MATLAB und Mathematik kompetent einsetzen

Eine Einführung für Ingenieure und Naturwissenschaftler

Bearbeitet von
Stefan Adam

2. überarbeitete und ergänzte Auflage 2016. Buch. 502 S. Softcover
ISBN 978 3 527 41262 4
Format (B x L): 16,9 x 24,4 cm
Gewicht: 974 g

Inhaltsverzeichnis

Teil I
Aufgabenstellungen

1
Übungen zum Kapitel 1

1.1
Eingebettete Übungen zu Kapitel 1

Übung 12–1 Die Fülle von neuen Begriffen soll sogleich mit ein paar einfachen Aufgaben gefestigt werden:

- Wieviele Freiheitsgrade (frei wählbare Zahlen) hat eine allgemeine mxn Matrix?
- Wieviele Freiheitsgrade hat eine nxn Diagonalmatrix?
- Wieviele Freiheitsgrade hat eine obere nxn Dreiecksmatrix?
- Welche Matrix erfüllt die Bedingungen einer oberen Dreiecksmatrix und gleichzeitig diejenigen einer unteren Dreiecksmatrix?
- Stellt ein gewöhnlicher Vektor eine „hohe“ oder eine „breite“ Matrix dar?
- Wieviele Elemente enthalten die beiden Nebendiagonalen einer nxn-Matrix?
- Vervollständigen Sie die englische Dimensionsdeklaration einer Matrix:
  “The size of a matrix is indicated by the number of . . . followed by the number of . . . . (with an ’x’ in between)!”

Übung 12–2 Welche Indizes haben die Elemente rechts oben, rechts unten, links unten, sowie in der Mitte der untersten Zeile bei einer 4x5 Matrix?

Übung 12–3 Geben Sie die Indizes der Diagonalelemente einer 4x4 Matrix der Reihe nach von links oben beginnend an!

Übung 12–4 Schreiben Sie die Transponierten z' des Zeilenvektors $z = (1\ 2\ 3\ 4)$ sowie M' der 2x3 Matrix M= $\begin{pmatrix} 1 & 2 & 3 \\ 10 & 20 & 30 \end{pmatrix}$ auf!

Übung 12–5 Bilden Sie die Transponierte A' der Matrix $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$ und anschließend $A' + A,\ \ A - A',\ \ (A + A),\ \ 2 * A$!

Übung 12–6 Verwenden Sie die Matrix $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$ und deren Transponierte A' zum Bilden der Produkte $A' * A$, $A * A'$, $A * A$!

Übung 13–1 – Diagonalmatrix: Im obigen Beispiel war nur eine einfache Schleife notwendig, ebenso genügt eine einfache Schleife zum Definieren einer Diagonalmatrix mit den Werten 1,2, ... ndim in der Diagonalen. Versuchen Sie es und testen Sie Ihr Mini-Programm mit verschiedenen Dimensionen!

Übung 13–2 – Indexwertmatrix: Ein kleines Doppelschleifenprogramm, das sehr nützlich ist zu Demonstrationszwecken, ist das folgende:
Füllen Sie eine Matrix mit Zahlenwerten, welche gerade dem Indexpaar entsprechen, aufgefasst als zweistellige Dezimalzahl. (Dieses Programm wird bei $n > 9$ keine sinnvollen Werte mehr liefern.) Die so definierte spezielle Matrix wird im Folgenden unter dem Namen Indexwertmatrix mehrfach verwendet.

Übung 13–3, Erstellen einer Distanztabelle Überlegen Sie sich, mit welchen Befehlsfolgen von MATLAB-Befehlen Sie die obenstehende Differenzenmatrix erhalten, wenn Sie vom Vektor der Höhenwerte `v = [ 4634 4478 3820 1620 678 ]'` ausgehen!

Übung 15–1 Zeichnen Sie die regulären Polygone vom 3- bis 12-Eck mit einer Ecke nach oben zeigend! Einzeln, und mehrere übereinander mit verschiedenen Farben.

1.2
Allgemeine Übungen zum Kapitel 1

Einfache Berechnungen
10–1 Die Erde in Zahlen

In vernünftiger Näherung beträgt der Erdumfang 40 000 km. Das Meter wurde so definiert, dass ein Kilometer gerade einem 'c', also einem Hundertstel eines Neugrades auf dem Äquator entspricht. Ein 'cc' ist dann also 10 Meter. (Die Seemeile war als eine Bogenminute der alten Grad-Einteilung definiert worden.)
Berechnen Sie die Oberfläche, das Volumen und die durchschnittliche Dichte der Erde (Die Masse der Erde beträgt $5.97 * 10^{24}$ kg.)
Die darin zu verwendenden Formeln sind $V_{\text{Kugel}} = 4/3 \cdot \pi \cdot r^3$, $O_{\text{Kugel}} = 4 \cdot \pi \cdot r^2$, $U = 2 \cdot \pi \cdot r$, $\rho = m/V$.
Verwenden Sie in allen Berechnungen dieses Abschnittes die Befehle `format ...` zur verständlichen Darstellung der Resultate.

10–2 Mond und Sonne

Die mittlere Distanz von der Erde zum Mond beträgt 384 400 km, diejenige zur Sonne

150 Millionen km. Beide Himmelskörper werden von der Erde aus mit dem gleichen totalen Öffnungswinkel von 31 Winkelminuten gesehen. (Bei einer Sonnenfinsternis passen beide genau übereinander.)
Berechnen Sie daraus die ungefähren Durchmesser von Mond und Sonne!

10–3 Der Eiffelturm

Die Gitterkonstruktion des Eiffelturms, erbaut für die Weltausstellung 1889, ist erstaunlich leicht, obwohl die Streben und Knotenelemente aus Stahl sind. Damals standen noch keine Aluminium-, oder Magnesiumlegierungen zur Verfügung und schon gar kein mit Kohlefasern verstärkter Kunststoff. Als Demonstration der leichten Bauweise soll die Masse der Luft in dem, den Turm umfassenden Zylinder berechnet werden und mit der Eisen-Masse des Bauwerks von 7000 t verglichen werden. Die Seitenlänge der Grundfläche beträgt 160 m und die Höhe 320 m, die Dichte von Luft kann als 1.2 kg/m^3 angenommen werden.

10–4 Physikalische Arbeitsleistung eines Bergwanderers

Bei Bergwanderungen wird als Richtwert für die Überwindung einer Höhendifferenz von 300 m eine Stunde eingesetzt. Welche durchschnittliche Leistung in Watt erbringt ein 75 kg schwerer Wanderer für den reinen Höhengewinn? Die Erdbeschleunigung beträgt 9.81 m/s^2.

10–5 Wieviele Sekunden hat ein Jahr?

Durch einfache Multiplikationen erhält man die Zahlen für die Anzahl Sekunden bzw. Minuten pro Tag, Woche, Jahr sowie Stunden pro Woche und Jahr.

10–6 Wie weit kommt ein Tanklaster mit seiner eigenen Ladung?

Von einem Tanklastfahrzeug kann man im Langstreckenbetrieb annehmen, dass es etwa 15 Liter Diesel-Treibstoff pro 100 km verbraucht. Für die volle Ladung kann man einen Wert von ungefähr 20 000 Liter einsetzen. Wie weit kann er also mit der eigenen Ladung fahren? Käme er rund um die Erde, falls es eine Straße gäbe?

10–7 Berechnungs-Spass beim Essen

Versuchen Sie die Gesamtlänge aller Pommes-Frites, bzw. aller Spaghetti auf einem Teller angenähert zu bestimmen, wenn diese in Längsrichtung geradegezogen aneinandergereiht werden. Hinweis: Für eine Portion kann man von etwa 300 Gramm ausgehen, das spezifische Gewicht der meisten Speisen ist nahezu 1 g/cm^3 und die Dicke von Pommes-Frites ist ca. 6 mm während der Durchmesser von gekochten Spaghetti etwa 2 mm beträgt.

Produktschreibweise mit ' *'

10–8 Binomische Formeln

Die Binomischen Formeln eignen sich besonders gut zum Einüben der in MATLAB immer verlangten Verwendung des Multiplikationsoperators '*' zwischen zwei Faktoren.

Wählen Sie zwei Zahlenwerte für `a` und `b`, z. B. `a = 10, b = 3`. Berechnen Sie nun die Resultate für $(a+b)^n$ und $(a-b)^n$, mit $n = 2, 3, 4$ je auf zwei Arten, nämlich direkt durch Bilden der Summe/Differenz und anschließendes Potenzieren, sowie aufwendiger durch Summieren der einzelnen Terme der binomischen Formeln, wie z. B. $a^3 - 3 * a^2 * b + 3 * a * b^2 - b^3$.
Geben Sie diese Formeln im symbolischen Modus ein und testen Sie die Eingaben durch Anwenden der Funktion factor(formel).

Implizite Schleifen und Summen

10–9 Summen von natürlichen Zahlenreihen

Der Mathematiker Karl Friedrich Gauß sollte als Primarschüler mit der Aufgabe beschäftigt werden, alle Zahlen von 1 bis und mit 100 zusammenzuzählen. Da er sofort das Prinzip herausfand, dass jede Zusammenfassung einer Zahl aus der unteren Hälfte mit einer passenden aus der oberen Hälfte den Wert 101 ergab, und dass es 50 solche Paare gab, fand er sehr schnell das Resultat $5050 = 101 * 50$.

Die allgemeine Formel für die Summe einer Reihe natürlicher Zahlen von a bis b lautet

$$s = 1/2 \cdot (a+b) \cdot (b-a+1).$$

Testen Sie diese Formel, indem Sie verschiedene Reihen mit dem Befehl `r = a:b` erzeugen und deren Summe mit `sum(r)`, sowie mit der obigen Formel berechnen.

10–10 Summen von allgemeinen arithmetischen Reihen

Eine allgemeine arithmetische Reihe ist definiert durch die Formel für das allgemeine Element: $a_k = a_1 + (k-1) \cdot d \;\; ; \quad k = 1 \ldots n$.

Die zugehörige Summenformel lautet:

$$s = 1/2 \cdot n \cdot (a_1 + a_n) = 1/2 \cdot n \cdot (2 \cdot a_1 + (n-1) * d).$$

Verwenden Sie wiederum den impliziten Schleifenoperator: (diesmal in der Form `a:d:b`), um verschiedene arithmetische Reihen zu erzeugen und anschließend deren Summe mit sum(r) zu berechnen. Vergleichen Sie jeweils den so bestimmten Summenwert mit dem Resultat der Formelauswertung!
(Mögliche Beispielwerte sind $a_1 = 1, 0, 10, -20, 0.1$, $d = 2, 3, -2, 5, 0.1$ und $n = 101, 12, 11, 10, 10$.) Trotz des Vorfaktors $1/2$ wird das Resultat für ganzzahlige a_1 und d immer ganzzahlig; ein ganz kleines mathematisches Wunder!

10–11 Summenwert von magischen Quadraten

Ein magisches Quadrat der Dimension nxn enthält alle natürlichen Zahlen zwischen 1 und n^2. Der Wert der überall gleichen Zeilen- und Spaltensummen lässt sich aus folgender Überlegung bestimmen: er muss n mal dem Durchschnittswert aller Elemente entsprechen. Berechnen Sie diesen Summenwert für $n = 3, 4, 6, 9$!

Matrixdefinition

10–12 Jedes Element hat seinen Platz!

Geben Sie in MATLAB alle möglichen 2x2 Matrizen ein, welche die vier Zahlen 1

.. 4, aber an verschiedenen Plätzen enthalten. Nennen Sie diese A1, A2 etc. Die 24 Matrizen sind alle verschieden, das können Sie mit `eqtest = A1 == A2` prüfen. Suchen Sie in diesen Matrizen Paare, welche sich durch die Abbildungen „Transponieren“ (')-Operator, sowie `fliplr()` bzw. `flipud()` ineinander überführen lassen. (Spiegelung an vertikaler bzw. horizontaler Mittellinie.)

10–13 Zeilen und Spalten

Aus den Zahlen 1 bis 12, alle der Reihe nach eingefügt, kann man auf verschiedene Weise Rechtecksmatrizen erzeugen. Diese haben die Dimensionen 1x12, 2x6, 3x4, 4x3, 6x2, 12x1. Geben Sie alle diese Varianten in MATLAB ein, wobei die Definition der Namen E, Z, D, V, S, C ein nachträgliches Abrufen zum Quervergleich der verschiedenen Matrizen erlaubt.
Die Funktion `reshape()` erlaubt, die verschiedenen Matrizen ineinander zu verwandeln. Finden Sie mit der MATLAB help Funktion selbst heraus, wie diese anzuwenden ist.
Testen Sie auch in diesem Fall, was passiert, wenn Sie versuchen zwei solche Matrizen zu addieren, zu subtrahieren oder zu vergleichen!
Wenden Sie die Funktionen `length()` und `[m n] = size()` auf diese Matrizen an und überlegen Sie sich anhand der Resultate die Arbeitsweise dieser Funktionen!

10–14 Matrizen als Bestandteile von Matrizen

Verwenden Sie die Möglichkeit, Matrizen selbst als Elemente in der Definition einer Matrix einzusetzen!
`Q = [ 1 0 ; 0 1 ]` kann in `H= [ Q Q ; Q Q]` verwendet werden. Erweitern Sie das Prinzip um eine 8x8-Matrix mit Schachbrettverteilung von 0 und 1 zu erzeugen!

10–15 Vektoren zu Matrizen zusammenfügen

Erzeugen Sie eine Serie von 4 (bzw. n) Zeilenvektoren mit unterschiedlicher Anzahl von Einsen in der Art `v1 = [ 1 0 0 0]`, `v2 = [ 1 1 0 0]`, `v3 = [ 1 1 1 0]`, `v4 = [ 1 1 1 1]`. Zeigen Sie dass durch Aufeinanderstapeln
`L = [ v1 ; v2 ; v3 ; v4]` der Vektoren eine untere Dreiecksmatrix entsteht.
Wenn Sie die Vektoren einzeln transponieren, so erhalten Sie durch seitliches Aneinanderfügen eine obere Dreiecksmatrix. `R = [ v1' v2' v3' v4']`.

10–16 Matrizen aneinanderfügen

Starten Sie mit einer 6x6 Einheitsmatrix `I6 = eye(6)`.
Erzeugen Sie geeignete Spalten- und Zeilenvektoren mit der Funktion `zeros(n,m)`, die Sie an `I6` anfügen können, um daraus je eine 7x7 Matrix zu erhalten, welche dann die Einsen in der oberen, bzw. unteren Nebendiagonalen aufweist!

10–17 Würfelkoordinaten

Erstellen Sie eine 3x8 Matrix W durch Nebeneinanderstellen der Eck-Koordinaten eines Würfels ABCD EFGH mit `A=[0 0 0]', B=[1 0 0]', C=[1 1 0]'`, und `E=[0 0 1]'` etc.. Setzen Sie die 3x16 Matrix Wl mit der Abfolge ABCDA E FBF GCG HDH E in `plot3(Wl(1,:),Wl(2,:),Wl(3,:))`ein!.

10–18 Einmaleins-Tabelle

Erzeugen Sie eine Einmaleins-Tabelle durch untereinander Anordnen von Zeilenvektoren mit impliziter Schleife der Art `k:k:10*k`, wobei k verschiedenen Werte annimmt.

10–19 Größter gemeinsamer Teiler, kleinstes gemeinsames Vielfaches

Bestimmen Sie zu den Zahlen 4, 7, 13, 20, 36, 49, 64, 72 eine (symmetrische) Matrix, welche die g.g.T. enthält, sowie eine mit den k.g.V.-Werten. Die beiden Funktionen in MATLAB heißen `gcd(a,b)` (greatest common divider) und `lcm(a,b)` (least common multiple).
Zeigen Sie mit Hilfe der Punkt-Multiplikation dieser Matrizen, dass gilt `gcd(a,b) .* lcm(a,b) = a .* b`

Fachausdrücke zur Matrizentheorie

10–20 Welche der folgenden Aussagen sind wahr?

- Eine symmetrische Matrix ist quadratisch.
- Eine antisymmetrische Matrix, addiert zur Transponierten ergibt die Nullmatrix.
- Eine Matrix mit lauter Nullen ist das Neutralelement der Matrixmultiplikation.
- Jede Diagonalmatrix ist regulär.
- Das Produkt quadratische Matrix mal Spaltenvektor ergibt wieder einen Spaltenvektor, falls die Multiplikation möglich ist.
- Das Produkt einer Rechtecksmatrix mit ihrer Transponierten ist immer möglich und ergibt eine quadratische Matrix.
- Jede symmetrische Matrix ist mit ihrer Transponierten identisch.

10–21 Zerlegung in symmetrischen und antisymmetrischen Anteil

Berechnen Sie aus der Matrix T den Ausdruck $T - T'$! Suchen Sie eine Methode, um den antisymmetrischen A und den symmetrischen Anteil S der Matrix T zu bestimmen, so dass gilt $A + S = T$!

$$T = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

10–22 Charakteristische Eigenschaften von speziellen Matrizen

Finden Sie je eine 3x3 Matrix der folgenden Typen: symmetrische Matrix, antisymmetrische Matrix, Nullmatrix, Diagonalmatrix, Einheitsmatrix, obere Dreiecksmatrix, untere Dreiecksmatrix. Bestimmen Sie von jedem Typ die Anzahl Freiheitsgrade. Suchen Sie möglichst viele Teilmengenrelationen der Art: „Die Einheitsmatrix bildet eine Teilmenge der Diagonalmatrizen."

Hilfsfunktionen zum Erzeugen von Matrizen

10–23 zeros(), ones(), eye()

Sehen Sie Sich die Resultate der MATLAB-Funktionen zeros(k), ones(k), eye(k) an für verschiedene quadratische Dimensionszahlen k, z. B. $k = 3, 7, 10$.
Wenden Sie zeros() und ones() zum Erzeugen einer nxm Rechtecksmatrix an.
Erzeugen Sie das Komplement zur Einheitsmatrix mit Nullen auf der Diagonalen und Einsen an den anderen Plätzen.

10–24 diag()

Sehen Sie in der Hilfefunktion nach, wie `diag()` anzuwenden ist. Erzeugen Sie anschließend eine Diagonalmatrix der Dimension nxn z. B $n = 15$ mit den fortlaufenden ungeraden Zahlen $1, 3, 5$ etc. in der Diagonalen.

10–25 Block-Diagonalmatrizen

Verwenden Sie das Prinzip, dass man Matrizen aus kleineren Matrizen zusammenfügen kann, um quadratische Matrizen der Dimension $k * n \times k * n$ ($k = 1, 2, 3, 4, 5$), ($n = 2, 3, 4$) zu erzeugen, welche auf der Diagonalen und auf k parallelen Linien im Abstand n dazu je Einsen haben und sonst Nullen. (Lauter Matrizen die mit eye(n) erzeugt wurden, werden zusammengefügt.)
Benutzen Sie die Funktion spy(M) zur grafischen Darstellung der von Null verschiedenen Werte einer Matrix. Diese Funktion ist besonders beim Arbeiten mit dünn besetzten Matrizen (englisch: sparse matrices) wertvoll.

10–26 Quadrieren von magischen Quadraten

Lernen Sie die MATLAB-Demo über magische Quadrate (Demo – Matrizen – Matrix-Manipulationen) kennen! Nehmen Sie daraus ein magisches Quadrat 3x3 oder 4x4 und probieren Sie das folgende Rezept aus: Ein magisches Quadrat der Dimension nxn kann man wie folgt quadrieren: In die nxn Teilmatrix am Platz (j,k) setzt man die Originalmatrix ein, plus den „Sockelwert" $(m_{jk} - 1) * n^2$: dies ergibt ein magisches Quadrat der Dimensionen $n^2 \times n^2$.

Logische Operatoren angewandt auf Matrizen

10–27 Invertiertes Schachbrettmuster

Die in einer früheren Übung erzeugte Matrix mit 0- und 1-Belegung in einem Schachbrettmuster soll durch Anwendung eines der logischen Vergleichsoperatoren '==', '<=', '>=', '<', '>' mit einer Matrix ones(8) oder zeros(8) verglichen werden. Das Resultat soll die in allen Elementen invertierte Matrix liefern (0 statt 1, 1 statt 0).

Logische Operatoren liefern 1 als 'true' und 0 als 'false' mit der gleichen Dimension wie die verglichenen Matrizen.

10–28 Negativbild durch logische Operatoren

Mit den Befehlen `E = zeros(7); E(6,3:6)=ones(1,4); E(4,3:5)= ones(1,3); E(2,3:6)= ones(1,4); E(2:6,2) = ones(5,1);` ergibt sich eine Matrix, welche mit der Funktion `spy(E)` einen Buchstaben E zeigt. Durch `N = E == zeros(7)` kann davon ein Negativbild erzeugt werden.

Übungen zum Programmieren von Schleifen

10–29 Indexwert-Dreiecksmatrix

Füllen Sie eine obere Dreiecksmatrix mit Zahlen welche gerade das eigene Indexpaar als zweistellige Zahl anzeigen (dies funktioniert nur für n<10), also z. B $a_{11} = 11$, $a_{34} = 34$, etc.

10–30 Füllen einer Dreiecksmatrix

Erstellen Sie ein M-File mit einer Doppelschleife, welches eine obere (rechte) Dreiecksmatrix mit 1 füllt für die allgemein vorgegebene Dimension 'n'.

10–31 Tridiagonalmatrix

Erstellen Sie ein M-File, das eine Tridiagonalmatrix mit den Werten 2 auf der Diagonalen und den Werten -1 auf den beiden Nebendiagonalen erzeugt.

10–32 Streifenmuster

Erstellen Sie ein MATLAB-Skript, welches eine Matrix der Dimension n mit folgendem Muster füllt:
Auf der Diagonale Einsen, auf den Nebendiagonalen Nullen, anschließend auf den nachfolgenden zu der Diagonalen parallelen Linien wieder Einsen, dann wieder Nullen etc.!

10–33 Spezielle untere Dreiecksmatrix

Füllen Sie eine untere Dreiecksmatrix mit den Werten, welche der Summe der beiden Indizes entsprechen.

„Turm"- und „Specht"-Matrizen

10–34 „Spechtmatrizen" – Effekt der Matrixmultiplikation

Füllen Sie eine 4x4 Matrix M der Reihe nach mit den Zahlen 1 .. 16.
Erzeugen Sie eine andere 4x4 Matrix S mit lauter Nullen, außer einer einzigen Eins. Beobachten Sie den Effekt, den die Multiplikationen S*M und M*S produzieren für verschieden gewählte Positionen der Eins innerhalb von S!

10–35 Spechtmatrizen – Zeilenselektion

Erstellen Sie als Testobjekt T eine 5x5 Indexwert-Matrix (mit zweistelligen Zahlen die den Indizes entsprechen, z. B.$a_{11} = 11$, $a_{34} = 34$)
Multiplizieren Sie diese mit der geeigneten Spechtmatrix von links her, mit dem Ziel, die erste Zeile von T in die 1., 2., 4. und 5. Zeile zu platzieren.
Bestimmen und testen Sie ebenso die notwendigen Spechtmatrizen für eine analoge Platzierung der 4. und der 5. Zeile von T.

10–36 Spechtmatrizen – Spaltenselektion:

Verwenden Sie wieder als Testobjekt T eine 5x5 Indexwert-Matrix.
Multiplizieren Sie diese Matrix mit der geeigneten Spechtmatrix von rechts her, mit dem Ziel, die erste Spalte von T in die 1., 2., 4. und 5. Spalte der Resultatmatrix zu platzieren.
Bestimmen und testen Sie ebenso die notwendigen Spechtmatrizen für eine analoge Platzierung der 4. und der 5. Spalte von T.

10–37 Durch Permutationsangabe definierte Turmmatrix

Erstellen Sie ein MATLAB-Programm (als Funktions-M-File), das einen Spaltenvektor mit den Zahlen von 1 bis n in beliebiger Reihenfolge als Eingabe benötigt, und daraus diejenige Turmmatrix produziert, welche die im Eingabevektor enthaltene Permutation erzeugt. Als Test kann die Multiplikation der Turmmatrix von links an einen Vektor mit der natürlichen Abfolge der Zahlen $1, 2, 3, \ldots, n$ dienen. Durch diese Multiplikation soll der vorgegebene Vektor erzeugt werden.

10–38 Scroll-up- und Scroll-down-Matrizen

Spezielle Turmmatrizen sind die Scroll-up- (bzw. Scroll-down)-Matrizen (fast alle Einsen direkt oberhalb/ unterhalb der Diagonalen). Bei der Multiplikation von links schieben Sie alle Zeilen der rechts stehenden Matrix (alle Werte eines rechts stehenden Vektors) um einen Platz nach oben (bzw. unten). Wird das am Rand herausfallende Element am anderen Rand eingefüllt, so nennt man die Scroll-up/down-Abbildung zyklisch. Fällt das Element am Rand weg, so sind die Matrizen keine eigentlichen Turmmatrizen mehr.
Erstellen Sie für die Dimension 5 alle 4 Typen: zyklische und nicht zyklische Scroll-up- und Scroll-down-Matrizen und testen Sie deren Wirkung an einem Spaltenvektor mit den Zahlen 1:5.
Überlegen Sie sich, welchen Rang diese 4 Matrizen haben und kontrollieren Sie Ihr Resultat mit der Funktion `rank()` von MATLAB.

10–39 Die Wirkung von Turmmatrizen

Testen Sie die Wirkung einiger Turmmatrizen (insbesondere der Sroll up/down Matrizen) beim Multiplizieren der Index-Anzeige-Matrix von links her und von rechts her.

10–40 Die Wirkung von Auswahl- und Permutationsmatrizen

Bestimmen Sie die Matrizen Pl und Pr so, dass für beliebige a_k .. d_k gilt:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ a_4 & a_3 & a_2 & a_1 \\ d_4 & d_3 & d_2 & d_1 \\ 0 & 0 & 0 & 0 \end{pmatrix} = Pl \cdot \begin{pmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \\ d_1 & d_2 & d_3 & d_4 \end{pmatrix} \cdot Pr$$

10–41 Turmmatrizen zu elementaren Permutationen

Turmmatrizen, welche sich nur an zwei Stellen von einer Einheitsmatrix unterscheiden: Beinahe-Einheitsmatrizen $B = I$, außer $b_{jk} = 1$, $b_{kj} = 1$ und $b_{jj} = 0$, $b_{kk} = 0$ bewirken nur eine einzige Permutation zwischen j und k. Aus Produkten von solchen Elementarpermutationen kann man jede Turmmatrix herstellen. Für jede B-Matrix gilt $B^2 = I$. Testen Sie diese beiden Eigenschaften an einfachen Beispielen, z. B. an der Permutation `[4 3 2 1]` aus `[1 2 3 4]`, bzw. an den Paaren j,k = 1,2, oder 1,4, oder 2,4.

Multiplikation von Matrizen

10–42 Matrixmultiplikation

Berechnen Sie von Hand die Matrizenprodukte

$$\begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} * \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \quad \text{und} \quad \begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} * \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

und kontrollieren Sie Ihr Resultat mit Hilfe von MATLAB!

10–43 Indexwertmatrix mal allgemeine Matrix

Multiplizieren Sie mit Bleistift und Papier je eine 2x2 und eine 3x3 Matrix deren Werte die Indizes als zweistellige Zahlen enthalten, jeweils von links und von rechts mit einer allgemeinen Matrix mit den Werten a, b, c, d bzw. a bis i.

10–44 Matrixmultiplikation „von Hand“

Multiplizieren Sie mit Hilfe von Bleistift und Papier eine 2x2 Matrix D mit den Zahlen (1 .. 4) mit sich selbst (d. h. berechnen Sie D*D von Hand). Multiplizieren Sie ebenso die 3x3 Matrix T mit den Zahlen (1 .. 9) mit sich selbst.

10–45 Formel für die Inverse einer 2x2 Matrix

Multiplizieren Sie mit Bleistift und Papier die zwei allgemeinen 2x2 Matrizen `[a b ; c d]` und `[u v ; w x ]`. Bestimmen Sie aus der Forderung, dass das Resultat die Einheitsmatrix `[1 0; 0 1]` ergeben muss vier Gleichungen, welche die Parameter u, v, w, x als Funktion von a, b, c, d erfüllen müssen.
Durch Auflösen dieser vier Gleichungen nach u, v, w, x erhält man eine geschlossenen formelmäßige Lösung für die Inverse einer 2x2 Matrix.
Im symbolischen Modus kann dieses Gleichungssystem als Matrizengleichung definiert werden.

10–46 Legalitätsüberlegung bei Matrizen- und Vektormultiplikationen
Mit den 4 Matrizen / Vektoren A 2x3, B 3x3, v 3x1, w 2x1 sollen die folgenden Operationen auf ihre Legalität überprüft werden:

A+B A+A B+B v+w w-v
A*B B*A A'*B B*A'
A*v v*A v*A' w*A' v*v' v'*v B*v B*w A'*v
A*A A'*A A*A' B*B

10–47 Legale und illegale Matrixmultiplikationen
Bilden Sie alle möglichen Rechtecksmatrizen, welche die Zahlen 1 bis 6 der Reihe nach enthalten, also E 1x6, Z 2x3, D 3x2 und S 6x1.
Bilden Sie zusätzlich deren Transponierte Et=E', Zt, etc. und suchen Sie alle legalen Multiplikationen, welche zwischen zwei von diesen 8 Matrizen möglich sind! Bestimmen Sie jeweils die Resultat-Dimensionen.

10–48 Legale und illegale Matrizen- und Vektor-Multiplikationen
Erzeugen Sie die vier Matrizen. bzw. Vektoren A(4x3), N(3x3), v(3x1), w(1x4), so dass die darin enthaltenen Zahlen den Index als zweistellige Zahl darstellen (z. B. $a_{12} = 12$, $n_{33} = 33$). Prüfen Sie mit MATLAB, welche der folgenden Multiplikationen legal sind und überlegen Sie sich jeweils vorgängig, ob Sie den Fall als legal eingestuft hätten: (M' steht fürM^T, d. h. M-transponiert)

w*A, w'*A, w*A', w'*A', A*w, A*w', A'*w, A'*w'
w*w, w*w', w'*w, v*v, v'*v, v*v'
N*v, v*N, v'*N, N*v'
A*N, A*N', A'*N, A'*N'

10–49 Nachprüfen einer Matrizengleichung
Zeigen Sie durch Nachrechnen von Hand, dass gilt:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{a*d-b*c} * \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

Formulieren von Gleichungssystemen in Matrizenform
10–50 In einzelnen Gleichungen vorgegebenes Gleichungssystem
Die folgenden Gleichungen bilden ein lineares Gleichungssystem. Stellen Sie dieses in Matrizenform dar und lösen Sie es mit MATLAB.

$$r+s+t+u=0, \qquad r-s=1, \qquad u+t-s=-1, \qquad t-r=1.$$

10–51 Gleichungssystem, durch einzelne Gleichungen gegeben
Lösen Sie das untenstehende Gleichungssystem mit MATLAB im Berechnungs- und im symbolischen Modus:

$$x=2*y, \qquad y+z=5, \qquad x-u=2, \qquad x+u=3*y+1.$$

10–52 In einem Schema vorgegebenes Gleichungssystem

Lösen Sie das untenstehende Gleichungssystem mit MATLAB, nachdem Sie es in Matrizenform gebracht haben.

$$\left|\begin{array}{rcrcrcrcr} x_1 & - & x_2 & & & & & = & 5 \\ & & x_2 & - & x_3 & + & x_4 & = & 3 \\ & & & & x_3 & - & x_4 & = & 1 \\ x_1 & & & & & + & x_4 & = & 3 \end{array}\right|$$

10–53 Gleichungssystem in gewohnter Form

Beachten Sie das richtige Einsetzen von Matrixelementen mit den Werten 0, 1 und -1, da diese Zahlen im Gleichungssystem nicht geschrieben werden. Lösen Sie dieses Gleichungssystem mit MATLAB, nachdem Sie es in Matrizenform gebracht haben.

$$\left|\begin{array}{rcrcrcr} 3x & + & y & - & z & = & 7 \\ x & & & + & 2z & = & 3 \\ & & y & - & z & = & 0 \end{array}\right|$$

Einfache Funktionsplots

10–54 Plot von geraden Potenzfunktionen

Erstellen Sie eine Grafik mit den Funktionen $y_1 = x^2$, $y_2 = x^4$, $y_3 = x^8$, $y_4 = x^{16}$ im Bereich $x = 0..2$, $y = 0..2$, alle in derselben Grafik, mit verschiedenen Farben.

10–55 Gerade und ungerade Potenzfunktionen

Stellen Sie die ersten 5 Potenzfunktionen $y_1 = x$, $y_2 = x^2$ etc. in einem gemeinsamen Bild im Bereich $-2 < x < 2$ und $-2 < y < 2$ dar. Wählen Sie für die ungeraden Potenzen die rote Farbe ('r') und für die geraden schwarz ('k').

10–56 Verschiedene Wurzelfunktionen

Erzeugen Sie eine gemeinsame Grafik der Funktionen

$$y_1(x) = x, \qquad y_2(x) = \sqrt{x}, \qquad y_3(x) = \sqrt[3]{x}, \qquad y_4(x) = \sqrt[4]{x}$$

im Bereich $x = (0..4]$

MATLAB-Operatoren und Grundfunktionen

10–57 MATLAB-Operatoren

Stellen Sie alle Ihnen bekannten Operatoren in MATLAB tabellarisch zusammen und geben Sie bei den speziellen Operatoren stichwortartig deren Bedeutung an!

10–58 Spiegelungsoperationen an Matrizen

Füllen Sie eine 3x3-Matrix mit den Werten von 1 bis 9. Testen Sie an dieser Matrix die drei „Spiegelungsoperationen“ Transponieren, flipud() und fliplr(). Suchen Sie eine Kombination aus diesen, welche eine „falsche Transposition“ bewirkt, d. h. eine Spiegelung an der von links unten nach rechts oben verlaufenden „falschen“ Diagonale.

Erarbeiten einer Funktion

10–59 Winkelfunktionen in Grad

Schreiben Sie selbst Funktions-M-Files sindeg(), cosdeg(), tandeg(), welche gleich funktionieren wie sind(), cosd(), tand(). Multiplizieren Sie dazu die Eingabewerte mit pi/180 und übergeben Sie das Resultat an die normalen Winkelfunktionen. Testen Sie die Anwendung von Matrizen und Vektoren als Eingabewerte.

10–60 Direkt aufrufbare Potenzfunktionen

Erzeugen Sie einige function-M-Files mit den Namen 'pow2', 'pow3', 'pow4', etc., welche die Funktionen `x.^2`, `x.^3`, etc. realisieren.

10–61 Teilvolumen in Pyramiden-Trichter

Suchen Sie die Formel für das Teilvolumen als Funktion des Flüssigkeitspegels in einem Trichter mit Pyramidenform. Die Neigung der Seitenwaäde soll 45° betragen.

10–62 Teilvolumen in liegendem Zylinder

Suchen Sie die Formel für das Teilvolumen als Funktion des Flüssigkeitspegels in einem liegenden Zylinder. Die Fläche des Kreis-Abschnittes in Funktion von h kann entweder geometrisch als Sektor minus Dreieck oder als
Integral(Sekantenlänge(y)*dy) bestimmt werden.

1.3 Miniprojekte zum MATLAB Einstieg

101 Fadenstern

Zeichnen Sie mit MATLAB einen vierstrahligen „Fadenstern“, indem Sie die Punkte von -10 bis 10 auf der x-Achse mit passenden Punkten auf der y-Achse durch Geraden verbinden.

102 Farbkreis

Benutzen Sie die Funktionen `fill()` oder `patch()`, um in einem Kreis angeordnete farbige Flächen mit den zu ihrer Winkelposition passenden Farben zu füllen. Die RBG-Werte zum Einfügen in die Grafikaufrufe erhalten Sie aus den Werten für Farbwinkel (hue), Sättigung (saturation) und Helligkeit (value) mit der Bibliotheksprozedur `[r,g,b]=hsv2rgb(h,s,v)`.

1.4 Selbsttests zum Kapitel 1

Diese Testserien dienen zur Selbstkontrolle der erworbenen Kenntnisse. Falls das erste Kapitel sorgfältig durchgearbeitet wurde, sollte es möglich sein, eine Serie in ca. 60 Minuten vollständig zu lösen.

Testserie 1.1

T111 **Verständnisfragen**

- Für welche arithmetischen Operatoren gibt es in MATLAB zugehörige Punkt-Operatoren und was ist die Bedeutung dieser Punkt-Operatoren?
- Geben Sie die notwendigen Befehle an, um mit einer Bibliotheksprozedur in MATLAB eine 4x4 Einheitsmatrix zu erzeugen.
- Wie erhält man in einer MATLAB-Grafik eine schwarze Linie?
- Wie extrahiert man die ganze 4. Zeile aus einer nxn Matrix (Annahme n ist mindestens 4)?

T112 Erzeugen Sie eine grafische Darstellung der Sinusfunktion über drei ganze Perioden mit gleich großen Einheiten in der x- und y-Richtung!

T113 Geben Sie die nötigen MATLAB-Befehle an, um eine 8x8 Matrix im Schachbrettmuster mit 0 und 1 zu füllen! Starten Sie mit der Eingabe einer 2x2-Matrix, fügen Sie vier davon zu einer 4x4 Matrix zusammen und dann 4 von diesen zur gesuchten 8x8 Matrix.

T114 Erzeugen Sie einen Vektor mit 505 Elementen, welcher 5 Perioden einer Sägezahnfunktion enthält, wobei jedesmal die Werte von–10 bis 10 laufen.

T115 Schreiben Sie die MATLAB-Befehle auf zum Lösen des Gleichungssystems im Berechnungs-Modus und im symbolischen Modus.

$$x + z = 4; \qquad y + z = 2; \qquad 2 * x - 4 * y = 2.$$

Testserie 1.2

T121 **Verständnisfragen**

- Wie erhält man in MATLAB die Lösung x eines in Matrixform gegebenen linearen Gleichungssystems A*x=b?
- Geben Sie die Befehle an zum Erzeugen der ausmultiplizierten Formel für $(a-b)^5$ im symbolischen Modus.
- Wie erreicht man, dass nachfolgende plot-Aufrufe in dasselbe Bild gezeichnet werden wie der vorangehende?
- Wie lautet der einzugebende Text, um in der Variablen E3 eine 3x3 Einheitsmatrix direkt einzugeben (ohne Verwendung einer Bibliotheksprozedur)?

T122 Erzeugen Sie eine simultane Grafik der Funktionen $y_q = \sqrt{x}$, $y_c = \sqrt[3]{x}$ und $y_f = \sqrt[4]{x}$, mit dem Intervall $\varepsilon < x < 2$.

T123 Das File 'tempmess.dat' enthalte in ASCII-Form eine Tabelle mit der Tages-Nummer als erster Zahl in jeder Zeile, gefolgt von drei Temperaturmessungen. Geben Sie die MATLAB-Befehle an, um dieses File einzulesen und anschließend die Temperaturwerte gegen die Messzeit in einem gemeinsamen Zeichenfeld zu plotten.

T124 Programmieren Sie in MATLAB eine Schleife, welche mit einer if-Konstruktion eine mit 201 Werten tabellierte Sinusfunktion bei Werten über 0.6 und unter -0.6 auf die Grenzwerte zurücksetzt (Clipping)!

T125 Berechnen Sie von Hand das nachfolgende Matrizenprodukt!

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 8 \\ 0 & 5 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 3 & 0 \end{pmatrix}$$

2
Übungen zum Kapitel 2

2.1
Eingebettete Übungen zum Kapitel 2

Übungen zu Relationen und Funktionen

Einige Übungsbeispiele dazu: (mit Hinweisen zum Arbeiten mit Ungleichungen)

Übung 21–1 Definieren Sie als X die Menge der reellen Zahlen $-10 < x < 10$ und als Y die Menge $0 < y < 10$. Dies ergibt als Kartesische Produktmenge das Rechteck oberhalb der x-Achse und beidseits der y-Achse mit dem Querformat 20 mal 10. Zeichnen Sie diese Fläche auf ein Blatt Papier! In diese Fläche können Sie nun verschiedene Teilflächen einzeichnen, welche je eine Relation charakterisieren. Als Beispiele für die Bedingungen zur Definition von Relationen sollen die folgenden Ungleichungen dienen:

a) $y < x$, b) $y < x^2$, c) $|y - x| < 1$, d) $|\,|x| + |y|\,| < 5$,

(beim Fall d) mit dem quadratischen Bereich $-10 < x < 10$ und $-10 < y < 10$).

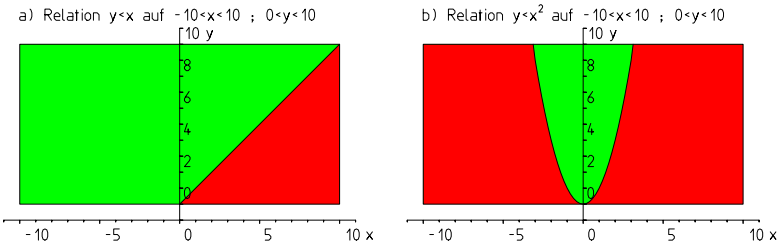


Abbildung 2.1 Relationen: Teilmengen einer kartesischen Produktmenge, dargestellt als Teilgebiete.

Erklärungen zu a) Für das Teilbeipiel a) bildet die Gerade $y = x$ die Trennlinie. Alle y -Werte unterhalb dieser Linie erfüllen die Relationsbedingung. Somit wird die Relation durch das Gebiet im Rechteck der Kronecker'schen Produktmenge sowie unterhalb der Gerade $y = x$ (Winkelhalbierende des ersten Quadranten) dargestellt.
Erklärungen zu b) Analog dazu ist die Relation des Teilbeispiels b) durch das Gebiet unterhalb der Parabel $y = x^2$ repräsentiert.

Erklärungen zu c) Bei der Lösung von c) spaltet sich die Relationsbedingung wegen der Betragsstriche in zwei Bedingungen auf, welche je in einem Teilgebiet gelten.

- Falls $y > x$ ist, so gilt $y - x < 1$
  (Weglassen der Betragsstriche bei positivem Inhalt):
  Trennlinie $y - x = 1$ bzw. $y = x + 1$,
  Gebiet mit y-Werten unterhalb Trennlinie erfüllt die Relationsbedingung.
- Falls $y < x$ ist, so gilt $-y + x < 1$
  (Vorzeichen ändern vor Weglassen der Betragsstriche bei negativem Inhalt):
  Trennlinie $x - y = 1$ bzw. $y = x - 1$,
  Gebiet mit y-Werten oberhalb Trennlinie erfüllt die Relationsbedingung.

Erklärungen zu d) Die Betragsstriche bei d) führen grundsätzlich zu einer Fallunterscheidung: abhängig vom Vorzeichen des Inhaltes können die Betragsstriche weggelassen werden (bei positivem Inhalt) oder sie müssen durch ein Minuszeichen ersetzt werden.

1.Fall: $x > 0$ und $y > 0$ (I. Quadrant): $|x| + |y| = x + y < 5$
Grenzlinie $y = 5 - x$, Testpunkt $(1/1)$: im Gebiet.
2.Fall: $x < 0$ und $y > 0$ (II. Quadrant): $|x| + |y| = -x + y < 5$
Grenzlinie $y = 5 + x$, Testpunkt $(-1/1)$: im Gebiet.
3.Fall: $x < 0$ und $y < 0$ (III. Quadrant): $|x| + |y| = -x - y < 5$
Grenzlinie $y = -5 - x$, Testpunkt $(-1/-1)$: im Gebiet.
4.Fall: $x > 0$ und $y > 0$ (IV. Quadrant): $|x| + |y| = x - y < 5$
Grenzlinie $y = -5 + x$, Testpunkt $(1/-1)$: im Gebiet.

Übung 21–2 Mit X definiert als $x > 0$ und Y als $y > 0$ wird die Produktmenge gleich dem I. Quadranten. Einfache Beispiele von Relationen in dieser Kombination sind gegeben durch die Bedingungen $y > x$, oder auch $|y - 5| < 1$.
In diesem Beispiel würde die Relationsbedingung $y < -x$ keine Elemente zulassen.

Übung 21–3 Relationen können auch auf endlichen Mengen definiert werden. Sei X die Menge der ganzen Zahlen $-10 \leq x \leq 10$ und Y enthalte die ganzen Zahlen $0 \leq y \leq 20$. Dann umfasst die Relation (die bereits auch eine Funktion ist) $y = x^2$ die Zahlenpaare (-4,16), (-3,9), (-2,4), (-1,1), (0,0) (1,1), (2,4), (3,9), (4,16)

Übungen zu den komplexen Zahlen

Übung 24–1 Schreiben Sie die folgenden reellen, rein imaginären oder komplexen Zahlen ausführlich, eventuell durch null ergänzt, in der kartesischen Form und anschließend in der Polarkoordinatenform, und zeichnen Sie die Zahlen in der Gauß'schen Zahlenebene ein:

$1\,;\ -1\,;\ 0\,;\ 5\,;\ -2;$
$i\,;\ i*3\,;\ -i\,;\ -i*2;$
$1+i\,;\ -1-i\,;\ 1+i*\sqrt{3}\,;\ 1-i*\sqrt{3}.$

Übung 24–2 Führen Sie die folgenden Operationen in der kartesischen Form durch und gleichzeitig vektoriell in der Gauß'schen Zahlenebene!

$4 + (2 + i * 3)$; $(6 + i) + i * 2$; $(1 + i) + (2 + 2 * i)$;
$(4 + i) + (-1 - i * 4)$; $(3 + i * 2) + (3 - i * 2)$; $(3 + i * 2) - (3 - i * 2)$;
$(1 + i * 2) + (-1 - i * 2)$; $(3 - i * 3) + (-2 + i * 2)$

Übung 24–3 Versuchen Sie mit Hilfe der Gauß'schen Zahlenebene die Systematik der vier unten angegebenen Zahlenfolgen zu erraten. Dabei sollten Sie die zwei unmittelbar nachfolgenden Elemente aufschreiben und das allgemeine Element dieser Folge ausgedrückt durch dessen Index 'k' als Formel angeben. Falls die Folge ein sinnvolles letztes Element aufweist, sollte dieses ebenfalls bestimmt werden.

a) $7 + i * 2$, $-3 + i * 2$, $6 + i * 2$, $-2 + i * 2$, $5 + i * 2$, $-1 + i * 2$ ...
b) $6 + i * 4$, $-4 + i * 6$, $-6 - i * 4$, $4 - i * 6$...
c) $\sqrt{2}$, $1 + i$, $i * \sqrt{2}$, $-1 + i$...
d) 1 , $-1 - i$, $-i * 2$, $-2 - i * 2$, -4...

Übung 24–4 Führen Sie die folgenden Berechnungen in der kartesischen und in der Polarkoordinatenform durch und vergleichen Sie die Resultate durch gegenseitiges Umwandeln der beiden Formen ineinander.

$2 * i * 2$; $\sqrt{2} * (3 + i * 3)$; $(1 + i) * (-1 + i)$;
$(\sqrt{3} + i) * (-\sqrt{3} - i)$; $i * (-2 - i * 2)$; $-i * 2 * (-4 - 4 * i)$;
$(2 + i * 2)/(-i)$; $1/(4 + i * 3)$; $(1 + i)/(1 - i)$

Übung 24–5 Drücken Sie die folgenden Funktionen bzw. Operationen durch arithmetische Ausdrücke in z und $\overline{z}$ aus, ohne die „Bestandteile" (weder a, b noch r, w) zu verwenden:

Beispiel: $\mathrm{real}(z) = (z + \overline{z})/2$
imag(z) = ? ; $|z| = ?$; arg(z) = ?

Und drücken Sie unter Zuhilfenahme der Gauß'schen Zahlenebene die folgenden Operationen aus:

- Drehen eines zu einer komplexen Zahl gehörenden Vektors um $\pm 90^{\circ}$
- Punktspiegelung einer komplexen Zahl am Koordinatenursprung.

Übung 24–6 Wo (Punkt, Linie, Teilfläche) in der Gauß'schen Zahlenebene befinden sich die Punkte, deren zugehörige komplexe Zahlen jeweils die folgenden Bedingungen erfüllen?

a) $z = \overline{z}$ b) $\mathrm{imag}(z) > 0$ c) $|z| < 2$
d) $\arg(z) = 135^{\circ}$ e) $0 < \arg(z^2) < 180^{\circ}$

Übung 24–7 Bestimmen Sie rechnerisch und grafisch jeweils alle Wurzeln aus (-1) nacheinander für die Wurzelindizes $n = 2, 3, 4, 5, 6$ und prüfen Sie die Resultate für $n < 5$ durch Ausmultiplizieren von jeweils einer der Wurzeln in der arithmetischen Form.

2.2
Allgemeine Übungen zum Kapitel 2

Funktionsplots

20–1 Grafische Demonstration von geraden und ungeraden Funktionen
Schreiben Sie ein Skript-M-File, das eine gerade (bzw. ungerade) Funktion grafisch darstellt. Programmieren Sie darin mit `v=input('Testwert-Eingabe')` oder `[x,y]=ginput(1)` die Abfrage eines x-Wertes. Zeichnen Sie aufgrund dieses x-Wertes die Verbindungslinie zwischen den Punkten (-x/y) (0/y) (x/y) für die gerade Funktion, bzw. (-x/-y) (0/0) (x/y) für die ungerade Funktion.

20–2 Darstellung der Periodizitätseigenschaft einer Funktion
Schreiben Sie ein Skript-M-File, das eine periodische Funktion über mehrere Perioden der Länge 'T' grafisch darstellt. Programmieren Sie darin mit `v=input('Testwert-Eingabe')` die Abfrage eines x-Wertes. Basierend auf diesem x-Wert demonstriert die Verbindungslinie zwischen den Punkten (x / y) und ((x+T) / y) grafisch die Periodizitätsbedingung.

20–3 Gemeinsamer Plot von Sinus und Kosinus
Stellen Sie die Sinus- und Kosinusfunktionen mit verschiedenen Farben in einem gemeinsamen Plot dar.

20–4 Darstellung der Kombination von Sinus und Kosinus
Erstellen Sie ein M-File Skript, das die Sinus- und Kosinusfunktionen in einem gemeinsamen Plot mit verschiedenen Farben darstellt. Anschließend sollen durch zwei interaktive Abfragen der Art `a = input('cos-Faktor?')` die Gewichte `a` und `b` bestimmt werden und in einer dritten Farbe `f3(w) = a*cos(w) + b*sin(w)` dazu gezeichnet werden.

20–5 Allgemeiner Plot mit parametrisierter Funktionswahl
Mit dem untenstehenden, speziellen M-File, einem sogenannten Funktions-M-File kann man die zu zeichnende Funktion als String (Zeichenkettenparameter) in Klammern eingeben z. B. `genfcplot('sin')`.

```
function y = genfcplot(fctnam)
x = 0.01:0.01:5 ;
y = feval(fctnam, x) ;
plot(x,y) ;
```

20–6 Die klassische Funktionskurve „Versiera di Agnesi“
Zeichnen Sie die Funktionskurve der „Versiera di Agnesi“ $y(x) = a^3/(a^2 + x^2)$ für $a = 2$.

20–7 Verschiedene Glockenkurven

Mit verschiedenen Exponenten $p > 1$ erhält man aus der Formel $y = (1 - x^2)^p$ mit $-1 < x < 1$ verschiedene Glockenkurven. Stellen Sie diese in einer gemeinsamen Grafik dar!

Lissajous-Figuren

20–8 Elementare Lissajous-Figuren

Beim Zeichnen von Lissajous-Figuren überstreicht der Laufparameter 't' natürlicherweise den Bereich von 0 bis 2π. Wählen Sie eine Unterteilung mit ca. 500 bis 700 Zwischenwerten (z. B. 628). Zeichnen Sie zuerst verschiedene Varianten der 1:1 Lissajous-Figuren (aufrechte und schiefstehende Ellipsen, sowie doppelt belegte Geraden, nach rechts oben bzw. nach rechts unten).
Wenden Sie sich dann den $(f_x : f_y =) 1 : 2$ Figuren zu (liegende Acht, nach oben und unten offene Parabel) und zum Schluss den 2:1 -Figuren (stehende Acht, nach links und rechts offene Parabel).

20–9 Kompliziertere Lissajous-Figuren

Zeichnen Sie Lissajous-Figuren mit Startphasenverschiebungen von $0°$ und $90°$ für die Verhältnisse 2:3, 3:4 und 5:7, sowie zu diesen ganz leicht phasenverschobene Figuren!

20–10 Parameter zu einer Lissajous-Figur bestimmen

Suchen Sie die Parameter, welche zu einer Lissajous-Figur gehören, mit einer Form, die dem griechischen Buchstaben 'alpha' entspricht, doppelt durchlaufen! Zeichnen Sie mit diesen Parametern die Figur zur Kontrolle. Hinweis: Durchlaufen Sie die Figur zweimal und tabellieren Sie im Ablauf markante Punkte wie maximale/minimale x/y-Werte und Schnittpunkte mit den Achsen, je in einer separaten Spalte für die x- und die y-Funktion.

Spiralen

20–11 Links- und rechtsläufige Spiralen

Erzeugen Sie eine gemeinsame Grafik der beiden archimedischen Spiralen (linksläufig und rechtsläufig), welche durch die Punkte (2/0), (3/0) und (4/0) gehen! Der Plot-Bereich soll in beiden Richtungen das Intervall [-5 ..5] umfassen.

20–12 Archimedische Spirale

Zeichnen Sie eine archimedische Spirale, welche durch die Punkte (x=/y=) (1/0) und (0/3) geht.

20–13 Logarithmische Spirale

Bestimmen Sie die Parameter a , k der logarithmischen Spirale $r(\theta) = a \cdot e^{k \cdot \theta}$ so, dass die Spirale rechtsdrehend durch die Punkte $(1/0)$ $(3/0)$ geht.
Zeichnen Sie die Spirale nachher mit dem Aufruf `polar(t,r)` und nach der Umwandlung in Polarkoordinaten mit `plot(x,y)`!

Kurven in Parameterdarstellung – mathematische Klassiker

20–14 Verschiedene Zykloiden

Erzeugen Sie eine Grafik, in der die normale, gestreckte und die verschlungene Zykloide (Kreis an einer Geraden abrollend, mit verschiedenen Lichtpunktradien) gemeinsam, aber mit verschiedenen Farben dargestellt sind.

20–15 Epizykloide

Entwickeln Sie selbst die Formel für die Epizykloide, bei der ein Kreis vom Radius r=1/2*R außen auf einem Kreis vom Radius R abrollt und zeichnen Sie diese Figur anschließend mit MATLAB.

20–16 Hypozykloiden

Überlegen Sie sich die Formeln für die 2:1 und 4:1 Hypozykloiden, bei denen ein Kreis vom Radius r=1/2 bzw. r=1/4 im Innern eines Kreises vom Radius 1 abrollt. Zeichnen Sie diese Kurven mit MATLAB.

20–17 Evolvente

Entwickeln Sie die Formel für die „Faden-Abwickelkurve“. Auf dem Einheitskreis sei ein Faden aufgewickelt. Am Ende des Fadens sei der Schreibstift, so dass er am Anfang im Punkt (1/0) steht. Die Evolvente ergibt sich als Kurvenverlauf des Schreibstiftes, wenn man den Faden beim Abwickeln immer gespannt lässt. (Tangentenabschnittslänge = bisher abgewickelte Bogenlänge.)

20–18 Kepler-Ellipse

Zeichnen Sie eine Kepler-Ellipse. (Johannes Kepler, 1571–1630, war der Entdecker der genauen mathematischen Form der Planetenbahnen):
$r(\phi) = \dfrac{p}{1 - \varepsilon \cdot \cos(\phi - \phi_0)}$. Mit ϕ_0 bestimmen Sie die Lage der Halbachse, mit ε die Abweichung von einem Kreis. Empfehlung zum Zeichnen $\varepsilon > 0.3$. Mit echten Werten von ε (Erde 0.017, Mars 0.093) sehen Sie keinen Unterschied zu einem Kreis!

20–19 Spline-Interpolation

Benutzen Sie die 5 Punkte der cos-Funktion zu den Winkeln $-1, -0.5, 0, 0.5, 1$ mitsamt deren Ableitungen, um die Funktion in den 4 Intervallen durch kubische Splines zu beschreiben. Wenden sie die Splines auf die Winkelwerte `-1:0.1:1` an und vergleichen Sie die Werte mit der “echten” cos-Funktion.

20–20 MATLAB spline

Verwenden Sie von der Funktion $1/(1 - x^2)$ die 4 Punkte $0, 0.5, 1, 1.5$ als Stützstellen/Stützwerte, um mit der Bibliotheksfunktion spline die interpolierten Werte für `x = 0:0.1:1.5` zu bestimmen. Zeichnen Sie die beiden Funktionen in derselbem Skala übereinander.

20–21 Bezier-Viertelkreis

Zwischen den Punkten $(0/5)$ und $(5/0)$ mit den Kontrollpunkten $(0/a)$ und $(a/0)$ ist

eine Bezierkurve in der Art eines Viertelkreises definiert. Experimentieren Sie mit Werten von a zwischen 1.5 und 3, um die Annäherung an den echten Vierelkreis zu verbessern.

Dreidimensionale Kurven in Parameterdarstellung

20–22 Einfache Schraubenlinien

Durch den Zusatz eines gleichmäßigen Vorschubes in der z-Richtung entsteht aus einer gleichmäßigen Drehbewegung in der x-y-Ebene (Kreislinie als Lissajous-Figur) eine Schraubenlinie. Erzeugen Sie einen solchen Satz von 3 Vektoren x(w), y(w), z(w) und stellen Sie diese Linie mit plot3(x,y,z) in 3D dar!
Überlegen Sie, wie Sie beeinflussen können, ob die Schraubenlinie linksgängig oder rechtsgängig ist.

20–23 Doppelhelix der Erbinformation

Erstellen Sie ein einfaches geometrisches Modell der B-DNA Doppelhelix mit Radius 1 und Ganghöhe 3.6 Angström. Die 10 Basen-Brücken pro volle Drehung sind also je 36 Grad zueinander verdreht. Die zwei Helix-Linien stehen ca. 150 Grad zuenander, sind also nicht ganz vis-a-vis.

20–24 Schraubenlinienpaar bei Parkhausauffahrt

Geben Sie ein M-File an, das die beiden Schraubenlinien des linken und rechten Randes der Fahrbahn einer Parkhausauffahrt mit dem Befehl "plot3(xarray,yarray,zarray)" grafisch darstellt! Parameter: Durchmesser des inneren Randes = 12 Meter, des äußeren Randes 18 Meter. Einfahrtshöhe = 0 m, Stockwerkabstand = 4 m, 3 Parkebenen plus Parterre.

20–25 Spiralbohrer

Bestimmen Sie die Raumkurven der beiden äusseren Schneidekanten eines Spiralbohrers mit 12 mm Durchmesser und 18 mm Ganghöhe.

Folgen und Reihen

20–26 Fundamentalaufgaben zu arithmetischen Folgen

Von den vier Schlüsselgrößen einer arithmetischen Folgea_1, a_n, d, n, erstes und letztes Element, Differenz und Anzahl der Glieder, müssen nur drei bekannt sein, dann kann das vierte berechnet werden. Stellen Sie die Formeln für die vier möglichen Grundaufgaben zusammen.

20–27 Belohnung für die Erfindung des Schachspiels

Nach der Legende soll der Erfinder des Schachspiels seinen König damit sehr erfreut haben, so dass dieser ihm eine große Belohnung versprochen habe. Als der Erfinder die gewünschte Belohnung wie folgt beschreibt, stimmt der König zu, ohne sich über das Ausmaß im Klaren zu sein. Der Erfinder wünscht sich für das erste Feld der Schachbrettes ein Reiskorn, für das zweite zwei, für das dritte vier usw. bis zum 64. Feld, für jedes doppelt so viele Reiskörner wie für das vorhergehende.

Wieviele Reiskörner ergibt diese Berechnung und wieviele Tonnen Reis sind das, wenn man 0.05 Gramm pro Korn annimmt?

20–28 Abzahlungsvertrag
Auf wieviel kann sich der bar zu bezahlende Betrag von 1200 Euro maximal erhöhen, wenn bei der Bezahlung in 12 Monatsraten ohne Anzahlung 15% Jahreszins berechnet wird. Dieser Prozentsatz gilt in einigen Ländern gerade noch nicht als Wucher.

20–29 Rentensparen
Im Verlauf von 30 Jahren soll ein Kapital von 500 000 Euro zusammengespart werden. Wieviel beträgt die monatliche Einzahlung, wenn alle Raten gleich hoch sein sollen und man mit einem Zinssatz von 3% rechnet? Die Verzinsung soll mit Zinseszins und jährlichem Zuschlag erfolgen.

20–30 Abgebrochener Abzahlungsvertrag
Ein Schuldner nimmt einen Kredit über 10 000 Euro auf, den er in 24 gleichen Monatsraten mit einem Zinssatz von 8% pro Jahr abzahlen möchte. Nach dem ersten Jahr, als gerade die 12. Monatsrate fällig ist, erhält er von seiner Firma einen Bonus, so dass er die Restschuld sofort begleichen kann. Wieviel beträgt die Restschuld?

Komplexe Zahlen

20–31 Addition von komplexen Zahlen – grafisch
Programmieren Sie die grafische Darstellung der Addition von zwei komplexen Zahlen in der Gauß'schen Zahlenebene! (Funktions-M-file mit zwei komplexen Zahlen als Eingabe und der Summe als Rückgabewert, sowie der gemeinsamen grafischen Darstellung der drei Zahlen (Vektoren), je in verschiedenen Farben).

20–32 Hilfsfunktion zum Ausdrucken von komplexen Zahlen
Programmieren sie eine Funktion `cplxdisp(z)`, welche von der eingegebnen komplexen Zahl `z` der Reihe nach a,b, r und w ausdruckt.

20–33 Sinus und Kosinus durch $\exp(i \cdot \phi)$ ausdrücken
Durch Bilden der Summe und der Differenz der Euler'sche Identität für positive und negative Winkel: $\mathrm{e}^{\pm i*w} = \cos(w) \pm i * \sin(w)$ erhält man die bekannten Formeln, welche die Funktionen cos(w) und sin(w) durch Exponentialfunktionen von w (bzw. von i*w) auszudrücken. Führen Sie diese Berechnungen selbst aus und kontrollieren Sie Ihr Resultat mit der Formelsammlung!

20–34 Mehrwinkelformeln aus den Potenzen der Euler'schen Identität
Wenn man die Euler'sche Identität potenziert, so erhält man die Moivre'schen Formeln:
$(\mathrm{e}^{i*w})^n = (\cos(w)+i*\sin(w))^n = \cos(n*w)+i*\sin(n*w)$, Benutzen Sie diese sowie die binomischen Formeln, um die bekannten Formeln für die trigonometrischen Funktionen für doppelte und dreifache Winkel selbst herzuleiten.

20–35 Komplexe Gleichung 4. Grades
Suchen Sie alle Lösungen der Gleichung $Z^4 = i$. Beachten Sie die kleinen aber wesentlichen Unterschiede zu den Lösungen des wohlbekannten Problems $Y^4 = 1$.

Grafiken mit komplexen Zahlen

20–36 Polygon im Einheitskreis
Erstellen Sie ein M-File, bei dem mit der Funktion `val=input('Prompt-Text')` die Anzahl Ecken abgefragt wird, und das anschließend das entsprechende reguläre Polygon zeichnet.

20–37 exp(j*w) ist der Einheitskreis
Erzeugen Sie eine Grafik des Einheitskreises in der Gauß'schen Zahlenebene mit der komplexen Exponentialfunktion:

```
w = 2*pi*(0:0.01:1); c = exp(j*w); plot(c)
```

20–38 Pseudozykloiden („Kugelschreiber-Einfahrkurven")
Mit `t = w/2/pi; p = f*t + c; plot(p)` erhalten Sie je nach den Werten des Gewichtsfaktors `f` verschiedene Pseudozykloiden. (`c` ist dabei der komplex definierte Einheitskreis mit mehreren Umgängen (`w = 2*pi*(0:0.01:6); c = exp(j*w)`)

20–39 Mit komplexen Zahlen eine Brezel zeichnen
Experimentieren Sie mit den Grenzwinkelwerten w1 und w2, sowie mit dem Vorschubfaktor k, bis die Figur

```
w = w1:0.01:w2;   z = exp(j*w)-k*w;    plot(z)
```

die Form einer Brezel ergibt.

20–40 Komplexe Spirale
Erzeugen Sie eine Grafik, in der eine Spirale durch die aufeinander folgenden Werte von aufsteigenden Potenzen einer komplexen Zahl definiert wird, also durch 1, z, z^2, z^3 etc. (z. B. $z = 1 + i * 0.05$).
Überlegen Sie sich, wie man zwischen den Punkten interpolieren könnte, da diese Spirale für gewisse Zahlen etwas eckig wird!

2.3
Miniprojekte zur Elementarmathematik

201 Pythagoreische Schnecke, Theodora-Spirale
Beim ersten rechtwinkligen Dreieck haben beiden Katheten die Länge 1. Platzieren Sie dieses im Koordinatensystem so, dass die Ecken bei (0/0), (1/0) und (1/1) liegen. Die Hypotenuse hat dann die Länge $\sqrt{2}$ und zeigt in die 45°-Richtung. Für jedes nachfolgende Dreieck wird die Hypotenuse des vorangegangenen zur neuen Kathete

und am äußeren Punkt wird eine Kathete der Länge 1 angefügt (im rechten Winkel, versteht sich).
Nach dem Satz von Pythagoras erhalten die Katheten die Längen $\sqrt{2}$, $\sqrt{3}$, $\sqrt{4} = 2$, $\sqrt{5}$, $\sqrt{6}$, etc. und die Winkel beim Nullpunkt werden immer kleiner. Erstellen Sie ein MATLAB M-File, das eine solche Serie von Dreiecken zeichnet und probieren Sie aus, wieviele Dreiecke man braucht, um bis zu 360° zu kommen. Freuen Sie sich an der hübschen Figur!
Diese Figur ist seit der Antike als Theodora-Spirale bekannt. Wegen der Folge der Radien heisst sie auch Wurzelschnecke.

202 3D Darstellung eines Moebius-Bandes

Benutzen Sie 3D Kurven in Parameterdarstellung zum Visualisieren eines Moebius-Bandes. Dazu muss eine kurze Strecke, deren Mittelpunkt auf einem Kreis verläuft und die immer in einer radialen Ebene bleibt, zusätzlich eine halbe Drehung (180°) um eine tangentiale Achse pro Umlauf um den Kreis ausführen.

203 Von Geisterhand aufgebaute Vielecke

Erzeugen Sie ein M-File, das einen Film nach dem folgenden Drehbuch ablaufen lässt:
In der ersten Phase wird aus dem Punkt (0/1) heraus ein gleichseitiges Dreieck generiert, das bereits beim Aufbau immer im Einheitskreis einbeschrieben ist. Jede Seite bildet zuerst eine immer länger werdende Sehne, solange bis sie die Seitenlänge des gleichseitigen Dreiecks erreicht hat. Dann bildet sich eine Ecke und die nächste Sehne startet mit der Länge 0 und wird immer länger, während die vorher produzierte Seite (bzw. Seiten) sich weiter dreht (drehen). Dies geht so lange, bis alle drei Sehnen, eine nach der anderen, die richtige Länge erreicht haben; dann ist das gleichseitige Dreieck fertig.
In den folgenden Phasen wird das Dreieck zum Quadrat ergänzt, dann das Quadrat zum regulären 5-Eck und so weiter bis ca. zum Achteck. Jedesmal beim Erweitern werden die bisherigen Seiten kürzer und drehen sich, um der langsam zur Länge der neuen Seite anwachsenden Sehne Platz zu machen.
Das Rückwärtslaufen dieses Films ist ebenso spannend, wenn der Punkt (0/1) die Linie langsam wieder auffrisst.

204 Film einer Welle in Bewegung

Erstellen Sie einen Film, welcher, ausgehend von der Wellengleichung

$$u(x,t) = \hat{u} \cdot \mathrm{e}^{j \cdot (k \cdot x - \omega \cdot t)}$$

die Abfolge der Funktionen $u(x,t)$ in Abhängigkeit von x für die verschiedenen Werte von t, $0 \leq t \leq 2\pi/\omega$, aufeinander folgend zeigt.
Wählen Sie also einen Wert für k, dazugehörig einen Plot-Bereich, so dass $(xmax - xmin) \cdot k$ mindestens 4, eher aber 7–8 ganze Wellenformen ergibt. Dann brauchen Sie nur noch etwa 20 bis 30 Varianten dieser Wellen für verschiedene Zeitpunkte nacheinander zu zeichnen und jeweils die vorhergehende Wellenform zu löschen.

2.4
Selbsttests zum Kapitel 2

Testserie 2.1

T211 **Verständnisfragen**

- Welche Eigenschaften des Kurvenverlaufs einer Kurve in der Ebene sind nur bei Kurven in Parameterdarstellung möglich?
- Geben Sie ein Beispiel für eine Funktion mit der Signatur: (komplexe Eingabe, komplexes Resultat)!
- Bestimmen Sie das Resultat von$(1+i)^3$.
- Welche der untenstehenden Funktionen sind gerade Funktionen?

$$\cos(x),\ \ x^2,\ \ x^3,\ \ x^{(1/2)},\ \ x^{16},\ \ \cosh(x),\ \ \cos(x+0.2)$$

T212 **Rekonstruktion einer Lissajous-Figur**
Rekonstruieren Sie die x- und y-Funktionen, welche zu einer Lissajous-Figur in der Form einer nach oben offenen Parabel führen und schreiben Sie ein MATLAB-Skript zum Zeichnen dieser Figur!

T213 **Normale Zykloide, mit komplexen Zahlen formuliert**
Bestimmen Sie eine komplexe Formel in der Art z = f(t, exp(j*t)), welche die als x-y-Parameterdarstellung mit `x = r*t + r*sin(-t)` und `y = -r*cos(t)` definierte, gewöhnliche Zykloide als komplexe Kurve erzeugt!

T214 **Komplexe Gleichung höheren Grades**
Bestimmen Sie alle Lösungen der Gleichung$z^4 = -i$

T215 **Archimedische Spirale**
Suchen Sie die Gleichung der archimedischen Spirale, welche im Uhrzeigersinn nach außen läuft und durch die Punkte (1/0) und (3/0) geht.
Geben Sie je die MATLAB-Befehle an, um diese in Polarkoordinaten und in kartesischen Koordinaten zu zeichnen.

Testserie 2.2

T221 **Verständnisfragen**

- Welche Bedingungen erfüllen die x- und y-Funktionen einer Kurve in Parameterdarstellung wenn ein Überschneidungspunkt entsteht?
- Geben Sie eine Formel an für die Bestimmung des Realteils und des Imaginärteils einer komplexen Zahl, unter alleiniger Verwendung der Funktion `zc = conj(z)` (= konjugiert komplexe Zahl zu z).
- Welche zusätzliche Bedingung muss eine Relation erfüllen, damit sie zu einer Funktion wird?
- Welche der untenstehenden Funktionen sind ungerade Funktionen?

$$\sin(x),\ \ x^2,\ \ x^3,\ \ \log(x),\ \ x^{15}+1,\ \ \sinh(x),\ \ \cos(x+\pi/2)$$

T222 **Rekonstruktion einer Lissajous-Figur**
Rekonstruieren Sie die x- und y-Funktionen, welche zu einer Lissajous-Figur in der Form einer stehenden, etwas breit geratenen „Acht“ gehören, und schreiben Sie ein MATLAB-Skript zum Zeichnen dieser Figur!

T223 **Schraubenlinien**
Bestimmen sie die Parameterdarstellungen für die innere Rille eines Rechtsgewindes und des entsprechenden Linksgewindes mit je einem Kerndurchmesser von 10.2 mm und einer Ganghöhe von 1.6 mm.

T224 **Komplexe Gleichung höheren Grades**
Bestimmen Sie alle Lösungen der Gleichung $z^{12} = i$

T225) **Logarithmische Spirale**
Suchen Sie die Gleichung der Logarithmischen Spirale, welche im Gegenuhrzeigersinn nach außen läuft und durch die Punkte (1/0) und (2/0) geht.
Geben Sie je die MATLAB-Befehle an, um diese in Polarkoordinaten und in kartesischen Koordinaten zu zeichnen.

3
Übungen zum Kapitel 3

3.1
Eingebettete Übungen zum Kapitel 3

31–1 Beispiel zum Berechnen einer Determinanten

Berechnen Sie die Determinante einer tridiagonalen Matrix der Dimension 5, welche die Werte 2 auf der Diagonalen und -1 auf den Nebendiagonalen aufweist.

31–2 Aufwandsvergleich zwischen Rechenverfahren

Berechnen Sie das Verhältnis zwischen dem Aufwand der zwei verschiedenen Lösungsverfahren Gauß-Elimination und Cramer'sche Regel mit Determinanten für die Dimensionen 4, 6, 8, 10. Gauß:$O(n^3)$ Determinante: $O(n!)$

31–3 Zwei Arten der Determinantenberechnung mit MATLAB

Für eine mit dem Zufallsgenerator erstellte Matrix, z. B.`A=rand(9)` kann in MATLAB mit `det(A)` die Determinante von A berechnet werden. Dies erfolgt natürlich nicht mit der allgemeinen Formel, sondern verwendet eine Dreieckstransformation. Eine solche können Sie in MATLAB mit`[L,R.P] = lu(a)` direkt erhalten. Mit den Befehlen`diagvec= diag(R)` und `De = prod(diagvec)` können sie die Determinante dann als Produkt der Diagonalelemente einer auf Dreiecksform transformierten Matrix berechnen, einigermaßen ähnlich zum Vorgehen, das in MATLAB eingesetzt wird. Damit dieses Verfahren vollständig richtig herauskommt, müssen die in der Prozedur`lu(A)` durchgeführten Zeilenvertauschungen berücksichtigt werden. Diese sind in der zusätzlichen Permutationsmatrix P protokolliert. Je nachdem, ob darin eine gerade oder ungerade Permutation enthalten ist, ergibt sich ein Faktor$+1$ oder -1, mit welchem der Wert`De = prod(diagvec)` multipliziert werden muss. Die Funktion`det(P)` liefert diesen Vorzeichenfaktor.

Übung 32–1 Betrachtet wird die Phasenschieber-Schaltung, die weitgehend analog zu einer Wheatstone'schen Messbrücke aussieht. Die Knoten sind 1=links, 2=oben, 3=unten, 4=rechts. Die Ausgangsspannung ist dabei diejenige zwischen K2 und K3. Die Ohm'schen Widerstände K1K3 und K3K4 sind gleich (z. B. $10 mega$). Zwischen K1 und K2 ist ein Kondensator und zwischen K2 und K4 ein variabler

Ohm'scher Widerstand R_{var}.
Berechnen Sie V_{out} mit MATLAB für eine feste Frequenz, z. B. 200 MHz und für verschiedene R_{var}. Prüfen Sie die vorausgesagte Phasenschiebereigenschaft.

3.2 Allgemeine Übungen zum Kapitel 3

Lineare Abhängigkeit

30–1 Paarweise lineare Unabhängigkeit

Konstruieren Sie zu der nebenstehenden Zweiergruppe von Vektoren einen dritten, so dass je zwei der Vektoren linear unabhängig sind, aber alle drei linear abhängig.
Hinweis: Streben Sie an, durch Linearkombinationen mit Probieren den Nullvektor zu erzeugen. Gelingt dies, so sind die Vektoren linear abhängig, ist dies unmöglich, so sind die Vektoren linear unabhängig.

$$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

30–2 Linear unabhängige Dreiergruppen

Die nebenstehende Gruppe von drei Vektoren ist linear unabhängig. Durch einen speziell gewählten vierten Vektor soll die Vierergruppe linear abhängig sein.
Anders zusammenstellbare Dreiergruppen sollen dagegen unabhängig sein.

$$\begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 1 \\ 2 \\ 3 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

30–3 Lineare Abhängigkeiten bei einer Gruppe von Spaltenvektoren

Suchen Sie unter den untenstehenden Vektoren möglichst große Gruppen, die voneinander linear unabhängig sind! Hinweis: durch Aneinanderfügen von Vektoren können Sie in MATLAB Matrizen erzeugen, deren Rang Sie mit der Funktion rank() bestimmen können.

$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 2 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 1 \\ 2 \\ 3 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Der Rang einer Matrix

30–4 Spaltenrang bei verschiedener Gruppierung

Bestimmen Sie jeweils den Rang aller möglichen Sets von 4 aus den gegebenen Vektoren ausgewählten Vektoren. Die Reihenfolge der Anordnung ist für den Rang un-

wichtig; kontrollieren Sie diese Aussage an zwei bis drei Beispielen.

$$\mathbf{a} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}, \quad \mathbf{c} = \begin{pmatrix} 4 \\ 3 \\ 2 \\ 1 \end{pmatrix}, \quad \mathbf{d} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{e} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix},$$

30–5 Maximaler und minimaler Rang

Suchen Sie Werte für a und b, so dass der Rang der untenstehenden Matrix möglichst groß, bzw. möglichst klein wird.

$$\begin{pmatrix} 1 & 2 & 1 & b \\ 1 & 2 & 2 & 1 \\ 0 & a & 0 & 0 \\ 0 & 0 & 4 & 2 \end{pmatrix}$$

30–6 Linear abhängige/unabhängige Vektoren

Bestimmen Sie für die untenstehenden Vektorgruppen jeweils Wertebereiche für die Parameter p und q, so dass eine Gruppe linear unabhängig, bzw. dass sie möglichst stark linear abhängig ist (Maximalrang=Vollrang, bzw. Minimalrang).

a) $\begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix} \begin{pmatrix} p \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ q \end{pmatrix}$ b) $\begin{pmatrix} 0 \\ p \\ 1 \\ -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 2 \\ 0 \end{pmatrix} \begin{pmatrix} 2 \\ 1 \\ q \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$

Lösung von angewandten Textaufgaben

30–7 Verschiedene Zahlen durch Vertauschen von Ziffern

Lösen Sie das folgende angewandte Gleichungssystem: Eine dreistellige Zahl hat die Quersumme 11. Durch Umstellen der drei verschiedenen Ziffern kann der Zahlenwert verändert werden. Die Differenz zwischen dem größten durch Umstellen erreichbaren Zahlenwert und dem kleinsten beträgt 594, der Unterschied zwischen dem größten und dem zweitgrößten ist 18. Wie lautet die kleinstmögliche Zahl?

30–8 Gleichungssystem aus Ziffernvertauschungen

Eine vierstellige Zahl hat die Quersumme 13. Die Differenz zwischen der größtmöglichen und der kleinstmöglichen Zahl beträgt 6264. Die zwei kleinsten Ziffern sind identisch. Die Differenz zwischen der größten und der kleinsten Ziffer beträgt 6.

30–9 Jahresumsätze

Eine aufstrebende Kleinfirma hat in den ersten 5 Jahren ihres Bestehens mit ihrem Umsatz in jedem Jahr das 1.5-fache des Vorjahresergebnisses um 100 000 Euro übertroffen. Wie groß war der Umsatz im ersten Jahr, wenn der Gesamtumsatz über die 5 Jahre 4 Mio ergab?

30–10 Führungsgehälter

Von 5 höheren Gehaltsstufen verdient jede nächsthöhere Stufe 20% mehr als die darunterliegende und dazu 20 000 Fr mehr pro Jahr. Welches sind die einzelnen Jahresgehälter wenn die Gesamtsumme 1 000 000 beträgt?

30–11 Antiquitätenhändler

Ein Antiquitätenhändler stellte seinen Stand an den Märkten von Aarberg, Burgdorf, Colombier, Düdingen und Erlach auf. Seine gesamten Bruttoeinnahmen aus diesen 5 Wochenenden ergaben 20 000 Euro Die Bruttoeinnahmen in D betrugen das Zehnfache der um die Standgebühr von 500 Fr verminderten Einnahmen von B. Die Bruttoeinnahmen in D entsprachen der Summe der Bruttoeinnahmen von A, B und E. Die hohe Standmiete in C hat sich gelohnt: nach Abzug dieser 1000 Euro blieben ihm in C soviel Einnahmen wie die Bruttoeinnahmen von A, D und E zusammen. Die Differenz zwischen den Bruttoeinnahmen in D und denjenigen in B beträgt das Doppelte der Bruttoeinnahmen in A. Stellen Sie die zugehörige Matrizengleichung auf und lösen Sie diese mit MATLAB!

30–12 Kiestransport

Für den Abtransport des Kiesvolumens eines Aushubs bei einer Großbaustelle wurde der Einsatz von drei großen Lastwagen während 5 mal 8 Stunden geplant. Am Anfang des dritten Tages fällt einer der Lastwagen aus. Am Anfang des 4. Tages kommen als Ersatz für diesen zwei kleinere Lastwagen mit je 80 Prozent der Kapazität des großen. Um wieviele Stunden kann der Auftrag damit früher als geplant fertig werden?

30–13 Pumpleistung

Eine Entwässerungspumpe in einer Baugrube stoppt nach einem kurzen Stromausfall abends um 7 Uhr und startet nicht wieder von selbst. Am anderen Morgen um 7 Uhr ist die Grube gerade vollständig gefüllt. Der Vorarbeiter setzt sie wieder in Betrieb und stellt um 8 Uhr fest, dass erst 1/6 der Grube ausgepumpt ist. Er beschafft eine zweite Pumpe gleichen Typs und kann diese um 9 Uhr in Betrieb setzen. Wann ist die Grube ausgepumpt?

30–14 Uhrzeiger

Bestimmen Sie die Anzahl Überdeckungen zwischen Stunden- und Minutenzeiger einer Uhr an einem Tag, sowie deren Zeitpunkte.

