im Ergebnis der Integration

$$\int_{supp\,\varphi_i} \varphi_i^2(x)\,dx = \frac{4h}{15},$$
$$\int_{supp\,\varphi_i} \varphi_i(x)\varphi_{i+1}(x)\,dx = \int_{supp\,\varphi_i} \varphi_i(x)\varphi_{i-1}(x)\,dx = -\frac{h}{30},$$
$$\int_{supp\,\varphi_{i+1/2}} \varphi_{i+1/2}^2(x)\,dx = \frac{8h}{15},$$
$$\int_{supp\,\varphi_{i+1/2}} \varphi_{i+1/2}(x)\varphi_i(x)\,dx = \int_{supp\,\varphi_{i+1/2}} \varphi_{i+1/2}(x)\varphi_{i+1}(x)\,dx = \frac{h}{15},$$

bzw.

$$\int_{supp\,\varphi_i} \varphi_i'^2(x)\,dx = \frac{14}{3h},$$
$$\int_{supp\,\varphi_i} \varphi_i'(x)\varphi_{i+1}'(x)\,dx = \int_{supp\,\varphi_i} \varphi_i'(x)\varphi_{i-1}'(x)\,dx = \frac{1}{3h},$$
$$\int_{supp\,\varphi_{i+1/2}} \varphi_{i+1/2}'^2(x)\,dx = \frac{16}{3h},$$
$$\int_{supp\,\varphi_{i+1/2}} \varphi_{i+1/2}'(x)\varphi_i'(x)\,dx = \int_{supp\,\varphi_{i+1/2}} \varphi_{i+1/2}'(x)\varphi_{i+1}'(x)\,dx = -\frac{8}{3h}\ .$$

Am rechten Intervallende ist statt über $supp\,\varphi_n$ über

$$supp\,\varphi_n \cap [0{,}1]$$

zu integrieren, so dass sich dort nur das $0{,}5$-fache der für $i < n$ angegebenen Werte ergibt. Für die weiteren Integrale ergibt sich

$$\begin{aligned}
\int_{supp\,\varphi_i} x^2\varphi_i'^2(x)\,dx &= \frac{14}{3h}[\frac{x_{i-1}^2 + x_i^2}{2} + \frac{h}{14}(x_{i-1} + 3x_i) + \frac{13h^2}{35}],\ 1 \le i < n,\\
\int_{supp\,\varphi_n \cap [0,1]} x^2\varphi_n'^2(x)\,dx &= \frac{7}{3h}[x_{n-1}^2 + \frac{11h}{7}x_{n-1} + \frac{23h^2}{35}],\\
\int_{supp\,\varphi_i} x^2\varphi_i'(x)\varphi_{i+1}'(x)\,dx &= \frac{1}{3h}[x_i^2 + hx_i + \frac{3h^2}{5}],\\
\int_{supp\,\varphi_i} x^2\varphi_i'(x)\varphi_{i-1}'(x)\,dx &= \frac{1}{3h}[x_{i-1}^2 + hx_{i-1} + \frac{3h^2}{5}],\\
\int_{supp\,\varphi_{i+1/2}} x^2\varphi_{i+1/2}'^2(x)\,dx &= \frac{16}{3h}[x_i^2 + hx_i + \frac{2h^2}{5}],\\
\int_{supp\,\varphi_{i+1/2}} x^2\varphi_{i+1/2}'(x)\varphi_i'(x)\,dx &= -\frac{8}{3h}[x_i^2 + \frac{h}{2}x_i + \frac{3h^2}{20}],\\
\int_{supp\,\varphi_{i+1/2}} x^2\varphi_{i+1/2}'(x)\varphi_{i+1}'(x)\,dx &= -\frac{8}{3h}[x_i^2 + \frac{3h}{2}x_i + \frac{13h^2}{20}]\ .
\end{aligned}$$

Zur Berechnung des Lastvektors benötigt man noch die Integrale

$$\int_{supp\,\varphi_i} e^{-x^2}\varphi_i(x)\,dx \approx \frac{h}{3}[e^{-x_{i-1/2}^2} + e^{-x_{i-1/2}^2}]/2\ ,\quad i = 1,\ldots,n-1$$

$$\int_{supp\,\varphi_n \cap [0,1]} e^{-x^2} \varphi_i(x)\,dx = \frac{h}{6} e^{-x_{i-1/2}^2}\ , \quad i = n$$

und

$$\int_{supp\,\varphi_{i+1/2}} e^{-x^2} \varphi_{i+1/2}(x)\,dx \approx \frac{2h}{3} e^{-x_{i+1/2}^2}\ , \quad i = 0, \ldots, n-1\ .$$