Kirchhoff'sche Netze

30–15 Netzwerk mit 3 quadratischen Maschen

Berechnen Sie die Ströme im untenstehenden Netzwerk (alle R in Ohm):

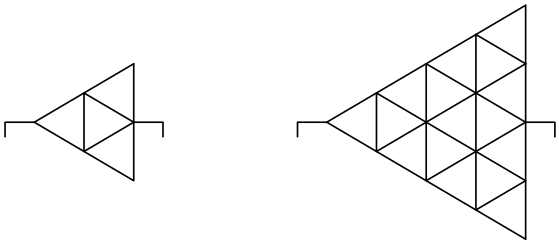
```
*---------------*----100----*----250----*
|               |           |           |
|             200         100         100
|               |           |           |
= 10 V          *----200----*---100-----*
|               |           |
|             100         50
|               |           |
*---------------*----250----*
```

30–16 Einfaches Netzwerk

Ein elektrisches Netzwerk hat 6 Knoten. Die Einspeisung erfolgt zwischen Knoten 1 (+10 V) und Knoten 6 (0 V). Die Verbindung und ihre Widerstände sind: 1-2, 2-3 100 Ω; 1-4 200 Ω; 2-5 400 Ω; 3-6 1 kΩ; 4-5 100 Ω; 5-6 500 Ω. Berechnen Sie den Ersatzwiderstand und die einzelnen Ströme!

30–17 Kirchhoff-Lösung für Dreiecksnetze

Berechnen Sie mit den Kirchhoff'schen Regeln und der Hilfe von MATLAB die Ersatzwiderstände der Dreieckstrukturen mit je 100Ω in den einzelnen Zweigen:



30–18 Skript-M-File für die Wheatstone'sche Messbrücke

Erstellen Sie ein M-File für die Berechnung des Stromes im Messzweig einer Wheatstone'schen Brücke mit 0.01Ω im Galvanometerzweig (= Messzweig) und den allgemein vor der Abarbeitung des M-Files definierten Widerständen R1, R2 (oberer Zweig) und R3, R4 (unterer Zweig). Experimentieren Sie mit verschiedenen Werten von R1 bis R4, um sich von der Abgleicheigenschaft zu überzeugen: I5=0, falls R1:R2 = R3:R4.

30–19 Quadratische Kirchhoff-Netze

Berechnen Sie den Ersatzwiderstand entlang der Diagonalen für ein Widerstandsnetz, das an allen Kanten und Unterteilungslinien eines in 4 (bzw. 9 , 16) Teilquadrate unterteilten Quadrates je einen Widerstand von 1 kΩ aufweist.

Kirchhoff'sche Netze mit stationären Wechselströmen

30–20 Parallelresonanzkreis mit Vorwiderstand

Berechnen Sie die komplexen Ströme im stationären Wechselstromnetzwerk für die Frequenzen ν = 10, 40, 50, 60 und 100 kHz.

```
*---------------*----R=10 Ohm-----*
|               |                  |
|          C=100E-9 Farad       L=1E-3 Henry
|               |                  |
~ 10 V          *------------------*
|               |
|           R=30 Ohm
|               |
*---------------*
```

30–21 Serienresonanzkreis

Erstellen Sie eine MATLAB Funktion für den komplexen Spannungsabfall über dem Widerstand von 10 Ohm, wenn mit diesem eine Spule von 0.2 mHy und ein Kondensator von 0.3 μFarad im Serie geschaltet sind und die am ganzen Kreis anliegende Spannung die Amplitude von 1 V hat. Frequenzbereich 10–100 kHz.

30–22 Phasenschieberschaltung

Erstellen Sie ein M-File für die Berechnung der komplexen Spannung in einer zur Wheatstone'schen Brücke ähnlichen Phasenschieberschaltung. Im Pendant zum Galvanometer, zwischen dem Mittelpunkt der Widerstände R3, R4, je 500 Ohm, und dem Punkt zwischen Z1 = Kondensator mit 50 Picofarad und R2 = Potentiometer mit 50 bis 2000 Ohm wird die Ausgangsspannung abgegriffen. Berechnen Sie U(A-B) für den Widerstandsbereich des Potentiometers und die Frequenz 10 kHz. Stellen Sie angle(A-B) gegen R2 und zur Kontrolle auch abs(U(A-B)) gegen R2 grafisch dar.

Funktions-M-Files

30–23 Programmieren einer Funktion zum Testen der Orthogonalität

Falls für alle Zeilenvektoren / Spaltenvektoren einer Matrix Q gilt:
$v_j \cdot v_k = 0$ für $j \neq k$ und $= 1$ für $j = k$. – dann ist die Matrix Q orthogonal.
Schreiben Sie ein MATLAB-Programm, Eingabe Q, Ausgabe 1, falls orthogonal und 0 sonst.

30–24 Selbst die Transpositions-Funktion programmieren

Schreiben Sie ein MATLAB-Programm, Eingabe quadratisches M, Ausgabe M^T.
Das Programm soll Elementweise mit Hilfe einer Doppelschleife arbeiten. Achtung ein Austausch von zwei Elementen braucht einen Zwischenspeicherplatz. Damit Ihr Programm für beliebige quadratische Matrizen funktioniert, verwenden Sie intern die Funktion [n,m] = size(M) und testen Sie vorgängig auf `m==n`.

Orthogonale Matrizen

30–25 Turmmatrizen sind orthogonal

Quadratische (nxn), sogenannte „Turmmatrizen“ haben in jeder Zeile und in jeder Kolonne genau eine Eins und sonst lauter Nullen. (So könnte man Türme auf ein Schachbrett stellen, die sich gegenseitig nicht bedrohen.) Diese Matrizen vertauschen die Reihenfolge der Zeilen von Matrizen die damit multipliziert werden.
Erzeugen Sie einige Beispiele von solchen Matrizen und überzeugen Sie sich davon, dass: - $T * T^T = I$ ist; - alle T orthogonal sind. Das Produkt einer orthogonalen Matrix mit Ihrer Transponierten ergibt die Einheitsmatrix.

30–26 Potenzen von Turmmatrizen

Potenzieren Sie die in der vorherigen Übung erstellten Turmmatrizen mit ganzzahligen Exponenten. Welches ist der kleinste Exponent, bei welchem $T^n = I$ wird?
Dieser kleinste Exponent kann nie größer als die Dimension sein!

30–27 Orthogonalitätstest

Prüfen Sie nach, dass die untenstehenden 3D-Abbildungsmatrizen die Orthogonalitätsbedingung $A^T = A^{-1}$ erfüllen:

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(w) & -\sin(w) \\ 0 & \sin(w) & \cos(w) \end{pmatrix}, \quad R_y = \begin{pmatrix} \cos(w) & 0 & -\sin(w) \\ 0 & 1 & 0 \\ \sin(w) & 0 & \cos(w) \end{pmatrix}$$

$$R_z = \begin{pmatrix} \cos(w) & -\sin(w) & 0 \\ \sin(w) & \cos(w) & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad SP_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

Gauß-Algorithmus und L-R-Zerlegung

30–28 Showtime Gauß-Algorithmus und L-R-Zerlegung

Sehen Sie sich die Demo-Programme showgauss.m und showlr.m für verschiedene Matrizen und rechte Seiten an, und zwar immer für eine Matrix M und einen Vektor b zuerst `showgauss(M,b)` und anschließend `showlr(M)`.
Die beiden Files sind auf der Website zum Buch abrufbar.

30–29 Dirty Prototype

Erstellen Sie eine MATLAB-Funktion, welche in einer Matrix in derselben Reihenfolge wie der Gauß-Algorithmus Nullen einsetzt. Durch das einfache Überschreiben mit Nullen wird natürlich kein Gleichungssystem gelöst, aber die richtige Schleifen-Organisation kann auf einfache Weise kontrolliert werden.

30–30 Pivot-Strategie

Erstellen Sie ein MATLAB-Script, welches für eine bereits teilweise in R-Form überführte Matrix in der Spalte k die möglichen Pivot-Elemente vergleicht, das betragsmäßig Größte sucht und die entsprechenden Teilzeilen miteinander vertauscht (=Zwischenschritt am Anfang jeder Spaltenverarbeitung).

30–31 Spezielles Rückwärtseinsetzen

Programmieren Sie die Lösung durch Rückwärtseinsetzen für ein Gleichungssystem $R * x = b$. Vom Gleichungssystem wird die spezielle Eigenschaft vorausgesetzt, dass die Rechts-Dreiecksmatrix R nur in der Diagonalen und in der direkt zur Diagonalen benachbarten Linie (d.h. der oberen Nebendiagonalen) Elemente mit Werten verschieden von Null aufweist.

30–32 L-Teilmatrizen in der L-R-Zerlegung

Führen Sie die einzelnen Schritte der Gauß-Elimination für die untenstehende Matrix mit Bleistift und Papier aus. Erzeugen Sie aus den Kombinationskoeffizienten die Teilmatrizen L1 und L2, so dass gilt:
1. Schritt: $\mathbf{A1} = \mathbf{L1} \cdot \mathbf{A}$, 2. Schritt: $\mathbf{R} = \mathbf{A2} = \mathbf{L2} \cdot \mathbf{A1}$
insgesamt: $\mathbf{R} = \mathbf{L2} \cdot \mathbf{L1} \cdot \mathbf{A}$
Bilden Sie anschließend mit MATLAB die Matrizen $L1^{-1}$ und $L2^{-1}$ und zeigen Sie, dass die von der Bibliotheksprozedur `[L,R,P] = lu(A)` gelieferte Matrix L dem Produkt $L1^{-1} \cdot L2^{-1}$ entspricht!

$$\mathbf{A} = \begin{pmatrix} 4 & 4 & 8 \\ 2 & -3 & 4 \\ 1 & 2 & 1 \end{pmatrix}$$

30–33 Bestandteil einer L-R-Zerlegung

Bestimmen Sie in der untenstehenden Matrizengleichung die Parameter p und q so, dass die Elemente c_{21} und c_{31} der Resultatmatrix $\mathbf{C}$ Null werden!

$$\mathbf{C} = \begin{pmatrix} 1 & 0 & 0 \\ p & 1 & 0 \\ q & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 3 & 4 \\ 3 & 1 & 1 \\ 5 & 0 & 1 \end{pmatrix}$$

30–34 Rückschlüsse aus der L-R-Zerlegung

Eine 3x3 R-Matrix hat die Werte `[ 1 2 3 ; 0 4 5 ; 0 0 6 ]`. Die Multiplikationsfaktoren in der Gauß-Elimination waren von oben nach unten in der ersten Spalte 0.5 und -1, in der zweiten Spalte -0.4. Wie lautete die Originalmatrix A?

30–35 L-R-Zerlegung mit Bestimmung der Inversen

Bestimmen Sie die zwei Zerlegungsmatrizen L und R zur untenstehenden Matrix A mit der Bibliotheksfunktion `[L,R,P] = lu(A)`. Bestimmen Sie anschließend die drei Lösungsvektoren zu den rechten Seiten [1 0 0]', [0 1 0]', [0 0 1]' durch Vorwärts-/Rückwärtseinsetzen in L und R von Hand. Damit haben Sie die Spaltenvektoren von A^{-1} berechnet. Testen Sie dies durch die Multiplikation $A^{-1} \cdot A$

$$\mathbf{A} = \begin{pmatrix} 4 & 8 & 4 \\ 2 & 2 & 4 \\ 1 & 3 & -1 \end{pmatrix}$$

Singuläre Systeme

30–36 Partikuläre und homogene Lösung

Bestimmen Sie für das Gleichungssystem$mathbf A \cdot x = b$ eine partikuläre Lösung und zwei Vektoren des Nullraumes.

$$\mathbf{A} = \begin{pmatrix} 2 & 1 & 2 & 4 \\ 0 & 4 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 6 \\ 8 \\ 0 \\ 00 \end{pmatrix}$$

30–37 Volle Pivotstrategie

Führen Sie den Gauss-Algorithmus mit voller Pivotstrategie mit der folgenden einfachen singulären Matrix aus und erleben Sie das Verschieben der Nullzeilen an den unteren Rand.

```
    [  3  2  1 ]
A = [  1  1  1 ]
    [  2  1  0 ]
```

3.3
Miniprojekte zur linearen Algebra

301 Allgemeines rechteckiges Widerstands-Netz

In einem Widerstands-Netz, das rein rechteckig ist, und n mal m quadratische Zellen aufweist, sollen die Widerstände in den horizontalen Linien in der Matrix H und diejenigen in den vertikalen Linien in der Matrix V vorgegeben werden. Erzeugen Sie ein MATLAB-M-File, das den Ersatzwiderstand berechnet bei gegebenen Werten von n und m und in H und V gespeicherten Widerstands-Werten. (Zuerst müssen Sie die richtigen Dimensionen von H und V bestimmen).

302 Einfache und vollständige Pivot-Strategie

Programmieren Sie in MATLAB den Gauß-Algorithmus mit vollständiger Pivot-Strategie. Bei dieser Variante wird das gesamte verbleibende Rechteck nach dem Element mit dem größten Betrag abgesucht. Dieses wird dann durch Vertauschung von Zeilen und Spalten zum Pivot-Element gemacht. Achtung! Beim Vertauschen von Spalten muss die Reihenfolge der Unbekannten in derselben Weise vertauscht werden.

Entwickeln Sie eine effizientes Verfahren zum Abspeichern aller im Berechnungsablauf vorgenommenen Spaltenvertauschungen und zum Rückgängigmachen derselben am Schluss der Berechnungen.

Vergleichen Sie für einige Gleichungssysteme die Resultate dieses Verfahrens mit demjenigen mit reiner Zeilenvertauschung. Zum Erzeugen von nahezu singulären Matrizen kann man einzelne Zeilen in zwei- oder mehrfach kopieren und dann die Kopien gegeneinander nur leicht variieren.

303 Wirkung der Asymmetrie bei Eigenwertproblemen

Erzeugen Sie beliebige symmetrische Matrizen durch `A=rand(ndim) ; S = A+A'`. Vergewissern Sie sich, dass deren Eigenwerte alle verschieden sind.

Ändern Sie nun ein einzelnes Außendiagonalelement in mehreren Stufen, um eine Asymmetrie einzuführen und beobachten Sie die Änderungen der Eigenwerte.

3.4 Selbsttests zum Kapitel 3

Jede von diesen Serien sollten in ca. 60 Minuten gelöst werden können.

Testserie 3.1

T311 **Verständnisfragen**

- Welches ist der maximale mögliche Rang einer 4x7 Matrix?
- Wie kann man sofort beweisen, dass eine Gruppe von Vektoren, welche den Nullvektor enthält, linear abhängig ist?
- Wie groß ist die Dimension des Nullraumes für eine reguläre Matrix?
- Wie kann man ein Gleichungssystem, das eine obere Dreiecksmatrix enthält, direkt lösen?

T312) **Schleifenprogrammierung**
Erstellen Sie ein Skript-M-File, welches für beliebig vorgegebenes 'n' eine Einheitsmatrix (durch Ausradieren) erzeugt, indem in einer Matrix mit lauter Einsen (erzeugt mit ones(n)) die Außen-Diagonalelemente in Doppelschleifen Null gesetzt werden.

T313 **Zeilenpermutationen**
Erzeugen Sie die beiden 6x6 Matrizen, welche durch Multiplizieren von links die Zeilen 1, 3, 5 einer beliebigen Matrix in a) die Zeilen 3, 5, 1 und b) in die Zeilen 5, 3, 1 platzieren und dabei die Zeilen 2, 4, 6 an ihrem Ort lassen.

T314 **Winkelberechnung**
Schreiben Sie ein Funktions-M-File, welches den Winkel zwischen zwei eingegebenen vierdimensionalen Vektoren bestimmt!

T315 **Polynombestimmung**
Stellen Sie das Gleichungssystem auf, welches die Koeffizienten des folgenden Polynoms bestimmt:
Ein Polynom 4. Grades geht durch die Punkte $(-1/0)$ $(0/0)$ und $(2/2)$ und hat im Punkt $(1/1)$ eine horizontale Tangente.

Testserie 3.2

T321 **Verständnisfragen**

- Wie erhält man zu einem gegebenen Gleichungssystem das zugehörige homogene Gleichungssystem?
- Beschreiben Sie Bildraum und Nullraum für die Matrix
  `A= [ 1 0 1 ; 0 0 0 ; 0 1 -1]`.
- Welches ist der Rang einer Turmmatrix und welches derjenige einer Spechtmatrix?
- Wie nennt man eine Matrix für welche gilt:$A^T = A^{-1}$?

T322 **Schleifenprogrammierung**
Erstellen Sie ein Skript-M-File, welches für beliebig vorgegebenes 'n' eine Matrix erzeugt mit den Werten 2 auf der Diagonalen und den Werten '-1' auf den beiden Parallelen zur Diagonalen im Abstand 2. Auf den Nebendiagonalen und außerhalb der '-1'-Linie sind die Werte Null.

T323 **Test auf Antisymmetrie**
Schreiben Sie ein Funktions-M-File, welches eine Matrix als Eingabe verwendet und 1 zurückgibt, falls diese Matrix antisymmetrisch ist, und Null im anderen Fall.

T324 **Projektionsmatrizen**
Geben Sie die drei Projektionsmatrizen an, welche die Vektoren des dreidimensionalen Raumes auf die x-Achse, die y-Achse und die z-Achse projizieren! Welchen Rang haben diese Matrizen?

T325 **Betrag eines n-dimensionalen Vektors**
Schreiben Sie ein Funktions-M-File, welches einen n-dimensionalen Vektor als Eingabe verwendet und daraus dessen Betrag bestimmt. (Ohne Verwendung der Bibliotheksfunktion norm(v). Die Funktionen length(v), size(v), sqrt(s) sind hingegen ausdrücklich erlaubt.)

4
Übungen zum Kapitel 4

Winkelbestimmungen

40–1 Orthogonaler Dreispitz

Von einem Dreispitz (dreiseitige Pyramide) mit den Bodenpunkten $(-1/2; \sqrt{3}/2)$, $(0; 0)$ und $(1/2; \sqrt{3}/2)$ wird die Höhe gesucht, so dass der Winkel im Raum zwischen je zwei der drei Kanten 90° ist. Wie groß ist dann die wahre Kantenlänge?

40–2 Berechnung der wahren Dachneigung

Ein Turm mit der Dachform einer vierseitigen Pyramide hat einen quadratischen Grundriss von 12 Meter Seitenlänge. Die Spitze des Turmes liegt 10 Meter über der unteren Dachkante. Wie groß ist der Neigungswinkel der Dachflächen (gegenüber der Horizontalen)? Berechnen Sie auch den wahren Winkel zwischen zwei aneinander angrenzenden Dachflächen.

40–3 Winkelberechnung am Würfel

Berechnen Sie die Winkel zwischen der Körperdiagonalen, welche die Punkte (0/0/0) und (1/1/1) im 3D-Einheitswürfel verbindet, und den 6 verschiedenen Richtungen von Flächendiagonalen!

Vektorgeometrie in der Ebene

40–4 Hesse'sche Normalform in der Ebene

Stellvertretend für die Gleichung einer Ebene im Raum soll die Gerade in der Ebene, welche durch die Punkte A=(4 /0) und B=(0/3) geht, in der Hesse'schen Normalform dargestellt werden. Dies erreicht man in folgenden Schritten:

- der senkrecht auf der Geraden stehende Vektor (Normalenvektor) ist zu bestimmen. Empfohlenes Vorzeichen: positiv im I. Quadranten.
- Normierung zu Einheitsvektor e_n
- Ansatz der Geradengleichung in der Hesse'schen Normalform: $e_n \cdot OP - e_n \cdot OA = 0$.

Testen Sie Punkte A und B und einen weiteren selbstgewählten Punkt auf Ihre Abstandswerte zu dieser Geraden.

40–5 Visualisierung der Hesse'schen Normalform

Erstellen Sie ein MATLAB-Skript, an das Sie mit dem Befehl `x = input('x-Wert')` bzw. `y = input('y-Wert')` zwei Koordinaten übergeben. Diese sollen anschließend als Originalvektor OP in der Ebene eingezeichnet werden. Ferner sollen die Projektion von OP auf $e_n = [0.8\ 0.6]'$ sowie die Gerade $e_n \cdot OP - 4.8 = 0$ (geht durch A= (6/0) und B=(0/8) eingezeichnet werden.
Als Zahlenausgabe sollen Sie den Ausdruck $d = e_n \cdot OP - 4.8$ auswerten, der genau dann Null wird, falls der eingegebene Punkt auf der Geraden liegt. (Dann liegt auch der Endpunkt des projizierten Vektors auf der Geraden.)
Probieren Sie mit den eingegebenen Punkten die Gerade zu treffen.
Aussage der Hesse'schen Normalform: Eine Gerade (Ebene) umfasst alle Punkte, deren Ortsvektoren bei Projektion auf den Normaleneinheitsvektor dieselbe Länge 'd' ergeben.

Vektorgeometrie in 3D

40–6 Normalenvektoren im Tetraeder:

Bestimmen Sie die vier Einheitsnormalenvektoren zu den Flächen eines regulären Tetraeders, welcher mit der Grundfläche auf der x-y-Ebene steht und eine Kante parallel zur x-Achse hat.

40–7 Gerade durch einen Oktaeder:

Ein Oktaeder hat die Ecken A=(4/0/0), B=(0/4/0) , C=(-4/0/0), D=(0/-4/0), E=(0/0/4), F=(0/0/-4). Zeichnen Sie diesen Oktaeder mit plot3() in 3D, indem Sie die Punkte zu einem Linienzug zusammenstellen (einzelne Kanten werden mehrfach durchlaufen). Bestimmen Sie die Gerade (Geradengleichung in Punkt–Richtungs-Form), welche den Mittelpunkt der Strecke DE mit dem Schwerpunkt des Dreiecks ABE verbindet und suchen Sie den Durchstoßpunkt dieser Geraden durch die x-y-Ebene! Zeichnen Sie diese Gerade mit den drei speziellen Punkten in dieselbe Grafik ein!

40–8 Vierseitige Pyramide

Eine Pyramide mit gleichseitigen Dreiecken als Seitenflächen mit der Kantenlänge 10 m hat ihre 4 Ecken genau nach Norden, Osten, Süden und Westen orientiert. An einem Sommertag steht die Sonne 15° über dem Horizont, wenn sie genau im Westen steht. Wie weit ist der Schatten der Pyramidenspitze vom Zentrum der Grundfläche entfernt, und welchen Winkel bilden dann die Schattengrenzen?

40–9 Ebenes Viereck im Raum

Suchen Sie die x-Koordinate des Punktes D=(x/0/0) so, dass er mit den drei Punkten A=(0/0/4), B=(-4/3/3) und C=(0/4/0) in derselben Ebene liegt. Berechnen sie anschließend den Schwerpunkt des so entstandenen ebenen Vierecks (als gewichtetes Mittel aus zwei Dreiecksschwerpunkten mit den Dreiecksflächen als Gewichte).

40–10 Ebene in Hesse'scher Normalform

Bestimmen Sie die Hesse'sche Normalform der Ebene durch die Punkte A=(3/0/0)

, B=(0/4/0), C=(0/0/3.2) und berechnen sie den Abstand des Koordinatenursprungs von dieser Ebene!

Ausflug in höhere Dimensionen

40–11 Winkel im vierdimensionalen Raum:

Berechnen Sie die Winkel zwischen den vierdimensionalen Vektoren

```
k1=[1 0 0 0]' , k3=[0 0 1 0]' ,   f =[1 1 0 0]' ,
v =[1 1 1 0]'  ,   r =[1 1 1 1]'.
```

Schreiben Sie ein M-File zur Winkelberechnung.

40–12 Winkel zwischen mehrdimensionalen Vektoren:

Berechnen Sie die Winkel zwischen allen Kombinationen der Vektoren k, f, r, v

```
k = [ 1 0 0 0 0]'      f = [ 1 1 0 0 0]'
r = [ 0 0 1 1 1]'     v = [ 1 1 1 1 1]'.
```

40–13 Zwei, drei und vierdimensionaler Einheits-„Würfel“

– Ein „Würfel“ in 2D hat die Ecken:

```
(0/0),     (0/1),    (1/0),   (1/1)
```

Daraus ergeben sich die Kantenvektoren `[0 1]'` und `[1 0]'`, je 2-fach. (Alle möglichen Differenzen zwischen je 2 Ecken.)
Bestimmen Sie die Diagonalenvektoren!

– Der echte Würfel in 3D hat die Ecken:

```
(0/0/0),    (0/0/1),    (0/1/0),    (0/1/1),
(1/0/0),    (1/0/1),    (1/1/0),    (1/1/1)
```

Bestimmen Sie die Kantenvektoren und deren Vielfachheit.
Suchen Sie die 2 verschiedenen Typen von Diagonalenvektoren! Bestimmen Sie deren Länge, sowie deren Winkel (je 3 Richtungskosinuswerte bezüglich der Koordinatenachsen).

– Ein „Würfel“ in 4D, also 4D-Hypercube hat die Ecken:

```
(0 0 0 0),    (0 0 0 1),    (0 0 1 0),    (0 0 1 1)
(0 1 0 0),    (0 1 0 1),    (0 1 1 0),    (0 1 1 1)
(1 0 0 0),    (1 0 0 1),    (1 0 1 0),    (1 0 1 1)
(1 1 0 0),    (1 1 0 1),    (1 1 1 0),    (1 1 1 1)
```

Suchen Sie auch für diesen Fall die Kantenvektoren mit deren Vielfachheit, sowie die (hier drei Typen von) Diagonalenvektoren mit deren Länge und deren (je 4) Winkeln!

Durch Matrizen definierte Abbildungen

40–14 Kongruente Abbildungen der Ebene Gegeben ist ein „L“ durch die Koordinaten (5/2), (5/0) und (6/0). Gesucht werden die 2x2 Matrizen, die Bildkoordinaten und ein gemeinsamer Plot von Urbild und Bild für die folgenden Abbildungen:

A Spiegeln der L an y-Achse
P Punktspiegelung an Koordinatenursprung (0/0)
R Drehungen um 45 und 90 Grad.

Homogene Koordinatentransformationen in der Ebene

40–15 2D homogene Koordinatentransformationen des „L“ Das bereits bekannte „L“, definiert durch die drei Punkte (5/2), (5/0) und (6/0), soll in homogene Koordinaten umgeschrieben werden, und die Ortsvektoren der Dimension 3x1 zu einer 3x3 Punktematrix zusammengestellt werden.
Gesucht sind die speziellen 3x3 Matrizen der homogenen Koordinaten, die Bildkoordinaten und ebenfalls ein gemeinsamer Plot von Bild und Urbild für die folgenden Abbildungen:

a) Verschiebung der L-Ecke nach a_1)(0/5), a_2)(0/0) , a_3) (-5/0)
b) Drehung um 90 Grad um Ursprung
c) Drehung um 45, sowie 90 Grad um die eigene Ecke des „L“

40–16 Drehungen und Translationen des „L“
Das Übungs- „L“ (5/2) (5/0) (6/0) soll 3 mal um je 45° um den Koordinatenursprung gedreht werden.
An den neuen Koordinaten sollen die gedrehten „L“ 's um ihre Ecke um -45°, -90°, -135° gedreht werden, so dass alle am neuen Ort wieder aufrecht stehen.
Wenn Sie die ursprüngliche Drehung, und dann von links die Drehung um die Ecke in homogenen Koordinaten zusammenmultiplizieren, so werden nur noch Translationen übrigbleiben.

40–17 2D-Abbildung des „L“ mit Gegenrotation
Schreiben Sie ein Skript zur Erzeugung und anschließenden Anwendung der Transformationsmatrix in homogenen Koordinaten, welche den Eckpunkt des „L“ um einen vorgegebenen Winkel um den Ursprung dreht und gleichzeitig die „L“-Figur in entgegengesetzter Richtung um ihre Ecke um denselben Winkel dreht.

40–18 2D-Bewegung des „L“ mit allgemeinem Winkelparameter Die Ecke des „L“, definiert durch die drei Punkte (5/2), (5/0) und (6/0) soll um den allgemeinen Winkelparameter 'w' auf einem Kreis (mit dem Radius 5) bewegt werden. Dabei soll

a) Die Orientierung des „L“ (vertikal) bei der Bewegung erhalten bleiben.
b) Das „L“ eine Drehung um den Winkel -w (Gegendrehung mit gleichem Winkelbetrag) ausführen.

Bestimmen Sie für die beiden Fälle die homogenen Transformationsmatrizen, je als

Funktion des Winkels 'w' und testen Sie diese Abbildungen mit einem Plot des abgebildeten „L“!

40–19 Dreiecksabbildung auf sich selbst:
Gegeben Sei ein gleichseitiges Dreieck A$(0/0)$, B$(10/0)$, C$(5/5\sqrt{3})$.
Gesucht sind die zwei Transformationsmatrizen in homogenen Koordinaten, welche das Dreieck auf sich selbst abbilden und zwar

a) A' = B, B' = C, C' = A und
b) A" = C, B" = A, C" = B.

Zeigen Sie dass $T_{a)} = T_{b)}^{-1}$ ist!

40–20 Mehrfache Abbildung eines Rechtecks
Das Rechteck $ABCD$, $A = (0/0)$, $B = (10/0)$, $C = (10/2)$, $D = (0/2)$ soll um den Punkt D um 90° gedreht werden. Das einmal gedrehte Rechteck $A_1B_1C_1D_1$ soll anschliessend um den Punkt B_1 ebenfalls um 90° gedreht werden, dann $A_2B_2C_2D_2$ um D_2. So bilden die vier Rechtecke ein Quadrat mit Seitenlänge 12 um einem Innenhof mit Seitenlänge 8. Führen Sie diese Abbildungen in homogenen Koordinaten aus und stellen sie grafisch dar. Zeigen Sie, dass, zusammen mit einer vierten Rotation um B_3, die aus allen vier Gesamttransformationen zusammengesetzte Transformationsmatrix die Identität ergibt ($A_4B_4C_4D_4$ ist wieder $ABCD$).

40–21 Zwei verschiedene Abbildungen überdecken sich
Das Rechteck $ABCD$, $A = (0/0)$, $B = (8/0)$, $C = (8/4)$, $D = (0/4)$ wird einmal an der Geraden CD gespiegelt und als zweite Abbildung um den Mittelpunkt der Strecke CD um 180° gedreht. Führen Sie diese beiden Transformationen in homogenen Koordinaten durch. Zeigen Sie in einer Grafik, dass sich die beiden Bilder überdecken. Die Überdeckung erfolge jedoch nicht Punkt für Punkt, deshalb gibt es eine Korrespondenz-Tabelle der Art $A_1 = B_2$ etc.

Homogene Koordinatentransformationen im Raum
40–22 3D-Abbildung eines Dreibeins
Das Dreibein mit den Ecken A=(0/0/0), B=(10/0/0), C(0/10/0) und D=(0/0/10) soll einer Punktspiegelung am Punkt (5/5/5) unterworfen werden. Der 3x3-Teil für die Punktspiegelung am Ursprung ist (`-eye(3)`).

40–23 3D Abbildung zwischen den Stufen einer Wendeltreppe
Für eine Wendeltreppe wird verinfachen angenommen, sie sei aus quaderförmigegen Blöcken der Dimension 100x50x18 cm gebaut. Der Innenradius sei 50 cm. Jede der 7 Stufen hat gegenüber der vorherigen eine Verdrehung von 15°. Bestimmen Sie die Transformation in homogenen Koordinaten, die von einem Block zum nächsten führt und erstellen Sie eine 3D Figur der Treppe.

40–24 Sich expandierender Würfel
Ein Würfel mit den Ecken $A = (-2/-2/-2)$, $B = (2/-2/-2)$, $C = (2/2/-2)$,

$E = (-2/-2/2)$, etc. soll an seinen 8 Ecken je durch eine räumliche Punktspiegelung nach aussen transformiert werden. Erstellen Sie eine 3D Grafik dieses Gebildes, das von den 27 Würfelplätzen im dreifach erweiterten Volumen nur 9 Plätze ausfüllt!

40–25 3D: "L" dreht sich vor dem Spiegel:
Ein 2 Meter grosses "L" mit 1 Meter Fusslänge steht aufrecht beim Punkt`[2 -4 0]'` und dreht sich um die z-Achse. Die x-z-Ebene sei ein Spiegel mit den Ecken (0/0/0) (4/0/0) (4/0/3) (0/0/3). Zeichnen Sie die 3D-Parallelperspektive von „L", seinem Spiegelbild und den Spiegelrändern von einem Punkt im Azimut im Bereich 290° bis 310° aus. Der Betrachtungspunkt soll etwas oberhalb der x-y-Ebene liegen.

40–26 Fünf Sterne in 3 Dimensionen
Die Eckpunkte eines Würfels mit Kantenlänge 2 ABCD-EFGH:

```
[-1 -1 -1]',[1 -1 -1]',[1 1 -1]',[-1 1 -1]',\\{}
[-1 -1 1]',[1 -1 1]',[1 1 1]',[-1 1 1]'
```

werden von jeder Fläche aus mit einer darüberliegenden Spitze TWNSOU

```
[6 0 0]', [-6 0 0]', [0 6 0]', [0 -6 0]', [0 0 6]',[0 0 -6]'
```

zu einer Pyramide verbunden. Dies ergibt einen 3D-Stern mit 6 Spitzen. Suchen Sie eine Punktfolge dieser 14 Punkte, so dass alle Kanten durchlaufen werden. Erzeugen Sie die dazugehörige Vektorzusammenstellungsmatrix, um diesen Stern in einem Plot-Aufruf zu zeichnen. Durch homogene Koordinatentransformationen im 3D soll dieser Stern je in x- und y-Richtung um 15 cm verschoben kopiert werden. Erzeugen Sie anschließend ein gedrehtes Bild dieser Anordnung von 5 Sternen und ändern Sie die Betrachtungswinkel.

40–27 Ansicht aus dem Helikopter

Die folgenden Befehle definieren in MATLAB die Eckpunkte und einen Linienzug, der ein einfaches Haus zeichnet (in homogenen Koordinaten).

```
v1 = [ 3 -2 0  1]' ;
v2 = [ 3 -2 3  1]' ;
v3 = [ 3  0 5  1]' ;
v4 = [ 3  2 3  1]' ;
v5 = [ 3  2 0  1]' ;

v6 = [-3 -2 0  1]' ;
v7 = [-3 -2 3  1]' ;
v8 = [-3  0 5  1]' ;
v9 = [-3  2 3  1]' ;
v10= [-3  2 0  1]' ;

H  = [ v1 v6 v7 v2 v3 v8 v9 v4 v5 v1 ...
~~~~  v2 v3 v4 v5 v10 v6 v7 v8 v9 v10 ];

lxi = H(1,:) ;   lyi = H(2,:) ;   lzi = H(3,:) ;
```

Wenden Sie zuerst eine 3D-Rotation in homogenen Koordinaten um die z-Achse um den (beliebigen) Winkel w an. Anschließend soll eine Rotation um den Winkel -t (negativ, weniger als 25 Grad) um die neue y'-Achse erfolgen, um das Haus vom Helikopter aus zu sehen! Achtung! Winkel von Grad in Radiant umrechnen.
Die Darstellung mit
`plot3(Hrot(1,:), Hrot(2,:), Hrot(2,:))` kann analog zum 2-dimensionalen Fall mit
`axis ([ xmin, xmax, ymin, ymax, zmin, zmax])` und `axis square` ergänzt werden.

4.1
Miniprojekte zu Raumgeometrie und Abbildungen

401 Generieren der fünf Platonischen Körper

Die fünf Platonischen Körper sind die einzigen Körper im dreidimensionalen Raum, welche von lauter gleichen regulären Polygonen begrenzt sind und bei welchen an jeder Ecke gleichviele Flächen aneinanderstoßen. Tetraeder, Oktaeder und Ikosaeder sind von gleichseitigen Dreiecken begrenzt, mit 3, 4 bzw. 5 Dreiecken die in jeder Ecke zusammenstoßen. Der Würfel, auch Hexaeder genannt, besteht aus Quadraten von denen je drei an jeder Ecke zusammenstoßen. Beim Dodekaeder (auch Pentagon-Dodekaeder) stoßen je drei Fünfecke in einer Ecke zusammen.

Durch eine Drehung um die y-Achse um den Winkel unter dem eine Kante vom Zentrum der umspannenden Kugel gesehen wird, ist eine Kante festlegbar zwischen dem Bild des höchsten Punktes auf der Kugel und dem aktuellen höchsten Punkt.

Durch Drehen um die z-Achse um 1/3, 1/4 bzw. 1/5 Umdrehung können weitere Kanten festgelegt werden. Durch Hinzunehmen einer Drehung um 180° um die z-Achse können alle fünf Polyeder sukzessive durch Drehungen und Hinzufügen des jeweils obersten Punktes aufgebaut werden.

402 Suchen der Großkreis-Flugrouten

Zwischen zwei Orten auf der Erdkugel, deren Positionen in Längen- und Breitengrad-Angaben bekannt sind, soll der direkt verbindende Großkreis, die Distanz, sowie die Abflugrichtung und und die höchste auf diesem Großkreis erreichte Breite bestimmt werden.

403 Visualisieren eines Tetraeder-Rings beim Umstülpen

Sechs leicht verlängerte Tetraeder (mit einem gleichseitigen Dreieck als Mittelebenen-Schnitt, und gleichschenkligen Dreiecken mit einem etwas spitzeren Winkel zwischen den gleichen Schenkeln als Begrenzungsflächen) können zu einem Ring zusammenfügt werden. In der Grundstellung liegen drei der aneinanderstoßenden Kanten von Nachbar-Tetraedern in der Mittelebene, die drei anderen gemeinsamen Kanten stehen dann senkrecht zur Mittelebene. Wenn die Verbindungen zwischen je zwei Tetraedern um die momentane Achsenrichtung der gemeinsamen Kante drehbar sind, so läßt sich das ganze Gebilde umstülpen. Diese Bewegung kann in 3D grafisch dargestellt werden.

404 Die Bewegung eines Autos auf einer Parkhausrampe

Mit 3D homogenen Koordinatentransformationen läßt sich die Bewegung eines Autos modellieren, das eine wendelförmige Parkhausrampe hinauffährt.

4.2
Selbsttests zum Kapitel 4

Testserie 4.1

T411) **Verständnisfragen**

- Wie kann man das Skalarprodukt zur Bestimmung der Länge eines Vektors einsetzen?
- Welche Elemente einer 3x3 Matrix für die homogene Koordinatentransformation der Ebene sind von vornherein festgelegt?
- Wie sieht die Matrix für die Punktspiegelung am Koordinatenursprung der Ebene in homogenen Koordinaten aus?
- Wie erzeugt man aus einem Vektor eine Projektionsmatrix, welche einen beliebigen räumlichen Vektor auf den vorgegebenen Vektor projiziert?

T412) Bestimmen Sie die Durchstoßpunkte der Geraden$(246)^T + \lambda \cdot (111)^T$ durch die drei Koordinatenebenen xy, xz und yz.

T413) Bestimmen Sie die Gleichung in der Hesse'schen Normalform für die Ebene E durch die drei Punkte A$(4/0/0)$, B$(0/3/0)$, C$(0/0/3.2)$. Bestimmen Sie ebenfalls die Gleichung der dazu parallelen Ebene F, welche den doppelten Abstand zum Koordinatenursprung aufweist wie E.

T414) Berechnen Sie die wahre Neigung der Ebenen ABC und ABD, sowie den Winkel zwischen diesen beiden Ebenen. A(0/0/0), B(2/2/8), C(10/0/0), D(0/4/0).

T415) Erarbeiten Sie die Matrizen der Gesamttransformationen in homogenen Koordinaten der Ebene für die Rotation des Quadrates ABCD (A(0/0), B(8/0), C(8/8)) um seinen Mittelpunkt, für die Drehwinkel 9θ, 180° und 270°. Testen Sie die Behauptung, dass das Quadrat durch jede der Abbildungen in sich selbst übergeht.

Testserie 4.2

T421) **Verständnisfragen**

- Welchem Wert entspricht der Betrag der Kreuzproduktes?
- Die Matrizen für kongruente Abbildungen der Ebene und des Raumes in gewöhnlichen Koordinaten (und damit die Teilmatrizen in der linken oberen Ecke bei homogenen Transformationen) gehören alle zu einem speziellen Matrizen-Typ. Zu welchem?
- Wie sieht die Matrix für eine Drehung um den Koordinatenursprung der Ebene um 90° in homogenen Koordinaten aus?
- Für zwei parallele Ebenen seien die Terme $e_n^T \cdot OV_1$ bzw. $e_n^T \cdot OV_2$ von verschiedenem Vorzeichen. Was bedeutet das für die Lage dieser zwei Ebenen in Bezug auf den Koordinatenursprung?

T422) Bestimmen Sie kürzeste Verbindung zwischen den windschiefen Geraden g: $(0\,0\,0)^T + \lambda \cdot (1\,1\,0)^T$ h: $(1\,0\,1)^T + \lambda \cdot (1\ -1\,0)^T$!

T423) Bestimmen Sie Distanz des Koordinatenursprungs von der Ebene E durch die drei Punkte A(16/0/0), B(0/15/0), C(0/0/20), sowie den Fusspunkt des Lotes vom Koordinatenursprung auf diese Ebene!

T424) Berechnen Sie die wahre Neigung der Seitenflächen und der Kanten für einen auf seiner Grundfläche stehenden regulären Tetraeder!

T425) Erstellen Sie ein M-File zum Zeichnen eines achtstrahligen Windrädchens mit Mittelpunkt (10/10) und der Flügelform entsprechend dem Dreieck (A(0/0), B(8/0), C(2/2))!

5
Übungen zum Kapitel 5

Potenzreihen

50–1 Potenzreihen von Exponentialfunktionen

Bestimmen Sie die Potenzreihen von e^x, e^{2x} und e^{x^2}.

50–2 Exponentialfunktionen mit verschiedenen Basen

Vergleichen Sie die Potenzreihen für e^x, 2^x, 10^x und allgemein a^x miteinander.

50–3 Verschiedene Entwicklungspunkte

Bestimmen Sie für den Wert $x = 3/2$ je die ersten 5 Terme der MacLaurin-Entwicklung von e^x, und der um den Punkt $x = 1$ entwickelten Taylor-Reihe derselben Funktion.

50–4 Approximation von 'e'

Bestimmen Sie die ersten 8 Glieder der Reihe, mit der die Zahl 'e' $e = e^1$ aus der MacLaurin-Entwicklung von e^x angenähert werden kann.

50–5 Elementweise Integration

Bestätigen Sie die unbestimmte Integrationsformel $\int \cos(x)\,dx = \sin(x) + C$ durch Vergleich der Potenzreihe mit der gliedweise integrierten Reihe.

50–6 Trigonometrische Ableitungen

Bestimmen Sie durch elementweise Ableitung der entsprechenden Potenzreihen die Ableitungen der Funktionen $\sin(x)$ und $\cos(x)$, sowie $\sin(5x)$ und $\cos(5x)$,

50–7 Konvergenzradius

Bestimmen Sie den Konvergenzradius der Potenzreihe $\sum_{n\to\infty}(x-2)^n/2^n$

50–8 Konvergenzradius einer Reihe mit fehlenden Potenzen

Bestimmen Sie den Konvergenzradius der Potenzreihe $\sum_{n\to\infty}(x^{3n}/3^n)$

50–9 Geometrische Reihe

In der Form $1 + q + q^2 + q^3 \ldots$ mit einem festen Wert von q als geometrische Reihe

bekannt, ist dieselbe Reihe in der Form $1+x+x^2+x^3\ldots$ eine besonders einfache Potenzreihe. Der Konvergenzbedingung $q<1$ entspricht der Konvergenzradius $R=1$. Erzeugen Sie diese Reihe durch Anwenden der Taylor- bzw. MacLaurin-Formel auf die Funktion $1/(1-x)$.

50–10 Elementare Binomische Reihe

Für ganzzahlige Exponenten n ist die binomische Form $(1+x)^n$ ein endliches Polynom. Für allgemeine, reelle ν definiert $(1+x)^\nu$ eine unendliche Potenzreihe $1+\binom{\nu}{1}x+\binom{\nu}{2}x^2\ldots$ Deren Koeffizienten $\binom{\nu}{k}$ sind die verallgemeinerten Binomialkoeffizienten $((\nu)(\nu-1)(\nu-2)\ldots(\nu-k+1))\ /\ (k!)$.
Bestimmen Sie nach diesem Muster die ersten 4 Terme von $\sqrt{1+x}=(1+x)^{1/2}$.

50–11 Approximative Wurzelberechnung

Benutzen Sie die ersten vier Terme der binomische Reihe zur approximativen Berechnung von $\sqrt[3]{9}$, definiert als $2\cdot(1+1/8)^{1/3}=2\cdot(1+\binom{1/3}{1}1/8+\binom{1/3}{2}1/8^2+\ldots)$.

Orthogonalpolynome

50–12 Simple Orthogonalpolynome

Konstruieren Sie die ersten vier Polynome, welche ohne Gewichtsfunktion über dem Intervall $[-1\ \ 1]$ orthogonal sind: $S_0=1$, $S_1=x$, $S_2=ax^2+bx+c$. Aus `int(S2*S0,x,-1,1)==0` und `int(S2*S1,x,-1,1)==0` lassen sich a,b,c bestimmen z.B. mit der Festlegung $c=-1$. Das analoge Verfahren ergibt die Koeffizienten von S_3.

50–13 Legendre Polynome

Mit der Funktion `legendreP(n,x)` im symbolischen Modus erhalten Sie das Legendre-Polynom vom Grad n. Die Legendre-Polynome sind ohne Gewichtsfunktion (bzw. mit $w=1$) auf dem Intervall $[-1\ \ 1]$ orthogonal. Testen Sie dies für einige Kombinationen n_1,n_2 mit `int(Pa*Pb,x,-1,1)`. Vergleichen sie P_2 und P_3 mit den Resultaten von 50–12.

50–14 Fit an Legendre-Polynome

Mit dem Fit-Ansatz `akoef = Y\f` erhalten Sie einen Fit der in der Design-Matrix `Y` eingespeicherten Teilfunktionen an die als Spaltenvektor vorgegebene Funktion `f`. Versuchen Sie dies mit den ersten fünf geraden Legendre-Polynomen und der Kosinus-Funktion im Intervall $[-1\ \ 1]$.

```
% legendrecos  Fit von cos(-1.. 1) an
%   gerade Legendre Polynome
x = -1:0.05:1; Y = 0*x+1;
for deg = 2:2:8
    Y = [Y ; legendreP(deg,x)];
end
  coslegcoef = (Y')\cos(x)'
```

Fourier-Analyse, Fourier-Transformation

50–15 Orthogonalität von Sinus und Kosinus

Erstellen Sie ein Funktions-M-File`sico()` mit den Eingabeparametern n, A, B und k, das eine Zahlenfolge der Länge n erzeugt, welche der Funktion$A * \cos(k * 2\pi * m/n) + B * \sin(k * 2\pi * m/n)\ ,\ \ m = 0 \ldots (n-1)$ entspricht.
Testen Sie an einigen Beispielen (nach Vorwahl von 'n') die Werte für `sum(sico(n,1,0,ka).*sico(n,1,0,kb))`für gleiche und für verschiedene 'ka', 'kb'. (= Skalarprodukt von Funktionen). Ebenso sollten `sum(sico(n,1,0,ka).*sico(n,0,1,kb))`für beliebige 'ka', 'kb' immer Null geben. Was bedeutet der Zahlenwert, den Sie beim Skalarprodukt von zwei identischen Folgen erhalten?

50–16 Grundeigenschaften der Fourier-Analyse

Sehen Sie sich die Resultatgrafiken des MATLAB-Skripts foudem.m für verschiedene Eingangsfunktionen an (fdefXXX.m können verschiedene Zeitfunktionen, alle mit dem Namen 'f' definiert werden). Beachten Sie besonders, dass:

- gerade Funktionen nur cos-Koeffizienten und
- ungerade nur sin-Koeffizienten aufweisen,
- Phasenverschiebungen die Gesamtamplitude konstant halten, aber die Anteile von sin und cos verändern, und dass
- bei steileren Flanken der Zeitfunktion die Amplituden der höheren Harmonischen zunehmen.

50–17 Verschiedene Darstellungen der Fourier-Transformierten

Betrachten Sie im Demo-Skript foushow.m die verschiedenen Darstellungsarten von Fourier-Transformierten! (benötigt rearrg.m) Die meisten der dazugehörenden Funktionen sind selbsterklärend.

fdefnarrpeak.m , fdefmedpeak.m , fdefwidpeak.m , fdefsquare.m

ergeben verschieden breite Rechteckpeaks,

fdefsin.m , fdefcos.m ,

ergeben die Elementarfunktionen und

fdeftriang.m , fdefsaegz.m , sowie fdefhat.m

ergeben verschiedene Arten von Dreiecksfunktionen.

50–18 Fourier-Transformation von Rechteckpulsen

Erzeugen Sie eine Funktion zum Herstellen von Zahlenreihen, welche Rechteckpulse darstellen. Als Eingabeparameter benötigen Sie die Anzahl Punkte, die linke Flanke und die rechte Flanke. Betrachten Sie jeweils mit foushow.m die Effekte einer Breitenänderung und einer Positionsänderung des Rechteckpeaks. Überlegen Sie sich in jedem Fall vorher, was Sie als Resultat erwarten, und versuchen Sie die Fälle zu verstehen, in denen Ihre Prognose falsch war.

50–19 Fourier-Koeffizienten mit negativen und positiven Frequenzen

Das Skript foushow.m (benötigt rearrg.m) zeigt alle Darstellungen nacheinander: die Fourier-Koeffizienten für cos und sin, dann die direkten Resultate der fft (fast fourier transform, ohne Normierung) und dann die Darstellung mit negativen und positiven Frequenzen.
In der Darstellung mit negativen und positiven Frequenzen müssen die Realteile der Koeffizienten symmetrisch zur y-Achse sein (gerade Funktion) und die Imaginärteile punktsymmetrisch zum Ursprung, damit eine reelle Zeitfunktion dargestellt wird.
Dann stimmen nämlich die Euler'sche Relation $e^{j\omega k} = \cos(\omega k) + j * \sin(\omega k)$ und die daraus abgeleiteten Formeln $\cos(\omega k) = 1/2 * (e^{j\omega k} + e^{-j\omega k})$ und $\sin(\omega k) = 1/2j * (e^{j\omega k} - e^{-j\omega k})$.
Mit foushowexp.m (benötigt ebenfalls rearrg.m) erhalten Sie direkt diese Darstellung.

50–20 Das Gibbs'sche Phänomen

Wenn zu einer Funktion die Fourier-Koeffizienten bestimmt werden, und diese vor der Synthese ab einer bestimmten Frequenz abrupt Null gesetzt werden, so entsteht das Gibbs'sche Phänomen, ein Überschwingen der rekonstruierten Funktion bei den steilen Flanken der Originalfunktion.
Dies kann mit dem M-File fclipm.m demonstriert werden.
Wie dieser Effekt durch ein sanftes Auslaufen der Fourier-Koeffizienten (mit Hilfe der Lanczos-Faktoren) vermieden werden kann, zeigt das Skript fclipa.m .

Das Prinzip des FFT-Algorithmus

50–21 FFT-Prinzip

Arbeiten Sie mit dem Demo-Programm shofftalgo.m (Hilfs-Files nicht vergessen!). Diese Files sind absichtlich als Skript konstruiert, damit man die verschiedenen internen Zwischenresultate betrachten kann. Nur im Inneren, zum Erzeugen der speziellen Matrizen, werden Funktionen verwendet. Schreiben Sie ein M-File, das den gleichen Effekt hat, wie die Multiplikation mit (OX+EX), das aber (mit einer Schleifenkonstruktion) nur die Summen/Differenzen und Multiplikationen mit w^k verwendet und damit ohne Matrixmultiplikation auskommt.

50–22 FFT kleiner Dimension

Schreiben Sie MATLAB-Skripts mit einzeln ausgeschriebenen Formeln für die DFT von Zahlenfolgen der Länge 2 und 4. Benutzen Sie ein Umstellen der Formeln, um die DFT der Länge 4 auf 2 DFT's der Länge 2 zurückzuführen. Benutzen Sie dasselbe Prinzip, um eine DFT der Länge 8 durch 2 DFT's der Länge 4 zu ersetzen. Kontrollieren Sie jeweils die Resultate mit dem Demo-Programm 'showfft.m'

Gewöhnliche Faltungen

50–23 Einführungsbeispiele von Faltungen

Berechnen Sie die Faltungen `conv(a,b)`, `conv(b,a)`, `conv(a,a)` und conv(b,b) von Hand und mit MATLAB:

```
a = [ 1 2 1 2 1] , b = [0 0 1 3 1 0 0]
```

50–24 Sukzessive Abrundung bei mehrmaliger Faltung

Falten Sie die Rechtecksfunktion `a = [ 0 0 0 1 1 1 1 0 0 0]` ein- bis mehrmals mit sich selbst und vergleichen Sie die grafischen Darstellungen der Folgen.

50–25 Darstellung der sukzessiven Addition bei der Faltung

Gegeben sind die zwei Zahlenfolgen (der Länge 14) g und w durch

```
g = [ 1 2 3 2 1 zeros(1,9) ]
w = [ 0 0 1 0 0 0 2 0 0 1 2 1 0 0 ]
```

Erzeugen Sie eine Shift-Funktion, mit der Sie den von Null verschiedenen Teil von g nach rechts verschieben können ohne die Länge zu ändern, z. B. wird:

```
shrgt(g,4) = [ 0 0 0 0 1 2 3 2 1 0 0 0 0 0]
```

Mit dieser Shift-Funktion kann die Faltung als schrittweise Addition von zueinander verschobenen Folgen g, je mit entsprechenden Gewichten w(k) nachvollzogen werden. Die Zahlenfolgen f1 bis f4 bilden dann die Teilresultate und f5 das Schlussresultat der Faltung von g und w. Diese sollen übereinander geplottet werden:

```
f1 = 1*g
f2 = 1*g + 2*shrgt(g,4)
f3 = 1*g + 2*shrgt(g,4) + 1*shrgt(g,7)
f4 = 1*g + 2*shrgt(g,4) + 1*shrgt(g,7) + 2*shrgt(g,8)
f5 = 1*g + 2*shrgt(g,4) + 1*shrgt(g,7) + 2*shrgt(g,8)
+ 1*shrgt(g,9)
```

`f5` kann anschließend mit der MATLAB-Funktion für die Faltung verglichen werden: `conv(g,w)`

50–26 Faltung von Standardfunktionen

Falten Sie die vier Standardfunktionen: Rechtecks-, (Dreiecks=) Hutfunktion, Parabelbogen ($\max(0, 1-x^2)$ und Glockenfunktion ($(1-x^2)^2$, $-1 < x < 1$) je mit sich selbst, sowie mit der Rechtecks und der Hutfunktion und stellen Sie die Original- und Resultatfunktionen grafisch dar.

50–27 Faltung einer Sinusfunktion mit sich selbst

Berechnen und zeichnen Sie die zirkuläre Faltung einer Sinusfunktion mit sich selbst.

50–28 Selbst programmierte Faltung

Erzeugen Sie ein Funktions-M-File, das eine Faltung von 2 Folgen berechnet, indem die verschobenen und mit dem Gewicht multiplizierten Vektoren sukzessive addiert werden.
Erzeugen Sie dazu eine intern zu verwendende Hilfsfunktion, welche einen Vektor an einem vorwählbaren Platz in einen längeren Nullvektor einsetzt, wie z. B.

```
bigvec = vecinsert(smallvec,lpos,biglength)
```

50–29 Nach der Formel programmierte Faltung

Schreiben Sie ein MATLAB-Skript, welches eine gewöhnliche Faltung programmiert. Diese ist durch die untenstehende Formel gegeben. Vergleichen Sie die Resultate Ihres Skripts mit der Bibliotheksprozedur `C=conv(A,B)`! Testen Sie auch die Symmetrieeigenschaft der Faltung und die Länge der Resultatfolge an einigen Beispielen!

$$c_j = \sum_{k=\max(1,j-n2+1)}^{\min(n1,j)} a_k \cdot b_{j-k+1}$$

Beachten Sie, dass die sprachliche Formulierung „`j-n2+1`, **aber mindestens** `1`“ mit der Funktion **max** `(1, j-k+1)` realisiert wird (nicht mit „min“ was die sprachliche Form suggeriert)!

Zirkuläre Faltungen

50–30 Zirkuläre Faltung mit einfacher Schleife

Unter Verwendung des untenstehenden Funktions-M-Files `shrgt.m`:

```
function vecret = shrgt(vecin, nshift)
ntot = length(vecin); nlast = ntot - nshift ;
vecret = [ vecin(nlast+1:ntot) , vecin(1:nlast)];
```

soll eine zirkuläre Faltung allgemein für die zwei gleich langen Folgen a und b durch eine einzelne Schleife programmiert werden.

Ausblick: Versuchen Sie `shrgt.m` so umzuprogrammieren, dass die Funktion robuster wird, d. h. dass negative Werte, Null oder zu große Werte von nshift erlaubt sind und sinnvoll umgesetzt werden!

50–31 Zirkuläre Faltung mit doppelter Schleife

Programmieren Sie eine Doppelschleife mit Bedingungen, so dass eine allgemeine zirkuläre Faltung entsprechend der Summenformel im Theorieteil berechnet wird (Ausprogrammieren einer Formel).

50–32 Faltung einer Sinus-Funktion mit sich selbst:

Berechnen und Zeichnen Sie die Zirkuläre Faltung einer Sinus-Funktion mit sich selbst.

50–33 Zirkuläre Faltung mit Faltungssatz

Testen Sie den Faltungssatz am Beispiel der zirkulären Faltung der beiden Folgen `a = [ 1 2 3 4 5 6]` und `b = [1 0 0 1 0 0]` und berechnen Sie die zirkuläre Faltung zur Kontrolle von Hand.

Der Faltungssatz sagt aus, dass eine zirkuläre Faltung durch je eine Fourier-Transformation der Ausgangsfolgen, eine elementweise Multiplikation (genannt Hadamard-Produkt) der beiden Fourier-Transformierten und eine anschließende Rücktransformation ersetzt werden kann:

```
at = fft(a); bt = fft(b);
ct = at .* bt; c = ifft(ct); cr = real(c);
```

50–34 Funktions-M-File für zirkuläre Faltung mit FFT

Ergänzen Sie die untenstehende Berechnungsvorschrift zu einem lauffähigen Funktions-M-File in MATLAB und berechnen Sie damit die zirkuläre Faltung für einige Beispiele. Testen Sie, wieviele Nullen Sie an jede der Eingangsfolgen anhängen müssen, damit Sie die normale Faltung bekommen. Weisen Sie auch nach, dass die Nullen am Anfang oder Ende angehängt werden können!

```
A = fft(a) ; B = fft(b); C = A.*B ; c = real(ifft(C))
;
```

Erhöhen Sie die **Robustheit** Ihrer Funktion durch Angleichen der Länge bei Eingabefolgen verschiedener Länge.

50–35 Normale Faltung auf zirkuläre Faltung zurückführen

Führen Sie die normale Faltung der Folgen `a = [ 1 2 3 2 1]` und `b = [1 2 1 1 1 2 1]` aus. Finden Sie heraus, wieviele Nullen (zero padding) Sie an jede der Folgen mindestens anhängen müssen, dass die zirkuläre Faltung der verlängerten ap und bp via Schnelle Fourier-Transformation dasselbe ergibt.

```
cr = real( ifft(fft(ap).*fft(bp)) ) ;
```

50–36 Vergleich von zirkulärer und gewöhnlicher Faltung

Zeichnen Sie die beiden Resultate der zirkulären und der gewöhnlichen Faltung der beiden Folgen a=[1 2 3 4 5 4 3 2 1] und b = [0 1 1 1 1 1 1 1 0]. Beachten Sie, dass bei der zirkulären Faltung kein auslaufender Rand entsteht.

501 Animation der Effekte von Phasenverschiebung und Pulsverbreiterung

In simultan dargestellten Teilbildern der komplexen Spektren kann der Effekt einer Phasenverschiebung sowie derjenige einer Verkleinerung und Vergrößerung der Pulsbreite eines Rechteckpulses als Animation dargestellt werden.

502 Heraussuchen von typischen Instrumentenstimmen

Suchen Sie in den .wav - Files von Solokonzerten langsame Stellen. Programmieren Sie in MATLAB das Einlesen von wav-Files und bestimmen Sie an geeigneten Abschnitten das Fourier-Spektrum des speziellen Soloinstrumentes. Vergleichen Sie verschiedenartige Instrumente in Bezug auf deren Fourier-Spektrum. Die Stärke und die Verteilung der Oberharmonischen bestimmen die Klangfarbe.

5.1
Selbsttests zum Kapitel 5

Testserie 5.1

T511) **Verständnisfragen**

- Welches ist der Konvergenzradius der simplen geometrischen Reihe $1 + x + x^2 + x^3 \ldots$?
- Wodurch können sich z. B. zwei Zeitfunktionen unterscheiden, welche dasselbe Powerspektrum haben?
- Welchen Konvergenzradius kann eine Potenzreihe höchstens aufweisen, wenn die Koeffizienten a_k gegen 1 streben?
- Wie erhält man den gewöhnlichen Sinus-Koeffizienten für die Frequenz $4\omega_0$ aus dem komplexen z_k- Tabelle $(k = 1 \ldots n)$

T512) Bestimmen Sie die Indizes, für welche die Terme $x^k/k!$ der Potenzreihe für $\exp(x)$ den Wert $1e-3$ unterschreiten, für $x = 1,\ \ 2,\ \ 10$.

T513) Geben Sie die Amplituden–Phasen-Darstellung und das Powerspektrum an für für die Fourier-Reihe:

$$f(t) = 1 + 1/2 \cdot \cos\left(\frac{2\pi}{T}t\right) + 1/2 \cdot \sin\left(\frac{2\pi}{T}t\right) - 1/8 \cdot \sin\left(\frac{6\pi}{T}t\right) + 1/8 \cdot \cos\left(\frac{6\pi}{T}t\right)$$

T514) Bestimmen Sie die ersten 6 Tschebyscheff-Polynome aus der Start-Angabe $T_0 = 1$, $T_1 = x$ und der Rekursion $T_{k+1} = 2x \cdot T_k - 3 \cdot T_{k-1}$

T515) Finden Sie die gewöhnlichen Fourier-Koeffizienten ahnand der Angabe, dass das Power-Spektrum $p_0 \ldots p_9$ die Werte $p_1 = 4$, $p_4 = 1/4$ und sonst Null aufweist, und dass die Funktion gerade ist.

Testserie 5.2

T521) **Verständnisfragen**

- Wieso setzt man die Gleichstromkomponente als $a_0/2$ in die Formel?
- Schreiben Sie die Formel $\sin^2(x) + \cos^2(x)$ mit den Taylor-Termen bis und mit x^2.
- Wieviele Koeffizienten kann eine Fourier-Darstellung maximal enthalten, wenn dies bis zur 10. Harmonischen geht?
- Wie kann man einen einzelnen komplexen Koeffizienten z. B c_k in einer Fourierreihe ersetzen, wenn man sonst alle komplexen Koeffizienten kennt?

T522) Führen Sie die einfache Faltung der Folgen `a=[1 2 3 4]` und `b = [1 0 1 1 0 1]` mit Bleistift und Papier durch, sowohl als direkte einfache Faltung, als auch als zirkuläre Faltung bei welcher die notwendige Anzahl Nullen angehängt wurde. Vergleichen Sie die Resultate.

T523) Geben Sie die Amplituden–Phasen-Darstellung, das Powerspektrum und die komplexen Koeffizienten an für die Fourier-Reihe:

$$f(t) = 1 + 1/2 \cdot \cos\left(\frac{2\pi}{T}t\right) + 1/4 \cdot \sin\left(\frac{4\pi}{T}t\right) - 1/8 \cdot \sin\left(\frac{6\pi}{T}t\right) + 1/10 \cdot \cos\left(\frac{10\pi}{T}t\right)$$

T524) Überlegen Sie sich, vorerst ohne nachzuschauen, wie die komplexe Matrix zu einer DFT von 4 Punkten, k=0 bis 3 aussieht.

T525) Schreiben Sie ein MATLAB-Skript, das durch Punkt-Multiplikation und Summation mit Funktionen der Art $\sin\ \cos(n \cdot k \cdot 2\pi/8)\ \ k = 0 \ldots 3,\ \ n = 0 \ldots 7$ zu einer tabellarisch gegebenen Funktion von 8 Werten die DFT bestimmt.

6
Übungen zum Kapitel 6

Funktionsdarstellung

60–1 Konturplot einer Halbkugel

Orientieren Sie sich anhand der Hilfefunktion von MATLAB über die Möglichkeiten zum Erstellen von Konturplots (contour). Stellen Sie die Oberfläche einer nach oben gewölbten Halbkugel grafisch dar. Die Werte außerhalb des Äquators sollen Null gesetzt werden (bzw. Null bleiben, wenn man mit zeros() beginnt).

60–2 Konturplot

Zeichnen Sie einen Konturplot der Funktion$F(x.y) = \sin(x) * \cos(y)$ im Bereich $0 \leq x < 2\pi \quad 0 \leq y < 2\pi$ und suchen Sie darin alle Punkte für die gleichzeitig gilt $\partial F/\partial x = 0$ und $\partial F/\partial y = 0$

60–3 Grafische Darstellung - „Jurahügel“

Füllen Sie eine rechteckige Matrix mit den Höhenwerten des Beispiels mit dem Jurahügel:
$h(x, y) = 1600 - (x/1000)^2 * 50 - (y/250)^2 * 50$ entsprechend den Intervallen $-1600 < x < 1600$ und $-800 < y < 800$. Ein noch besseres Bild ergibt sich mit max(h,1200), das entspricht dem Hügelteil über dem Nebelmeer.
Stellen Sie diese Funktion mit 'contour' und 'surf' grafisch dar.

60–4 Konturplots einfacher Matrizen:

Verwenden Sie die Index-Anzeige-Matrix A aus den Schleifen-Programmierübungen mit $n = 9$ als Eingabefunktion für eine Kontur- oder Surf-Darstellung.
Ebenso eignen sich die oberen und unteren Dreiecksmatrizen mit den Werten, welche mit dem Abstand von der Diagonalen ansteigen, zur grafischen Darstellung als Fläche. Mit der oberen Dreiecksmatrix`Du` dieses Typs erhalten Sie durch
`Ph = [ Du fliplr(Du) ]`und `P = [ flipud(Ph) ; Ph ]`die Funktion einer Pyramide.

60–5 Konturplots von verschiedenen Matrizenprodukten

Vergleichen Sie die Kontur-/Surf-Formen der folgenden nxn Matrizen miteinander (n ca. zwischen 9 und 20). ('Du' ist eine obere Dreiecksmatrix mit Werten, welche

mit der Distanz zur Diagonalen zunehmen.)

```
Du, Du*Du, Du.*Du, Du^ 3 , Du.^ 3
C=fliplr(Du), C*C, C.*C, C^ 3 , C.^ 3
```

60–6 Trichterfunktion
Bestimmen Sie eine Funktion, welche die Form eines Trichters beschreibt. Durchmesser der Ausflussöffnung 1 cm, Randdurchmesser 21 cm, Neigung 45°.

60–7 Kissenfunktion
Entwerfen Sie eine Funktion, welche die Oberflächenform eines gut gefüllten Kissens beschreibt!

60–8 Futtersilo mit Bodenlagerung
Ein auf einem flachen Platz von 2x6 m aufgeschütteter Haufen von Silofutter wird mit einer Plane zugedeckt und mit Seilen festgebunden. Die Höhe des Haufens über dem Platz sei durch die Funktion$z(x, y) = 0.2 \cdot x \cdot (6 - x) \cdot y \cdot (2 - y)$ gegeben.
Berechnen Sie das Gesamtvolumen dieses Bodenspeichers.
Bestimmen Sie die 3D-Parameterdarstellung des Verlaufes von einigen der Fixierseile, welche zwischen den 16 Pflöcken (in den Ecken und 1 Pflock pro Meter) diagonal gespannt sind (Richtung im Grundriss$+45^{\circ}$ und -45°).
Stellen Sie zur Kontrolle die Fläche und die Seile mit MATLAB 3D-Grafik dar.

Ausblick: Bestimmen Sie das Restvolumen als Funktion der x-Position, bis zu welcher der Haufen bereits abgetragen ist.

Partielle Ableitungen

60–9 Analytische Bestimmung von partiellen Ableitungen
Bestimmen Sie je alle möglichen partiellen Ableitungen der folgenden Funktionen:

a) $x^2 e^y \sin(z)$
b) $e^{xy} + e^{-xy}$
c) $\sin^2(x) \cdot \cos^3(z)$
d) $x^3 y^2 z \frac{1}{u}$

60–10 Bestimmen von partiellen Ableitungen:
Bestimmen Sie jeweils analytisch alle partiellen Ableitungen der untenstehenden Funktionen:

$$\text{a) } F(x, y, z) = \frac{\sqrt{x^2 + y^2}}{z}$$
$$\text{b) } F(x, y) = \sin(x * \cos(4y))$$
$$\text{c) } F(x, y, z) = z^2 \sqrt{xz + y/z}$$

60–11 Gradient: alle partiellen Ableitungen in einem Vektor zusammengefasst
Bestimmen Sie jegrad$F = \begin{bmatrix} \frac{\partial F}{\partial x} & \frac{\partial F}{\partial y} & \frac{\partial F}{\partial z} \end{bmatrix}'$

a) $F = x^2 + y^2$
b) $F = \sin(x) \cdot \cos(y) \quad 0 \leq x, y \leq 2\pi$
c) $F = \frac{1}{\sqrt{x^2+y^2+z^2}}$

60–12 Gradientenplot
Zeichnen Sie die zuerst die Höhenlinien der Funktion $F(x, y) = \sin(\pi \cdot x) \cdot \sin(\pi \cdot y)$ im Bereich $x,\ y = 0 \ldots 1$
Darin soll nun ein Feld von kleinen Strichen (bzw Pfeilen) eingezeichnet werden, welche die Gradientenvektoren darstellen. Dazu existieren die Bibliotheksfunktionen `gradient()` `quiver()`;

60–13 Partielle Ableitungen und Gradientenfunktion:
Bilden Sie die beiden partiellen Ableitungen der Jurahügelfunktion $h(x, y) = 1600 - (x/1000)^2 * 50 - (y/250)^2 * 50$ und stellen Sie diese ebenfalls je als zweidimensionale Funktion dar!
Zeichnen Sie in den Konturplot der Originalfunktion die Gradientenvektoren ein!

60–14 Konturplots und Gradientenvektoren
Zeichnen Sie die Höhenlinien und 3D-Flächendarstellungen (`surf`) für die folgenden Funktionen: (alle im Bereich $x,\ y = \pm 1$)

a) $F(x, y) = 1 - x^2 - 0.2 \cdot y^2$
b) $F(x, y) = \sin(\pi \cdot x) \cdot \sin(\pi \cdot y)$
c) $F(x, y) = (x^2 - |x^3|) \cdot (y^2 - |y^3|)$

Der Gradientenvektor, d. h. die Einreihung der partiellen Ableitungen als Komponenten eines Vektors, liefert eine Art Funktion von mehreren Variablen mit der speziellen Eigenschaft, dass für jeden Punkt (bzw. jede Kombination der unabhängigen Variablen) mehrere Funktionswerte geliefert werden. Man nennt dies eine vektorwertige Funktion.
Bestimmen Sie die Gradientenvektoren der oben angegebenen Funktionen.

60–15 Stationäre Punkte
Suchen Sie die stationären Punkte (beide partiellen Ableitungen gleichzeitig Null) in der Funktion $F(x, y) = (x^2 - x^4) \cdot (y^2 - y^4)$ und vergleichen Sie das Ergebnis mit den Formen der Höhenlinien in der vorhergehenden Aufgabe!

60–16 Biquartische Funktion

Zeichnen Sie einen Konturplot der biquartischen Funktion (in beiden Richtungen vom 4. Grad), welche im untenstehenden M-File definiert wird. Berechnen Sie die partiellen Ableitungen dieser Funktion und zeichnen Sie Gradientenvektoren in den Kontur-Plot ein!

```
for k=1:50
  for l = 1:50
    x = (k -26)*0.07; y = (l-26)*0.07;
    V(l,k)=(0.25*x^4-0.5*x^2)-0.12*x +...
        (0.25*y^4-0.5*y^2)+1-0.05*y;
  end
 end
```

60–17 Das elektrische Potential einer Punktladung:

Das elektrische Potential einer Punktladung ist gegeben durch die Formel:
$\phi(r) = \dfrac{Q}{4\pi\varepsilon_0} \cdot \dfrac{1}{r}$ Dabei ist Q die Ladung und ε_0 die Dielektrizitätskonstante. Schreiben Sie diese Funktion in eine Funktion der kartesischen Koordinaten $x,\ y,\ z$ um und berechnen Sie die drei partiellen Ableitungen analytisch, je ebenfalls als Funktionen der drei Variablen $x,\ y,\ z$. Durch Zusammenstellen dieser drei Funktionen als Gradientenvektor erhalten Sie die Vektorfunktion des elektrischen Feldes $E = grad(\phi)$.

Fitprobleme

60–18 Geraden- Parabel- und kubischer Fit

Bestimmen Sie die Gleichungen für einen Parabelfit $y = p_1 * x^2 + p_2 * x + p_3$ und einen Fit an eine Funktion dritten Grades $y = p_1 * x^3 + p_2 * x^2 + p_3 * x + p_4$ analog zur Herleitung für den Geradenfit.
Dazu werden die partiellen Ableitungen der Zielfunktionen
$\sum (p_1 * x_k^2 + p_2 * x_k + p_3 - y_k)^2$ nach $p_1 \ldots p_3$ bzw.
$\sum (p_1 * x_k^3 + p_2 * x_k^2 + p_3 * x_k + p_4 - y_k)^2$ nach $p_1 \ldots p_4$ Null gesetzt werden.
Programmieren Sie diese zwei Fälle sowie den Geradenfit in MATLAB und wenden Sie die M-Files auf die untenstehenden Punkte an!
$x_k = 0.0,\ 1.0,\ 2.0,\ 3.0,\ 4.0$
$y_k = 1.0,\ 1.2,\ 2.2,\ 3.0,\ 3.6$

60–19 Kubischer Fit

Suchen Sie die beste Kurve 3. Grades (bzw. die Koeffizienten $a_3,\ a_2,\ a_1,\ ,a_0$) durch die Punkte $(-3/-7),\ (-2/-5),\ (-1/-2),\ (0/0),\ (1/-1),\ (2/2),\ (3/0)$ durch Aufstellen der Fehlergleichungen und anschließende Standard MATLAB-Lösung.

60–20 Fit mit Gewicht:

Bestimmen Sie nach dem Gauß'schen Ausgleichsprinzip die beste Gerade durch die Punkte $(-2/-2),\ (-1/0),\ (0/1),\ (1/1.5),\ (2/3)$
a) durch Lösen der Normalengleichungen
b) durch Aufstellen der Fehlergleichungen und MATLAB-Lösung `A \ b`

Diese beiden Fits sollen anschließend wiederholt werden unter Anwendung der Gewichte (4, 2, 1, 2, 4) für die gegebenen Punkte. Vergleichen Sie die verschiedenen Resultate!

60–21 Fehlergleichungen

Wenn man die Formeln genauer analysiert, sieht man, dass die einfachen Fitverfahren ein Problem der Art $A^T \cdot A \cdot p = A^T \cdot y$ lösen.
A ist dabei die Matrix aus den Vektoren `[ x. ^2 x ones(n,1) ]`
In MATLAB kann man aber direkt das überbestimmte Fehlergleichungssystem lösen:
MATLAB löst mit $p = A \backslash y$ ein überbestimmtes System $A \cdot p = y$ automatisch im Sinne eines Fits!
Lösen Sie die obigen Fitprobleme auch noch mit dem viel einfacheren Prinzip der Fehlergleichungen!

60–22 Fit nach Potenzfunktionen

Gegeben sind die Punkte (1/1) (2/3) (3/4) (4/2) (5/1). Suchen Sie die bestem Parameter A, B, C, (evtl. D), damit die Funktionen

a) $A + B \cdot x + C \cdot x^2$
b) $A + B \cdot x + C \cdot x^2 + D \cdot x^3$

möglichst nahe an die Punkte kommen. Stellen Sie die Funktionskurve und die Punkte mit MATLAB grafisch dar!
Lösen Sie jede der Aufgaben sowohl mit den Normalengleichungen als auch mit den Fehlergleichungen und testen Sie jeweils die Formel $A^T \cdot A = N$.

60–23 Fit nach frei festlegbaren Funktionen

Schreiben Sie ein Skript-M-File, das einen Fit n gegebenen Punkten x_x, y_k an eine Linearkombination von drei frei festlegbaren Funktionen `A*fitfun1(x)+B*fitfun2(x) =C*fitfun3(x)` durchführt!
Testen Sie dieses, indem Sie 5 Punkte der Sinusfunktion zwischen 0 und π wählen und als fitfun1 bis 3 die Konstante 1, die Funktion x und die Funktion x^2 wählen.

60–24 Orthogonale Funktionen beim Fitten:

Fitten Sie die untenstehenden Punktreihen nacheinander (mit der Methode der Fehlergleichungen) nach 1, 2 oder allen 3 der Funktionen $\sin(2\pi \cdot x), \sin(2 \cdot 2\pi \cdot x), \sin(4 \cdot 2\pi \cdot x)$
Beachten Sie, dass die bei der kleineren Gruppe gefundenen Koeffizienten dieselben sind, wie wenn nach einer größere Anzahl solcher Funktionen gefittet wird.
Dies ist die Eigenschaft der in der Fourier-Zerlegung verwendeten Funktionen die man Orthogonalität von Funktionen nennt.

$x_k = 0.0,\ 0.1,\ 0.2,\ 0.3,\ 0.4,\ 0.5,\ 0.6,\ 0.7,\ 0.8,\ 0.9$
$y_k = 0.0,\ 1.0,\ 1.2,\ 0,\ 1.0,\ 0.0,\ 0,\ 0,\ 0,\ -1.0$

Methode der Lagrange-Multiplikatoren

60–25 Lagrange-Multiplikator bei einer „Hügelform"-Funktion

Stellen Sie die Gleichungen auf, welche das Optimum der Funktion $F(x, y) = 10 - 0.2 * x^2 - 0.05 * y^2$ definieren, unter der Bedingung, dass $y = 5 - x$ und lösen Sie das entstehende Gleichungssystem.

60–26 Lagrange-Multiplikatoren:

Suchen Sie das Minimum der Funktion
$F(x, y, z) = x^2 + 2y^2 + 4z^2$
unter der Nebenbedingung
$x - y - 2z = 8$

60–27 Nichtlineare Nebenbedingung und Abstandsfunktion

Stellen Sie das nichtlineare Gleichungssystem auf, welches das Optimum der Funktion $F(x, y) = \sqrt{x^2 + y^2}$ definiert, unter der Bedingung, dass $y = 5/x^3$.

60–28 Visualisierung der Optimierung mit Nebenbedingung

Die Aufgabe, den höchsten Punkt der Fläche $z(x, y) = 10 - 0.02 \cdot x^2 - 0.1 \cdot y^2$ unter der Bedingung $y = x - 4$ soll mit der Lagrange-Multiplikator-Methode gelöst werden.
Dann soll ein Surf- oder Konturplot der Fläche gezeichnet werden und die 3D-Linie x(t),y(t),z(t), welche auf der Fläche liegt und die Bedingung erfüllt, soll eingezeichnet werden. Die modifizierte Fläche mit h=z für y≥x-4 und 0 für y< x-4 soll ebenfalls gezeichnet werden.

Singular Value Decomposition und Pseudoinverse

60–29 Anwenden der pseudoinversen Matrix

Bestimmen Sie mit Hilfe der Bibliotheksfunktion svd die Pseudoinverse zur Fehlergleichungsmatrix, welche eine Funktion 3. Grades an die Punkte $x = -3 : 3$, mit verschiedenen Sätzen von $y = y_k$ fittet, so dass für jeden Satz von y-Werten die Parameter direkt durch $p = A^+ * y$ berechnet werden können. Zeichnen Sie das Resultat je für die y-Werte `[ -1 1 1 0 0 1 3 ]` und `[ 1 2 1 2 3 3 5 ]` und vergleichen Sie jeweils das Resultat mit der direkten Fehlergleichungslösung $A\backslash y$.

Nichtlineare Gleichungssysteme

60–30 Jacobi-Matrix:

Bestimmen Sie alle partiellen Ableitungen aller Funktionen
$\partial f_j / \partial x_k, \;\; j = 1..3, \;\; k = 1..3$ für die Funktionen:

$$f_1(x_1, x_2, x_3) = \sqrt{x_1^2 + x_2^2 + x_3^2},$$
$$f_2(x_1, x_2, x_3) = \mathrm{atan}(x_2/x_1),$$
$$f_3(x_1, x_2, x_3) = \mathrm{atan}(x_3/\sqrt{x_1^2 + x_2^2})$$

60–31 Nichtlineare Gleichungssysteme

Suchen Sie iterativ mit dem zweidimensionalen Newton-Verfahren ab dem Startwert (2,2) eine Lösung des Gleichungssystems

$$(x-4)^2 + y^2 = 10$$
$$x^4 + y^4 = 20$$

60–32 Kugelkoordinaten

Die Kugelkoordinaten ergeben ein gutes Beipiel für die Auflösung von nichtlinearen Gleichungssystemen. Bestimmen Sie zu den drei untenstehenden Funktionen von drei Variablen zuerst analytisch die Jacobi-Matrix. Berechnen Sie anschließend 2–3 Verbesserungsschritte, ausgehend vom Schätzwert:

$$x_1 = 4,\ x_2 = 0.4.\ x_3 = 0.6$$
$$f_1(x_1, x_2, x_3) = x_1 \cdot \cos(x_2) \cdot \cos(x_3) - 3$$
$$f_2(x_1, x_2, x_3) = x_1 \cdot \sin(x_2) \cdot \cos(x_3) - 1$$
$$f_3(x_1, x_2, x_3) = x_1 \cdot \sin(x_3) - 2$$

60–33 Funkfeuerortung

Lange vor der Verfügbarkeit des GPS wurden Positionsbestimmungen vorgenommen durch die Auswertung der Zeitdifferenzen zwischen dem Empfang von zeitsynchron ausgestrahlten Signalen mehrerer Funkfeuer, deren Standort man kannte. Dies kann auf die Aufgabe reduziert werden, dass die Differenz d1 der Distanzen zu A(0/0) und B(0/10), sowie d2 zwischen A(0/0) und C(10/0) bekannt ist und daraus die Position (x/y) gesucht wird. Programmieren Sie das Lösungsverfahren nach dem 2D-Newton-Prinzip, um x, y aus d1, d2 zu finden.

6.1
Miniprojekt zu Funktionen mit mehreren Variablen

601 Elektrisches Potential

Durch eine kleine Zahl von Punktladungen, die in der Ebene verteilt liegen und deren positive oder negative Ladungsstärke bekannt ist, wird die zugehörige elektrische Potentialfunktion definiert. Erstellen Sie ein Programm zur Darstellung der Potentialfunktion aufgrund einer Eingabe von Ladungspositionen mit dem Cursor. Die Ladungsstärken können die Default-Werte$+1$, -1, $+1$, etc. annehmen oder durch globale Variablen vordefiniert werden.

6.2 Selbsttest zum Kapitel 6

Testserie 6.1

T611) **Verständnisfragen**

- Bestimmen Sie die Komponenten eines Tangentialvektors zu einer Höhenlinie der allgemeinen Funktion $F(x, y)$ von zwei Variablen!
- Im Grundriss eines 2D Konturplots zeigt der Gradientvektor in die Richtung des steilsten Aufstiegs, entlang der sogenanten Fall-Linie. Überlegen Sie sich, welche z-Komponente man einem solchen Vektor in einem 3D Flächenplot zuordnen müsste, damit er zur Oberfläche tangential verläuft!
- Warum ist es unumgänglich, dass beim Aufstellen der Lagrange-Funktion die Nebenbedingung in der "$= 0$-Form" also als $G(x_1, x_2, \ldots) = 0$ einbezogen wird?
- Bei der eindimensionalen Newton-Iteration zur Nullstellensuche verwendet man den Vorfaktor $-1/f'(x^{(k)})$, um aus der noch vorhandenen Abweichung von Null $f'(x^{(k)})$ eine x-Korrektur zu erhalten. Wie lautet das mehrdimensionale Analogon zum Term $-1/f'(x^{(k)})$?

T612) Produzieren Sie ein M-File zur Erstellung eines 3D-Konturplots der rotierten Glockenkurve mit den Werten $h(x, y) = (1 - x^2 - y^2)^2$ für $x^2 + y^2 < 1$ und $h = 0$ sonst!

T613) Über einem Quadrat A(0/0), B(5/0), C(5/5), D(0/5) sind die Höhenwerte an den Ecken hA=0, hB=1, hC=4, hD=1. Dazwischen ist die Höhenfunktion durch bilineare Interpolation bestimmt. Mit $f = x/5$ und $g = y/5$ ist
$h(x, y) = hA + f(hB - hA) + g(hD - hA) + f \cdot g(hC - hD - hB + hA)$
Erstellen Sie ein M-File zur 3D-Darstellung dieser Fläche, sowie je in einer anderen Figur die Darstellungen der beiden Flächen mit den Werten der partiellen Ableitungen!

T614) Lösen sie das Optimierungproblem Z = min $Z(x, y, z) = 4x^2 + y^2 + z$ unter der Nebenbedingung $(x + 2y + 4z = 8)$!

T615) Erstellen Sie ein M-File, das die Nullstelle (x/y) des Funktionspaars $F_1 = x^4 + y^4 - 20$ und $F_2 = x^2 - y^2 - 1$ mit der Newton-Iteration sucht.

7
Übungen zum Kapitel 7

Analytische Lösungen im Symbolischen Modus

70–1 Geraden durch den Koordinatenursprung

Das Geradenbündel durch den Koordinatenursprung wird durch die Gleichungen $y(x) = C \cdot x$ beschrieben, mit der Steigung C als Scharparameter. Die Differentialgleichung, mit diesem Bündel als allgemeine Lösung, ergibt sich durch Ableiten und anschliessendes Eliminieren von C aus den beiden Gleichungen (Original und Ableitung). Aus $y = Cx$ und $y' = C$ wird also $y' = y/x$, wie zu erwarten war. Entlang jeder Geraden ist die Steigung y' konstant und stimmt überall mit dem Quotienten y/x überein. Lösen Sie diese Differentialgleichung mit `syms` und `dsolve`, um das Geradenbündel wieder zu erhalten.
Beachten Sie dabei: `Dy` bedeutet `dy/dt`. Das hier anzuwendende `diff(y)==..` für das mit `syms y(x)` definierte `y` darf nicht in Apostroph-Zeichen stehen. (`'Dy==..'` muss hingegen in Apostroph stehen). Ebenfalls zu beachten ist, dass im symbolichen Rechnen Gleichheit mit `==` beschrieben wird.

70–2 Konzentrische Kreise

Bei der Kurvenschar von konzentrischen Kreisen um den Ursprung $x^2 + y^2 = C$ stehen alle Kreise senkrecht zu allen Geraden des Bündels $y = Cx$. Die zwei Scharen sind je zueinander orthogonale Trajektorien. Die Differentialgleichung der Kreise ergibt sich deshalb indem y' bei den Geraden durch $-1/y'$ ersetzt wird. Testen Sie diesen Sachverhalt, indem Sie nun statt $y' = y/x$ (Geraden) die neue Differentialgleichung $y' = -x/y$ (Kreise) lösen.

70–3 Differentialgleichung zur Parabelschar

Die Schar von Parabeln $y = C \cdot x^2$ mit demselben Scheitel bei $(0/0)$ ist die allgemeine Lösung der Differentialgleichung $y' = 2y/x$.
Die symbolische Lösung liefert noch offene Integrationskonstanten, hier `C3`, welche aber nicht zugreifbar sind. Um die Konstante in der allgemeinen Lösung festzulegen benutzt man am besten `syms K` und
`pa=dsolve(diff(y) == 2*y/x,'y(1)==K')` Beachten Sie: Der x-Wert für die Festlegung muss nicht unbedingt 0 sein (hier wäre 0 nicht anwendbar). In dieser Lösung können Sie auf K zugreifen und mit

`pas = subs(pa,K,[1 2 4 8])` sogar eine Schar von Parabeln definieren.

70–4 Orthogonale Trajektorien zu den Parabeln
Mit dem Ersetzen von y' durch $-1/y'$ in der vorherigen Übung erhält man deren orthogonale Trajektorien: $y' = -x/2y$ (Ellipsen). Lösen Sie diese Differentialgleichung allgemein analytisch und zeichnen Sie einige Vertreter der beiden Kurvenscharen in derselben Grafik. (Unbedingt `axis([-6 6 0 12]);` `axis square` verlangen.)

70–5 Differentialgleichung 3. Ordnung mit konstanten Koeffizienten
Suchen Sie die allgemeine analytische Lösung der Differentialgleichung
$y''' + y'' + y' + y = 1$

70–6 Hyperbolische Lösung
Bei der Differentialgleichung $y^2 = 4(1 + (y')^2)$ liefert MATLAB nur eine Lösung als Ausdruck mit $\mathrm{e}^{f(x)}$-Termen. Dies muss der Anwender eventuell selbst in $\cosh()$ umformen. Bei dieser Differentialgleichung treten zusätzliche triviale Lösungen auf.

70–7 Implizite Lösung
Die mit einem integrierenden Faktor von $1/(x^2+y^2)$ in ein totales Differential transformierbare Differentialgleichung $x+y\cdot y'+(x^2+y^2)\cdot 4y^3\cdot y'$ besitzt keine explizite Lösung der Art $y = f(x)$. Dies wird beim Aufruf von `dsolve` angezeigt und es wird die implizite Lösung $F(x, y)$ angegeben.

Lösungen mit der Laplace-Transformation im Symbolischen Modus
70–8 Radioaktiver Tochterkern, Laplace-Lösung
Lösen Sie das Differentialgeichungs-System
$N' = -\lambda_1 \cdot N, \quad M' = \lambda_1 \cdot N - \lambda_2 \cdot M$
mit der Methode der Laplace-Transformation im Symbolischen Modus von MATLAB.

70–9 Differentialgleichung 2. Ordnung, Laplace-Lösung
Bestimmen Sie mit Hilfe der Laplace Transformation die Lösung der Differentialgleichung $y'' - 4y = \mathrm{e}^{3x}$. Die Anfangsbedingungen seien $y(0) = 1$ und $y'(0) = -1$

70–10 Einfacher Oszillator, Laplace-Lösung
Suchen Sie die Lösung des angeregten Oszillators $y'' + y = \cos(x)$ mit Laplace. Anfangsbedingungen: $y(0) = 2$, $y'(0) = 0$.

70–11 Resonanz-Aufschaukelung, Laplace-Lösung
Das schwingungsfähige System $y''+0.1y'+9y = \sin(3x)$ mit $y(0) = y'(0) = 0$ wird durch die Anregung auf eine Maximalamplitude aufgeschaukelt. Bestimmen Sie die Schwingungsfunktion und die Maximalamplitude mit der Laplace-Methode.

70–12 Oszillator mit Anregung ausser Resonanz, Laplace-Lösung

Der Oszillator it Anregung$y'' + 4y = \sin(3x)$ mit $y(0) = y'(0) = 0$ gerät gegenüber der Anregung immer wieder aus dem Takt. Bestimmen Sie die sich auf- und abbauende Schwingungsfunktion mit der Laplace-Methode.

Numerische Lösung von Differentialgleichungen

70–13 Wegweiserfeld

Eine hervorragende Visualisierung des Lösungsvorgangs bei Differentialgleichungen ist die Suche des Kurvenverlaufes von einem Startpunkt aus, in einem Feld von Wegweisern, welche die lokal gültigen Tangentialrichtungen vorschreiben.
Das Erzeugen eines solchen Wegweiserfeld-Plots ist gleichzeitig eine einfache Programmierübung. (Empfehlung: Einheiten = cm$\Delta x = 25\,\text{cm}$, $\Delta y = 16\,\text{cm}$.
Differentialgleichung:$y' = -0.1 * y$, 2.Fall $y' = -0.3 * y$)

70–14 Einfache Differentialgleichung:

Lösen Sie die Differentialgleichung$y' = -0.2 * y$, $y(0) = 15$ t=0..25
– analytisch
– grafisch, indem Sie in einem Richtungsfeld den Verlauf einzeichnen
– mit MATLAB:

```
yin = 15;
ysol = ode45('decay02',25, yin);
%und dem function-M-file decay02.m
function deri  = decay02(tval,yval);
deri = -0.2* yval;
```

70–15 Euler-Verfahren eindimensional

Programmieren Sie in einem einfachen Skript die Lösung einer radioaktiven Zerfalls-Gleichung$y' = -0.1*y$ nach dem Euler-Verfahren$y_{n+1} = y_n + h \cdot y'_n$ mit konstantem Schritt h.

70–16 Einzelschritte mit dem Euler-Verfahren

Das Differentialgleichungssystem 2. Ordnung eines geladenen Teilchens im konstanten Magnetfeld $\ddot{x} = -0.1 \cdot \dot{y}$; $\ddot{y} = 0.1 \cdot \dot{x}$ mit den Anfangsbedingungen $x(0) = 10$, $y(0) = 0$, $\dot{x}(0) = 0$, $\dot{y}(0) = 1$ muss zuerst in ein System 1. Ordnung transformiert werden. Anschließend sollen drei Schritte nach dem Euler-Verfahren mit Schrittweiten von 0.2 durch Berechnen aller Zwischenresultate „von Hand“ berechnet werden.

70–17 Euler-Verfahren analog zur Bibliotheksprozedur

Mit Hilfe des Funktions-Aufrufes`fresult = feval('prozedurname',parameter1, parameter2)` lässt sich eine Prozedur zum Lösen eines Differentialgleichungssystems erstellen, welche dieselbe Signatur aufweist wie die Bibliotheksprozeduren (z.B. ode45()), die aber nach

dem ganz einfache Euler-Verfahren arbeitet. Der Quervergleich zwischen den Lösungen mit der selbst erstellten Euler-Lösung und der Bibliotheksprozedur ode45 zeigt bei Schwingungs-Differentialgleichungen das interessante Resultat, dass das einfache Verfahren bei genügend kleiner Schrittweite nur wenig vom professionellen Verfahren abweicht, und zwar vorwiegend bei der Phasenlage.

Differentialgleichungen höherer Ordnung

70–18 Umsetzen von Differentialgleichungen höherer Ordnung in Systeme erster Ordnung

Die Differentialgleichung des gedämpften, angeregten Oszillators
$y'' + d * y' + w^2 * y = a * \sin(w * t)$ mit z.B. w=2, d= 0.1, a = 0.5 soll zu den Anfangsbedingungen $y(0) = 0$ und $y'(0) = 0$ mit MATLAB gelöst werden. Beachten Sie besonders die Umsetzung in ein System erster Ordnung durch das Einführen einer neuen Variablen $z = y'$. Die Regeln für das Erstellen des Funktions-M-Files und der Aufruf mit vordefinierten Startwerten sind ebenfalls zu beachten!

```
yin=[0 0]';[tsol ysol] = ode45('dampedosc',[0 20],yin)
```

70–19 Differentialgleichungen höherer Ordnung

Stellen Sie die Differentialgleichungssysteme auf für die folgenden Differentialgleichungen höherer Ordnung:
a) $y'' = -y$
b) $y'' + 0.1 * y' + 25 * y = 0$
c) $y^{IV} = 0$
Lösen Sie diese mit MATLAB für die Anfangswerte $y(0) = 1$ und $y', (y'', y''') = 0$ im Bereich $x = 0..4$

70–20 Freier Fall mit Luftwiderstand

Wenn man der Differentialgleichung für den freien Fall $y'' = -g$,
(y = Höhendifferenz zu Startpunkt, immer stärker negativ werdend, Erdbeschleunigung $g = 9.81$ m/s^2) einen Bremsterm proportional zu v^2 für den Luftwiderstand beifügt $y'' = -g + C \cdot (y')^2$, ergibt sich ein Grenzwert für die Fallgeschwindigkeit, der natürlich stark vom Bremsfaktor abhängt (Fallschirm, Meteorit, Körperform des Luftakrobaten). Experimentieren Sie mit dieser Differentialgleichung bei verschiedenen Bremsfaktoren!

70–21 Oszillator mit Parametereingabe von außen

Verwenden Sie die Möglichkeit, Variablen die als 'global' deklariert sind, um die Eigenfrequenz, die Dämpfung und die Anregungsfrequenz aus dem, den ode-Löser aufrufenden Skript, in die Ableitungsfunktion hinein zu übermitteln. Machen Sie damit verschiedene Versuche über die Wirkung der Dämpfung, und der Differenz zwischen Anregung und Eigenfrequenz.

70–22 Pendel ohne Kleinwinkel-Approximation

Ein Physikalisches Pendel bewegt sich ohne Reibung entsprechend der Differential-

gleichung
$\ddot{\alpha} + g/l_{eff} \cdot \sin(\alpha) = 0$ für den Auslenk-Winkel α. (Die effektive Länge l_{eff} ergibt sich aus dem Ausdruck $\Theta/(m \cdot d)$ mit dem Rotations-Trägheitsmoment Θ, der Gesamtmasse des schwingenden Körpers m und der Distanz d zwischen Drehpunkt und Schwerpunkt.
Lösen Sie diese Gleichung numerisch und vergleichen Sie das Resultat mit demjenigen der Kleinwinkel-Approximation $\sin(\alpha) \simeq \alpha$, also
$\ddot{\alpha} + g/l_{eff} \cdot \alpha = 0$!
(Das reale Pendel schwingt bei grösseren Amplituden langsamer.)

Systeme von Differentialgleichungen

70–23 Schiefer Wurf in 3D

Ein einfaches System von 3 Differentialgleichungen je 2. Ordnung sind die Bewegungen unter alleiniger Wirkung der Schwerkraft $x'' = 0$, $y'' = 0$, $z'' = -9.81$. Die Lösung mit MATLAB ergibt ein System mit 6 Variablen. Die Anfangsbedingungen können x_0, y_0, z_0 alle Null setzen, die Anfangsgeschwindigkeiten (in m/sec) bestimmen dann die Flugbahn.
(z_0' sollte positiv sein und mindestens eines der x_0', y_0' verschieden von 0).

70–24 Radioaktiver Tochterzerfall

Unter der Annahme, dass beim Start N_0 Kerne des Isotopes N mit der Zerfallskonstanten λ_n und keine des Tochterkerns M $M_0 = 0$ mit λ_m vorhanden sind soll das System von gekoppelten Differentialgleichungen
$dN/dt = -\lambda_n N(t)$ und $dM/dt = -\lambda_m M(t) + \lambda_n N(t)$ gelöst werden. (Die beim Zerfall des Typs N entstehenden Tochter-Isotope vom Typ M zerfallen selbst mit der Zeitkonstante λ_m.)

70–25 Geladenes Teilchen in 2D im konstanten Magnetfeld

Ein elektrisch geladenes Teilchen im konstanten Magnetfeld senkrecht zur x-y-Ebene bewegt sich auf einem Kreis. Demonstrieren Sie diese Tatsache durch Lösen des Differentialgleichungs-Systems $\ddot{x} = -C \cdot v_y$ und $\ddot{y} = C \cdot v_x$.

70–26 E $\times$ B -Drift:

Berechnen und zeichnen Sie die Bahnen von geladenen Teilchen unter dem Einfluss eines konstanten Magnetfeldes, das von oben senkrecht in die Tafelebene zeigt und einem simultanen elektrischen Feld in x-Richtung. Zur Beschleunigung senkrecht zur momentanen Geschwindigkeit, verursacht durch das Magnetfeld, wie in der letzten Aufgabe ($\ddot{x} = -C \cdot v_y$ und $\ddot{y} = C \cdot v_x$) kommt eine konstante Beschleunigung in x-Richtung durch das E-Feld hinzu: $\Delta\ddot{x} = k \cdot E$.

70–27 3D-Bahnen unter konstantem Magnetfeld beliebiger Richtung

Bestimmen Sie aus der allgemeinen Formel für die Lorentz-Kraft $F = e \cdot v \times B$ die 3D Bewegung eines geladenen Teilchens in einem beliebigen konstanten Magnetfeld ($[B_x, B_y, B_z]'$) durch Aufstellen, Transformieren und Lösen der Differenti-

algleichung.

70–28 Massenpunkt im Potentialtopf:
Bestimmen Sie die Bahngleichungen einer in einem parabelförmigen, rotationssymmetrischen Topf reibungsfrei gleitenden Kugel und integrieren Sie einige Beispielbahnen!

Traktrix und andere Verfolgungsprobleme
Weitere Information finden Sie unter:
www.solvingproblems.ethz.ch/sample.pdf
www.educ.ethz.ch/unt/um/mathe/gb/Verfolgungsprobleme.pdf

70–29 Klassische Traktrix-Differentialgleichung
Beim Start befinde sich ein Holzklotz bei der Position $(0/a)$ und der wandernde Punkt im Abstand a bei $(0/0)$. Der durch die Schnur konstanter Länge a mit dem wandernden Punkt (x_w/y_w) verbundene Klotz bewegt sich auf einer Traktrix (auch Tractrix), wenn sich der Punkt entlang der x-Achse bewegt.
Die direkte numerische Suche nach einer Funktion $y(x)$ versagt hier wegen $y'(0) = -\infty$.
Die Suche nach einer Parameterdarstellung $x(t)$, $y(t)$, und dem Ansatz $(x_w/y_w) = (v_w \cdot t/0)$, mit dem System $\dot{x} = v_w \cdot (x_w - x)/a$, $\dot{y} = -v_w \cdot y/a$ führt zu einer erfolgreichen numerischen Lösung. Vergleichen Sie diese für verschiedene Wandergeschwindigkeiten v_w!

70–30 Verallgemeinerte Traktrix
Als Verallgemeinerung der klassischen Traktrix kann man den wandernden Punkt auf einer beliebigen Kurve $(x_w(t)\ /\ y_w(t))$ wandern lassen. Zudem kann man die Schnur durch eine Stange ersetzen, so dass der Holzklotz nicht nur gezogen, sondern auch geschoben werden kann.
Die Zeitfunktion des wandernden Punktes kann in derselben Rahmenfunktion eingebaut werden und ist dann im Innern der Ableitungsberechnung aufrufbar. Der Bewegungsvektor des Schleppers $dx_w/dt,\ dy_w/dt$ muss in die Schnur/Stangenrichtung $x_w - x,\ y_w - y$ projiziert werden, um $dx/dt,\ dy/dt$ zu erhalten. (Der Anteil senkrecht zur Schnur dreht nur deren Richtung ohne den Klotz zu bewegen). Bei gegebener Zeitfunktion des wandernden Punktes ergibt sich die Länge a implizit aus dem Startpunkt des geschleppten Holzklotzes.

70–31 Weitere Verallgemeinerung - variable Länge
Wenn dem "geschleppten" Objekt eine eigenständige Geschwindigkeit zugewiesen wird, aber weiterhin in Richtung auf den wandernden Punkt hin, so erhält man die vielerorts beschriebene Jogger-Hund Differentialgleichung bzw. die "Hundekurve" entlang derer ein Jagdhund seinem Meister nacheilt $\dot{x} = v_H \cdot (x_w - x)/|svec|$, $\dot{y} = v_H \cdot (y_w - y)/|svec|$, mit $svec = [x_w - x; y_w - y]$.
Allgemein gehören diese Aufgabenstellungen in die Kategorie Verfolgungsprobleme.

70–32 Das Verfolgungsproblem der vier Käfer

Das Verfolgungsproblem einer Anzahl von Käfern ist, wie die Taktrix, ein uralter Klassiker. Im Beispiel mit vier Käfern sind deren Bahnen gesucht, wenn diese an den Ecken eines Quadrates starten und jeder Käfer mit konstanter Geschwindigkeit auf den nächsten in der zyklischen Abfolge der Ecken zusteuert. Also K1 verfolgt K2, K2 verfolgt K3, etc. bis K4 wieder K1 verfolgt. Programmieren Sie die numerische Lösung dieses Problems und freuen Sie sich an der Ästhetik der Lösungsgrafik!

70–33 Äquitangentiale

Ein Umkehrproblem zur allgemeinen Traktrix ist das Suchen einer Äquitangentiale: Zu einer gegebenen Kurve (Hinterrad-Spur) ist diejenige Kurve (Vorderrad-Spur) gesucht, zu welcher die vorgegebene die Traktrix darstellt. Diese ist nicht durch eine Differentialgleichung gegeben, sondern kann ganz einfach numerisch konstruiert werden, indem man an eine Serie von Punkten der gegebenen Kurve jeweils die Tangenten mit gleichbleibender Länge (Radstand) anfügt. Zeichnen Sie dazu zwei Beispiele: Gerade, Kreisbogen (90 bzw. 180 Grad), Gerade. Vergleichen Sie Ihre Resultate mit der Beobachtung eines Buschauffeurs.

Spline-Interpolationsfunktionen

70–34 Spline-Interpolationsfunktion als Randwertproblem

Die Spline-Funktion erfüllt die Differentialgleichung $y^{(IV)} = 0$. Suchen Sie die analytische Lösung dieser Differentialgleichung. Diese ergibt ein Polynom 3.Grades, das durch die zwei Stützwerte an den Intervallrändern und die zwei Ableitungen an den Intervallrändern eindeutig bestimmt ist (Randwertproblem).

a) Stellen Sie die Gleichungen für die Bestimmung der Polymomkoeffizienten $a_3,\ a_2,\ a_1,\ a_0$ auf! (Beispiel-Werte: $y(0) = 1$, $y(1) = 1.2$, $y'(0) = 0.5$, $y'(1) = -0.7$).

b) **Elementfunktionen:** Bestimmen Sie die 4 Elementfunktionen, zur kubischen Spline-Interpolation (jede hat ihre eigenen 4 Koeffizienten), so dass für jede Funktion eine der Bestimmungsgrößen 1 und die andern drei Bestimmungsgrößen = 0 sind.

Die so gefundenen Funktionen sind die Elementfunktionen für eindimensionale finite Elemente.

70–35 Elementarfunktionen der Spline-Interpolation

Testen Sie die Symmetrieeigenschaften der elementaren Spline-Funktionen

```
B1 [ 1 0 0 0 ], B2 [ 0 1 0 0 ], B3 [ 0 0 1 0 ], B4 [ 0 0 0 1 ]
```

die Werte entsprechen `[y0 y1 y0' y1']`.
Formulieren Sie diese Polynome zuerst als Polynome 3. Grades in 't'.
Zeigen Sie, dass durch Ersetzen von t durch $(1-t)$ B1 und B2, sowie B3 und B4 ihre Rolle vertauschen.
Faktorisieren Sie diese 4 Polynome und suchen Sie den Zusammenhang zwischen den Faktoren und den Nullstellen mit horizontaler oder schräger Tangente.

7.0.1
Miniprojekt zu den Differentialgleichungen

701 Algen- und Fischbestand

Stellen Sie das Differentialgleichungssystem auf für das durch die logistische Gleichung beschriebene Algenwachstum in einem Teich, wobei zusätzlich angenommen wird, dass die Algen als Nahrung für eine große Zahl von kleinen Fischen dienen (Algenfrass proportional zu Algendichte mal Fischdichte). Die Reproduktionsrate der Fische wird dann als proportional zur Algendichte angesetzt und die Sterberate der Fische proportional zum Quotienten Fischdichte/Algendichte. Im Gegensatz zum Jäger-Beute Ansatz der Lotka-Volterra Gleichung kann die Populationsdichte der Fische für deren Nachzucht vernachlässigt werden, da Fische normalerweise einen riesigen Überschuss an Laich produzieren. Beobachten Sie die zeitliche Entwicklung dieses simulierten Systems für verschiedene Werte der Grundparameter zum Algenwachstum, zum Sättigungswert der Algendichte, zur Gefräßigkeit der Fische und zur Reproduktionsrate der Fische in Abhängigkeit von der Algendichte.

7.1
Selbsttests zum Kapitel 7

Testserie 7.1

T711) **Verständnisfragen**

- Wodurch unterscheidet sich ein Differentialgleichungs-System von einem "normalen" linearen Gleichungssystem?
- Wieviele Differentialgleichungen 1. Ordnung umfasst ein System zur Lösung von drei Differentialgleichungen, die jede von 2. Ordnung sind? (z. B. Bewegungsgleichungen eines Massenpunktes im Raum.)
- Die ode-Prozeduren in MATLAB benützen alle eine interne Schrittweitensteuerung beim Suchen der Lösung. Worauf muss man deshalb bei der grafischen Darstellung der Resultate achten?
- Warum muss man beim Erstellen der Funktion zur Bestimmung der Ableitungen einen Eingabeparameter mit der aktuellen unabhängigen Variablen (i.A. der Zeit) in die Parameterliste aufnehmen, auch wenn man diesen Wert bei autonomen Systemen gar nicht benötigt?

T712) Stellen Sie die M-Files zusammen, welche die Differentialgleichung der Entleerung eines vertikal stehenden zylindrischen Tanks von 2.6 m Durchmesser und 3.5 m Höhe durch eine Öffnung von 80 mm Durchmesser beschreiben! Die Ausflussgeschwindigkeit kann mit $\sqrt{2gh}$ angenähert werden. (g Erdbeschleunigung, h Flüssigkeitsstand über Ausfluss)

T713) Erstellen Sie die M-Files zur Beschreibung der Bahnen von elektrisch geladenen Teilchen in der Ebene, zu welcher ein Magnetfeld mit der Stärke proportional zur x-Koordinate senkrecht steht.

T714) Zum folgenden System von Differentialgleichungen

$$\frac{d^2x}{dt^2} = 0.05 \cdot \frac{dy}{dt}$$
$$\frac{d^2y}{dt^2} = -0.05 \cdot \frac{dx}{dt}$$

mit den Anfangsbedingungen $\begin{matrix} x(0) = 0 \\ \dot{x}(0) = 1.0 \end{matrix}$ und $\begin{matrix} y(0) = 20 \\ \dot{y}(0) = 0 \end{matrix}$ soll das zugehörige System 1. Ordnung bestimmt werden. Anschließend sind zwei Lösungsschritte nach dem Euler-Verfahren mit je einer Schrittweite von $dt = 0.2$ durchzurechnen.

T715) Erstellen Sie ein Skript zur Lösung der Differentialgleichung $y'' + 0.1 \cdot y' = 0$ mit dem Paket für symbolisches Rechnen. Die Anfangsbedingungen dazu sind: $y(0) = 10$, $y'(0) = 1$.

Testserie 7.2

T721) **Verständnisfragen**

- Wieviele Anfangswerte sind zur Festlegung der Lösungen einer einzelnen Differentialgleichung 3. Ordnung nötig.
- Warum wird bei der Lösung mit der Laplace-Transformation der Zusatz-Aufwand von Transformation und Rücktransformation in Kauf genommen?
- Was passiert, wenn man versucht, eine steife Differentialgeichung mit konventionellen Lösungsmethoden zu bearbeiten?
- Welcher Typ von Vektor muss als Rückgabe-Parameter bei der Ableitungs-Berechnung definiert werden?

T722) Formulieren Sie ein M-File für die numerische Lösung der Differentialgleichung der Kettenlinie: $y'' = p \cdot \sqrt{1 + (y')^2}$. Für die Startwerte soll der tiefste Punkt $y(0) = 0$; $y'(0) = 0$ gewählt werden.
Formulieren Sie die Ableitungs-Berechung als eingebettete Funktion, so dass als Parameter der Rahmenfunktion verschiedene Werte des Krümmungsparameters p im Scheitelpunkt wählbar sind.

T723) Erstellen Sie die M-Files zur Lösung der Differentialgleichung $y' = -x/y$ mit dem Paket für symbolisches Rechnen.

T724) Formulieren Sie ein Skript zum Lösen der logistischen oder Verhulst Gleichung $y' = c \cdot y \cdot (1 - y/M)$ mit der Methode der Laplace Transformation. Als Anfangsbedingung soll $y(0) = 1$ gewählt werden, und für die Parameter sollen $c = 0.2$ und $M = 20$ eingesetzt werden. (Für die Lösung der algebraischen Gleichungen im Laplace-Raum müssen die Terme `laplace(y(t))` durch einfache symbolische Variablen ersetzt werden.)

T725) Erstellen Sie die M-Files für die numerische Lösung der Differentialgleichung des freien Falls: a) unter Vernachlässigung des Luftwiderstandes, und b) unter Einbezug einer Bremskraft $F = -C \cdot |v|^2 \cdot \vec{v}/|v|$, die proportional zur Geschwindigkeit im Quadrat ist.

8
Übungen zum Kapitel 8

Zusammenfassung grosser Datenmengen

80–1 Median und Quartile für einfache Folge

Bestimmen Sie den Median und die Quartile für die gleichverteilte Folge der ganzen Zahlen von 10 bis 30.

80–2 Median und Quartile bei Sägezahn-Funktion

Berechnen Sie den Median und die Quartile für die Daten mit den Sägezahn-Funktionen als Häufigkeits-Diagramm: 1x1, 2x2, 3x3, 4x4, 5x5, 6x6 (21 Elemente) und 6x6,5x5 etc. (das Spiegelbild).

80–3 Parameter der Differenzbasierten Familie

Für die beiden Sägezahn-Datensätze sollen je die Werte für Mittelwert, Standardabweichung, Schiefe und Exzess berechnet werden. Welche Sägezahn-Funktion hat welche Schiefe?

80–4 Parameter bei einzelnen Datenpunkten

Bestimmen Sie den Median und die Quartile, sowie den Mittelwert und die Standardabweichung für die auf drei Werte verteilten Daten $5 \times (-1)$, 10×0, 5×1

80–5 Quantile von normalverteilten Daten

Erzeugen Sie in mehreren Versuchen mit `randn` jeweils einen normalverteilten Datensatz von 20, 100, und 1000 Werten. Bestimmen Sie für jeden die 20,40, 60 und 80-Percentile.

80–6 Elementare Glockenkurven

Durch die Funktionen $y = \max(0, (1 - x^2))^k$ werden für $k = 1$ die umgekehrte Parabel und für $k > 1$ verschiedene Glockenkurven definiert. Bestimmen Sie für einige dieser Fälle σ (μ ist immer Null) und die dazu passende Gauß'sche Glockenkurve und erzeugen Sie eine gemeinsame Grafik der Kurvenpaare.

80–7 Vergleich zu Normalverteilungs-Dichte

Testen Sie die Nähe bzw. die Abweichung der in 80–6 beschriebenen Glockenkurven

$y = \max(0, (1 - x^2))^k$ zur Normalverteilung mit Hilfe der Funktionen `qqplot` und `normplot`

80–8 Daten in Klasseneinteilung
Erzeugen Sie mit `rand` einen Datensatz mit der Normalverteilung $N(x, 5, 2)$ und 1000 Punkten. Ordnen Sie diese in Klassen der Breite 0.25 ein (binning). Berechnen Sie $< x >$ und s für die Verteilung der Klassenhäufigkeiten. Bestimmen Sie auch s_k mit der Sheppard'schen Korrektur $s_k^2 = s^2 - b^2/12$.

Wahrscheinlichkeitsrechnung

80–9 Buchstaben-Anordnungen
Auf wieviele Arten können die Buchstaben des Wortes 'MAMMAMIA' angeordnet werden?

80–10 Gruppeneinteilung
Auf wieviele Arten können aus 20 Personen vier Fünfergruppen zusammengestellt werden?

80–11 Kursteilnehmer-Zuordnung
Auf wieviele Arten können 15 Kursteilnehmer in Fünfergruppen aufgeteilt und drei Kursleitern zugeordnet werden.

80–12 Halbzeitresultate
Wieviele verschiedene Halbzeitresultate sind bei einem Fussballspiel möglich, das 5:6 endete?

80–13 Tanzpartner
In einem Tanzkurs erscheinen an einem Abend 5 Herren und 7 Damen, so dass eine Dame den Herrenpart übernehmen muss. Wieviele Möglichkeiten von 6 Tanzpaaren auf der Tanzfläche sind denkbar?

80–14 Seltene Münzwurf-Fälle
Bei 15 Münzwürfen nacheinander wird gesucht, wie viel häufiger die Resultate 1 mal Kopf, bzw. zweimal Kopf und sonst immer Zahl sind als das spezielle Resultat 15 mal Zahl.

80–15 Telefonnummern
Wieviele fünfstellige Telefonnummern können in einem Kreis mit derselben Vorwahl vergeben werden, wenn man berücksichtigt, dass für die erste Ziffer 0 und 1 ausfallen.

80–16 Augensumme bei zwei Würfeln
Wieviele Möglichkeiten gibt es, mit 3 Würfeln die Augensumme 10 zu erhalten?

80–17 Fünf Würfel

Beim Würfelspiel "Yahtzee" werden 5 ununterscheidbare Würfel gleichzeitig geworfen. Bestimmen Sie die Wahrscheinlichkeiten, dass beim ersten Wurf a) drei gleiche Augenzahlen oben liegen; b) vier gleiche Augenzahlen oben liegen; c) ein "full house" vorliegt, also drei und zwei zueinander passende Augenzahlen erscheinen. (Die vollständige Diskussion dieses Spiels wird durch die Möglichkeit zum "Verbessern" wesentlich komplizierter. Zweimal dürfen beliebig auswählbare Würfel nochmals geworfen werden.)

Effekt der grossen Zahlen

80–18 Statistische Bestimmung der Zahl Pi

Erzeugen Sie mit `rand` eine grössere Anzahl von Paaren x_k , y_k, je gleichverteilt über $[0 \;\; 1]$. Zählen Sie dabei, wie oft die Bedingung $x_k^2 + y_k^2 < 1$ erfüllt ist (dann liegt der Punkt im Viertelkreis um $(0/0)$). Das Verhältnis innen/total nähert sich so dem Wert $\pi/4$ an.

80–19 Annäherung an den Durchschitt

Bilden Sie mit dem Würfelsimulator mehrere Vektoren der Länge n, in denen die simulierten Augenzahlen stehen. Berechen Sie daraus je den seit dem Start aufgelaufenen Durchschittswert `cumsum(v)./(1:n)`. Für Werte von n über 1000 (bis 10000) ist die sukzessive Annäherung an den Durchschittwert von 3.5 in einer Grafik eindrücklich zu bobachten

80–20 Binomialverteilung zu grossen Zahlen

Erzeugen Sie mit `y = pdf('binomial',x,n,0.5)` mehrere Binomialverteilungen zu Versuchszahlen $n = 10, \;\; 20, \;\; 40, \;\; 80$, alle mit dem Parameter $p = 1/2$. Vergleichen Sie die Resultate jeweils mit Hilfe von `normfit` mit der Normalverteilung.

Statistische Verteilungen

80–21 Abfüllmaschine

Eine Abfüllmaschine für 1 kg-Säcke Mehl weist einen normalverteilten Fehler mit einer Streuung von 4 Gramm auf. Wie hoch muss man den Mittelwert vorwählen, damit weniger als 1% der abgefüllten Säcke einen Inhalt von weniger als 995 Gramm aufweisen. (Zuviel ist erlaubt, da reklamiert der Kunde nicht.)

8.1
Miniprojekt zur Wahrscheinlichkeitsrechnung

801 Ziehungen ohne Zurücklegen

Erstellen Sie ein MATLAB Programm, das den Bayes'schen Baum einer Anzahl Ziehungen ohne Zurücklegen auflistet mit den zugehörigen Wahrscheinlichkeiten für die jeweiligen Farbkombinationen der Kugeln. Als Eingabe sollen die Anzahl Farben (maximal 6, 'rgbcmy') und die jeweilige Häufigkeit in der Startkombination, sowie die Anzahl Ziehungen vorgewählt werden.

8.2
Selbsttest zum Kapitel 8

Testserie 8.1

T811) **Verständnisfragen**

- Wie kann aus einer stetigen Dichtefunktion ein Wahrscheinlichkeitswert gewonnen werden?
- Wieviele Permutationen gibt es für die Buchstaben 'AAAAAB'
- Wie gross ist die Spannweite eines Datensatzes, ausgedrückt im Interquartils-Abstand, nachdem Ausreisser ausserhalb des Faktors 1.5 eliminiert wurden.
- Welcher Prozentsatz aller Werte einer Normalverteilung liegen innerhalb des Intervalls $\mu - 2\sigma..\mu + 2\sigma$?

T812) Bestimmen Sie die Wahrscheinlichkeiten für 0 bis 5 gehaltene Tore bei 5-maligen Elfmeterschiessen falls der Torhüter im Mittel $1/10$ der Schüsse hält. Vergewissern Sie sich, dass die Summe aller Wahrscheinlichkeiten 1 ergibt.

T813) Erstellen Sie eine Grafik mit der diskreten Binomialverteilung für $n = 30$ und $p = 1/2$ und zeichnen Sie die passende Normalverteilungs-Dichte hinein.

T814) Bestimmen Sie die theoretisch möglichen Halbzeit- und Endresultate nach der regulären Spielzeit für ein Fussballspiel, das nach der Verlängerung $5 : 4$ stand. Beachten Sie den Gleichstand nach der regulären Spielzeit!

T815) Zeichnen Sie die Wahrscheinlichkeits-Tabelle (Bayes-Baum) für die Ziehung von zwei Kugeln ohne Zurücklegen bei ursprünglich zwei weissen und zwei roten Kugeln!

Teil II
Lösungshinweise

11
Lösungshinweise zum Kapitel 1

11.1
Im Text eingebettete Übungen

Übung 12–1 Allgemeine nxm Matrix: n*m Freiheitsgrade.
Diagonalmatrix, nxn: n FG.
Obere Dreiecksmatrix nxn:$n * (n+1)/2$ FG.
> Das ist $n^2/2$ plus halbe Diagonale =$n^2/2 + n/2 = n(n+1)/2$.
Gleichzeitig obere und untere Dreiecksmatrix: Diagonalmatrix.
Gewöhnlicher (Spalten-) Vektor ist eine hohe Matrix.
Nebendiagonalen in nxn, je n-1, also total 2*n-2.
"Indicating the size of a matrix, the number of**rows** is **followed** by the number of **columns**."

Übung 12–2 4x5 Matrix: rechts oben (1,5); rechts unten (4,5); links unten (4,1); Mitte der untersten Zeile (4,3).

11	12	13	14	**15**
21	22	23	24	25
31	32	33	34	35
41	42	**43**	44	**45**

Übung 12–3 Eine 4x4 Matrix hat in der Diagonalen die Indizes (1,1), (2,2), (3,3) und (4,4).

Übung 12–4 Transponierte eines Zeilenvektors und einer breiten Matrix:

$$z^T = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} \quad , \quad M^T = \begin{pmatrix} 1 & 10 \\ 2 & 20 \\ 3 & 30 \end{pmatrix}$$

Übung 12–5 $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$ $A^T = \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix}$

$$A + A = 2 * A = \begin{pmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \\ 14 & 16 & 18 \end{pmatrix}$$

Bemerkung: $A + A^T$ ist symmetrisch, $A - A^T$ ist antisymmetrisch.

$A + A^T = \begin{pmatrix} 2 & 6 & 10 \\ 6 & 10 & 14 \\ 10 & 14 & 18 \end{pmatrix}$ $A - A^T = \begin{pmatrix} 0 & -2 & -4 \\ 2 & 0 & -2 \\ 4 & 2 & 0 \end{pmatrix}$ Die gleiche Übung mit MATLAB:

```
A = [1 2 3; 4 5 6; 7 8 9]
A' , A'+A  A-A', A+A
```

Übung 12-6 A'*A und A*A' existieren immer und sind symmetrisch, A*A existiert nur für quadratische Matrizen.

$A * A^T = \begin{pmatrix} 14 & 32 & 50 \\ 32 & 77 & 122 \\ 50 & 122 & 194 \end{pmatrix}$ $A^T * A = \begin{pmatrix} 66 & 78 & 90 \\ 78 & 93 & 108 \\ 90 & 108 & 126 \end{pmatrix}$

$A * A = \begin{pmatrix} 30 & 36 & 42 \\ 66 & 81 & 96 \\ 102 & 126 & 150 \end{pmatrix}$ Die gleiche Übung mit MATLAB:

```
A = [1 2 3; 4 5 6; 7 8 9]
A'* A, A*A', A*A
```

Übung 13–1 – Diagonalmatrix:

```
function d1n = getdiag1n(ndim)
  d1n = zeros(ndim);  % erzeugt Nullmatrix ndim x ndim
  for k=1:ndim        % Einfuellen 1:ndim in Diagonale
    d1n(k,k) = k;
  end
end  %function
```

Übung 13–2 – Indexwertmatrix:

```
function iwm = indwertmatq(ndim)
% Indexwert-Matrix quadratisch
% Element-Werte (zweistellig) zeigen Indizes an
  iwm = zeros(ndim);
  for zei=1:min(ndim,9) % Absicherung: ndim einstellig
% Hinweis: zei ist 'maximal' 9 heisst zei = min(ndim,9)
%   also Begrenzung auf Maximalwert mit min()
    for spa = 1:min(ndim,9)
      iwm(zei,spa) = 10*zei + spa;
    end
  end
end %function
```

Übung 13–3, Erstellen einer Distanztabelle

```
% makedisttab.m
v = [4634 4478 3820 1620 678]
% repetiere v 5 mal  vertikal, 1 mal horizontal
H = repmat(v,5,1); % jede Spalte enthält Hoehe 5 mal
Dtab = H - H'  % antisymmetrisch! meist nur unterer Teil
% oder auch
onesvert = ones(5,1);
G = onesvert*v;   % Dyadisches Produkt
Dt2 = G - G'
```

Übung 15–1

```
% polygonimkreis
%  zuerste alle im Einheitskreis
for eck = 3:12
  wtab = (0:eck)*2*pi/eck + pi/2;
  plot(cos(wtab),sin(wtab)) ;  hold on
end
axis equal; hold off;
disp('<ret> druecken zum weiterfahren'), pause
%  dann alle mit Seitenlaenge 1
rvec = 0.5/sin(pi./(3:12)); % r damit Seite=1
for eck = 3:12
  wtab = (0:eck)*2*pi/eck + pi/2;
%  Arrays beginnen bei 1, Seitenzahl aber bei 3
%     also ind = eck-2
  plot(rvec(eck-2)*cos(wtab),rvec(eck-2)*sin(wtab)) ;
  hold on
end
axis equal; hold off
```

11.2
Lösungen der allgemeinen Übungen zum Kapitel 1

Einfache Berechnungen

10–1 Die Erde in Zahlen

r[m] = 40000*1000/pi/2 = 6.366E06, V[m3] = 1.0808E21 rho [t/m3] = 5.97E21/1.0808E21 = 5.52

10–2 Mond und Sonne

w = 30/360/60*2*pi = 0.009018, (Winkel in Radian)
DL = w*384400 = 3466 km, (Monddurchmesser)
DH = w*150E06 = 1.35E06 km (Sonnendurchmesser)

10–3 Der Eiffelturm

r=sqrt(2)*160/2 = 162.6 [m], (Radius umschreibender Zylinder)
V=r*r*pi*320 = 1.26E07 [m3], (Luftvolumen)
MLuft = V*0.0012 = 15’ 400 t (Masse der Luft) > MEisen

10–4 Physikalische Arbeitsleistung eines Bergwanderers

P = m*g*h/t = 75*9.81*300/3600 = 61.3 W (Vergleich: Hometrainer-Einstellung je nach Fitness 80-200 Watt, 1 PS = 750 W)

10–5 Wieviele Sekunden hat ein Jahr?

sd = md*60 = 86400, mw = md*7 = 10080,
sw = mw*60 = 604800, my = md*365 = 525600, sy = my*60 = 31536000,
hw = 24*7 = 168, hy = 365*24 = 8760

10–6 Wie weit kommt ein Tanklaster mit seiner eigenen Ladung?

20000/15*100 = 133’ 333 km, mehr als 3 mal um die Erde.

10–7 Berechnungs-Spass beim Essen

300 cm^3 / (0.6 * 0.6) cm^2 = 833 cm = 8 Meter Pommes frites.
300/(0.1*0.1*pi) = 9549 cm = 95 m Spaghetti, also etwa 100 Meter.

10–1 bis 10–7 als M-File

```
% rechenbeisp1to7
%% 10-1 Erde in Zahlen
Ukm   = 40000;
rmet  = Ukm*1000/(2*pi)  %      = 6.3662e+06
Vm3   = 4*pi/3*rmet^3    %      = 1.0808E21
Mston = 5.97e21
rho   = Mston/Vm3        % rho  =  5.5239
%% 10-2 Mond und Sonne
Dmkm = 384400
Dskm = 150e6
Siwimin = 31
Siwideg = Siwimin/60       % = 0.5167
format shortE
Siwirad = Siwideg*pi/180   %  = 9.0175e-03
DiamMkm   = Siwirad*Dmkm   %  = 3466 km(=0.27*DiamErde)
DiamSkm   = Siwirad*Dskm   %   =  1.35e+06
%% 10-3 Eiffelturm
Zylrad = sqrt(2)*160/2         % 113.2
Zylh   = 320
Zylvolm3 = Zylrad^2*pi * Zylh  %   1.2868e+07
rholuft = 0.0012               % t/m3
ZylMasse = Zylvolm3*rholuft    %   1.5442e+04 = 2*7000
```

```
%% 10-4 Bergwanderer hier alles in SI Einheiten (mks)
format short
Mw = 75  ; g = 9.81 ;  H = 300;
delE = Mw*g*H      %
T = 1 * 60 * 60    % = 3600
P = delE/T         % = 61.3  Watt
%% 10-5 Sekunden, Minuten, Stunden, Jahre
md = 24*60   % =     1440
sd = md * 60 % =   86'400
mw = 7*md    % =   10'080
sw = mw * 60 % =  604'800
my = md*365  % =  525'600
sy = my * 60 % = 31'536'000
slife = sy * 80 % = 2.5e09 = 2.5 Milliarden
hw = 24*7    % = 168
hy = 24*365  % = 8760
%% 10-6 Tanklaster
Vol = 20000   % Ladung; 20 Tonnen/Liter
Verbr = 15    % Liter/100 km
Dist = Vol/Verbr *100   % 1.33e05 = 133'000 km
133000/40000  %  = 3.3 mal um die Erde
%% 10-7 Essen
Vol = 300 % Kubikcentimenter, etwa gleich Gramm
Qfrit = 0.6*0.6        % =  0.36  cm2
Lfrit = Vol/Qfrit      % =  833 cm = 8.3 m
Qspag = 0.2*0.2*pi/4   % D*D*pi/4 =  0.0314 cm2
Lspag = Vol/Qspag      % = 9549 cm = 95 m
```

Produktschreibweise mit '*'

10–8 Binomische Formeln

Als Beispiel:

```
a^4 - 4*a^3*b + 6*a^2*b^2 - 4*a*b^3 + b^4
```

```
syms a b
bino = a^4 - 4*a^3*b + 6*a^2*b^2 - 4*a*b^3 + b^4
factor(bino)
```

Implizite Schleifen und Summen

10–9 Summen von natürlichen Zahlenreihen

Beispiel: Summe 1..99 = sum(1:99) = 99*100/2 = 4950

```
syms a b d
symsum(a:d:b)
```

10–10 Summen von allgemeinen arithmetischen Reihen
Beispiel: Summe 20:2:40 = 330 = 11*60/2
Und nun nur fröhlich weiter experimentieren!

10–11 Summenwert von magischen Quadraten
SMQL(3) = 15, SMQL(4) = 34, SMQL(6) = 111, SMQL(9) = 369

Matrixdefinition
10–12 Jedes Element hat seinen Platz!
Z01 = [1 2 ; 3 4], Z02 = [1 2 ; 4 3] , Z03 = [1 3; 2 4], Z04 =[1 3 ; 4 2], etc.
Z24 = [4 3; 2 1] , z. B. Z24 = flipud(filplr(Z01))

10–13 Zeilen und Spalten
z. B. V = reshape(Z',3,4)'

```
%reshapetry  Test der reshape-Funktion
format compact
E = 1:12, C = E'
Z = reshape(E,2,6)
D = reshape(E,3,4)
V = reshape(E,4,3)
V2 = reshape(D,4,3)  % V, V2 identisch
S = reshape(E,6,2)
S2 = reshape(C,6,2)  % S,S2 identisch
cmp6 = S == Z'       % nur 1. und letztes Element
cmp3 = D == V'       % sind gleich!
```

10–14 Matrizen als Bestandteile von Matrizen

```
Q = [ 1 0; 0 1], H = [Q Q; Q Q], S = [ H H; H H]
```

10–15 Vektoren zu Matrizen zusammenfügen

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad R = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

10–16 Matrizen aneinanderfügen

```
I6=eye(6), kurz = zeros(1,6), lang = zeros(1,7)
    R7 = [ kurz' I6; lang]
    L7 = [lang; I6 kurz']
```

10–17 Würfelkoordinaten

```
A = [0 0 0]' ; B = [1 0 0]'; C = [1 1 0]'; D = [0 1 0]';
E = [0 0 1]' ; F = [1 0 1]'; G = [1 1 1]'; H = [0 1 1]';
Wl = [A B C D A E F B F G C G H D H E];
plot3(Wl(1,:),Wl(2,:),Wl(3,:))
axis equal
```

Mit der Schaltfläche mit dem runden Pfeil und anschließender Cursorbewegung die Kameraposition variieren!

10–18 Einmaleins-Tabelle

```
  P=[] ;            % zuerst ist P leer
  for k=1:10
    P = [P ; k:k:10*k]; % neue Zeile unten angefuegen
  end
```

10–19 Größter gemeinsamer Teiler, kleinstes gemeinsames Vielfaches

```
% ggtkgv.m  Test der Formel ggT*kgV = r^2
% Funktionen ggT: gcd greatest common divider
%            kgV: lcm least common multiple
v = [ 4     7    13    20    36    49    64    72 ];
e = ones(1,length(v));
H = v'*e        % dyadisches Produkt
T = gcd(H,H'),  V = lcm(H,H')  % beide symmetrisch!\\
T .* V - v'*v      % (= Nullmatrix)
```

Fachausdrücke zur Matrizentheorie

10–20 Welche der folgenden Aussagen sind wahr? – Ja, eine symmetrische Matrix ist quadratisch.
– Ja, denn durch Transponieren wechselt das Vorzeichen.
– Nein, sondern das Neutralelement der Addition, NE der Multiplikation ist I.
– Nein, eine Diagonalmatrix kann auch in der Diagonalen Nullen haben, dann ist sie singulär.
– Ja, (nxn)*(nx1) = (nx1), der Resultatvektor hat dieselbe Länge.
– Ja, (nxm)*(mxn) ist immer möglich und ergibt (nxn). Der ungewöhnlichste Fall (nx1)*(1xn) ergibt aus zwei Vektoren eine Matrix und heißt „dyadisches Produkt".
– Ja, die Definition einer symmetrischen Matrix lautet gerade M' = M.

10–21 Zerlegung in symmetrischen und antisymmetrischen Anteil

```
 T = rand(4)
 S = (T + T')/2
 A = (T - T')/2
Chk =  (T + T')/2 + (T - T')/2
%    = 2*T/2 = T
```

10–22 Charakteristische Eigenschaften von speziellen Matrizen
S: $n*(n+1)/2$ Fg.; A: $n*(n-1)/2$ Fg.;
Nl: 0 Fg.; D: n Fg.; I: 0 Fg.;
R: $n*(n+1)/2$ Fg.; L: $n*(n+1)/2$ Fg.
Nl ist D, Nl ist S, Nl ist A,
D ist S, I ist S, I ist D, D ist R, D ist L.

Hilfsfunktionen zum Erzeugen von Matrizen

10–23 zeros(), ones(), eye()
z.B. `A = ones(5) - eye(5)`

10–24 diag()

```
Di = diag(1:2:29)
```

10–25 Block-Diagonalmatrizen

```
n = 4; z = [ eye(n) eye(n) eye(n)]
B= [z ; z; z]; % oder
Bb = repmat(z,3,3); spy(B)
```

10–26 Quadrieren von magischen Quadraten

Beispiel für n=3 im File magquadquad.m

```
% magquadquad.m Quadrieren mag. Quadrat M(3)
M=magic(3) ; MM= [];
for k=1:3
  H = [ (M(k,1)-1)*9*ones(3)+M ...
        (M(k,2)-1)*9*ones(3)+M ...
        (M(k,3)-1)*9*ones(3)+M];
  MM = [MM; H];
end
MM
% Test der Zeilen- und Spaltensummen
sum(MM)
sum(MM')
```

Logische Operatoren angewandt auf Matrizen

10–27 Invertiertes Schachbrettmuster

```
 Z = [1 0 ; 0 1]; V = [Z Z; Z Z]; S = [V V; V V], Si = S==0
```

10–28 Negativbild durch logische Operatoren

```
E = zeros(7); E(6,3:6)=1;
 E(4,3:5)=1;  E(2,3:6)=1; E(2:6,2)=1, N= E==0
```

Übungen zum Programmieren von Schleifen

10–29 Indexwert-Dreiecksmatrix

```
function rimat = iwmatr(ndim)
% function rimat = iwmatr(ndim)
% Rechts-Dreiecks-Matrix mit Index-Werten
  rimat = zeros(ndim)
  for spa = 1:min(ndim,9)  % alle Spalten
    % (min: Absicherung gegen zweistellig)
    for zei = 1:spa   % nur Zeilen bis Diag. (spa,spa)
      rimat(zei,spa) = 10*spa + zei;
    end
  end
end %function
```

10–30 Füllen einer Dreiecksmatrix

```
function rones = triuones(ndim)
% Rechts-Dreiecks-Matrix mit Einsen fuellen
  rones = zeros(ndim);
  for zei = 1:ndim  % alle Zeilen
    for spa = zei:ndim   % nur Spalten ab Diag (zei,zei)
      rones(zei,spa) = 1;
    end
  end
end %function
```

10–31 Tridiagonalmatrix

```
function trd = tridiag2m1(ndim)
% function trd = tridiag2m1(ndim)
%     Tridiagonal mit 2 und -1
  trd = 2*eye(ndim) ;
  for k = 1:ndim-1  % Nebendiagonalen um 1 kuerzer
    trd(k,k+1) = -1;
    trd(k+1,k) = -1;
  end
end %function
```

10–32 Streifenmuster

```
function pardiag = pardiagodd(ndim)
% function pardiag=pardiagodd(ndim)
% Streifenmuster-Matrix
  pardiag = eye(ndim) ;
  for dist = 2:2:ndim
% horizontal gesehen: Distanzen 2,4,6 etc.
    for k = 1:ndim-dist
% Parallelen zur Diagonalen um dist kuerzer
      pardiag(k,k+dist) = 1;
      pardiag(k+dist,k) = 1;
    end
  end
%  Test eventuell mit spy(pardiag)
end %function
```

10–33 Spezielle untere Dreiecksmatrix

```
function trdsum = tridsum(ndim)
%  function trdsum = tridsum(ndim)
%  Links-Dreiecks-Matrix mit Indexsummen fuellen
  trdsum = zeros(ndim);
  for zei = 1:ndim  % alle Zeilen
    for spa = 1:zei % nur Spalten bis Diag (zei,zei)
      trdsum(zei,spa) = zei+spa;
    end
  end
end %function
```

Turm und Spechtmatrizen

10–34 „Spechtmatrizen“ – Effekt der Matrixmultiplikation
Beachten Sie: “innerer Index selektiert, äusserer Index platziert”.

```
% specht4try  Test der Spechtmatrix-Wirkung
v = 1:16; M = reshape(v,4,4), format compact
S11 = zeros(4); S11(1,1)=1;
L11 = S11*M, R11 = M*S11
S13 = zeros(4); S13(1,3)=1;
L13 = S13*M, R13 = M*S13
S41 = zeros(4); S41(4,1)=1;
L41 = S41*M, R41 = M*S41
S43 = zeros(4); S43(4,3)=1;
L43 = S43*M, R43 = M*S43
```

Die anschliessend verwendeten Funktionen sind
Spechtmat:

```
function spm = spechtmat(ndim,ezei,espa)
  spm = zeros(ndim);
  spm(ezei,espa) = 1;
end % end function
```

quadratische Indexwert-Matrix

```
function iwmq =  indwertmatq(ndim)
  iwmq = zeros(min(ndim,9));
  for zei = 1:ndim
    iwmq(zei,:) = 10*zei + (1:ndim);
  end
end % end function
```

10–35 Spechtmatrizen – Zeilenselektion

```
format compact; M5 = indwertmatq(5)
Z1p1 = spechtmat(5,1,1)*M5
Z1p2 = spechtmat(5,2,1)*M5
Z1p4 = spechtmat(5,4,1)*M5
Z1p5 = spechtmat(5,5,1)*M5
%
Z4p1 = spechtmat(5,1,4)*M5
Z4p2 = spechtmat(5,2,4)*M5
Z4p4 = spechtmat(5,4,4)*M5
Z4p5 = spechtmat(5,5,4)*M5
%
Z5p1 = spechtmat(5,1,5)*M5
Z5p2 = spechtmat(5,2,5)*M5
Z5p4 = spechtmat(5,4,5)*M5
Z5p5 = spechtmat(5,5,5)*M5
```

10–36 Spechtmatrizen – Spaltenselektion:
"innerer Index selektiert, äusserer platziert".

```
format compact; M5 = indwertmatq(5)
S1p1 = M5*spechtmat(5,1,1)
S1p2 = M5*spechtmat(5,1,2)
S1p4 = M5*spechtmat(5,1,4)
S1p5 = M5*spechtmat(5,1,5)
%
S4p1 = M5*spechtmat(5,4,1)
S4p2 = M5*spechtmat(5,4,2)
S4p4 = M5*spechtmat(5,4,4)
S4p5 = M5*spechtmat(5,4,5)
%
S5p1 = M5*spechtmat(5,5,1)
S5p2 = M5*spechtmat(5,5,2)
S5p4 = M5*spechtmat(5,5,4)
S5p5 = M5*spechtmat(5,5,5)
```

10–37 Durch Permutationsangabe definierte Turmmatrix
Permutationsmatrix (Turmmatrix) aus Permutationsvektor

```
function [Mp, Tm] = permvectomat(Mo,v)
% function Mp = permvectomat(Mo,v)
%  Mo, Mp Original, Zeilen-permutierte Matrix
%  (Fuer Spaltenpermutation  Sp = So*Tm')
%  v Vektor der Permutation
% (Dimensionen nicht kontrolliert)
  Tm = zeros(length(v));
  for k=1:length(v)
```

```
    Tm(k,v(k)) = 1;
  end
  Mp = Tm*Mo;
end %function
```

10–38 Scroll-up- und Scroll-down-Matrizen

```
% scrollupdown  (10-38)
format compact;
Scrup5 = [[zeros(4,1) eye(4)] ; zeros(1,5)]
Scrup5z = Scrup5; Scrup5z (5,1)= 1
ru5 = rank(Scrup5),ru5z = rank(Scrup5z)
Scrdw5 = [zeros(1,5) ; [eye(4) zeros(4,1)] ]
Scrdw5z = Scrdw5; Scrdw5z (1,5)= 1
rd5 = rank(Scrdw5),rd5z = rank(Scrdw5z)
M5 = indwertmatq(5)
M5u = Scrup5*M5
M5uz = Scrup5z*M5
M5d = Scrdw5*M5
M5dz = Scrdw5z*M5
```

10–39 Die Wirkung von Turmmatrizen

Rechts-Links Multiplikation von Turm-Matrizen (10–38 vorher ausführen)

```
M5Sl = Scrup5*M5
M5Sr = M5*Scrup5  % andere Spaltenpermutation
M5Srt = M5*Scrup5'  % Scroll-left
M5Szl = Scrup5z*M5
M5Szr = M5*Scrup5z
M5Szrt = M5*Scrup5z' % Scroll-left zyklisch
M5Sdzl = Scrdw5z*M5
M5Sdzr = M5*Scrdw5z
M5Sdzrt = M5*Scrdw5z' % Scroll-right zyklisch
```

10–40 Die Wirkung von Auswahl- und Permutationsmatrizen

```
A = indwertmatq(4);
Pl = [0 0 0 0; 1 0 0 0; 0 0 0 1; 0 0 0 0]
Pr = [0 0 0 1; 0 0 1 0; 0 1 0 0; 1 0 0 0]
Pl*A*Pr %Test ob vorgesehene Vertauschungen erfolgen
```

10–41 Turmmatrizen zu elementaren Permutationen

```
function B = elemperm(ndim, j, k)
```

```
%  function B = elemperm(ndim, j, k)
%  elementare Permutationsmatrix, vertauscht nur j,k
  B = eye(ndim); B(j,j) = 0; B(k,k) = 0;
  B(j,k) = 1; B(k,j) = 1;
end  %function

% Anwendungs-Test, separates M-File oder direkt
v = (1:4)';  vp = elemperm(4,1,4)*elemperm(4,2,3)*v
```

Multiplikation von Matrizen

10–42 Matrixmultiplikation

1. Produkt: [13 12 11; 23 22 21; 33 32 31] ;
2. Produkt: [11 23 36; 21 43 66; 31 63 96]

10–43 Indexwertmatrix mal allgemeine Matrix

Beispiel: R = [11*a+12*c, 11*b+12*d; 21*a+22*c, 21*b+22*d]

```
syms a b c d e f g h i
Z = [a b ; c d]
D = [a b c; d e f ; g h i]
M2 = [11 12; 21 22]
M3 = [11 12 13; 21 22 23; 31 32 33]
MZ = M2*Z, ZM =  Z*M2
MD = M3*D, DM = D*M3
```

10–44 Matrixmultiplikation „von Hand“

Zw*Zw = [17 10; 15 22] ; Dr*Dr = [30 36 42; 66 81 96; 102 126 150]

10–45 Formel für die Inverse einer 2x2 Matrix

```
syms a b c d u v w x
sol = solve(a*u+b*w == 1,a*v+b*x == 0, ...
            c*u+d*w == 0 , c*v+d*x == 1, ...
                u,v,w,x)
sol.u, sol.v, sol.w, sol.x
% D = a*d-b*c ; u = d/D; v = -b/D; w = -c/D; x = a/D
```

10–46 Legalitätsüberlegung bei Matrizen- und Vektormultiplikationen

legal; A+A, B+B, A*B, B*A’, A*v, v*A’, v*v’, v’*v, B*v, A’*A, A*A’ B*B

illegal: A+B, v+w, w-v, B*A, A’*B, v*A, w*A’, B*w, A’*v, A*A

10–47 Legale und illegale Matrixmultiplikationen

legal: alle Produkte mit eigener Transponierten

illegal: alle Produkte mit eigener Originalmatrix

legal: E*S, S*E, E’*S’, S’*E’, Z*D, Z’*D’, D*Z, D’*Z’

```
function [zei,spa] = dimiflegal(L,R)
% function [zei,spa] = dimiflegal(L,R)
% Produktdimension zweier Matizen, falls P. legal
[z1,s1] = size(L);  [z2,s2] = size(R);
zei = 0; spa = 0;
if  s1 == z2
    zei = z1 ; spa = s2;
end

%  in separatem File
% legmulttest.m  Test auf legale Multiplikation
%  cell array kann verschiedenartige Dinge enthalten
E = 1:6; S = E'; Z = reshape(E,2,3); D = Z';
Et = E'; Zt = Z'; Dt = D'; St = S';
divm = cell(1,8);  dimmat = zeros(8);
divm{1} = E; divm{2} = Et; divm{3} = Z; divm{4} = Zt;
divm{5} = D; divm{6} = Dt; divm{7} = S; divm{8} = St;
for zei = 1:8
    for spa = 1:8
        [ze,sp] = dimiflegal(divm{zei},divm{spa});
        dimmat(zei,spa) = 10*ze+sp
    end
end
dimmat  % 0 heisst illegal, 33 heisst 3x3
```

10–48 Legale und illegale Matrizen- und Vektor-Multiplikationen

legal: w*A, A'*w', w*w', w'*w, v'*v, v*v',N*v, v'*N, A*N, A*N',

10–49 Nachprüfen einer Matrizengleichung

$[ab; cd] * [d - b; -ca] = [a * d - b * c, 0; 0, -c * b + d * a]$

Regel 2x2 Matrizen: Diagonalelemente wechseln Platz, die andern das Vorzeichen, dann alles durch det(M) dividieren.

Formulieren von Gleichungssystemen in Matrizenform

10–50 In einzelnen Gleichungen vorgegebenes Gleichungssystem

```
 M = [ 1 1 1 1; 1 -1 0 0; 0 -1 1 1; -1 0 1 0]
 b= [0 1 -1 1]';
 x=M \ b            %    = [ 1 0 2 -3]'
```

10–51 Gleichungssystem, durch einzelne Gleichungen gegeben

```
M = [ 1 -2 0 0; 0 1 1 0; 1 0 0 -1; 1 -3 0 1]
 b = [ 0 5 2 1]'
```

```
x = M \ b      %  =   [  6  3  2  4]'
```

10–52 In einem Schema vorgegebenes Gleichungssystem

```
  M = [ 1 -1 0 0; 0 1 -1 1; 0 0 1 -1; 1 0 0 1]
  b= [5 3 1 3]'
  x=M \ b    % = [ 9 4 -5 -6]'
```

10–53 Gleichungssystem in gewohnter Form

```
  M = [ 3 1 -1;  1 0 2;  0 1 -1]
  b= [7 3 0]'
  x=M \ b      %   = [2.333 0.333 0.333 ]'
```

Skript 10-50 bis 10-53

```
% lingl50to53
%% Uebung 10-50
  M50 = [ 1 1 1 1;  1 -1 0 0;  0 -1 1 1;  -1 0 1 0]
  b50 = [0 1 -1 1]';
  x50 = M50\b50   % = [ 1 0 2 -3]'
%% Uebung 10-51
  M51 = [ 1 -2 0 0; 0 1 1 0; 1 0 0 -1; 1 -3 0 1]
  b51 = [ 0 5 2 1]' ;
  x51 = M51 \ b51  %  =   [  6  3  2  4]'
%% Uebung 10-52
  M52 = [ 1 -1 0 0; 0 1 -1 1; 0 0 1 -1; 1 0 0 1]
  b52 = [5 3 1 3]';
  x52 = M52 \ b52 % = [ 9 4 -5 -6]'
%% Uebung 10-53
  M53 = [ 3 1 -1;  1 0 2;  0 1 -1]
  b53 = [7 3 0]';
  x53 = M53 \ b53   % = [2.333 0.333 0.333 ]'
```

Einfache Funktionsplots

10–54 Plot von geraden Potenzfunktionen

```
x=0:0.01:2 ; y1=x.^2; y2=y1.^2;
y3=y2.^2;  y4=y3.^2;
plot(x,y1,'r'); hold on;
axis([0 2 0 2]) ; axis square;
plot(x,y2,'g');  plot(x,y3,'b');  plot(x,y4,'k');
```

10–55 Gerade und ungerade Potenzfunktionen

```
 x=-2:0.01:2 ; y1=x.^2; y2=x.^3;
 y3=x.^4; y4=x.^5;
 plot(x,x,'r'); hold on;
 axis([-2 2 -2 2]) ; axis square;
 plot(x,y1,'k');  plot(x,y2,'r');
 plot(x,y3,'k');  plot(x,y4,'r');
```

10–56 Verschiedene Wurzelfunktionen

```
 x=0.01:0.01:4 ; y2=sqrt(x);
 y3=x.^(1/3); y4=x.^(1/4);
 plot(x,x,'r'); hold on;
 axis([0 4 0 4]) ; axis square;
 plot(x,y2,'g');  plot(x,y3,'b');  plot(x,y4,'k');
```

Skript 10–54 bis 10-56

```
% simpgra54to56
%% Uebung 10-54
 x=0:0.01:2 ; y1=x.^2; y2=y1.^2;
 y3=y2.^2; y4=y3.^2;
 plot(x,y1,'r'); hold on;
 axis([0 2 0 2]) ; axis square;
 plot(x,y2,'g');  plot(x,y3,'b');
 plot(x,y4,'k'); hold off
pause
%% Uebung 10-55
x=-2:0.01:2 ; y1=x.^2; y2=x.^3;
y3=x.^4; y4=x.^5;
plot(x,x,'r'); hold on;
axis([-2 2 -2 2]) ; axis square;
 plot(x,y1,'k');  plot(x,y2,'r');
 plot(x,y3,'k');  plot(x,y4,'r'); hold off
pause
%% Uebung 10-56
x=0.01:0.01:4 ; y2=sqrt(x);
y3=x.^(1/3); y4=x.^(1/4);
plot(x,x,'r'); hold on;
axis([0 4 0 4]) ; axis square;
plot(x,y2,'g');  plot(x,y3,'b');
plot(x,y4,'k'); hold off
```

MATLAB-Operatoren und Grundfunktionen

10–57 MATLAB-Operatoren

Siehe Zusammenstellung der MATLAB-Operatoren. Vergleichen Sie erst nach Ihrer eigenen Aufzeichnung!

10–58 Spiegelungsoperationen an Matrizen

Zum Beispiel `fliplr(flipud(M'))`

Erarbeiten einer Funktion

10–59 Winkelfunktionen in Grad

Winkelfunktionen in Grad erhalten ein 'd' angehängt sind(), cosd() etc. Der Versuch, das selbst zu programmieren ist aber recht lehrreich, weil der Input auch eine Matrix sein kann.

```
function sd = sindeg(w)
sd = sin(w*pi/180);
% w-input und w*pi/180 gleiche Dimension
```

10–60 Direkt aufrufbare Potenzfunktionen

```
function p2 = pow2(x)
% function p2 = pow2(x) elementweise Quadrieren
p2 = x.^2
end % function
```

10–61 Teilvolumen in Pyramiden-Trichter

```
function vtricht = pvoltricht(h)
% function vtricht = pvoltricht(h)
vtricht = 4/3*h^3;
end % function
```

10–62 Teilvolumen in liegendem Zylinder

Teilvolumen liegender Zylinder

```
function tvzyl = teilvolzyl(R,L,h)
% function tvzyl = teilvolzyl(R,L,h)
heff = min(abs(h),2*R); % Absicherung
x = -R+heff; % h=0 -> x = -R
y = sqrt(R.^2-x.^2); % Breite Wasserspiegel/2
w = acos(-x./R); % Zentrumswinkel Fluessigkeitsrand
tvzyl = L*2*(R.^2.*w/2 + x.*y/2);
%   L * 2 (  halber Sektor  -/+ Dreieck)
end % function
```

Lösungen zum Miniprojekt 101 Fadenstern

```
xi = [-10:10]; yi = zeros(1,21);
xf = zeros(1,21); yf = [0:10 9:-1:0];
mx = [xi xi ; xf xf] ; my = [yi yi ; yf -yf]
plot(mx,my,'k')
```

Lösung zum Miniprojekt 102 Farbkreis

```
%   Skript   Farbkreis
%  Mitte gesaettigte Farben.
% gegen Innen grau-Anteile,
%  gegen aussen  aufgehellt
sat = [0.4 0.8 1 0.8 0.4];
val = [0.4 0.8 1  1   1];
%
for ri = 2:2:10
  for huebas = 0:1/24:23/24
    rpat = [ri ri ri+2 ri+2];
    wpat =90-360*([huebas huebas+1/24 ...
           huebas+1/24 huebas]-1/48);
    xpat = rpat.*cosd(wpat); ypat = rpat.*sind(wpat);
    [r,g,b] = hsv2rgb(huebas,sat(ri/2),val(ri/2))
    patch(xpat,ypat,[r g b])
  end
end  %  for ri
axis equal
```

11.3
Lösungen zu den Selbsttests Kapitel 1

Lösungen zur Testserie 1.2

T111 – Punkt-Operatoren verlangen elementweise Verarbeitung,
es gibt sie für `'*' '/' '\' '^'`.

```
-eye(4) -plot(x,y,'k') % fuer blac'k' -zei4=M(4,:)
```

T112 `t = (0:0.01:3)*2*pi ; plot(t,sin(t)); axis equal`

T113 `Z=[1 0; 0 1]; V=[Z Z; Z Z] ; S=[V V; V V]`

T114 `p=-10:0.2:10; s=[p p p p p]`

T115

```
M=[1 0 1; 0 1 1; -2 -4 0]; b=[4 2 2]'; x=M\b
%  x = [1 ;-1 ; 3]
 clear all ; syms x y z
 sl = solve(x+y==4, y + z == 2, 2*x-4*y == 2, x, y, z)
 sl.x, sl.y, sl.z~
```

Lösungen zur Testserie 1.2

T121 `-xsol = A`

```
-syms a b; expand((a-b)^5) -hold on
-for zei=1:3; for spa=1:3; E3(zei,spa)=0;end;end;
for zei = 1:3; E3(zei,zei) = 1; end; E3
```

T122 `x=0.02:0.02:3; plot(x,x.^(1/2),x,x.^(1/3),x,x.^(1/4))`

T123

```
load tempmess.dat; D=tempmess(:,2:4);
 plot(tempmess(:,1),D)
```

T124

```
for k=1:201; if y(k) > 0.6; y(k) = 0.6;
  elseif y(k) < -0.6; y(k) = -0.6;  end; end
```

T125 `B*C= [4 13 1; 10 32 2; 0 10 0]`

12
Lösungshinweise zum Kapitel 2

12.1
Im Text enthaltene Übungen

Übungen zu Relationen und Funktionen

Übung 21–1 Bereich $x = -10 : 10$, $y = 0 : 10$ bzw. $y = -10 : 10$

```
figure(1); clf
xv = -10:0.2:10; Xm = repmat(xv,51,1);
yv = (0:0.2:10)'; Ym = repmat(yv,1,101);
R1 = Ym < Xm;    % Fall a)
contourf(xv,yv,R1,4); axis equal; pause
R2 = Ym < Xm.^2;  % Fall b)
contourf(xv,yv,R2,4); axis equal; pause
R3 = abs(Ym - Xm) < 1;  % Fall c)
contourf(xv,yv,R3,4); axis equal; pause
% Fall d)
xq = -10:0.2:10;   Xs = repmat(xq,101,1);
yq = (-10:0.2:10)'; Ys = repmat(yq,1,101);
R4 = abs(abs(Xs) + abs(Ys)) < 5;
contourf(xq,yq,R4,4);  axis equal;
```

Übung 21–2 Im I. Quadranten: a) $y > x$: Gebiet oberhalb der Winkelhalbierenden $y = x$
b) $|y - 5| < 1$ 1.Fall $y - 5 > 0$ d.h. $y > 5$, || weglassen, $y - 5 < 1$, $y < 6$
zusammen: $y > 5$ und $y < 6$, horizontaler Streifen $5 < y < 6$
2.Fall $y - 5 < 0$, d.h. $y < 5$: || durch $-$ ersetzen und Vergleichssymbol kehren
$5 - y > 1$, $y > 4$: zusammen mit $y < 5$, horizontaler Streifen $4 < y < 5$
Gemeinsam: horizontaler Streifen $4 < y < 6$

```
xv = 0:0.1:10; Xm = repmat(xv,101,1);
yv = (0:0.1:10)'; Ym = repmat(yv,1,101);
Rst   = abs(Ym-5) <1;
contourf(xv,yv,Rst,4)
```

Übung 21–3
Bereich $x = -20 : 1 : 20$, $y = 0 : 20$. Relation $y = x.^2$

```
xv = -20:20; Xm = repmat(xv,21,1);
yv = (0:20)'; Ym = repmat(yv,1,41);
Rf = Xm.^2 == Ym;
spy(flipud(Rf))
```

Übungen zu den komplexen Zahlen
Übung 24–1 $1 + j * 0$; $-1 + j * 0$; 0; $5 + 0i$; $-2 + 0i$;
$0 + 1i$; $0 + 3i$; $0 - j$; $0 - 2j$;
$1 + j$; $-1 - j$; $1 + j\sqrt{3}$; $1 - j\sqrt{3}$;

```
1*exp(j*0); 1*exp(j*pi); 5*exp(j*0); 2*exp(j*pi);
1*exp(j*pi/2); 3*exp(j*pi/2); 1*exp(j*3*pi/2);
2*exp(j*3*pi/2);                    sqrt(2)*exp(j*pi/4);
sqrt(2)*exp(-j*pi/4); 2*exp(j*pi/3); 2*exp(-j*pi/3);
```

Übung 24–2 Additionen $6 + 3j$; $6 + 3j$; $3 + 3j$;
$3 - 3j$; 6; $4j$;
0; $1 - j$

Übung 24–3 a)...$4+2j$, $0+2j$, $3+2j$; $2-(-1)^k * (6 - int(k/2) + 2j)$, $2+2j$
b)$(6 + 4j) * j^{(k-1)}$
c))`sqrt(2)*exp(j*(k-1)*pi/4)`
d)`sqrt(2)^(k-1) *exp(j*(k-1)*3*pi/4)`

Übung 24–4 $4 * j$; $6 * exp(j * pi/4)$; $j * j - 1 = -2$;
$4 * exp(j * pi/6 * (1 + 7))$; $2 - 2j$; $-8 + 8j$;
$-2 + 2j$; $4/25 - j * 3/25$; sqrt(2)/sqrt(2)*exp(j*(pi/4-(-pi/4)))=j

Übung 24–5 imag(z) = $(z - \overline{z})/2$; $|z| = \sqrt{z * \overline{z}}$; $\arg(z) = \log(z/\sqrt{z * \overline{z}})/j$
Drehen: z*±i ; Punktspiegelung: -z

Übung 24–6
$z = \overline{z}$: reelle Achse;
imag(z)>0: obere Halbebene;
$|z| < 2$: Inneres des Kreises mit r=2;
arg(z) = 135° : Winkelhalbierende II. Quadrant;
$0 < \arg(z^2) < 180^o$: I. Quadrant und III. Quadrant

Übung 24–7 Zum Beispiel
(1*exp(j*pi/4))^4 = sqrt(2)^4/2^4*(1+j)^4=4/16*(1+4j+6j^2+4j^3+j^4) = -1

12.2
Lösungen der allgemeinen Übungen zum Kapitel 2

Funktionsplots

20–1 Grafische Demonstration von geraden und ungeraden Funktionen

Für weiter gehende Kommentare zu diesen beiden M-Files siehe im Teil III: Beispiele von M-Files zum Kapitel 2.