Ausgehend von der schwachen Formulierung erhält man mit dem Ansatz

$$u_h(x) = \sum_{j=1}^{2n} u_{j/2} \varphi_{j/2}(x)$$

das lineare Gleichungssystem

$$A(u_h, \varphi_{i/2}) = F(\varphi_{i/2})\ , \quad i = 1, \ldots, 2n$$

zur Bestimmung der Koeffizienten

$$u_{1/2},\ u_1,\ u_{3/2},\ u_2, \ldots, u_{n-1/2},\ u_n\ .$$

Die Integrale zur Berechnung des Lastvektors können bei Bedarf durch eine Quadraturformel genauer als hier bestimmt werden. Im nachfolgenden Programm ist die FE-Methode implementiert.

```
# Programm zur Loesung der Aufgabe 9.8, FEM, quadratische Elemente
# -((1+c*x^2)u_x)_x + d*u = exp(-x^2), u(0)=0, [u_x + alpha*u](1) = peta
# Input:  c,d,alpha,peta und Zahl der ganzen Indizes n
# Output: Loesung u an den Stuetzstellen x, Matrix a und rechte Seite f
# Aufruf: [u,x,a,f] = aufg98fem2(c,d,alpha,peta,n);
function [u,x,a,f] = aufg98fem2(c,d,alpha,peta,n);
m = 2*n; h = 1/n; h2 = h/2; h3 = 1/(h*3); h15 = h/15;
h163 = 16/(h*3); h83 = 8/(h*3); h143 = 14/(h*3);
x(1) = 0;
f = zeros(m,1);
a = zeros(2*n,2*n);
for i=1:m
   x(i+1)=i*h2;
endfor
# Aufbau der Steifigkeitsmatrix a und des Lastvektors f
for i=1:n
   jh = 2*i-1;
   j = 2*i;
   if (i == 1)
     a(jh,jh) = h163*(1+c*(x(jh)^2+h*x(jh)+2*h^2/5))+d*8*h15;
     a(jh,jh+1) = -h83*(1+c*(x(jh)^2+3*h*x(jh)/2+13*h^2/20))+d*h15;
     a(j,j-1) = -h83*(1+c*(x(j-1)^2+3*h*x(j-1)/2+13*h^2/20))+d*h15;
     a(j,j) = h143*(1+c*((x(j-1)^2+x(j+1)^2)/2+11*h*x(j-1)/14 ...
                        +3*h*x(j+1)/14+13*h^2/35))+d*4*h15;
     a(j,j+1) = -h83*(1+c*(x(j+1)^2+h*x(j+1)/2+3*h^2/20))+d*h15;
     a(j,j+2) = h3*(1+c*(x(j+1)^2+h*x(j+1)+3*h^2/5))-d*h15/2;
   endif
   if (2 < j && j < m )
     a(j,j-2) = h3*(1+c*(x(j-1)^2+h*x(j-1)+3*h^2/5))-d*h15/2;
     a(j,j-1) = -h83*(1+c*(x(j-1)^2+3*h*x(j-1)/2+13*h^2/20))+d*h15;
     a(j,j) = h143*(1+c*((x(j-1)^2+x(j+1)^2)/2+11*h*x(j-1)/14 ...
                        +3*h*x(j+1)/14+13*h^2/35))+d*4*h15;
     a(j,j+1) = -h83*(1+c*(x(j+1)^2+h*x(j+1)/2+3*h^2/20))+d*h15;
     a(j,j+2) = h3*(1+c*(x(j+1)^2+h*x(j+1)+3*h^2/5))-d*h15/2;
   endif
   if (1 < jh )
     a(jh,jh-1) = -h83*(1+c*(x(jh)^2+h*x(jh)/2+3*h^2/20))+d*h15;
     a(jh,jh) = h163*(1+c*(x(jh)^2+h*x(jh)+2*h^2/5))+d*8*h15;
     a(jh,jh+1) = -h83*(1+c*(x(jh)^2+3*h*x(jh)/2+13*h^2/20))+d*h15;
   endif
   if (j == m)
```

```
      a(j,j-2) = h3*(1+c*(x(j-1)^2+h*x(j-1)+3*h^2/5))-d*h15/2;
      a(j,j-1) = -h83*(1+c*(x(j-1)^2+3*h*x(j-1)/2+13*h^2/20))+d*h15;
      a(j,j) = 7*h3*(1+c*(x(j-1)^2+11*h*x(j-1)/7+23*h^2/35))+d*2*h15+2*alpha;
    endif
    f(jh) = exp(-x(j)^2)*2*h/3;
    f(j) = (exp(-x(j)^2)+exp(-x(j+1)^2))*h/6;
    if (j == m)
       f(j) = exp(-x(j)^2)*h/6 + 2*peta;
    endif
endfor
# Loesung
uu = a\f;
u(1) = 0;
for i=1:m
   u(i+1) = uu(i);
endfor
endfunction
```

Zum Vergleich mit der FE-Methode mit quadratischen Elementen sind in den nachfolgenden Programmen ein FE-Verfahren mit linearen Elementen und ein FV-Verfahren zur Lösung des Randwertproblems implementiert.

```
# Programm zur Loesung der Aufgabe 9.8 mit einem FE-Verfahren, lineare Elemente
# -((1+c*x^2)u_x)_x + d*u = exp(-x^2), u(0)=0, [u_x + alpha*u](1) = peta
# Input:  c,d,alpha,peta und Zahl der Elemente n
# Output: Loesung u an den Stuetzstellen x, Matrix a und rechte Seite f
# Aufruf: [u,x,a,f] = aufg98fem1(c,d,alpha,peta,n);
function [u,x,a,f] = aufg98fem1(c,d,alpha,peta,n);
h = 1/n;
f = zeros(n,1);
x(1)=0;
for i=1:n
   x(i+1)=i*h;
endfor
# Aufbau der Steifigkeitsmatrix a und des Lastvektors f
for i=1:n
   if (i == 1)
     a(i,i) = (2+c*(x(i)^2+x(i+1)^2+h*(x(i)+x(i+1))+2*h^2/3))/h + d*2*h/3;
     a(i,i+1) = -(1+c*(x(i+1)^2+h*x(i+1)+h^2/3))/h +d*h/6;
     f(i) = exp(-(x(i)+h/2)^2)*h/3+exp(-(x(i+1)+h/2)^2)*h/3+exp(-x(i+1)^2)*h/3;
   endif
   if (1 < i && i < n)
     a(i,i-1) = -(1+c*(x(i)^2+h*x(i)+h^2/3))/h + d*h/6;
     a(i,i+1) = -(1+c*(x(i+1)^2+h*x(i+1)+h^2/3))/h +d*h/6;
     a(i,i) = (2+c*(x(i)^2+x(i+1)^2+h*(x(i)+x(i+1))+2*h^2/3))/h + d*2*h/3;
     f(i) = exp(-(x(i)+h/2)^2)*h/3+exp(-(x(i+1)+h/2)^2)*h/3+exp(-x(i+1)^2)*h/3;
   endif
   if (i == n)
     a(i,i-1) = -(1+c*(x(i)^2+h*x(i)+h^2/3))/h + d*h/6;
     a(i,i) = (1+c*(x(i)^2+h*x(i)+h^2/3))/h + d*1*h/3 +2*alpha;
     f(i) = exp(-(x(i)+h/2)^2)*h/3+exp(-x(i+1)^2)*h/6 +2*peta;
   endif
endfor
# Loesung
uu = a\f;
u(1)=0;
for i=1:n
   u(i+1) = uu(i);
endfor
endfunction
```

```
# Programm zur Loesung der Aufgabe 9.8 mit einem FV-Verfahren
# -((1+c*x^2)u_x)_x + d*u = exp(-x^2), u(0)=0, [u_x + alpha*u](1) = peta
# Stuetzwerte u_i an den Stuetzstellen x_{i-1/2}, i=1,...,n
# Input:  c,d,alpha,peta und Zahl der Stuetzstellen n
# Output: Loesung u an den Stuetzstellen x, Matrix a und rechte Seite f
# Aufruf: [u,x,a,f] = aufg98fvm(c,d,alpha,peta,n);
function [u,x,a,f] = aufg98fvm(c,d,alpha,peta,n);
h = 1/n;
f = zeros(n,1);
for i=1:n
   x(i)=(i-1/2)*h;
endfor
```

```
# Aufbau der Steifigkeitsmatrix a und des Lastvektors f
for i=1:n
   if (i == 1)
     a(i,i) = 2/h + (1+c*h^2)/h + d*h; a(i,i+1) = -(1+c*h^2)/h;
     f(i) = exp(-(h/2)^2)*h;
   endif
   if (1 < i && i < n)
     a(i,i-1) = -(1+c*((i-1)*h)^2)/h; a(i,i+1) = -(1+c*(i*h)^2)/h;
     a(i,i) = -(a(i,i-1)+a(i,i+1))+d*h;
     f(i) = exp(-((i-1/2)*h)^2)*h;
   endif
   if (i == n)
     c1 = 1/h + alpha/2;
     a(i,i-1) = -(1+c*((i-1)*h)^2)/h;
     a(i,i) = (1+c*((i-1)*h)^2)/h + alpha*(1+c*(i*h)^2/(h*c1)) + d*h;
     f(i) = exp(-((i-1/2)*h)^2)*h+ (1+c*(i*h)^2)*peta/(h*c1);
   endif
endfor
# Loesung
u = a\f;
endfunction
```