```
%EVENODD Skript m-File zur Demonstration
%  gerader und ungerader Funktionen
% die Vektoren x,y  muessen vorher definiert werden
v = input('Bitte Vergleichs-x-Wert eingeben: ');
[mr,vr] = min(abs(x-v)); [ml,vl] = min(abs(x+v));
mg = min(abs(y(vl)-y(vr))); mu = min(abs(y(vl)+y(vr)));
plot(x,y,'k'); hold on;  axis equal
if mu < mg
  plot([x(vl) 0  x(vr)],  [-y(vr) 0 y(vr)], '-or')
else
  plot([x(vl) 0  x(vr)],  [y(vr)  y(vr) y(vr)], '-og')
end
hold off
```

20–2 Darstellung der Periodizitätseigenschaft einer Funktion

```
%PERIODIC Skript m-File zur einfachen
%  Demonstration der Periodizitaet
% Vektoren x,y, sowie Periodendauer T
%  muessen vorher definiert werden
v = input('Bitte Vergleichs-x-Wert eingeben: ');
[ml,vl] = min(abs(x-v));
[mr,vr] = min(abs(x-v-T));
plot(x,y,'k');
hold on
plot([x(vl) x(vr)], [y(vl) y(vl)], '-or')
[mx,vx] = min(abs(x-v-2*T));
if mx < 0.1*T
plot([x(vr) x(vx)], [y(vr) y(vx)], ':or')
end
hold off
```

20–3 Gemeinsamer Plot von Sinus und Kosinus

```
t=(0:0.01:1)*2*pi; plot(t,cos(t));
hold on; plot(t,sin(t),'r')
```

20–4 Darstellung der Kombination von Sinus und Kosinus
Sinus und Kosinus kombinieren:

```
a = input('cos-Faktor');
b = input('sin-Faktor');
t=(0:0.01:1)*2*pi;
plot(t,cos(t)); hold on; plot(t,sin(t)) ;
plot(t,a*cos(t)+b*sin(t),'r')
```

20–5 Allgemeiner Plot mit parametrisierter Funktionswahl
Versuchen Sie z. B.

`genfcplot('cos'), genfcplot('tan'), genfcplot('exp')`

20–6 Die klassische Funktionskurve „Versiera di Agnesi“
Zeichnen der Versiera di Agnesi

```
a=2; x=-5:0.05:5;
y = a^3./(a^2+x.^2) ; plot(x,y)
```

20–7 Verschiedene Glockenkurven
Glockenkurven

```
x=-1:0.01:1;
y1=(1-x.^2).^1 ;  y2=(1-x.^2).^2 ; y3=(1-x.^2).^3 ;
plot(x,y1); hold on;
plot(x,y2); plot(x,y3)
```

Kurven in Parameterdarstellung, Lissajous-Figuren

20–8 Elementare Lissajous-Figuren
Lissajous-Figuren 1:1 und 1:2

```
t=(0:0.002:1)*2*pi; x=cos(t) ;
d=0; y=sin(t+d) ; plot(x,y)
d=pi/2 ; % oder: d=0.12, d=3*pi/4, d=p
% 1:2 mit: x=cos(t) ;
% d=variabel; y=sin(2*t+d) ; plot(x,y)
% 2:1 mit: x=cos(2*t) ;
% d=variabel; y=sin(t+d) ; plot(x,y)
```

20–9 Kompliziertere Lissajous-Figuren

allgemeine Lissajous-Figuren (z. B. 2:3)

```
d = 0.2  %  mit verschiedensten Werten
         % von d, k, l experimentieren!
k = 2
l = 3
t=(0:0.002:1)*2*pi;
x=cos(k*t) ;
y=sin(l*t+d) ;
plot(x,y)
```

20–10 Parameter zu einer Lissajous-Figur bestimmen

„Gezieltes Erraten“: welche sin/cos-Funktion passt in die Punkteschar
für x bzw. für y, die man beim Durchlaufen (Abfahren) der Kurve ausliest?

```
x = 1, 0.5, -0.5, -1, -0.5, 0.5, 1, 0.5, -0.5, -1, -0.5, 0.5, 1
y = 1, 0,   -1  ,  0,  1,   0,  -1, 0,    1,    0, -1,   0,   1
```

x – 2 Schwingungen, beginnend mit 1: x=cos(2*t),
y – 3 Schwingungen, beginnend mit 1: y=cos(3*t)

Kurven in Parameterdarstellung, Spiralen

20–11 Links- und rechtsläufige Spiralen

Der Radius muss pro Umgang um 1 größer werden d. h. c=1/(2*pi).
Startwinkel w0=0, weil r(0) = 0, r(2*pi)=1; r(4*pi)=2, r(6*pi)=2, etc.
Bei Linksdrehung zunehmend
(positiver mathematischer Winkelvorschub): c=1/(2*pi).
Bei Rechtsdrehung zunehmend
(negativer mathematischer Winkelvorschub): c=-1/(2*pi).
t = (0:0.01:5)*2*pi; r=1/(2*pi)*t, bzw. r=-1/(2*pi)*t; polar(t,r);
oder x = r.*cos(t) y = r.*sin(t)

20–12 Archimedische Spirale

Ansatz: r(w) = c*(w-w0),
angewendet auf (3/0): r(0)= c*(0-w0) = 3,
angewendet auf (0/4): r(pi/2) = c*(pi/2-w0) = 4.
w0 aus 1. Gleichung: x0 = -3/c,
eingesetzt in 2. Gleichung c*(pi/2 + 3/c) = 4 = c*pi/2 + 3 ergibt c*pi/2 = 1,
also c=2/pi und w0 = -3*pi/2
t = (0:0.01:5)*2*pi; r=(2/pi)*t, polar(t,r); oder x = r.*cos(t) y = r.*sin(t)

20–13 Logarithmische Spirale

Ansatz: r(w) = a*exp(k*w),

angewendet auf (1/0): r(0)= a*exp(k*0) = 1, also a=1

angewendet auf (3/0): r(2*pi) = 1*exp(k*2*pi) = 3, also k = log(3)/(2*pi)

t = (0:0.01:2)*2*pi; r=exp(k*t), polar(t,r); oder x = r.*cos(t) y = r.*sin(t)

Kurven in Parameterdarstellung – mathematische Klassiker

20–14 Verschiedene Zykloiden

rP = 1; t = (0:0.01:1)*2*pi; xOM = t; yOM = 1;

xMP= -rP*sin(t); yMP = -rP*cos(t);

x = xOM+xMP; y = yOM+yMP; plot(x,y);

andere Fälle mit rP = 0.6 (gestreckte Z.) , rP = 1.4 (verschlungene Z.)

20–15 Epizykloide

R= 1; r=0.5; t = (0:0.01:1)*2*pi; xOM = (R+r)*sin(t); yOM = (R+r)*cos(t);

xMP= -r*sin(t+R/r*t); yMP = -r*cos(t+R/r*t);

x = xOM+xMP; y = yOM+yMP; plot(x,y);

auch mit anderen r-Werten. Siehe epizyklodemo

bei den Beispiel-M-Files im Anhang.

20–16 Hypozykloiden

R= 1; r=0.5 oder r=0.25;

t = (0:0.01:1)*2*pi; xOM = (R-r)*sin(t); yOM = (R-r)*cos(t);

xMP= -r*sin(t-R/r*t); yMP = -r*cos(t-R/r*t);

x = xOM+xMP; y = yOM+yMP; plot(x,y);

Siehe epizyklodemo bei den Beispiel-M-Files im Anhang mit negativen Werten

für die Vorausdefinition von ZYKRADFAC.

Der Wert r=0.5 ergibt ein gerades Streckenstück,

eine allerdings etwas komplizierte Art, eine Gerade zu zeichnen.

20–17 Evolvente

R= 1; t = (0:0.01:1)*2*pi; xOM = R*cos(t); yOM = R*sin(t);

xMP= R*t*sin(t); yMP = -R*t*cos(t);

x = xOM+xMP; y = yOM+yMP; plot(x,y);

auch mit anderen r-Werten. Siehe evoldemo bei den Beispiel-M-Files im Anhang.

20–18 Kepler-Ellipse

p = 10; eps = 0.4; w0=0.8; w=(0:0.01:1)*2*pi; r = p./(1-eps*cos(w-w0)

ergibt bereits ein (w,r)-Paar von Vektoren zum Plotten mit polar(w,r)

x = r .* cos(w) ; y = .*sin(w) ; plot(x,y) plottet in Kartesischen Koordinaten.

Beachten Sie die ('./' und '.*') - Operationen!

20–19 Spline-Interpolation

```
% Skript cossplin
%  Interpolation von diskreten Punkten
%    der cos-Funktion
xp = -1:0.5:1;
yp = cos(xp); ypp = -sin(xp);
xfc = -1:0.1:1;  yfc = cos(xfc);
plot(xfc,yfc,'r--'); hold on
plot(xp,yp,'o')
% Einzelne Interpolationen in jedem der 4 Intervalle
%   der Breite 0.5
for pt = 1:4
  xl = xp(pt) ; yl = yp(pt);      % linker Punkt
  xr = xp(pt+1) ; yr = yp(pt+1); % rechter Punkt
  ylp = ypp(pt)*0.5; % Ableitung bezueglich t,
%            statt x
  yrp = ypp(pt+1)*0.5;
  % Interpolationsprinzip besser sichtbar durch
  %    feinere Einteilung
  xi = 0:0.01:0.5;  t = xi/(xr-xl);
  %  t-Funktionen zu yl=1, yr = 1,
  %     ylp = 1, yrp = 1
  %    (und jeweils alle anderen =0) einsetzen
  yi = yl*(2*t.^3-3*t.^2+1) + yr*(-2*t.^3+3*t.^2) +...
         ylp*(t.^3 -2*t.^2 +t)  + yrp*(t.^3 -t.^2)
  plot(xi+xl, yi,'k')
end
hold off
axis equal
```

20–20 MATLAB spline

Mit der angegebenen Funktion $1/(1-x^2)$ müsste durch Null dividiert werden, gemeint war aber $1/(1+x^2)$.

```
%  Skript mlspline.m
%  Interpolation von diskreten Punkten der Funktion
%  1/(1+x^2)
xp = 0:0.5:1.5
yp = 1./(1+xp.^2)
xf = 0:0.01:1.5;
yfo =  1./(1+xf.^2)
yf = spline(xp,yp,xf)
plot(xp,yp,'o'); hold on
plot(xf,yf); plot(xf,yfo,'r--')
```

```
hold off
```

20–21 Bezier-Viertelkreis

```
% Skript beziervkreis.m
%  Annaeherung eines Viertelkreises
%  im 1. Quadranten durch
%  eine Bezier-Interpolation
U1 = [1;0];  U2 = [0; 1];   % Start- und Endpunkt
% Die beiden Hilfspunkte liegen auf den Tangenten
p = 0.555
H1 = [1; p] ; H2 = [p; 1];
t = 0:0.01:1;
Bpt = U2*t.^3 - U1*(t - 1).^3 + ...
       3*H1*(t.*(t - 1).^2) - 3*H2*((t.^2).*(t - 1))
plot(Bpt(1,:),Bpt(2,:))
hold on
plot(cos(t*pi/2),sin(t*pi/2),'r--') % Vergleich Kreis
axis equal; hold off
```

Dreidimensionale Kurven in Parameterdarstellung

20–22 Einfache Schraubenlinien

t=(0:0.01:3)*2*pi; x=cos(t) ; y = sin(t) ; z = 0.1*t; plot3(x,y,z)
ylinks = -sin(t); plot3(x,ylinks,z)

20–23 Doppelhelix der Erbinformation

```
% Skript dnamodel.m
% einfaches Geometrie-Modell der B-DNA Doppelhelix
%  alle Dimensionen in Nanometern, Winkel in Grad
r = 1 ; gh =3.4; dw = 36
%  1. Helix, 15 Stellen
w1 = (0:15)*dw;
x1 = r*cosd(w1) ; y1 = r*sind(w1); z1 = gh/360*w1;
%  2. Helix nur 150 Grad vis a vis
%   (grosse und kleine Furche)
w2 = w1 + 150; x2 = r*cosd(w2) ; y2 = r*sind(w2);
z2 = z1 + 2*r*tand(6); % Winkel zwischen
%    Basenpaaren und Achse
mvx = [x1;x2] ; mvy = [y1 ; y2]; mvz = [z1 ; z2];
figure(1); clf
plot3(x1,y1,z1,'r*-') ; hold on
plot3(x2,y2,z2,'mo-')
plot3(mvx, mvy, mvz,'k')
```

```
axis equal;  view(10,14) ; hold off
```

20–24 Schraubenlinienpaar bei Parkhausauffahrt

```
t=(0:0.01:3)*2*pi;
xi = 6*cos(t);
yi = 6*sin(t);
xa = 9*cos(t);
ya = 9*sin(t);
z = 4*t/2/pi;
plot3(xi,yi,z);hold on; plot3(xa,ya,z);
```

20–25 Spiralbohrer

```
% Skript bohrerkante.m
% Durchmesser 12 mm, Radius 6mm
% Ganghoehe 18 mm, Länge ca 90 mm, 5 Umgaenge
t = (0:0.01:5)*2*pi; r = 6; gh = 18;
x1 = r * cos(t); x2 = r*cos(t+pi);
y1 = r * sin(t); y2 = r*sin(t+pi);
z = t*gh/(2*pi);
plot3(x1,y1,z,x2,y2,z)
axis equal
```

Folgen und Reihen

20–26 Fundamentalaufgaben zu arithmetischen Folgen

$d = (a_n - a_1)/(n-1)$ $a_n = a_1 + (n-1) \cdot d$ $n = (a_n - a_1)/d + 1$ $a_1 = a_n - (n-1) \cdot d$

20–27 Belohnung für die Erfindung des Schachspiels

$q = 2$, $n = 64$ $S = (q^n - 1)/(q-1) = 1.8447E19$ Körner also 9.2E11 Tonnen!

20–28 Abzahlungsvertrag

auf 1297.50 Euro

20–29 Rentensparen

Einzahlung pro Jahr 12*R, Einfacher Zins pro Jahr 66/12*3/100*R
Einlage/Jahr E = 12.165*R; Zinseszins-Endkapital = E*47.575=R*587.75; R=863.92 Euro/Monat

20–30 Abgebrochener Abzahlungsvertrag

Rückzahlung A=10000/24; Zinstotal Z= 10000/24*25*24/2*8/100/12=833.33; Rate R= (A+Z)/24 = 451.40 . Nach einem Jahr noch ausstehende Ratensumme, wenn 12. Rate schon bezahlt, ergibt 5416.54. Bei Berücksichtigung der für 5000 Euro für die wegfallende Abzahlung über 1 Jahr nicht mehr geschuldeten Zinsen von

5000/12*13*12/2*8/100/12 = 216.66 würde sich diese Restschuld auf 5199.90 reduzieren. Zusammen mit der 12. Rate 5651.30 Euro. Der nicht mehr geschuldete Zins bei vorzeitiger Rückzahlung wird aber in der Praxis meist nicht vergütet (entgangenes Kreditgeschäft).

Komplexe Zahlen

20–31 Addition von komplexen Zahlen – grafisch

```
      % sum = cplxadd(z1,z2) demo Funktion zur Addition
von komplexen Zahlen
      function sumbk = cplxadd(z,u)
      r = max(abs(z), abs(u)) +2; sumbk = z+u;
      plot([0 real(z)], [0 imag(z)],'b'); hold on;
axis([-r r -r r]);
      plot([0 real(u)], [0 imag(u)],'g')
      plot([real(z) real(z)+real(u)], [imag(z)
imag(z)+imag(u)],'g')
      plot([0 real(z)+real(u)], [0 imag(z)+imag(u)],'r')
```

20–32 Hilfsfunktion zum Ausdrucken von komplexen Zahlen

```
function cplxdisp(z)
fprintf('a+ib = %8.4f +i* %8.4f |z| = %8.4f arg(z) = %8.4f \n',...
          real(z), imag(z), abs(z), angle(z))
end % function
```

20–33 Sinus und Kosinus durch $\exp(i \cdot \phi)$ ausdrücken

$\mathrm{e}^{i*w} + \mathrm{e}^{-i*w} = \cos(w) + i * \sin(w) + \cos(w) - i * \sin(w)$
also gilt:$\mathrm{e}^{i*w} + \mathrm{e}^{-i*w} = 2 \cdot cos(w)$
$\mathrm{e}^{i*w} - \mathrm{e}^{-i*w} = \cos(w) + i * \sin(w) - \cos(w) + i * \sin(w)$
somit gilt auch:$\mathrm{e}^{i*w} - \mathrm{e}^{-i*w} = 2i \cdot sin(w)$
und damit $sin(w) = (\mathrm{e}^{i*w} - \mathrm{e}^{-i*w})/2i$.

20–34 Mehrwinkelformeln aus den Potenzen der Euler'schen Identität

Für dreifache Winkel:

$$\cos^3(w) + 3 * j * \cos^2(w) * \sin(w) - 3 * \cos(w) * \sin^2(w) - j * \sin^3(w)$$
$$= \cos(3 * w) + j * \sin(3 * w)$$

Realteil: $\cos(3 * w) = \cos^3(w) - 3 * \cos(w) * \sin^2(w)$
Imaginärteil: $\sin(3 * w) = 3 * \cos^2(w) * \sin(w) - \sin^3(w)$

20–35 Komplexe Gleichung 4. Grades

Z1 .. Z4 = exp(j*(pi/8 + k*2*pi)) ;

Das Quadrat mit den 4 Lösungen ist verdreht, keine reelle Lösung.

Grafiken mit komplexen Zahlen

20–36 Polygon im Einheitskreis

```
val = input('Eingabe der Anzahl Ecken des Polygons')
Z = exp(j*(0:val)/val*2*pi) ; plot( Z); axis equal
```

20–37 exp(j*w) ist der Einheitskreis

```
 w = 2*pi*(0:0.01:1); c = exp(j*w); plot(c); axis equal
```

20–38 Pseudozykloiden („Kugelschreiber-Einfahrkurven“)

```
 w = 2*pi*(0:0.01:6); c = exp(j*w);
 t = w/(2*pi); p = f*t + c; plot(p)
```

mit den Parameter-Werten für f ergibt sich
f=0: Kreis; f=0.1 bis f=0.6: „Kugelschreiber-Einfahrkurve“
f=1.333: „Gartenzaun“; f=2*pi: Zykloide.

20–39 Mit komplexen Zahlen eine Brezel zeichnen

```
w = 0.67*pi:0.02*pi:4.33*pi;
b= exp(j*w) - w/1.5/pi;
plot(b,'r*')
axis equal
```

20–40 Komplexe Spirale

Die Punkte (a+j*b)^k liegen auf einer logarithmischen Spirale
mit der Gleichung r(t) = 1*exp(log(R)/w*t)
mit R=sqrt(a^2 + b^2) und w = atan(b/a).

12.3
Miniprojekte zur Elementarmathematik

201 Pythagoreische Schnecke, Theodora-Spirale

```
% Skript theodoradisp.m
thlin = 1 ; thac = 1;
nmax = 18;
for k = 1:nmax
    % der neue Punkt ist thac + i*angle(thac)
    %  Betrag = 1 und sekrecht zur Richtung von thac
    thnew = thac+i*exp(i*angle(thac));
    thlin = [thlin thnew];
    thac = thnew
end
plot(thlin)
hold on;  axis equal
```

```
zstr = [thlin ; zeros(1,length(thlin))];
plot(zstr,'r') ;
plot([0 0],[-5 5],'k'); plot([-5 5],[0 0],'k');
hold off
```

202 3D Darstellung eines Moebius-Bandes

Die zwei Parameter auf denen die 3D Darstellung aufbaut sind der Winkel `w` entlang dem Mittelkreis und der Verwindungs-Winkel `t`, welchen die Fläche gegenüber der Kreis-Ebene aufweist. Für `w = 1:360` ergibt sich für `t = 1:180`
Für die Darstellung wurde eine Hilfsfunktion konstruiert, dvmoeb:

```
% dvmoeb.m Hilfsfunktion zum Moebius-Band moebidef.m
function dv = dvmoeb(alf,tet,fac)
d0 = [fac 0 0]';
ca = cos(alf);
sa = sin(alf);
ct = cos(tet);
st = sin(tet);

dv = [ca -sa 0; sa ca 0 ; 0 0 1]* ...
        [ct  0 -st ; 0 1 0; st 0 ct] * d0;
end % function
```

Dadurch wird das Zeichnen der 3D-Figur wesentlich vereinfacht:

```
% Skript moebidef.m
% Zeichnen eines Moebius-Bandes mit plot3()
t = (0:3:720)*pi/180;
dr = t/2;
xm = 10*cos(t);
ym = 10*sin(t);
zm = zeros(1,length(t));
ML  = [xm ; ym ; zm];
plot3(xm,ym,zm,'r')
     axis([-12.5 12.5 -12.5 12.5 -12.5 12.5])
axis square
hold on
DL = [];
%
for j=1:6
wamp = j*0.4;
DL = [];
for k=0:240
DL = [DL dvmoeb(t(k+1),t(k+1)/2,wamp)];
end
LL = ML+DL;
```

```
if j==6
for k=1:10:120
plot3([LL(1,k) LL(1,k+120) ], [LL(2,k) LL(2,k+120) ], ...
      [ LL(3,k) LL(3,k+120)]  ,'r')
end
plot3(LL(1,:),LL(2,:), LL(3,:),'r')
     else
plot3(LL(1,:),LL(2,:), LL(3,:),'k')
end
end
hold off
```

203 Von Geisterhand aufgebaute Vielecke

```
% Skript  polygonfilm.m -
%   Aufbau und Abbau von regulaeren
%  Dreiecken bis zu Achtecken
ntot=9;
w = zeros(1,ntot);
%
t = (0:0.005:1)*2*pi;
xk = cos(t); yk = sin(t);
plot(xk,yk,'k','LineWidth',2.5); hold on;
axis equal; axis off
k=0 ;
wn = 0;
 w = [zeros(1,ntot-1) wn];
for wn = 0:0.005:1
  w(ntot) = wn ;
  if wn > 1/3
    w(ntot-1) = 1/3;
  end
  if wn > 2/3
    w(ntot-1) = 2/3;
    w(ntot-2) = 1/3;
  end
  if k>0
    delete(plhd)
  end
  k=1;
  x=sin(2*pi*w);
  y=cos(2*pi*w);
  plhd = plot(x,y,'r','LineWidth',4);
  pause(0.02)
```

```
  axis equal
end
%
pause(0.2)
for div =4:ntot-1
  ist = ntot-div+1;
  for wn=0:0.005:1/div
    w  = [zeros(1,ist-1) linspace(wn,1,div)];
      delete(plhd)
    x=sin(2*pi*w);
    y=cos(2*pi*w);
    plhd = plot(x,y,'r','LineWidth',4);
    pause(0.02)
  end
  pause(0.2)
end
%
pause(0.5)
for div =ntot-1:-1:4
  ist = ntot-div+1;
  for wn=1/div:-0.005:0
    w  = [zeros(1,ist-1) linspace(wn,1,div)];
      delete(plhd)
    x=sin(2*pi*w);
    y=cos(2*pi*w);
    plhd = plot(x,y,'r','LineWidth',4);
    pause(0.02)
  end
  pause(0.2)
end
%
w = [zeros(1,ntot-3) 1/3 2/3 1];
for wn = 1:-0.005:0
  w(ntot) = wn  ;
  if wn < 2/3
    w(ntot-2) = 0;
    w(ntot-1) = 1/3;
  end
  if wn < 1/3
    w(ntot-1) = 0;
  end
    delete(plhd)
  k=1;
  x=sin(2*pi*w);
  y=cos(2*pi*w);
```

```
  plhd = plot(x,y,'r','LineWidth',4);
  pause(0.02)
  axis equal
end
```

204 Film einer Welle in Bewegung

```
% Skript wellenfilm.m
%  Beispiel langsame Duenung
%  am Meer lam = 8 m, w = 0.25 /sec
%  Wellenhoehe 1 m, damit A= 0.5 m
lam =  8;
k = 2* pi/lam ;  % Wellenzahl, inverse Laenge
w = 0.25 ;
figure(1); clf;
x = 0:0.2:20;
for t = 0:0.2:200
  wel = real(exp(i*(k*x - w*t) )) ;
  hdnew = plot(x,wel,'b','linewidth',2);
  if t == 0
    axis equal ; axis([0 20 -2 6]);
    hold on
    pause(1.8);  % Pause bei Start
  end % if hdol
  pause(0.02)
  if t ~= 200  % letzte Welle bleibt
    delete(hdnew)
  end
end
hold off
```

12.4 Lösungen zu den Selbsttests Kapitel 2

Lösungen der Testserie 2.1

T211 – Mehrere Linien beim gleichen x-Wert, sowie Kreuzungspunkte.
$-\overline{z} = \mathrm{conj}(z)$; $--2 + 2 * j$ $-\cos(x)$, x^2, x^{16}, $\cosh(x)$
T212 `x =[1 0 -1 0 1]; y = [1 -1 1 -1 1]`
`t=(0:0.02:1)*2*pi; x = cos(t); y=cos(2*t); plot(x,y)`
T213 `t=(0:0.01:1)*2*pi; plot(5*(t-j*exp(-j*t)))`

T214 $\exp(j * (3 * pi/8 + k * pi/2))$
T215 $r(w) = a * (w - w_0)$ $r(0) = 1, r(2\pi) = 3$ $a = -2/2\pi$ $w_0 = \pi$

```
t=-(0:0.02:3)*2*pi; w=t-pi; r =-w/pi;
x = r.*cos(t); y=r.*sin(t); plot(x,y)
```

Lösungen der Testserie 2.2

T221 – $x(t*) = y(t*)$
– `zr = (z+conj(z))/2 ; zi=-j* (z-conj(z))/2`
– Bei einer Funktion darf einem Wert der unabhängigen Variablen nur ein einziger Wert der abhängigen Variablen zugeordnet sein.
– $\sin(x)$, x^3, $\sinh(x)$, $\cos(x + \pi/2)$
T222

```
x = [ 0 -1 0 1 0 -1 0 1 0];
 y = [  0 -0.87 -1 -0.87 0 0.87 1 0.87 0]
t=(0:0.02:1)*2*pi; x = -sin(2*t); y=-sin(t); plot(x,y)
```

T223

```
r=5.1; t=(0:0.02:5)*2*pi; p=1.6/2/pi
x=r*cos(t); yl=r*sin(t); yr=-r*sin(t); z=p*t;
plot3(x,yl,z,x,yr,z)
```

T224 $\exp(j * (pi/24 + k * pi/6))$
T225 $r(w) = a * \exp(k * w)$ $r(0) = 1, r(2\pi) = 2$ $a = 1$ $k = \log(2)/2\pi$

```
t=(0:0.02:3)*2*pi;  r =exp(t*log(2)/2/pi);
polar(t,r); x=r.*cos(t); y=r.*sin(t); plot(x,y)
```

13
Lösungshinweise zum Kapitel 3

13.1
Im Text eingefügte Übungen

31–1 Beispiel zum Berechnen einer Determinanten
M= 5x5 tridiag(-1 2 -1), Det(M) = 6

31–2 Aufwandsvergleich zwischen Rechenverfahren
```
n= 4,6,8,10;   n! = 24, 720, 40320, 3628800
n^3= 64, 216, 512, 1000
```

31–3 Zwei Arten der Determinantenberechnung mit MATLAB
Ein Beispiel det(A)= -0.004091 [L,R,P]= lu(A), prod(diag(R)) = -0.004091

32–1 Phasenschieberschaltung für stationäre Schwingungen
R= 10 Ohm, Rvar= 1..100 Ohm, C=50 pF, f = 200 MHz.

```
C=-j/(200e6*2*pi*0.05e-9);
R = 10
Rout = 1E6
for Rvar = 1:100
M = [1 -1 0 -1 0 0;0 1 -1 0 0 -1; 0 0 0 1 -1 1; ...
     0 C 0 -R  0 Rout; 0 0 Rvar 0 -R -Rout; ...
     0 C Rvar 0 0 0];
b=[0 0 0  0 0 1]';
cur= M\b;
cph(Rvar) = cur(6);
end
plot(angle(cph)*180/pi);
```

13.2
Lösungen der allgemeinen Übungen zum Kapitel 3

Lineare Abhängigkeit

30–1 Paarweise lineare Unabhängigkeit

v - w = [0 1 0]'; mit u = [0 1 0]' wird also v-w-u=0.
die 2er Gruppen v,w ; w,u und v,u
sind aber linear unabhängig.

30–2 Linear unabhängige Dreiergruppen

Die Differenz zwischen a und b ist [1 1 1 1]' mit x = [0 0 1 1]' werden a,b,c,x linear abhängig. Ob alle 3-er Gruppen dann linear unabhängig sind, muss noch geprüft werden, ist aber aus der Analogie zwischen c und x zu erwarten.

30–3 Lineare Abhängigkeiten bei einer Gruppe von Spaltenvektoren

Es gilt b-c = d, also muss einer dieser drei Vektoren wegfallen. Alle Kombinationen, bei denen einer dieser Vektoren fehlt, haben Rang 4.

Der Rang einer Matrix

30–4 Spaltenrang bei verschiedener Gruppierung

es gilt a–5*b = –c und d+e=a. Damit haben [a b c d], [a b c e], [a b d e], etc. alle Rang 3.

30–5 Maximaler und minimaler Rang

Mit a=0 werden die 1. und 2. Spalte identisch. Mit b=0.5 werden die 3. und 4. Spalte Vielfache voneinander. Somit ergibt a=0, b=0.5 Minimalrang 2. Mit a=1 und b=0 wird Rang=4 erreicht.

30–6 Linear abhängige/unabhängige Vektoren

a) q=2, p=beliebig oder q≅ 2 und p=0 liefern Rang 2.
b) mit q=3 gilt unter den Spalten b+d=c. Mit p=-1 gilt zusätzlich b-d=a. Beide Bedingungen gleichzeitig erfüllt, ergeben Minimalrang 2, beide gleichzeitig verletzt ergeben Maximalrang 4.

30–7 Verschiedene Zahlen durch Vertauschen von Ziffern

a+b+c=11; 100*a+10*b+c - (100*c+10*b+a) = 594;
100*a+10*b+c - (100*a+10*c+b) = 18;
M= [1 1 1; 99 0 -99; 0 9 -9] ; b=[11 594 18]';
z = [7 3 1]

30–8 Gleichungssystem aus Ziffernvertauschungen

a+b+c+d = 13; 1000*a+100*b+10*c+d - (1000*d+100*c+10*b+a) = 6264; c=d; a-d=6;
M=[1 1 1 1; 999 90 -90 -999; 0 0 1 -1; 1 0 0 -1]; b=[13 6264 0 6]'; z=[7 4 1 1];

30–9 Jahresumsätze

u2=1.5*u1+100; u3=1.5*u2+100; u4=1.5*u3+100; u5=1.5*u4+100;
u1+u2+u3+u4+u5=4000;
M=[-1.5 1 0 0 0; 0 -1.5 1 0 0; 0 0 -1.5 1 0; 0 0 0 -1.5 1; 1 1 1 1 1] b = [100 100 100 100 4000] u=[179.15 368.72 653.08 1079.62 1719.43]' Euro.

30–10 Führungsgehälter

s2=s1*1.2+20; s3=s2*1.2+20; s4=s3*1.2+20; s5=s4*1.2+20; s1+s2+S3+S4+S5=1000; M=[-1.2 1 0 0 0; 0 -1.2 1 0 0; 0 0 -1.2 1 0; 0 0 0 -1.2 1; 1 1 1 1 1] b=[20 20 20 20 1000]' ; s= [101.57 141.88 190.26 248.31 317.97]' kFr.

30–11 Antiquitätenhändler

A+B+C+D+E = 20; D= 10*(B-0.5); D = A+B+E; C-1= A+D+E; D-B = 2*A; M= [1 1 1 1 1; 0 -10 0 1 0; -1 -1 0 1 -1; -1 0 1 -1 -1; 2 1 0 -1 0] b = [20 -5 0 1 0]'; Br=[2 1 10 5 2];

30–12 Kiestransport

Stand am Anfang des 4. Tages = 120–48–16=56 GLh. Zeitbedarf bei Leistung (2+2*0.8) = 1.6 : 15.56h Die Arbeit wird theoretisch (0.44h) 26 Minuten früher fertig.

30–13 Pumpleistung

Nacht: 12 h Zufluss mit G/12 pro Stunde.
7-8h (Pumpleistung x/h + Zufluss G/12/h)*1h = -G/6, also x=G*3/12/h
8-9h weiter wie 7-8 ergibt Füllstand G*2/3
9-t h Ab Füllstand 2/3 + (Zufluss G/12 - 2*Pumpleistung G*3/12)*t = 0
t = 1.6 h, die Grube ist um 10h 36 leer (statt um 13 Uhr ohne 2. Pumpe)

30–14 Uhrzeiger

G läuft mit der Geschwindigkeit 1U/h, k mit U/12/h.
Es gilt mit Start ab 12h (bzw. 0h) G*t = k*t + n, n= 1,2 ...11
(1-1/12)*t=n hat 11 Lösungen für n=1:11 nämlich
1.0909= 12/11, 2.1818=2*12/11 etc bis 11*12/11 =12h

Kirchhoff'sche Netze

30–15 Netzwerk mit 3 quadratischen Maschen

```
*---------------*----100----*----250----*
|               |           |           |
|              200         100         100
|               |           |           |
= 10 V           *----200----*---100-----*
|               |           |
|              100         50
|               |           |
*---------------*----250----*
```

```
 M= [ 1 -1 0 -1  0   0 0  0 0  0 0; 0 1 -1 0 -1  0  0 0  0 0 0;
      0  0 1  0  0  -1 0  0 0  0 0; 0 0  0 1  0  0 -1 0 -1 0 0;
      0  0 0  0  1   0 1 -1 0 -1 0; 0 0  0 0  0  1  0 1  0 0 0;
      0  0 0  0  0   0 0  0 0  1 1; 0 100 0 -200 100  0 -200 0 0 0 0;
    0 0 250 0 -100  100 0 -100 0 0 0; 0 0 0 0 0  0 200 0 -100 50 -250;
      0  0 0 200 0   0 0 0 100 0 0]
b=[ 0 0 0 0 0   0 0 0   0 0 10 ]'
cur = M\b*1000
%
%~[56.3 25.5 4.6 30.8 20.8 4.6  -7.6 -4.6 38.4 17.9 -17.9]'*0.001
```

30–16 Einfaches Netzwerk

Doppeltes Quadrat: Knoten 1,2,3 oben, 4 5 6 unten, i0 = Batterie, i1 = 1-2, i2= 2-3, nach rechts;
i3=1-4, i4=2-5, i5=3-6, nach unten; i6 = 4-5, i7 = 5-6 nach rechts

```
   M= [ 1 -1 0  -1  0  0  0  0; 0 1 -1 0  -1  0  0  0 ; ...
       0 0  1 0 0 -1 0 0; ...
 0 0 0 1 0 0 -1 0; 0 0 0 0 1 0 1 -1; ...
  0 100 0 -200  400 0 -100 0; ...
  0  0  100 0 -400 1000 0 -500 ;...
 0 100 100 0 0 1000 0 0   ]
b=[ 0 0 0 0 0 0 0 -10 ]'
c= M\b
%   c = [ 22.09 12.27 7.98 9.82 4.29 ...
    7.98  9.82  14.11 ]' *0.001
```

30–17 Kirchhoff-Lösung für Dreiecksnetze

Nur obere Hälfte, Kirchhoff-Matrix

```
R=100
M= [ 1 -1  0  0 -1    0  0  0  0  0     0  0  0  0 ; ...
     0  1 -1  0  0   -1  0  0  0  0    -1  0  0  0 ; ...
     0  0  1 -1  0    0 -1  0  0  0     0 -1  0  0 ; ...
     0  0  0  1  0    0  0  0  0  0     0  0 -1  0 ; ...
     0  0  0  0  1    0  0 -1  0  0     1  0  0  0 ; ...
     0  0  0  0  0    1  0  1 -1 -1     0  1  0  0 ; ...
     0  0  0  0  0    0  1  0  1  0     0  0  1 -1 ; ...
%
     0  R  0  0 -R    0  0  0  0  0     R  0  0  0 ; ...
     0  0  R  0  0   -R  0  0  0  0     0  R  0  0 ; ...
     0  0  0  R  0    0 -R  0  0  0     0  0  R  0 ; ...
     0  0  0  0  0    R  0 -R  0  0    -R  0  0  0 ; ...
     0  0  0  0  0    0  R  0 -R  0     0 -R  0  0 ; ...
     0  0  0  0  0    0  0  0  R -R     0  0  0  R ; ...
     R  R  R  R  0    0  0  0  0  0     0  0  R  R ]
b = [0  0  0  0  0    0  0  0  0  0     0  0  0 10]'
cur = M\b
% Loesung: Cur = [41.14 22.07 12.04  3.68 19.06 13.04 ...
%   7.36 16.05 6.35 23.75 -3.01 1.00 3.68 17.39 ]
% Ersatzwiderstand obere Haelfte 243.1 Ohm, symmetrich 121.5 Ohm
```

30–18 Skript-M-File für die Wheatstone'sche Messbrücke

```
% ---Matlab-File--- wheatstone.m --------
% die Widerstaende R1, R2,   R3, R4 und R5 muessen vor
%  dem Ausfuehren dieses m-Files definiert werden,
%  ein kleiner Wert von R5 erhoeht die Empfindlichkeit.
   M =[  1  -1   0  -1   0   0 ;
         0   1  -1   0   0  -1 ;
         0   0   0   1  -1   1 ;
         0  R1   0 -R3   0  R5 ;
         0   0  R2   0 -R4 -R5 ;
         0  R1  R2   0   0   0 ] ;
  b = [ 0  0  0  0  0  10 ]'  ;
  isol = M \ b;
% Das hauptsaechlich interessierende Resultat ist
%  der Strom in der Bruecken-Strecke R5
  tx = 'Strom in Brueckenzweig:' ;
  tx   i5 = isol(6) ;   i5
% -- end --- wheatstone.m ---------
```

30–19 Quadratische Kirchhoff-Netze

Quadrat 2x2 obere Hälfte 3kΩ, total 1.5kΩ
Quadrat 3x3 obere Hälfte 3.71kΩ, total 1.86kΩ
Quadrat 4x4 , total 2.14kΩ

Kirchhoff'sche Netze mit stationären Wechselströmen

30–20 Parallelresonanzkreis mit Vorwiderstand

```
*---------------*----R=10 Ohm-----*
|               |                 |
|          C=100E-9 Farad       L=1E-3 Henry
|               |                 |
~ 10 V          *-----------------*
|               |
|           R=30 Ohm
|               |
*---------------*
```

```
function icplx=seriereso(freq)
     w = 2*pi*freq;
     C = -j/0.3e-6/w;
L = j*w*2e-4;
R = 10;
ct = 1/(R+C+L);
icplx = R*ct;
```

30–21 Serienresonanzkreis

```
function ivec=resonet(freq)
     w = 2*pi*freq;
     C = -j/100.0e-9/w;
L = j*w*10e-4;
R = 10;
Rv = 30;
M = [ 1 -1 -1 0 0; 0 1 0 -1 0; 0 0 1 0 -1;...
      0 R -C L 0; 0 0 C 0 Rv];
b = [0 0 0 0 10]';
ivec = M\b;
```

30–22 Phasenschieberschaltung

```
Up = 10; Ua = Up/2; w=2*pi*10000;
for k=1:40
  cb = Up/(k*50-j/(w*50e-9));
  Ub = Up - cb*k*50;
  dU(k) = Ua-Ub;
end
```

```
plot(angle(dU)*180/pi)
```

Funktions-M-Files

30–23 Programmieren einer Funktion zum Testen der Orthogonalität

```
function iforth = orthotest(Q)
[n,m]=size(Q);
if n==m
  if max(max(abs(Q.'*Q -eye(n))) ) < 1e-14
    iforth = 1;
  else
    iforth = 0;
  end
else
iforth=0;
end
```

30–24 Selbst die Transpositions-Funktion programmieren

```
function Mtra = selftransp(Mori)
[n,m]=size(Mori);
if n == m
  for zei = 2:n
    for spa = 1:zei-1
      h = Mori(zei,spa);
    % Vertauschen braucht Hilfsplatz
      Mori(zei,spa) = Mori(spa,zei);
      Mori(spa,zei) = h;
    end
  end
  Mtra = Mori;
else
  Mtra = [];
end
```

Orthogonale Matrizen

30–25 Turmmatrizen sind orthogonal

Am besten kann man vielfältige Tests durchführen, indem man einen Permutationsvektor definiert und mit der unten nochmals gezeigten Lösung der Übung 10–37 daraus eine Turmmatrix erstellt.

```
function [Mp, Tm] = permvectomat(Mo,v)
% function Mp = permvectomat(Mo,v)
%  Mo, Mp Original, Zeilen-permutierte Matrix
```

```
%  (Fuer Spaltenpermutation  Sp = So*Tm')
%  v Vektor der Permutation
% (Dimensionen nicht kontrolliert)
  Tm = zeros(length(v));
  for k=1:length(v)
    Tm(k,v(k)) = 1;
  end
  Mp = Tm*Mo;
end %function
```

anschliessend lautet der Test`Tm*Tm'`

30–26 Potenzen von Turmmatrizen

Die obenstehende Funktion zum Erstellen einer Turmmatrix aus einem Permutationsvektor leistet auch bei diesen Versuchen gute Dienste mit jeder dieser Matrizen sind verschiedene Potenzen durchzuprobieren:

```
v = [1 3 2 4 5]
[Mp,T]= permvectomat(eye(5),v)
% Exponenten einzeln testen
format compact
for k= 1:5
  T^k
end
```

30–27 Orthogonalitätstest

Für R_x gilt $I = R^T \cdot R = R_x =$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(w)^2 + \sin(w)^2 & -\sin(w)\cdot\cos(w) + \sin(w)\cdot\cos(w) \\ 0 & -\sin(w)\cdot\cos(w) + \cos(w)\cdot\sin(w) & \sin(w)^2 + \cos(w)^2 \end{pmatrix}$$

Gauß-Algorithmus und L-R-Zerlegung

30–28 Showtime Gauß-Algorithmus und L-R-Zerlegung

Benutzen Sie die Möglichkeit zum Eingeben einer von Ihnen gewählten Matrix! Als Spezialbeispiel kann eine bereits in Dreiecksform vorliegende Matrix eingegeben werden.

30–29 Dirty Prototype

```
% code Fragment Verarbeitung von A
%   steht bei Spalte 'k'
xmx = A(k,k);
pzei = 0;
for ze = (k+1):ndim
 if A(ze,k) > xmx
```

```
   pzei = ze;
   xmx = A(ze,k);
 end
end
if pzei > 0
  H = A(k,k:ndim);
  A(k,k:ndim) = A(pzei,k:ndim)
  A(pzei,k:ndim) = H
end
```

30–30 Pivot-Strategie

```
function Mnew = mxpivunterhalb(Mold,spa)
Mnew = Mold;
  testvec = Mold(spa:end,spa);
  [mxv,mxp] = max(abs(testvec));
  if mxp > 1
    imx = mxp + spa-1;
    Mnew(imx,:) = Mold(spa,:);
    Mnew(spa,:) = Mold(imx,:);
  end
end % function
```

30–31 Spezielles Rückwärtseinsetzen

Funktioniert nur für eine sowohl tridiagonale als auch gleichzeitig Rechts-Dreiecksmatrix (also eigentlich eine rechts-bidiagonale Matrix).

```
function xsol = trirbksub(T,b)
[m,n]= size(T);
xsol(n) = b(n)/T(n,n);
for k = (n-1):(-1):1
   x(k) = (b(k) -x(k+1)*T(k,k+1);
end
end % function
```

30–32 L-Teilmatrizen in der L-R-Zerlegung

```
A=[4 4 8; ;2 -3 4 ; 1 2 1]
L1 = [1 0 0; -1/2 1 0; -1/4 0 1]
L1I = [1 0 0; 1/2 1 0; 1/4 0 1]
L1*A
L2 = [1 0 0; 0 1 0; 0 1/5 1]
L2I = [1 0 0; 0 1 0; 0 -1/5 1]
L2*L1*A
```

```
[LL,RR,P]= lu(A)
LI = L1I*L2I
```

30–33 Bestandteil einer L-R-Zerlegung

```
C = [ 1 0 0; -3/2 1 0; -5/2 0 1]
M = [ 2 3 4; 3 1 1 ; 5 0 1]
C*M
```

30–34 Rückschlüsse aus der L-R-Zerlegung

```
RR=[1 2 3 ; 0 4 5; 0 0 6]
L1 = [1 0 0; 0.5  1 0;  -1 0 1]
L1I = [1 0 0; -0.5  1 0;  1 0 1]
L2 = [1 0 0; 0  1 0;  0  -0.4 1]
L2I = [1 0 0; 0  1 0;  0  0.4 1]
L1I*L2I*RR
[L R P ] = lu(ans)
```

30–35 L-R-Zerlegung mit Bestimmung der Inversen

```
A = [4 8 4; 2 2 4; 1 3 -1 ]
[L R P] = lu(A);  L\eye(3)
x = R\y

y = [1 0 0; -1/2 1 0; -1/2  1/2 1]
x = [-7/4 5/2 3; 3/4 -1 -1; 1/2 -1/2 -1]
Ainv = [-1.75 2.5 3; 0.75 -1 -1; 0.5 -0.5 -1]
```

Singuläre Systeme

30–36 Partikuläre und homogene Lösung

Ob eine partikuläre Lösung existiert, kann man testen, indem man den Rang der Matrix vergleicht mit dem Rang der um die rechten Seiten erweiterten Matrix.

```
% Skript parthomog.m  partikuläre Loesung und Nullraum
A =  [2 1 2 4; 0 4 0 1; 0 0 0 0; 0 0 0 0]
b = [6 8 0 0]';
rank(A)
rank([A b])
% x3, x4 = 0,0, dann von Hand einfach geloest,
%   Anschliessend Test der partikulären Lösung:
pl = [2 2 0 0]'
A*pl
```

Falls eine partikuläre Lösung existiert, so findet man diese am einfachsten, indem man $x_3 = 0$ und $x_4 = 0$ setzt und für das Rest-System für die restlichen x_1 und x_2 löst, wie oben gezeigt.
In der Gleichung $A * x = b$ gehören die nach unten verschobenen Nullzeilen zu den Unbekannten x_3 und x_4. Weil deren Werte in jeder Spalte mit Null multipliziert werden, kann für jede dieser Nullzeilen-Unbekannten ein beliebiger Wert angenommen werden. Mit der Wahl der zwei Fälle $x_3, x_4 = 0, 1$ und $x_3, x_4 = 1, 0$ und Einsetzen dieser Wertepaare in die homogene Gleichung $A * x = 0$ findet man zwei Vektoren, die den Nullraum aufspannen. (es wären aber auch unendlich viele andere Vektorpaare möglich).
Mit diesem Vorgehen findet man die Vektoren

```
nl1 = [-1 0 1 0]'  % und
nl2 = [-15/8 -1/4  0  1]
%  Nullraum zu parthomog
%  Test A*nl1 = 0-vec  A*nl2 = 0-vec
A =  [2 1 2 4; 0 4 0 1; 0 0 0 0; 0 0 0 0]
A*nl1
A*nl2
```

Die allgemeine Lösung ist also
`x = pl + c1*nl1 + c2*nl2` mit beliebigen c1,c2.

30–37 Volle Pivotstrategie

Mit einer Diagnostik-Version des Gauß-Algorithmus mit voller Pivotstrategie lässt sich das Entstehen von Nullzeilen am unteren Ende gut verfolgen. Als Beispiele sind die Matrix `A = [3 2 1; 1 1 1; 2 1 0]` , sowie Matrizen, die aus zwei identischen Rechteck-Teilen übereinander oder nebeneinander zusammengesetzt wurden. (für diese Tests empfiehlt sich die Option `format compact`.

```
function [R,perm,bt,rgdef]=gauss2fpivd(A,b)
  % A -> R full Pivoting
  % function [R,perm,bt,rgdef]=gauss2fpiv(A,b)
  % Diagnostik-Version
    [zz,sz] = size(A) ;  perm = 1:sz;
    rgdef = 0; tol = max(max(A))*eps; spmx = min(zz-1,sz);
    for spa = 1:spmx   % hoch, alle; quad+breit zz-1
      % ---- Pivotsuche im ganzen Feld, ev Vertauschung
      [amxsv,kxsv] = max(abs(A(spa:end,spa:end))) ;
    % Spaltenmaxima
      [amx,kx] = max(amxsv) ;
    % Maximum aller Spaltenmaxima
      ixz = spa-1+kxsv(kx); ixs = spa-1+kx;
    % Absolute Indizes
      if amx  < tol; break ; end
    % nur noch Null-Pivots, Abbruch
```

```
      if ixs > spa
    % ev. Vertauschung Spalten, (Protokoll in perm)
        A(:,[spa,ixs])=A(:,[ixs,spa]);
         perm([spa,ixs])=perm([ixs,spa]);
      end %-- ev. Vertauschung  Spalten
      if ixz > spa %  ev. Vertauschung Zeilen
        % (mit Index-Liste)
        A([spa ixz],spa:end) = A([ixz spa],spa:end);
        b([spa ixz]) = b([ixz spa]);
      end %-- ev. Vertauschung Zeilen
% Nullen unterhalb Diag erstellen,
% eigentliche Gauss-Schritte
      for zei = (spa+1):zz
% alle Zeilen unterhalb der Diagonalen
        comb = A(zei,spa)/A(spa,spa);
% Kombinationsfaktor
        A(zei,spa:end)=A(zei,spa:end)-A(spa,spa:end)*comb;
        b(zei) = b(zei) - b(spa)*comb;
      end  %for zei
      adiag = [A b]
 % Zwischen-Ausgabe der Teil-Verarbeitung
    end %-- for spa
     R = A; bt = b;
     for ztr = min(sz,zz):(-1):1
  % Nullzeilen zaehlen = rgdef
       if  norm(R(ztr,:)) > tol ; break ;
         else ; rgdef = rgdef+1 ; end
     end
   end %-- Funktion gauss2fpiv
```

13.3 Miniprojekte zur linearen Algebra

301 Allgemeines rechteckiges Widerstands-Netz

```
function [cur,M,V] = kirchgensol(hmat,vmat)
% KIRCHGENSOL [cur,M,V] = kirchgensol(hmat,vmat)
% Kirchhoff-Loesung in einem Rechteckschema
%   mit nhor x nvert quadratischen Maschen.
% hmat(nver+1,nhor) Widerstaende der horizontalen Linien
% vmat(nvert,nhor+1) Widerstaende der vertikalen Linien
[nhl,nhi] = size(hmat)
[nvi,nvl] = size(vmat)
```

```
if  nvl ~= nhi+1 | nhl ~= nvi+1
  disp('sorry')
else
% Zeilennummer = Knotennummer, dann Maschennummer
% Spaltennummer = Zweignummer Zeilenweise aus hmat, vmat
% Abfolge:  alle hoizontalen, dann alle vert., dann i0
  nk = nvl*nhl-1;
  nh = nhi*nhl;
  nv = nvl*nvi;
  ndim = nh+nv +1;
  M = zeros(ndim);
  V = zeros(nhl,nvl);
  % Knoten-Schleife
  % 1. Zeile
  % 1. Knoten lru
  M(1,ndim) = 1; M(1,1) = -1; M(1,nh+1) = -1;
  % 1. Zeile, Mittelteil lru
  for kh = 2:(nvl-1)
    M(kh,kh-1) = 1; M(kh,kh) = -1; M(kh,nh+kh) = -1;
  end
  % 1. Zeile, Endknoten  lu
  M(nvl,nvl-1) = 1;  M(nvl,nh+nvl) = -1;
%
% Mittlere Knoten-Zeilengruppe
  for zei = 2:nhl-1
    kbas = (zei-1)*nvl;
    hbas = (zei-1)*nhi;
    % vorderster Knoten ou r
    M(kbas+1,nh+kbas-nvl+1) = 1;
    M(kbas+1,nh+kbas+1) = -1;
    M(kbas+1,hbas+1) = -1;
    % mittlere Zeilen, Mittelteil ou lr
    for kh = 2:(nvl-1)
      M(kbas+kh,kbas-nvl+nh+kh) = 1;
      M(kbas+kh,kbas+kh+nh) = -1;
      M(kbas+kh,hbas+kh-1) = 1;
      M(kbas+kh,hbas+kh) = -1;
    end
    % Mittlere Zeilen, Endknoten l ou
      M(kbas+nvl,hbas+nvl-1) = 1;
      M(kbas+nvl,kbas+nh) = 1;
      M(kbas+nvl,kbas+nvl+nh) = -1;
  end
%
% unterste Zeile
```

```
    kbas = (nhl-1)*nvl;
    hbas = (nhl-1)*nhi;
    % vorderster Knoten o r
    M(kbas+1,nh+kbas-nvl+1) = 1;
    M(kbas+1,hbas+1) = -1;
    % unterste Zeilen, Mittelteil o lr
    for kh = 2:(nvl-1)
      M(kbas+kh,kbas-nvl+nh+kh) = 1;
      M(kbas+kh,hbas+kh-1) = 1;
      M(kbas+kh,hbas+kh) = -1;
    end
    % unterste Zeilen, Endknoten faellt weg
%Maschengleichungen
  ngl = nk;
  for mz = 1:nvi
    for ms = 1:nhi
      ngl = ngl  + 1;
      M(ngl,(mz-1)*nhi+ms) = hmat(mz,ms);
      M(ngl,(mz)*nhi+ms) = -hmat(mz+1,ms);
      M(ngl,nh+(mz-1)*nvl+ms) = -vmat(mz,ms);
      M(ngl,nh+(mz-1)*nvl+ms+1) = vmat(mz,ms+1);
    end
  end
% Schlussgleichung
  for k=1:nhi
     M(ndim,k) = hmat(1,k);
  end
  for k=1:nvi
     M(ndim,nh+k*nvl) = vmat(k,nvl);
  end
  b=zeros(ndim,1);
  b(ndim) = 1;
%  cur = 0
  cur=M\b;
%
    for spa = 2:nvl
      V(1,spa) = V(1,spa-1) + cur(spa-1)*hmat(spa-1,1) ;
    end
%
  for zei = 2:nhl
    V(zei,1) = V(zei-1,1) +...
       cur(nh+(zei-2)*nvl+1)*vmat(zei-1,1);
    for spa = 2:nvl
      V(zei,spa) = V(zei,spa-1) +...
       cur((zei-1)*nhi+spa-1)*hmat(zei,spa-1)
```

```
    end
  end
end % end if
```

302 Einfache und vollständige Pivot-Strategie

Die drei folgenden Quellcode-Paare zeigen der Reihe nach den simplen Gauß-Algorithmus ohne Pivot-Kontrolle (`gauss0, gaussbk0`), die Version mit einfacher Pivot-Kontrolle und Zeilenvertauschung (`gaussidx, gaussbkidx`) und den Algorithmus mit voller Pivotkontrolle und Zeilen, sowie Spaltenvertauschungen (`gaussidx2, gaussbkidx2`). Diese Serie lässt alle Matrixelemente an ihrem Platz und bewerkstelligt die Effekte von Zeilen- und Spaltenvertauschungen durch indirekte Adressierung. Deshalb gibt es zu jeder Version den passenden Code zum Rückwärts-Einsetzen, der ebenfalls mit indirekter Adressierung arbeitet.

```
function [R,b] = gauss0(Ainp,binp)
% function R = gauss0(Ainp,binp)
%  simple elementare Gauss-Elimination
  [n,ns]= size(Ainp); A = Ainp; b = binp;
  for spa = 1:(n-1)
% alle Spalten, von 1 bis (n-1)
    for zei = (spa+1):n
% alle Zeilen unterhalb der Diagonalen
      comb = A(zei,spa)/A(spa,spa);
% Faktor: Nullkandidat/Pivot
% entsprechende Zeilen kombinieren
%   (Elementaufzaehlung in Zeile
%         mit impliziter Schleife (:))
      A(zei,spa:n) = A(zei,spa:n) - A(spa,spa:n)*comb;
% Mit-Transformiren de rechten Seiten
      b(zei) = b(zei) - b(spa)*comb;
     end
   end
   R = A  % Kontrolle der R-Form
%   (Ouput i.a. unueblich in Funktion)
end % function gauss0

function x = gaussbk0(R,b)
% function x = gaussbk0(R,b)
% Rueckwaerts-Einsetzen nach Gauss-Elimination
  [n,ns] = size(R)
  x=zeros(n,1);
% x-Laenge durch Nullvektor vordefinieren
  for k=n:(-1):1
% alle Zeilen/Unbekannten von unten nach oben
% bisherige Unbekannte beruecksichtigen heisst b korrigieren
```

```
%  mit Skalarprodukt: (Matrix-Teilzeile * Vektor-Teilspalte)
    x(k) = (b(k) -  R(k,(k+1):n)*x((k+1):n) )/R(k,k)
%  (b(n) nicht korrigiert weil (k+1 > n) leere Schleife)
  end
end % function gaussbk0

function [R,b,ivv] = gaussidx(Ainp,binp)
% function R = gaussidx(Ainp,binp)
%  simple Gauss-Elimination mit indirekter Zeilen-Indizierung
%  so, dass Zeilen-Pivot-Kontrolle stattfindet
%  liefert R, b-transformiert und Vertauschungsvektor ivv
  [n,ns]= size(Ainp); A = Ainp; b = binp;  ivv = 1:n;
  for spa = 1:(n-1)     % alle Spalten, von 1 bis (n-1)
% Pivot-Kandidaten auf und unterhalb Diagonale max(abs())
    pvx = A(ivv(spa:end),spa); [~,mxp] = max(abs(pvx));
    if mxp >1 ; h = ivv(spa); ivv(spa) = ivv(mxp+spa-1);
            ivv(mxp+spa-1) = h;
    end; % Vertauschen der ivv-Werte
    for zei = (spa+1):n
% alle Zeilen unterhalb der Diagonalen
      comb = A(ivv(zei),spa)/A(ivv(spa),spa);
% Faktor: Nullkandidat/Pivot
% entsprechende Zeilen kombinieren
%   (Elementaufzaehlung in Zeile mit impliziter Schleife (:))
      A(ivv(zei),spa:n) = A(ivv(zei),spa:n) -...
           A(ivv(spa),spa:n)*comb;
% Mit-Transformiren der rechten Seiten
      b(ivv(zei)) = b(ivv(zei)) - b(ivv(spa))*comb;
     end
   end
R = A ; % Kontrolle der R-Form
% (Ouput unueblich in Funktion)
end % function gaussidx

function x = gaussbkidx(R,b,ivv)
% function x = gaussbkidx(R,b,ivv)
% Rueckwaerts-Einsetzen nach Gauss-Elimination
%   mit Vertauschung der Zeilen in R gemaess ivv
%   ivv Vektor der indirekten Zeilen-Indizierung
  [n,ns] = size(R);
  x=zeros(n,1);
% x-Laenge durch Nullvektor vordefinieren
  for k=n:(-1):1
% alle Zeilen/Unbekannten von unten nach oben
% bisherige Unbekannte beruecksichtigen heisst b korrigieren
```

```
%   mit Skalarprodukt: (Matrix-Teilzeile * Vektor-Teilspalte)
    x(k) = (b(ivv(k)) -  ...
         R(ivv(k),(k+1):n)*x((k+1):n))/R(ivv(k),k) ;
%  (b(n) nicht korrigiert weil (k+1 > n) leere Schleife)
  end
end % function gaussbkidx

function [R,b,ivv,ivh] = gaussidx2(Ainp,binp)
% function [R,b,ivv,ihv] = gaussidx2(Ainp,binp)
% Simple Gauss-Elimination mit indir. Zeilen-Indizierung
%    und indirekter Spalten-Indizierung so, dass volle
%    Pivot-Kontrolle stattfindet
%    ohne Umplatzieren der Elemente
%  liefert R, b-transformiert und
%    Vertauschungsvektoren ivv, ivh
  [n,ns]= size(Ainp);   spmx = min(n-1,ns);
  A = Ainp; b = binp;  ivv = 1:n; ivh = 1:ns;
  for spa = 1:spmx     % alle Spalten, von 1 bis (n-1)
% ---- Pivotsuche im ganzen verbleibenden Feld
    [amxsv,kxsv]=max(abs(A(ivv(spa:end),ivh(spa:end))));
%spmxv
    [amx,kx] = max(amxsv) ; % Maximum aller Spaltenmaxima
    ixz = spa-1+kxsv(kx); ixs = spa-1+kx;
% Absolute Indizes
    if ixs > spa ; h = ivh(spa); ivh(spa) = ivh(ixs);
                ivh(ixs) = h; end;
% Vertauschen der ivh-Werte
    if ixz > spa ; g = ivv(spa); ivv(spa) = ivv(ixz);
         ivv(ixz) = g; end; % Vertauschen der ivv-Werte
%
    for zei = (spa+1):n
% alle Zeilen unterhalb der Diagonalen
      comb = A(ivv(zei),ivh(spa))/A(ivv(spa),ivh(spa));
% Faktor: Nullkandidat/Pivot
% entsprechende Zeilen kombinieren
% (Elementaufzaehlung in Zeile mit impliziter Schleife (:))
      A(ivv(zei),ivh(spa:n)) = ...
          A(ivv(zei),ivh(spa:n))-A(ivv(spa),ivh(spa:n))*comb;
% Mit-Transformiren de rechten Seiten
      b(ivv(zei)) = b(ivv(zei)) - b(ivv(spa))*comb;
     end
   end
R = A ; % Kontrolle der R-Form
%  (Ouput unueblich in Funktion)
end % function gaussidx2
```

```
function x = gaussbkidx2(R,b,ivv,ivh)
% function x = gaussbkidx2(R,b,ivv,ivh)
% Rueckwaerts-Einsetzen nach Gauss-Elimination
%   mit Vertauschung der Zeilen in R gemaess ivv
%   ivv Vektor der indirekten Zeilen-Indizierung
%   und indirekter Spalten-Indizierung mit ivh
  [n,ns] = size(R);
  x=zeros(n,1);
% x-Laenge durch Nullvektor vordefinieren
  for k=n:(-1):1
% alle Zeilen/Unbekannten von unten nach oben
% bisherige Unbekannte beruecksichtigen bedeutet b korrigieren
%    mit Skalarprodukt: (Matrix-Teilzeile * Vektor-Teilspalte)
% xsel = x(ivh((k+1):n)) ;
% x-Teil unterhalb aktueller Zeile
    x(ivh(k)) = (b(ivv(k)) -  ...
     R(ivv(k),ivh((k+1):n))*x(ivh((k+1):n)))/R(ivv(k),ivh(k));
%  (b(n) nicht korrigiert weil (k+1 > n) leere Schleife)
  end
end % function gaussbkidx2
```

303 Wirkung der Asymmetrie bei Eigenwertproblemen

```
% Skript asymeigval
ndim = 5; asy = [0.01, 0.1, 0.5 ,1]
A = rand(ndim); S = A'+A;
E1 = (eig(S))'
for asy = 1:4
  Se = S; Se(ndim,1) = Se(ndim,1) + deler(asy);
  En = (eig(Se))'
end
```

13.4
Lösungen zu den Selbsttests

Lösungen zur Testserie 3.1

T311 – Rmax = 4 – Man setzt den Koeffizienten dieses Vektors verschieden von Null und alle anderen Null und erhält den Nullvektor mit Koeffizienten die nicht alle Null sind.
– DimNull(reg) = 0 – Durch Rückwärts-Einsetzen

T312 `Es=ones(n);`
```
for zei=1:n-1;for spa=zei+1:n; Es(zei,spa)=0;Es(spa,zei)=0; end;end
```

T313 `I=eye(6);I(1,1)=0;I(3,3)=0;I(5,5)=0`
```
Pa=I; Pa(3,1)=1; Pa(5,3)=1; Pa(1,5)=1;
Pb=I; Pb(5,1)=1; Pb(3,3)=1; Pb(1,5)=1;
```

T314

```
% w=angofvect(w,v) Winkel zwischen zwei Vektoren
function angval = angofvect(v,w)
angval = acos( v'*w/sqrt((v'*v)*(w'*w)) )*180/pi;
```

T315 `b=[0 0 2 1 0]'`
```
M=[1 -1 1 -1 1; 0 0 0 0 0; 16 8 4 2 1; 1 1 1 1 1; 4 3 2 1 0]
x=M\b
```

Lösungen zur Testserie 3.2

T321 – b=0 setzen

– Bildraum aufgespannt durch [1 0 0]' und [0 0 1]', Nullraum: [0 p p]'
– Turmmatrix:n, Spechtmatrix: 1 - orthogonal

T322 `T=2*eye(n)`
`for spa=3:n;T(spa-2,spa)=-1;T(spa,spa-2)=-1;  end`

T323

```
% iasymm = asymmtest(A) testet asymmetrie (=1) sonst (0)
function iasymm = asymmtest(A)
[nn,n]=size(A); iasymm = 1;
for zei = 1:n; for spa=zei:n;
 if A(spa,zei) ~= -A(zei,spa); iasymm = 0; break; end;
end; end;
```

T324 `Px=[1 0 0; 0 0 0; 0 0 0]`
`Py=[0 0 0; 0 1 0; 0 0 0]  Pz=[0 0 0; 0 0 0; 0 0 1]` alle Rang 1

T325 `n=length(v);  s = 0`
`for k=1:n; s=s+v(k)^2; end; vecnor = sqrt(s)`

14
Lösungshinweise zum Kapitel 4

14.1
Lösungen zu den allgemeinen Übungen des Kapitels 4

Winkelbestimmungen

40–1 Orthogonaler Dreispitz
Bodendreieck ABC gleichseitig, also ist$S = (0/(\sqrt{3}/3)/h)$, finde h aus SA'*SC = 0 = [-0.5 $\sqrt{3}/6$ -h]' * [0.5 $\sqrt{3}/6$ -h], h = sqrt(1/6) = 0.4082, Betrag(SA) = sqrt(1/2) = 0.7071

40–2 Berechnung der wahren Dachneigung
A,B,C,D = ($\pm$ 6 ($\pm$ 6 0), S=(0 0 10), n= SAxSB, Neigung = acos([0 0 1]*n/Betrag(n)) = 59.03°. m= SBxSC, Winkel ABS-BCS = acos(m'*n/(Betrag(n)*Betrag(m)) = 74.65°.

40–3 Winkelberechnung am Würfel
Im Würfel ABCD-EFGH bilden die Diagonalen AC, AF, AH mit der K-Diag. AG den Winkel 35.26°. BE, DE, BD stehen senkrecht auf AG.

Vektorgeometrie in der Ebene

40–4 Hesse'sche Normalform in der Ebene
OA = [4 0]'; OB = [0 3]'; AB = [-4 3]' ; N = [3 4]'; en = [3/5 4/5]';
en'*OP - en'*[4 0] = en'*OP - 12/5 = 0; en'*[0 3]' - en'*[4 0] = 0;

40–5 Visualisierung der Hesse'schen Normalform
siehe Beispiel-M-Files "showhessenf.m"

Vektorgeometrie in 3D

40–6 Normalenvektoren im Tetraeder:

A=(0 0 0) ; B= (1 0 0) ; C= (0.5 sqrt(3)/2 0) ; S= (0.5 sqrt(3)/6 sqrt(2/3)); enABC = [0 0 -0.866]' ; enABS = [0 -0.8165 0.2887]'; enACS = [-0.707 0.4082 0.2887]'; enBCS = [0.707 0.4082 0.2887]';

40–7 Gerade durch einen Oktaeder:

A=[4 0 0]', B=[0 4 0]', C=[-4 0 0]', D=[0 -4 0]', E=[0 0 4]' F=[0 0 -4]';
Linienzug: ABCDAFAEABFDEB; T= [0 /2 /2]' ; S=[4/3 4/3 4/3]; OP = OT + w*r = [0 /2 /2]'+ w*[4 4 4]'; V = [-2 -2 0] bei w=-0.5

40–8 Vierseitige Pyramide

4-seitige Pyramide aus gleichseitigen Dreiecken = oberer Teil eines Oktaeders h = s*sqrt(2)/2 = 7.07 m. ds = 7.07/tan(15*pi/180) 27.6157 m Schattengrenzen-Winkel = 2*15 Grad = 30 Grad.

40–9 Ebenes Viereck im Raum

Ebene durch ABC, Hesse'sche NF: n = cross(C-A,B-A) = [8 16 16]'; en = [1/3 2/3 2/3]' ; d = 2.666 = 8/3; x*1/3 = 8/3; x=8; sABC = [-4/3 7/3 7/3] sACD = [8/3 4/3 4/3] aABC = 24;whos aACD = 48 stot = (sABC*24+sACD*48)/(24+48)= [4/3 5/3 5/3]

40–10 Ebene in Hesse'scher Normalform

Ebene in HesseNF en=[0.64 0.48 0.60]' ; en'*OP - 1.92 = 0.

Ausflug in höhere Dimensionen

40–11 Winkel im vierdimensionalen Raum:

```
function wbk = muldiwi(v,w)
  wbk = acos( v'*w/norm(v)/norm(w))*180/pi;
end
%  alle Winkel in Grad
%   w(r,v)=30 ;  w(r,f)=45;  w(r,k1)=w(r,k3)=60;
%   w(v,f)=35.2644;  w(v,k1) = w(v,k3)=54.7356;
%   w(f,k1)=45;  w(f,k3)=90;
```

40–12 Winkel zwischen mehrdimensionalen Vektoren:

w(k,f)= 45 ; w(k,r)= 90 ; w(k,v)= 63.435 ;
w(f,r)= 90 ; w(f,v)= 50.7685; w(r,v)= 39.2315;

40–13 Zwei, drei und vierdimensionaler Einheits-„Würfel“

Quadrat-Diagonalen: [1 1]’ und [1 -1]’
Würfel-Kantenvektoren: [1 0 0]’, [0 1 0]’, [0 0 1]’, je 4-fach
Würfel-Flächendiagonalen z.B. [1 1 0]’ (2 Einsen, 1 Null), Raumdiagonalen [1 1 1]’ drei Einsen.
Bei 16 Ecken gibt es 15*16/2 Verbindungsstrecken = 120. Dabei sind 4 Typen von Kanten, je 8 fach (3 konstante Bits in allen Kombinationen), 12 Arten von Flächendiagonalen (2 bits aus 4 auswählen = 6, dann je ++ oder +-), je 4fach, 16 3D-Körperdiagonalen (3 Bit aus 4, dann +++, ++-, +-+, -++) je 2 fach (Startpunkt bei 4. Dimension 0 oder 1) und 8 Arten 4D Raumdiagonalen, je einfach. (an jedem der 4 Plätze + oder -, geteilt durch 2 wegen vorwärts- rückwärts Identität). Also, 8+32+48+32=120.
Winkel Flächendiagonalen$\pm 45^o$ $\pm 135^o$ $\pm 90^o$
Winkel 3D-Raumdiagonalen$\pm 54.73^o$, $\pm 125.26^o$ $\pm 90^o$
Winkel 4D-Raumdiagonalen$\pm 60^o$, $\pm 120^o$

Durch Matrizen definierte Abbildungen

40–14 Kongruente Abbildungen der Ebene

```
My =   [-1 0 ; 0 1];   Lmy = [ -5 -5 -6; 2 0 0];
Mp =   [-1 0 ; 0 -1];  Lmp = [ -5 -5 -6; -2 0 0];
R45 =  [0.707 -0.707; 0.707 0.707 ];
Lr45 = [ 2.12 3.53 4.24; 4.45 3.53 4.24];
R90 =  [0 -1; 1 0];    Lr90 = [-2 0 0;  5 5 6];
```

Homogene Koordinatentransformationen in der Ebene

40–15 2D homogene Koordinatentransformationen des „L“

```
Ta1 = [1 0 -5 ; 0 1 5; 0 0 1];
Ta2 = [1 0 -5 ; 0 1 0; 0 0 1];
Ta3 = [1 0 -10 ; 0 1 0; 0 0 1];
Tb = [0 -1 0; 1 0 0; 0 0 1];
Tc1 = [0.707 -0.707 1.465; 0.707 0.707 -3.54; 0 0 1];
Lc1 = [3.58 5 5.707; 1.414 0 0.707; 1 1 1];
Tc2 = [1 -1 5; 1 0 -5; 0 0 1];
Lc2 = [ 3 5 5; 0 0 1; 1 1 1]}
```

40–16 Drehungen und Translationen des „L“

```
Ra1 = [0.707 -0.707 0; 0.707 0.707 0; 0 0 1];
Ra2 = Ra1^2; Ra3 = Ra1^3;
La1 = [ 2.12 3.54 4.24; 4.95 3.54 4.24; 1 1 1];
```

```
Tb1 = [1 0 -3.54; 0 1 -3.54; 0 0 1];
Tb1b= [1 0  3.54; 0 1  3.54; 0 0 1];
La2 = [-2 0 0; 5 5 6; 1 1 1]
Tb2 = [1 0 0; 0 1 -5; 0 0 1]
Tb2b= [1 0 0; 0 1  5; 0 0 1]
La3 = [ -4.95 -3.54 -4.24; 2.12 3.54 4.24;1 1 1]
Tb3 = [1 0  3.54; 0 1 -3.54; 0 0 1];
Tb3b= [1 0 -3.54; 0 1  3.54; 0 0 1];
Rb1 = [0.707 0.707 0; -0.707 0.707 0; 0 0 1];
Rb2 = Rb1^2; Rb3 = Rb1^3;
G1 = Tb1b*Rb1*Tb1*Ra1  % = [ 1 0 -1.47; 0 1 3.54; 0 0 1]
G2 = Tb2b*Rb2*Tb2*Ra2  % = [ 1 0  -5; 0 1 5; 0 0 1]
G3 = Tb3b*Rb3*Tb3*Ra3  % = [ 1 0 -8.54; 0 1 3.54; 0 0 1]
```

40–17 2D-Abbildung des „L“ mit Gegenrotation

Interessante Lösungsvariante: zuerst Gegenrotation um Ecke des “L”
Tc = [1 0 -5; 0 1 0 ; 0 0 1] ; R = [cos(w) sin(w) 0; -sin(w) cos(w) 0 ; 0 0 1];
Tb = [1 0 5; 0 1 0 ; 0 0 1] ;
Gesamt Gegendrehung G= Tb*R*Tc
Dann Drehung um (0/0) um w R = [cos(w) -sin(w) 0; sin(w) cos(w) 0 ; 0 0 1]

40–18 2D-Bewegung des „L“ mit allgemeinem Winkelparameter a) Tges = [1 0 5*(cos(w)-1); 0 1 5*sin(w) ; 0 0 0]
b)Tr2= [cos(w) sin(w) 5*(cos(2*w)-1); -sin(w) cos(w) 5*sin(2*w) ; 0 0 1]

40–19 Dreiecksabbildung auf sich selbst:

```
Dr = [0 10 5; 0 0 5*sqrt(3); 1 1 1] ;
T = [1 0 -5; 0 1 -5*sqrt(3)/3; 0 0 1]
w = 2*pi/3 ;
R = [cos(w) -sin(w) 0; sin(w) cos(w) 0; 0 0 1]
Tb = [1 0 5; 0 1 5*sqrt(3)/3; 0 0 1];
TT = Tb*R*T; TT^3
Dr2 = TT*Dr ;  Dr3 = TT^2*Dr;
[Dr Dr2 Dr3]
```

40–20 Mehrfache Abbildung eines Rechtecks

```
ABCD = [0 10 10 0 0; 0 0 2 2 0; 1 1 1 1 1];
T1a = [1 0 0; 0 1 -2; 0 0 1];
T1b = [0 -1 0; 1 0 2; 0 0 1];
ABCD1 = T1b*T1a*ABCD
 T2a = [1 0 -2; 0 1 -12; 0 0 1];
T2b = [0 -1 2; 1 0 12; 0 0 1];
```

```
ABCD2 = T2b*T2a*ABCD1
T3a = [1 0 -12; 0 1 -10; 0 0 1];
T3b = [0 -1 12; 1 0 10; 0 0 1];
ABCD3 = T3b*T3a*ABCD2
T4a = [1 0 -10; 0 1 0; 0 0 1];
T4b = [0 -1 10; 1 0 0; 0 0 1];
T4b*T4a* T3b*T3a* T2b*T2a*T1b*T1a
ABCD4 = T4b*T4a*ABCD3
plot(ABCD(1,:),ABCD(2,:),'k'); hold on;
plot(ABCD1(1,:),ABCD1(2,:),'r')
plot(ABCD2(1,:),ABCD2(2,:),'g');
plot(ABCD3(1,:),ABCD3(2,:),'b'); axis equal
```

40–21 Zwei verschiedene Abbildungen überdecken sich

```
F = [0 8 8 0 0; 0 0 4 4 0; 1 1 1 1 1];
Tr4 = [1 0 0; 0 1 -4; 0 0 1]
Mi = [1 0 0; 0 -1 4; 0 0 1]; F1 = Mi*Tr4*F
Tr44 = [1 0 -4; 0 1 -4; 0 0 1];
R = [-1 0 4; 0 -1 4; 0 0 1];
F2 = R*Tr44*F
```

Korrespondenz: A1=B2, B1=A2, C1=D2, D1=C2.

Homogene Koordinatentransformationen im Raum

40–22 3D-Abbildung eines Dreibeins

```
A=[0 0 0 1]'; B=[10 0 0 1]'; C=[0 10 0 1]'; D=[0 0 10 1]';
Db = [B A C B D A C D];
Mp = [[-eye(3) [0 0 0]']; 0 0 0 1]; Dbt = Mp*Db;
plot3(Db(1,:), Db(2,:), Db(3,:)); hold on;
plot3(Dbt(1,:), Dbt(2,:), Dbt(3,:),'r');   axis equal
set(gca,'CameraPosition',[-150 45 70]); hold off
```

40–23 3D Abbildung zwischen den Stufen einer Wendeltreppe

```
Q=[50 50 50 50 150 150 150 150 150 150  50 50  50 150 150 50 50; ...
   50  0  0  0   0   0   0  50  50  50  50 50   0   0  50 50  0 ; ...
    0  0 18  0   0  18   0   0  18   0   0 18  18  18  18 18 18 ; ...
    1  1  1  1   1   1   1   1   1   1   1  1   1   1   1  1  1  ];
plot3(Q(1,:),Q(2,:),Q(3,:),'k');  hold on
w = 15*pi/180;
```

```
Trl = [cos(w) -sin(w) 0 0; sin(w) cos(w) 0 0; 0 0 1 18; 0 0 0 1];
Q1 = Trl*Q;  plot3(Q1(1,:),Q1(2,:),Q1(3,:),'k');
Q2 = Trl*Q1; plot3(Q2(1,:),Q2(2,:),Q2(3,:),'k');
Q3 = Trl*Q2; plot3(Q3(1,:),Q3(2,:),Q3(3,:),'k');
Q4 = Trl*Q3; plot3(Q4(1,:),Q4(2,:),Q4(3,:),'k');
Q5 = Trl*Q4; plot3(Q5(1,:),Q5(2,:),Q5(3,:),'k');
Q6 = Trl*Q5; plot3(Q6(1,:),Q6(2,:),Q6(3,:),'k');
axis equal; hold off
```

40–24 Sich expandierender Würfel

```
W = [-2 -2 -2 -2  2  2  2  2  2  2 -2 -2 -2  2  2 -2 -2 ;...
      2 -2 -2 -2 -2 -2  2  2  2  2  2 -2 -2  2  2 -2 ;...
     -2 -2  2 -2 -2  2 -2 -2  2 -2 -2  2  2  2  2  2  2 ;...
      1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 ]
plot3(W(1,:),W(2,:),W(3,:),'k');  hold on
for ix = 0:1 ; for iy = 0:1; for iz = 0:1
t = [2-4*ix 2-4*iy 2-4*iz]'; Ta=[eye(3) t; 0 0 0 1];
Tb = [-eye(3) -t; 0 0 0 1]; Wt = Tb*Ta*W;
plot3(Wt(1,:),Wt(2,:),Wt(3,:),'r');
end; end; end; axis equal; hold off
```

40–25 3D: “L” dreht sich vor dem Spiegel:

Zuerst die Hilfsfunktion plothcl3:

```
% plothandle = plothcl3(Lmat,colorstring)
% 3D-Plot einer Linie aus einer
%  Matrix von Spaltenvektoren
%   in 3D homogenen Koordinaten
function plothandle = plothcl3(lmat,colorstring)
if exist('colorstring') == 0
  col = 'k';
else
  col = colorstring;
end
plothandle = plot3(lmat(1,:),lmat(2,:),lmat(3,:),col);
end % function
```

Dann die eigentliche Lösung, ein Film durch abwechselndes Löschen und neu Zeichnen:

```
% lrotmirr3D.m - 3D "L" dreht sich vor dem Spiegel
spiegel=[0 4 4 0 0;0 0 0 0 0;0 0 3 3 0;1 1 1 1 1];
% Randpunkte des Spiegels und Spiegelmatrix
spm=[1 0 0 0;0 -1 0 0;0 0 1 0;0 0 0 1];
plothcl3(spiegel,'b') ; hold on
axis ([-6 6,-6 6 0 12])  ; axis square
set(gca,'CameraPosition',[-15 -100 27])
```

```
% Drehen des "L" in Schritten von 5 Grad
for lw=0:2:360
  w=lw*pi/180;  kopf= [2 -4 2 1]';
  ecke=[2 -4 0 1]';  zehe=[2+cos(w) -4+sin(w) 0 1]';
% "L" zusammensetzen und spiegeln
  LM=[kopf ecke zehe]; LV=spm*LM;
  hm=plothcl3(LM,'k');
  hv=plothcl3(LV,'r');
  pause(0.1)
  if lw < 360
    delete(hv)
    delete(hm)
  end
end
```

40–26 Fünf Sterne in 3 Dimensionen

```
A=[-1 -1 -1 1]'; B=[1 -1 -1 1]';
C=[1 1 -1 1]';D=[-1 1 -1 1]';
E=[-1 -1  1 1]'; F=[1 -1  1 1]';
G=[1 1  1 1]';H=[-1 1  1 1]';
T=[0 0 6 1]'; U=[0 0 -6 1]';
N=[-6 0 0 1]'; S=[6 0 0 1]';
O=[0 6 0 1]'; W=[0 -6 0 1]';
liz = [E T G F T H E N D H N A E F S C G S B F ...
       W A B W E A U C D U  B C O H G O D C D A]
plot3(liz(1,:),liz(2,:),liz(3,:),'k') ; hold on
cols = ['r','b';'g','c']
sh = -15;
for k=1:2
  Tr= eye(4);   Tr(1,4)= sh;   li = Tr*liz;
  plot3(li(1,:),li(2,:),li(3,:),cols(1,k))
  sh = 15;
end
sh = -15;
for k=1:2
  Tr= eye(4);   Tr(2,4)= sh;   li = Tr*liz;
  plot3(li(1,:),li(2,:),li(3,:),cols(2,k))
  sh = 15;
end ; axis equal ; hold off
```

40–27 Ansicht aus dem Helikopter

```
% showhaus.m
```

```
v1 = [ 3 -2 0  1]' ;
v2 = [ 3 -2 3  1]' ;
v3 = [ 3  0 5  1]' ;
v4 = [ 3  2 3  1]' ;
v5 = [ 3  2 0  1]' ;

v6 = [-3 -2 0  1]' ;
v7 = [-3 -2 3  1]' ;
v8 = [-3  0 5  1]' ;
v9 = [-3  2 3  1]' ;
v10= [-3  2 0  1]' ;
%
L  = [ v1 v6 v7 v2 v3 v8 v9 v4 v5 ...
      v1 v2 v3 v4 v5 v10 v6 v7 v8 v9 v10 ];
%
lxi = L(1,:) ;
lyi = L(2,:) ;
lzi = L(3,:) ;

% -------------start -----file showhaus.m ------------
% Dieses File 'showhaus.m' fasst die Erstellung des
% Linienzuges in 3D homogenen Koordinaten,
%  die Matrizen-Berechnungen, die Transformationen
% und die Darstellung der Projektion
%    in die y'' z''-Ebene zusammen

%  m-file 'hausdef.m' definiert 4x20 Matrix der
%    Kanten-Lininen in 3D homogenen Koordinaten

 hausdef

% die m-files 'zrotmat.m' und  'yrotmat.m' definieren
%   die Rotationsmatrizen  'Zrm' und 'Yrm' aufgrund der
%   globalen Variabeln 'phi' und 'tet'

 zrotmat
 yrotmat

% Die Transformation von L  zu Lt

 Lt =  Yrm*Zrm*L ;

% Herauspicken der 20 y und y-Koordinaten

lyt = Lt(2,:);
```

```
lzt = Lt(3,:);

% And: the show must go on!

plot(lyt,lzt)
axis( [-6 6 -4 8] )
axis square

% -------------end -----file showhaus.m ------------
```

Die zwei Skripte yrotmat und zrotmat

```
% Skript yrotmat
Yrm = eye(4);

% globale Variable tet (in Grad)

Yrm(1,1) = cos(tet*pi/180);
Yrm(3,3) = cos(tet*pi/180);

Yrm(1,3) = sin(tet*pi/180);
Yrm(3,1) = -sin(tet*pi/180);

% Skript zrotmat
Zrm = eye(4);
%  globale Variable uebernommen: phi (in Grad)
Zrm(1,1) = cos(phi*pi/180);
Zrm(2,2) = cos(phi*pi/180);
Zrm(1,2) = -sin(phi*pi/180);
Zrm(2,1) =  sin(phi*pi/180);
```

14.2
Miniprojekte zur Raumgeometrie und den Abbildungen

401 Generieren der fünf Platonischen Körper

Die Funktion platoshow(n,e) erwartet die Werte n für die Anzahl Ecken der Polygone im Platonischen Körper und e für die Anzahl Polygone, die in einer Ecke zusammenkommen.