Beim FV-Verfahren wurde die Randbedingung $u = 0$ am linken Intervallende $x = 0$ durch

$$\frac{u_0 + u_1}{2} = 0$$

und die Randbedingung $\frac{du}{dx} + \alpha u = \beta$ am rechten Intervallende $x = 1$ durch

$$\frac{u_{n+1} - u_n}{h} + \alpha \frac{u_{n+1} + u_n}{2} = \beta$$

approximiert. Die Ghostwerte u_0 und u_{n+1} wurden mittels der Randbedingungen aus den Approximationen der Differentialgleichung an den Punkten $x_{1/2}$ und $x_{n-1/2}$ eliminiert.
In allen 3 Programmen wurde die Tridiagonalität der Steifigkeitsmatrizen nicht berücksichtigt. Zur Speicherplatzeinsparung und zur Effizienzsteigerung sei hier eine Modifizierung der Programme zwecks Nutzung der Routine ”tridia.m” zur Lösung der linearen Gleichungssysteme empfohlen.

Kapitel 10

1) Das folgende Programm ergibt die numerischen Lösungen im Vergleich zur exakten Lösung.

```
# Programm 10.5 em_milstein_allgemein_vs_exakt_gb.m ;
# Euler-Maruyama und Milstein-Methode zur numerischen Loesung von
#
# SDE: dX = (alpha + mu*X) dt + (peta + sigma*X) dW, X(0) = Xzero ,
# alpha = 2 , mu = -3 , peta = 1 , sigma = 1 und Xzero = 1 .
#
# im Vergleich zur exakten Loesung
#
# Wiener-Prozess ueber [0,1], dt = 2^(-8).
# Eulerâ^'Maruyama, Milstein auf einem Gitter mit Dt = R*dt (R Zweierpotenz)
randn('state',100) # random seeding
alpha = 2 ; mu = -3 ; peta = 1 ; sigma = 1 ; Xzero = 1 ; # Problemparameter
T = 1 ; N = 2^8 ; dt = 1/N;
dW = zeros(1,N); W = zeros(1,N); Xtrue = zeros(1,N); # Pre-Allokierung
dW = sqrt(dt)*randn(1,N) ; # Inkremente des Wiener-Prozesses
W = cumsum(dW) ; # Wiener-Prozess
# Aufbau der exakten Loesung
j = 0;
while (j < N)
   j = j + 1;
   ct1 = 0;
   k = 0;
   while (k < j)
     k = k + 1;
     summa = exp( (0.5*sigma^2-mu)*k*dt - sigma*W(k) );
     ct1 = ct1 + summa*dt;
     ct2(k) = exp( (0.5*sigma^2-mu)*k*dt - sigma*W(k) );
   endwhile
   wito = ito_integral_gb(ct2(1:j),dW(1:j),j);
   Xtrue(j) = ( Xzero+exp(sigma*W(1))*( (alpha-peta*sigma)*ct1+peta*wito ) )...
              *exp( (mu-0.5*sigma^2)*j*dt+sigma*(W(j)-W(1)) );
endwhile
plot([0:dt:T],[Xzero,Xtrue],'g-'), hold on
xlabel('t','FontSize',16); ylabel('X_t','FontSize',16);
R = 4 ; Dt = R*dt ; L = N/R; # L Schritte der numerischen Verfahren Dt = R*dt
Xem = zeros(1,L); Yem = zeros(1,L); # Pre-Allokierung
Xtemp = Xzero ;
Ytemp = Xzero ;
j = 0;
while (j < L)
   j = j + 1;
   sWinc = sqrt(Dt)*sign(sum(dW(R*(j-1)+1:R*j))) ;
   Xtemp = Xtemp + Dt*(alpha+mu*Xtemp)+(peta+sigma*Xtemp)*sWinc ;
   Xem(j) = Xtemp ;    #   schwache Euler-Maruyama-Loesung
   Ytemp = Ytemp+Dt*(alpha+mu*Ytemp)+(peta+sigma*Ytemp)*sWinc...
           +0.5*sigma*(peta+sigma*Ytemp)*(sWinc^2-Dt);
   Yem(j) = Ytemp ;    #   schwache Milstein-Loesung
endwhile
plot([0:Dt:T],[Xzero,Xem],'r--*'), hold off
figure(2)
plot([0:dt:T],[Xzero,Xtrue],'g-'), hold on
plot([0:Dt:T],[Xzero,Yem],'b--*'), hold off
xlabel('t','FontSize',16); ylabel('X_t','FontSize',16);
eerrx = abs(Xem(end)-Xtrue(end))
merry = abs(Yem(end)-Xtrue(end))
# Programmende
```

2) Hier muss nur die Lösungsformel (10.17) konkretisiert werden, es gilt dann

$$
\begin{aligned}
c_t &= 1 + \int_0^t \frac{1}{h_s}(2 - 3 * 1)\, ds + \int_0^t \frac{-3}{h_s}\, dW_s \,, \\
h_t &= \exp(\int_0^t (1 - \frac{1}{2}1^2)\, ds + \int_0^t dW_s) = \exp(\frac{1}{2}t + W_t - W_0) \,.
\end{aligned}
$$

und

$$X_t = c_t h_t = [1 - \int_0^t \frac{1}{h_s}\, ds - 3 \int_0^t \frac{1}{h_s}\, dW_s] h_t \ .$$

3) Mit dem folgenden Programm löst man die Aufgabe. einer effizienteren Programmierung.

```
# Programm loesung103.m
# Milstein-Methode zur numerischen Loesung von
#
# SDE: dX = (alpha + mu*X) dt + (peta + sigma*X) dW, X(0) = Xzero ,
# alpha = 2 , mu = -3 , peta = 1 , sigma = 1 und Xzero = 1 .
#
# Mittelwert von M Loesungspfaden
# der numerischen Loesung im Vergleich zur exakten Loesung
#
# Wiener-Prozess ueber [0,1], dt = 2^(-8).
# Milstein auf einem Gitter mit Dt = R*dt (R Zweierpotenz)
randn('state',100) # random seeding
alpha = 2 ; mu = -3 ; peta = 1 ; sigma = 1 ; Xzero = 1 ; # Problemparameter
T = 1 ; N = 2^8 ; dt = 1/N; M = 100; #
R = 4 ; Dt = R*dt ; L = N/R; # L Schritte der numerischen Verfahren Dt = R*dt
dW = zeros(M,N); W = zeros(M,N); Xtrue = zeros(1,N); # Pre-Allokierung
dW = sqrt(dt)*randn(M,N) ; # Inkremente des Wiener-Prozesses
W = cumsum(dW,2) ; # Wiener-Prozess
Uemean = zeros(1,N); Unmean = zeros(1,L); # Pre-Allokierung
s = 0;
while (s < M)
   s = s + 1;
   dw = dW(s,:);
   w = W(s,:);
#  Aufbau des s-ten Pfades der exakten Loesung
   j = 0;
   while (j < N)
     j = j + 1;
     ct1 = 0;
     k = 0;
     while (k < j)
       k = k + 1;
       summa = exp( (0.5*sigma^2-mu)*k*dt - sigma*w(k) );
       ct1 = ct1 + summa*dt;
       ct2(k) = exp( (0.5*sigma^2-mu)*k*dt - sigma*w(k) );
     endwhile
     wito = ito_integral_gb(ct2(1:j),dw(1:j),j);
     Xtrue(j) = ( Xzero+exp(sigma*w(1))*( (alpha-peta*sigma)*ct1+peta*wito ) )...
              *exp( (mu-0.5*sigma^2)*j*dt+sigma*(w(j)-w(1)) );
   endwhile
   Uemean = Uemean + Xtrue;
#  Berechnung des s-ten Pfades des numerischen Milstein-Loesung
   Yem = zeros(1,L) ; # Pre-Allokierung
   Ytemp = Xzero ;
   j = 0;
   while (j < L)
     j = j + 1;
     sWinc = sqrt(Dt)*sign(sum(dw(R*(j-1)+1:R*j))) ;
     Ytemp = Ytemp+Dt*(alpha+mu*Ytemp)+(peta+sigma*Ytemp)*sWinc...
             +0.5*sigma*(peta+sigma*Ytemp)*(sWinc^2-Dt);
     Yem(j) = Ytemp ;    #    schwache Milstein-Loesung
   endwhile
   Unmean = Unmean + Yem;
endwhile
Uemean = Uemean/M;
Unmean = Unmean/M;
plot([0:dt:T],[Xzero,Uemean],'b--'), hold on
plot([0:Dt:T],[Xzero,Unmean],'r--*'), hold off
xlabel('t','FontSize',16); ylabel('X_{mean}','FontSize',16);
for j=1:L
   Uem(j) = Uemean(R*(j-1)+1);
endfor
error = norm(Uem-Unmean)
# Programmende
```

4) Mit dem folgenden Programm löst man die Aufgabe.

```
# Programm loesung104.m
# Milstein-Methode zur numerischen Loesung von
# SDE: dX = kappa(theta - X) dt + sigma*sqrt(X) dW, X(0) = Xzero ,
# Wurzel-Diffusionsprozess
#
# Wiener-Prozess ueber [0,1], dt = 2^(-8).
# Milstein auf einem Gitter mit Dt = R*dt
# Input-Parameter: kappa, theta, sigma und Xzero
# Aufruf: loesung104(kappa,theta,sigma,x0);
function loesung104(kappa,theta,sigma,Xzero);
randn('state',100) # random seeding
T = 1 ; N = 2^8 ; dt = 1/N;
dW = zeros(1,N); W = zeros(1,N); # Pre-Allokierung
dW = sqrt(dt)*randn(1,N) ; # Inkremente des Wiener-Prozesses
W = cumsum(dW) ; # Wiener-Prozess
# Mit R = 4,2,1 waehlt man die Feinheit der Zeitdiskretisierung (R=4 -> grob...)
R = 2 ; Dt = R*dt ; L = N/R; # L Schritte der numerischen Verfahren Dt = R*dt
Yem = zeros(1,L); # Pre-Allokierung
Ytemp = Xzero ;
j = 0;
while (j < L)
    j = j + 1;
    Winc = sum(dW(R*(j-1)+1:R*j)) ;
    Ytemp = Ytemp+Dt*kappa*(theta- Ytemp)+sigma*sqrt(Ytemp)*Winc...
                +0.25*sigma^2*(Winc^2-Dt);
    Yem(j) = Ytemp ;    #    Milstein-Loesung
endwhile
plot([0:Dt:T],[Xzero,Yem],'g--*')
xlabel('t','FontSize',16); ylabel('X_t','FontSize',16);
endfunction
```

5) Wir gehen von

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-\xi^2}\, d\xi \quad \text{und} \quad \Phi(x) = \Phi(x;0,1) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^x e^{-\frac{\xi^2}{2}}\, d\xi$$

aus und erhalten nach Differentiation

$$\mathrm{erf}'(x) = \frac{2}{\sqrt{\pi}} e^{-x^2} \quad \text{und} \quad \Phi'(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \ ,$$

also gilt

$$\mathrm{erf}'(\frac{x}{\sqrt{2}}) = \frac{2}{\sqrt{\pi}} e^{-\frac{x^2}{2}} = 2\sqrt{2}\Phi'(x) \Longleftrightarrow \mathrm{erf}'(\frac{x}{\sqrt{2}}) = 2\sqrt{2}\Phi'(x) \ .$$

Durch Integration ergibt sich

$$\sqrt{2}\mathrm{erf}(\frac{x}{\sqrt{2}}) = 2\sqrt{2}\Phi(x) + C$$

mit der Integrationskonstanten C. Da $\mathrm{erf}(0) = 0$ und $\Phi(0) = \frac{1}{2}$ gilt, folgt für die Integrationskonstante $C = -\sqrt{2}$. Das bedeutet

$$\mathrm{erf}(\frac{x}{\sqrt{2}}) = 2\Phi(x) - 1 \quad \text{bzw.} \quad \Phi(x) = \frac{1}{2} + \frac{1}{2}\mathrm{erf}(\frac{x}{\sqrt{2}}) \ .$$

6) Es gilt

$$P\{x_1 \le X < x_2\} = \frac{1}{\sigma\sqrt{2\pi}} \int_{x_1}^{x_2} e^{-\frac{(\xi-\mu)^2}{2\sigma^2}}\, d\xi \ .$$

Die Substitution $\xi = \mu + \sigma\eta$, $d\xi = \sigma d\eta$ liefert

$$P\{x_1 \le X < x_2\} = \frac{1}{\sqrt{2\pi}} \int_{\frac{x_1-\mu}{\sigma}}^{\frac{x_2-\mu}{\sigma}} e^{-\frac{1}{2}\eta^2}\, d\eta \ ,$$

also ist mit $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}\xi^2}\, d\xi$

$$P\{x_1 \leq X < x_2\} = \Phi(\frac{x_2 - \mu}{\sigma}) - \Phi(\frac{x_1 - \mu}{\sigma}) \; .$$

Appendix

In diesem Anhang sind die Themen ”Monte-Carlo Methode” und ”Lineare Optimierung” kurz erläutert. Zur Monte-Carlo-Methode sei zum genaueren Lesen auf das sehr gehaltvolle Buch des sowjetischen Mathematikers I.M. Sobol ”Die Monte-Carlo-Methode” (deutsche Übersetzung z.B. vom ”Deutschen Verlag der Wissenschaften”, Berlin 1971) verwiesen.
Beschreibungen zur Lösung linearer Optimierungsprobleme findet man in den im Buch zitierten ”Numerical Recipes” oder z.B. in der Monographie ”Methoden und Probleme der Unternehmensforschung” von Sasieni, Yaspan, Friedman (deutsche Übersetzung ”Physica-Verlag”, Würzburg 1965).
Die Quelltexte der in diesem Anhang angegebenen Programme sind in den Aufgabenlösungs-Files zu finden.