```
function r = platoshow(n,e)
% platobuild.m Erzeugen der Platonischen Koerper
%  mit Elementar-Rotationen
%  n = Anzahl Ecken der Polygone,
%  e = Anzahl Polygone pro Ecke
%  benoetigt:
```

```
%  RZ.m, addanode.m, goback.m, addapoint.m
% n = input('Erzeugen der Platonischen Koerper: ...
 n  der n-Ecke = ?');
% e = input(' wieviele Flaechen stossen an ...
einer Ecke zusammen?');
% Legalitaetspruefung und Verbindungstopologie
cond = 0;
if n < 6 && n > 2
 if n > 3 && e == 3
   cond = 1;
   if n == 4
     % Hexaeder , Wuerfel
     typ = 3;
'Hexaeder, Wuerfel e=8, k=12, f=6 '
   else
     % dodekaeder
     typ = 5;
'Dodekaeder, Pentagon-Dodekaeder, e=20, k=30, f=12 '
   end
 else
  if (n == 3) && (e > 2) && (e < 6)
    cond = 1;
    if e == 3
      %  tetraeder
      typ = 1;
'Tetraeder, e=4, k=6, f=4'
    else
      if e == 4
        %  oktaeder
        typ = 2;
'Oktaeder,  e=6, k=12, f=8'
      else
%  ikosaeder
        typ = 4;
'Ikosaeder, e=12, k=30, f=20'
      end
    end
  end
 end
end
%
cols = ['r','g','b','c','m'];
lpa = zeros(1,e+1);
if cond == 1
k=1;  % Kantenlaenge
```

```
rf = k/2/sin(pi/n) ;
se = 2*rf*sin(2*pi/n);
rp = se/2/sin(pi/e) ;  % Pyramidenradius
hp = sqrt(1-rp^2);
%rf ; rp; hp;
rk = k^2/2/hp;   % Radius der umschreibenden Kugel
rl= 1.2*rk;   % Plotgrenzen
'Radius der umschreibenden Kugel:', rk
wk = 2*asin(k/2/rk);
wkg = wk*180/pi;
% dreibein
figure(1);
clf;
tri = [ 0 1.2 0  0   0 0; ...
        0 0   0  1.2 0 0; ...
        0 0   0  0   0 1.2];
%
plot3( tri(1,:),tri(2,:),tri(3,:),'k','linewidth',1.3 );
hold on
axis ([ -1.6 1.6 -1.6 1.6 -1.6 1.6]);
axis square
view(70,22)
plhand = 0;
% oberster Punkt
ppo = [0 0 rk]';
plot3( ppo(1,1),ppo(2,1),ppo(3,1),'ok' );
% Grundtransformation
trtot = eye(3);
  Royf = [cos(wk/2) 0 -sin(wk/2)  ;...
          0 1 0; sin(wk/2) 0 cos(wk/2)  ];
  Rozf = [ cos(k*2*pi/e) sin(k*2*pi/e) 0 ; ...
    -sin(k*2*pi/e) cos(k*2*pi/e) 0 ; 0 0 1 ];
  Rozh = [-1 0 0   ; 0 -1 0; 0 0 1  ];
%
rzac = 1;
if typ == 1
  ppt = ppo;
  ndepth = 1;
  bkmat(1:3,1:3,ndepth) = eye(3);
%
  i1 = 1; i2 = 0;
  for nod = 1:3
    RZ ; addanode;
%
       RZ ; addanode;
```

```
      goback; addapoint;
%
    goback; addapoint;
    sz = size(ppt); if i2 == 0 i2 = sz(2); end
    plot3(ppt(1,i1:i2),ppt(2,i1:i2),ppt(3,i1:i2),...
  cols(nod),'LineWidth',2);
   pause(1)
  end
%
elseif typ == 2
  ppt = ppo;
  ndepth = 1;
  bkmat(1:3,1:3,ndepth) = eye(3);
%
  i1 = 1; i2 = 0;
  for nod = 1:4
    RZ ; addanode;
%
    for nodb = 1:2
              RZ ; addanode;
      goback; addapoint;
    end
%
    goback; addapoint;
    sz = size(ppt);    if i2 == 0 i2 = sz(2); end
    plot3(ppt(1,i1:i2),ppt(2,i1:i2),ppt(3,i1:i2), ...
  cols(nod),'LineWidth',2);
    pause(1)
  end
elseif typ == 3
  ppt = ppo;
  ndepth = 1;
  bkmat(1:3,1:3,ndepth) = eye(3);
%
  i1 = 1; i2 = 0;
  for nod = 1:3
    RZ ; addanode;
%
    RZ ; addanode;
%
    for nodb = 1:2
      RZ ; addanode;
      goback; addapoint;
    end
%
```

```
    goback; addapoint;
%
    goback; addapoint;
    sz = size(ppt); if i2 == 0 i2 = sz(2); end
    plot3(ppt(1,i1:i2),ppt(2,i1:i2),ppt(3,i1:i2),...
      cols(nod),'LineWidth',2);
   pause(1)
  end
elseif typ == 4
  ppt = ppo;
  ndepth = 1;
  bkmat(1:3,1:3,ndepth) = eye(3);
%
  i1 = 1; i2 = 0;
  for nod = 1:5
    RZ ; addanode;
%
    RZ ; addanode;
%
    goback; addapoint;
%
    RZ ; addanode;
%
    for nodb = 1:3
      RZ ; addanode;
      goback; addapoint;
    end
%
    goback; addapoint;
%
    goback; addapoint;
    sz = size(ppt); if i2 == 0 i2 = sz(2); end
    plot3(ppt(1,i1:i2),ppt(2,i1:i2),ppt(3,i1:i2), ...
  cols(nod),'LineWidth',2);
   pause(1)
  end
elseif typ == 5
  ppt = ppo;
  ndepth = 1;
  bkmat(1:3,1:3,ndepth) = eye(3);
%
  i1 = 1; i2 = 0;
  for nod = 1:3
    RZ ; addanode;
%
```

```
      RZ ; addanode;
%lev2
          for nodb = 1:2
% 2x lev3 am 1. lev2
            RZ ; addanode;
            goback; addapoint;
          end

      goback; addapoint;

      RZ ; addanode;
%lev2
        RZ ; addanode;
% lev3
%   1. lev 4 am lev3
          RZ ; addanode;
          goback; addapoint;
% 2. lev 4
          RZ ; addanode;

            for nodc = 1:2
%  lev5 2x
              RZ ; addanode;
              goback; addapoint;
            end

          goback; addapoint;

        goback; addapoint;

      goback; addapoint;

%
    goback; addapoint;
    sz = size(ppt); if i2 == 0 i2 = sz(2); end
    plot3(ppt(1,i1:i2),ppt(2,i1:i2),ppt(3,i1:i2),...
      cols(nod),'LineWidth',2)
    pause(1)
  end  % for
end

else
 input('sorry, diesen regulaeren Polyeder gibt es nicht!')
end
```

Dazu werden folgende Skripts (im gleichen Directory) benötigt:

```
% RZ.m Z-Rotation - Hilfs-Skript zu platobuild.m
bkmat(1:3,1:3,ndepth) = Rozf^rzac*bkmat(1:3,1:3,ndepth);
ppt = Rozf^rzac*ppt;

% addanode.m  Neuen Knoten anfuegen,
%   Hilfsskript zu platobuild.m
ndepth = ndepth+1;
bkmat(1:3,1:3,ndepth) = Royf^2*eye(3);
ppt = [Royf^2*ppt ppo];
bkmat(1:3,1:3,ndepth) = Rozh*bkmat(1:3,1:3,ndepth) ;
ppt = Rozh*ppt;

%  addapoint.m  Hilfsskript zu platobuild.m
ppt = [ppt ppo];

% goback.m gehe zum vorherigen Knoten,
%  Hifsskript zu platobuild.m
ppt = (bkmat(1:3,1:3,ndepth)^-1)*ppt ;
ndepth = ndepth-1;
```

402 Suchen der Großkreis-Flugrouten

Lösung mit mehreren Hilfsfunktionen:

```
% breitkreisfct.m Funktion zum
%    Erzeugen der Koordinatentripel
%    eines Breitnmeridians der Breite lati
function coordmat = breitkreisfct(lati)
t = linspace(0,2*pi);
coordmat=[cos(t)*cos(lati*pi/180); ...
sin(t)*cos(lati*pi/180);...
 sin(lati*pi/180)*ones(1,length(t)) ];

% grosskreisfct.m Funktion zum
%   Erzeugen der Koordinatentripel
%    eines Laengenmeridian-Sektors zur Laenge long
%    und der Winkeldistanz dist ab dem Nordpol
function coordmat = grosskreisfct(long,dist)
t = linspace(0,dist*pi/180);
coordmat = [sin(t)*cos(long*pi/180); ...
          sin(t)*sin(long*pi/180); cos(t)];

% meridiannetz.m  Zeichnen eines Meridian-Netzes
hold on
for long = 0:30:90
```

```
  pl = grosskreisfct(long, 360);
  plot3(pl(1,:),pl(2,:),pl(3,:),'b' )
  plot3(-pl(1,:),pl(2,:),pl(3,:),'b' )
  if long == 0
    plot3(pl(1,:),pl(2,:),pl(3,:),'k','LineWidth',2 )
  end
end
for lat = -60:15:60
  pb = breitkreisfct(lat);
  plot3(pb(1,:),pb(2,:),pb(3,:),'b' )
  if lat == 0
    plot3(pb(1,:),pb(2,:),pb(3,:),'k','LineWidth',2 )
  end
end
axis([-1.2 1.2 -1.2 1.2 -1.2 1.2]); axis square;
view(70,22)

function [dist,azi,maxbreit,norvec] = ...
 grosskreis(long1,lat1,long2,lat2)
%[dist,azi,maxbreit,norvec] =
%   grosskreis(long1,lati1,long2,lati2)
% Berechnung von Distanz, Startazimut,
% maximaler Breite und Normalenvektor
% zur Grosskreis-Flugroute zwischen zwei
% Orten auf der Welt, deren
% Laengenposition (Ost=+, West= -) und
% Breitenposition (N=+) bekannt ist.
dgr = pi/180;
v1=[cos(lat1*dgr)*cos(long1*dgr) ...
      cos(lat1*dgr)*sin(long1*dgr) ...
     sin(lat1*dgr)]';
v1n=[-sin(lat1*dgr)*cos(long1*dgr)...
        -sin(lat1*dgr)*sin(long1*dgr) ...
     cos(lat1*dgr)]';
v2=[cos(lat2*dgr)*cos(long2*dgr)...
      cos(lat2*dgr)*sin(long2*dgr) ...
      sin(lat2*dgr)]';
dist=abs(acos(v1'*v2))*6366.2;
% Winkel in rad = acos(Skalarprodukt)
if dist > 0   % Ausschliessen identische Punkte
  norvec = cross(v1,v2);
  if norm(norvec) > 0;   % Ausschliessen Antipoden
     norvec = norvec/norm(norvec);
     maxbreit =  acos(norvec(3)/norm(norvec))/dgr;
     takoff = cross(norvec,v1);
```

```
     norcomp =  (takoff'*v1n/norm(v1n))*v1n;
     westcomp = takoff-norcomp;
     azi = acos(takoff'*v1n/norm(takoff)/norm(v1n))/dgr;
     if long2 > long1 ;
        azi = -azi;
     else
        maxbreit = 180-maxbreit;
     end;
   else; disp('Antipoden!'); end;
else;  disp('identische Punkte!');  end

% plk=flugfct(long,lat,azi,dist)
% Flugroute ab Startort long lat
%   mit Abflug-Azimuth azi und Flugdistanz dist
function plk=flugfct(long,lat,azi,dist)
pm = grosskreisfct(180+azi, dist);
wy = (-90+lat)*pi/180;
Rym = [ cos(wy) 0 -sin(wy); 0 1 0; sin(wy) 0 cos(wy)];
wz = long*pi/180;
Rzm = [ cos(wz)  -sin(wz) 0 ;  sin(wz)  cos(wz) 0 ; 0 0 1];
plk = Rzm*Rym*pm;

% showflug.m Beipiel-Plots f\"ur Flugrouten
clf
meridiannetz;
%Start = Frankfurt
longs = 8.6833; lats = 50.1167;
%      Ankara,   Athen,   Berlin,
%      Kairo,   Kapstadt, London,
%       Bejing, Moskau, Rio, Sydney ,
%      Tokyo,  Frisco,  NYC , Chicago
lat = [39.9167  37.9667    52.5  ...
        30.0333   -35.9167  39.9167 ...
         55.75  -22.95  -34.0  35.6667 ...
           37.6167  40.65  41.7833   ];
long = [32.9167  23.7167  13.4167 ...
         31.35   18.3667  ...
        116.25  37.6     -43.2 ...
        151.0  139.75  -122.3833  -73.7833 -95.35];
for k = 1:length(long)
  [dist,azi,maxbreit,norvec] =...
      grosskreis(longs,lats,long(k),lat(k));
  pl = flugfct(longs,lats,azi,dist/6336.2*180/pi);
  plot3(pl(1,:), pl(2,:), pl(3,:),'r','LineWidth',2 );
end
hold off
```

403 Visualisieren eines Tetraeder-Rings beim Umstülpen

```
% tetringfilm.m  Simulation des Umstuelpens eines Rings
%   aus 6 verlaengerten Tetraedern
s = 1;
h = sqrt(3)/2;
d = 1.0;
f = 0.93;
%
p1 = [d 0 0 0]' + [0 h 0 0]' + [0 0 s/2 0]' + [0 0 0 1]';
p2 = [d 0 0 0]' + [0 h 0 0]' - [0 0 s/2 0]' + [0 0 0 1]';
p3 = [d-s/2 0 0 1]';
p4 = [d+s/2 0 0 1]';
zt = (p1 + p2 + p3 + p4)/4;
zt(4) = 1;
Zm = zt * ones(1,8);
To= [p1 p2 p3 p1 p4 p3 p4 p2];
%
px0 = [0 0 0]';
px1 = [s 0 0]';
px2 = [-s/2 s*sqrt(3)/2 0]';
px3 = [-s/2 -s*sqrt(3)/2 0]';
pxo = [ 0 0 1]';
pxu = [0 0 -1]';
%
Xa = 1.1*[px0 px1 px0 px2 px0 px3 px0 pxo pxu];
%
plot3( Xa(1,1:7), Xa(2,1:7), Xa(3,1:7),'k' );
     view(-53, 42);
To  = Zm + f*(To-Zm);
Tk = [p1 p2 p3 p1 p4 p3 p4 p2];
%plot3(To(1,:), To(2,:), To(3,:) )
%plot3(Tk(1,:), Tk(2,:), Tk(3,:),'g' )
axis([-1.2 1.2 -1.2 1.2 -1.2 1.2]);
axis square;
hold on;
%
Mv = [1 0 0 -d ; 0 1 0 0; 0 0 1 0; 0 0 0 1] ;
Mi =  [1 0 0 0 ; 0 -1 0 0; 0 0 1 0; 0 0 0 1] ;
Rt =  [cos(120*pi/180) -sin(120*pi/180)  0 0 ;
       sin(120*pi/180) cos(120*pi/180) 0 0;...
        0 0 1 0; 0 0 0 1] ;
%
w=(0:2:360)*pi/180;
```

```
al = atan(sqrt(3)*sin(w));
xca = h*(cos(al)/sqrt(3) + tan(al)/sqrt(3).*sin(al));
%
for k = 1:length(w)
  Ra = [ 1 0 0 0; 0 cos(al(k)) -sin(al(k)) 0;...
       0 sin(al(k)) cos(al(k)) 0 ;
         0 0 0 1] ;
  Rw = [ cos(w(k)) 0 -sin(w(k))  0; 0 1 0 0;...
          sin(w(k)) 0 cos(w(k)) 0;
          0 0 0 1];
  xc = h* (  cos(w(k))/sqrt(3) + sin(w(k))*sin(al(k)) ) ;
  Mb = [1 0 0 xca(k) ; 0 1 0 0; 0 0 1 0; 0 0 0 1] ;

  Tt = Mb*Rw*Ra*Mv*To;
  Tf = Mb*Rw*Ra*Mv*Tk;
  if k > 1
    delete(plhd)
    delete(plhdf)
    delete(plhdg)
    delete(plhdm)
    delete(plhdmf)
    delete(plhdmg)
    delete(plhd1)
    delete(plhdm1)
    delete(plhd1f)
    delete(plhd1g)
    delete(plhdm1f)
    delete(plhdm1g)
    delete(plhd2)
    delete(plhdm2)
    delete(plhd2f)
    delete(plhd2g)
    delete(plhdm2f)
    delete(plhdm2g)
  end
  plhd = plot3(Tt(1,:), Tt(2,:), Tt(3,:), 'r' );
  plhdf = plot3(Tf(1,1:2), Tf(2,1:2), Tf(3,1:2), 'b' );
  plhdg = plot3(Tf(1,6:7), Tf(2,6:7), Tf(3,6:7), 'b' );
%
  Tm = Mi*Tt;
  Tmk = Mi*Tf;
  plhdm = plot3(Tm(1,:), Tm(2,:), Tm(3,:), 'r' );
  plhdmf = plot3(Tmk(1,1:2), Tmk(2,1:2), Tmk(3,1:2), 'b' );
  plhdmg = plot3(Tmk(1,6:7), Tmk(2,6:7), Tmk(3,6:7), 'b' );
  T1 = Rt*Tt;
```

```
  Tm1 = Rt*Tm;
  T1k = Rt*Tf;
  Tm1k = Rt*Tmk;
  plhd1 = plot3(T1(1,:), T1(2,:), T1(3,:), 'r' );
  plhd1f = plot3(T1k(1,1:2), T1k(2,1:2), T1k(3,1:2), 'b' );
  plhd1g = plot3(T1k(1,6:7), T1k(2,6:7), T1k(3,6:7), 'b' );
  plhdm1 = plot3(Tm1(1,:), Tm1(2,:), Tm1(3,:), 'r' );
  plhdm1f = plot3(Tm1k(1,1:2), Tm1k(2,1:2), Tm1k(3,1:2), 'b' );
  plhdm1g = plot3(Tm1k(1,6:7), Tm1k(2,6:7), Tm1k(3,6:7), 'b' );
  T2 = Rt*Rt*Tt;
  Tm2 = Rt*Rt*Tm;
  T2k = Rt* Rt*Tf;
  Tm2k = Rt*Rt*Tmk;
  plhd2 = plot3(T2(1,:), T2(2,:), T2(3,:), 'r' );
  plhd2f = plot3(T2k(1,1:2), T2k(2,1:2), T2k(3,1:2), 'b' );
  plhd2g = plot3(T2k(1,6:7), T2k(2,6:7), T2k(3,6:7), 'b' );
  plhdm2 = plot3(Tm2(1,:), Tm2(2,:), Tm2(3,:), 'r' );
  plhdm2f = plot3(Tm2k(1,1:2), Tm2k(2,1:2), Tm2k(3,1:2), 'b' );
  plhdm2g = plot3(Tm2k(1,6:7), Tm2k(2,6:7), Tm2k(3,6:7), 'b' );
%
  if k<2
  pause(0.8)
   else
  pause(0.2)
  end
end
%
hold off
```

404 Die Bewegung eines Autos auf einer Parkhausrampe

```
% garagefilm.m simple Simulation der Bewegung
% eines vereinfachten Autos auf einer
% Parkhausrampe
P=[5.15 6.85 6.85 5.15 5.15 ;
    -0.6  -0.6    4.2    4.2    -0.6;
    0.3   0.3  0.3  0.3  0.3;
    1 1 1 1 1];
t = (0:0.01:1)*2*pi;
xi = 4.2*cos(t);
yi = 4.2*sin(t);
xa = 8.5*cos(t);
ya = 8.5*sin(t);
z = t/2/pi*3;
```

```
plot3(xi,yi,z);
hold on
plot3(xa,ya,z);
axis ([-10 10 -10 10 -2 18]);
axis square
for k=1:101
  tx = t(k);
  T = [cos(tx) -sin(tx) 0  0; ...
     sin(tx) cos(tx) 0 0; 0 0 1 z(k); 0 0 0 1];
  Q=T*P;
  if k > 1
    delete(ph)
  end
  ph = plot3(Q(1,:),Q(2,:),Q(3,:),'r');
  pause(0.1)
end
hold off
```

14.3
Lösungen zu den Selbsttests im Kapitel 4

Lösungen zum Selbsttests 4.1

T411 –`sqrt(v'*v)` – Die unterste Zeie ist [0 0 1] –`P=[-1 0 0; 0 -1 0; 0 0 1]` –`v*v'/(v'*v)`

T412 `[-4 -2 0]', [-2 0 2]', [0 2 4]'`

T413 `en = [0.48 0.64 0.6]'`

`E: en'*OP -1.92 = 0  F: en'*OP -3.84 = 0`

T414 `en1 = [0 -0.971 0.2425]' [-0.971 0 0.2425]'`

Neigung, beide 75.96°, Zwischenwinkel 86.63°.

T415 `V=[1 0 -4; 0 1 -4; 0 0 1]`

`R=[0 -1 4; 1 0 4; 0 0 1],  T=R*V= [0 -1 8; 1 0 0; 0 0 1]  R^4=I`

Lösungen zum Selbsttests 4.1

T421 $-|v| \cdot |w|\sin(\alpha)$ – Der Abbildungsteil kongruenter Abbildungsmatrizen ist orthogonal, darum sind sie Längen- und Winkeltreu. – `R=[0 -1 0; 1 0 0; 0 0 1]}` – Die Ebenen liegen auf verschiedenen Seiten des Koordinatenursprungs.

T422 `R=[0.5 0.5 0]' und S=[0.5 0.5 1]'`

T423 Distanz = 9.6, OF=[5.76 6.144 4.608]'

T424 Neigung Fläche 70.5288°, Neigung Kante 54.7356°,

T425
```
P = [0 8 2 0; 0 0 2 0];plot(P(1,:),P(2,:))
s=sqrt(2)/2 ; hold on; M=[s -s; s s]; plot(P(1,:),P(2,:))
for k=1:7; Q=M^k*P; plot(Q(1,:),Q(2,:))
```

15
Lösungshinweise zum Kapitel 5

15.1
Lösungen zu den allgemeinen Übungen im Kapitel 5

Potenzreihen
50–1 Potenzreihen von Exponentialfunktionen

Bestimmen Sie die Potenzreihen von e^x, e^{2x} und e^{x^2}.

```
% Symbolic Skript divetaylor.m
% verschiedene Potenzreihen von e-funktionen
syms x
taylor(exp(x),'order',10)
%  x^9/362880 + x^8/40320 + x^7/5040 + x^6/720 + x^5/120 ...
%      + x^4/24 + x^3/6 + x^2/2 + x + 1
taylor(exp(2*x),'order',10)
% (4*x^9)/2835 + (2*x^8)/315 + (8*x^7)/315 + (4*x^6)/45 +...
%  (4*x^5)/15 + (2*x^4)/3 + (4*x^3)/3 + 2*x^2 + 2*x + 1
 taylor(exp(x^2),'order',10)
%  x^8/24 + x^6/6 + x^4/2 + x^2 + 1
```

50–2 Exponentialfunktionen mit verschiedenen Basen

```
% Symbolic Skript divexptaylor.m
% verschiedene Potenzreihen von Exponentialfunktionen
syms x a
taylor(exp(x),'order',8)
%  x^7/5040 + x^6/720 + x^5/120 + x^4/24 + x^3/6 + x^2/2 + x + 1
taylor(10^x,'order',8)
% (log(10)^7*x^7)/5040 + (log(10)^6*x^6)/720 + ...
%  (log(10)^5*x^5)/120 + (log(10)^4*x^4)/24 + ...
%  (log(10)^3*x^3)/6 + (log(10)^2*x^2)/2 + log(10)*x + 1
taylor(2^x,'order',8)
%  (log(2)^7*x^7)/5040 + (log(2)^6*x^6)/720 + ...
%  (log(2)^5*x^5)/120 + (log(2)^4*x^4)/24 + (log(2)^3*x^3)/6 +...
%  (log(2)^2*x^2)/2 + log(2)*x + 1
taylor(a^x,'order',8)
```

```
% (x^2*log(a)^2)/2 + (x^3*log(a)^3)/6 + (x^4*log(a)^4)/24 +...
%    (x^5*log(a)^5)/120 + (x^6*log(a)^6)/720 + ...
%    (x^7*log(a)^7)/5040 + x*log(a) + 1
```

50–3 Verschiedene Entwicklungspunkte

```
% Symbolic Skript mclaurincomp.m
% verschiedene Entwicklungspunkte von Exponentialfunktionen
syms x a
tl = taylor(exp(x),x,'order',6)
% x^5/120 + x^4/24 + x^3/6 + x^2/2 + x + 1
subs(tl,x,1.5)
% x^5/120 + x^4/24 + x^3/6 + x^2/2 + x + 1
%
ml = taylor(exp(x),x,1,'order',6)
% exp(1) + exp(1)*(x - 1) + (exp(1)*(x - 1)^2)/2 + ...
%    (exp(1)*(x - 1)^3)/6 + (exp(1)*(x - 1)^4)/24 + ...
%    (exp(1)*(x - 1)^5)/120
subs(ml,x,1.5)
%  (6331*exp(1))/3840
```

50–4 Approximation von ’e’

```
% Symbolic Skript ewertbest.m
% Annaeherung von e
syms x
tl = taylor(exp(x),x,'order',9)
%  x^8/40320 + x^7/5040 + x^6/720 + x^5/120 + ...
%     x^4/24 + x^3/6 + x^2/2 + x + 1
subs(tl,x,1)
%  109601/40320   % Naeherungswert fuer e
tlh = taylor(exp(x),x,'order',15)
subs(tlh,x,1)
% 47395032961/17435658240 % besserer Naeherungswert
```

50–5 Elementweise Integration

```
% Symbolic Skript cosint.m
syms x
int(taylor(cos(x),x,'order',12),x)
% - x^11/39916800 + x^9/362880 - x^7/5040 + x^5/120 - x^3/6 + x
taylor(sin(x),'order',12)
% - x^11/39916800 + x^9/362880 - x^7/5040 + x^5/120 - x^3/6 + x
```

50–6 Trigonometrische Ableitungen

```
% Skript symtrigabl.m
syms x
sr = taylor(sin(x))  % ev noch 'order',10 einsetzen
srd = diff(sr,x)
cr = taylor(cos(x))
crd = diff(cr,x)
sr5 = taylor(sin(5*x))
sr5d = diff(sr5,x)
cr5 = taylor(cos(5*x))
cr5d = diff(cr5,x)
```

50–7 Konvergenzradius

Bestimmen Sie den Konvergenzradius der Potenzreihe $\sum_{n\to\infty}(x-2)^n/2^n$

```
%  konvergu7%  Quotientenkriterium: Grenzwert bestimen
syms x n
qu = ((x-2)^(n+1)/(x-2)^n * 2^n/2^(n+1))
limit(qu,n,inf)
limfct = simplify(ans)
%  limfct = x/2-1 % also Konvergenzbereich 0 < x < 4
```

50–8 Konvergenzradius einer Reihe mit fehlenden Potenzen

Bestimmen Sie den Konvergenzradius der Potenzreihe $\sum_{n\to\infty}(x^{3n}/3^n)$

```
% konvergu8
%  Quotientenkriterium: Grenzwert bestimen
syms x n
qu = (x^(3*n+3))/(x^(3*n)) * 3^n/3^(n+1)
limit(qu,n,inf)
limfct = simplify(ans)
%limfct =
%  x^3/3   also Konvergenz bei |x| < 3^(1/3)
```

50–9 Geometrische Reihe

In der Form $1+q+q^2+q^3\ldots$ mit einem festen Wert von q als geometrische Reihe bekannt, ist dieselbe Reihe in der Form $1+x+x^2+x^3\ldots$ eine besonders einfache Potenzreihe. Der Konvergenzbedingung $q<1$ entspricht der Konvergenzradius $R=1$. Erzeugen Sie diese Reihe durch Anwenden der Taylor- bzw. MacLaurin-Formel auf die Funktion $1/(1-x)$.

```
% taylorgeom
syms x
taylor(1/(1-x), 'order',10')
```

50–10 Elementare Binomische Reihe

Für ganzzahlige Exponenten n ist die binomische Form $(1+x)^n$ ein endliches Polynom. Für allgemeine, reelle ν definiert $(1+x)^\nu$ eine unendliche Potenzreihe $1+\binom{\nu}{1}x+\binom{\nu}{2}x^2\ldots$ Deren Koeffizienten $\binom{\nu}{k}$ sind die verallgemeinerten Binomialkoeffizienten $((\nu)(\nu-1)(\nu-2)\ldots(\nu-k+1))\ /\ (k!)$.
Bestimmen Sie nach diesem Muster die ersten 4 Terme von $\sqrt{1+x}=(1+x)^{1/2}$.

```
%  Skript binomwurzel
syms x wap nu
nu = 1/2;
bi1 = nu, bi2 = nu*(nu-1)/2, bi3 = nu*(nu-1)*(nu-2)/6
bi4 = nu*(nu-1)*(nu-2)*(nu-3)/24
wap = 1 + bi1*x + bi2*x^2 + bi3*x^3 + bi4*x^4
% Zum Vergleich
taylor((1+x)^(1/2))
```

50–11 Approximative Wurzelberechnung

Benutzen Sie die ersten vier Terme der binomische Reihe zur approximativen Berechnung von $\sqrt[3]{9}$, definiert als $2\cdot(1+1/8)^{1/3}=2\cdot(1+\binom{1/3}{1}1/8+\binom{1/3}{2}1/8^2+\ldots)$.

```
% Skript taylorwurzel
syms x
tr = taylor((1+x)^(1/3))
wwrt = 2*subs(tr,x,1/8)
%  Vergleich
double(wwrt)
9^(1/3)
```

Orthogonalpolynome

50–12 Simple Orthogonalpolynome

Konstruieren Sie die ersten vier Polynome, welche ohne Gewichtsfunktion über dem Intervall $[-1\ \ 1]$ orthogonal sind: $S_0=1$, $S_1=x$, $S_2=ax^2+bx+c$. Aus `int(S2*S0,x,-1,1)==0` und `int(S2*S1,x,-1,1)==0` lassen sich a,b,c bestimmen z.B. mit der Festlegung $c=-1$. Das analoge Verfahren ergibt die Koeffizienten von S_3.

```
% simporthpoly.m  Aufbau einfache orthogonale Polynome
syms x a b c d
p0 = 1 ; p1 = x ;
% Ansatz p2, allgemein
p2 = a*x^2 + b*x + c   % p2 allgemein 2.Grades
eq1 = int(p0*p2,x,-1,1)==0 % t2 orth t0
% eq1 = (2*a)/3 + 2*c == 0 ; d.h. a = -3*c
eq2 = int(p1*p2,x,-1,1)==0
%  eq2 = (2*b)/3 == 0% p2 orth p1 => b=0
p2 = 3*x^2 - 1; % c = -1 gewählt, dann a= 3
```

```
% p2 festgelegt auf p2 = 3*x^2 - 1, p3 Ansatz allgemein
p3 = a*x^3 + b*x^2 + c*x + d % p3 allgemein 3.Grades
eq3 = int(p2*p3,x,-1,1)==0
eq4 = int(p0*p3,x,-1,1)==0
eq5 = int(p1*p3,x,-1,1)==0
% eq3 = (8*b)/15 == 0   %  % p3 orth p2 => b=0
% eq4 = (2*b)/3 + 2*d == 0 % p3 orth p0; b+3d=0 => d=0
% eq5 = (2*a)/5 + (2*c)/3 == 0 % p3 orth p1; 3a+5c = 0
p3f = 5*x^3 - 3*x
%  p3 festgelegt auf p3 = 5x^3 - 3x , Kontrolle
teq3 = int(p2*p3f,x,-1,1)
teq4 = int(p0*p3f,x,-1,1)
teq5 = int(p1*p3f,x,-1,1)
```

50–13 Legendre Polynome

Mit der Funktion `legendreP(n,x)` im symbolischen Modus erhalten Sie das Legendre-Polynom vom Grad n. Die Legendre-Polynome sind ohne Gewichtsfunktion (bzw. mit $w = 1$) auf dem Intervall $[-1\ \ 1]$ orthogonal. Testen Sie dies für einige Kombinationen n_1, n_2 mit `int(Pa*Pb,x,-1,1)`. Vergleichen sie P_2 und P_3 mit den Resultaten von 50–12.

```
% Skript legendcomp
% Vergleich Legendre Polynome
% und Orthogonalitaetstest
syms x
p1 = legendreP(1,x)
p2 = legendreP(2,x)
p3 = legendreP(3,x)
p4 = legendreP(4,x)
to34 = int(p4*p3,x,-1,1)
to24 = int(p2*p4,x,-1,1)
to14 = int(p1*p4,x,-1,1)
to23 = int(p2*p3,x,-1,1)
to13 = int(p1*p3,x,-1,1)
```

50–14 Fit an Legendre-Polynome

Mit dem Fit-Ansatz `akoef = Y\f` erhalten Sie einen Fit der in der Design-Matrix `Y` eingespeicherten Teilfunktionen an die als Spaltenvektor vorgegebene Funktion `f`. Versuchen Sie dies mit den ersten fünf geraden Legendre-Polynomen und der Kosinus-Funktion im Intervall $[-1\ \ 1]$.

```
% legendrecos  Fit von cos(-1.. 1) an
%   gerade Legendre Polynome
x = -1:0.05:1; Y = 0*x+1;
for deg = 2:2:8
    Y = [Y ; legendreP(deg,x)];
```

```
end
  coslegcoef = (Y')\cos(x)'
```

Fourier-Analyse, Fourier-Transformation
Fourier-Analyse, Fourier-Transformation

50–15 Orthogonalität von Sinus und Kosinus
Das Skalarprodukt von zwei Folgen entspricht dem Entegral über das Produkt der vorliegenden Funktionen.

```
function sicovec = sico(n,cc,cs,k)
% sicovec = sico(npt,coscoef,sincoef,kharm)
t = (0:n-1)*2*pi/n;
sicovec = cc*cos(k*t)+cs*sin(k*t);
```

50–16 bis 50–20 siehe Beispiel-M-Files.

Das Prinzip des FFT-Algorithmus
50–21 FFT-Prinzip siehe Beispiel-M-Files.

50–22 FFT kleiner Dimension
$c0 = f0 + f1, \quad c1 = f0 - f1$
4dim: $c0 = f0 + f1 + f2 + f3 = g0 + g1; \quad g0 = f0 + f2; \quad g1 = f1 + f3;$
$c2 = f0 - f1 + f2 - f3 = g0 - g1;$
$c1 = f0 + j * f1 - f2 - j * f3 = u0 + u1; \quad u0 = f0 - f2; \quad u1 = j * (f1 - f3)$
$c3 = f0 - j * f1 - f1 + j * f3 = u0 - u1$
8dim: $g0 = f0 + f4; \quad g1 = f1 + f5; \quad g2 = f2 + f6; \quad g3 = f3 + f7;$
damit 1. 4dim aus g: $c0 = g0 + g1 + g2 + g3 \quad c4 = g0 - g1 + g2 - g3$
$c2 = g0 + j * g1 - g2 - j * g3; \quad c6 = g0 - j * g1 - g1 + j * g3$
$w = exp(j \cdot \pi/4); \quad u0 = f0 - f4\ ; \quad u1 = w * (f1 - f5);$
$u2 = w^2 \cdot (f2 - f6)\ ; \quad u3 = w^3 \cdot (f3 - f7)\ ;$
damit 2. 4dim aus u $c1 = u0 + u1 + u2 + u3; \quad c3 = u0 - u1 + u2 - u3$
$c5 = u0 + j * u1 - u2 - j * u3\ ; \quad c7 = u0 - j * u1 - u2 + j * u3$

Gewöhnliche Faltungen
50–23 Einführungsbeispiele von Faltungen
a = [1 2 1 2 1] ; b = [0 0 1 3 1 0 0]
conv(a,a)=[1 4 6 8 11 8 6 4 1]; conv(b,b) = [0 0 0 0 1 6 11 6 1 0 0 0 0];
conv(a,b) = conv(b,a) = [0 0 1 5 8 7 8 5 1 0 0]; wie z. B.

```
0  0
      1  2  1  2  1
         3  6  3  6  3
            1  2  1  2  1  0  0
0  0  1  5  8  7  8  5  1  0  0
```

50–24 Sukzessive Abrundung bei mehrmaliger Faltung

a= [0 0 0 1 1 1 1 0 0 0]; a2=conv(a,a) ; a3=conv(a2,a) ; a4=conv(a3,a) ;
a4=conv(a3,a) ; a5=conv(a4,a) ; a6=conv(a5,a) ;
an = a/max(a); a2n = a2/max(a2); a3n = a3/max(a3); a4n = a4/max(a4);
a5n = a5/max(a5); a6n = a6/max(a6); plot(an) ; hold on ; plot(a2n) ; plot(a3n) ;
plot(a4n) ; plot(a5n) ;
plot(a6n) ; axis([0 50 0 1.5])

50–25 Darstellung der sukzessiven Addition bei der Faltung

```
function shvbk = shrgt(v,nsh)
nel = length(v);
shvbk = [v(nel-nsh+1:nel) v(1:nel-nsh)]
end %function
```

Angewandt auf die Faltung

```
g = [1 2 3 2 1 0 0 0 0 0 0 0 0 0]; w = [0 0 1 0 0 0 2 0 0 1 2 1 0 0 ]
conv(w,g) =[     0     0     1     2     3     2     3     4     6     5     6
     8    10     8     4     1     0     0     0     0     0     0
     0     0     0     0     0]
cres = 1*g + 2*shrgt(g,4) +  1*shrgt(g,7) +  2*shrgt(g,8)+
1*shrgt(g,9) ; plot(cres);
cres = 1*g  ; plot(cres); hold on;
 cres = 1*g + 2*shrgt(g,4); plot(cres);
cres = 1*g + 2*shrgt(g,4) +  1*shrgt(g,7)  ; plot(cres);
cres = 1*g + 2*shrgt(g,4) +  1*shrgt(g,7) +  2*shrgt(g,8) ;
plot(cres);
```

50–26 Faltung von Standardfunktionen

```
r = [zeros(1,40) ones(1,40)];
h = [(1:40) (39:-1:0)]/40;
p = 1-(-1:0.025:0.975).^2 ;
g = (1-(-1:0.025:0.975).^2).^2 ;
rh = conv(r,h); plot(rh); pause;
rp = conv(r,p); plot(rp); pause;
rg = conv(r,g); plot(rg); pause;
hp = conv(h,p); plot(hp); pause;
hg = conv(h,g); plot(hg); pause;
pg = conv(p,g); plot(pg);
```

50–27 Faltung einer Sinusfunktion mit sich selbst

```
function sinfsin = sinfaltsin(npt)
% function sinfsin = sinfaltsin(npt)
```

```
% Faltung einer Sinusfunktion mit sich selbst
%  mit anschliessender Grafik
sf = sin(0:(2*pi/npt):2*pi);
sfz = [zeros(1,npt-1) sf];
sff = real(ifft(fft(sfz).*fft(sfz)));
sinfsin = sff;
plot(sf); hold on ; plot(sff); hold off
end % function
```

50–28 Selbst programmierte Faltung

```
function vbk = vecinsert(lg,v,nsh)
nel = length(v);
vbk = zeros(1,lg);
for k=1+nsh:nsh+lg
 vbk = v(k-nsh)
%------------
function  sfres = selffalt(a,b)
la = length(a)
lb = length(b)
sfres = zeros(1, la+lb-1)
for k=1:lb
  sfres = sfres + b(k)* vecinsert(la+lb-1, a, k-1)
end
```

50–29 Nach der Formel programmierte Faltung

```
function c = formfalt(a,b)
% c = formfalt(a,b) gewoehnliche Faltung nach Formel
n1 = length(a) ; n2 = length(b); nr = n1+n2-1;
for j=1:nr
  k1 = max(1,j-n2+1); k2 = min(n1,j);
  c(j) = 0;
  for k=k1:k2
    c(j)= c(j)+a(k)*b(j-k+1);
  end
end
```

Zirkuläre Faltungen

50–30 Zirkuläre Faltung mit einfacher Schleife

Mit der zirkuläeren shift-Funktion shrgt

```
function vecret = shrgt(vecin, nshift)
 ntot = length(vecin);
```

```
 nlast = ntot - nshift ;
 vecret = [ vecin(nlast+1:ntot) , vecin(1:nlast)];
end % function
```

kann eine einzelne Schleife die Zirkuläre Faltung berechnen:

```
function [afb,afbm] = simpcircfalt(a,b)
  % einfache zirkuläre Faltung mit Anzeige der
  % zu summierendem Zeilen
  % benoetigt shrgt(a,n)
  afbm(1,:) = a(1)*b; n= length(a);
  for k=1:length(a)-1
      afbm(k+1,:) = a(k+1)*shrgt(b,k);
  end
  afb = sum(afbm,1);
end % function
```

50–31 Zirkuläre Faltung mit doppelter Schleife

```
function c = formfaltz(a,b)
% c = formfaltz(a,b) zirkulaere Faltung nach Formel
n1 = length(a) ; n2 = length(b);  al = a; bl = b;
if n1 < n2
  al = [al zeros(1,n2-n1)];   n1 = n2;
elseif n1 > n2
  bl = [bl zeros(1,n1-n2)];
end
for j=1:n1
  c(j) = 0;
  for k=1:j
    c(j)= c(j)+b(k)*a(j-k+1);
  end
  for k=j+1:n1
    c(j)= c(j)+b(k)*a(j-k+n1+1);
  end
end
```

50–32 Zirkuläre Faltung einer Sinus-Funktion mit sich selbst:

```
t = 0:pi/40:2*pi-pi/40;  si = sin(t);
si2 = real(ifft(fft(si).*fft(si))); plot(si2); hold on;
plot(-40*cos(t)+1,'--r'); hold off
```

50–33 Zirkuläre Faltung mit Faltungssatz

```
a = [1 2 3 4 5 6]; b = [1 0 0 1 0 0]; c = [5 7 9 5 7 9];
cf = real(ifft(fft(a).*fft(b)))
```

50–34 Funktions-M-File für zirkuläre Faltung mit FFT

```
function c = fftfaltz(a,b)
% c = fftfaltz(a,b) zirkulaere Faltung nach Faltungssatz
n1 = length(a) ; n2 = length(b);  al = a; bl = b;
if n1 < n2  % Laengen-Test mit Nullen nachfuellen
  al = [al zeros(1,n2-n1)];   n1 = n2;
elseif n1 > n2
  bl = [bl zeros(1,n1-n2)];
end
c = real(ifft(fft(al).*fft(bl)));
```

50–35 Normale Faltung auf zirkuläre Faltung zurückführen

```
function c = fftfaltgew(a,b)
% c = fftfaltgew(a,b) gewoehnliche Faltung
%  nach Faltungssatz
n1 = length(a) ; n2 = length(b);
al = a; bl = b; nr = n1+n2-1;
% Nullen auffuellen bis zu Resultatlaenge
%    der gewoehnlichen Faltung
  al = [al zeros(1,nr-n1)];
  bl = [bl zeros(1,nr-n2)];
c = real(ifft(fft(al).*fft(bl)));
```

50–36 Vergleich von zirkulärer und gewöhnlicher Faltung

Cz = [22 20 18 16 16 18 20 22 23]
Cg = [0 1 2 6 10 15 19 22 23 22 19 15 10 6 3 1 0]

15.2
Lösungen zu den Selbsttests

Lösungen zur Testserie 5.1

T511

– Die bekannte Bedingung $q < 1$, also r= 1

– Durch verschiedene Phasenlagen

– $r < 1 \;\; - -i \cdot z_5$

T512 Mit der Hilfsfunktion für die Koeffizienten der e-Entwicklung erhält man

```
function ec = ecoef(x,n)
  ec = x.^n./factorial(n);
```

für $x = 1$ $c7 = 1.9841e - 04$

für $x = 2$ $c10 = 2.8219e - 04$

für $x = 10$ $c32 = 3.8004e - 04$

T513

$c_0 = 1 \;\; c_1 = \sqrt{2}/2 \;\; c_2 = 0 \;\; c_3 = 1/\sqrt{32}$

$\phi_0 = 0 \;\; \phi_1 = \pi/2 \;\; \phi_2 = 0 \;\; \phi_3 = 3\pi/4$

$p_0 = 1 \;\; p_1 = 1/2 \;\; p_2 = 0 \;\; p_3 = 1/32$

T514

Bestimmen Sie die ersten 6 Tschebyscheff-Polynome aus der Start-Angabe $T_0 = 1$, $T_1 = x$ und der Rekursion $T_{k+1} = 2x \cdot T_k - 3 \cdot T_{k-1}$

```
% Skript tschebyserie
syms x
t0 = 1, t1 = x
t2 = simplify(collect(2*x*t1 - t0))
% t2 = 2*x^2 - 1;
t3 = simplify(collect(2*x*t2 - t1))
% t3 = 4*x^3 - 3*x
t4 = simplify(collect(2*x*t3 - t2))
%  t4 = 8*x^4 - 8*x^2 + 1
t5 = expand(2*x*t4 - t3)
%  t5 = 16*x^5 - 20*x^3 + 5*x
t6 = simplify(collect(2*x*t5 - t4))
%  t6 = 32*x^6 - 48*x^4 + 18*x^2 - 1
```

T515

Finden Sie die gewöhnlichen Fourier-Koeffizienten ahnand der Angabe, dass das Power-Spektrum $p_0 \ldots p_9$ die Werte $p_1 = 4$, $p_4 = 1/4$ und sonst Null aufweist, und dass die Funktion gerade ist.

Gerade Funktion heisst: Nur cos-Koeffizienten - alle $b_k = 0$

$a_0 = 0$, $a_1 = 2$, $a_4 = 1/2$

Lösungen zur Testserie 5.2

T521

– Damit die Formel
$a_k = \frac{2}{T} \cdot \int_0^T f(t) \cdot \cos\left(k \cdot \frac{2\pi}{T}\, t\right)\, dt$
für $k = 0$ keine Ausnahme machen muss sondern gleich lautet.

–
Schreiben Sie die Formel$\sin^2(x) + \cos^2(x)$ mit den Taylor-Termen bis und mitx^2.
$sin^2(x) = -x^4/3 + x^2$, $cos^2(x) = x^4/3 - x^2 + 1$, zusammen$sin^2(x) + cos^2(x) = 1$.

–
Wieviele Koeffizienten kann eine Fourier-Darstellung maximal enthalten, wenn diese bis zur 10. Harmonischen geht?
Maximal 21, 1 Gleichstrom-Anteil und je 10 cos- und sin-Koeffizienten.

–
Wie kann man einen einzelnen komplexen Koeffizienten z. Bc_k in einer Fourierreihe ersetzen, wenn man sonst alle komplexen Koeffizienten kennt?
$c_k = \mathrm{Re}(c_{-k} - \mathrm{Im}(c_{-k})$

T522

Führen Sie die einfache Faltung der Folgen`a=[1 2 3 4]` und `b = [1 0 1 1 0 1]` mit Bleistift und Papier durch, sowohl als direkte einfache Faltung, als auch als zirkuläre Faltung bei welcher die notwendige Anzahl Nullen angehängt wurde. Vergleichen Sie die Resultate.

```
conv(a,b) = conv(b,a) = [1 2 4 7 5 8 6 3 4]
ay = [a 0 0 0 0 0]     by = [b 0 0 0]
 ifft(fft(ay).*fft(by))
% =  1.0 2.0 4.0 7.0 5.0 8.0 6.0 3.0 4.0
```

T523

$c_0 = 1 \;\; c_1 = \sqrt{2}/2 \;\; c_2 = 0 \;\; c_3 = 1/\sqrt{32}$
$\phi_0 = 0 \;\; \phi_1 = \pi/2 \;\; \phi_2 = 0 \;\; \phi_3 = 3\pi/4$
$p_0 = 1 \;\; p_1 = 1/2 \;\; p_2 = 0 \;\; p_3 = 1/32$

T524

Überlegen Sie sich, vorerst ohne nachzuschauen, wie die komplexe Matrix zu einer DFT von 4 Punkten, k=0 bis 3 aussieht.

```
DFT4 = [1 1 1 1; 1 j -1 -j; 1 -1 1 -1; 1 -j -1 j]
```

T525

Schreiben Sie ein MATLAB-Skript, das durch Punkt-Multiplikation und Summation mit Funktionen der Art$\sin\ \cos(n \cdot k \cdot 2\pi/8)\;\; k = 0 \ldots 3,\;\; n = 0 \ldots 7$ zu einer tabellarisch gegebenen Funktion von 8 Werten die DFT bestimmt.

```
w=(0:7)/8*2*pi; a=zeros(1,8); b = zeros(1,8);
for k=0:3
 a(k+1) = 2/8*sum(cos(k*w).*f); b(k+1) = 2/8*sum(sin(k*w).*f);
end
```

16
Lösungshinweise zum Kapitel 6

16.1
Lösungen zu den allgemeinen Übungen des Kapitels 6

Funktionsdarstellung
60–1 Konturplot einer Halbkugel

```
% halbkugelkontur.m
z=zeros(41);
for zei = 1:41
  for spa = 1:41
    y(zei) = (zei-21)*0.05;
    x(spa) = (spa-21)*0.05;
    if x(spa)^2+y(zei)^2 < 1
      z(zei,spa)= sqrt(1-x(spa)^2-y(zei)^2);
    end
  end
end
contour(x,y,z,25) ; axis([-1 1 -1 1]); axis square; pause
contour3(x,y,z) ; axis equal; pause
surf(x,y,z) ; axis equal;
```

60–2 Konturplot

```
% sincoskontur.m
z=zeros(41);
for zei = 1:41
  for spa = 1:41
    y(zei) = (zei-1)*0.05*pi;
    x(spa) = (spa-1)*0.05*pi;
    z(zei,spa)= sin(x(spa))*cos(y(zei));
  end
end
```

```
contour(x,y,z,25) ; axis([0 6.3 0 6.3]);
axis square; hold on
xstat = [0.5 1.5 0.5 1.5 0.5 1.5 0 1 2 0 1 2]*pi;
ystat = [ 0   0    1   1  2   2  ...
    0.5 0.5 0.5 1.5 1.5 1.5]*pi;
plot(xstat,ystat,'or'); hold off;   pause
contour3(x,y,z,20) ; axis equal; pause
surf(x,y,z) ; axis equal;
```

60–3 Grafische Darstellung - „Jurahügel“

```
% jurakontur.m
z=zeros(41);
for zei = 1:41
  for spa = 1:81
    y(zei) = (zei-21)*40;
    x(spa) = (spa-41)*80;
    z(zei,spa)=max(1100,1600-x(spa)^2/20000-y(zei)^2/1250);
  end
end
contour(x,y,z,15) ; axis equal;
  pause
contour3(x,y,z,20) ; axis equal; pause
surf(x,y,z) ; axis equal;
```

60–4 Konturplots einfacher Matrizen:

```
% specmatkontur.m
z=indmatf(9); contour(z); pause
Du = zeros(21);
for zei = 1:21
  for spa = zei:41
    Du(zei,spa)= spa-zei;
  end
end
Ph = [ Du fliplr(Du) ]; P = [ flipud(Ph) ; Ph ];surf(P)
axis equal ; view(-20,62)
```

60–5 Konturplots von verschiedenen Matrizenprodukten

```
% dumatkontur.m
surf(Du); pause; surf(Du^2); pause;
surf(Du.^2); pause
```

```
C = fliplr(Du); surf(C.^3); pause;
surf(C^3); pause; surf(C*Du)
```

60–6 Trichterfunktion

```
% trichterkontur.m
z=zeros(85);
for zei = 1:85
  for spa = 1:85
    y(zei) = (zei-43)*0.5;
    x(spa) = (spa-43)*0.5;
    if x(spa)^2+y(zei)^2 <= 21^2
      z(zei,spa)= sqrt(x(spa)^2+y(zei)^2)-0.5;
    end
  end
end
contour(x,y,z,25) ; axis([-22 22 -22 22]);
axis square; pause
contour3(x,y,z,20) ; axis equal; pause
surf(x,y,z) ; axis equal;
```

60–7 Kissenfunktion

```
% kissenkontur.m
z=zeros(41,61);
for zei = 1:41
  for spa = 1:71
    y(zei) = (zei-21)*0.05;
    x(spa) = (spa-36)*0.05;
      z(zei,spa)= 0.4* (1-(x(spa)*4/7)^8)*(1-y(zei)^4);
  end
end
contour(x,y,z,15) ;  axis equal; pause
contour3(x,y,z,20) ; axis equal; pause
surf(x,y,z) ; axis equal ; colormap(winter)
```

60–8 Futtersilo mit Bodenlagerung

```
% silo.m H\"ohenlinien, Volumen und Abspannseile
%   fuer Futtersilo mit Bodenlagerung
h=zeros(61);  xpl=zeros(1,61);  ypl=zeros(1,61);
[xg,yg]= meshgrid(0:0.1:6,0:0.1:6 );
for k= 1:61
```

```
  x = (k-1)*0.1;   xpl(k) = x;
  for j=1:21
    y = (j-1)*0.1;  ypl(j) = y;
    h(k,j) = 0.2*x*(6-x) * y*(2-y);
  end
end
% Volumen in Liter = 10 mal (h in m)
%  Summe aller Hoehen je 10x10cm
V=10*sum(sum(h))
contour3(xg,yg,h,30)
hold on
hmx = max(max(h))
for k=0:4
  xs1 =k+ (0:0.1:2) ; xs1(1);   ys1 = 0:0.1:2;
  zs1 =  0.2*xs1.*(6-xs1) .* ys1.*(2-ys1);
  plot3(ys1,xs1,zs1,'k')
end
for k=2:6
  xs1 =k+ (0:-0.1:-2) ;  xs1(1);   ys1 = 0:0.1:2;
  zs1 =  0.2*xs1.*(6-xs1) .* ys1.*(2-ys1);
  plot3(ys1,xs1,zs1,'k')
end
hold off;  vx=sum(h');  vpart=zeros(1,61);
for k=1:61
  vpart(k) =10*sum(vx(1:k));
end
vpart
figure(2); plot(xpl,vpart) ; figure(1)
```

Partielle Ableitungen

60–9 Analytische Bestimmung von partiellen Ableitungen

a) $\partial f/\partial x = 2xe^y \sin(z)$, $\partial f/\partial y = x^2 e^y \sin(z)$, $\partial f/\partial z = x^2 e^y \cos(z)$
b)]$\partial f/\partial x = e^{xy} \cdot y - e^{-xy} \cdot y$, $\partial f/\partial y = e^{xy} \cdot x - e^{-xy} \cdot x$
c) $\partial f/\partial x = 2\sin(x) \cdot \cos(x) \cdot \cos^3(z)$ $\partial f/\partial z = sin^2(x) \cdot (-3\cos^2(z) \cdot \sin(z))$
d)] $\partial f/\partial x = 3x^2 y^2 z \frac{1}{u}$, $\partial f/\partial y = x^3 2yz \frac{1}{u}$ $\partial f/\partial z = x^3 y^2 \frac{1}{u}$ $\partial f/\partial u = -x^3 y^2 z \frac{1}{u^2}$

60–10 Bestimmen von partiellen Ableitungen:

a) $\partial F/\partial x = \frac{x}{\sqrt{x^2+y^2} \cdot z}$ $\partial F/\partial y = \frac{y}{\sqrt{x^2+y^2} \cdot z}$ $\partial F/\partial z = -\frac{\sqrt{x^2+y^2}}{z^2}$
b)$\partial F/\partial x = \cos(x * \cos(4y)) \cdot \cos(4y)$ $\partial F/\partial y = -4\cos(x * \cos(4y)) \cdot x * \sin(4y)$
c) $\partial F/\partial x = z^2/\sqrt{xz + y/z}\,/2 \cdot z$ $\partial F/\partial y = z^2/\sqrt{xz + y/z}\,/2 \cdot (1/z)$ $\partial F/\partial z = 2z\sqrt{xz + y/z} + z^2/\sqrt{xz + y/z}\,/2 \cdot (x - y/z^2)$

60–11 Gradient: alle partiellen Ableitungen in einem Vektor zusammengefasst

```
syms x y z
f1 = x^2+y^2 ; pdx1 = diff(f1,x); pdy1 = diff(f1,y)
%pdx1,pdy1  =   2*x  , 2*y
f2 = sin(x)*cos(y); pdx2 = diff(f2,x);
pdy2 = diff(f2,y)
%pdx2,pdy2  =   cos(x)*cos(y);  -sin(x)*sin(y);
f3 = 1/sqrt(x^2+y^2+z^2); pdx3 = diff(f3,x);
pdy3 = diff(f3,y); pdz3 = diff(f3,y)y3 = diff(f3,y)
%pdx3, pdy3 =  -x/(sqrt(x^2+y^2+z^2))^3,
%     -y/(sqrt(x^2+y^2+z^2))^3
%   pdz3 =  -z/(sqrt(x^2+y^2+z^2))^3
```

60–12 Gradientenplot

```
%sincontgrad.m Konturplot mit eingezeichnetem Gradient
[X,Y] = meshgrid([0:.1:1]);
Z = sin(pi*X).*sin(pi*Y);
contour3(X,Y,Z,30)
figure(2); clf
[px,py] = gradient(Z,.2,.2);
contour(X,Y,Z)  ;  hold on
quiver(X,Y,px,py)             % malt Pfeile
axis equal ; hold off
```

60–13 Partielle Ableitungen und Gradientenfunktion:

```
%jhillcontgrad.m Konturplot mit eingezeichnetem Gradient
[X,Y] = meshgrid([-1600:50:1600],[-400:50:400]);
Z = max(1600 -(X/1000).^2*50 -(Y/250).^2*50,1400) ;
contour3(X,Y,Z,20); axis equal
figure(2); clf
[px,py] = gradient(Z,50,50);   %Abst\"ande zwischen Punkten
subplot(3,1,1)
contour(X,Y,Z,15)  ;  hold on
% Ausduennen der gezeichneten Pfeile
xsel = 1:4:65; ysel=1:2:17;
Xs = X(ysel,xsel);Ys = Y(ysel,xsel);
 pxs = px(ysel,xsel) ; pys = py(ysel,xsel);
quiver(Xs,Ys,pxs,pys)                % erstellt Pfeile
axis equal ;
```

```
subplot(3,1,2); contour(X,Y,px,15)
subplot(3,1,3); contour(X,Y,py,15)
 hold off
```

60–14 Konturplots und Gradientenvektoren

Die folgenden Funktions-M-Files können im allgemeinen Programm `divfctcont()` eingesetzt werden.

```
%z=minihillf(Xgr,Ygr) Funktion zur Definition der Flaeche
%    1 - x^2 - 0.2*y^2
function z=minihillf(Xgr,Ygr)
z = 1 - Xgr.^2 - 0.2*Ygr.^2 ;

%z=sinsinf(Xgr,Ygr) Funktion zur Definition der Flaeche
%    sin(pi*x) * sin(pi*y)
function z=sinsinf(Xgr,Ygr)
z = sin(pi*Xgr).*sin(pi*Ygr);

%z=pwdiff(Xgr,Ygr) Funktion zur Definition der Flaeche
%    (x^2 - abs(x^3)) *  (y^2 - abs(y^3))
function z=pwdiff(Xgr,Ygr)
z =(Xgr.^2 - abs(Xgr.^3)) .*  (Ygr.^2 - abs(Ygr.^3))  ;

function plhd = divfctcont(fctnam,xvec,yvec)
% plhd = divfctcont(fctnam,Xgrid,Ygrid)
%    Konturplot von mit 'fctnam' waehlbarer Funktion
[X,Y] = meshgrid(xvec,yvec);
Z = feval(fctnam,X,Y);
figure(1); clf
contour3(X,Y,Z,30)
figure(2); clf
[px,py] = gradient(Z,xvec,yvec);
plhd = contour(X,Y,Z,25)  ;  hold on
quiver(X,Y,px,py)                  %erstellt Pfeile
axis equal ; hold off
```

60–15 Stationäre Punkte

```
% p2p4findstat.m - Symbolic script sucht Stationaere Punkte
syms xs ys
fs = (xs^2-xs^4)*(ys^2-ys^4)
dfx = diff(fs,xs)
dfy = diff(fs,ys)
stptsol = solve(dfx,dfy)
stptsol.xs
```

```
stptsol.ys
%{xs 0} {0 ys}{+-1 +-1} {+-\sqrt(2)/2 +-\sqrt(2)/2}

%p2p4stat.m Konturplot mit stationaeren Punkten
[X,Y] = meshgrid([-1.1:0.1:1.1]);
Z = 10*(X.^2-X.^4).*(Y.^2-Y.^4) ;
figure(1); clf
contour3(X,Y,Z,20);
axis equal
figure(2); clf
contour(X,Y,Z,15)  ;
```

60–16 Biquartische Funktion

Mit der Funktion

```
%z=biquartf(Xgr,Ygr) Funktion zur Definition der Flaeche
%   0.25x^4-0.5x^2 + 0.25y^4-0.5y^2 -0.12x -0.05y + 1
function z=biquartf(Xgr,Ygr)
z=(0.25*Xgr.^4-0.5*Xgr.^2)-0.12*Xgr  ...
      +(0.25*Ygr.^4-0.5*Ygr.^2)+1-0.05*Ygr;
```

ergibt sich der Aufruf ganz einfach als:

```
divfctcont('biquartf',-1.7:0.2:1.7, -1.7:0.2:1.7)
```

60–17 Das elektrische Potential einer Punktladung:

Die Funktion

$\phi(r) = \frac{Q}{4\pi\varepsilon_0} \cdot \frac{1}{\sqrt{x^2+y^2+z^2}}$ hat die drei partiellen Ableitungen

$-x/\sqrt{x^2+y^2+z^2}^3$ bzw. $-y/\ldots$, $-z/\ldots$ damit ist

$E = \text{grad}\phi = -qf \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} \frac{1}{r^3}$

Fitprobleme

60–18 Geraden- Parabel- und kubischer Fit

```
x = 0:4 ; y = [1 1.2 2.2 3 3.6];
M1 = [sum(x.^2) sum(x); sum(x) 5];
b1 =[ sum(x.*y) ; sum(y)];
M2 = [sum(x.^4) sum(x.^3) sum(x.^2); ...
       sum(x.^3) sum(x.^2) sum(x); ...
       sum(x.^2) sum(x) 5];
b2 =[ sum(x.^2.*y) ; sum(x.*y) ; sum(y)];
M3 = [sum(x.^6) sum(x.^5) sum(x.^4) sum(x.^3);...
      sum(x.^5) sum(x.^4) sum(x.^3) sum(x.^2);
```

```
      sum(x.^4) sum(x.^3) sum(x.^2) sum(x);
      sum(x.^3) sum(x.^2) sum(x) 5 ];
b3 =[ sum(x.^3.*y); sum(x.^2.*y) ; sum(x.*y) ; sum(y)];
p1 = M1\b1
p2 = M2\b2
p3 = M3\b3
```

60–19 Kubischer Fit

```
x = [-3 -2 -1 0 1 2 3]'; A=[x.^3 x.^2 x [1 1 1 1 1 1 1]'];
b = [-7 -5 -2 0 -1 2 0]'; p=A\b
```

60–20 Fit mit Gewicht:

```
x = (-2:2)' ; y = [-2 0 1 1.5 3]'; w = [4 2 1 2 4]';
Mn = [sum(x.^2) sum(x); sum(x) 5]; bn =[ sum(x.*y) ; sum(y)];
A = [x ones(5,1)];  pn = Mn\bn ; pf = A\y;
Aw = A.*[w w]; yw = y.*w;  pfw = Aw\yw;
Mw = [sum(w.^2.*x.^2) sum(x.*w.^2); sum(x.*w.^2) sum(w.^2)];
bw =[ sum(w.^2.*x.*y) ; sum(w.^2.*y)]; pnw = Mw\bw;
pn, pf, pnw, pfw
```

60–21 Fehlergleichungen

Zur Demonstration dass die Normalengleichungen einer Multiplikation von links mit A^T entsprechen, sollten die folgenden Fälle alle Nullmatrizen oder Nullvektoren ergeben.

```
A'*A-Mn
A'*y-bn
Aw'*Aw-Mw
Aw'*y-bw
```

60–22 Fit nach Potenzfunktionen

```
x = (1:5)' ; y = [1 3 4 2 1]';
p2 = polyfit(x,y,2) ; p3 = polyfit(x,y,3)
xf = 0:0.05:6; y2 = polyval(p2,xf); y3 = polyval(p3,xf);
plot(x,y,'or'); hold on; plot(xf,y2); plot(xf,y3,'g')
```

60–23 Fit nach frei festlegbaren Funktionen

```
x = (0:0.2:1)'*pi ; y = sin(x);
varfctfit(x,y,'pow0fct','pow1fct','pow2fct')
```

```
% pkoef = varfctfit(x,y,'f1','f2', ...)
%  linearer Fit nach einer variablen Anzahl Funktionen fk
function pkoef = varfctfit(varargin)
A=[];
for nf = 3:nargin
  A=[A feval(varargin{nf},varargin{1})];
       % Auswertung einer mit Namen bekannten Funktion
end
pkoef = A\varargin{2};
```

60–24 Orthogonale Funktionen beim Fitten:
Zuerst die Funktions-Definitionen, dann diverse Fit-Fälle:

```
function s4 = sintpif(x)
  s1 = sin(2*pi*x)
end % function
%  --- trennen, separate Files ----
function s2 = sin2tpif(x)
  s2 = sin(2*2*pi*x)
end % function
%  --- trennen, separate Files ----
function s4 = sin4tpif(x)
  s4 = sin(4*2*pi*x)
end % function
%  --- trennen, separate Files ----

x = (0:0.1:0.9)' ;
y = [0 1 1.2 0  1 0 0 0 0 -1]';
varfctfit(x,y,'sintpif')
varfctfit(x,y,'sintpif','sin2tpif')
varfctfit(x,y,'sintpif','sin2tpif','sin4tpif')
```

Methode der Lagrange-Multiplikatoren

60–25 Lagrange-Multiplikator bei einer „Hügelform“-Funktion

$$L(x,y,\lambda) = 10 - 0.2x^2 - 0.05y^2 + \lambda \cdot (x + y - 5)$$

$$\left| \begin{array}{lllll} \frac{\partial L}{\partial x} & = & -0.4x + \lambda & = & 0 \\ \frac{\partial L}{\partial y} & = & -0.1y + \lambda & = & 0 \\ \frac{\partial L}{\partial \lambda} & = & x + y & = & 5 \end{array} \right| \quad \rightarrow \quad x = 1,\ y = 4,\ \lambda = -.4$$

60–26 Lagrange-Multiplikatoren:

$$L(x,y,z,\lambda) = x^2 + 2y^2 + 4z^2 + \lambda \cdot (x - y - 2z - 8)$$

$$\left|\begin{array}{lllll} \frac{\partial L}{\partial x} & = & 2x+\lambda & = & 0 \\ \frac{\partial L}{\partial y} & = & 4y-\lambda & = & 0 \\ \frac{\partial L}{\partial z} & = & 8z-2\lambda & = & 0 \\ \frac{\partial L}{\partial \lambda} & = & x-y-2z & = & 8 \end{array}\right| \rightarrow \; x=3.2,\; y=-1.6,\; ,z=-1.6\lambda=-6.4$$

60 - 27 Nichtlineare Nebenbedingung und Abstandsfunktion

$$L(x,y,\lambda)=\sqrt{x^2+y^2}+\lambda\cdot(y-5/x^3)$$

$$\left|\begin{array}{lllll} \frac{\partial L}{\partial x} & = & x/\sqrt{x^2+y^2}+\lambda\cdot 15/x^4 & = & 0 \\ \frac{\partial L}{\partial y} & = & y/\sqrt{x^2+y^2}+\lambda & = & 0 \\ \frac{\partial L}{\partial \lambda} & = & y-5/x^3 & = & 0 \end{array}\right| \rightarrow \; x=1,\; y=4,\; \lambda=-.4$$

60–28 Visualisierung der Optimierung mit Nebenbedingung

```
% hillagrange.m Figur zu Optimierung mit Nebenbedingung
xv = -6:0.1:6;
[xg,yg]=meshgrid(xv);
H =  10 - 0.02*xg.^2 - 0.1*yg.^2;
H = H .* (yg > xg-4) ; H=max(H,9);
figure(1) ; clf;  surf(xg,yg,H); view(15.5,11)
figure(2) ; clf
contour3(xg,yg,H)  ; hold on
LM = [-0.04 0 1 ; 0 -0.2 1 ; 1 -1 0]; Lb = [ 0 0 4]';
p= LM\Lb
xel = 0:0.1:6;  yel = -4+xel;
zel =  10 - 0.02*xel.^2 - 0.1*yel.^2;
zel = max(zel,9); plot3(xel,yel,zel,'k')
view(15.5,18.0) ; hold off
```

Singular Value Decomposition und Pseudoinverse
60–29 Anwenden der pseudoinversen Matrix

```
x=(-3:3)'; A=[x.^3 x.^2 x x.^0];
y1 = [-1 1 1 0 0 1 3]'; y2 = [1 2 1 2 3 3 5]';
% Aufruf der Singular Value Decomposition
[U , S , V] = svd(A);  Si = S
% nur S muss elementweise invertiert werden
for k=1:4
   Si(k,k) = 1/S(k,k);
end
Si
```

```
% U und V sind orthogonal: U'*U=I
Api= V*Si'*U';
p1 = A\y1
pp1 = Api*y1
p2 = A\y2
pp2 = Api*y2
```

Nichtlineare Gleichungssysteme

60–30 Jacobi-Matrix:

$J(x_1, x_2, x_3) =$

$$\begin{pmatrix} x_1/\sqrt{x_1^2+x_2^2+x_3^2} & x_2/\sqrt{x_1^2+x_2^2+x_3^2} & x_3/\sqrt{x_1^2+x_2^2+x_3^2} \\ -x_2/(x_1^2(1+x_2^2/x_1^2)) & 1/(x_1(1+x_2^2/x_1^2)) & 0 \\ -x_3x_1/\sqrt{x_1^2+x_2^2}^3/cc & -x_3x_2/\sqrt{x_1^2+x_2^2}^3/cc & 1/\sqrt{x_1^2+x_2^2}/cc \end{pmatrix}$$

$cc = (1 + x_3^2/(x_1^2 + x_2^2))$

60–31 Nichtlineare Gleichungssysteme

```
x= [5 5]';  fct2p2p4iter;   fct2p2p4iter; fct2p2p4iter;
% fct2p2p4iter.m - run Jacobi-Iteration
fvec= fct2p2p4(x(1),x(2))
delx = fct2p2p4jac(x(1),x(2))\ fct2p2p4(x(1),x(2))
x = x-delx
% fvec = fct2p2p4(x,y) Funktion zur jacobi-Iteration
function fvec = fct2p2p4(x,y)
fvec = zeros(2,1);
fvec(1) = (x-4)^2+ y^2 -10;
fvec(2) = x^4+ y^4 -20;
%jac = fct2p2p4jac(x,y) Evaluation der Jacobi-Matrix
function jac = fct2p2p4jac(x,y)
jac = zeros(2);
jac(1,1) = 2*x-8;
jac(1,2) = 2*y;
jac(2,1) = 4*x^3;
jac(2,2) = 4*y^3;
```

60–32 Kugelkoordinaten

Ein Einzelschritt der Newton-Iteration zum Invertieren der Kugelfunktion wird im File kugjacrun.m gezeigt. Zum Iterieren muss dieses Skript mehrfach aufgerufen werden.

```
% kugjacrun.m 3D Newton Iteration
xstart = [5 0.5 0.3]';
xaktuell = xstart
```

```
for iter = 1:8
ferr = kugfct(xaktuell)
delx = -kugjac(xaktuell)\ferr
xaktuell = xaktuell + delx
pause
end
```

Dabei werden die Funktionen kugfct.m und kugjac.m aufgerufen.

```
% fvec=kugfct(xvec) Abweichungen bei
%    Kugelfunktions-Iteration
function fvec = kugfct(xvec)
fvec = zeros(3,1);
fvec(1) = xvec(1)*cos(xvec(2))*cos(xvec(3)) - 2;
fvec(2) = xvec(1)*cos(xvec(2))*sin(xvec(3)) - 7;
fvec(3) = xvec(1)*sin(xvec(2)) - 3;

% Jkug = kugjac(xvec) Jacobi Matrix zu Kugelkoordinaten
function Jkug = kugjac(xvec)
Jkug = zeros(3);
rac = xvec(1); tac = xvec(2); wac = xvec(3);
Jkug(1,1)=     cos(tac)*cos(wac);
Jkug(1,2)= -rac*sin(tac)*cos(wac);
Jkug(1,3)= -rac*cos(tac)*sin(wac);
Jkug(2,1)=     cos(tac)*sin(wac);
Jkug(2,2)= -rac*sin(tac)*sin(wac);
Jkug(2,3)=  rac*cos(tac)*cos(wac);
Jkug(3,1)=     sin(tac);
Jkug(3,2)=  rac*cos(tac);
Jkug(3,3)=     0 ;
```

60–33 Funkfeuerortung

Im File funkfeuer.m zur Lösung der Übung 60 – 33 sind die beiden Funktionen zur Evaluation der Abweichungen und zur Bestimmung der Jacobimatrix als Unterfunktionen definiert. Dazu muss die Hauptfunktion aber selbst eine Funktion sein, ein Skript ist nicht erlaubt.