Monte-Carlo Methode

Im Folgenden soll die Monte-Carlo Methode am Beispiel der Berechnung eines bestimmten Integrals erläutert werden.
Soll irgendeine unbekannte Größe m berechnet werden, dann versucht man eine Zufallsgröße ξ zu finden, für die der Erwartungswert der Beziehung

$$E[\xi] = m$$

genügt. Die Varianz (Dispersion) sei $V[\xi] = b^2$. Sind nun $\xi_1, \xi_2, \ldots, \xi_N$ unabhängige Zufallszahlen, deren Verteilung mit der von ξ übereinstimmt, und ist N hinreichend groß, dann gilt

$$P\{|\frac{1}{N}\sum_{j=1}^{N}\xi_j - m| < \frac{3b}{\sqrt{N}}\} \approx 0{,}997\ . \tag{3}$$

D.h. man kann m durch den Mittelwert näherungsweise berechnen, also

$$m \approx \frac{1}{N}\sum_{j=1}^{N}\xi_j\ ,$$

und man hat durch

$$\frac{3b}{\sqrt{N}}$$

eine Fehlerinfomation im Sinne der Beziehung (3).
Die eben beschriebene näherungsweise Berechnung einer unbekannten Größe soll nun am Beispiel der Berechnung des Integrals

$$I = \int_a^b g(x)\,dx \tag{4}$$

verwendet werden. Sei $p_\xi(x)$ eine beliebige Verteilungsdichte, die im Intervall $]a,b[$ definiert ist, mit den bekannten Dichteeigenschaften

$$p_\xi(x) > 0\ \ (a < x < b), \quad \int_a^b p_\xi(x)\,dx = 1\ .$$

Mit der im Intervall $]a,b[$ definierten Zufallsgröße ξ mit der Dichte p_ξ führen wir die Zufallsgröße

$$\eta = \frac{g(\xi)}{p_\xi(\xi)}$$

ein. Aus der Wahrscheinlichkeitsrechnung ist nun die Beziehung

$$E[\eta] = \int_a^b \eta\, p_\xi(x)\,dx = \int_a^b \frac{g(x)}{p_\xi(x)}\, p_\xi(x)\,dx = I$$

für den Erwartungswert von η bekannt. D.h. man kann das Integral I durch die Beziehung

$$\frac{1}{N}\sum_{j=1}^{N}\eta_j = \frac{1}{N}\sum_{j=1}^{N}\frac{g(\xi_j)}{p_\xi(\xi_j)} \approx I$$

mit $\xi_1, \xi_2, \ldots, \xi_N$ unabhängigen Zufallszahlen aus dem Intervall $]a, b[$, deren Verteilung mit der von ξ übereinstimmt, berechnen. Der Fehler übersteigt dabei den Wert $3\sqrt{\frac{V[\eta]}{N}}$ mit großer Wahrscheinlichkeit nicht. Die einfachste Wahl von $p_\xi(x)$ ist offensichtlich

$$p_\xi(x) = \frac{1}{b-a} \; .$$

Man kann aber leicht zeigen, dass die Varianz $V[\eta]$

$$V[\eta] = E[\eta^2] - (E[\eta])^2 = \int_a^b \frac{g^2(x)}{p_\xi(x)}\, dx - I^2$$

dann minimal wird, wenn $p_\xi(x)$ proportional zu $|g(x)|$ ist. Wir werden beim nachfolgenden Beispiel sehen, dass sich die Wahl einer geeigneten nichtkonstanten Dichte $p_\xi(x)$ durchaus lohnt.

Beispiel:
Es soll das Integral

$$\int_0^{\pi/2} \sin x \, dx$$

berechnet werden. Wir setzen das Vorhandensein einer zwischen 0 und 1 **gleichverteilten** Zufallsgröße γ voraus. Über die Beziehung

$$\int_a^\xi p_\xi(x)\, dx = \gamma \tag{5}$$

bestimmt man eine Zufallsgröße ξ, die im Intervall $]a, b[$ mit der Dichte $p_\xi(x)$ verteilt ist. Den Nachweis der Beziehung (5) zum sogenannten "auslosen" von Zufallszahlen mit einer Dichte $p_\xi(x)$ und viele andere kluge Ausführungen findet man z.B. in "Die Monte-Carlo Methode" von I.M. Sobol vom VEB Deutscher Verlag der Wissenschaften, Berlin 1983 oder in seinem sehr guten Buch "Numerische Monte-Carlo-Methoden" (russ.), Verlag "Nauka", Moskau 1973.
Mit der konstanten Dichte

$$p_\xi(x) = \frac{2}{\pi}$$

ergibt sich im Beispielfall gemäß (5) aus

$$\int_0^\xi p_\xi(x)\, dx = \int_0^\xi \frac{2}{\pi}\, dx = \gamma$$

die in $]0, \frac{\pi}{2}[$ mit der Dichte $p_\xi(x) = \frac{2}{\pi}$ verteilte Zufallsgröße

$$\xi = \frac{\gamma\pi}{2} \; .$$

Mit dem Programm

```
# Programm zur Berechnung des Integrals \int_0^\pi sin(x) dx
# mit der Monte-Carlo-Methode
# input:  z, Feld von nsamples Zufallszahlen
```

```
# output: integral genaeherter Integralwert, fehler
# Aufruf: [integral,fehler] = montecarlo1(z,nsamples);
function [integral,fehler] = montecarlo1(z,nsamples);
dichte = 2/pi;
mw = 0;
var = 0;
for j=1:nsamples
  xi = pi*z(j)/2;
  mw = mw + sin(xi)/dichte;
  var = var + (sin(xi)/dichte)^2;
endfor
integral = mw/nsamples;
fehler = sqrt((var/nsamples - integral^2)/nsamples);
endfunction
```

wird die näherungsweise Integralberechnung mit der konstanten Dichte realisiert. Wählt man nun die Dichte

$$p_\xi(x) = \frac{8x}{\pi^2} \ ,$$

von der als Übung die Dichteeigenschaft gezeigt werden sollte, dann erhält man gemäß (5) aus

$$\int_0^\xi \frac{8x}{\pi^2}\, dx = \gamma$$

die in $]0, \frac{\pi}{2}[$ mit der Dichte $p_\xi(x) = \frac{8x}{\pi^2}$ verteilte Zufallsgröße

$$\xi = \frac{\pi}{2}\sqrt{\gamma} \ .$$

Im folgenden Programm ist die Integralberechnung des Beispiels mit der nicht-konstanten linearen Dichte $p_\xi(x) = \frac{8x}{\pi^2}$ realisiert.

```
# Programm zur Berechnung des Integrals \int_0^{\pi/2} sin(x) dx
# mit der Monte-Carlo-Methode und einer nicht-konstanten Dichte p = 8x/pi^2
# input:  z, Feld von nsamples Zufallszahlen
# output: integral genaeherter Integralwert, fehler
# Aufruf: [integral,fehler] = montecarlo2(z,nsamples);
function [integral,fehler] = montecarlo2(z,nsamples);
mw = 0;
var = 0;
for j=1:nsamples
  xi = pi*sqrt(z(j))/2;
  dichte = 8*xi/pi^2;
  mw = mw + sin(xi)/dichte;
  var = var + (sin(xi)/dichte)^2;
endfor
integral = mw/nsamples;
fehler = sqrt((var/nsamples - integral^2)/nsamples);
endfunction
```

Die Anwendung der Programme zeigt den Vorteil der Verwendung der linearen Dichte $p_\xi(x) = \frac{8x}{\pi^2}$, die zwar nicht proportional zur Funktion $g(x) = \sin x$ ist, aber einen ähnlichen Verlauf hat. Mit den 10 Zufallszahlen $\gamma_1, \ldots, \gamma_{10}$ (hier könnte man mit dem Octave-Befehl *rand(1,N)* N gleichverteilte Zufallszahlen erzeugen)

j	1	2	3	4	5	6	7	8	9	10
γ_j	0,865	0,159	0,079	0,566	0,155	0,664	0,345	0,655	0,812	0,332

erhält man mit der konstanten Dichte (Programm "montecarlo1") den Näherungswert

$$I \approx 0{,}95179$$

und mit der linearen Dichte (Programm ”montecarlo2”) den deutlich genaueren Wert

$$I \approx 1{,}0160 \ .$$

Zu der beschriebenen Methode mit einer nicht-konstanten Dichte ist der Aufwand der Berechnung von ξ ausgehend von γ über die Beziehung (5) zu berücksichtigen. Dieser sollte sinnvollerweise nicht zu hoch sein, d.h., die Dichten sollten nicht zu kompliziert sein oder eben konstant, falls man keine geeignete Dichte für die zu integrierende Funktion findet.

Die für die Integralberechnung beschriebene Monte-Carlo-Methode ist aber nur dann Quadraturformeln vorzuziehen, wenn es sich um mehr- oder hochdimensionale Integrale handelt. Denn im Fall von Mehrfach-Integralen ist die Realsierung von Quadraturformeln ungeheuer aufwendig, während die beschriebene Monte-Carlo-Methode sehr einfach zu implementieren ist.

Lineare Optimierung

Die lineare Optimierung oder auch lineare Programmierung hat die Bestimmung von Extremwerten (wir betrachten hier die Suche nach einem Minimum, den Fall der Suche eines Maximums erfasst man durch die Suche nach dem Minimum von $-f(x_1, x_2, \ldots, x_n)$) einer linearen Funktion

$$f(x_1, x_2, \ldots, x_n) = \sum_{k=1}^{n} x_i c_i = \min! \quad (c_i \in \mathbb{R}) \tag{6}$$

unter Berücksichtigung von Nebenbedingungen der Form

$$\begin{aligned} x_1 a_{j1} + x_2 a_{j2} + \cdots + x_n a_{jn} &\leq b_j \quad (j = 1, \ldots, m_1) && (7)\\ x_1 a_{j1} + x_2 a_{j2} + \cdots + x_n a_{jn} &\geq b_j \quad (j = m_1 + 1, \ldots, m_1 + m_2) && (8)\\ x_1 a_{j1} + x_2 a_{j2} + \cdots + x_n a_{jn} &= b_j \quad (j = m_1 + m_2 + 1, \ldots, m_1 + m_2 + m_3), && (9) \end{aligned}$$

zum Ziel, wobei $x_j \geq 0,\ b_j \geq 0$ ($j = 1, \cdots, m = m_1 + m_2 + m_3$) gefordert ist. Mit der Einführung sogenannter Schlupfvariablen

$$x_{n+1}, x_{n+2}, \ldots, x_{n+m_1+m_2} \geq 0$$

kann man das Problem (6), (7)-(9) in der Form

$$f(x_1, x_2, \ldots, x_n, x_{n+1}, \ldots, x_{n+m_1+m_2}) = \sum_{k=1}^{n+m_1+m_2} x_i c_i = \min! \tag{10}$$

($c_i \in \mathbb{R}, c_{n+1} = \cdots = c_{n+m_1+m_2} = 0$) mit den Nebenbedingungen

$$\begin{aligned} x_1 a_{j1} + x_2 a_{j2} + \cdots + x_n a_{jn} + x_{jn+j} &= b_j \ , j = 1, \ldots, m_1, && (11)\\ x_1 a_{j1} + x_2 a_{j2} + \cdots + x_n a_{jn} - x_{jn+m_1+j} &= b_j \ , j = m_1 + 1, \ldots, m_1 + m_2 && (12)\\ x_1 a_{j1} + x_2 a_{j2} + \cdots + x_n a_{jn} &= b_j \ , j = m_1 + m_2 + 1, \ldots, m_1 + m_2 + m_3, && (13) \end{aligned}$$

aufschreiben. Mit den Verabredungen $N = n + m_1 + m_2$ und $M = m_1 + m_2 + m_3$ und

$$\vec{x} = (x_1, x_2, \ldots, x_N)^T \ , \quad \vec{c} = (c_1, \ldots, c_n, 0, \ldots, 0)^T \in \mathbb{R}^N \ , \quad \vec{b} = (b_1, \ldots, b_M)^T$$

und der Koeffizientenmatrix A vom Typ $M \times N$ des Gleichungssystems (11),(12),(13) kann man das Problem (10),(11),(12),(13) in der kompakten Form

$$\begin{aligned} f(\vec{x}) = \vec{c}^{\,T} \cdot \vec{x} &= \min! && (14)\\ A\vec{x} &= \vec{b} && (15)\\ \vec{x}, \vec{b} &\geq \vec{0} && (16) \end{aligned}$$

aufschreiben. Dabei bedeutet $\vec{a} \geq \vec{d}$ bzw. $A \geq D$, dass die Vektor- bzw. Matrixkomponenten von $\vec{a}$ bzw. A größer oder gleich denen von $\vec{d}$ bzw. D sind.

Die Nebenbedingungen (7)-(9) bzw. (11)-(13) beschreiben ein konvexes Polyeder des $\mathbb{R}^M$, das durch Hyperebenen berandet ist (jede Gleichung (11), (12) oder (13) beschreibt eine Hyperebene im $\mathbb{R}^M$). Es lässt sich zeigen, dass lineare Funktionen der Form (10) bzw. (14) ihre Extremwerte in den Ecken des Polyeders (oder auf Randflächen, falls die Ebenenschar $\vec{x} \cdot \vec{c} = \gamma,\ \gamma \in \mathbb{R}$ parallel zu einer Randfläche des Polyeders ist) annehmen. Hat man eine Ecke des Polyeders, d.h. eine nichtnegative Lösung $\vec{x}$ der Gleichung

$$A\vec{x} = \vec{b}$$

gefunden, kann man sich auf den Polyderkanten bewegen und prüfen, ob der Wert der Zielfunktion fällt oder nicht. Wird der Wert auf sämtlichen Kanten, die in der betreffenden Ecke beginnen, größer, dann ist man fertig und hat das Minimum gefunden. Anderenfalls muss man zu einer anderen Ecke wechseln, die einen kleineren Funktionalwert ergibt. Dabei sucht man sich die Kante aus, auf der der Funktionalwert am stärksten fällt. Eine Polyederecke $\vec{x}$, d.h. eine nichtnegative Lösung des Gleichungssystems (15) nennt man auch **zulässige Lösung** oder **Basislösung**. Diese Methode des systematischen Suchens der optimalen Ecke nennt man **Simplexverfahren**.
Beschreibungen des Simplexverfahrens findet man z.B. in den Monographien von Sasieni, Yaspan, Friedman "Methoden und Probleme der Unternehmensforschung" (Physica-Verlag Würzburg 1965) oder in den grundlegenden Büchern von H. P. Künzi.