```
%[x,y]= funkfeuer(d1,d2) 2D Newton Iteration
function [xpos,ypos] = funkfeuer(d1,d2)
xv = [5 5]'; dv = [d1;d2];

for iter =1:8
dx = funkfeuerjac(xv)  \ funkfeuerfct(xv,dv);
xv = xv -dx;
end
xpos = xv(1); ypos = xv(2);
```

```
% subfunction funkfeuerfct
function fmom = funkfeuerfct(xv,dv)
fmom=zeros(2,1);
fmom(1) = sqrt(xv(1)^2+xv(2)^2) - ...
     sqrt(xv(1)^2+(xv(2)-10)^2) - dv(1);
fmom(2) = sqrt(xv(1)^2+xv(2)^2) -  ...
      sqrt((xv(1)-10)^2+xv(2)^2) - dv(2);

% subfunction funkfeuerjac
function jbk = funkfeuerjac(xv)
jbk(1,1) = xv(1)/sqrt(xv(1)^2+xv(2)^2) - ...
     xv(1)/sqrt(xv(1)^2+(xv(2)-10)^2);
jbk(1,2) = xv(2)/sqrt(xv(1)^2+xv(2)^2) - ...
     (xv(2)-10)/sqrt(xv(1)^2+(xv(2)-10)^2);
jbk(2,1) = xv(1)/sqrt(xv(1)^2+xv(2)^2) - ...
     (xv(1)-10)/ sqrt((xv(1)-10)^2+xv(2)^2);
jbk(2,2) = xv(2)/sqrt(xv(1)^2+xv(2)^2) - ...
     xv(2)/ sqrt((xv(1)-10)^2+xv(2)^2);
```

16.2 Miniprojekt zu den Funktionen mit mehreren Variablen

601 Elektrisches Potential

Lösungen zum Selbsttest

T611 – Senkrecht zum Gradientenvektor, also $t_h = [\partial F/\partial y \quad -\partial F/\partial x]^T$
– $v_z = \Delta F = \partial F/\partial x \cdot \Delta x + \partial F/\partial y \cdot \Delta y = (\partial F/\partial x)^2 + (\partial F/\partial y)^2$
– Nur so wird die Nebenbedingung als $\partial L/\partial \lambda = 0$ ein Teil des Gleichungssystems.
– $(-J^{-1})$

T612

```
[X,Y]=meshgrid(-1:0.02:1); h=max(1-X.^2-Y.^2,0); h=h.^2;
contour3(X,Y,h,25); axis equal
```

T613

```
hA=0; hB=1; hC=4; hD=1; [X,Y]=meshgrid(0:0.1:5);
h=hA + (hB-hA)*X/5 + (hD-hA)*Y/5 + (hC+hA-hD-hB)*Y.*X/25;
contour3(X,Y,h,25); axis equal; box on; pause
[gx,gy] = gradient(h,0.1,0.1); contour(X,Y,gx); pause
contour(X,Y,gy);
```

T614

$$L = 4x^2 + y^2 + z + \lambda(x + 2y + 4z - 12)$$

$$\left|\begin{array}{lcll} \frac{\partial L}{\partial x} & = & 8x+\lambda & = 0 \\ \frac{\partial L}{\partial y} & = & 2y+2\lambda & = 0 \\ \frac{\partial L}{\partial z} & = & 1+4\lambda & = 0 \\ \frac{\partial L}{\partial \lambda} & = & x+2y+4z-8 & = 0 \end{array}\right|$$

```
M=[8 0 0 1; 0 2 0 0; 0 0 0 4; 1 2 4 0]; b=[0 0 -1 8];
 x = [0.312 0.25 1.8672 -0.25]'
```

T615

```
% newtexa als Funktion damit f und jac intern sind
function xf = newtexa(xi,yi)
xv=[xi;yi];
for k=1:8; xv = xv-newtexajac(xv)\newtexafct(xv); end;
xf=xv;
% fvec = newtexafct(xv) Beispielfunktion 2D Newton
function fvec = newtexafct(xv)
fvec = zeros(2,1); fvec(1) = xv(1)^4+xv(2)^4-20;
fvec(2) = xv(1)^2 -xv(2)^2 -1;
%  newtexajac(xv) Beispielfunktion jac zu 2D Newton
function jac = newtexajac(xv)
jac(1,1)=4*xv(1)^3 ; jac(1,2)=4*xv(2)^3 ;
jac(2,1)=2*xv(1) ; jac(2,2)=-2*xv(2) ;
```

17
Lösungshinweise zum Kapitel 7

17.1
Lösungen der allgemeinen Übungen zum Kapitel 7

Analytische Lösungen im Symbolischen Modus
70–1 Geraden durch den Koordinatenursprung

```
syms x y(x)
ysol = dsolve(diff(y(x)) == y/x)
% ysol =
% C1*x
```

70–2 Konzentrische Kreise

```
syms x y(x)
ysol = dsolve(diff(y(x)) == -x/y)
% ysol =
%  2^(1/2)*(- x^2/2 + C2)^(1/2)
% -2^(1/2)*(- x^2/2 + C2)^(1/2)
% pretty(ysol)
%/              /    2      \ \
%|              |   x       | |
%|  sqrt(2) sqrt| - -- + C2 | |
%|              \    2      / |
```

70–3 Differentialgleichung zur Parabelschar

```
syms x y(x) K
para = dsolve(diff(y(x)) == 2*y/x, 'y(1)==K')
% para =
% K*x^2
```

70–4 Orthogonale Trajektorien zu den Parabeln

```
syms x y(x)
ellia = dsolve(diff(y(x)) == -x/(2*y), 'y(0) == 1' )
ellib = dsolve(diff(y(x)) == -x/(2*y), 'y(0) == 2' )
ellic = dsolve(diff(y(x)) == -x/(2*y), 'y(0) == 4' )
ezplot(ellia); hold on
ezplot(ellib);ezplot(ellic);
axis([-6 6 0 12]); axis square; hold off
```

70–5 Differentialgleichung 3. Ordnung mit konstanten Koeffizienten

```
syms x y(x)
solord3 =  dsolve('D3y + D2y -Dy +y == 1','x')
% solord3 =
%  C1*exp(-x*((19 - 3*33^(1/2))^(1/3)/3 +
%      (3*33^(1/2) + 19)^(1/3)/3 +1/3)) +...
%  C2*cos((3^(1/2)*x*((19 - 3*33^(1/2))^(1/3) -
%     (3*33^(1/2) + 19)^(1/3)))/6)*...
% exp(x*((19 - 3*33^(1/2))^(1/3)/6 + ...
%   (3*33^(1/2) + 19)^(1/3)/6 -1/3))-...
% C3*sin((3^(1/2)*x*((19 - 3*33^(1/2))^(1/3) -...
%   (3*33^(1/2) +19)^(1/3)))/6)*...
% exp(x*((19 - 3*33^(1/2))^(1/3)/6 + ...
% (3*33^(1/2) + 19)^(1/3)/6 - 1/3)) + 1
```

70–6 Hyperbolische Lösung

```
syms x y(x)
hyp = dsolve(y^2 == 4*(1+(diff(y(x))^2)))
% hyp =
%
%                                  2
%                                 -2
% 2*exp(- C6 - x/2) + exp(C6 + x/2)/2
%   2*exp(x/2 - C2) + exp(C2 - x/2)/2
```

70–7 Implizite Lösung

```
syms x y(x)
dsolve(x + y*diff(y) + (x^2+y^2)* 4*y^3*diff(y))
%  implizite Loesung
% ans =
```

```
% solve((x^2*exp(2*y^4))/2 + (y^2*exp(2*y^4))/2 == -C4, y)
```

Lösungen mit der Laplace-Transformation im Symbolischen Modus

70–8 Radioaktiver Tochterkern, Laplace-Lösung

```
% tochtlaplace.m  Laplace Loesung RA Zerfall Tochterkern
clear all ; syms ym(t) yt(t) t s k h N M u v ; format compact
% Zerfalls-Gleichung 1  dym(t)/dt = -k*ym(t)
% Zerfalls-Gleichung2   dyt(t)/dt = k*ym(t) - h*yt(t)
%  k, h Zerfallskonstanten Mutter- Tochterkerne
deqm = diff(ym(t),t) == -k*ym(t)  % Zerfall Mutterkerne
deqt = diff(yt(t),t) == k*ym(t) - h*yt(t) % Tochterkerne
deqml = laplace(deqm, t,s)  % Lapl Mutter
deqtl = laplace(deqt, t,s)  % Lapl Tochter
% im alg Glsyst. u und v für L(yn) und L(yt)
eqla = subs(deqml, laplace(ym(t), t, s), u ) % alg Gl 1
eqlb = subs(deqtl, laplace(ym(t), t, s), u ) % alg Gl 2a
eqlc = subs(eqlb, laplace(yt(t), t, s), v ) % alg Gl 2b
[M,N] = solve(eqla,eqlc,u,v)
ym(t) = ilaplace(M,s,t)  % exp(-k*t)*ym(0)
yt(t) = ilaplace(N,s,t)  % n(t) = exp(-k*t)*n(0)
% yt(t) = (k*exp(-k*t)*ym(0))/(h - k) - ...
%      (exp(-h*t)*(k*ym(0) - h*yt(0) + k*yt(0)))/(h - k)
```

70–9 Differentialgleichung 2. Ordnung, Laplace-Lösung

```
syms s t YY yy
YY  = sym('Y(t)')
d2 =    laplace(diff(diff(YY)))
d0 = laplace(YY)
Gl = d2  + 4*d0 == laplace(exp(3*t))
Gli1 = subs(Gl,'Y(0)',1)
Gli2 = subs(Gli1,'D(Y)(0)',-1)
Glyy = subs(Gli2,'laplace(Y(t),t,s)',yy)
y    = solve (Glyy,yy)
Y = simplify(ilaplace(y),20)
% Glyy =  yy*s^2 - s + 4*yy + 1 == 1/(s - 3)
% y = (s + 1/(s - 3) - 1)/(s^2 + 4)
% Y = (12*cos(2*t))/13 + exp(3*t)/13 - (8*sin(2*t))/13
```

70–10 Einfacher Oszillator, Laplace-Lösung

```
syms s t YY yy
YY  = sym('Y(t)')
d2 =    laplace(diff(diff(YY)))
d0 = laplace(YY)
```

```
Gl = d2 + d0 ==  laplace(cos(t))
Gli1 = subs(Gl,'Y(0)',2)
Gli2 = subs(Gli1,'D(Y)(0)',0)
Glyy = subs(Gli2,'laplace(Y(t),t,s)',yy)
y    = solve (Glyy,yy)
Y = simplify(ilaplace(y),20)
ezplot(Y,[0 150 -30 30])
% unbeschraenkte Aufschaukelung exakt in Resonanz
```

70–11 Resonanz-Aufschaukelung, Laplace-Lösung

```
% Skript laposcdmpreso
syms s t YY yy
YY  = sym('Y(t)')
d2 =    laplace(diff(diff(YY)))
d1 = laplace(diff(YY))
d0 = laplace(YY)
Gl = d2 +0.1*d1 + 9* d0 ==  laplace(sin(3*t))
Gli1 = subs(Gl,'Y(0)',0)
Gli2 = subs(Gli1,'D(Y)(0)',0)
Glyy = subs(Gli2,'laplace(Y(t),t,s)',yy)
y    = solve (Glyy,yy)
Y = simplify(ilaplace(y),20) % Aufschaukeln bis Anregung
 ezplot(Y,[0 100 -20 20])    % und Daempfung ausgeglichen
```

70–12 Oszillator mit Anregung ausser Resonanz, Laplace-Lösung

```
% Skript laposcoffres
%  Laplace Loesung Oszillator
%   mit Anregung ausserhalb Resonanz
syms s t YY yy
YY  = sym('Y(t)')
d2 =    laplace(diff(diff(YY)))
d0 = laplace(YY)
Gl = d2 + 4* d0 ==  laplace(sin(3*t))
Gli1 = subs(Gl,'Y(0)',0)
Gli2 = subs(Gli1,'D(Y)(0)',0)
Glyy = subs(Gli2,'laplace(Y(t),t,s)',yy)
y    = solve (Glyy,yy)
Y = simplify(ilaplace(y),20)
 ezplot(Y,[0 50 -3 3])
```

Numerische Lösung von Differentialgleichungen
70–13 Wegweiserfeld

```
% wegweiser(xmx,ymx,f) zeichnet ein Wegweiserfeld zu y'=-f*y
%   mit Pfeilen an den Positionen 0:xmx 0:ymx
function voidbk = wegweiser(xmx,ymx,f)
hold on
for x=0:1:xmx
  for y=0:1:ymx
    w=atan(f*y);   putarrow(x,y,w);
  end
end
hold off
% internal function
function voidbk = putarrow(xpos,ypos,win)
xars = [0.08 0.25 -0.25 0.25 0.08; ...
        0.01  0     0     0    -0.01 ; 1 1 1 1 1];
xpl= [cos(win) -sin(win) xpos;...
     sin(win) cos(win) ypos; 0 0 1]*xars;
plot(xpl(1,:),xpl(2,:))
for x=0:1:25
  for y=0:1:18
    w=atan(-0.1*y);     putarrow(x,y,w);
  end
end
```

70–14 Einfache Differentialgleichung:
$dy/dt = -0.2 \cdot y$ separieren $dy/y = -0.2dt$ integrieren $\ln(y) = -0.2t + C$
$y(x) = y(0) \cdot \mathrm{e}^{-0.2t}$

```
% simpdecay(f) L\"osung der simplen Dgl. y'=-f*y
yin = 15;
[tsol,ysol] = ode45('simpdecayeval',25, yin);
plot(tsol,ysol);

% deri= simpdecayeval(tval,yval) die Ableitung y'=-0.2*y
function deri  = simpdecayeval(tval,yval);
deri = -0.2* yval;
```

70–15 Euler-Verfahren eindimensional

```
% selfeulerdecay  - ausprogrammiertes Euler-Verfahren
%   fuer Zerfallsgleichung  y'=-0.1*y
ttot = 50;  yin = 20;  lam = -0.1;
```

```
h = ttot/1000;  tsol = [0:1000]'*h;
ysol= zeros(1001,1);
%init
ysol(1) = yin;
%loop
for k=2:1001
  ysol(k) = ysol(k-1) + h* (lam*ysol(k-1));
end
```

70–16 Einzelschritte mit dem Euler-Verfahren

$$dx/dt = u$$
$$du/dt = -0.1 \cdot v$$
$$dy/dt = v$$
$$dv/dt = 0.1 \cdot u$$

```
  yvec0 dyvec0  yvec1 dyvec1   yvec2   dyvec2   yvec3
 x  10   0       10    -0.02    9.996  -0.04     9.988
 u   0  -0.1     -0.02 -0.1    -0.04   -0.1     -0.06
 y   0   1        0.2   1       0.4    0.9996   0.5999
 v   1   0        1   -0.002   0.9996  -0.004   0.9988
 t = 0           0.2            0.4             0.6
```

70–17 Euler-Verfahren analog zur Bibliotheksprozedur

```
% [tsol,ysol]= selfeuler(fctnam, ttot, yin)
%  ausprogrammierter ode-solver nach  Euler
%  mit gleicher signatur wie ode45()
function [tsol,ysol]= selfeuler(fctnam, ttot, yin)
h = ttot/1000;  tsol = [0:1000]'*h;
ysol= zeros(1001,length(yin));
%init
ysol(1,:) = yin';
%loop
for k=2:1001
  ysol(k,:) = ysol(k-1,:) +...
      h* ( feval(fctnam, (k-1)*h , ysol(k-1,:)') )';
end
```

Differentialgleichungen höherer Ordnung

70–18 Umsetzen von Differentialgleichungen höherer Ordnung in Systeme erster Ordnung

```
% deriv = dmposc(t,yac) Beispiel gedaempfter Oszillator
```

```
function deriv = dmposc(t,yac)
w = 1.08;
d = 0.01;
deriv = zeros(2,1);
deriv(1) = yac(2);
deriv(2) = -d*yac(2) -w*w*yac(1) + 0.2*sin(t) ;

% Skript rundmposc Integrieren und Zeichnen
yin = [0 0]';
[tsol, ysol] = ode45('dmposc', 200 , yin);
figure(1); clf
plot(tsol,ysol(:,1))
figure(2); clf
plot(tsol,ysol(:,2),'r')
```

70–19 Differentialgleichungen höherer Ordnung

```
% haronmosc einfacher harmonischer Oszillator
function deri = harmonosc(tac,yac)
deri = zeros(2,1); deri(1) = yac(2) ; deri(2) = -yac(1);
% dmposc501  Beispiel gedaempfter Oszillator
function deri = dmposc501(tac,yac)
deri = zeros(2,1); deri(1) = yac(2) ;
deri(2) = -25*yac(1) -0.1*deri(1);
% splinederiv Ableitung der Spline-Bedingung y(IV) = 0
function deri = splinederiv(tac,yac)
deri = zeros(4,1); deri(1) = yac(2) ;
deri(2) = yac(3); deri(3) = yac(4) ; deri(4) = 0;
```

70–20 Freier Fall mit Luftwiderstand

```
% runluftfall  - freier Fall mit Luftwiderstand
%   vorgaengig CLUFTBREMS definieren (0.5 - 0.001)
global CLUFTBREMS
[tsol,ysol] = ode45('luftfallderi',100,[0,0]);
figure(1); clf; plot(tsol,ysol(:,1))
figure(2); clf; plot(tsol,ysol(:,2)); figure(1)

% deri = luftfallderi(tac,yac) Ableitung Fall in Luft
function deri = luftfallderi(tac,yac)
global CLUFTBREMS
deri = zeros(2,1);
deri(1)=yac(2);
deri(2)=-9.81 + CLUFTBREMS*(yac(2))^2;
```

70–21 Oszillator mit Parametereingabe von außen

Die Ableitungsfunktion mit globalen Parametern lautet:

```
function deriv = dmposcpare(t,yac)
global EFREQ
global OSCDAMP
global ANRFREQ
global ANRAMP
w = EFREQ;
d = OSCDAMP;
afr = ANRFREQ;
A = ANRAMP;
deriv = zeros(2,1);
deriv(1) = yac(2);
deriv(2) = -d*yac(2) -w*w*yac(1) + A*sin(afr*t) ;
```

Im Anwendungs-Skript können verschiedene Varianten ausprobiert werden, indem man Zeilen deaktiviert (mit dem %-Zeichen) und andere modifiziert.

```
%yin = [10 0]';
yin = [0 0]';
global EFREQ
global OSCDAMP
global ANRAMP
global ANRFREQ
% EFREQ = 1.2; OSCDAMP = 0.05; ANRAMP = 0; ANRFREQ = 0;
% EFREQ = 2; OSCDAMP = 0.1; ANRAMP = 1; ANRFREQ = 2.5;
EFREQ = 2; OSCDAMP = 0.1; ANRAMP = 1; ANRFREQ = 2;
[tsol, ysol] = ode45('dmposcpare', 50 , yin);
[teu, yeu] = odeeuler('dmposcpare', 50 , yin);
figure(1)
clf
plot(tsol,ysol(:,1))
hold on
plot(teu,yeu(:,1),'k')
hold off
figure(2)
clf
```

Hier wurde gerade noch der selbst prorammierte Euler-Löser zum Vergleich eingesetzt. Damit man nicht lange suchen muss, wird der Code hier nochmals aufgeführt.

```
function [tsol,ysol]= odeeuler(fctnam, bereich, ystart)
h = bereich/8000;
tsol = [0:8000]'*h;
ysol= zeros(8001,length(ystart));
%init
```

```
ysol(1,:) = ystart';
ywrk = ystart;
%loop
for k=2:8001
% Extrapolationsschritt ergibt naechstes y
  ywrk = ywrk + h* feval(fctnam, ysol(k-1) , ywrk);
% Abfuellen
  ysol(k,:) = ywrk';
end
```

70–22 Pendel ohne Kleinwinkel-Approximation

```
function realpendel(ampini)
%  function realpendel(ampini)
% Auslenkwinkel der Schwingung in Radian
leff = 1; g = 9.81;
  yin = [ampini 0];
  [tsh,ysh] = ode45(@harmpend, [0 15], yin);
  [tss,yss] = ode45(@sinpend, [0 15], yin);
  figure(1)
  plot(tsh,ysh(:,1)); hold on;
  plot(tss,yss(:,1),'--'); hold off
  figure(2)
   plot(ysh(:,1),ysh(:,2)); hold on;
   plot(yss(:,1),yss(:,2)); hold off
  %
  function yp = harmpend(t,y)
    yp(1,1) = y(2); yp(2,1) = -g/leff*y(1);
  end
  function yp = sinpend(t,y)
    yp(1,1) = y(2); yp(2,1) = -g/leff*sin(y(1));
  end

end % function
```

Systeme von Differentialgleichungen

70–23 Schiefer Wurf in 3D

Lösung mit einem Beispiel für die Event-Handhabung:

```
function [ts,ysv,te,ye,ie] = schiefwurf(vo,azi,elev)
% ts, ysv Flugbahnen des schiefen Wurfes ohne Luftwiderstand
% te,ye   Positionen der Events vz=0(Scheitel), z=0(Ende), z=10
y0 = [0 vo*cosd(elev)*cosd(azi) 0 vo*cosd(elev)*sind(azi) ...
        0 vo*sind(elev) ];
tfinal = y0(6)/4.5;
  options = odeset('Events',@scheitelboden);
  [ts,ysv,te,ye,ie] = ode45(@wurfabl,[0 tfinal],y0,options);
```

```
% ---------------------------------------------------------------
function ablvec = wurfabl(t,rvc)
% function ablvec = wurfabl(t,rvc)
%  berechnet die ersten Ableitungen von
%    x,vx, y,vy, z,vz bei Gravitation ohne Luftwiderstand
   ablvec = zeros(6,1);   % Spaltenvektor vordefinieren
   ablvec(1) = rvc(2);    % x' = vx  (vx'= 0 vordefiniert)
   ablvec(3) = rvc(4);    % y' = vy  (vy'= 0 vordefiniert)
   ablvec(5) = rvc(6);    % z' = vz
   ablvec(6) = -9.81;     % z'' = vz' = - g
end
% ---------------------------------------------------------------
function [val,enddef,richtung] = scheitelboden(tev,yev)
% function [value,isterminal,direction] = scheitelboden(tev,yev)
% Eventfunktion zum definieren der Punkte bei denen val = 0 wird
  val(1) = yev(6);     % Scheitelpunkt: vz = 0
  enddef(1) = 0;       % hier weiterfahren
  richtung(1) = -1;    % 0 von Pos zu Neg ueberqueren
  val(2) = yev(5);     % Bodenhoehe Null:  z = 0
  enddef(2) = 1;       % Integration abbrechen
  richtung(2) = -1;    % 0 von Pos zu Neg ueberqueren
  val(3) = yev(5)-10;  % Bodenhoehe 10:  z -10 = 0
  enddef(3) = 0;       % weiterfahren
  richtung(3) = 0;     % Richtung egal
end
end
```

70–24 Radioaktiver Tochterzerfall

```
% runtochzerfpar zeigt Zeitfunktionen eines radioaktiven
%   Zerfalls mit Tochterkern
%   globale Parameter: LAMBDA1  LAMBDA2 interaqktiv abgefragt
global LAMBDA1;   global LAMBDA2
  LAMBDA1 = input('lambda1 eingeben');
  LAMBDA2 = input('lambda2 eingeben');
yin = [10 0]'
[xsol, ysol] = ode45('tochzerfpar', 50 , yin);
plot(xsol,ysol)

% deri = tochzerfpar(t,y) Ableitungen zum RA-Tochterzerfall
function deri = tochzerfpar(t,y)
global LAMBDA1 ; global LAMBDA2
deri = zeros(2,1);
deri(1) = -LAMBDA1*y(1);
deri(2) = LAMBDA1*y(1) - LAMBDA2*y(2);
```

70–25 Geladenes Teilchen in 2D im konstanten Magnetfeld

```
% runchargpart.m
ystart=[ 0 0 0 1]';
[t45,y45]= ode45('chargpartderi',200,ystart);
plot(y45(:,1),y45(:,3),'r')

% deriv = chargpartderi(t,yac) Ableitung Teilchen 2D
%   in konstantem Magnetfeld
function deriv = chargpartderi(t,yac)
B=-1/10;  deriv = zeros(4,1);
deriv(1) = yac(2);  deriv(2) =  yac(4)*B ;
deriv(3) = yac(4);  deriv(4) = -yac(2)*B ;
```

70–26 bis 70–28 siehe M-File Sammlung

Traktrix und andere Verfolgungsprobleme
70–29 Klassische Traktrix-Differentialgleichung

```
function  [ts, ys] = tractrixn(a)
% function  [ts, ys] = tractrixn(a)
%  Numerische Lösung der klassischen
%  Traktrix-Differentialgleichung als System x(t),y(t)
yini = [0 a];   vw = 0.1;
[ts,ys] = ode45(@trxndiff, [0 10*a/vw], yini);
plot(ys(:,1),ys(:,2))
    function ydiff = trxndiff(t,yac)
        ydiff(1,1)=  vw*(vw*t-yac(1)) / (a);
        ydiff(2,1)= -vw*yac(2) / (a);
    end
end
```

70–30 Verallgemeinerte Traktrix

Als Anwendungsbeispielen wird eine kreisförmige Schleppkurve angenommen

```
function  [ts, ys] = tractrixcirc(yvstart,rad,wrot)
% function  [ts, ys] = tractrixcirc(yvstart,rad,wrot)
%  Numerische Lösung der verallgemeinerten
%  Traktrix-Differentialgleichung
%   mit kreisfoermiger Schleppkurve
yini = yvstart; a = sqrt(yini(1)^2+yini(2)^2);
[ts,ys] = ode45(@trxgendiff, [0 300], yini);
plot(ys(:,1),ys(:,2))
    function [wp , dwp] = wptf(t,rad,wrot)
```

```
        wp(1,1) = rad*cos(wrot*t);
        wp(2,1) = rad*sin(wrot*t);
        dwp(1,1) = -rad*wrot*sin(wrot*t);
        dwp(2,1) = rad*wrot*cos(wrot*t);
    end
    function ydiff = trxgendiff(t,yac)
        [wp,dwp] = wptf(t,rad,wrot);
        svec = [wp(1)-yac(1); wp(2)-yac(2)];
        % dx,y/dt ist Projektion von dwp auf svec
        dmov = svec*(svec'*dwp)/(svec'*svec);
        ydiff(1,1)=  dmov(1);
        ydiff(2,1)=  dmov(2);
    end
end
```

Das nächste Beispiel sieht eine Schleppkurve entlang der Geraden `wpini + t* wplin` mit dem Ortsvektor wpini und dem Richtungsvektor wplin vor. Vielseitige Experimente sind angesagt!

```
function  [ts, ys] = tractrixgen(yvstart,wpini,wplin)
% function  [ts, ys] = tractrixgen(yvstart,wpini,wplin)
%  Numerische Lösung der verallgemeinerten
%  Traktrix-Differentialgleichung
%  mit der Geraden wpini+t*wplin als Schleppkurve
yini = yvstart; a = sqrt(yini(1)^2+yini(2)^2);
[ts,ys] = ode45(@trxgendiff, [0 300], yini);
plot(ys(:,1),ys(:,2))
    function [wp , dwp] = wptf(t,wpini,wplin)
        wp(1,1) = wpini(1) + wplin(1)*t;
        wp(2,1) = wpini(2) + wplin(2)*t;
        dwp(1,1) = wplin(1); dwp(2,1) = wplin(2);
    end
    function ydiff = trxgendiff(t,yac)
        [wp,dwp] = wptf(t,wpini,wplin);
        svec = [wp(1)-yac(1); wp(2)-yac(2)];
        % dx,y/dt ist Projektion von dwp auf svec
        dmov = svec*(svec'*dwp)/(svec'*svec);
        ydiff(1,1)=  dmov(1);
        ydiff(2,1)=  dmov(2);
    end
end
```

70–31 Weitere Verallgemeinerung - variable Länge

Durch Angabe einer Geschwindigkeit des Verfolgers kann sich die Schlepp-Länge ändern, und der Verfolger kann den Verfolgten bei genügend hoher Geschwindigkeit einholen. Dies ist ein Anwendungsbeipiel für die Event-Steuerung, welche für den Abbruch der Berechnung beim Zusammentreffen sorgt.

```
function  [ts, ys, yw] = tractrixmeetev(yvstart,wpini,wplin,speed)
% function  [ts, ys] = tractrixmeet(yvstart,wpini,wplin,speed)
%  Numerische Lösung der verallgemeinerten
%  Traktrix-Differentialgleichung
%  mit der Geraden wpini+t*wplin als Schleppkurve.0
%  und der Geschwindigkeit des Verfolgers speed
yini = yvstart; a = sqrt(yini(1)^2+yini(2)^2);
termopt = odeset('Events',@targmet);
[ts,ys,te,ye,ie] = ode45(@trxgendiff, [0 300], yini,termopt);
yw(:,1) = wpini(1) + wplin(1)*ts ;
yw(:,2) = wpini(2) + wplin(2)*ts ;
plot(ys(:,1),ys(:,2))
    function [wp , dwp] = wptf(t,wpini,wplin)
        wp(1,1) = wpini(1) + wplin(1)*t;
        wp(2,1) = wpini(2) + wplin(2)*t;
        dwp(1,1) = wplin(1); dwp(2,1) = wplin(2);
    end
    function ydiff = trxgendiff(t,yac)
        [wp,dwp] = wptf(t,wpini,wplin);
        svec = [wp(1)-yac(1); wp(2)-yac(2)];
        % dx,y/dt ist Projektion von dwp auf svec
        dmov = svec/norm(svec)*speed;
        ydiff(1,1)=  dmov(1);
        ydiff(2,1)=  dmov(2);
    end
    function [val,enddef,richtg] = targmet(tev,yev)
        tol = 0.0001;
         val  = (yev(1) - wpini(1) -tev*wplin(1))^2 + ...
            (yev(2) - wpini(2) -tev*wplin(2))^2 - tol;
        enddef = 1;
        richtg = -1;
    end
end
```

70–32 Das Verfolgungsproblem der vier Käfer

```
function [ts,ys] = trak4()
% trak4 Klassiker vier Kaefer
%  vier gegenseitig aufeinander zulaufende Linien
  yini = [10 10 -10 10 -10 -10 10 -10]';
    function yp = tr4diff(t,y)
        v = 0.5; v2 = 0.5; yp = zeros(7,1);
        yp(1) = v*(y(3)-y(1))/sqrt((y(3)-y(1))^2+(y(4)-y(2))^2);
        yp(2) = v*(y(4)-y(2))/sqrt((y(3)-y(1))^2+(y(4)-y(2))^2);
        yp(3) = v2*(y(5)-y(3))/sqrt((y(5)-y(3))^2+(y(6)-y(4))^2);
        yp(4) = v2*(y(6)-y(4))/sqrt((y(5)-y(3))^2+(y(6)-y(4))^2);
```

```
        yp(5) = v*(y(7)-y(5))/sqrt((y(7)-y(5))^2+(y(8)-y(6))^2);
        yp(6) = v*(y(8)-y(6))/sqrt((y(7)-y(5))^2+(y(8)-y(6))^2);
        yp(7) = v2*(y(1)-y(7))/sqrt((y(1)-y(7))^2+(y(2)-y(8))^2);
        yp(8) = v2*(y(2)-y(8))/sqrt((y(1)-y(7))^2+(y(2)-y(8))^2);
    end
  [ts,ys] = ode45(@tr4diff,[0 40], yini);
  plot(ys(:,1),ys(:,2)); hold on
  axis([-10 10 -10 10]); axis square
  plot(ys(:,3),ys(:,4));
  plot(ys(:,5),ys(:,6));
  plot(ys(:,7),ys(:,8)); hold off
end
```

70–33 Äquitangentiale

Die Funktion aequitang erwartet als Input eine Anzahl Teilstrecken, mit typ = 1,2,3 für Gerade, Rechtskurve und Linkskurve. Der Parameter lengwin gibt die Länge in Meter bzw. den Winkel in Grad an. Die Resultate poshint und posvord zeiden die Spur der Hinterräder bzw. der Vorderräder

```
function [poshint,posvord] = aequitang(rdstand,typ,lengwin)
% function [poshint,posvord] = aequitang(rdstand,typ,leng)
%  Aequitangentiale am Beispiel eines Autos
%  rdstand  = Radstand
%  typ 1 = Gerade 2 = Rechtskurve, 3 = Linkskurve
% lengwin = Laenge oder Winkel
dl = rdstand/10; Rkurv = 8; dwbas = dl/Rkurv*180/pi;
lfz = 7.5;
poshint = [0 0]; posac = [0 0]; w = 0; war = w;
for k = 1:length(typ)
    if typ(k) == 1
        nstep = ceil(lengwin(k)/dl);
        for s = 1:nstep
            posac=posac+lengwin(k)/nstep*[cosd(w) sind(w)];
            poshint = [poshint ; posac];
            war = [war ; w];
        end
    elseif typ(k) == 2
        nstep = ceil(lengwin(k)/dwbas);
          for s = 1:nstep
            dw = - lengwin(k)/nstep;
            dpos = [cosd(w) -sind(w); sind(w) cosd(w)]*...
                [-Rkurv*sind(dw); -Rkurv*(1-cosd(dw))];
            posac = posac + dpos';
            poshint = [poshint ; posac];
            w = w +dw;
            war = [war ; w];
        end
```

```
    elseif typ(k) == 3
         nstep = ceil(lengwin(k)/dwbas);
        for s = 1:nstep
            dw = lengwin(k)/nstep;
            dpos = [cosd(w) -sind(w); sind(w) cosd(w)]*...
                 [Rkurv*sind(dw); Rkurv*(1-cosd(dw))];
            posac = posac + dpos';
            poshint = [poshint ; posac];
            w = w +dw;
            war = [war ; w];
        end
    end
end
  %for p = 1:length(poshint)
    posvord = poshint+lfz*[cosd(war) sind(war)];
  %end
end
```

Als Beispiel kann eine Abfolge von Geraden und Kurven mit dem folgenden Skript berechnet und gezeichnet werden. Typisch ist das über die Kurve Hinausfahren und dann ganz kurzfristige Einlenken.

```
[poshint,posvord] = aequitang(7.5,[1 3 1 3 1 2 3 2 1],...
      [30 90 15 90 20 180 45 45 15]);
clf
plot(poshint(:,1),poshint(:,2))
axis equal ;  hold on;
plot(posvord(:,1),posvord(:,2))  ; hold off
```

Spline-Interpolationsfunktionen

70–34 Spline-Interpolationsfunktion als Randwertproblem

a) $y^{IV} = 0$ hat die allgemeine Lösung $f(x) = ax^3 + bx^2 + cx + d$. Die 4 Bedingungen an den beiden Rändern legen $f(0)$, $f(1)$, $f'(0)$ und $f'(1)$ fest. Das Gleichungssystem

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ f_0' \\ f_1' \end{pmatrix}$$

liefert $d = f_0,\ c = f_0',\ b = -3f_0 + 3f_1 - 2f_0' - f_1',\ a = 2f_0 - 2f_1 + f_0' + f_1'$
Dies ergibt für das Beispiel: $f(x) = -0.6x^3 + 0.3x^2 + 0.5x + 1$
b) Daraus ergeben sich die 4 elementaren Splinefunktionen zu $f_0 = 1$: $2x^3 - 3x^2 + 1$, zu $f_1 = 1$: $-2x^3 + 3x^2$, zu $f_0' = 1$: $x^3 - 2x^2 + x$, zu $f_1' = 1$: $x^3 - x^2$.

70–35 Elementarfunktionen der Spline-Interpolation

Die 4 Elementar-Splines sind:
B1:$2t^3 - 3t^2 + 1 = (2t+1)\cdot(t-1)^2$ Doppelte Nullstelle bei 1
B2:$-2t^3 + 3t^2 = t^2 \cdot (3-2t)$ Doppelte Nullstelle bei 0
B3:$t^3 - 2t^2 + t = t \cdot (t-1)^2$ Doppelte Nullstelle bei 1, einfache bei 0
B4:$t^3 - t^2 = t^2 \cdot (t-1)$. Doppelte Nullstelle bei 1, einfache bei 0
Beim Auswechseln von$t$ gegen $(1-t)$ wird aus B4 $(1-t)^2 \cdot t$, also B3 und aus B2 $-t^2 \cdot (2t-3)$ wird $(1-t)^2 \cdot (2t+1)$, also B1.

17.2
Miniprojekt zu den Differentialgleichungen

701 Algen- und Fischbestand

```
function [ts,ys] = algenfische(M,c1,c2,c3,c4)
%  function [ts,ys] = algenfische(c1,c2,c3,c4)
%  Algen- und Fischbestand
    yini = [10 2];
    [ts,ys] = ode45(@algfideri,[0 100],yini)
    plot(ts,ys)
    function yp = algfideri(t,y)
        yp(1,1) = c1*y(1)*(1-y(1)/M) - c2*y(2)*y(1);
        yp(2,1) = c3*y(1) - c4*y(2)/y(1);
    end
end
```

17.3
Lösungen zu den Selbsttests

Lösungen der Testserie 7.1

T711 – Differentialgleichungen sind Funktionsgleichungen; die Gleichheit muss in einem ganzen Bereich der unabhängigen Variablen überall erfüllt sein.
–3 mal 2 = 6
– Man muss die Lösung ysol gegen tsol plotten nicht nur ysol allein.
– Damit die Signatur (Struktur der Parameter beim Aufruf) für die im innern aufgerufene Ableitungfunktion immer gleich ist.
T712 $dV = 1.3^2\pi \cdot dh$ $dV/dt = \sqrt{2 \cdot 9.81 \cdot h} \cdot 0.04^2\pi\ dh/dt = \sqrt{2 \cdot 9.81 \cdot h} \cdot 0.04^2\pi/(1.3^2\pi)$

```
% deri = tankleerderi(tac,yac) Ableitung Tank-Entleerung
function deri = tankleerderi(tac,yac)
if yac > 0 % Absichern: Wurzelnenner muss positiv sein
  deri= -sqrt(2 * 9.81 * yac) * 0.04^2*pi/(1.3^2*pi);
else
  deri=0;
end
```

T713

```
% varimagderi Ableitung Teilchen im variablen Magnetfeld
function deri = varimagderi(tac,yac)
B=0.1*yac(1);
deri = zeros(4,1); deri(1)=yac(2); deri(2) = 1/B*yac(4);
deri(3)=yac(4); deri(4) = -1/B*yac(2);
```

T714

```
% circmotionderi  Ableitung zu Kreisbewegung
function deri = circmotionderi(tac,yac)
deri = zeros(4,1); deri(1)=yac(2); deri(2) = 0.05*yac(4);
deri(3)=yac(4); deri(4) = -0.05*yac(2);
```

$$dx/dt = u$$
$$du/dt = 0.05 \cdot v$$
$$dy/dt = v$$
$$dv/dt = -0.05 \cdot u$$

	yvec0	dyvec0	yvec1	dyvec1	yvec2	dyvec2	yvec3
x	0	1	0.2	1	0.4	0.9999	0.6
u	1	0	1	-0.0005	0.9999	-0.001	0.9997

```
 y  20   0       20    -0.01   19.998   -0.02   19.994
 v   0 -0.05    -0.01  -0.05   -0.02    -0.05    -0.03
 t = 0           0.2            0.4             0.6
```

T715

```
syms t y(t) c
ys = dsolve('D2y +c*Dy == 0','y(0)== 10','Dy(0) == 1')
%  ys = (10*c + 1)/c - exp(-c*t)/c
```

Lösungen der Testserie 7.2

T721 – Drei Anfangswerte

– Weil oft die algebraischen Gleichungen im Laplace-Raum einfacher zu handhaben sind als die Differentialgleichungen im Zeitraum.

– Der Rechenaufwand steigt meist enorm an, weil eine Schrittweitensteuerung immer kleinere Schritte verlangt.

– Ein Spaltenvektorm für die Ableitungen, welcher in der Abfolge dem Spaltenvektor für die gesuchten Funktionen entspricht.

T722

```
function  [ts, ys] = kettlinnum(p)
% function  [ts, ys] = kettlinnum(p)
%  Numerische Lösung der Kettenlinie
yini = [0 0];
[ts,ys] = ode45(@kettldiff, [0 10], yini);
%plot(ys(:,1),ys(:,2))
    function ydiff = kettldiff(t,yac)
        ydiff = zeros(2,1);
        ydiff(1)= yac(2) ;
        ydiff(2)=  p*sqrt(1 + yac(2)^2) ;
    end
end
```

T723

```
syms x y(x)
ysol = dsolve(diff(y(x)) == -x/y)
% Ableitung senkrecht zur Richtung gegen 0/0
%   ergibt konzentrische Kreise
% ysol =
%  2^(1/2)*(- x^2/2 + C2)^(1/2)
% -2^(1/2)*(- x^2/2 + C2)^(1/2)
% pretty(ysol)
%/              /    2      \ \
%|              |   x       | |
%|  sqrt(2) sqrt| - -- + C2 | |
%|              \    2      / |
```

T724

```
% verhulstlap.m  Laplace Loesung Verhulst Dgl
clear all ; syms y(t) yy t s  u ; format compact
% Verhulst-Gleichung $y' = c * y * (1-y/M) vereinfacht 3/2*y*(1-y)
% deq = diff(y(t),t)== 3/2*y(t)*(1-y(t))
deq = diff(y(t),t)== 3/2*(y(t)-y(t)^2)% t-Differentialgleichung
deql = laplace(deq, t, s)    % L-transfomierte Dgl
eqla = subs(deql, laplace(y(t), t, s), u ) % alg Gl.
yy = solve(eqla, u)   % yy = alg. Gl. nach u aufgeloest
y(t) = ilaplace(yy,s,t)
% y(t) = exp((3*t)/2)*y(0) -
%     (3*ilaplace(laplace(y(t)^2, t, s)/(s - 3/2), s, t))/2
%   nicht analytisch loesbar
% sorry, bei Aufgabenstellung nicht genuegend recherchiert
```

T725

```
% deri= fallluftderi(tac,yac) Ableitung
%  freier Fall mit Luftwiderstand
function deri = fallluftderi(tac,yac)
deri = zeros(2,1); deri(1)=yac(2);
% a) deri(2) = -9.81
deri(2) = -9.81 + deri(2)^2*0.05;
```

18
Lösungshinweise zum Kapitel 8

18.1
Lösungen zu den allgemeinen Übungen im Kapitel 8

Zusammenfassung grosser Datenmengen
80–1 Median und Quartile für einfache Folge

```
% Skript medqgleich
x = 10:30;
xm = median(x), q1 = prctile(x,25) ,q3 = prctile(x,75)
%  xm = 20   q1 =  14.7500   q3 =    25.2500
```

80–2 Median und Quartile bei Sägezahn-Funktion

```
% Skript medqsaegz
x = [1 2 2 3 3 3 4 4 4 4 5 5 5 5 5 6 6 6 6 6 6];
xm = median(x), q1 = prctile(x,25) ,q3 = prctile(x,75)
%  xm =  5  q1 =   3  q3 = 6
z = [1 1 1 1 1 1 2 2 2 2 2 3 3 3 3 4 4 4 5 5 6];
zm = median(z), qz1 = prctile(z,25) ,qz3 = prctile(z,75)
%  zm =  2  qz1 =   1  qz3 = 4
```

80–3 Parameter der Differenzbasierten Familie

```
% Skript stdsaegz
x = [1 2 2 3 3 3 4 4 4 4 5 5 5 5 5 6 6 6 6 6 6];
xm = mean(x), sx = std(x)
skx = skewness(x), curx = kurtosis(x)
%  xm = 4.3333   sx = 1.5275
%   skx = -0.5814   curx =  2.3100  (-3)
z = [1 1 1 1 1 1 2 2 2 2 2 3 3 3 3 4 4 4 5 5 6];
zm = mean(z), sz = std(z)
skz = skewness(z), curz = kurtosis(z)
%  zm = 2.6667  sz =  1.5275
%   skz =  0.5814   curz =  2.3100
```

```
% Schiefe fuer z positiv (nach rechts auslaufend)
```

80–4 Parameter bei einzelnen Datenpunkten

```
% Skript tridistr  Dreipunkte Verteilung
x = [-ones(1,5) zeros(1,10) ones(1,5)];
mdix = median(x), q1 = prctile(x,25), q3 = prctile(x,75)
% mdix =  0    q1 =  -0.5000  q3 =  0.5000
xm = mean(x), xstd = std(x)
%  xm =   0   xstd =   0.7255
```

80–5 Quantile von normalverteilten Daten

```
% skript normquantil
siz = [20 200 1000];
przmat = zeros(10,4);
for s = 1:3
  for r = 1:10
    z = randn(1,siz(s));
    prcv = prctile(z,[20,40,60,80]);
    przmat(r,:) = prcv;
  end
  prmat{s} = przmat;
end
prmat{1}
prmat{2}
prmat{3}
```

80–6 Elementare Glockenkurven

Durch die Funktionen $y = \max(0, (1 - x^2))^k$ werden für $k = 1$ die umgekehrte Parabel und für $k > 1$ verschiedene Glockenkurven definiert. Bestimmen Sie für einige dieser Fälle σ (μ ist immer Null) und die dazu passende Gauß'sche Glockenkurve und erzeugen Sie eine gemeinsame Grafik der Kurvenpaare.
Der Exponent `kexp` kann in der folgenden Funktion vorgewählt werden:

```
function glockenvergleich(kexp)
% glockenvergleich Vergleich Glockenkurven
%    mit Normalverteilung
xgl = -1:0.05:1; xnor = -1.5:0.05:1.5;
ygl = (1-xgl.^2).^kexp;
plot(xgl,ygl,'b','linewidth',2)
 [me,fs] = normparf(xgl,ygl)
 si = sum(ygl)*(xgl(2)-xgl(1));
 ynor = normpdf(xnor,me,fs)*si;
 hold on
 plot(xnor,ynor,'r','linewidth',2)
```

```
 hold off
   function [fctmean,fctstd] = normparf(xv,yv)
     m1  = sum(xv.*yv); w = sum(yv);
     fctmean = m1/w;
     fctstd = sqrt(sum((xv-fctmean).^2.*yv)/w);
   end
end
```

80–7 Vergleich zu Normalverteilungs-Dichte

Testen Sie die Nähe bzw. die Abweichung der in 80–6 beschriebenen Glockenkurven $y = \max(0, (1-x^2)^k$ zur Normalverteilung mit Hilfe der Funktionen `qqplot` und `normplot`

```
function glockenqq(kexp)
% glockenqq qq- und normplot von Glockenkurven
xgl = -1:0.1:1;
ygl = (1-xgl.^2).^kexp;
ycnt = floor(ygl*1000);
% Umsetzen in 1-dim Vektor mit Hfg prop zu ygl
xv = [];
for k = 1: length(xgl)
    xv = [xv xgl(k)*ones(1,ycnt(k)) ]
end
figure(1)
qqplot(xv)
figure(2)
normplot(xv)
end
```

80–8 Daten in Klasseneinteilung

Erzeugen Sie mit `randn` einen Datensatz mit der Normalverteilung $N(x, 5, 2)$ und 1000 Punkten. Ordnen Sie diese in Klassen der Breite 0.25 ein (binning). Berechnen Sie $< x >$ und s für die Verteilung der Klassenhäufigkeiten. Bestimmen Sie auch s_k mit der Sheppard'schen Korrektur $s_k^2 = s^2 - b^2/12$.

```
% Skript binningshow
  x = 5+2*randn(1,1000);  br = 0.25;
% ev auch groessere br
  [klasscent,klasshf] = klasshfg(x,5,br);
  bar(klasscent,klasshf)
% Umsetzen in 1-dim Vektor mit Hfg prop zu ybi
  xv = [];
  for k = 1: length(klasscent)
    xv = [xv klasscent(k)*ones(1,klasshf(k)) ];
  end
%  mu und sd der Klassendaten mit Normfit berechnen
```

```
[muv,sdv]=normfit(xv);
sshep = sqrt(sdv^2 - br^2/12);
[mu,sd]=normfit(x);
disp(sdv, sshep, sd)
```

Die dabei verwendete Binning-Funktion wird hier nochmals aufgeführt.

```
function  [klasscent,klasshf] = klasshfg(dvec,midval,klbr)
% function  [klasscent,klasshf] = klasshfg(dvec,midval,klbr)
%  Haeufigkeitsbestimmung in Klassen gleicher Breite
%  dvec = Beobachtungsvektor,
%  midval = ein Klassen-Zentralwert; klbr = Klassenbreite
%  klasscent = Vektor der Klassen-Zentralwerte
%  klasshf   = Haeufigkeit der einzelnen Klassen
  dvs = sort(dvec); ndat = length(dvs);
  nklu = ceil((midval-klbr/2 -dvs(1))/klbr);
  nklo = ceil((dvs(end) - midval-klbr/2)/klbr);
  klasscent = (-nklu:nklo)*klbr + midval;
  klasshf = zeros(1,length(klasscent));
  ind = 1; k = 1;
  while k <= ndat
      if dvs(k) < klasscent(ind)+klbr/2
          klasshf(ind) = klasshf(ind)+1;
          k = k+1;
      else
          ind = ind+1;
      end
  end
```

Wahrscheinlichkeitsrechnung

80–9 Buchstaben-Anordnungen

Auf wieviele Arten können die Buchstaben des Wortes 'MAMMAMIA' angeordnet werden?

`4*'M', 3*'A', 1*'I', total 8: Pw = 8!/(4!*3!) = 280`

80–10 Gruppeneinteilung

Auf wieviele Arten können aus 20 Personen vier Fünfergruppen zusammengestellt werden?

1. Gr. 5 aus 20, 2. Gr. 5 aus 15, 3. Gr. 5 aus 10

`nchoosek(20,5) = 15504, nchoosek(15,5) =  3003`

`nchoosek(10,5) = 252, total  = 11'732'745'024`

Andere Betrachtungweise mit demselben Resultat – Die Aufstellung auf die 20 Plätze in allen 5 Gruppem ergibt `P20 = 20!`. In jeder der 4 Gruppen ist die Anordung auf die 5 Plätze unwichtig, also muss durch `(5!)^4` dividiert werden.

80–11 Kursteilnehmer-Zuordnung
Auf wieviele Arten können 15 Kursteilnehmer in Fünfergruppen aufgeteilt und drei Kursleitern zugeordnet werden.

```
nchoosek(15,5) =  3003, nchoosek(10,5) = 252,
total  = 756756  * 6  = 4540536
```

80–12 Halbzeitresultate
Wieviele verschiedene Halbzeitresultate sind bei einem Fussballspiel möglich, das 5:6 endete?
Die Pause kann vor, zwischen oder am Ende jeder Torserie gewesen sein, das ergibt 6*7 = 42 theoretisch mögliche Halbzeitresultate.

80–13 Tanzpartner
In einem Tanzkurs erscheinen an einem Abend 5 Herren und 7 Damen, so dass eine Dame den Herrenpart übernehmen muss. Wieviele Möglichkeiten von 6 Tanzpaaren auf der Tanzfläche sind denkbar?
Die Anzahl errechnet sich aus der Anzahl Möglichkeiten, wie die 5 Herren aus 7 Damen 5 auswählen können. also Variationen von 7 Elementen zur 5. Klasse $= 7!/2! = 2520$. Die zwei übrigbleibenden Damen bilden dann das Tanzpaar mit dem unechten männlichen Parter. Falls unterschieden wird, welche der zwei übrigbleibenden Damen den Herrenpart übernimmt, verdoppelt sich die Zahl.

80–14 Seltene Münzwurf-Fälle
Bei 15 Münzwürfen nacheinander wird gesucht, wie viel häufiger die Resultate 1 mal Kopf, bzw. zweimal Kopf und sonst immer Zahl sind als das spezielle Resultat 15 mal Zahl.

```
nchoosek(15,1) = 15; nchoosek(15,2) = 105
```

80–15 Telefonnummern
Wieviele fünfstellige Telefonnummern können in einem Kreis mit derselben Vorwahl vergeben werden, wenn man berücksichtigt, dass für die erste Ziffer 0 und 1 ausfallen.
$8 * 10^4$

80–16 Augensumme bei zwei Würfeln
Wieviele Möglichkeiten gibt es, mit 3 Würfeln die Augensumme 10 zu erhalten?
Mit dem Skript dreiwuerfel.m

```
valvec =
  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
hfg =
  1  3  6 10 15 21 25 27 27 25 21 15 10  6  3  1
hfg zu valvec = 10  ist 27
```

80–17 Fünf Würfel
Beim Würfelspiel "Yahtzee" werden 5 ununterscheidbare Würfel gleichzeitig gewor-

fen. Bestimmen Sie die Wahrscheinlichkeiten, dass beim ersten Wurf a) drei gleiche Augenzahlen oben liegen; b) vier gleiche Augenzahlen oben liegen; c) ein "full house" vorliegt, also drei und zwei zueinander passende Augenzahlen erscheinen. (Die vollständige Diskussion dieses Spiels wird durch die Möglichkeit zum "Verbessern" wesentlich komplizierter. Zweimal dürfen beliebig auswählbare Würfel nochmals geworfen werden.)
Es soll hier eine konstruktive, durch ungeübte Kombinatoriker nachvollziehbare Herleitung gezeigt werden, nicht nur die Formel angeben, bei der man nicht klar weiss, warum gerade diese gilt, und dann die Zahlen einsetzen.
Die Lösung lässt sich am einfachsten von oben her aufbauen:
Mit Berücksichtigung der Reihenfolge gibt es an jedem der 5 Plätze 6 Möglichkeiten, also total $6^5 = 7776$ mögliche Fälle.
Davon besteht eine einzige aus fünf Einsen. Die Wahrscheinlichkeit für 5 * '1' ist also $1/7776$.
Weil die Auswahl, welche Augenzahl fünfmal vorkommen soll, beliebig ist, ist die Wahrscheinlichkeit für fünf gleiche Augenzahlen 6 mal so gross, also $6/7776$.
Bei 4 Einsen ist ein Platz für die 5 anderen Augenzahlen frei. Die verschiedenen Anordnungen von 4*1 und 1*A sind als Permutation mit Wiederholung $Pw5_{4,1} = 5!/(4! \cdot 1!) = 5$ bezifferbar. (Das 'A', die andere Zahl, kann vor, in den 3 Mittelplätzen oder am Schluss stehen, was auch 5 Möglichkeiten ergibt.)
Die Wahrscheinlichkeit für 4 * '1' ist also $5 * 5/7776 = 25/7776$.
Damit ist die Wahrscheinlickeit für 4 gleiche Augenzahlen b) $6 * 25/7776 = 150/7776$.
Die verschiedenen geordneten Möglichkeiten 111AB für 3 Einsen und zwei andere, unter sich verschiedene mir $A < B$, ereben sich aus der Häufigkeit, 2 verschiedene Zahlen aus den 5 Nicht-Einsen auszuwählen $K5_2 = 5!/(3! \cdot 2!) = 5 \cdot 4/2 = 10$
Die Anordnungen von 3 Einsen und 2 weiteren Zahlen folgen der Permutation mit Wiederholung $Pw5_{3,1,1} = 5!/(3! \cdot 1! \cdot 1!) = 5 \cdot 4 = 20$
Die Wahrscheinlichkeit für 3*'1' ist somit $10 * 20/7776 = 200/7776$.
Damit ist die Wahrscheinlichkeit für 3 gleiche Augenzahlen a) $6 * 10 * 20/7776 = 1200/7776$.
Nun sind wir auch bereit, für die Berechnung der Wahrscheinlichkeit für ein 'full house'.
Wir starten wieder mit 3*'1', hängen aber 2 gleiche an: '111AA'. Das gibt 5 Möglichkeiten für die Auswahl der anderen Zahl 'A'. Von dieser Auswahl ergeben sich wieder die verschiedenen Anordnungen durch Permutationen mit Wiederholung: $Pw5_{3,2} = 5!/(3! \cdot 2!) = 5 \cdot 2 = 10$
Die Wahrscheinlichkeit für 3*'1' und 2 unter sich gleichen Zahlen ist somit $5 * 10/7776 = 50/7776$.
Wieder ist die Wahrscheinlichkeit für drei beliebige Augenzahlen und zwei andere unter sich gleiche, also für ein 'full house' c) sechsmal so gross $6 * 5 * 10/7776 = 300/7776$.
Die obigen Resultate wurden hier ausführlicher erarbeitet, aber ergeben dieselben Zahlen wie in der homepage für Stochastik-Formeln von Dr. Werner Brefeld (http://www.brefeld.homepage.t-online.de/stochastik-formeln.html). Dort wer-

den aber noch viele andere interessante mathematische Tatsachen für Nicht-Mathematiker aufbereitet.

Effekt der grossen Zahlen

80–18 Statistische Bestimmung der Zahl Pi

```
% Skript pirand
%n = 1000;
%n = 10000;
n = 100000;
for rep = 1:20
  x = rand(1,n); y = rand(1,n);
  sel = (x.^2+y.^2) < 1;
  piapprox = sum(sel)/n*4
end
```

80–19 Annäherung an den Durchschitt

```
function wav = wdurchschnitt(n)
% function wav = wdurchschnitt(n)
wvec = wuerfel(1,n);
wav = cumsum(wvec)./(1:n);
  function wz = wuerfel(nz,ns)
  %  function wz = wuerfel(nz,ns)  simuliert einen Wuerfel
  %    liefert gleichverteilte ganze Zahlen zwischen 1 und 6
  %    in einer wz(nz,ns)-Matrix; benutzt rand(nz,ns)
    wz = ceil(6*rand(nz,ns));
  end
end
% Anschliessend plot(wav-3.5)
```

80–20 Binomialverteilung zu grossen Zahlen

Erzeugen Sie mit `y = pdf('binomial',x,n,0.5)` mehrere Binomialverteilungen zu Versuchszahlen $n = 10,\ 20,\ 40,\ 80$, alle mit dem Parameter $p = 1/2$. Vergleichen Sie die Resultate jeweils mit der Normalverteilung.
Verschiedene Werte von `n` sind einfach als Parameter in der untenstehenden Funktion `binomnorm` einsetzbar:

```
function binomnorm(npt)
% binomnorm Vergleich Binom- mit Normalverteilung
xbi = 0:npt; xnor = -5:0.1:npt+5;
ybi = pdf('binomial',xbi,npt,0.5);
bar(xbi,ybi,'g')
% Umsetzen in 1-dim Vektor mit Hfg prop zu ybi
xv = [];  ycnt = floor(ybi*10000);
for k = 1: length(xbi)
```

```
    xv = [xv xbi(k)*ones(1,ycnt(k)) ];
end
%  mu und sd mit Normfit berechnen
[muv,sdv]=normfit(xv)
% Alternative mit normparf
 [me,fs] = normparf(xbi,ybi)
 ynor = normpdf(xnor,me,fs);
 hold on
 plot(xnor,ynor,'k','linewidth',2)
 hold off
   function [fctmean,fctstd] = normparf(xv,yv)
     % Normparameter einer Dichtefunktion
     m1  = sum(xv.*yv); w = sum(yv);
     fctmean = m1/w;
     fctstd = sqrt(sum((xv-fctmean).^2.*yv)/w);
   end
end
```

Statistische Verteilungen

80–21 Abfüllmaschine

Eine Abfüllmaschine für 1 kg-Säcke Mehl weist einen normalverteilten Fehler mit einer Streuung von 4 Gramm auf. Wie hoch muss man den Mittelwert vorwählen, damit weniger als 1% der abgefüllten Säcke einen Inhalt von weniger als 995 Gramm aufweisen. (Zuviel ist erlaubt, da reklamiert der Kunde nicht.)
Mit `icdf('norm', 0.01, 0,4)` (inverse cumulative distribution) erhält man das Resultat, dass der 1% Wert für eine Streuung von 4 Gramm bei -9.305 Gramm liegt. Also muss der Mittelwert auf den Wert $995+9.305 = 1004.305$ gesetzt werden, um die Forderung zu erfüllen.

18.2 Miniprojekt zur Wahrscheinlichkeitsrechnung

801 Ziehungen ohne Zurücklegen

```
% Skript bayestrsk
nfarb = 3;
farb(1:nfarb) = [3 2 1]
nziehg = 2;
%
remfarb = farb;
fatot = sum(remfarb);
totknoten = 1; knoten = [ 1, 1, zeros(1,nfarb)];
kncount(1) = 0; kncount(2) = 1;
for zi  = 1:nziehg
```

```
  for knakt = kncount(zi)+1:kncount(zi+1)
    remfarb = farb - knoten(knakt,3:end);
    fatot = sum(remfarb);
    kndat = [zi, knakt, remfarb];
    for fa = 1:nfarb
      if remfarb(fa) > 0
        totknoten = totknoten + 1;
        kncod = knoten(knakt,3:end);
        kncod(fa) = kncod(fa) + 1;
        wzhl = knoten(knakt,1) *remfarb(fa);
        wnn = knoten(knakt,2) * fatot;
        knlin =  [zi,knakt,totknoten , wzhl, wnn, kncod];
rslin = sprintf('  %2d %4d %5d   %3d / %5d     %d%d%d%d%d%d ',...
       knlin)
        knoten = [knoten; knlin(4:end)];
      end
    end
  end
  kncount(zi+2) = totknoten;
  levres = [zi,kncount(zi+1)+1,totknoten]
end
for zit = 1:nziehg
   il = kncount(zit+1)+1; ih = kncount(zit+2);
   wtlev = sum(knoten(il:ih,1)./knoten(il:ih,2))
```

18.3
Lösungshinweise zum Selbsttest Kapitel 8

Testserie 8.1

T811) **Verständnisfragen**

– Durch Bilden des Integrals über den Bereich der x-Werte zur gesuchten Wahrscheinlichkeit.

– $6!/5! = 6$ Das B kann zuvorderst, zuhinterst oder an den vier Plätzen zwischen den A stehen. – Plus und Minus 1.5*iqd = 3 *iqd.

– 68.3 %

T812) Bestimmen Sie die Wahrscheinlichkeiten für 0 bis 5 gehaltene Tore bei 5-maligen Elfmeterschiessen falls der Torhüter im Mittel $1/10$ der Schüsse hält. Vergewissern Sis sich, dass die Summe aller Wahrscheinlichkeiten 1 ergibt.

```
wk = nchoosek(5,k)*0.9^(5-k) * 0.1^k k=0..5
```

$wk = 0.5905, 0.3281, 0.0729, 0.0081, 4.5000e-04, 1.0000e-05$

T813)
```
for k = 0:30
  hf(k+1) = nchoosek(30,k)*(0.5)^30;
end
plot(hf)
```

T814) Vor Beginn der Verlängerung sind bei 0:0 1, bei 1:1 4, bei 2:2 9, bei 3:3 16 Halbzeitresultate möglich. Das gibt total 30 Möglichkeiten.

Alle anderen Tore fallen jeweils nach der Verlängerung und ergeben keine Zusatzmöglichkeiten.

T815)
```
r 1/4    rr  (1/4)*(1/3) = 1/12
r 1/4    rw  (1/4)*(2/3) = 1/6
w 1/4    wr  (1/4)*(2/3) = 1/6
w 1/4    ww  (1/4)*(2/3) = 1/12
```

Teil III
M-File Sammlung

20
Allgemeine M-Files

Bei diesen kleinen Helfern kann es sich lohnen, sie in einem speziellen Pfad abzuspeichern und diesen in die Liste der von MATLAB durchsuchten Dateipfade einzufügen.

20.0.1
Spezielle Funktionen

Eine Serie von simplen, aber manchmal praktischen Funktionen sind die Winkelfunktionen und ihre Inversen, welche den Winkel in Grad verarbeiten statt in Radian wie die offiziellen mathematische Funktionen. Inzwischen sind diese Funktionen durch Anhängen eines 'd' wie in `sind()` in der MATLAB Bibliothek.
Einzige Ausnahme bildet atan2; für diese Funktion kann man die unten angeführte einsetzen:

```
% degatan2 arcus-tangens-2 Funktion mit Grad-Rueckgabe
%  wird mit 2 Parametern, dem  Zähler (enu) delta-y und
%           und  dem Nenner (deno) delta-x
%  aufgerufen und liefert Winkel im Bereich 0 bis 360 Grad
function winbk = degatan2(enu,deno)
  winbk  = 180/pi*atan2(enu,deno);
end %function
```

20.0.2
Matrizenoperationen

Die allgemeine Indexwertmatrix-Funktion `indmatf.m` wurde schon bei den M-Files zum Kapitel 4 erwähnt. Sie wird hier nochmals aufgeführt.

```
function IM = indmatf(varargin)
% indmatf.m indmatf erzeugt eine Indexwertmatrix
% Aufruf: IM=indmatf   ergibt 3x3
%         IM=indmatf(ndim) ndim x ndim
%         IM=indmatf(nzei,nspa) nzei x nspa
if nargin == 0
% Fall ohne Parameter gibt 3x3
```

```
   nzei = 3;
   nspa = 3;
elseif nargin == 1
% Fall mit einem Parameter quadratisch
%   Beachte: varargin ist "cell array"
%   und wird mit {ind} indiziert
  nzei = varargin{1}
  nspa = nzei
else
% Fall 2 Parameter nzei x nspa
  nzei = varargin{1}
  nspa = varargin{2}
end
%  Doch noch die Matrix definieren
for zei = 1:nzei
  for spa = 1:nspa
    IM(zei,spa) = 10*zei + spa;
  end
end
```

Zwei weitere praktische Funktionen `getrow` und `getcol` erlauben die Extraktion einer einzelnen Zeile bzw. einer einzelnen Spalte aus einer Matrix:

```
% zeivec = getrow(M,nzei) Extraktion einer Zeile
function zeivec = getrow(M,nzei)
zeivec = M(nzei,:);

% spavec = getcol(M,nspa) Extraktion einer Spalte
function vbk = getcol(M,nspa)
spavec = M(:,nspa);
```

Die Funktion `randsummen` liefert einen Spalten- und einen Zeilenvektor

```
%[spav,zeiv]=randsummen(M) - Zeilen- und Spaltensummen
function [spav,zeiv]=randsummen(M)
spav = sum(M.').';
zeiv = sum(M);
```

21
M-Files zu Kapitel 1

21.1
M-Files: Bekanntschaft schließen mit MATLAB

Tee-Mug teemug.m

```
%  Volumenberechnung Tee-Mug
D = 7.9 ;
H = 9.1 ;
Wand = 0.3; Boden = 0.8;
Fuellgrad = 0.85;
Vol = (D-2*Wand)^2 *pi/4 * (H-Boden) * Fuellgrad
```

Fluzzeit des Sonnenlichtes `sonnenlicht.m`

```
  150E6 / 300e3
%   ans =
%              500.0000}   % Resultat in Sekunden
  tlichtmin = ans/60       % gelaeufiger in Minuten
%  tlichtmin =
%                8.3333
```

Berechung der mittleren Dichte der Erde `erddichte.m`

```
  Erad = 6371*1000 ;   % inkl. Umwandlung km in m
  Emass = 5.9736e24;   % in kg
  Edicht = Emass/(4*pi/3*Erad\^{}3)}\\
% Edicht =
%            5.5147e+03  %   in kg/m^3
%                        % in bekannten Einheiten 5.5 kg/Liter
```

Volumen eines Tee-Mugs `mugvolumen.m`

```
%  Volumenberechnung Tee-Mug
D = 7.9 ;
H = 9.1 ;
```

```
Wand = 0.3; Boden = 0.8;
Fuellgrad = 0.85;
Vol = (D-2*Wand)^2 *pi/4 * (H-Boden) * Fuellgrad
%     Vol =
%            295.2790
```

Binomische Formeln im symbolischen Modus `binomsymb.m`

```
% Deklaration von s und t als symbolische Variablen
syms s t
expand( (s-t)^2)
% ans = s^2 - 2*s*t + t^2
expand( (s-t)^3)
% ans = s^3 - 3*s^2*t + 3*s*t^2 - t^3
expand( (s-t)^4)
% ans = s^4 - 4*s^3*t + 6*s^2*t^2 - 4*s*t^3 + t^4
expand(s-t)^50
```

Binomische Formeln im Berechnungs-Modus `binomcalc.m`

```
a = 10 , b = 2
p3 = (a-b)^3
q3 = a^3  - 3*a^2*b + 3*a*b^2  - b^3
p4 =   (a-b)^4
q4 =  a^4  - 4*a^3*b +    6*a^2*b^2    -4*a*b^3+ b^4
p5 =   (a-b)^5
q5 = a^5 -5*a^4*b + 10*a^3*b^2 -10*a^2*b^3 +5*a*b^4 -b^5
```

Horner-Schema im symbolischen Modus `hornersymb.m`

```
syms  q4 q3 q2 q1 q0 x
P4 = q4*x^4 +  q3*x^3 + q2*x^2 + q1*x +  q0
P4H = (((q4*x + q3)*x + q2)*x + q1)*x + q0
P4Hf = horner(P4)
% P4Hf =
% q0 + x*(q1 + x*(q2 + x*(q3 + q4*x)))
```

Funktion zur Polynom-Auswertung mit dem Horner-Schema im Berechnungs-Modus `hornereval.m`

```
function pv = hornereval(cv, xv)
% function pv = hornereval(cv, xv)  Polynom-Evaluation
%   cv = Koeffizienten-Folge, absteigend,
%   xv = auszuwertende x-Positionen (Vektoren erlaubt)
nkoef = length(cv);
pv = cv(1);     % Start der Polynomwert(e)
for cnum = 2:nkoef  % in jedem Schritt: (..)*xv + cv(cnum)
   pv = pv.*xv + cv(cnum); % '.*' erlaubt xv-Vektoren
end
```

Polynom-Plot mit feinem und grobem x-Raster usehorner.m

```
% Skript usehorner.m
%  Polynom-Plot mit feinem und grobem x-Raster
xpt = -3:3; xfin = -3.5:0.1:3.5;   % x grob / fein
ppt = hornereval(cf,xpt)    % Funktionswerte zu xpt
pfin = hornereval(cf,xfin); % Funktionswerte zu xfin
%  grafische Darstellung
plot(xpt,ppt,'+') ; hold on ;
plot(xfin,pfin) ; grid on; hold off;
```

Beispiele für fprintf und sprintf

```
%  Skript fsprintf.m : Demonstration der printf-Formatierung
%  fprintf(fhdl,' Matrixdimensionen: %d x %d \n',...
%      nzeilen,nspalten)
  txlin = sprintf(' Matrixdimensionen: %d x %d \n', 4,6)
%     txlin =
%         ' Matrixdimensionen: 4 x 6 '
% fprintf(fhdl,'Gleitkomma-  %15.6f und Exponentialform %15.8e \n',...
%    wert, wert )
  txl2 =  sprintf('Gleitkomma-  %15.6f und Exponentialform %15.8e \n',...
    1835.217, 1835.217 )
% txl2 =
%      'Gleitkomma-      1835.217000 und Exponentialform 1.83521700e+03'
 mdz = sfprintf(' M = [ %8.4f %8.4f; %8.4f %8.4f ] \n',11,12, 21,22)
% mdz =
%       'M = [  11.0000  12.0000;  21.0000  22.0000 ]'
```

21.2
M-Files: Grundlagen der Matrizenrechnung

Beispiel zum Transponieren

```
% Skript transposeexa.m
M = [11 12 13 14;  21 22 23 24; 31 32 33 34]
Mt = [11 21 31; 12 22 23; 13 23 33; 14 24 34 ]
M'
Mt'
bZ = M - Mt'
hZ = Mt - M'
```

Matrizenprodukt, aufgelöst in einzelne Produkte von je zwei Vektoren

```
function  C = matmulvec(A,B)
% function  C = matmulvec(A,B)
% Demonstrations Matrix-Multiplikation,
% bei der die Elemente je als Produkt eines
%  Zeilen- mal eines Spaltenvektors entstehen
```

```
  [n1,m1] = size(A); [n2,m2] = size(B);
  if m1 == n2
    for ro = 1:n1
      for co = 1:m2
        C(ro,co) = getrow(A,ro)*getcol(B,co);
      end
    end
  else
    disp('MATMULVEC: innere Diensionen verschieden')
    C= zeros(n1,m2);
  end
  function selrow = getrow(M,nro)
    selrow = M(nro,:);
  end
  function selcol = getcol(M,ncol)
    selcol = M(:,ncol);
  end
end
```

21.3 M-Files: Matrizenrechnung mit MATLAB

Demonstration Rechtecksmatrix mal ihre Transponierte

```
% Skript recttransp.m M"M' und M'*M immer möglich
  M = [ 1 2 3 4 5 ; 1 0 2 0 3]
  M * M'
  M'*M
```

Suchen des 2x2 Additions-Neutralelementes im symbolischen Modus

```
% Skript symbaaddmat.m
syms a b c d     % fuer symbolische 2x2 Matrix
S = [a b ; c d] % Definition der symbolischen Matrix
Na = zeros(2)    % Nullmatrix
Sp = S + Na      % Summe S + Nullmatrix gibt wieder S
Z = rand(2)
Zp = Z + Na      % Summe Z + Nullmatrix gibt wieder Z
ndim = 5         % Waehle irgend eine Dimensionszahl
M = rand(ndim)
Mp = M + zeros(ndim) % Summe M + zeros(ndim) = M
```

Das Neutralelement der Multiplikation ist immer die Einheitsmatrix, bei Rechtecksmatrizen hat sie aber von links und von rechts verschiedene Dimensionen.

```
% Skript lrneutrect.m
```

```
M = [ 1 2 3 4 5 ; 1 0 2 0 3]
IdL = eye(2)
IdR = eye(5)
MtL = IdL * M
MtR = M * IdR
```

Einfache Schleife für Diagonalmatrix

```
% Skript diagfillones.m
%  Einheitsmatrix selbst abfuellen
  ndim = 6
  I = zeros(ndim);
  for k=1:ndim
    I(k,k) = 1;
  end
I   % Kontrollausgabe
```

Doppelschleife zum Füllen einer oberen Dreiecksmatrix mit Einsen

```
%  Skript rfillones
R = zeros(ndim);
% der erste (Zeilen-)Index muss von 1 bis ndim
%    alle Zeilen durchlaufen
for zeil = 1:ndim
%   der Spaltenindex beginnt jeweils erst in der Diagonalen
%   das entspricht in der zeil-ten Zeile dem Wert 'zeil';
%   von dort laeuft er bis zu 'ndim', dem rechten Rand.
   for spal = zeil:ndim
      R(zeil,spal) = 1;
   end
end
R  % Kontrollausgabe nach erfuellter Aufgabe
```

Hilfsmatrix Indexwertmatrix, mit Werten, welche die Indizes anzeigen

```
function iwm = indwertmatgen(nzei, nspa)
%  function iwm = indwertmatgen(nzei, nspa)
%  Allgemeine Indexwertmatrix
  iwm = [];
  for ze = 1:min(nzei,9)  % Absicherung gegen nzei > 9
    iwm = [iwm ; 10*ze+(1:min(nspa,9)) ];
  end
end
```

21.4
M-Files: Schritte zum eigenen Programm

Geraden- und Polynomfit mit zugehöriger Grafik

```
%%  Geraden- und Polynomfit
%% synthetische Punkte aus sinus
x = (-1:0.1:1)*pi/2;
y = sin(x);
%% zeichne x,y in leere Figur 1, Bild bleibt stehen
figure(1); clf; plot(x,y,'o-'); hold on
format compact    % Ausgabe eng auf Bildschirm
%% Geradenfit und Polynomfit 3. Grades
pc1 = polyfit(x,y,1)   % Fit 1. Grades = Gerade
yg1 = polyval(pc1,x) ;  plot(x,yg1,'k'); % Farbe blac'k'
pc3 = polyfit(x,y,3)    % fit 3. Grades
yp3 = polyval(pc3,x) ;  plot(x,yp3,'r'); % Farbe 'r'ed
hold off
```

Berechungen von Oberfläche und Volumen einer Kugel

```
  function [oberfl, vol] = Kugeldaten(R)
%  function [oberfl, vol] = Kugeldaten(R)
% Diese ersten Kommentarzeilen erscheinen bei 'help Kugeldaten'
% Beispiel einer Funktion mit 2 Rueckgabe-Parametern
% function [oberfl,vol] = Kugeldaten(R)
% So kann man die Signatur kurz ueberpruefen, weiss also wie
% Aufrufen, und welche Ein- Ausgabe-Parameter einsetzen
    oberfl = 4*pi*R.^2;
    vol = 4*pi*R.^3/3;
  end  % end zu 'function' fakultativ aber empfohlen
```


```
%  function [oberfl, vol] = Kugeldaten(R)
% Diese ersten Kommentarzeilen erscheinen bei 'help Kugeldaten'
% Beispiel einer Funktion mit 2 Rueckgabe-Parametern
% function [oberfl,vol] = Kugeldaten(R)
% So kann man die Signatur kurz ueberpruefen, weiss also wie
% Aufrufen, und welche Ein- Ausgabe-Parameter einsetzen
    oberfl = 4*pi*R.^2;
    vol = 4*pi*R.^3/3;
  end  % end zu 'function' fakultativ aber empfohlen
```


Skript im symbolischen Modus zum Erarbeiten der Formel für das Teilvolumrn eines Kugelbehälters in Abhängigkeit von der Füllhöhe.

```
%% symbolisches Skript teilvolformel.m
  % zum Erarbeiten der Formel Teilvolumen(Fluessigkeitspegel)
syms R  z  h % Kugelradius, Hoehe der Schnittebene, Pegel
%% A) mit bestimmtem Integral
  rho = sqrt(R^2 - (R-z)^2)  % Kreisradius in Schnittebene
  %  bestimmtes Integral ueber Kreisflaeche(z)*dz, von 0 bis h
  tvol = int(rho^2*pi,0,h)   % Formel fuer das Teilvolumen(h)
%% B) stereometrische Herleitung
  okseg = 2*pi*R*h           % Oberflaeche Kugelsegment
  % Kugelsektor-Volumen = Oberflaeche * R/3
  ksekt = okseg*R/3
  % ergaenzender Kegel um Kugelsegment zu erhalten
  % Radius bei Schnitthoehe h
  rhoh = subs(rho, z, h)     % ersetze z durch h in rho(z)
  keg = pi*rhoh^2 * (h-R)/3
  %  Kugelsegment = Kugelsektor + Kegel
  tvolster = ksekt + keg
  tvssimp = simplify(tvolster)% Vereinfachen zum Vergleich
%% Test fuer Halbkugel und volle Kugel h=R und h=2*R
  halbvol = subs(tvol, h, R)  , gesvol = subs(tvol, h, 2*R)
%% Formel-Resultat
%  tvssimp =
%  (pi*h^2*(3*R - h))/3
```

Verwenden der Formel in einer MATLAB Funktion Volumen als Funktiion der Füllhöhe (und des Kugelradius)

```
function [ teilvol ] = kugelteilvol( R, h )
%  function [ teilvol ] = kugelteilvol( R, h )
%  berechnet das Teilvolumen der Fluessigkeit in
%  einer Kugel mit Radius R unterhalb des Pegels h
teilvol = pi*h.^2.*(3*R - h)/3;
%  Zusatz zur robusten Programmierung: Melden von
%   sinnlosen h-Werten ausserhalb 0 < h < 2*R
   if min(h) < 0 | max(h) > 2*R
     warning('Aufruf kugelteilvol mit h ausserhalb Bereich')
   end
end
```

Anwenden der Funktion und Funktionsgrafik

```
% Skript kugelbehaelter.m zeigt Volumen(Fuellhoehe)
% R, h in Dezimetern ergibt Volumen in Litern
  R = 7.5
  hvec = (0:0.02:2)*R;
  teilvolvec  = kugelteilvol( R, hvec );
  plot(hvec, teilvolvec)
```

Rekursive Funktion zum Berechnen des Fakultät-Wertes

```
%fakurekur.m, Aufruf: nfakultaet = fakurekur(n)
% rekursive Funktion zum Berechnen der Fakultaet
function fw = fakurekur(n)
if n <= 1
  fw = 1;
else
  fw = n * fakurekur(n-1);
end
```

Rekursive Lösung des Spiels der Türme von Hanoi

```
% hanoireku.m
% Uebung der Tuerme von Hanoi
% Aufruf-Beispiel hanoireku(4,'A','B','C')
function hanoireku(n,  von , via, nach)
format compact
if n > 1
  hanoireku(n-1,  von, nach, via);
    disp(sprintf('Scheibe %d von %s nach %s ', n, von, nach ));
  hanoireku(n-1,  via, von, nach);
else
  disp(sprintf('Scheibe %d von %s nach %s ', 1, von, nach ));
end
```

Das kleine Beispiel zun Objekt Orientierten Programmieren mit der Klasse ooblume

```
classdef ooblume < handle
    % Grafik einer Blume in objektorientierter (OO) Syntax
    %   Konstruktor:  blobjnam = ooblume(xpos, ypos, blgroesse)
    %   Die meisten Parameter sind als Default vorgegeben, mit
    %   blobjnam.propnam = ... koennen sie aber veraendert werden
    properties    % allgemein zugaengliche interne Daten
        farbe = 'y';
        liniendicke = 3;
        xpos = 0;   ypos = 0;
        stiellg = 4; petallg = 1;
        numpetal = 21; petalhw = 10;
    end
    properties (SetAccess = private, GetAccess = private)
        linhdls;   % abgeschirmte interne Daten
    end
    methods
        function blobj = ooblume(xposi,yposi,farbi)
          blobj.xpos  =  xposi;
          blobj.ypos = yposi;
          if nargin >2 blobj.farbe = farbi; end
        end
```

```
        function  zeichne(blobj)
           [petalx, petaly] = makepetal(blobj);
           try   blobj.radiere;  end  % loeschen falls vorhanden
           blobj.linhdls = [];
           stielhd = plot([blobj.xpos blobj.xpos],...
               [blobj.ypos blobj.ypos+blobj.stiellg],'g',...
                 'linewidth',blobj.liniendicke);
           blobj.linhdls = [blobj.linhdls ; stielhd];
%
           blthd=plot(petalx, petaly,'color',blobj.farbe,...
                  'linewidth',blobj.liniendicke);
           blobj.linhdls = [blobj.linhdls ; blthd];
        end    % end function zeichne
        function radiere(blobj)
            delete(blobj.linhdls)
        end
        function delete(blobj)
            radiere(blobj)
        end
    end   % end methods
end   % end class ooblume
function [petx,pety] = makepetal(blobj) % rein interne Funktion
  for k = 1:blobj.numpetal
    dw = 2*blobj.petalhw/(blobj.numpetal-1);
    w = 90-blobj.petalhw + dw*(k-1);
    petx(:,k) = [0 ; cosd(w)*blobj.petallg];
    pety(:,k) = [0 ; sind(w)*blobj.petallg];
  end
  petx = petx+blobj.xpos;
  pety = pety+blobj.ypos + blobj.stiellg;
end  % function makepetal
```

Die weiter führende Klasse dynablume erbt die meisten Eigenschaften un Methoden von ooblume

```
classdef dynablume < handle & ooblume
    %  Unterklasse von ooblume
    %  waechst und oeffnet sich dynamisch
    properties
       wzeit = 2.5; oeffzeit = 4; petwmax = 120;
    end
    methods
      function dblo = dynablume(xposi,yposi,farbi)
      if nargin < 3  farbj = 'y'; else  farbj = farbi; end
        dblo@ooblume(xposi,yposi,farbj);
      end
      function wachsstart(dblo)
        dblo.petalhw = dblo.petwmax*0.1;
          for k=0:10  % Stiel waechst zuerst
            dblo.stiellg = 2+k*0.2;
            dblo.zeichne;  pause(0.1*dblo.wzeit)
```

```
            end
            for k=0:10  % dann oeffnet sich die Bluete
              dblo.petalhw = dblo.petwmax*(0.1+0.09*k);
              dblo.zeichne;  pause(0.1*dblo.oeffzeit)
            end
        end
      end
end
```

Ein Beispiel zur Anwendung der beiden Klassen

```
figure(1); clf;
axis([-7 7 -1 8]) ; hold on; axis equal; axis off
Bl1 = ooblume(4,0,'r');  Bl1.zeichne ; % Bl1 zeichnen
Bl2 = ooblume(0,1); Bl2.petalhw = 80; Bl2.petallg = 0.4;
Bl2.farbe = [1 0.6 0] ; Bl2.zeichne;   % Bl2 zeichnen
Bl3 = ooblume(-1,0); Bl3.petalhw = 180; Bl3.stiellg = 3;
Bl3.zeichne ; pause(1.8)               % Bl3 zeichnen
Bl1.petalhw = 100;  Bl1.zeichne; pause(1.8) % Bl1 modifiziert
Wbl = dynablume(-4,0.2,'m'); Wbl.petwmax = 180; Wbl.wachsstart;
Vbl = dynablume(1.5,0.1); Vbl.petwmax = 90; Vbl.wachsstart;
```

21.5 M-Files: einfach grafische Darstellungen

Demonstration der Bedeutung der Abszisssen-Einteilung bei Kurvenplots

```
% Skript grobfeinparabel.m
% mit den Befehlen
x = -5:5
y = x.^2
plot(x,y)
axis([-3 3 0 9])   % ( allg. axis([xmin,xmax,ymin,ymax]) )
%                       nach dem  plot()  Aufruf
%  wird die obige Parabel ein wenig eckig.
disp('<ret> zum Weiterfahren')
pause
% Anhalten der Berechnung bis return gedrueckt wird
%      Abhilfe ist einfach
% (Angabe einer Schrittweite
%    kleiner als Eins in der Aufzaehlung)
xf = -10:0.02:10
yf = xf.^2
plot(xf,yf)
axis([-3 3 0 9])
```

Demonstration des spaltenweisen Plottens bei der Eingabe einer Matrix

```
% Skript matspaltenplot.m
  z1 = 0:0.1:1;    z2 = z1.^2 ;
  z3 = z1 .^3;  z4 = z1.^4;
  P= [ z1 ; z2 ; z3 ];
  figure(1);   plot(P);
  figure(2);  plot(P');
```

Einen Kreis zeichnen als einfachste Lissajous-Figur

```
% Skript kreiswgrad.m
wg = 0:5:360;    % sind, cosd arbeiten mit Grad
si = sind(wg);   co = cosd(wg);
plot(co,si)
axis equal   % Wichtig! sonst entsteht eine Ellipse
```

Polygone und Sterne

```
% Skript genpolygon.m  Sterne und Polygone
  neck = 7  % irgend eine Eckenzahl vorwaehlen
  wini = 90 % Startwinkel fuer erste Ecke
  wp = wini + 0:(360/neck):360;
  xp = cosd(wp) ; yp = sind(xp)
  plot(xp,yp); axis equal  % Polygon im EKreis
  pause   % return druecken zum Weiterfahren
  rs1 = 0.5/cos(360/(neck*2))  % Umkreisradius
  plot(rs1*xp,rs1*yp); axis equal  % Seitenlaenge = 1
  pause   % return druecken zum Weiterfahren
  if floor(neck/2) == neck/2  % Test auf gerade/ungerade
    ws = wini + 0:(720/neck):720;
    xs = cosd(ws) ; ys = sind(ws)
    plot(xs,ys); axis equal  % Stern ungerade
    pause
    fill(xs,ys,[1 1 0])  % Orange zwischen rot und gelb
    axis equal; hold off
  else
    ws1 = wini + 0:(720/neck):360;
    xs1 = cosd(ws1) ; ys1 = sind(ws1)
    plot(xs1,ys2); axis equal  % Stern gerade
    hold on
    ws2 = wini + 360/neck + 0:(720/neck):360;
    xs2 = cosd(ws2) ; ys2 = sind(ws2)
    plot(xs2,ys2);  % Stern gerade 2. Teil
    hold off
    pause
    fill(xs1,ys1,'r')  %  rot
```

```
    hold on
    fill(xs2,ys2,'y')  %  gelb
    axis equal; hold off
  end
```

Die Umrechnungsfunktion der Farbefinitionen von hsv (hue, saturation, value; Farbwinkel, Sättigung, Helligkeit) in die rot, grün, blau Anteile gibt Einblick in das Prinzip der Zusammenstellung aller Farben aus den Grundfarben. Diese funktion ist fast identisch mit der MATLAB Bibliotheksfunktion hsv2rgb, der einzige Unterschied ist der Farbwinkel, der hier mit 0 bis 360 Grad angegeben wird, (statt 0 bis 1).