Bestimmung einer Basislösung

Hat man es bei den Nebenbedingungen nur mit Bedingungen der Form (11) zu tun, dann sieht man sofort, dass

$$\vec{x} = (0,0,\ldots,0,x_{1n+1},x_{2n+2},\ldots,x_{m_1n+m_1})^T = (0,0,\ldots,0,b_1,b_2,\ldots,b_{m_1})^T$$

eine Basislösung ist, mit der das Simplexverfahren gestartet werden kann. Zur Bestimmung einer Basislösung des allgemeinen Problems (14), (15), (16) formuliert man ein Hilfsproblem der Form

$$g(\vec{x},\vec{y}) = y_1 + y_2 + \cdots + y_M = \min! \tag{17}$$

$$A\vec{x} + \vec{y} = \vec{b} \tag{18}$$

$$\vec{x},\vec{y},\vec{b} \geq \vec{0} \tag{19}$$

mit einem Hilfsvektor $\vec{y} = (y_1, y_2, \ldots, y_M)^T \in \mathbb{R}^M$. Die optimale Lösung des Problems (17), (18), (19) ist offensichtlich $(\vec{x},\vec{y}) = (\vec{x},\vec{0})$ mit der nichtnegativen Lösung $\vec{x}$ des Gleichungssystems (15), falls das Gleichungssystem (15) eine nichtnegative Lösung hat, wovon wir ausgehen. D.h., mit der optimalen Lösung des Problems (17), (18), (19) hat man eine Basislösung $\vec{x}$ des eigentlichen Ausgangsproblems (14), (15), (16) bestimmt.
Diese Überlegungen führen dazu, dass man die Lösung des linearen Optimierungsproblems in 2 Schritten ermittelt.

1) Man bestimmt die optimale Lösung des Hilfs-Problems (17), (18), (19), für das man eine triviale Basislösung hat, mit dem Simplexverfahren. Im Ergebnis erhält man eine Basislösung des eigentlichen Ausgangsproblems (14), (15), (16).

2) Mit der im 1. Schritt bestimmten Basislösung $\vec{x}$ bestimmt man mit dem Simplexverfahren die optimale Lösung des Ausgangsproblems (14), (15), (16).

Im folgenden Programm von Björn Harzer (Stuttgart 1995) ist die beschriebene Methode implementiert.

```
# Programm zum Simplexverfahren der linearen Optimierung/Programmierung
# Berechnet die Loesung X eines linearen Programms in der Standardform (*)
#   C'X -> min (Zielfunktion)
#   A*X=B
#   X>=0
#   B>=0
#
# Eingabe: A = m x n - Matrix
#          B = m - Vektor
#          C = n - Vektor
# Ausgabe: X = n - Spaltenvektor
#
# Aufruf: X=simplex_o(A,B,C)
```

```
function X=simplex_o(A,B,C);
# 19.6.95 Bjoern Harzer
eps = 1.0e-10;
[m,n] = size(A);
B = B(:);
C = C(:);
b = length(B);
c = length(C);
if (b~=m | c~=n)
     disp('error in format');
     return
endif
# Eine Basisloesung des Problems (**)
#   y(1)+...+y(m) -> min
#   A*X+Y=B
#   X,Y>=0
# ist [X;Y] = [0;B]
II = n+1:n+m;                  # Indexmenge
X = B;                         # Basisloesung
U = eye(m);                    # Inverse zu a(:,II)
c = [zeros(n,1);ones(m,1)]; # Schlupfvariablen einfuehren
a = [A,eye(m)];
# Phase1: optimale Loesung von (**) bestimmen
K = 1:n+m; K(II) = 0*II; K = find(K~=0);
D = c(K)-a(:,K)'*U'*c(II);
while any(D<-eps)
  disp('Phase1');
  [min1,k]=min(D);
  k=K(k);
  Z=U*a(:,k);
  ZI=find(Z>eps);
  P=X(ZI)./Z(ZI);
  [min1,l] = min(P);
  j=II(ZI(l));
  Uneu=U(ZI(l),:)./Z(ZI(l));
  U=U-Z*Uneu;
  U(ZI(l),:)=Uneu;
  Xneu=X(ZI(l))./Z(ZI(l));
  X=X-Z*Xneu;
  X(ZI(l))=Xneu;
  II(ZI(l))=k;
  K = 1:n+m; K(II) = 0*II; K = find(K~=0);
  D = c(K)-a(:,K)'*U'*c(II);
endwhile
# Phase2: optimale Loesung von (**) gleich Basisloesung von (*),
#         nun optimale Loesung von (*) bestimmen
K = 1:n; K(II) = 0*II; K = find(K~=0);
for jj=1:length(II)
  if (II(jj) > length(C))
    disp('Problem nicht loesbar, Nebenbedingungsmenge ist leer!');
    X = [];
    return
  endif
endfor
D = C(K)-A(:,K)'*U'*C(II);
while any(D<-eps)
  disp('Phase2');
  [min1,k]=min(D);
  k=K(k);
  Z=U*A(:,k);
  ZI=find(Z>eps);
  if isempty(ZI)
     X=inf*C(:);
     disp('Zielfunktion unbeschraenkt!');
     return
  endif
  P=X(ZI)./Z(ZI);
  [min1,l] = min(P);
  j=II(ZI(l));
  Uneu=U(ZI(l),:)./Z(ZI(l));
  U=U-Z*Uneu;
  U(ZI(l),:)=Uneu;
  Xneu=X(ZI(l))./Z(ZI(l));
  X=X-Z*Xneu;
  X(ZI(l))=Xneu;
```

```
  II(ZI(1))=k;
  K = 1:n; K(II) = 0*II; K = find(K~=0);
  D = C(K)-A(:,K)'*U'*C(II);
endwhile
Y = X;
X = zeros(n,1);
X(II) = Y;
endfunction
# Ende des Programms
```

Als erstes Beispiel betrachten wir das Problem

$$\begin{aligned} f(x_1,x_2) &= x_1 - 3x_2 = \min! \\ -x_1 + x_2 &\leq 1 \\ \frac{3}{4}x_1 + x_2 &\leq 3\,. \end{aligned}$$

Als Eingangsparameter für das Programm **simplex_o.m** ergeben sich

$$a = \begin{pmatrix} -1 & 1 & 1 & 0 \\ \frac{3}{4} & 1 & 0 & 1 \end{pmatrix},\ b = \begin{pmatrix} 1 \\ 3 \end{pmatrix},\ c = \begin{pmatrix} 1 \\ -3 \\ 0 \\ 0 \end{pmatrix}\,.$$

Der Aufruf

```
x = simplex_o(a,b,c)
```

liefert das Ergebnis

$$x_1 = 1{,}14286\,, \quad x_2 = 2{,}14286$$

sowie $x_3 = x_4 = 0$. Bei der Eingabe von "a" und "c" muss man immer darauf achten, dass die Schlupfvariablen auch berücksichtigt werden.
Als zweites Beispiel betrachten wir das Problem

$$\begin{aligned} f(x_1,x_2,x_3) &= 2x_1 + x_2 - 3x_3 = \min! \\ 2x_1 + 3x_2 + x_3 &\leq 3 \\ x_1 - x_2 + 2x_3 &\leq 2 \\ x_1 + x_2 - x_3 &\geq 1\,. \end{aligned}$$

Als Eingangsparameter für das Programm **simplex_o.m** ergeben sich

$$a = \begin{pmatrix} 2 & 3 & 1 & 1 & 0 & 0 \\ 1 & -1 & 2 & 0 & 1 & 0 \\ 1 & 1 & -1 & 0 & 0 & -1 \end{pmatrix},\ b = \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix},\ c = \begin{pmatrix} 2 \\ 1 \\ -3 \\ 0 \\ 0 \\ 0 \end{pmatrix}\,.$$

Der Aufruf

```
x = simplex_o(a,b,c)
```

liefert das Ergebnis

$$x_1 = 0\,, \quad x_2 = 1\,, \quad x_3 = 0$$

sowie $x_4 = x_6 = 0,\ x_5 = 3$.
Als letztes Beispiel betrachten wir die lineare Optimierungsaufgabe

$$\begin{aligned} f(x_1, x_2, x_3) &= x_1 + 3x_2 + 2x_3 = \min! \\ 2x_1 + 3x_2 + x_3 &\leq 3 \\ x_1 - x_2 + 2x_3 &\leq 5 \\ x_1 + x_2 - x_3 &= 1\ . \end{aligned}$$

Als Eingangsparameter für das Programm **simplex_o.m** ergeben sich

$$a = \begin{pmatrix} 2 & 3 & 1 & 1 & 0 \\ 1 & -1 & 2 & 0 & 1 \\ 1 & 1 & -1 & 0 & 0 \end{pmatrix},\ b = \begin{pmatrix} 3 \\ 5 \\ 1 \end{pmatrix},\ c = \begin{pmatrix} 1 \\ 3 \\ 2 \\ 0 \\ 0 \end{pmatrix}\ .$$

Der Aufruf

```
x = simplex_o(a,b,c)
```

liefert das Ergebnis

$$x_1 = 1\ , \quad x_2 = 0\ , \quad x_3 = 0$$

sowie $x_4 = 1,\ x_5 = 4$.

Errata

Seite	Zeile	Text im Buch	zu ersetzen durch
40	3	$-\alpha_{i-1}x_n$	$-\alpha_{n-1}x_n$
40	12	$(i = 1, \ldots, n-1)$	$(i = n-1, \ldots, 1)$
147	7	$2[\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 2] = 3$	$1 \cdot [\frac{1}{2} \cdot 1 + 3 + \frac{1}{2} \cdot 2] = 4{,}5$
165	6	$t : [a, b] \to [-1{,}1]$	$t : [-1{,}1] \to [0, \sqrt{\frac{\pi}{2}}]$
382	3	rand('state',100)	randn('state',100)
384	5	$W_t = (W_t^{(1)}, W_t^{(2)}, \ldots, W_t^{(2)})$	$W_t = (W_t^{(1)}, W_t^{(2)}, \ldots, W_t^{(n)})$
386	15	...Fall $N = M$...	...Fall $n = m$...

Auf Seite 214 fehlt ein Abschnitt über das SOR-Verfahren (Gauß-Seidel-Verfahren mit Relaxation):

...und einer Gauß-Seidel-Iteration gleich sind.

7.5.3 Gauß-Seidel-Verfahren mit Relaxation

Ohne den Rechenaufwand gegenüber dem Gauß-Seidel-Verfahren wesentlich zu erhöhen, kann man durch eine Dämpfung oder Relaxation ein eventuelles Überschießen während des Iterationsprozesses vermeiden und in der Regel die Konvergenz beschleunigen. Das ist dann besonders hilfreich, wenn die Gauß-Seidel- Iterationsfolge alternierend konvergiert, d.h. $x_j^{(k)} < x_j$, $x_j^{(k+1)} > x_j$ usw., wobei x_j eine Koordinate der Lösung bezeichnet. Man macht also einen Gauß-Seidel-Schritt

$$\hat{\vec{x}}^{(k)} = D^{-1}(-L\vec{x}^{(k)} - U\vec{x}^{(k-1)} + \vec{b})$$

und berechnet durch

$$\vec{x}^{(k)} = \omega\hat{\vec{x}}^{(k)} + (1-\omega)\vec{x}^{(k-1)}$$

die neue Iterierte. In der Koordinatenform bedeutet die Relaxation

$$\begin{aligned} \hat{x}_j^{(k)} &= \frac{1}{a_{jj}}(b_j - \sum_{i=1}^{j-1} a_{ji}x_j^{(k)} - \sum_{i=j+1}^{n} a_{ji}x_j^{(k-1)}) \\ x_j^{(k)} &= \omega\hat{x}_j^{(k)} + (1-\omega)x_j^{(k-1)} \end{aligned}$$

Fasst man die Schritte zusammen, erhält man die Berechnungsformel

$$\vec{x}^{(k)} = S_\omega\vec{x}^{(k-1)} + \omega(D + \omega L)^{-1}\vec{b}$$

mit der Iterationsmatrix

$$S_\omega = (D + \omega L)^{-1}[(1-\omega)D - \omega U] = E - \omega(D + \omega L)^{-1}A\ ,$$

wobei wir wieder von der Zerlegung $A = L + D + U$ ausgehen und $A\vec{x} = \vec{b}$ lösen wollen. Bei $\omega > 1$ spricht man von Überrelaxation und bei $\omega < 1$ von Unterrelaxation. Das Verfahren heißt **Gauß-Seidel-Verfahren mit Relaxation** oder **Sukzessives Überrelaxationsverfahren**, abgekürzt **SOR-Verfahren** (successive overrelaxation). Das SOR-Verfahren wurde von D.M. Young in seiner Dissertation 1949 entwickelt und begründet und gilt als Grundstein für die moderne numerische Mathematik. Zur optimalen Wahl...

Die nachfolgenden Abschnitte werden in der Nummerierung zu 7.5.4 und 7.5.5.