```
function [ r,g,b ] = hsvtorgb( h,s,v )
% function [ r,g,b ] = hsvtorgb( h,s,v )
%    h=0..360, s,v= 0..1
% r,g,b aus Farbwinkel h(Grad),
% Saettigung s und Helligkeit v
rp=[1 1 0 0 0 1 1]; gp=[0 1 1 1 0 0 0]; bp=[0 0 0 1 1 1 0];
% hi ganzzahlig  1 bis 6, f ist Zwischenwert-Bruchteil
hreal=h/60; hint=floor(hreal);
f=hreal-hint; hi=mod(hint,6) + 1;
% mit h interpolierte rgb Basiswerte
  rb = rp(hi).*(1-f) + rp(hi+1).*f;
  gb = gp(hi).*(1-f) + gp(hi+1).*f;
  bb = bp(hi).*(1-f) + bp(hi+1).*f;
%  Saettigung ergibt Weiss-Sockel vom Wert (1-s)
  rs = s.*rb + (1-s);
  gs = s.*gb + (1-s);  bs = s.*bb + (1-s);
% Helligkeit ergibt allgemeine Reduktion
  r = rs.*v; g = gs.*v; b = bs.*v;
end
```

21.6 M-Files: Befehls-Übersicht

Befehls-Beispiel für das analytische Lösen einer Gleichung

```
syms x y z r a b
rform = sqrt(x^2+y^2+z^2) % Punkt auf Kugel
eq1 = rr == rform % = definiert, == fuer Gleichung
xf = solve(eq1,x) % Formel für x aus r,y,z
% ergibt     xf =  (r^2 - y^2 - z^2)^(1/2)
% 2 Loesungen     -(r^2 - y^2 - z^2)^(1/2)
% Gleichungssystem, Gleichungen direkt eingegeben
sol = solve('y==a*x','y==-x+b',x,y); % 2 Geraden
```

```
solvec = [sol.x, sol.y] % Struktur mit sol.x, sol.y
% Resultat: solvec = [ b/(a + 1), (a*b)/(a + 1)]
```

Befehls-Beispiel für eine Struct-Definition

```
% 3D Achsenkreuz-Dreibein als struct-Variable
drb = struct('name', 'Achsenkreuz-Dreibein');
drb.x = [1 0 0 0 0];
drb.y = [0 0 1 0 0]; drb.z = [0 0 0 0 1];
plot3(drb.x, drb.y, drb.z) % anzeigen, anwenden
```

Als Beipiel für die leicht andere Syntax des symbolischen Modus wird hier das symbolische Skript gezeigt, mit dem durch stufenweise Lösung von bestimmten Integralen das Volumen von mehrdimensionalen Hyperkugeln bestimmt wird.

```
syms x r R L K M H G D n real
s2 =simple( 2*int(sqrt(r^2-x^2), -r, r));
     s3 = simple( int(subs(s2,r,sqrt(R^2-x^2)), -R, R));
     s4 = simple(int(subs(s3,R,sqrt(L^2-x^2)), -L, L));
     s5 = simple(int(subs(s4,L,sqrt(K^2-x^2)), -K, K));
     s6 = simple(int(subs(s5,K,sqrt(M^2-x^2)), -M, M));
     s7 = simple(int(subs(s6,M,sqrt(H^2-x^2)), -H, H));
     s8 = simple(int(subs(s7,H,sqrt(G^2-x^2)), -G, G));
     s9 = simple(int(subs(s8,G,sqrt(D^2-x^2)), -D, D));
s2
%     subs('2*pi^(n/2)/(n/2-1)!*R^n/n',n,2)
 s3
%     subs('2^((n+1)/2)*pi^((n-1)/2)/(n-2)!*R^n/n',n,3)
     subs('R^n/n*2*pi^(n/2)/gamma(n/2)',n,3)
 s4
%     subs('2*pi^(n/2)/(n/2-1)!*R^n/n',n,4)
     subs('R^n/n*2*pi^(n/2)/gamma(n/2)',n,4)
 s5
%     subs('2^((n+1)/2)*pi^((n-1)/2)/(n-2)!*R^n/n',n,5)
     subs('R^n/n*2*pi^(n/2)/gamma(n/2)',n,5)
s6
%     subs('2*pi^(n/2)/(n/2-1)!*R^n/n',n,6)
     subs('R^n/n*2*pi^(n/2)/gamma(n/2)',n,6)
 s7
%     subs('2^((n+1)/2)*pi^((n-1)/2)/(n-2)!*R^n/n',n,7)
     subs('R^n/n*2*pi^(n/2)/gamma(n/2)',n,7)
s8
%     subs('2*pi^(n/2)/(n/2-1)!*R^n/n',n,8)
     subs('R^n/n*2*pi^(n/2)/gamma(n/2)',n,8)
%
vols =     [ s2 s3; s4 s5; s6 s7; s8 s9]
subs('R^n/n*2*pi^(n/2)/gamma(n/2)',n,[2 3; 4 5; 6 7; 8 9])
```

22
M-Files zu Kapitel 2

22.1
M-Files zum Funktionsbegriff

Befehls-Beipiele für die grafische Darstellung einer Ungleichungs-Lösung

```
xv = -10:0.5:10;     % mittelfeiner Vektor um 0
X = ones(41,1)*xv;   % x-Streifenmatrix
Y = xv'*ones(1,41);  % y-Streifenmatrix
R = abs(X) + abs(Y) < 5; % oder beliebige andere Ungleichung
spy(R)               % Anzeige der 'true' Bereiches
```

22.1.1
Spezielle Funktionen

Gerade und ungerade Funktionen

Mit MATLAB läßt sich die Eigenschaft gerade/ungerade Funktion elegant grafisch darstellen, indem zwei aufeinander bezogene Punkte markiert und mit einer Linie verbunden werden.

Das Skript-M-File evenodd.m erwartet ein Paar von vordefinierten Abszissen–Funktionswert-Vektoren in den Variablen 'x' und 'y'. Beim Start des Skriptes wird ein x-Wert 'v' für den Vergleichspunkt abgefragt.

```
%EVENODD Skript m-File zur Demonstration gerader
%    und ungerader Funktionen
% die Vektoren x,y  muessen vorher definiert werden
v = input('Bitte Vergleichs-x-Wert eingeben: ');
[mr,vr] = min(abs(x-v)); [ml,vl] = min(abs(x+v));
mg = min(abs(y(vl)-y(vr))); mu = min(abs(y(vl)+y(vr)));
plot(x,y,'k'); hold on;  axis equal
if mu < mg
  plot([x(vl) 0  x(vr)],  [-y(vr) 0 y(vr)], '-or')
else
  plot([x(vl) 0  x(vr)],  [y(vr)  y(vr) y(vr)], '-og')
end
```

```
hold off
```

22.1.2
Periodische Funktionen

Die Definition einer periodischen Funktion verlangt, daß bei Zunahme des Argumentes um die Periodendauer 'T' wieder derselbe Funktionswert $f(x+T) - f(t)$ erreicht wird. Dies läßt sich mit einer MATLAB-Grafik leicht visualisieren.
Das Skript-M-File periodic.m erwartet ein Paar von vordefinierten Abszissen–Funktionswert-Vektoren in den Variablen 'x' und 'y', sowie einen vordefinierten Wert 'T' für die Periodendauer. Beim Start des Skripts wird ein x-Wert 'v' für den Vergleichspunkt abgefragt.

```
%PERIODIC Skript m-File zur einfachen Demonstration
%  der Periodizitaet
% Vektoren x,y, sowie Periodendauer T
%  muessen vorher definiert werden
v = input('Bitte Vergleichs-x-Wert eingeben: ');
[ml,vl] = min(abs(x-v));
[mr,vr] = min(abs(x-v-T));
plot(x,y,'k');
hold on
plot([x(vl) x(vr)], [y(vl) y(vl)], '-or')
[mx,vx] = min(abs(x-v-2*T));
if mx < 0.1*T
plot([x(vr) x(vx)], [y(vr) y(vx)], ':or')
end
hold off
```

Demonstration einer Funktion mit ihrer inversen Funktion

Das M-file invfctdemo.m zeigt die Exponentialfunktion und die dazu inverse Logarithmusfunktion. Mit der Maus kann ein Punkt angewählt werden, worauf das Paar der einander zugeordneten Punkte markiert wird.

```
%INVFCTDEMO  Demonstration von Funktion und Umkehrfunktion
% am Beipiel von exp und log
xa = -4:0.01:4   ;  ya = exp(xa) ;
xb = 0.01:0.01:4 ;  yb = log(xb) ;
plot(xa,ya,'g') ;
axis ([-4 4 -4 4]) ; axis square ; hold on
plot(xb,yb,'b')
plot([-4 4] ,[-4 4],'r') ;  % Diagonale
plot([0 0], [-4 4],'k') ; plot( [-4 4],[0 0],'k') % Achsen
% Eingabe Punkt auf einer der Kurven
disp('Bitte 5 mal mit dem Cursor ')
disp(' einen Punkt auf einer der Kurven waehlen!')
```

```
for irep = 1:5
  [xcu,ycu] = ginput(1);
  if length(ycu) == 0
    xcu = 2; ycu = 0.8;
  end
  if ycu > xcu
% Exponentialfunktion
    xl(1) = xcu  ;   yl(1) = exp(xcu);
    xl(3) = yl(1);   yl(3) = log(xl(3)) ;
  else
    if xcu > 1
      xl(1) = xcu;      yl(1) = log(xcu);
      xl(3) = yl(1);    yl(3) = exp(xl(3)) ;
    else
      yl(1) = ycu;      xl(1) = exp(ycu);
      xl(3) = yl(1);    yl(3) = exp(xl(3)) ;
    end
  end
  xl(2) =( xl(1) + xl(3))/2;  yl(2) =( yl(1) + yl(3))/2;
  if irep > 1
    delete( hd)
  end
  hd =  plot(xl,yl,'-or');
end
hold off; disp('invfctdemo beendet!')
```

22.2
M-Files zu den Linienplots in MATLAB

Stehende Lissajous-Figuren

Im M-File `paramliss.m` werden die drei Parameter m,n, (aus dem Verhältnis m:n), sowie delta (Phasenverschiebung), die man zur Festlegung einer Lissajous-Figur braucht, nacheinander abgefragt. Damit kann man die Darstellung jeder beliebigen stehenden Lissajous-Figur verlangen.

```
%paramliss Lissajous-Figur mit Parameterabfrage
% Vor dem Start wird clf empfohlen, ist aber nicht enthalten
a=input('Eingabe Lissajous-Zahl vor  Doppelpunkt ');
b=input('Eingabe Lissajous-Zahl nach Doppelpunkt ');
d=input('Eingabe Lissajous Phasenverschiebung in Grad ');
t=2*pi*(0:0.005:1);
x = cos(a*t);
y = sin(b*t+d*pi/180);
plot(x,y)
```

Laufende Lissajous-Figuren

Vor dem Aufruf dieses M-Files müssen die Verhältniszahlen MLISS:NLISS durch Definition der globalen Variablen 'MLISS' und 'NLISS' festgelegt werden. Anschließend kann das M-File`lissmov` gestartet werden für das Zeichnen einer laufenden, ständig die Phase verändernden Lissajous-Figur.

```
% lissmov Laufende Lissajous-Figur mit
%   durchlaufender Phasenverschiebung
% MLISS:NLISS muss durch die Definition
%  von MLISS und NLISS gegeben sein
a = MLISS;
b = NLISS;
for d = 0:3:1800
  t=2*pi*(0:0.005:1);
  x = cos(a*t);
  y = sin(b*t+d*pi/180);
  plot(x,y)
  axis square
  pause (0.1)
end
```

3D-Visualisierung einer laufenden Lissajous-Figur

Das M-File `liss3dmn` zeigt eine bewegte dreidimensionale Darstellung einer Lissajous-Figur. Die Variablen MLISS und NLISS können das Verhältnis vordefinieren, falls man etwas anderes will als 1:2.

```
%liss3dmn Film einer Lissajous-Figur m:n (1:2) in 3D
t = 0:0.1:6.3;
if (exist(MLISS) & exist(NLISS)) == 0
  a=1; b = 2;
else
  a=MLISS ; b = NLISS;
end
x = cos(a*t) ;  y = sin(a*t) ;  z = cos(b*t);
zc1 = ones(1,length(t)) ;  zc2 = -ones(1,length(t));
xl = [1 1] ;  yl = [0 0] ;  zl = [-1 1];
for again = 1:5
  clf
  tet = input(' 3D-Lissajous-Figur -1:2- Eingabe Aufsicht-Winkel (grad):');
  if isempty(tet)
    tet = 0;
  end
  hdl=plot3(x,y,z,'k') ;  hold on
  axis ([-1.1 1.1 -1.1 1.1 -1.1 1.1]) ;  axis square;  view(0,tet);
  plot3(x,y,zc1,'g') ;  plot3(x,y,zc2,'g')
  hdlr = plot3(xl,yl,zl,'r') ;   hdlb = plot3(yl,xl,zl,'b');
  hold off
% Drehung
  for k=2:2:360
    rotate(hdl,[0 0 1],2) ;    rotate(hdlr,[0 0 1],2)
    rotate(hdlb,[0 0 1],2);    pause(0.1)
```

```
  end
  rep = input(' nochmals (Zahl = yes, ret=nein)?');
  if isempty(rep)
    break
  end
end
```

Zeichnen einer Archimedischen Spirale

```
function [xpt,ypt] = drawarchspir(nturn,apar)
% function [xpt,ypt] = drawarchspir(nturn,apar)
%  Parameter nturn = Anz. Umgaenge
%    (negativ fuer rechtsdrehend)
%            apar = Spiralenparameter r = apar*w/(2*pi)
%                 = Radialgewinn pro Umgang
w = 0:(1.5*sign(nturn)):360*nturn;  r = abs(w)/360;
xpt = r.*cosd(w); ypt = r .*sind(w);
plot(xpt,ypt) ; axis equal
end % function
```

Zeichnen einer Logarithmischen Spirale

```
function [xpt,ypt] = drawlogspir(nturn,lampar)
%  Formel: r(w) = a*exp(k*w),
%  Parameter nturn = Anz. Umgaenge
%    (negativ fuer rechtsdrehend)
w = 0:(sign(nturn)*pi/200):nturn*2*pi;
r = exp(abs(w)*lampar);
xpt = r.*cos(w); ypt = r .*sin(w);
plot(xpt,ypt) ; axis equal
end % function
```

Die Kreis-Evolvente

Eine technisch wichtige Kurve ist die Evolvente. Die Flankenform von Zahnrädern folgt einer Evolventenform. Die Entstehung dieser Kurve kann man sich denken als das Abrollen eines gespannten Fadens von einer zylindrischen Spule mit dem Schreibstift am Fadenende. Eine animierte Demo ist im File `evoldemo.m` verfügbar.

```
%EVOLDEMO  animierte grafische Demo
%  zur Definition von Kreis-Evolventen
%   Copyright 2017, Dr. Stefan Adam
t = 2*pi*(0:0.01:1.4) ;  w = 2*pi*(0:0.01:1);
clf  ; hold on
R=1 ; polx = R ;  poly = 0;
xkri = R*cos(w) ;  ykri = R*sin(w);
plot(xkri,ykri,'k') ;  axis([-12 12 -12 12]) ; axis square
Mofram=moviein(length(t));
```

```
for k=1:length(t)
  lx = 0:0.005*2*pi:t(k);
  bix = (R)*cos(lx) ;   biy = (R)*sin(lx);
  vx = [(R)*cos(t(k)) (R)*cos(t(k))+R*t(k)*sin(t(k))];
  vy = [(R)*sin(t(k))  (R)*sin(t(k))-R*t(k)*cos(t(k))];
  pnewx = (R)*cos(t(k)) + R*t(k)*sin(t(k));
  pnewy = (R)*sin(t(k)) - R*t(k)*cos(t(k));
  if k > 1
    delete(arc) ;  delete(vec) ;  delete(pt);
  end
  arc  = plot(bix,biy,'g') ;   vec = plot(vx,vy,'g') ;
  pt = plot(pnewx,pnewy,'m.') ;
plot([polx pnewx],[poly pnewy],'r');
  polx = pnewx ;  poly = pnewy;
  plot(bix, biy,'g');
  Mofram(:,k)=getframe;
  pause(0.1)
% input('weiter?') % Test beim Erstellen des Programms:
end
pause(0.8);
movie(Mofram,-3) ;  movie(Mofram,1);
```

Kurven in Polarkoordinaten

Das Funktions-M-File ohne Rückgabewert `polarcurve.m` zeichnet eine in Polarkoordinaten definierte Kurve. Die mit einer String-Variablen ausgewählte Radiusfunktion vom Winkel 'w' definiert die Kurve.

```
function iretvoid = polarcurve(funcnam, win)
% polarcurve(fnam, winarray)  Polarkoordinaten-Plot mit
% dem Funktionsnamen der
%  Polarkoordinatenfunktion als Parameter
%   und einem Array von Winkelwerten in Radiant
iretvoid = 0;
x = feval(funcnam, win).*cos(win);
y = feval(funcnam, win).*sin(win);
plot (x,y)
axis equal
axis square
```

Die Definition in Polarkoordinaten im File `archipolexymple.m` beschreibt z. B. eine archimedische Spirale.

```
% rad=polexample(win)  Beispiel einer Polarkoordinaten-Funktion
%   fuer die archimedische Spirale (win in Radiant)
%  Anwendung:  polarcurve('archipol',0:pi/300:4*pi)
function rad = archipol(win)
```

```
rad = 0.1*win;
```

Durch Einsetzen dieser Funktion wie in `polarcurve('archipol',winarray)` wird die archimedische Spirale gezeichnet; das Einsetzen anderer Funktionen in Polarkoordinaten ergibt die entsprechenden Plots.

Die "Versiere di Agnesi" im symbolischen Modus

```
% versierasymb.m   Versiera di Agnesi im symbolic Mode
syms a w x
yw = a/2*(1+cos(2*w)); xw = a*tan(w) % also: w = atan(xw/a)
yx  = a/2*(1 + cos(2*atan(x/a)))
simple(yx)          % liefert   y = a^3/(a^2 + x^2)
figure(1); clf  ;   % zum Plotten Wert: a = 2 festlegen:
ywp = subs(yw,a,2) ; xwp = subs(xw,a,2);  yxp = subs(yx,a,2);
ezplot(xwp,ywp,[-1.4 1.4]) ;     hold on
ezplot(yxp) ; hold off
```

Versiera mit numerischen Ableitungen

```
% Kurve und ihre Ableitung
t = -1.2:0.02:1.2;  n = length(t);
y = 1+ cos(t);    x = 2*sin(t)./(1+cos(t));
yp = (y(2:n) - y(1:n-1)) ./(x(2:n) - x(1:n-1));
xp = (x(2:n) + x(1:n-1))/2;
hold on;  plot(xp, yp, 'g')
% Zusatzberechnung der lokalen Kruemmung
w = atan(yd) ;
sd = sqrt( (y(2:n)-y(1:n-1)).^2 + (x(2:n)-x(1:n-1)).^2 ) ;
sdd = (sd(2:n-1)+sd(1:n-2))/2 ; % Laengenaenderung
wd = (w(2:n-1)-w(1:n-2)) ;      % Winkelaenderung
k=wd./sdd  ;                    % Kappa: k = 1./(sdd./wd)
xdd = (xp(2:n) + xp(1:n-1))/2 ;  plot(xdd,k,'b')
```

22.2.1 Zykloiden

Normale, gestreckte und verschlungene Zykloiden

Eine animierte Demo mit der 'movie'-Funktion von MATLAB kann mit dem M-File `zyklodemo.m` vorgeführt werden. Durch Vordefinieren der globalen Variablen ZYKRAD kann der Typ der Zykloiden gewählt werden (verschlungene > 1, normale $= 1$ oder undefiniert, gestreckte < 1).

```
%ZYKLODEMO  animierte grafische Demo zur Definition von Zykloiden
% Durch Vordefinieren der globalen Variablen ZYKRAD
% (kleiner, = , groesser als 1, fuer gestreckte, normale, verschlungene)
% kann der Typ der Zykloiden gew\"ahlt werden (Defaultwert = 1).
%   Copyright 2003 HSZ-T, Zuerich , Dr. Stefan Adam
t = 2*pi*(0:0.05:1.8) ;  w = 2*pi*(0:0.05:1);
```

```
clf ;  hold on
if  exist('ZYKRAD') == 0
  ZYKRAD = 1
end
R=ZYKRAD ;  polx = 0 ;  poly = -R;
plot([0 4*pi], [-1 -1], 'k') ; axis([-0.3 12.3 -4 8.6]); axis square
Mofram=moviein(length(t));
for k=1:length(t)
  x=t(k) + cos(w) ;   y= sin(w);
  lx = 0:0.05*2*pi:t(k) ;  ly = -ones(1,length(lx));
  bx = t(k) - sin(lx) ;  by = -cos(lx);
  vx = [t(k) t(k)-R*sin(t(k))] ;  vy = [0  -R*cos(t(k))];
  pnewx =  t(k)-R*sin(t(k)) ;  pnewy =  -R*cos(t(k));
  if k > 1
    delete(circ); delete(cent); delete(arc);
    delete(vec); delete(pt);
  end
  circ = plot(x,y,'k') ;  cent = plot(t(k),0,'k.');
  arc  = plot(bx,by,'g') ;  vec = plot(vx,vy,'b');
  pt = plot(pnewx,pnewy,'m.');  plot([polx pnewx],[poly pnewy],'r');
  polx = pnewx ;  poly = pnewy ;  plot(lx,ly,'g');
  Mofram(:,k)=getframe;   pause(0.1);
end
pause(0.8);  movie(Mofram,-3);  movie(Mofram,1);
```

Epi- und Hypozykloiden

Zeichnen von Epi- und Hypozykloiden

```
% epihypozykloide.m   Skript zum Zeichnen von
%  Epizykloiden  (m positiv) und Hypozykloiden (m negativ)
%   m ist Quotient von Rollkreis/Standkreis (im Skript vorwählen)
%   m=1 Cardioide, m=-1/2 Gerade, m=-1/3 Deltoide, m=-1/4 Astroide
%   f ist Quotient Leuchtradius/Rollkreisradius  (Normalfall 1)
r = 5;    m = -1/4;   f = 1;
% interessanter Fall: Knopfloch m=-1/2; f = 0.9;
w = (0:0.001:1)*2*pi;
yc = r*(1+m)*cos(w);    xc = r*(1+m)*sin(w); % Rollkreiszentrum
dy = -r*f*m*cos(w*(1+1/m));    dx = r*f*m*sin(-w*(1+1/m));
figure(1) ;    plot(xc+dx,yc+dy) ;   axis equal
```

Beipiel zur Astroide im symbolic Modus

```
% Astroide im Festkreis mit Radius 1 im symbolic Mode
syms w xx yy real
yc = (3/4)*cos(w);    xc = (3/4)*sin(w);  % Zentrum Rollkreis
dy = 1/4*cos(w*(-3));    dx = -1/4*sin(-w*(-3));
xp = xc + dx;  yp = yc + dy;
figure(1) ; clf;  ezplot(xp,yp) ;  hold on
xs = simple(xp); ys = simple(yp); %xs = sin(w)^3; ys = cos(w)^3
fxy = xx^(2/3) == 1 - yy^(2/3)  % weil: sin(w)^2 = 1 - cos(w)^2
ezplot(fxy,[0 1])   ; hold  off ; % zeichnet einen der 4 Boegen
```

Eine animierte Demo für Epi- und Hypozykloiden analog zu derjenigen für gewöhnliche Zykloiden ergibt sich beim Abarbeiten des Skripts `epizyklodemo.m`

Durch Vordefinieren der globalen Variablen ZYKRADFAC kann das Verhältnis zwischen dem äußeren und inneren Kreis bestimmt werden. (Default, ohne Vordefinition = 0.5). Negative Werte mit Betrag kleiner als 1 liefern Hypozykloiden.

```
%EPIZYKLODEMO  animierte grafische Demo
%  zur Definition von Epizykloiden
% Durch Vordefinieren der globalen Variablen ZYKRADFAC
% (kleiner oder =  1, Radiusverhaeltnis,
%      negativ Hypozykloide)
% kann der Typ der Epizykloiden gew\"ahlt werden
%    (Defaultwert = 1/2).
%   Copyright 2017 HSZ-T, Dr. Stefan Adam
t = 2*pi*(0:0.01:1.4) ;  w = 2*pi*(0:0.01:1); clf ;  hold on
if  exist('ZYKRADFAC') == 0 ;  ZYKRADFAC = 0.5  ; end
R=6 ;  Ra=ZYKRADFAC*R;
frc = 1+ R/Ra ; fr = R/Ra;
polx = 0 ; poly = R;
xkri = R*cos(w) ;  ykri = R*sin(w);
plot(xkri,ykri,'k');  axis([-20 20 -20 20]); axis square
Mofram=moviein(length(t));
for k=1:length(t)
  x=(R+Ra)*sin(t(k)) + Ra*cos(w);  y=(R+Ra)*cos(t(k)) + Ra*sin(w);
  lx = 0:0.005*2*pi:t(k);
  bix = (R)*sin(lx);  biy = (R)*cos(lx);
  bx=(R+Ra)*sin(t(k)) - Ra*sin(t(k)+fr*lx);
  by=(R+Ra)*cos(t(k)) - Ra*cos(t(k)+fr*lx);
  vx = [(R+Ra)*sin(t(k))  (R+Ra)*sin(t(k))-Ra*sin(frc*t(k))];
  vy = [(R+Ra)*cos(t(k))  (R+Ra)*cos(t(k))-Ra*cos(frc*t(k))];
  pnewx = (R+Ra)*sin(t(k)) - Ra*sin(frc*t(k));
  pnewy = (R+Ra)*cos(t(k)) - Ra*cos(frc*t(k));
  if k > 1
    delete(circ); delete(cent);  delete(arc);
    delete(vec);  delete(pt);
  end
  circ = plot(x,y,'k');
  cent = plot( (R+Ra)*sin(t(k)), (R+Ra)*cos(t(k)) ,'k.');
  arc  = plot(bx,by,'g');   vec = plot(vx,vy,'b');
  pt = plot(pnewx,pnewy,'m.');
  plot([polx pnewx],[poly pnewy],'r');
  polx = pnewx;  poly = pnewy;
  plot(bix, biy,'g');
  Mofram(:,k)=getframe;  pause(0.1)
end
pause(0.8);  movie(Mofram,-3); movie(Mofram,1);
```

22.2.2
Bezier Funktionen

Bezier-Interpolation führt zu Bernstein-Polynomen, symbolisches Skript zum Demonstrieren dieser Tatsache:

```
%bezierformel  Kaskadierte lineare Interpolation
```

```
syms P1 H1 H2 P2 t v          % ergibt Bernstein-Polynom
Q1 = v*P1 + t*H1;   Q2 = v*H1 + t*H2;   Q3 = v*H2 + t*P2;
R1 = v*Q1 + t*Q2;   R2 = v*Q2 + t*Q3;   Ba = v*R1 + t*R2
Bb = simplify(Ba),   B = subs(Bb, v, 1-t)
```

Bezier-Demo

```
% bezierdemo.m  Bezierkurve aus 4 gewaehlten Punkten
disp('Eingabe P1, H1, H2, P2 (4Pte) fuer Bezierkurve')
figure(1) ; clf ; clear t ; t = 0:0.01:1;
 [xp,yp] = ginput(4); %Grafische Eingabe von 4 Punkten
xc = xp(1)*(1-t).^3 + 3*xp(2)*t.*(1-t).^2 + ...
       3*xp(3)*t.^2.*(1-t) + xp(4)*t.^3;
yc = yp(1)*(1-t).^3 + 3*yp(2)*t.*(1-t).^2 + ...
       3*yp(3)*t.^2.*(1-t) + yp(4)*t.^3;
plot(xp,yp,'ro-'); hold on
plot(xc,yc,'k') ; hold off
```

Für die Erstellung des Schraubenlinien-Bildes leistet die allgemeine Funktion zur Bestimmung einer Schraubenlinie gute Dienste

```
function [x,y,z] = schraublinie(xap,yap,r,nt,gh,w0,sr)
%function [x,y,z] = schraublinie(xap,yap,r,nt,gh,w0,sr)
% xap, yap = Anfangspunkt, r,nt = Radius, Anzahl Umgaenge
%  w0, sr = Startwinkel, Drehrichtung
w = (0:0.01:nt)*2*pi;
z = gh*w/(2*pi);
x = r*cos(w0 + sr*w) + xap;
y = r*sin(w0 + sr*w) + yap;
end % function
```

Die äussere Schraubenlinie ist rechtsgängig, die innere linksgängig

```
% Skript lrschraublin
%  ineinander gezeichnete lichks und rechtsgaengige
%  Schraubenlinien (Helix)
%  benuetzt Funktion schraublinie
[xa,za,ya] = schraublinie(0,0,1.6,  4,1.5,0,-1)
plot3(xa, ya, za); hold on;
plot3([1.6 0 0 1.6],[0 0 6 6],[0 0 0 0])
[xi,zi,yi] = schraublinie(0,0,1.2,4,1.5,0,1)
plot3(xi, yi, zi); hold off;
axis equal; view(-55,30)
```

Erstellung des Loxodromen-Bildes

```
KURSWINKEL = pi/4
%Loxodrome. Kurve mit konstantem Kompasskurs auf
% Kugelobeflaeche
%  dy/dx = tan(alf) = c = dtet*R/(dphi*cos(tet))
% Seperation und Integeration liefert
% k * phi + K = log ( 1/cos(tet) + sin(tet) / cos(tet) )
w = (0:0.01:1)*2*pi
xa = cos(w);
ya = sin(w);
za = zeros(length(w));
plot3(xa,ya,za,'k')
hold on
plot3([0 0], [0 0], [-1.1 1.1],'k')
kurs = 1/tan(KURSWINKEL)
tet = (-1+0.03:0.01:1-0.03)*pi/2;
for phigrd = 0:30:360
phin = phigrd*pi/180;
phi =  log ( 1./cos(tet) + sin(tet) ./...
        cos(tet) )/kurs + phin;
R = 1;
x = R*cos(tet).*cos(phi);
y = R*cos(tet).*sin(phi);
z = R*sin(tet);
plot3(x,y,z)
axis equal
end
```

Ein M-File zu den Farbdefinitionen mit Dursichtigkeits- Koeffizienten:

```
% tiffany.m  Anwendung der Farbdefinition mit
%   einem zus\"atzlichen
%  Durchsichtigkeits-Koeffizienten   'FaceAlpha'
%   auf einen Rotationk\"orper, dessen Oberfl\'ache mit
%   einer Patchstruktur bedeckt ist.
w = 0:pi/6:2*pi;
r = 0:0.075:0.9;
x = r'*cos(w);
y = r'*sin(w);
colar = r'*ones(1,length(w));
rr = r.^4;
z = 1-1.2*rr'*ones(1,length(w));
%
verx = reshape(x,length(r)*length(w),1);
very = reshape(y,length(r)*length(w),1);
```

```
verz = reshape(z,length(r)*length(w),1);
vert = [verx very verz];
cols =  reshape(colar,length(r)*length(w),1);
%
fcol = ( 1:length(w)*(length(r)-1)  )' ;
fccond = mod(fcol,13) == 0 ;
fcol(fccond) = [];
%
facmat = [fcol fcol+1 fcol+14 fcol+13];
coltabe = (1:4:4*169)';
%
tifhdl = patch('vertices',vert,'faces',facmat, ...
   'FaceVertexCdata',coltabe,...
    'FaceColor','flat','FaceAlpha',0.4);
axis equal;  view(28,16);
disp('Bitte return  druecken fuer Abschluss')
pause
delete(tifhdl); delete(gca);
```

22.3
M-Files zu Folgen und Reihen

Zenonplot

```
% Skript zenonplot.m
%  Grafische Darstellung zum Trugschluss des Zenon
vach = 1.2 ; vtort = 0.3 % Meter pro Sekunde
tbetr = 2 ;
% Zeit, fuer welche der Film zur Betrachtung stoppt
%  die vertikalen Linien zeigen die Positionen
%  nach dem Einholen der vorherigen Vorspungs
dstart = 20  % Vorsprung der Schildkroete
xa(1) = 0 ; ya(1) = 0;
xt(1) = dstart ; yt(1) = 0;
for rep = 2:2:12
  dtim = (xt(rep-1) - xa(rep-1))/vach;
  ya(rep) = ya(rep-1) + dtim; ya(rep+1) = ya(rep)+tbetr;
  yt(rep) = yt(rep-1) + dtim; yt(rep+1) = yt(rep)+tbetr;
  xa(rep) =  xa(rep-1) + vach*dtim; xa(rep+1) = xa(rep);
  xt(rep) =  xt(rep-1) + vtort*dtim; xt(rep+1) = xt(rep);
end
plot(xa,ya,'r') ; hold on ; plot(xt,yt,'k'); hold off
```

Fibonacci-Folge aus Formel und aus Rekursion

```
% fibotest.m
% Test der ersten Glieder der Fibonacci-Folge
% Rekursionsformel gegen Formel des allgemeinen Gliedes
a(1) = 1;  a(2) = 1;
for k=1:20
  a(k) = fiboreku(k);
  f(k) = ( (1+sqrt(5))^k - (1-sqrt(5))^k )/sqrt(5)/2^k;
  d(k) = f(k) -a(k);
end
a ,d
```

Rekursive Berechnungsfunktion für Fibonacci-Folgen

```
function r = fiboreku(n)
% fiboreku.m
% fn=fiboreku(n) berechnet Fibonacci-Zahl zu n rekursiv
if n > 2
  r = fiboreku(n-1) + fiboreku(n-2);
else
  r =  1;
end
```

23
M-Files zu Kapitel 3

23.1
M-Files: Anwendungen linearer Gleichungssysteme

Wheatstone'sche Brücke im abgeglichenen Zustand

```
% ------ wheatstone.m ------------
% die Widerstaende R1, R2,   R3, R4 und R5 muessen vor
%  dem Ausfuehren dieses M-Files definiert werden,
%  ein kleiner Wert von R5 erhoeht die Empfindlichkeit.
   M =[  1  -1   0  -1   0   0 ;
         0   1  -1   0   0  -1 ;
         0   0   0   1  -1   1 ;
         0  R1   0 -R3   0  R5 ;
         0   0  R2   0 -R4 -R5 ;
         0  R1  R2   0   0   0 ] ;
  b = [ 0  0  0  0  0  10 ]'  ;
  isol = M \ b;
% Das hauptsaechlich interessierende Resultat ist
%  der Strom in der Bruecken-Strecke R5
  i5 = isol(6) ;
  disp([ 'Strom in Brueckenzweig:' num2str(i5)] ) ;

% -- end --- wheatstone.m ---------
```

23.2
M-Files zu Orthogonalität und Projektionen

```
% Skript zum testen der Ortogonalitaet
% von sinus- und cosinus Funktionen
% in der Form von Zahlenfolgen
f(:,1) = sico(100,1,0,1);
f(:,2) = sico(100,0,1,1);
```

```
f(:,3) = sico(100,1,0,2);
f(:,4) = sico(100,0,1,2);
f(:,5) = sico(100,1,0,3);
f(:,6) = sico(100,0,1,3);
for zei=1:6
  for spa = 1:6
    Fp(zei,spa) = sum(f(:,zei).*f(:,spa))/100;
  end
end
Fp
```

23.3 M-Files zu Lösungsverfahren

Der Ablauf der einfachen Gauss-Elimination, d.h. der Transformation der gegebenen Matrix A in eine Rechts-Dreiecksmatrix folgt der Sequenz 1. Spalte unterhalb der Diagonalen, 2. Spalte unterhalb der Diagonalen etc. bis in der (n-1)-te Spalte nur nocn ein Element zu Null gemacht werden soll.

Das "dirty prototyle" Programm organisiert die Doppelschleife der zu Null zu machenden Elemente und überschreibt diese mit Null. Das ist natürlich keine Lösung, aber zeigt, ob tatsächlich durch die Schleifen-Organisation eine obere Dreiecksmatix entsteht.

```
function R = dirtygauss(A)
% function R = dirtygauss(A)
% reines Ueberschreiben der
%   unter-Diagonalelemente mit Null
%  als Test der Schleifen-Organisation
%   fuer den Gauss-Algorithmus
  [nz,ns] = size(A); R = A;
  for sp = 1:ns-1
    for ze = sp+1:nz
      R(ze,sp)=0;
    end
  end
end
```

Die Einfache Gauss-Elimination ohne Pivot-Kontrolle

```
function [R,b] = gauss0(Ainp,binp)
% function R = gauss0(Ainp,binp)
%  simple elementare Gauss-Elimination
  [n,ns]= size(Ainp); A = Ainp; b = binp;
  for spa = 1:(n-1)
% alle Spalten, von 1 bis (n-1)
```

```
    for zei = (spa+1):n
% alle Zeilen unterhalb der Diagonalen
      comb = A(zei,spa)/A(spa,spa);
% Faktor: Nullkandidat/Pivot
      % entsprechende Zeilen kombinieren
      %   (Elementaufzaehlung in Zeile
      % mit impliziter Schleife (:))
      A(zei,spa:n) = A(zei,spa:n) - A(spa,spa:n)*comb;
      % Mit-Transformiren de rechten Seiten
      b(zei) = b(zei) - b(spa)*comb;
     end
   end
   R = A  % Kontrolle der R-Form
% (Ouput unueblich in Funktion)
end % function gauss0
```

Nach der Gauss-Elimination erhält man die Lösung erst, wenn man auch noch das Rückwärts-Einsetzen absolviert hat.

```
function x = gaussbk0(R,b)
% function x = gaussbk0(R,b)
% Rueckwaerts-Einsetzen nach Gauss-Elimination
  [n,ns] = size(R)
  x=zeros(n,1);
% x-Laenge durch Nullvektor vordefinieren
  for k=n:(-1):1
% alle Zeilen/Unbekannten von unten nach oben
% bisherige Unbekannte beruecksichtigen
%     bedeutet b korrigieren
%    mit Skalarprodukt:
%    (Matrix-Teilzeile * Vektor-Teilspalte)
    x(k) = (b(k) -  R(k,(k+1):n)*x((k+1):n) )/R(k,k)
%  (b(n) nicht korrigiert weil (k+1 > n) leere Schleife)
  end
end % function gaussbk0
```

Anwenden der einfachen Funktionen funktioniert fast immer bei Random-Eingaben, da diese kaum zufällige Null-Pivot-Elemente enthalten.

```
n  = 5; Ao = rand(n);  bo = rand(n,1);
[R,b] = gauss0(Ao,bo);
x = gaussbk0(R,b)
zerotest = Ao*x - bo
```

Die mit Pivotkontrolle und Zeilenvertauschung arbeitende Gauss-Elimination 'gauss1' ist wesentlich komplizierter

```
function  [xsol,regu]= gauss1(A,b)
% function [xsol,regu] = gauss1(A,b)
% Loesung lineares Gl.System
%    Gauss-Algorithmus mit einfacher Pivot-Suche
%   und anschliessendem Rückwärts-Einsetzen
  nv=size(A); n=nv(1); % Dimension automatisch bestimmen
  [R,bt,regu] = gauss1getr(A,b,nv);
  if regu
    xsol = gauss1bk(R,bt,n);
  else
    R = A; xsol = zeros(n,1);
  end;
end
%  interne Funktion gauss1getr
  function [R,bt,regu]=gauss1getr(A,b,n)  % intern A >> R
    regu = true;
    for spa = 1:(n-1)   % alle Spalten ausser der letzten
        % ---- Pivotsuche, Vertauschung
        [amx,kx] = max(abs(A(spa:n,spa)))
% max. Betrag fuer Pivot
        if amx  == 0  % groesster Betrag ist 0, Abbruch
          regu = false; disp( 'Matrix singulaer!') ;
          break
        end
        kmx=kx+spa-1; % Absoluter Vertauschungsindex
        if kmx > spa % vertausche Rest-Zeilen falls sinnvoll
         % Index-Liste erlaubt Vertauschen ohne Hilfsplatz
         A([spa kmx],spa:n) = A([kmx spa],spa:n);
         b([spa kmx]) = b([kmx spa]);
        end %if kmx  ---- Ende Pivotsuche und Vertauschung
      for zei = (spa+1):n
% alle Zeilen unterhalb der Diagonalen
        comb = A(zei,spa)/A(spa,spa); % Division nur 1 mal
%  entsprechende Zeilen kombinieren
        A(zei,spa:n) = A(zei,spa:n) - A(spa,spa:n)*comb;
        b(zei) = b(zei) - b(spa)*comb;
      end  %for zei
    end % for spa
    if A(n,n)  == 0  % Test unterstes Diagonalelement
       regu = false; disp( 'Matrix singulaer!')
     end
     R = A; bt = b;
   end % interne Funktion gauss1getr
%  Rueckwaerts-Einsetzen ist enthalten
   function xsol = gauss1bk(R,b,n) % Rueckwaertseinsetzen
```

```
     xsol = zeros(n,1);   % Rueckgabevektor initialisieren
% alle Zeilen/Unbekannten von unten nach oben
     for k = n:(-1):1
       xsol(k =(b(k)-R(k,(k+1):n)*xsol((k+1):n) )/R(k,k);
     end % for k
   end % interne Funktion gauss1bk
end % funktion gauss1
```

23.3.1
Showtime Gauß-Elimination

Das M-File `showgauss` zeigt den Ablauf einer Gauß-Elimination Schritt für Schritt. Als Rückgabe-Werte sind der Lösungsvektor und die Dreiecksmatrix vorgesehen.

```
function [xsol,R] = showgauss(A,b)
%SHOWGAUSS Schrittweise Anzeige der Aktionen im Gauss-Algorithmus
%   showgauss(A,b) fuehrt die Gauss-Elimination fuer das System A*x=b durch
%   showgauss   ohne Parameter loest ein default-Gleichungssystem
%   Copyright 2002 HSZ-T, Zuerich , Dr. Stefan Adam
  if nargin < 1
%   Default-System
    b = [ 10  0  4 ]'  ;
    A = [  4  4  8 ;  2  -3   9 ; 1  2  2 ] ;
  end
format rat;  more off; clc; home
%
disp(' showgauss: Ablauf-Demonstration des Gauss-Algorithmus ')
disp(' Start-Geichungssystem:  A * x = b ')
 A, b
disp('<return> zum Weiterfahren ..'), pause
[m1,n1] = size(A);  n = max(m1,n1);  xsol = zeros(n,1);
%
% alle Spalten ausser der letzten, (beginnend mit 1)
%
for kol = 1:(n-1)
% Null-Test des Pivot-Kandidaten, evtl. vertauschen
  if A(kol,kol) == 0
    [amx,kx] = max(abs(A(kol:n,kol)));
     if amx  == 0
      disp( 'Matrix singulaer!' );  return
     end
% Absoluter Vertauschungsindex, statt relativ zu kol
     kx=kx+kol-1;
home
disp([' Zeilenvertauschung noetig: ', int2str(kol) , ' <==> ', int2str(kx) ])
% vertauschen Restzeilen
     hz=A(kol,kol:n);
     A(kol,kol:n) = A(kx,kol:n);
     A(kx,kol:n) = hz;
% vertauschen rechte Seiten
     hb = b(kol);    b(kol) = b(kx);    b(kx) = hb;    A , b
  end
  Aol = A;  bol = b;
```

```
% Pivot-Absicherung OK, Kombinationen durchfuehren
% alle Zeilen unterhalb der Diagonalen
  for lin = (kol+1):n
% ~~~Division nur einmal ausfuehren, da immer dieselbe
    comb = A(lin,kol)/A(kol,kol);
% ~~~ entsprechende Zeilen kombinieren
% ~~~ Elementaufzaehlung mit impliziter Schleife
    A(lin,kol:n) = A(lin,kol:n) - A(kol,kol:n)*comb;
    b(lin) = b(lin) - b(kol)*comb;  R=A;  clc;  home
disp(' showgauss: Ablauf-Demonstration des Gauss-Algorithmus ')
    Aol , bol
disp([' Schritt ' int2str(kol) ' ,  Sub-Schritt ' int2str(lin-kol) ...
':  Element A(' int2str(lin) ',' int2str(kol) ')  wird  0 .  ' ...
 '( b wird angepasst )']) 
disp(['      durch A(',int2str(lin),',:) = A(',int2str(lin),':) + (', ...
   num2str(-comb),')*A(',int2str(kol),',:)'   ])
    if (kol == n-1 ) & ( lin == n) ;   R,b
    else  ;      A,b
    end;  Aol = A;  bol = b;
disp('<return> zum Weiterfahren ..'), pause
  end
end
% Schluss-Test auf Ann == 0
  if A(n,n)  == 0
    disp( 'Matrix singulaer!')
    R=0; xsol=0;  return
  end
% Rueckwaerts-Einsetzen
  clc;  home;  R , b
disp('  Rueckwaerts-Einsetzen ')
% alle Zeilen/Unbekannten von unten nach oben
  for k=n:(-1):1
    xsol(k) = (b(k) -  A(k,(k+1):n)*xsol((k+1):n) )/A(k,k);
disp([' x(' int2str(k) ') = (' num2str(b(k)) ' - ' ...
 num2str( A(k,(k+1):n)*xsol((k+1):n) ) ' ) / ( '  num2str(A(k,k)) ...
 ' )   =  ' num2str(xsol(k)) ])
disp('<return> zum Weiterfahren ..'); pause
  end ;  xsol
```

23.3.2 Showtime L-R-Faktorisierung

Das M-File `showlr.m` zeigt analog zur Gauss-Elimination die L-R Faktorisierung Schritt für Schritt.

```
function [L,R,P] = showlr(A)
%SHOWLR Schrittweise Anzeige der Aktionen in der L-R-Faktorisierung
%   showlr(A) fuehrt die L-R-Faktorisierung fuer die Matrix A durch
%   showlr  benuetzt ein default-Gleichungssystem
%   Copyright 2002 HSZ-T , Dr. Stefan Adam
  if nargin < 1
%   Default-System
    A =  [  4   4   8  ;  2  -3   9   ; 1    2   2 ] ;
  end
  format rat; more off; clc
```

```
home
disp(' showlr: Ablauf-Demonstration der L-R-Faktorisierung ')
disp(' Start-Matrix:  A ');  A
disp('<return> zum Weiterfahren ..'), pause
  [m1, n1] = size(A);  n = max(m1,n1);
  P = eye(n);  L = eye(n);
%
% alle Spalten ausser der letzten, (beginnend mit 1)
for kol = 1:(n-1)
%     ['L' int2str(kol)] = eye(n);
     Ltmp = eye(n);
% Null-Test des Pivot-Kandidaten, evtl. vertauschen
  if A(kol,kol) == 0
    [amx,kx] = max(abs(A(kol:n,kol)));
     if amx  == 0; disp('Matrix singulaer!'); return; end
% Absoluter Vertauschungs-Index, statt relativ zu kol
     kx=kx+kol-1;
home
disp([' Zeilenvertauschung noetig: ', int2str(kol) , ' <==> ', int2str(kx) ])
% vertauschen Rest-Zeilen
     hz=A(kol,kol:n);   A(kol,kol:n) = A(kx,kol:n);  A(kx,kol:n) = hz;
% vertauschen rechte Seiten
     hb = b(kol);  b(kol) = b(kx);  b(kx) = hb;
% partielle und dann totale Permutationsmatrix
     PP = eye(n); PP(kol,kol) = 0;  PP(kx,kx) = 0;
     PP(kx,kol) = 1;  PP(kol,kx) = 1;  P = PP*P;
     P
  end
  Aol = A;
% Pivot-Absicherung OK, Kombinationen durchfuehren
% alle Zeilen unterhalb der Diagonalen
  for lin = (kol+1):n
% ~~~Division nur einmal ausf\"uhren, da immer dieselbe
    comb = A(lin,kol)/A(kol,kol);
% ~~~ entsprechende Zeilen kombinieren
% ~~~ Elementaufzaehlung mit impliziter Schleife
    A(lin,kol:n) = A(lin,kol:n) - A(kol,kol:n)*comb;
    Ltmp(lin,kol) = -comb;   R=A;   clc
home
disp(' showlr: Ablauf-Demonstration der L-R-Faktorisierung ')
Aol
disp([' Schritt ' int2str(kol) ' ,  Sub-Schritt ' int2str(lin-kol) ...
':  Element A(' int2str(lin) ',' int2str(kol) ')  wird  0 .  ' ])
disp(['      durch A(',int2str(lin),',:) = A(',int2str(lin),':) + (', ...
    num2str(-comb),')*A(',int2str(kol),',:)'   ])
%
    if (kol == n-1 ) & ( lin == n);  R
    else ;  A
    end
Ltmp
%['L' int2str(kol)] = Ltmp
    Aol = A;
disp('<return> zum Weiterfahren ..'), pause
  end
%  L akkumulieren mit Rechts-Multiplikation von Ltmp
  L = L*(Ltmp^(-1));
```

```
end
% Schluss-Test auf Ann == 0
     if A(n,n) == 0; disp('Matrix singulaer!'); return; end
% L-Matrizen zusammenstellen
clc;  home
disp(' showlr: Ablauf-Demonstration der L-R-Faktorisierung ')
P,  R ,L
format short
```

23.3.3 Showtime Gauss-Jordan-Algorithmus

Das M-File `showgaussjor.m` zeigt analog zur Gauss-Elimination den Gauss–Jordan-Algorithmus zur Matrix-Inversion Schritt für Schritt. (Aufruf:`Ainv = showgaussjor(A)` oder einfach `showgaussjor`)

```
function Ainv = showgaussjor(A)
%SHOWGAUSSJOR Schrittweise Anzeige der Aktionen im Gau{\ss}--Jordan-Algorithmus
%   showgaussjor(A) Invertiert  A
%   showgaussjor  invertiert eine default-Matrix
%   Copyright 2017, Dr. Stefan Adam
  if nargin < 1
%   Default-System
    A =  [  4   4   8  ;  2  -3   9   ; 1    2   2 ] ;
  end
%format rat  % kann eventuell reaktiviert werden fuer einfachere Anzeigen
  more off ;  clc
  [m1,n1] = size(A);  n = max(m1,n1);  pvec = 1:n; AI = [A eye(n)];
home
disp(' showgaussjor: Ablauf-Demonstration des Gauss--Jordan-Algorithmus ')
disp(' Start-Matrix:  [A I] ');  AI
disp('<return> zum Weiterfahren ..'), pause
  Rang = n;
%
% 1. Gau{\ss}-Eliminations-Teil
% alle Spalten ausser der letzten, (beginnend mit 1)
%
for kol = 1:(n-1)
% Null-Test des Pivot-Kandidaten, evtl. vertauschen
  if AI(kol,kol) == 0
    [amx,kx] = max(abs(AI(kol:n,kol)));
    if amx  == 0; disp('Matrix singulaer!'); Rang = Rang -1;
       break;  end
% Absoluter Vertauschungs-Index, statt relativ zu kol
    kx=kx+kol-1;  home
disp([' Zeilenvertauschung noetig: ', int2str(kol) , ' <==> ', int2str(kx) ])
% vertauschen Rest-Zeilen
    hp = pvec(kol);  pvec(kol) = pvec(kx);  pvec(kx) = hp;
    hz=AI(kol,kol:n*2); AI(kol,kol:n*2)=AI(kx,kol:n*2); AI(kx,kol:n*2)=hz;
    AI
  end
% Pivot-Absicherung OK, Kombinationen durchfuehren
% alle Zeilen unterhalb der Diagonalen
  for lin = (kol+1):n
    AIol = AI;
```

```
% ~~~Division nur einmal ausfuehren, da immer dieselbe
    comb =  AI(lin,kol)/AI(kol,kol);
% ~~~ entsprechende Zeilen kombinieren
% ~~~ Elementaufzaehlung mit impliziter Schleife
    AI(lin,kol:n*2) = AI(lin,kol:n*2) - AI(kol,kol:n*2)*comb;
    RH=AI;  clc; home
disp(' showgauss: Ablauf-Demonstration des Gauss-Algorithmus ')
    AIol
disp([' Schritt ' int2str(kol) ' ,  Sub-Schritt ' int2str(lin-kol) ...
 ':  Element A(' int2str(lin) ',' int2str(kol) ')  wird  0 .  ' ])
disp(['      durch A(',int2str(lin),',:) = A(',int2str(lin),':) + (', ...
    num2str(-comb),')*A(',int2str(kol),',:)'   ])
    if (kol == n-1 ) & ( lin == n);  RH
    else;  AI
    end;   AIol = AI;
disp('<return> zum Weiterfahren ..'), pause
  end
end
% Schluss-Test auf Ann == 0
     if AI(n,n) == 0; disp('Matrix singulaer!'); Rang = Rang -1; end
Rang
% Jordan Eliminations-Teil
clc  ; RH = AI
home; disp('  Jordan-Eliminations-Teil  ')
% rueckwaerts alle Spalten ausser der ersten, (beginnend mit n)
%
for kol = n:-1:2
% keine Pivot-Vertauschungen mehr noetig
% Null-Test des Pivot-Kandidaten trotzdem sinnvoll
  if RH(kol,kol) == 0; disp('Matrix singulaer!') ; return
  else;  RHol = RH;
% alle Zeilen oberhalb der Diagonalen
    for lin = (kol-1):-1:1
% ~~~Division nur einmal ausf\"uhren, da immer dieselbe
      comb = RH(lin,kol)/RH(kol,kol);
% ~~~ entsprechende Zeilen kombinieren
% ~~~ Elementaufzaehlung mit impliziter Schleife
      RH(lin,lin:n*2) = RH(lin,lin:n*2) - RH(kol,lin:n*2)*comb;
      DG=RH;  clc;      home
disp(' showgaussjor: Ablauf-Demonstration Jordan-Elimination ')
      RHol
disp([' Jordan-Schritt ' int2str(kol) ' ,  Sub-Schritt ' int2str(kol-lin) ...
      ':  Element A(' int2str(lin) ',' int2str(kol) ')  wird  0 .  ' ])
disp(['      durch A(',int2str(lin),',:) = A(',int2str(lin),':) + (', ...
      num2str(-comb),')*A(',int2str(kol),',:)'   ])
      if (kol == 2 ) & ( lin == 1); DG
      else;  RH
      end;    RHol = RH;
disp('<return> zum Weiterfahren ..'), pause
    end    %  Verarbeitung dieser Spalte fertig
  end   %  Schleife 1:n-1 ueber Spalten
end
% Skalierungs-Teil
clc
home
DG
```

```
IB = DG;
disp('  Skalierungs--Teil  ')
for k = 1:n
  s = 1/IB(k,k); IB(k,k) = 1; IB(k,n+1:2*n) = s*IB(k,n+1:2*n);
end
IB
 Ainv = IB(:,n+1:2*n)
pvec
disp('<return> zum Weiterfahren ..'), pause
```

23.3.4 Volle Pivot-Kontrolle

Für singuläre Systeme braucht es eine volle Pivot-Kontrolle. Nur so verschieben sich die Nullzeilen ans Ende der R-Matrix

```
  function [R,perm,bt,rgdef]=gauss2fpiv(A,b)  % A -> R full Pivoting
  % function [R,perm,bt,rgdef]=gauss2fpiv(A,b)
    [zz,sz] = size(A) ;  perm = 1:sz;
    rgdef = 0; tol = max(max(A))*eps; spmx = min(zz-1,sz);
    for spa = 1:spmx   % hoch, alle; quad+breit zz-1
      % ---- Pivotsuche im ganzen Feld, ev Vertauschung
      [amxsv,kxsv] = max(abs(A(spa:end,spa:end))) ;  % Spaltenmaxima
      [amx,kx] = max(amxsv) ; % Maximum aller Spaltenmaxima
      ixz = spa-1+kxsv(kx); ixs = spa-1+kx; % Absolute Indizes
      if amx  < tol; break ; end % nur noch Null-Pivots, Abbruch
      if ixs > spa % ev. Vertauschung Spalten, (Protokoll in perm)
        A(:,[spa,ixs])=A(:,[ixs,spa]); perm([spa,ixs])=perm([ixs,spa]);
      end %-- ev. Vertauschung  Spalten
      if ixz > spa %  ev. Vertauschung Zeilen (mit Index-Liste)
        A([spa ixz],spa:end) = A([ixz spa],spa:end);
        b([spa ixz]) = b([ixz spa]);
      end %-- ev. Vertauschung Zeilen
% Nullen unterhalb Diag erstellen, eigentliche Gauss-Schritte
      for zei = (spa+1):zz  % alle Zeilen unterhalb der Diagonalen
        comb = A(zei,spa)/A(spa,spa); % Kombinationsfaktor
        A(zei,spa:end) = A(zei,spa:end) - A(spa,spa:end)*comb;
        b(zei) = b(zei) - b(spa)*comb;
      end  %for zei
    end %-- for spa
     R = A; bt = b;
     for ztr = min(sz,zz):(-1):1  % Nullzeilen zaehlen = rgdef
       if  norm(R(ztr,:)) > tol ; break ; else ; rgdef = rgdef+1 ; end
     end
   end %-- Funktion gauss2fpiv
```

Die Auswertung der Nullzeilen bei vollständiger Pivot-Konmtrolle ergibt neben der Lösung auch den Nullraum

```
function [ xpa, nlsp ] = gauss2gensol(R,perm,bt)
%  function [ xpa, nlsp ] = gauss2gensol(R,perm,bt)
%  Allgemeine Lösung lineares Gleichungssystem
%  xpa = partikulaere (bzw. einzige) Loesung,
%  nlsp = Nullraum
```

```
[zz,sz] = size(R);  tol = max(max(R)) * eps; bx = bt;
xpt = zeros(sz,1);  rgdef = 0;
    for ztr = min(sz,zz):(-1):1
% Nullzeilen zaehlen = rgdef
      if  norm(R(ztr,:)) > tol ; break ;
% Erste Nicht-Nullzeile
      else ; rgdef = rgdef+1 ; bx(zz+1-rgdef) = 0; end
    end %--for ztr
    if norm(bt) > norm(bx);
       disp('Widerspruch in Gleichungen'); end
    for zput = 1:rgdef; xpt(sz+1-rgdef) = 0;  end %--for zput
    for zsol = zz-rgdef:(-1):1
% Rueckwaerts-Einsetzen fuer xpt
      bcor = bt(zsol)-R(zsol,:)*xpt;
      xpt(zsol) = bcor /R(zsol,zsol);
    end %--for zsol
    xpa(perm,1) = xpt;
% Spaltenvertauschungen beruecksichtigen
    nlsp = []; nldim = sz-zz+rgdef
% Nullraum aufbauen
    for nlnum = 1:nldim
% fuer jede Nullzeile eine 1 setzen, sonst 0
      xncand = zeros(sz,1); xncand(sz+1-nlnum) = 1;
      for zsol = zz-rgdef:(-1):1
% Rueckwaerts-Einsetzen Nullraum
        bcor = -R(zsol,:)*xncand;
        xncand(zsol) = bcor /R(zsol,zsol);
      end %--zsol
      xncand(perm,1) = xncand;
      nlsp = [nlsp xncand];
% Nullraumvektoren nebeneinanderstellen
    end %--for nlnum
end %----function
```

23.3.5
QR-Zerlegung

Die QR-Zerlegung benutzt die Housholder-Transformation, eine orthogonale Spiegelung in mehreren Dimensionen.

```
function Qp = houstra(A,spa)
% function Qp = houstra(A,spa)
% houstra: Einzelne Housholder Teil-Q-Matrix
% Qp*A hat in Spalte spa unterhalb Diagonalen nur 0
  n = size(A,1); s = zeros(n,1); re = zeros(n,1);
```

```
  s(spa:n) = A(spa:n,spa); re(spa) = 1;
   % s ist Teil-Spaltenvektor unterhalb Diag von A
  v = s + norm(s)*re;
   % v zeigt von s zu +/-(re*norm(s)) und ist damit
   %    Normalenvektor auf Spiegelebene
   % 1.Element von s negativ: v zeigt zu (re*norm(s))
  if s(spa) < 0   v = s - norm(s)*re;   end;
  ve = v/norm(v);
  Qp = eye(n) - 2*ve*ve'; % Housholder Spiegelung
end % function Qp
```

Damit kann eine Version der QR-Zerlegung selbst programmiert werden

```
function [Qaco,R] = myqr(A)
%  function [Q,R] = myqr(A)  selbst programmierte QR Zerlegung
%  berechnet einzelne Spalten-Transformationen mit houstra.m
[n,m] = size(A);  R=A ; Qacc = eye(n);
nstep = min(m, n-1); % Anpassung fuer Ausgleichsprobleme n>m
for spa = 1:nstep
    Qpart = houstra(R,spa);
    R = Qpart*R; Qacc = Qpart*Qacc;
end
Qaco= Qacc';
end
```

Für die eigentliche Lösung braucht es auch bei der QR-Zerlegung ein Rückwärts-Einsetzen

```
function [xsol] = myqrbksub(R,b)
% function [xsol] = myqrbksub(R,b)
% Rueckwaerts-Einsetzen nach QR, auch fuer Ausgleichsprobleme
[nd,md] = size(R); xsol = zeros(md,1);
  for k=n:(-1):1
    xsol(k) = (b(k) -  R(k,(k+1):md)*xsol((k+1):md) )/R(k,k);
  end
end
```

23.4 M-Files zu Eigenwerten und Eigenvektoren

Iterierte Abbildung durch eine 2x2 Matrix Um die grafische Darstellung der iterierten Abbildung durch eine Matrix sehen zu können kann eine beliebige 2x2 Matrix `M` eingesetzt werden; ebenso kann die Anzahl Iterationen`iter` vorgewählt werden. Ein sehr hübsches Beispiel liefert die drehende und kontrahierende Abbildung M=[0.9 -0.1 ; 0.1 0.9], welche als Default eingesetzt wird. Das

Funktions-M-File heißt `iterativemap.m` und wird als `iterativemap` oder `iterativemap(iter,M)` aufgerufen.

```
% voidpar = iterativemap(iter,M)
% Skript zum Verfolgen einer mehrfachen Abbildung
% durch die eingegebene Matrix M. Es wird 'iter'-mal abgebildet.
function iterativemap(varargin)
if nargin > 1
  Mac = varargin{2};
else
  Mac = [0.9 -0.1 ; 0.1 0.9];
end
if nargin > 0
  iterc = varargin{1};
else
  iterc = 6;
end

clf  ; hold on
xy = zeros(20,2,11,11);
for ili = 1:11
  for iro = 1:11
% Start-Koordinatenpaare bei x,y = -10:2:+10
    xy(1,1,ili,iro) = 2*(iro-6);
    xy(1,2,ili,iro) = 2*(ili-6);
% wiederkehrende Abbildung, (iter-Mal) mit M
    for k=1:iterc
      xy(k+1,:,ili,iro) = (Mac*(xy(k,:,ili,iro))' )';
    end
  end
end
% Plot der Abbildungs-Geschichte
for ili = 1:11
  for iro = 1:11
    plot(xy(1:iterc+1,1,ili,iro),xy(1:iterc+1,2,ili,iro))
    plot(xy(1,1,ili,iro),xy(1,2,ili,iro),'k+')
    plot(xy(2:iterc+1,1,ili,iro),xy(2:iterc+1,2,ili,iro),'ro')
  end
end
axis equal; hold off
```

Bestimmung der iterativen Abbildung durch eine Matrix mit Hilfe der Zerlegung einer Matrix in ihre Eigenwerte und Eigenvektoren

```
function [Miter,eigs,eigv] = itermateig(M,niter)
%function [Miter,eigs,eigv] = itermateig(M,niter)
[evec,eval] = eig(M);
```

Symbolisches Skript zur Bestimmung des allgemeinen Gliedes der Fibonacci-Folge.

```
%  skript fibonaccisymb.m
%  Allgemeines Glied der Fibonacci Folge
syms k  ;  ftab = [];
M = sym([1 1; 1 0])  % Iterationsmatrix
[V,D ] = eig(M)      % Eigenvektoren und Eigenwerte
cv = V\[ 1; 1]       % Zerlegung Startvektor
%  Formel allgemeines, k-tes Element
av1 = D(1,1)^(k-1)*cv(1)*V(2,1) + D(2,2)^(k-1)*cv(2)*V(2,2)
%  av1 = -(5^(1/2)*((1/2 - 5^(1/2)/2)^k - ...
%     (5^(1/2)/2 + 1/2)^k))/5
for n=1:15
    fibns = subs(av1,k,n); fibnn = simplify(fibns);
    ftab = [ftab, fibnn];
% Anhaengen neues Element an Tabelle
end
ftab
```

23.5
M-Files zum Thema endliche Rechengenauigkeit

IEE745 Display

```
function [exp, man, octstr] = ieeedisplay(z)
```

Konditionszahlen verschiederne Hilbertmatrizen

```
for dm = 5:5:20
  HM = hilb(dm);
  [dm, cond(HM), log10( cond(HM))]
end
```

Konditionszahlen von annähernd singulären 2x2- Matrizen

```
format long g
for dex = 2:2:10
  PM = [1 1 ; 1 1+10^(-dex)] ;
  [-dex, cond(PM), log10( cond(PM))]
end
```

Konditionszahlen von annähernd singulären Matrizen höherer Dimension

```
I5bs = eye(5); I5bs(1,1)= 1e-8;
c5 = cond(I5bs)
I10bs = eye(10); I10bs(1,1)= 1e-8;
c10 = cond(I10bs)
I20bs = eye(20); I20bs(1,1)= 1e-8;
c 20 = cond(I20bs)
I100bs = eye(100); I100bs(1,1)= 1e-8;
c100 = cond(I100bs)
```

24
M-Files zu Kapitel 4

24.1
M-Files zu Vektoren in der Elementargeometrie

Demonstration, dass das Vektorprodukt einer antisymmetrischen Matrix entspricht

```
function shovecprod(u,v)
% function shovecprod(u,v)
% Zeigt Vektorprodukt auf 2 Arten berechent
p = cross(u,v)
pm = u*v' - u'*v  % Differenz der dyadischen Produkte
pmv = [pm(2,3) -pm(1,3) pm(1,2) ]'
dp = p-pmv
```

24.2
M-Files zur Raumgeometrie

Demonstration der Hesse'schen Normalform

Das Programm `showhessenf.m` zeigt die Verbindungsgerade zwischen zwei Punkten, dann den zugehörigen Normaleneinheitsvektor mit seiner Trägergeraden und dann den Ortsvektor eines dritten Punktes. Die Projektion dieses Ortsvektors minus die Projektion eines Verankerungspunktes der Geraden zeigt den Abstand des dritten Punktes von der Geraden. Ist dieser Abstand Null, so liegt der Punkt auf der Geraden.

```
% SHOWHESSENF Darstellung der Funtionsweise der Hesse'schen Normalform
clf; hold on
disp('SHOWHESSENF Darstellung der Funtionsweise der Hesse'schen Normalform')
disp('Bitte mit dem Cursor 2 Punkte fuer eine Gerade eingeben')
disp('und einen weiteren Punkt fuer den Test-Vektor!')
plot([-6 6], [0 0],'k'); plot([0 0],[-6 6], 'k')
axis([-6 6 -6 6]); axis square
[x1 y1] = ginput(1);  plot(x1, y1, 'or')
[x2 y2] = ginput(1);  plot(x2, y2, 'or')
x0 = x1 -20*(x2-x1);  y0 = y1 -20*(y2-y1);
x3 = x1 +20*(x2-x1);  y3 = y1 +20*(y2-y1);
```

```
plot([ x0 x3], [y0 y3],'b' );  plot([0 x1 x2], [0 y1 y2],'g' )
vn = [y2-y1 x1-x2];
if vn*[x1 y1]' >0
  en = vn/norm(vn);
else
  en = -vn/norm(vn);
end
plot([0 10*en(1)], [0 10*en(2)],':g' ); plot([0 en(1)], [0 en(2)],'g' )
q=en*[x1 y1]';  plot(en(1)*q, en(2)*q, '+g' )
[hx hy] = ginput(1);  plot(hx, hy, 'or');  vh = [hx hy]';  p=en*vh;
plot([ 0  hx], [0 hy],'r' );  plot([en(1)*q en(1)*p], [en(2)*q en(2)*p],'r' )
plot(en(1)*p, en(2)*p,'*r' )
```

Die Berechnung der Minimaldistanz zweier Geraden

```
function [ lam, mu, P, Q ,d] = mindist2ger( a, r, b, s)
%function [ lam, mu, P, Q ,d ] = mindist2ger( a, r, b, s)
%  Minimaldistanz/Schnittpunkt 2 Geraden (windschief/in Ebene)
%  Gerade 1: a + lam*r   Gerade 2: b+ mu*s
%  a, b: Ortsvektoren, r,s, Richtungsvektoren (3D oder 2D)
%  lam, mu Laufparameter fuer Minimaldistanz
%  P, Q Punkte auf Gerade 1 bzw 2 für Minimaldistanz d
  M = zeros(2) ;     rhs = zeros(2,1);
  M(1,1) = r'*r;     M(1,2) = - r'*s;
  M(2,1) = -r'*s ;   M(2,2) = s'*s;
  rhs(1) = r'*(b-a); rhs(2) = s'*(a-b);
  psol = M\rhs;  lam = psol(1);  mu = psol(2);
  P = a+lam*r;   Q = b + mu*s;   d = norm(Q-P);
end
```

Beispiel Kirchturmdach

```
% Kirchturmdach
% Koordinaten (Ortsvektoren) Grundfläche und Spitze
A = [-3 -3 0]'; B = [3 -3 0]'; C = [3 3 0]'; D = [-3 3 0]';
S = [0 0 4]';  format compact
% Dachkanten, Grundkanten und Fallinie vom Spitze aus
lgkante = norm(B-A), ldkante = norm(S-A), MS = norm(S-(A+B)/2)
%  Normalenvektor und Flächenberechnung
nabs = cross(B-A,S-A), Arpara = norm(nabs);
% Kreuzprod, Betrag
Ardrei = Arpara/2 % Dreiecksfläche = Parallelogramm/2
Ardrei2 = lgkante*MS/2 % Kontrolle mit Grundformel
% Normalen-Einheitsvektoren, Winkel in Grad
neabs = nabs/(norm(nabs)), nexy = [0 0 1]'
% Nvec auf x-y-Ebene
wdachebene = acosd(nexy'*neabs)  % Dachneigung
wkanteebene = 90-acosd(nexy'*(S-A)/norm(S-A))
nbcs = cross(C-B,S-B)
nebcs = nbcs/norm(nbcs)
wdachdach = acosd(neabs'*nebcs)  % Dach-Dach-Winkel
```

"Ein Hexagon im Hexaeder, ganz ohne Hexerei!"

```
% Wuerfelschnitt.m, senkrecht zur Koerperdiagonalen
A=[0 0 0]'; B=[1 0 0]'; C=[1 1 0]'; D=[0 1 0]';
E=[0 0 1]'; F=[1 0 1]'; G=[1 1 1]'; H=[0 1 1]';
ne = (G-A)/norm(G-A);  M=[0.5 0.5 0.5]'; d=ne'*M;
% 6 Kanten-Geraden mit Verankerung und Richtung
Kv = [B B E E D D]; Kr = [C-B F-B F-E H-E H-D C-D];
% ne'*Kv -lam*ne'*Kr -d = 0; lam = (d -ne'*Kv)./(ne'*Kr)
lam = -(ne'*Kv -d)./(ne'*Kr) % Serie von 6 lambda-Werten
Dpts = Kv + repmat(lam,3,1).*Kr % alle 6 Durchstosspunkte
plot3(Dpts(1,[1:end 1]),Dpts(2,[1:end 1]),Dpts(3,[1:end 1]))
wlin = [A B C D A E F B F G C G H D H E]; hold on
plot3(wlin(1,:), wlin(2,:), wlin(3,:),'r');
axis equal; view(69,18); hold off
```

24.3
M-Files: Längen und Winkeln in höheren Dimensionen

Die Winkel zwischen den Diagonalen des 4D Hypercube (Die Winkel zwischen beliebigen Kanten sind entweder 0 oder 90 Grad)

```
function [whd3, whd4] =  hypcub4diag
% function [whd3, whd4] =  hypcub4diag
%  Winkel zwischen Kanten und Diagonalen im 4D Hypercube
% Kantenvektoren
kv4 = [[1 0 0 0]', [0 1 0 0]', [0 0 1 0]', [0 0 0 1]']
% 3D Diag
d3diamat = [ ...
[1  1  1  0]' , [1 -1  1  0]' , [1  1 -1  0]' , [-1  1  1  0]',...
[1  1  0  1]' , [1 -1  0  1]' , [1  1  0 -1]' , [-1  1  0  1]'....
[1  0  1  1]' , [1  0 -1  1]' , [1  0  1 -1]' , [-1  0  1  1]',...
[0  1  1  1]' , [0  1 -1  1]' , [0  1  1 -1]' , [ 0 -1  1  1]']
nk = 4; nd3 = 16
for k= 1:nk
  for j = 1:nd3
    whd3(k,j)= wink4(kv4(:,1),d3diamat(:,j))
  end
end
% 4D Diag
d4diamat = [...
[1  1  1  1]' , [1 -1  1  1]' , [1  1 -1  1]' , [1  1  1 -1]',...
[1 -1 -1  1]' , [1 -1  1 -1]' , [1  1 -1 -1]' , [1 -1 -1 -1]']
nd4 = 8;
for k= 1:nk
  for j = 1:nd4
    whd4(k,j)= wink4(kv4(:,1),d4diamat(:,j))
  end
end
%
  function w4 = wink4(u,v)
```

```
    w4 = acosd(u'*v/(norm(u)*norm(v)));
  end
end
```

24.4 M-Files zu geometrischen Abbildungen

Die Abbildungen mit 2x2 Matrizen ohne Einsatz der homogenen Koordinaten beschränken sich auf Spiegelungen und Drehungen um den Nullpunkt. Als Hilfsfunktion für alle Darstellungen derartiger Abbildungen dient dispmap22.

```
function Lmx = dispmap22(Mx, Lur)
% function Lmx = dispmap22(Mx, Lur)
% Zeichne Urbild und Bild bei Abbildung durch 2x2 Matrix
  Lmx = Mx*Lur
  plot(Lur(1,:), Lur(2,:),'k','linewidth',2)
  axis([-8 8 -8 8])
  axis square
  hold on
  plot([-8 8],[0 0])
  plot([0 0],[-8 8])
  plot(Lmx(1,:), Lmx(2,:), 'r','linewidth',3)
end
```

Nun werden der Reihe nach verschiedene Abbildungen demonstriert: Spiegelung an x-Achse, an y-Achse, Punktspiegelung am Zentrum, Rotation um 90 Grad und mehrfache Rotation um je 30 Grad:

```
% Skript lmap22mat
Lur = [5 5 6; 2 0 0]
figure(1); clf
Mx = [1 0 ; 0 -1]
dispmap22(Mx,Lur)
figure(2)
My = [-1 0; 0 1]
dispmap22(My,Lur)
figure(3)
Pori = [-1 0; 0 -1]
dispmap22(Pori,Lur)
figure(4)
R90 = [0 -1; 1 0]
dispmap22(R90, Lur)
figure(5)
R30 = [0.866 -0.5; 0.5 0.866]
Lr1 = dispmap22(R30, Lur)
pause(2)  % return zum Weiterfahren
```

```
Lr2 = dispmap22(R30, Lr1);
Lr3 = dispmap22(R30, Lr2);
pause(2)
Lr4 = dispmap22(R30, Lr3);
Lr5 = dispmap22(R30, Lr4);
Lr6 = dispmap22(R30, Lr5);
Lr7 = dispmap22(R30, Lr6);
Lr8 = dispmap22(R30, Lr7);
Lr9 = dispmap22(R30, Lr8);
```

Definition des aus dem “L” erweiterten Dreibeins im Dreidimensionalen mit anschliessendem Plot

```
Dur = [0 0 1 0 0 1 0 0 ;...
       0 0 0 0 2 0 0 2; ...
       3 0 0 3 0 0 0 0 ]
plot3(Dur(1,:),Dur(2,:),Dur(3,:),'m')
axis equal
```

24.5 M-Files zu Abbildungen in homogenen Koordinaten

Einzel-Abbildungen der „L“-Figur Zum Definieren der „L“-Figur in homogenen Koordinaten dient das kleine M-File `ldef.m`

```
% ldef - Definition des Uebungs-"L"
%  in der globalen Variablen L
%   als 3x3 Matrix von 3 Spaltenvektoren
%vvvin 2D homogenen Koordinaten
L = [5 5 6; 2 0 0; 1 1 1];}
```

Das um „L“ original oder „Ltr“ nach einer Transformation grafisch darzustellen kann die Funktion `plothclin(L,'Farbe')` oder `plothclin(L)` aus dem M-File `plothclin.m` aufgerufen werden:

```
% plothandle = plothclin(Lmat,colorstring)
% Plot einer Linie aus einer Matrix von Spaltenvektoren
%   in 2D homogenen Koordinaten
function plothandle = plothclin(lmat,colorstring)
if exist('colorstring') == 0;  col = 'k';  else
                      col = colorstring;  end
plothandle = plot(lmat(1,:),lmat(2,:),col);
```

Die Festlegung geeigneter Achsen für diese Darstellungen ist ebenfalls in einem M-File zusammengefaßt: `stdhcaxis.m`

```
% stdhcaxis - Definieren von Standardbildbereich und Achsen
```

```
%  fuer Darstellungen von Transformationen in
%  2D homogenen Koordinaten
axis([-12 12 -12 12])  ; axis square
```

Unter Verwendung dieser Funktionen kann eine Abbildung des „L“ in 2D homogenen Koordinaten mit Hilfe der folgenden Befehlsfolge visualisiert werden (M ist dabei eine 3x3 Matrix für eine 2D-Transformation in homogenen Koordinaten, in diesem Beispiel eine 90° Drehung mit unvollständigem Zurückverschieben): `lrotexample.m`

```
% lrotexample.m = Befehle zum Definieren,
% Transformieren und Plotten des ``L''
M = [0 -1 4; 1 0 -5; 0 0 1];
ldef; % ab
Ltr = M*L;
plothclin(L,'k');  hold on;
stdaxis;
plothclin(Ltr,'r');
```

Mit der Methode der homogenen Koordinaten soll das Transformations-lern-L um -90 Grad (d.h. im Uhrzeigersinn) um seine Ecke gedreht werden. Das bedeutet Translation um `[-5 0]` (Ecke zu Nullpunkt) Rotation um -90 Grad `[0 1; -1 0]`, Rücktranslation zur ursprünglichen Ecke `[5 0]` .

```
%  Skript lroteckm90.m
%  Drehung des "L" um -90 Grad um seine Ecke
% Definition des L und Bereitstellen des Plots
L=[5 5 6; 2 0 0; 1 1 1];
plot([-10 10],[0 0]);  hold on;  plot([0 0],[-10 10]);
axis([-4 10 -4 10]);  axis square
plot(L(1,:),L(2,:),'k','linewidth',2)
% Translation der Ecke (5/0) in (0/0)
Ta=[1 0 -5; 0 1 0; 0 0 1];  La = Ta*L
plot(La(1,:),La(2,:),'r')
% Rotation  um -90 deg und Ruecktranslation
%  Verschiebung um [5 0]' direkt angehaengt
Tbc=[0 1 5 ; -1 0 0 ; 0 0 1]
% Gesamtabbildung mit Produktmatrix
Tg = Tbc*Ta ; Lg = Tg*L;
plot(Lg(1,:),Lg(2,:),'r','linewidth',3)
```

Das M-File zum schematisierten Industrieroboter

```
function robo3d(mpa)
% function robo3d(mpa) Schematisierter Indutrieroboter
%  Die Parameter mpa(1..6) sind:
% Achsen: 1. z in Fuss, 2. x Fuss-Armb, 3. x in armb-arme,
%  4. x in arme-Kopf, 5. z Kopf-Zange, 6. Zangenoeffnung
```

```
% mpa = [45 -65 135 50 25 1]; ergibt gutes Beispiel
yv =[0 2 4 4 2 -2 -4 -4 -2 0]; zv=[-4 -4 -2 62 64 64 62 -2 -4 -4];
blxv = [-4 4 4 0.5 0.5 -4 -4]; blzv = [-12 -12 5 5 -5 -5 -12];
zax = [0 2 2 1 0 0 0]; zaz = [0 0 3 7 7 3 0];
onv = @(v,n) v*ones(1,n); % Geometrische Definition der Glieder
zao =  [[zax; onv(3,7); zaz; onv(1,7)] [zax; onv(-3,7); zaz; onv(1,7)]];
zau =  [[-zax; onv(3,7); zaz; onv(1,7)] [-zax; onv(-3,7); zaz; onv(1,7)]];
blb = [[blxv; onv(-4,7); blzv; onv(1,7)] [blxv; onv(4,7); blzv; onv(1,7)]];
idxb = reshape(repmat(1:6,3,1),1,18); idxb(2:3:17)= idxb(2:3:17)+7;
idxb = [idxb 7 6 13 8:14]; %   unter Verwendung von Indexvektoren
ble = blb; ble(1,:)=-ble(1,:) ;  ble(3,:)=-ble(3,:);
arme = [ [onv(0.5,10); yv; zv; onv(1,10)] [onv(4,10); yv; zv; onv(1,10)]];
armb = [[-onv(0.5,10); yv; zv; onv(1,10)] [-onv(4,10); yv; zv; onv(1,10)]];
idxf = reshape(repmat(2:9,3,1),1,24); idxf(2:3:25)= idxf(2:3:25)+10;
idxf = [1, idxf, 10 2 12:20 12];
% Transformationen mit anonymen Funktionen
rotmatx = @(w) [1 0 0 0; 0 cosd(w) -sind(w) 0; 0 sind(w) cosd(w) 0; 0 0 0 1];
rotmatz = @(w) [cosd(w) -sind(w) 0 0;  sind(w) cosd(w) 0 0; 0 0 1 0; 0 0 0 1];
transmat = @(tx,ty,tz) [1 0 0 tx; 0 1 0 ty; 0 0 1 tz; 0 0 0 1];
% Teil-Transformationsmatrizen Tr1 (Stamm) bis Tr9 (Zange)
Tr1 = transmat(0,0,12); Tr2 = rotmatz(mpa(1));
Tr3 = rotmatx(mpa(2)); Tr4 = transmat(0,0,60); Tr5 = rotmatx(mpa(3));
Tr6 = Tr4; Tr7 = rotmatx(mpa(4)); Tr8 = transmat(0,0,12.3);
Tr9 = rotmatz(mpa(5));
Trxo = transmat(mpa(6),0,0);
Trxu = transmat(-mpa(6),0,0);
blbt = Tr1*Tr2*blb; arbt = Tr1*Tr2*Tr3*armb;
aret = Tr1*Tr2*Tr3*Tr4*Tr5*arme;
blet =  Tr1*Tr2*Tr3*Tr4*Tr5*Tr6*Tr7*ble;
zaot =  Tr1*Tr2*Tr3*Tr4*Tr5*Tr6*Tr7*Tr8*Tr9*Trxo*zao;
zaut =  Tr1*Tr2*Tr3*Tr4*Tr5*Tr6*Tr7*Tr8*Tr9*Trxu*zau;
% Grafik des Roboters in der verlangten Stellung
figure(1); clf
plot3(blbt(1,idxb),blbt(2,idxb),blbt(3,idxb),'m') ;
hold on; axis([-50 50 -50 50 0 100]); axis square
plot3(arbt(1,idxf),arbt(2,idxf),arbt(3,idxf)) ;
plot3(aret(1,idxf),aret(2,idxf),aret(3,idxf)) ;
plot3(blet(1,idxb),blet(2,idxb),blet(3,idxb),'r') ;
plot3(zaot(1,idxb),zaot(2,idxb),zaot(3,idxb),'k') ;
plot3(zaut(1,idxb),zaut(2,idxb),zaut(3,idxb),'k') ;
view(16,32)
```

25
M-Files zu Kapitel 5

25.1
M-Files zu unendlichen Reihen von Funktionen

Symbolisches Skript zur Bestimmung des Konvergenzradius mit derm Quotientenkriterium

```
%  Quotientenkriterium: Grenzwert bestimen
syms x n
qu = (x^(n+1)/x^n \cdot n/(n+1))
limit(qu,n,inf)
limfct = simplify(ans)
%  limfct = x % also Konvergenz bei |x| < 1
```

Beipiel einer Taylor-Entwicklung im symbolischen Modus.

```
% taylorexa.m  Beispiele zur symbolischen Funktion taylor
syms  a  x
fvers = a^3/(a^2+x^2)   % Versiera di Agnesi
tvers = taylor(fvers,x)
tverslg = taylor(fvers,x,'order',12)
%  tvers =  a - x^2/a + x^4/a^3
%  tverslg = a - x^2/a + x^4/a^3 - x^6/a^5 + x^8/a^7 - x^10/a^9
tlne1 =  taylor(log(x),x,1) % natürlicher Logarithmus, Ept = 1
%  tlne1 = x-1 - (x-1)^2/2 + (x-1)^3/3 - (x-1)^4/4 + (x-1)^5/5
tlog10e1 =  taylor(log10(x),x,1); pretty(tlog10e1) % 10-er Log.
%                   2           3           4           5
%  x - 1      (x - 1)      (x - 1)      (x - 1)      (x - 1)
% ------- - --------- + --------- - --------- + ---------
% log(10)   2 log(10)   3 log(10)   4 log(10)   5 log(10)
```

Die komplexe Taylor-Entwicklung bon exp(i*x) zeigt die Identität exp(i*x) = cos(x) + i* sin(x)

```
% taylorimag.m  Taylorentwicklung komplex
texp = taylor(exp(i*x),'order',8); % exp komplex
tcos = taylor(cos(x),'order',8);   % cos = Realteile
tsin = taylor(sin(x),'order',8);   % sin = Imaginärteile
```

```
pretty(texp) , pretty(tcos), pretty(tsin)
%    7        6    5       4    3       2
%   x  1i    x    x  1i   x    x  1i   x
% - ----- - --- + ----- + -- - ----- - -- + x 1i + 1
%    5040   720    120    24     6      2
%
%             6            4            2
%            x            x            x
%         - ---         + --         - --        + 1
%           720           24            2
%
%      7             5           3
%     x             x           x
%  - ----        + ---        - --        + x
%    5040           120           6
```

Potenzreihen-Integration der Fehlerfunktion `int(exp(-x^2))`

```
% potrint.m Potenzreihen Integration erf(x)
syms x t
nor = exp(-x^2)
nort8  = taylor(nor,x,0,'order',9)
nort16 = taylor(nor,x,0,'order',17)
nort32 = taylor(nor,x,0,'order',33)
%  formale Integration der Potenzreihen
erint8  = 2/sqrt(pi)*int(nort8,x,0,t)
erint16 = 2/sqrt(pi)*int(nort16,x,0,t)
erint32 = 2/sqrt(pi)*int(nort32,x,0,t)
% gemeinsame Grafik der verschiedenen Approximationen
ezplot(erint8,-3,3) ; hold on ;
ezplot(erint16,-3,3); ezplot(erint32,-3,3) ;
ezplot(erf(t),-3,3);  axis([-3 3 -1.1 1.1]); hold off
```

25.2
M-Files zu Orthogonalpolynomen

Polynom-Fits an die Funktion sin(x)/x ergeben bei zunehmendem Grad immer neue Werte der tieferen Koeffizienten, weil die Potenzfunktionen nicht orthogonal sind.

```
% sincmultifit.m
% verschiedene Polynomfits an sin(x)/x [-4.5 4.5]
xv = -4.5:0.2:4.5; yv = sin(xv)./xv; xf = xv/4.5;
cmat = zeros(6,13); % Bereich auf [-1 1] reduziert
cmat(1,11:13) = polyfit(xf,yv,2);
cmat(2,9:13)  = polyfit(xf,yv,4);
```

```
cmat(3,7:13)  = polyfit(xf,yv,6);
cmat(4,5:13)  = polyfit(xf,yv,8);
cmat(5,3:13)  = polyfit(xf,yv,10)
cmat(6,1:13)  = polyfit(xf,yv,12);
gmat = cmat; gmat(:,2:2:12)=[]  % nur gerade Koeff
tr = taylor(sin(9*x/2)/(9*x/2),x,0,'order',13);
sym2poly(tr)
```

Bei Fourier-Fits sind die tieferen Koeffizienten viel stabiler:

```
% sincfoufit.m
% verschiedene Fourier-Fits an sin(x)/x [-4.5 4.5]
xv = -4.5:0.05:4.5; yv = sin(xv)./xv; xf = xv/4.5;
yv((length(xf)+1)/2) = 1; % hebbare Singularitaet
xw = (pi*xv(1:end-1))'; yw = yv(1:end-1)';
% Bereich auf [-1 1] reduziert. Funktion ist gerade!
% Design-Matrix der Funktionenfits nur nach cos(n*x)
Y = [ones(length(xf)-1,1) cos(xw) cos(2*xw) cos(3*xw)...
        cos(4*xw)  cos(5*xw) cos(6*xw)];
% sukzessive Anwendung des linken Teils der Design-Matrix
cs2 = Y(:,1:2)\yw, cs3 = Y(:,1:3)\yw,cs4 = Y(:,1:4)\yw,
cs5 = Y(:,1:5)\yw, cs6 = Y(:,1:6)\yw, cs7 =  Y(:,1:7)\yw
```

Aufbau der Tschebyscheff-Polynomen 1. Art durch orthogonale Konstuktion aus denjenigen mit tieferem Grad.

```
% tschebyorth.m  Aufbau orthogonale T-Polynome
syms x a b c d
w(x) = 1/sqrt(1-x^2); t0 = 1 ; t1 = x ;
% Ansatz t2, allgemein
t2 = a*x^2 + b*x + c   % t2 allgemein 2.Grades
eq1 = int(t0*t2*w,x,-1,1)==0 % t2 orth t0
% eq1 = (pi*(a + 2*c))/2 == 0 ; d.h. a = -2*c
eq2 = int(t1*t2*w,x,-1,1)==0 % t2 orth t1 => b=0
t2 = 2*x^2 - 1; % c = -1 gewählt, dann a= 2
% t2 festgelegt auf t2 = 2*x^2 - 1, t3 Ansatz allgemein
t3 = a*x^3 + b*x^2 + c*x + d % t3 allgemein 3.Grades
eq3 = int(t2*t3*w,x,-1,1)==0 % t3 orth t2 => b=0
eq4 = int(t0*t3*w,x,-1,1)==0 % t3 orth t0; b+2d=0 => d=0
eq5 = int(t1*t3*w,x,-1,1)==0 % t3 orth t1; 3a+4c = 0
t3 = 4*x^3 - 3*x
%  t3 festgelegt auf t3 = 4x^3 - 3x = 2x(2x^2-1) - x
%  Rekursive Definitiion einige Schritte weiter
t4 = simplify(collect(2*x*t3 - t2))
%  t4 = 8*x^4 - 8*x^2 + 1
t5 = expand(2*x*t4 - t3)
%  t5 = 16*x^5 - 20*x^3 + 5*x
```

```
t6 = simplify(collect(2*x*t5 - t4))
%  t6 = 32*x^6 - 48*x^4 + 18*x^2 - 1
```

T-Polynome aus der Definition mit den Mehrwinkel-Formeln herleiten:

```
% tschebycos.m T-Polynome aus Mehrwinkelformeln
syms x w   % allg:  Tn = cos( n* acos(x) )
t2 = cos(2*w); t2c = expand(t2); subs(t2c,cos(w),x)
%  T2 = 2*x^2 - 1
t3 = cos(3*w); t3c = expand(t3); subs(t3c,cos(w),x)
%  T3 = 4*x^3 - 3*x
t4 = cos(4*w); t4c = expand(t4); subs(t4c,cos(w),x)
%  T4 = 4*x^3 - 3*x
t5 = cos(5*w); t5c = expand(t5); subs(t5c,cos(w),x)
%  T5 = 16*x^5 - 20*x^3 + 5*x
```

25.3 M-Files zu Diskreter Fourier-Transformation und FFT

Die schrittweise Halbierung der Länge der zu transformierenden Folge bewirkt die Reduktion der Anzahl Operationen bei der schnellen Fouriertransformation FFT. Die einzelnen Schritte können durch Betrachten der Matrizen für die Zwischenstufen sichtbar gemacht werden.

```
% shofftalgo.m Skript zur Demonstration einer FFT
% Die globale Variable 'ndim' muss vordefiniert werden
%  die globale Vektorfunktion 'fin' kann, muss aber nicht
%  vordefiniert sein. Wenn nicht, wird ein Zufallsvektor erzeugt
%
% in den Stufen x=2,4,8 sind
%    Ex und Ox fuer die Zusammenfassungen Even/Odd
%    Tx fuer die Transformation
%    Ux fuer Umkehrung der Anordnung unscramble
%    Fxs die Resultate direkt nach Tx, scrambled
%    F ist das Original fft-Resultat zum Vergleich
%    GUx die zusammengefassten g und u-Werte
%
%
   if exist('fin') ==1
% falls eine Vektorfunktion 'fin' existiert gilt diese
     f = fin;
   else
% die besten allgemeinen Testf\"alle sind Zufallsvektoren
      f = rand(ndim,1);
   end
   F = fft(f);
% 1. -- ohne Zerlegung
   difvec1 =  dftmatrix(ndim)*f - fft(f);
disp(' 1 -- Differenz DFT volle Matrix - FFT')
   difsum1 = sum(abs(difvec1))
```

```
% 2. -- eine Zerlegung  in ndim/2 lange Teile
if ndim > 2
  [U2, T2, E2, O2] = hfftmatrix(ndim);
  GU2 = (E2+O2)*f;
  F2s = T2*(E2+O2)*f;
  difvec2 = U2*T2*(E2+O2)*f - fft(f);
disp(' 2 -- Differenz DFT 1 Zerlegung - FFT')
   difsum2 = sum(abs(difvec2))
end
% 3. -- 2 Zerlegungen in ndim/4 lange Teile
if ndim > 4
  [U4, T4, E4, O4] = xfftmatrix(ndim/2,2);
  GU4 =  (E4+O4)*(E2+O2)*f ;
  F4s = T4*(E4+O4)*(E2+O2)*f ;
  difvec3 = U2*U4*T4*(E4+O4)*(E2+O2)*f - fft(f);
disp(' 3 -- Differenz DFT 2 Zerlegungen - FFT')
   difsum3 = sum(abs(difvec3))
end
% 4. -- 3 Zerlegungen in ndim/8 lange Teile
if ndim > 8
  [U8, T8, E8, O8] = xfftmatrix(ndim/4,4);
  GU8 = (E8+O8)*(E4+O4)*(E2+O2)*f ;
  F8s = T8*(E8+O8)*(E4+O4)*(E2+O2)*f ;
  difvec4 = U2*U4*U8*T8*(E8+O8)*(E4+O4)*(E2+O2)*f - fft(f);
disp(' 4 -- Differenz DFT 3 Zerlegungen - FFT')
   difsum4 = sum(abs(difvec4))
end
```

Darin werden die folgenden Funktionen aufgerufen:
dftmatrix.m

```
function dfm = dftmatrix(nel)
 w =  exp(-i*2*pi/nel);
 dfm = ones(nel);
 for k=2:nel
  for n=2:nel
    dfm(k,n) = w^((k-1)*(n-1));
  end
 end
```

und die Funktion
xfftmatrix.m
Diese produziert die 4 Matrizen U (unscramble), F (mehrere verkürzte DFT), E (Zusammenfassen der Even-Terme), O (Odd-Terme). Die Teilmatrizen der Dimension 'nel' sind 'nr' mal repetiert.

```
function [usc,fm,sme,smo]=xfftmatrix(nel,nr)
     vv= exp(-i*2*pi/nel);
 fmp = zeros(nel);
 smop = fmp;
 smep = fmp;
 uscp = fmp;
```

```
 for k=1:nel/2
   smep(k,k)= 1;
   smep(k,k+nel/2)= 1;
 end
 for k=1:nel/2
   smop(k+nel/2,k)= vv^(k-1);
   smop(k+nel/2,k+nel/2)= -vv^(k-1);
 end
fmp = [dftmatrix(nel/2)   zeros(nel/2)      ; ...
       zeros(nel/2)       dftmatrix(nel/2) ];
 for k=1:nel/2
   uscp(2*k-1,k)= 1;
   uscp(2*k,k+nel/2)= 1;
 end
 fm = zeros(nel*nr);
 smo = fm;
 sme = fm;
 usc = fm;
nini = nel;
for l=1:nr
 for r=1:nel
  for c = 1:nel
    fm(r+(l-1)*nel,c+(l-1)*nel)= fmp(r,c);
    sme(r+(l-1)*nel,c+(l-1)*nel)= smep(r,c);
    smo(r+(l-1)*nel,c+(l-1)*nel)= smop(r,c);
    usc(r+(l-1)*nel,c+(l-1)*nel)= uscp(r,c);
  end
 end
end
```

Das Spiegelungsprinzip der DFT wird mit foumirror grafisch demonstriert:

```
nel = input('Eingabe Anzahl Abtast-Stellen: ');

harm = input('Eingabe der betrachteten Harmonischen: ');

% die Winkel-Werte der Abtast-Punkte und ihre sin und cos
dw = 360/nel;
wt = pi/180*(0:dw:360);
st = sin(harm*wt);
ct = cos(harm*wt);

%
wi = pi/180*(0:2:360);
so = sin(harm*wi);
sm = -sin((nel-harm)*wi);
```

```
co = cos(harm*wi);
cm = cos((nel-harm)*wi);

clf
subplot(2,1,1)
plot(wi,co,'b')
hold on
plot(wi,cm,':k')
plot(wt,ct,'.r')

subplot(2,1,2)
plot(wi,so,'g')
hold on
plot(wi,sm,':k')
plot(wt,st,'.r')
```

25.4
M-Files: Kennenlernen der Fourier-Transformation

Mit den verschiedenen fdef... Skripts kann jeweils die Funktion 'f' vordefiniert werden und anschliessend die Fourier-Analyse dazu aufgerufen werden.

```
% fdefcos.m Skript f = cos(0:dw:2pi-dw)
xvector = 0:99;
f = cos( xvector/50*pi); % 100 Elemente
```

```
% fdefhat.m Skript f - Hutfunktion,
%    Breite und Position wird interaktiv abgefragt
f = zeros(1,100);
disp('Definition einer Hut-Funktion durch Angabe von');
disp('  Position und Breite');
pos = input(' Eingabe Position der Hut-Spitze (0..99):');
wid = input(' Eingabe halbe Breite des Spitz-Hutes:');
f(pos+1) = 1;
for k=pos+2:min(pos+1+wid,100)
     f(k) = 1-(k-pos-1)/wid;
end
for k=max(pos+2-100,1):pos+1+wid-100
     f(k) = 1-(k-pos-1+100)/wid;
end
for k=pos:-1:max(pos-wid+1,1)
     f(k) = 1+(k-pos-1)/wid;
end
for k=min(pos+100,100):-1:pos-wid+1+100
```

```
    f(k) = 1+(k-pos-1-100)/wid;
end

% fdefmedpeak.m Skript f = Rechteckpuls 9 breit,
%   100 Elemente
f = [ 1 1 1 1 1  zeros(1,91)  1 1 1 1];

% fdefminisaegz.m Skript f= Saegezahn-Funktion,
%    Phasenverschoben
h = [ 0:20 -20:-1  ]/20 ;
f = [ h(10:41) h(1:9)] ;  41 Elemente

% fdefnarrpeak.m Skript f = Rechteckpuls 3 breit,
%  100 Elemente
f = [ 1 1   zeros(1,97)   1];

% fdefpeak.m Skript f = Rechteckpuls 5 breit, 100 Elemente
f = [ 1 1 1 zeros(1,95) 1 1];

% fdefpeak.m Skript f = Rechteckpuls 5 breit, 100 Elemente
f = [ 1 1 1 zeros(1,95) 1 1];

% fdefsaegz.m Skript f= Saegezahn-Funktion, ungerade
f = [ 0:40 -40:-1  ]/40 ;   % 81 Elemente

% fdefsin.m Skript f = sin(0:dw:2pi-dw)
xvector = 0:99;
f = sin( xvector/50*pi); % 100 Elemente

% fdefsquare.m - Skript f= quadrat. Puls, gerade, a0 klein
f = [ 1 ones(1,20) -1*ones(1,40) ones(1,20)]; % 80 Elemente

% fdeftriang.m - Skript f = Zickzack-Funktion, ungerade
f = [ 0:20 19:-1:-20 -19:-1 ] ; % 80 Elemente

% fdeftriangl.m - Skript f = Zickzack-Funktion, ungerade
f = [ 0:25 24:-1:-25 -24:-1 ] ; % 100 Elemente

% fdefwidpeak.m Skript f = Rechteckpuls 15 breit,
%   100 Elemente
f = [ 1 1 1 1 1 1 1 1 zeros(1,85) 1 1 1 1 1 1 1];
```

Von der mit einem der obigen Skripts vordefinierten Funktion ’f’ werden nun verschiedene Fouriertransformationen durchgeführt und dargestellt.

```
% die einfachste verfuegbare Fourier-Transformation ist fft
ft = fft(f);
n = length(f);
```

```
% Rechen-Ungenauigkeits-Effekte werden Null gesetzt
if max(real(ft)) < 1e-12
  ft = j*imag(ft);
end
if max(imag(ft)) < 1e-12
  ft = real(ft);
end
%
% ganz gewoehnliche Fourier-Koeffizienten
ftn = 2*ft/n;
ibas=1:n/2;
clf
subplot(2,1,1)
bar(ibas,real(ftn(1:n/2)))
subplot(2,1,2)
bar(ibas,-imag(ftn(1:n/2)))
```

```
% foushow.m -  Demo Skript zum Zeichnen der verschiedenen
%  Darstellungsformen der Fourierkoeffizienten
% die Zeitfunktion muss im Vektor   f  vordefiniert werden
ft = fft(f);  n = length(f);
% die einfachste verfuegbare Fourier-Transformation ist fft
fmi = min(f); fma = max(f);
% Rechen-Ungenauigkeits-Effekte werden Null gesetzt
if max(real(ft)) < 1e-12;  ft = j*imag(ft);  end
if max(imag(ft)) < 1e-12 ; ft = real(ft);  end
% 1. ganz gewoehnliche Fourier-Koeffizienten
ftn = 2*ft/n;  nh = floor(n/2);  ibas=1:nh;
figure(1);  clf;  figure(2);  clf
figure(1) ; subplot(2,1,1);  bar(ibas,real(ftn(1:nh)))
subplot(2,1,2);  bar(ibas,-imag(ftn(1:nh)))
figure(2);  plot(f)  % Die Zeitfunktion erscheint in Figur 2
axis([-3 n+3 fmi-0.15*(fma-fmi) fma+0.15*(fma-fmi)])
figure(1)
% Stop durch Tastatur-Abfrage
cont = input('1. in Fig. 1: a und b-Koeffizienten - weiter?');
clf;  bar(ibas,abs(ftn(1:nh)))
% 2.  Power-Spektrum,  Stop durch Tastatur-Abfrage
cont = input('2. im Fig. 1: Power-Spektrum -weiter?');
%    alle diskret - Transformierten Koeffizienten
igew = 1:n; subplot(2,1,1); bar(igew,real(ft))
subplot(2,1,2); bar(igew,-imag(ft))
% 3. DFT Koeffizienten, Stop durch Tastatur-Abfrage
cont = input('3. im Fig. 1: DFT-Koeffizienten - weiter?');
% 4. Komplexe Darstellung
ftr = rearrg(ft); lorig = length(f);
idx = -floor(lorig/2):floor(lorig/2); ftrn = ftr/n;
clf; subplot(2,1,1) ; bar(idx,real(ftrn))
subplot(2,1,2); bar(idx,imag(ftrn))
disp('4. im Fig. 1: Re, Im der komplexen Koeffizienten - Show fertig')
```

```
% foushowexp.m Darstellung der Fourier-Koeffizienten
%   zur vordefinierten Funktion f
% in der komplexen Exponenten-Form ck*exp(j*k*w), k=-N..+N,
ft = fft(f); n = length(f);
ftr = rearrg(ft); lorig = length(f);
idx = -floor(lorig/2):floor(lorig/2);
rcoef = real(ftr);
% Rechen-Ungenauigkeits-Effekte werden Null gesetzt
if max(rcoef) < 1e-12
  rcoef = zeros(1,length(ftr));
end
xicoef = imag(ftr);
if max(xicoef) < 1e-12
  xicoef = zeros(1,length(ftr));
end
% plot in zwei Teilbildern uebereinander
subplot(2,1,1);  bar(idx,rcoef)
subplot(2,1,2);  bar(idx,xicoef)
```

```
% foushowexp.m Darstellung der Fourier-Koeffizienten
%   zur vordefinierten Funktion f
% in der komplexen Exponenten-Form ck*exp(j*k*w), k=-N..+N,
ft = fft(f); n = length(f);
ftr = rearrg(ft); lorig = length(f);
idx = -floor(lorig/2):floor(lorig/2);
rcoef = real(ftr);
% Rechen-Ungenauigkeits-Effekte werden Null gesetzt
if max(rcoef) < 1e-12
  rcoef = zeros(1,length(ftr));
end
xicoef = imag(ftr);
if max(xicoef) < 1e-12
  xicoef = zeros(1,length(ftr));
end
% plot in zwei Teilbildern uebereinander
subplot(2,1,1);  bar(idx,rcoef)
subplot(2,1,2);  bar(idx,xicoef)
```

```
% Demonstration des Gibbs'schen Phaenomens
% 1. Input globale Zahlenfolge (Start-Funktion):  f
% 2. Input globaler Wert:  nkshow
%     (hoechste Harmonische vor dem Abschneiden)
%
%  Fourier-Transformierte bestimmen
ft = fft(f);
```

```
ltot = length(f);
fmi = min(f);
fma = max(f);
% zu grosse nkshow ergeben kein clipping
nksh = min([nkshow , (ltot-1)/2 ]);
% Zusammenstellen der Fourier-Koeffizienten
% nach dem Abschneiden
ftc =  zeros(1,ltot);
ftc(1) = ft(1);
for k=1:nksh
  ftc(1+k) = ft(1+k);
  ftc(ltot+1-k) = ft(ltot+1-k);
end
% Ruecktransformation
fc = ifft(ftc);
plot(real(fc),'r')
axis([-3 ltot+3 fmi-0.3*(fma-fmi) fma+0.3*(fma-fmi)])
hold on
plot(f)
hold off

% Demonstration des Gibbs'schen Phaenomens
% 1. Input globale Zahlenfolge (Start-Funktion):  f
% 2. Input globaler Wert:  nkshow
%     (hoechste Harmonische vor dem Abschneiden)
%
%  Fourier-Transformierte bestimmen
ft = fft(f);
ltot = length(f);
fmi = min(f);
fma = max(f);
% zu grosse nkshow ergeben kein clipping
nksh = min([nkshow , (ltot-1)/2 ]);
% Berechnen der Lanczos Abschwaechungsfaktoren
xwgt = 0:nksh-1;
wgt = cos(xwgt*pi/2/(nksh-1));
% Zusammenstellen der Fourier-Koeffizienten
% nach dem Abschneiden
ftc =  zeros(1,ltot);
ftc(1) = ft(1);
for k=1:nksh
  ftc(1+k) = ft(1+k)*wgt(k);
  ftc(ltot+1-k) = ft(ltot+1-k)*wgt(k);
end
% Ruecktransformation
```

```
fc = ifft(ftc);
plot(real(fc),'r')
axis([-3 ltot+3 fmi-0.3*(fma-fmi) fma+0.3*(fma-fmi)])
hold on
plot(f)
hold off
```

25.5 M-Files zur einfachen Faltung

```
function c = formfalt(a,b)
n1 = length(a) ; n2 = length(b); nr = n1+n2-1;
for j=1:nr
  k1 = max(1,j-n2+1); k2 = min(n1,j);
  c(j) = 0;
  for k=k1:k2
    c(j)= c(j)+a(k)*b(j-k+1);
  end
end

function c = formfaltz(a,b)
% c = formfaltz(a,b) zirkulaere Faltung nach Formel
n1 = length(a) ; n2 = length(b);  al = a; bl = b;
if n1 < n2
  al = [al zeros(1,n2-n1)];   n1 = n2;
elseif n1 > n2
  bl = [bl zeros(1,n1-n2)];
end
for j=1:n1
  c(j) = 0;
  for k=1:j
    c(j)= c(j)+bl(k)*al(j-k+1);
  end
  for k=j+1:n1
    c(j)= c(j)+bl(k)*al(j-k+n1+1);
  end
end
```

25.6
M-Files zur Zirkulären Faltung und dem Faltungssatz

Die Formulierung dieses Verfahrens in MATLAB ergibt, in kleine Schritte aufgelöst, die folgende Befehlssequenz für die Faltung der Folgen a und b:

```
% FFT-Transformation der einzelnen Folgen
  atr = fft(a);
  btr = fft(b);
% Hadamard Produkt der transformierten Folgen
  ctr = atr .* btr;
% Ruecktransformation
  c = ifft(ctr);
% Damit ist c die zyklische Faltung von a und b
```

Diese zirkuläre Faltung mit dem Umweg über die schnelle Fouriertransformation kann auch viel kompakter geschrieben werden als:

```
c = ifft( fft(a) .* fft(b) )
```

Für eine gewöhnliche Faltung darf man das zero-padding nicht vergessen: fftfaltzeropad.m

```
Skript fftfaltzeropad.m
% Gewoehnliche Faltung mit Zero-Padding und FFT
%    im Vergleich zur Bibliotheksprozedur conv()
n = length(a);
m = length(b);
% zu a mit der Laenge n kommen m-1 Nullen dazu
az = [ a zeros(1,m-1) ]
% zu b mit der Laenge m kommen n-1 Nullen dazu
bz = [ b zeros(1,n-1) ]
% Somit haben az, bz und c je die Laenge m+n-1
c = ifft( fft(az) .* fft(bz) );
cbib = conv(a,b);
% der Vergleich mit der Bibliotheksfunktion gibt 0
diff = sum( abs(c -cbib) )

function c = fftfaltzirk(a,b)
% c = fftfaltzirk(a,b) zirkulaere Faltung nach Faltungssatz
n1 = length(a) ; n2 = length(b);  al = a; bl = b;
if n1 < n2  % Laengen-Test mit Nullen nachfuellen
  al = [al zeros(1,n2-n1)];   n1 = n2;
elseif n1 > n2
  bl = [bl zeros(1,n1-n2)];
end
c = real(ifft(fft(al).*fft(bl)));
```

In verschiedenen Faltungs-Algorithmen wird die Funktion shrg verwendet:

```
% vback = shrg(vin,nsh) - Rechts-Verschiebung zirkulaer
function vback = shrg(vin,nsh)
  ntot = length(vin);
  nlast = ntot -nsh;
  vback = [ vin(nlast+1:ntot) , vin(1:nlast) ];
```

26
M-Files zu Kapitel 6

26.1
M-Files zum Bilden von partiellen Ableitungen

Symbolisches Skrpt zum Bestimmen des Gradienten im elektrischen Potential:
epot.m

```
% epot.m  Gradient elektrisches Potential
syms x y z Q ep0
P = Q/(4 * pi *ep0) * 1/sqrt(x^2+y^2+z^2)
gx = diff(P,x), gy = diff(P,y), gz = diff(P,z)
%  gx = -(Q*x)/(4*pi*ep0*(x^2 + y^2 + z^2)^(3/2))
%  gy = -(Q*y)/(4*pi*ep0*(x^2 + y^2 + z^2)^(3/2))
%  gz = -(Q*z)/(4*pi*ep0*(x^2 + y^2 + z^2)^(3/2))
% Dafuer existiert auch eine direkte Funktion
E = gradient(P,[x y z])
```

26.2
M-Files zu Höhenlinien- und Flächenplots

Surface-Plot im symbolischen Modus, das Atoll im blauen Meer:
atollplot.m

```
% atollplot.m ezsurf-Grafik im symbolischen Modus
%  Eine ringfoermige Insel ragt aus dem blauen Meer
syms x y real
ezsurf(exp(-abs(sqrt(x^2+y^2)-5)),[-8 8 -8 8])
colormap jet; axis equal
axis off
```

Illustration der Funktionsweise der Funktion meshgrid meshgrideffekt.m

```
% Einfacher Aufruf von meshgrid
[Xg , Yg] = meshgrid( 1:4 , 5:7 )
```

```
%  x-Positionen 1 bis 4
%  y-Positionen 5 bis 7
% Als Resultat erhaelt man die vertikale Streifenmatrix
%Xg =
%     1  2  3  4
%     1  2  3  4
%     1  2  3  4
% und die horizontal gestreifte Matrix
%Yg =
%     5  5  5  5
%     6  6  6  6
%     7  7  7  7
```

Höhenlinien und darin eingezeichnete Gradienten
sinsingradi.m

```
% sinsingradi.m  Kontur und Gradientplot
% der Funktion sin(pi*x)*sin(pi*y)
[Xf,Yf] = meshgrid(-1:0.05:1,0:0.05:1);
Zf = sin(pi*Xf).*sin(pi*Yf);
contour(Xf,Yf,Zf,16,'k','linewidth',1);
axis equal; hold on
%  Numerische Bildung des Gradienten
[gx,gy ]= gradient(Zf);
% Ausduennen der Pfeilgrafik
sel = repmat([0 0; 0 1],11,21);
sel(22,:)=[]; sel(:,42)=[]; sl = sel == 1;
quiver(Xf(sl),Yf(sl),gx(sl),gy(sl),'linewidth',1)
hold off
```

Das M-File zum Titelbild auf dem Buchdeckel mit aufeinander bezogenen Flächen- und Konturplots einer Funktion mit mehreren Typen von stationären Punkten:
coverplot.m

```
% Figur starten
  figure(1) ; clf;
% Gitter der Variablen-Werte und Funktion Z(X,Y)
  [X,Y] = meshgrid(-0.1:0.1:2.8,0:0.2:4.8);
  Z = (sin(X)+0.18979*cos(5*X)).*sin(Y).*exp(-0.35*Y)+0.5;
% 3D Darstellung;  weiter zeichnen
  surf(X,Y,Z); hold on; colormap jet
% Rahmen-Viereck
  levx = [X(1,1) X(25,1) X(25,30) X(1,30) X(1,1)];
  levy = [Y(1,1) Y(25,1) Y(25,30) Y(1,30) Y(1,1)];
  plot3(levx,levy,0.5*ones(5,1),'k');  plot(levx,levy,'k');
% Markier-Kreuze der stationaeren Punkte
  xlinx = [-0.08 0.08 0 0 0];  xliny = [0 0 0 -0.1 0.1];
```

```
% lokale (anonyme) Funktion definieren und anwenden
  putmrk = @(x,y) plot(xlinx+x,xliny+y,'k','linewidth',2);
  putmrk(0.327,1.235); putmrk(1.3114,1.235);
  putmrk(2.347,1.235);
  putmrk(1.969,1.235); putmrk(1.3114,4.376);
% 3D Achsen-Box-Bereiche, mit Kamerawinkeln
  axis([-0.3 3 -0.4 5.2 0 1.2]); grid off; view(-58, 29);
% zugehoerige Hoehenlinien-Grafik
  contour(X,Y,Z,20,'-','linewidth',1.5)
% Papierformat, Aufloesung
  set(gca,'FontSize',16)
  set(gcf,'PaperType','A4')
  print -djpeg -r450 stathill
```

26.3 M-Files zur Ausgleichsrechnung

Fit-Funktion zum Bestimmen des besten allgemeinen Polynoms durch eine Anzahl gegebene Punkte

```
function koef = mulpolyfit(x,y,nk)
% function koef = mulpolyfit(x,y,nk)
% koef = mulpolyfit(x,y,nk)
% Fit nach Polynom-Grad nk-1
%    hat nk Koeffizienten als Resultat
% Berechnung mit  Normalengleichungen
% Vorbereitung Matrix und rhs-Vektor
N = zeros(nk);
b = zeros(nk,1);
koef=zeros(nk,1);
% Doppelschleife
for zei=1:nk
  eb = nk-zei;
  b(zei)=sum(y.*x.^eb);
  for spa=1:nk
    ex = 2*nk-zei-spa;
    N(zei,spa) = sum(x.^ex);
  end
end
koef=N\b;
```

Geradenfit durch 4 Punkte mit Fehlergleichungs-Methode

```
%Eingeben der Punktkoordinaten als 2 Spaltenvektoren
x = [1   2 3 4]';
```

```
y = [1.9 2 3 3]';
% Definition der ueberbestimmten Fehlergleichungen
A = [ x ones(4,1)]
% Loesung mit Links-Division wie beim normalen System
p = A\y
plot(x,y,'+') ; hold on; axis([0 5 0 5])
plot([0 10] , [p(2) p(2)+10*p(1)] ) ; hold off
```

Allgemeiner Polynomfit mit Fehlergleichungen gelöst.

```
function koef = fglpolyfit(x,y,nkoef)
% fglpolyfit:  koef = fglpolyfit(x,y,nkoef)
% Polynomfit an Punkt x,y mit Fehlergleichungsmethode
A = [ones(length(x),1)];
for k=1:nkoef-1
  A= [x.^k A];
end
koef = A\y;
```

Eine kleine einfache Anwendung davon auf einen Fit nach einem Polynom 3. Grades durch 8 gegebene Punkte:

```
x = (1:8)'; y = [ 0 2 3 1 0 -1 2 4]';
ko= fglpolyfit(x,y,4);   xx = 0:0.2:10;
yy = ko(1)*xx.^3 + ko(2)*xx.^2 +  ko(3)*xx + ko(4)
plot(x,y,'+'); hold on ;  plot(xx,yy)
```

26.4 M-Files zur Methode der Lagrange-Multiplikatoren

```
% hillagrange.m Figur zu Optimierung mit Nebenbedingung
xv = -6:0.1:6;
[xg,yg]=meshgrid(xv);
H =  10 - 0.02*xg.^2 - 0.1*yg.^2;
H = H .* (yg > xg-4) ; H=max(H,9);
figure(1) ; clf;  surf(xg,yg,H); view(15.5,11)
figure(2) ; clf
contour3(xg,yg,H)  ; hold on
LM = [-0.04 0 1 ; 0 -0.2 1 ; 1 -1 0]; Lb = [ 0 0 4]';
p= LM\Lb
xel = 0:0.1:6;  yel = -4+xel;
zel =  10 - 0.02*xel.^2 - 0.1*yel.^2;
zel = max(zel,9); plot3(xel,yel,zel,'k')
view(15.5,18.0) ; hold off
```

26.5
M-Files zu Nichtlinearen Gleichungssystemen

Suche von stationären Punkten mit der zweidmensionalen Newton-Iteration

```
function xypath = statpt2d(xst,yst)
% function xypath = statpt2d(xst,yst)
% statpt2d.m Suche stationären Punkt von F(x,y)
% Iterationsstart bei xst,yst, Suchpfadmatrix xypath
  syms x y  % Funktion, part. Abl. , Jacobian symbolisch
  F(x,y) = (sin(x)+0.18979*cos(5*x))*sin(y)*exp(-0.35*y)+0.5;
  dpFx = diff(F,x);  dpFy = diff(F,y);
% partielle Ableitungen
  Jsym = jacobian([dpFx, dpFy], [x,y]);
  rac = [xst;yst] ; xypath = rac'; % Startwert
  for k=1:7
% Fuer numerische Iteration: Umwandlung in double
    dFv = [double(subs(dpFx,[x,y],[rac(1),rac(2)])); ...
              double(subs(dpFy,[x,y],[rac(1),rac(2)]))];
    Jnum = double(subs(Jsym,[x,y],[rac(1), rac(2)]));
    dr = -Jnum\dFv; rac = rac+dr;
% Newton 2D Verbesserung
    disp([rac',dFv']), xypath = [xypath ; rac'];
  end
end
```

```
function fvec = fct2p2p4(x,y)
fvec = zeros(2,1);
fvec(1) = (x-4)^2+ y^2 -10;
fvec(2) = x^4+ y^4 -20;
%
function jac = fct2p2p4jac(x,y)
jac = zeros(2);
jac(1,1) = 2*x-8;
jac(1,2) = 2*y;
jac(2,1) = 4*x^3;
jac(2,2) = 4*y^3;
%
fvec= fct2p2p4(x(1),x(2))
delx = fct2p2p4jac(x(1),x(2))\ fct2p2p4(x(1),x(2))
x = x-delx
```

```
%[x,y]= funkfeuer(d1,d2) 2D Newton Iteration
function [xpos,ypos] = funkfeuer(d1,d2)
xv = [5 5]'; dv = [d1;d2];
```

```
for iter =1:8
dx = funkfeuerjac(xv)  \ funkfeuerfct(xv,dv);
xv = xv -dx;
end
xpos = xv(1); ypos = xv(2);

% subfunction funkfeuerfct
function fmom = funkfeuerfct(xv,dv)
fmom=zeros(2,1);
fmom(1) = sqrt(xv(1)^2+xv(2)^2) - ...
     sqrt(xv(1)^2+(xv(2)-10)^2) - dv(1);
fmom(2) = sqrt(xv(1)^2+xv(2)^2) -  ...
      sqrt((xv(1)-10)^2+xv(2)^2) - dv(2);

% subfunction funkfeuerjac
function jbk = funkfeuerjac(xv)
jbk(1,1) = xv(1)/sqrt(xv(1)^2+xv(2)^2) - ...
     xv(1)/sqrt(xv(1)^2+(xv(2)-10)^2);
jbk(1,2) = xv(2)/sqrt(xv(1)^2+xv(2)^2) - ...
     (xv(2)-10)/sqrt(xv(1)^2+(xv(2)-10)^2);
jbk(2,1) = xv(1)/sqrt(xv(1)^2+xv(2)^2) - ...
     (xv(1)-10)/ sqrt((xv(1)-10)^2+xv(2)^2);
jbk(2,2) = xv(2)/sqrt(xv(1)^2+xv(2)^2) - ...
     xv(2)/ sqrt((xv(1)-10)^2+xv(2)^2);
```

27
M-Files zum Kapitel 7

27.1
M-Files: Analytische Lösungen von Differentialgleichungen

Die einfache Dgl zum freien Fall im symbolischen Modus

```
dsolve('D2h = -g')
% ans =
% C3 + C2*t - (g*t^2)/2
```

Etwas allgemeiner: freier Fall und Wurf nach oben oder unten

```
%  fallwurf.m  Dgl. freier Fall/vertikaler Wurf
syms h(t) t g v0
h(t) = dsolve('D2h = -g', 'h(0)=0', 'Dh(0)=v0')
%  h(t) =
%  t*v0-(g*t^2)/2
sol = subs(h,[g v0],[9.81 0]) % g,v0 einsetzen
%  sol(t) =
%    -(981*t^2)/200
%  nun kann die Lösung gezeichnet werden mit
ezplot(sol, 0, 5)
```

Der freie Fall mit Luftwiderstend für verschiedene Bremsverhältnisse (Fallschirm, zusammengekauerte Person, ausgestreckte arme und Beine)

```
% Fall mit Luftwiderstand  h(t) nach oben positiv
% Kraft auf Körper m*h'' = -m*g + k*(h')^2
% k=1/2*rho*A*cw ,Luftdichte, exponierte Flaeche, Form-Koeffizient
format compact   %  Definition numerische Konstanten:
gn = 9.81,  rho = 1.2, mn = 100  % (Wahl: mn = 100 kg) SI Einheiten
% Grenzgeschwindigkeit h''=0 bei m*g=k*vl^2 -> vl = sqrt(m*g/k)
% Fallschirm d =8 m, A = 50 m^2, cw = 1.3
kf = 1/2*rho*50*1.3,  vlf=sqrt(mn*gn/kf) % vlf =  5 m/s = 18 km/h
% gestreckte Arme und Beine A = 60% von 0.8*2 = 0.96, cw = 0.8
kg = 1/2*rho*0.96*0.8, vlg=sqrt(mn*gn/kg)% vlg =  46 m/s = 165 km/h
% zusammengekauert, Kugel d=0.5, A = 0.25*3.14/4 =0.16 cw = 0.5
kk = 1/2*rho*0.16*0.5, vlk=sqrt(mn*gn/kk)% vlk = 143 m/s = 515 km/h
```

```
% analytische Lösungen der Dgln fuer h(t) und v(t)
syms h(t) v(t) t g k m vfrei(t)
h(t) = dsolve('D2h = -g + k/m*(Dh)^2','h(0)=0','Dh(0)=0')
v(t) = dsolve('Dv = -g + k/m*v^2','v(0)=0') % (v(t) verstaendlicher)
% v(t)=-(g^(1/2)*m^(1/2)*tanh((g^(1/2)*k^(1/2)*t)/m^(1/2)))/k^(1/2)
vf(t) = abs(subs(v(t),[g,m,k],[gn,mn,kf]))
vg(t) = abs(subs(v(t),[g,m,k],[gn,mn,kg]))
vk(t) = abs(subs(v(t),[g,m,k],[gn,mn,kk]))
ezplot(vf,0,10) ; hold on; ezplot(vg,0,10) ; ezplot(vk,0,10) ;
vfrei(t) = 9.81*t; ezplot(vfrei,0,10); hold off
```

Vergleich der Aufruf-Konventionen von dsolve() im symbolischen Modus:

```
% symdglkonv.m  % Konventionen fuer dsolve()
clear all;  syms y(x)  S
fa = dsolve(diff(y) == y/x) % OK
% fb = dsolve('diff(y) == y/x')   Falsch!
%    diff() darf nicht in String stehen
% fc = dsolve(Dy == y/x)        Falsch!
%    Kurzform Dy muss in String stehen
fd = dsolve('Dy == y/x')   % Falsch
%  Dy default ist dy/dt, sucht also y(t,x)
fe = dsolve('Dy == y/x',x)  % OK
ff = dsolve('Dy == y/x','x') % auch OK
% fg = dsolve('Dy == y/x','x','y(1)==S')
%   Falsch: Variablenwahl 'x' nach Bedingung:
fh = dsolve('Dy == y/x','y(1)==S','x') %OK
```

Analytische Lösung der Kettenlinie unter Verwendung der Tatsache, dass nur eine Funktion F(y”,y’) vorliegt (ohne y).

```
% kettlin
syms y(x) x p u(x)
u(x) = dsolve(diff(u) == p*sqrt(1 + u^2))
y(x) = int(u)  %y' = u bedingt: y = int(u)
%  u(x) =  sinh(C3 + p*x)
%  y(x) =  cosh(C3 + p*x)/p  (+C4)
```

Bestimmung der zur Kettenlinie weitgehend analogen Parabel aus den Aufhängepunkten und dem Durchhang

```
%kettpar   Parabel und Kettenlinie
%  Randpunkte links (S/H),  rechts(R/K),  Durchang D
syms S H R K D x f u  hr(f)  t(x) q xsp
hr(f) = H*(1-f) + K*f - u*(1-f)*f    % Gerade minus Durchhang
D = u*(0.5^2)     % max Durchhang unter der Geraden > u=4*D
t(x) = (x-S)/(R-S)      % Teilfunktion f 0..1 als t(x) S..R
pa(x) = simplify(subs(hr,f,t))       % Parabel(x)
krs = diff(diff(pa))    % Parabelkrümmung im Scheitel
xs = solve(diff(pa)==0), ys = pa(xs) % Scheitelpunkt xs,ys
```

Einsetzen von konkreten Zahlenwerten in die analytisch bestimmte Parabel

```
%  numerische Daten fuer Parabel und Kettenlinie
Sn = -150; Hn = 50; Rn = 150; Kn = 70; Dn = 35; un = 4*Dn;
pan(x) = subs(pa,[S H R K u], [Sn Hn Rn Kn un])
xsn = double(solve(diff(pan)==0))
ysn = double(pan(xsn))
krn = double(2*un/(Rn-Sn)^2), Rads = 1/krn
% cosh-Krümmungsparameter
xp = linspace(Sn,Rn,61);     % cosh numerisch yp(xp)
yp = ysn - 1/krn + cosh(-krn*xsn+krn*xp)/krn;
ezplot(pan,Sn,Rn) ; axis equal ; hold on
plot(double(xsn),double(ysn),'ro') ; plot(xp,yp,'m')
plot([Sn Rn],[Hn Kn],'kd')  ; hold off
```

Iterative Lösung der nichtlinearen Gleichung zur Bestimmung der an die vorher bestimmte Parabel angepassten Kettenlinie

```
%  Iteration Kettenlinien-Parameter q, xsp;
%  Scheitelhoehe bleibt
yss = Hn*(Rn - xsn)/(Rn - Sn) - Kn*(Sn - xsn)/(Rn - Sn) - Dn
ych(x) = yss - 1/q + cosh(-q*xsp+q*x)/q; % Kettenlinie
y1(x) = subs(ych,x,Sn);
y2(x) = subs(ych,x,Rn);
Jm = jacobian([y1, y2],[q ; xsp]);
%  (Spalten-) Vektor der anzupassenden Parameter
format long ; pm = [krn; xsn]
for k = 1:4
  ye1 = double(subs(y1,[q xsp], [pm(1) pm(2)]))-Hn;
  ye2 = double(subs(y2,[q xsp], [pm(1) pm(2)]))-Kn;
  if norm([ye1;ye2]) < 1e-4 ; break ; end
  Jn = double(subs(Jm,[q xsp],[pm(1) pm(2)]));
  pm = pm - Jn\[ye1;ye2]
end
format short
```

Analytische Lösung der Raketengleichung

```
%% raketengl.m  Raketengleichung, vertikal nach oben
clear all ; syms v(t) g m(t) w C m0
% inifinitesimale Impuls-Portionen
%    dptot = -m*g, dpgas = w*dm, dprak = m*dv
% m(t)*diff(v(t),t)+ diff(m(t),t)*w == -m(t)*g
% diff(v(t),t) == -w/m(t)* diff(m(t),t) == -g
ls = int(diff(v,t),t)
rs = int((-w*diff(m,t)/m-g),t) +C
fgl = ls == rs
```

```
% Resultat: fgl(t) =  v(t) == C - w*log(m(t)) - g*t
% C bestimmen aus Anfangsbedingung, einsetzen
Cx = solve(subs(fgl,{'t','v(t)','m(t)'},{0,0,m0}),C)
fgli = subs(fgl,C,Cx)
% Schlussresultat: v(t) == w*log(m0) - g*t - w*log(m(t))
```

Anwendung der Raketengleichung auf eine Feuerwerks-Rakete

```
%% fwrak.m Raketengleichung angewandt
%      auf Feuerwerks-Rakete
syms t w g m0 v(t) m(t) p u
v(t) = -g*t + w*log(m0/m(t))
%  w so, dass Schubkraft beim Start
%  w*p*m0 > 10*g*m0 = 100 m0, also w*p = 100
p = 0.5  % Massen-Ausstoss pro Sekunde: m0*p
vn(t) = subs(v(t),[g w],[9.81 200])
vm(t) = subs(vn(t),m(t),m0*(1-p*t)) % ezplot(vm,0,2)
% Steighoehe s(u) ist das Integral v(t)*dt von 0 bis u
s(u) = int(vm(t),0,u)
ezplot(s,0,2); axis([0 1.5 0 150])
```

Analytische Lösung der Verhulst-Gleichung

```
% verhulst.m  logistische oder Verhulst-Gleichung
syms y(t) t a M
y(t) = dsolve('Dy == a*y*(1-y/M)','y(0)==1')
% Res: y(t) = M/(exp(M*(log(M - 1)/M - (a*t)/M)) + 1)
yb = subs(y,[a,M],[0.5,15])
% Zahlenbeispiel a = 0.5, M = 15 zeichnen
ezplot(yb,[0,16])
```

Analytische Lösung eines ausschwingenden gedämpften harmonischen Oszillators

```
%% Oszillator analytisch
clear all
syms y(t) t w b v A
%% Oszillator ungedämpft
yu(t) = dsolve('D2y + w^2*y == 0')
%  liefert:  yu(t) =
%            C2*exp(t*w*i) + C3*exp(-t*w*i)
%% mit Dämpfung 2*b
yd(t) = dsolve('D2y + 2*b*Dy + w^2*y == 0')
yds(t) = simplify(yd,'steps',10)
% liefert: yds(t) =
%  yds(t) =
%    C5*exp(-b*t)*exp(t*(b^2 - w^2)^(1/2)) + ...
%      C6*exp(-b*t)*exp(-t*(b^2 - w^2)^(1/2))
```

Vergleichende Grafik eines Oszillators mit verschiedenen Dämpfungen:

```
%% Qszillator mit Dämpfung 2*b
%  und mit Anfangsbedingung y(0)=10; Dy(0) = 0;
clear all; syms y t w b % Skript fuer die Vergleichsgrafik
ydb(t) = dsolve('D2y + 2*b*Dy + w^2*y == 0',...
          'y(0) == 10', 'Dy(0) == 0')
yosc(t) = subs(ydb(t),w,2)   % w  auf 2 festlegen
%% diverse Dämpfungen einsetzen und dann zeichnen
yo1(t) = subs(yosc,b,0.005); ezplot(yo1,[0,15]); hold on
yo2(t) = subs(yosc,b,0.05);  ezplot(yo2,[0,15])
yo3(t) = subs(yosc,b,0.25);  ezplot(yo3,[0,15])
yo4(t) = subs(yosc,b,1.99);  ezplot(yo4,[0,15])
title('Ausschwingen mit verschiedenen Dämpfungen')
axis([0, 15, -12, 12]);  hold off
```

Analytische Lösung eines gedämpften Oszillators mit Anregung. Am Schluss wird die Resonanz-Überhöhung berechnet.

```
%% oscsinan: Symbolic Skript zum gedaempften Oszillator mit Anregung
clear all ; clc  ;   %   y'' + 2*b*y' + w^2*y = A*sin(v*t)
syms y(t) t w b v A R S p(t)
%% mit Dämpfung 2*b einfachere Formeln, zuerst homogene Lösung yds(t):
y(t)=dsolve('D2y + 2*b*Dy + w^2*y == 0'); yh(t)=simplify(y,'steps',10)
%%  fuer partikulaere Loesung  p(t): Stoerungs-Analoger Ansatz
p(t) = R*cos(v*t) + S*sin(v*t) % Ansatz analog A*sin(v*t), einsetzen:
pes=diff(diff(p(t),t),t)+2*b*diff(p(t),t)+w^2*p(t) -A*sin(v*t) == 0;
sitrm=collect(pes,'sin(v*t)'); sico=subs(sitrm,'cos(v*t)','0')/sin(v*t)
cotrm=collect(pes,'cos(v*t)'); coco=subs(cotrm,'sin(v*t)','0')/cos(t*v)
%%  Gleichungssystem in R,S: zuerst nach R aufloesen, gleichsetzen
rsi=solve(sico,R); rco=solve(coco,R); Sg = solve(rsi == rco, 'S');
%  dann beide nach S aufloesen und gleichsetzen ergibt Rg, Sg zeigen
ssi=solve(sico,S); sco=solve(coco,S); Rg = solve(ssi == sco, 'R'), Sg
% Gesamt-Amplitude sqrt(Sg^2+Rg^2) / A ergibt Resonanz-Ueberhoehung
Cga = sqrt((Sg/A)^2+(Rg/A)^2), Resu = simplify(Cga,'steps',20)
% Resu = 1/(4*b^2*v^2 + v^4 - 2*v^2*w^2 + w^4)^(1/2)
```

27.2
M-Files zu Lösungen mit Laplace-Transformationen

Lösung des einfachen Beispiels des radioaktiven Zerfalls mit Hilfe der Laplace-Transformation

```
% decaylaplace.m  Laplace Loesung radioaktiver Zerfall
clear all ; syms n(t) t s k  N u ; format compact
% Zerfalls-Gleichung  dn(t)/dt = -k*n(t)
deq = diff(n(t),t)==-k*n(t)  % t-Differentialgleichung
deql = laplace(deq, t, s)    % L-transfomierte Dgl
eqla = subs(deql, laplace(n(t), t, s), u ) % alg Gl.
```

```
N = solve(eqla, u)   % N = alg. Gl. nach u aufgeloest
n(t) = ilaplace(N,s,t)  % n(t) = exp(-k*t)*n(0)
```

Laplace-Lösung des ausschwingenden gedämpften Oszillators:

```
% oscdeclaplace.m  Laplace Loesung gedämpfte Schwingung
clear all ; syms y(t) t s b w  Y u ; format compact
% gedämpfter Oszillator  D2y + 2*b*Dy + w^2*y == 0
deq = diff(diff(y,t),t) + 2*b*diff(y,t) + w^2*y == 0 % t-Diffgl
deql = laplace(deq, t, s)    % L-transfomierte Dgl
%  Startbedingungen y'(0) = 0, y(0) = 10 einsetzen
deqli = subs(deql,'D(y)(0)',0)
deqlj = subs(deqli,'y(0)',10)
eqla = subs(deqlj, laplace(y(t), t, s), u ) % alg Gl.
Y = solve(eqla, u)      % Y = alg. Gl. nach u aufgeloest
y(t) = ilaplace(Y,s,t)  % konkrete Loesung: w,b, einsetzen
yc(t) = subs(y(t),[w, b],[2, 0.2]) % w = 2, b = 0.2
yd(t) = simplify(yc)
%yd(t) = 10*exp(-t/5)*(cos((3*11^(1/2)*t)/5) +
%                      (11^(1/2)*sin((3*11^(1/2)*t)/5))/33)
ezplot(yd,0,12)
```

Laplace-Lösung eines Systems von gekoppelten Oszillatoren

```
% couplosclaplace.m  Laplace Loesung gekoppelte Oszillatoren
clear all ; syms ya(t) yb(t) t s b w q  Y u v ; format compact
% gekoppelte Oszillatoren
%  D2ya + 2*b*Dya + w^2*ya + q*(yb-ya) == 0
%  D2yb + 2*b*Dyb + w^2*yb - q*(yb-ya) == 0
%  zuerst t-Gleichungen, dann L-transformierte dqal und dqbl
deqa=diff(diff(ya,t),t) + 2*b*diff(ya,t) + w^2*ya + q*(yb-ya)== 0
deqb=diff(diff(yb,t),t) + 2*b*diff(yb,t) + w^2*yb - q*(yb-ya)== 0
dqal = laplace(deqa, t, s),  dqbl = laplace(deqb, t, s) % L-tr.
% Startbedingungen ya(0)=10, ya', yb, yb'(0) = 0 einsetzen
dqai = subs(dqal,'ya(0)',10) ;  dqaj = subs(dqai,'D(ya)(0)',0)
dqbi = subs(dqbl,'yb(0)',0)  ;  dqbj = subs(dqbi,'D(yb)(0)',0)
% [dqaj; dqbj] System von 2 Gleichungen, neu Funktionen u,v gesucht
equb = subs([dqaj;dqbj] , 'laplace(ya(t), t, s)', u ) % alg Gl. u
equv = subs(equb , 'laplace(yb(t), t, s)', v ) % alg Gl. u, v
% nun ist im L-Raum die algebraische Gleichung fuer u und v bereit:
Y = solve(equv, [u,v])   % Y = algebr Gl.syst. nach u,v aufgeloest
yas = ilaplace(Y.u,s,t)  % u ruecktransformieren;  dann w,b,q in
ybs = ilaplace(Y.v,s,t)  % v ruecktransformieren;  beiden einsetzen
yac(t) = subs(yas,[w, b, q],[2, 0.01, 0.2]) % w=2, b=0.01, q=0.2
yad(t) = simplify(yac)
ybc(t) = subs(ybs,[w, b, q],[2, 0.01, 0.2]) % w=2, b=0.01, q=0.2
ybd(t) = simplify(ybc)
clf ;  subplot(2,1,1);  ezplot(yad,0,120) ; hold on
subplot(2,1,2);  ezplot(ybd,0,120)   ; hold off
```

27.3
M-Files: Numerische Lösung von Anfangswertproblemen

Differentialgleichungs-Löser mit vorwählbarem Butcher-Tableau, das bei der Lösung mit fester Schrittweite zur Anwendung kommen soll:

```
function [ts,ys] = odebutab(funct,tin,yin,h,nstep,cab)
% function [ts,ys] = odebutab(funct,tin,yin,h,nstep,cab)
%    odebutab.m  Dgl-Loeser mit Butcher Tableau
%    funct: Funktions-Name, tin,yin: Anfangswerte
%    h, nstep: Schrittweite und Anzahl Schritte
%    cab: Butcher-Tableau [ckoef amatrix bkoef']
  neq = length(yin);  nis = size(cab,1);
  ts = tin+(0:nstep)'*h;  % feste Schrittweite h
  ys = zeros(nstep+1,neq); % ys-Array vorbereiten
% ckoef, amat, bkoef fuer Butcher-Tableau
  c = cab(:,1); a = cab(:,2:end-1); b = cab(:,end);
% Hauptschleife nstep Schritte der Weite h
  yac(1:neq,1) = yin(1:end); ys(1,:) = yac';
  for k=2:nstep+1
    tac = ts(k-1);          % Basis t-Wert
    kis = zeros(neq,nis);   % Ableitungs-Zwischenwerte
    for i = 1:nis           % interne Teilschritte
      yef = yac + h*kis * a(i,:)'; % kis komb. zu yef
 % Aufruf der Ableitungs-Funktion bei yac+c*h, yef
      kis(:,i) = feval(funct,tac+c(i)*h,yef);
    end  % end interne Teilschritte
    yac = yac + h*(kis*b); % kis mit b mitteln
    ys(k,:) = yac';   % y-Werte in Tabelle eintragen
  end  % end Hauptschleife
end
```

27.4
M-Files: Anfangswertprobleme mit MATLAB

Differentialgleichungslösung mit den MATLAB Prozeduren für das einfache Beispiel des radioaktiven Zerfalls.,
1.Teil: Definition der an den Dgl-Löser zu übergebenden Ableitungsfunktion raddecay: (Parameter mit global-Deklaration festgelegt)

```
function dndt = raddecay(~,nactual)
% function dndt = raddecay(t,nactual) liefert die Ableitung
%   fuer einen einfachen radioaktiven Zerfall
% der Parameter t ist in der Liste obligatorisch,
%   wird hier aber nicht benutzt, daher '~'
```

```
% benutzt:  global  RADDECAYCONST
global RADDECAYCONST;  % Zugriff zu Globalparameter
dndt = -RADDECAYCONST*nactual;
```

2. Teil Aufruf des Dgl-Lösers ode45 und Grafik

```
   global RADDECAYCONST;     % Global-Deklaration
    RADDECAYCONST = 1.e-02;  % Zuweisung Globalparameter
   [tsol, nactive] = ode45('raddecay',25, 1.E20);
   plot(tsol, nactive)
```

Radioaktiver Zerfall mit zerfallendem Tochterkern (Parameter durch eingebettete Funktion transferiert)

```
function ratochtzerf(n1,n2,lam1,lam2,tmax)
% function ratochtzerf(n1,n2,lam1,lam2,tmax) Rahmenfunktion
%  Radioaktiver Zerfall mit Tochterkern
%----------------------------------------
   function dndt = tochtzerf(t,nsactual)
   % function dndt = tochtzerf(t,nsactual)
   % embedded function in ratochtzerf.m: kennt lam1 und lam2
     dndt = zeros(2,1); % forcieren eines Spaltenvektors
     dndt(1) = -lam1*nsactual(1);
     dndt(2) = lam1*nsactual(1) - lam2*nsactual(2) ;
   end
%----------------------------------------
%  die Konstruktion @tochtzerf erzeugt ein Handle auf die
%   oben definierte 'embedded function'
   [tsol, nsactive] = ode45(@tochtzerf,[0,tmax], [n1, n2]);
   plot(tsol, nsactive)
end
```

Zwei verschiedene Fälle der Stabilität des Tochterkerns im Vergleich:

```
% Beispiel Tochterkern viel stabiler
%  Anzahl Tochterkerne ca. = zerfallene Mutterkerne
 ratochtzerf(1e5, 0, 1e-3, 1e-9 , 4000)
% Beispiel Tochterkern nur leicht stabiler
%  Aufbau auf etwa 1/2 N0 Mutterkern
 ratochtzerf(1e5, 0, 1e-3, 5e-4 , 4000)
```

Der schiefe Wurf in 3D ergibt ein Beipiel für ein System von Differentialgleichungen und für die Anwendung einer Eventfunktion

```
function [ts,ysv,te,ye,ie] = schiefwurf(v0,azi,elev)
% v0 Startgeschwindigkeit; azi, elev: Azimuth, Elevation in Grad
% ts, ysv Flugbahnen des schiefen Wurfes ohne Luftwiderstand
% te,ye   Positionen der Events vz=0(Scheitel), z=0(Ende), z=10
  y0 = [0 v0*cosd(elev)*cosd(azi) 0 v0*cosd(elev)*sind(azi) ...
```

```
        0 v0*sind(elev) ];  % Startwerte
  tfinal = 2.1*y0(6)/9.8;
  options = odeset('Events',@scheitelboden); % Eventfunktion def.
% ode-Loeser mit Event-Aktivierung aufrufen
  [ts,ysv,te,ye,ie] = ode45(@wurfabl,[0 tfinal],y0,options);
% ---------------------------------------------------------------
  function ablvec = wurfabl(t,rvc)
  % function ablvec = wurfabl(t,rvc)
  %  berechnet die ersten Ableitungen von
  %    x,vx, y,vy, z,vz bei Gravitation ohne Luftwiderstand
     ablvec = zeros(6,1);   % Spaltenvektor vordefinieren
     ablvec(1) = rvc(2);    % x' = vx  (vx'= 0 vordefiniert)
     ablvec(3) = rvc(4);    % y' = vy  (vy'= 0 vordefiniert)
     ablvec(5) = rvc(6);    % z' = vz
     ablvec(6) = -9.81;     % z'' = vz' = - g
  end
% ---------------------------------------------------------------
  function [val,enddef,richtung] = scheitelboden(tev,yev)
  % function [value,isterminal,direction] = scheitelboden(tev,yev)
  % Eventfunktion zum definieren der Punkte bei denen val = 0 wird
    val(1) = yev(6);      % Scheitelpunkt: vz = 0
    enddef(1) = 0;        % hier weiterfahren
    richtung(1) = -1;     % 0 von Pos zu Neg ueberqueren
    val(2) = yev(5);      % Bodenhoehe Null:  z = 0
    enddef(2) = 1;        % Integration abbrechen
    richtung(2) = -1;     % 0 von Pos zu Neg ueberqueren
    val(3) = yev(5)-10;   % Bodenhoehe 10:  z -10 = 0
    enddef(3) = 0;        % weiterfahren
    richtung(3) = 0;      % Richtung egal
  end
% ----------------------------------------------------------------
end
```

Analytisch lösbare lineare Dämpfung bei einem ausschwingenden Oszillator im Vergleich zur nur numerisch lösbaren quadratischen Dämpfung.

```
function [t1,y1,t2,y2] = daempfvergl(a0,tmax, w, blin, bquad)
% function daempfvergl(a0,tmax, w, blin, bquad)
%  vergleich lineare und quadratische Daempfung
%  Gutes Beipiel: daempfvergl(20,40,1,0.1,0.1);
% ----- (eingebettete) Ableitungsfunktion lineare Daempfung --
    function ylabl = lindmp(~,yld) % ~ fuer nicht gebrauchtes t
      ylabl = zeros(2,1);
      ylabl(1) = yld(2);
      ylabl(2) = -w^2*yld(1)-2*blin*yld(2);
    end
% ----- Ableitungsfunktion quadratische Daempfung  ---------
    function yqabl = quaddmp(~,yqd)
      yqabl = zeros(2,1);
      yqabl(1) = yqd(2);
      yqabl(2) = -w^2*yqd(1)-2*bquad*yqd(2)*abs(yqd(2));
    end
```

```
% --- Dgl. Loesungen mit anschliessender Grafik --------
  [t1,y1] = ode45(@lindmp, [0 tmax], [a0 0]);
  plot(t1,y1(:,1));  hold on
  [t2,y2] = ode45(@quaddmp, [0 tmax], [a0 0]);
  plot(t2,y2(:,1),'k--'); hold off
end
```

Demonstration des Steifheit-Effektes: allgemeine Dgl-Löser benötigen meist viel zu hohen Rechenaufawnd:

```
function flammstart(eps)
%http://www.mathworks.com/company/newsletters/
%         articles/stiff-differential-equations.html
% flammstart   Loesungsverhalten steife Differentialgleichung
%  ----- Ableitungs-Funktion, in Rahmenfunktion eingebettet --
   function yabl = flammdgl(t,y) % Differentialgleichung
       yabl = y^2-y^3;           % beschreibt den Radius
   end                           % einer Flamme beim Start
%  -----
   odeopt4 = odeset('reltol',1e-4); % odeset: Toleranz-Vorwahl
% ode-Loeser zeichnet direkt bei Aufruf ohne Rueckgabeparameter
   subplot(2,1,1)
% 1. Versuch mit ode45
   ode45(@flammdgl,[0 2/eps],eps,odeopt4)
% 2. Versuch mit ode23s
   subplot(2,1,2)
   ode23s(@flammdgl,[0 2/eps],eps,odeopt4)
end
```

Die Lotka-Volterra Differentialgleichungen als Modell eines Jäger-Beute-Problems:

```
function lotkavolt(cre,cca,cej,cdj,inib,inij,tmax)
% function lotkavolt(cej,cdj,cre,cca,inij,inib,tmax)
%  Simulation des Lotka-Volterra Modells
% cre, cca: Reproduktion / Bejagungs-Koeff. Beute
% cej, cdj: Erfolgs- / Verminderungs-Koeff. Jaeger
% Parameter-Beispiel: lotkavolt (1,0.01,0.02,1,20,20,15)
% langlebige Raeuber:  lotkavolt(1,0.01,0.02,0.2,100,20,80)
yini = [inib, inij];
fxp = [ cdj/cej  cre/cca ]  % Fixpunkt der Dgln.
[ts,ys] = ode23s(@lotvdiff,[0 tmax], yini);
plot(ts,ys)
figure(2); plot(ys(:,1),ys(:,2))
% ------ eingebettete Ableitungsfunktion ------------------
    function  yd = lotvdiff(t,yac)
        yd(1,1) = cre*yac(1) - cca*yac(1)*yac(2);
        yd(2,1) = cej*yac(1)*yac(2) - cdj*yac(2);
    end
end
```

Geladene Teilchen im Magnetfeld: benutzen die Ableitungsfunktion

```
function deriv = chargpart3(t,yac)
B3=-1/20;  B2=-1/170;  B1 = 0;
deriv = zeros(6,1);
deriv(1) = yac(2);
deriv(2) = yac(4)*B3 - yac(6)*B2  ;
deriv(3) = yac(4);
deriv(4) = yac(6)*B1 - yac(2)*B3  ;
deriv(5) = yac(6);
deriv(6) = yac(2)*B2 - yac(4)*B1 ;
```

Die zugehörigen Anwendungskommandi lauten:

```
ystart=[ 0 0 0 1 0 0]';
[t45,y45]= ode45('chargpart3',600,ystart);
plot3(y45(:,1),y45(:,3),y45(:,5),'r'); axis equal
```

Erweiterung um ein zusäztliches elektrisches Feld senkrecht zum Magnetfeld ergibt eine Simulation des E×B Drifts: dazu die Ableitungsfunktion

```
function deriv = einbfeld(t,yac)
B=-1/20;
deriv = zeros(4,1);
deriv(1) = yac(2);
deriv(2) = yac(4)*B ;
deriv(3) = yac(4);
deriv(4) = -yac(2)*B + 0.015 ;
```

Die zugehörigen Anwendungskommandi lauten:

```
yin = [0 0 0 1]';
[tsol, ysol] = ode45('einbfeld', 300 , yin);
plot(ysol(:,1),ysol(:,3))
axis equal
```

27.5 M-File Beispiele zu chaotischem Verhalten

Die Ableitungs-Funktion des Lorenz-Strange-Attractors

```
function xderi = loratt(tac,xac)
xderi = zeros(3,1);
si = 10;
b=7/3;
r = 28;
% x' = -si*x + si*y
```

```
% y' = -  y + r*x - x*z
% z' = -b*z + x*y
xderi(1) = -si*xac(1) + si*xac(2);
xderi(2) = -xac(2) + r*xac(1) -xac(1)*xac(3);
xderi(3) = -b*xac(3) + xac(1)*xac(2) ;
```

Verschiedene Parameter-Werte und Grafiken für den Lorenz-Strange-Attractor:

```
xini = [5 5.02 0.000001]';
[tsol,xsol ]= ode45('loratt',50,xini);
plot3(xsol(:,1),xsol(:,2),xsol(:,3))
figure(2)
plot(tsol,xsol(:,1),'g')
hold on
xini = [5 5 0.000002]';
[tsol,xsol ]= ode45('loratt',50,xini);
plot(tsol,xsol(:,1),'r')
plot(tsol,xsol(:,3),'k')
hold off
```

28
M-Files zum Kapitel 8

28.1
M-Files zum Überblick über grosse Datenmengen

Streuparameter für verschiedene normalverteilte simulierte Beobachtungsreihen:

```
% normtest.m  Test der Parameter von verschiedenen
%   normalverteilten Reihen aus dem Zufallsgenerator
  npt = [50 100 100 100 1000 10000 100000];
  for k=1:7  % verschiedene normalverteilte Reihen
    rndn = randn(npt(k),1); % (~,1) wichtig, sonst Matrix!
    disp([log10(npt(k)),mean(rndn),std(rndn),...
                skewness(rndn),kurtosis(rndn)])
  end
% Resultate:
%    1.6990   -0.0190    1.0300    0.9888    4.0920
%    2.0000   -0.0591    1.0923    0.2175    2.7237
%    2.0000    0.0292    0.8286   -0.0343    2.6936
%    2.0000   -0.0621    1.0577    0.5182    3.2886
%    3.0000   -0.0189    1.0028    0.0261    2.9118
%    4.0000    0.0015    0.9959   -0.0311    3.0455
%    5.0000    0.0053    1.0003   -0.0001    3.0105
```

Einteilung einer Beobachtungreihe in Klassen gleicher Breite (binning):

```
function  [klasscent,klasshf] = klasshfg(dvec,midval,klbr)
% function  [klasscent,klasshf] = klasshfg(dvec,midval,klbr)
%  Haeufigkeitsbestimmung in Klassen gleicher Breite
%  dvec = Beobachtungsvektor,
%  midval = ein Klassen-Zentralwert; klbr = Klassenbreite
%  klasscent = Vektor der Klassen-Zentralwerte
%  klasshf   = Haeufigkeit der einzelnen Klassen
  dvs = sort(dvec); ndat = length(dvs);
  nklu = ceil((midval-klbr/2 -dvs(1))/klbr);
  nklo = ceil((dvs(end) - midval-klbr/2)/klbr);
```

```
klasscent = (-nklu:nklo)*klbr + midval;
klasshf = zeros(1,length(klasscent));
ind = 1; k = 1;
while k <= ndat
    if dvs(k) < klasscent(ind)+klbr/2
        klasshf(ind) = klasshf(ind)+1;
        k = k+1;
    else
        ind = ind+1;
    end
end
```

28.2 M-Files zur Regressions-Analyse

Geradenfit nach dem Prinzip des totalen least squares Verfahren, bei dem Fehler in x und in y gleichermassen berücksichtigt werden:

```
function pdop = doppelgfitsvd(X,Y,fx,fy)
%  function pdop = doppelgfitsvd(X,Y,fx,fy)
%  Geradenfit gemischt aus rx und ry-Residuen
%  X,Y Vektorpaar, fx,fy i.A. std(X), std(Y)
%  Normierung und Zentrierung
xn = (X - mean(X))./fx ; yn = (Y - mean(Y))./fy ;
pdp = svdtls(xn,yn)  % total least squares
pdop(2) = pdp(2)*fy/fx; % Ruecktransformation
pdop(1) = (pdp(1)- mean(X)*pdp(2)/fx)*fy +mean(Y);
```

Die eigentliche Fit-Prozedur für einen Geradenfit mit dem total least squares Verfahren:

```
function gpar = svdtls(x,y)
% Geradenfit mit total least squares
dim = 2; A=[ones(length(x),1) x y];
%   function [c,n] = clsq(A,dim); ref: W. Gander
%   solves the constrained least squares Problem
%   A (c n)' ~ 0 subject to norm(n,2)=1
R = triu (qr (A));  m=3;  p = 3;
[U,S,V] = svd(R(p-dim+1:m,p-dim+1:p));  n = V(:,dim);
c = -R(1:p-dim,1:p-dim)\R(1:p-dim,p-dim+1:p)*n;
gpar(1) = -c/n(2) ;  gpar(2) = -n(1)/n(2);
```

28.3
M-Files zur Wahrscheinlichkeitsrechnung

Bestimmung der Häufigkeitsfunktion für die Augensumme von drei Würfeln:

```
% dreiwuerfel.m   6x6x6 Kubus fuellen
  for i=1:6
    for j=1:6
        for k=1:6;  s(i,j,k) = i+j+k; end
    end
  end
  sl = sort(reshape(s,1,216)) ; % in Vektor, sortiert
  valvec = 3:18;    % vorkommende Augensummen-Werte
  for val = valvec  % Haeufigkeiten bestimmen für val-Werte
    hfg(val-2) = sum(sl == val);  % 1=true Werte zaehlen
  end
  bar(valvec,hfg)   % Balkendiagramm der Augensummen
```

Simulation der Augenzahl eines fairen Würfels als Abwandlung der gleichverteilten Wahrscheinlichkeitswerte zwischen Null und Eins.

```
  function wz = wuerfel(nz,ns)
  %  function wz = wuerfel(nz,ns)  simuliert einen Wuerfel
  %    liefert gleichverteilte ganze Zahlen zwischen 1 und 6
  %    in einer wz(nz,ns)-Matrix; benutzt rand(nz,ns)
    wz = ceil(6*rand(nz,ns));
  end
```

Simulation der Augensumme von drei Würfeln:

```
function ncount = dreiwsim(nvers)
% dreiwsim.m    Simulation der Augensumme von 3 Wuerfeln
% nvers:        Zahl oder Vektor mit den Versuchszahlen
% ncount(nfall,1:16): Haeufigkeiten der Augensummen 3 bis 18
  nfall = length(nvers); ncount = zeros(nfall,16);
  for fall = 1:nfall
    augs = wuerfel(3,nvers(fall)); sm = sum(augs);
    for z = 1:nvers(fall)
      ncount(fall,sm(z)-2)=ncount(fall,sm(z)-2)+1;
    end
  end
end %function
```

Hilfsfunktion skyline zum Bestimmen der oberen Randkurve eines Histogramms. Diese erlaubt mehrere Histogramme gleicher Art übereinander zu zeichnen.

```
function [xsky, ysky ] = skyline( ind,val )
%  function [xsky, ysky ] = skyline( ind,val )
```

```
%  berechnet die obere Randlinie eines Balkendiagramms
  vdop = repmat(val,2,1); indop = [ind-0.5;ind+0.5];
  ysky = [0 reshape(vdop,1,length(val)*2),0];
  xsky = [indop(1,1), ...
      reshape(indop,1,length(val)*2),indop(2,end)];
end
```

Auflisten und zählen der Kombinationen ohne Wiederholung

```
function [ nkomb, kombar ] = kombiliste(n,k)
% zählt und zeigt Kombinationen o.W. K(n,k)
% function [ nkomb, kombar ] = kombiliste(n,k)
% nkomb =  Anzahl Kombinationen K(n,k) ohne Wiederholungen
% kombar = Matrix mit einem Zeilenvektor pro Kombination
  nkomb = nchoosek(n,k);
  kombar = zeros(nkomb,k);   % Array vormerken
  mx = n ;
  for kb = k:-1:1  %  Maximalwerte der k Stellen
      mxv(kb) = mx;   mx = mx - 1;
  end;
  % Start-Kombination 1:k ist erster Zeilenvektor
  v = 1:k ;   kombar(1,:) = v;  ntot = 1;
  % Prinzip: Aufwaerts zaehlen mit Uebertrag
  cpos = k;
  while 1  % Abbruch via break
    if v(k) < mxv(k)   % hinterste Stelle erhoehen
      v(k) = v(k)+1; ntot = ntot +1;
      kombar(ntot,:) = v;     % nächste Komb. abspeichern
    else
      cpos = cpos-1;
      if cpos < 1 break; end; % Abbruch: es ex. kein v(0)
      if v(cpos) < mxv(cpos)  %Stelle bei cpos erhoehen
        v(cpos) = v(cpos)+1;  % nachfolgende aufsteigend
        for cp = cpos+1:k ;  v(cp) = v(cp-1)+1;  end;
        cpos = k ; ntot = ntot + 1;
        kombar(ntot,:) = v;   % neue Komb. abspeichern
      end  % end if v(cpos) ..
    end  % end if v(k) .. else
  end  % while 1
end    % function kombiliste
```

28.4
M-Files zu Stichproben und Tests

Simulation einer Stichproben-Erhebung zu einer Volksabstimmung:

```
% turnhalle.m Stichproben-Simulation
grst = [80,160]; prcsim = 60;
% Stichprobengroessen, Ja-Anteil
hpt= 20, brpt = 40 % Aufteilung der 800 Stimmberechtigten
%  Diese Parameter sind beliebig variierbar fuer weitere
%   Simulationsbeispiele
ntot = hpt*brpt, nja = floor(prcsim*brpt/100),
nnei = ntot-nja
for gr = 1:2
  siz = grst(gr);
  for rep = 1:30
    % M = [ ones(20,nja) zeros(20,nnei)];
    %    ergibt Variante der Darstellung mit allen ja links
    H = rand(1,ntot); limh = prctile(H,prcsim);
    M = (reshape(H,npt,brpt) < limh); ytot = sum(sum(M));
    srd = rand(1,ntot); limval = prctile(srd,100*siz/ntot);
    sel = (reshape(srd,hpt,brpt) < limval);
    cntchk = sum(sum(sel));
    Mst = M.*sel;  Mstn = (1-M).*sel;
    cnty = sum(sum(Mst)); stres(rep,gr) = 100*cnty/cntchk;
    disp([gr,rep ,cntchk,cnty, 100*cnty/cntchk]);
    clf; spy(Mstn,'r'); hold on; spy(Mst,'g'); pause(0.6)
  end
end
```

Verschiedene Signifikanz-Tests mit simulierten Stichproben und den MATLAB Bibliotheksfunktionen `ttest` und `ttest2`:

```
% statitests  Anwendungsbeipiel Koerpergroessen
mudu = 178 ; sddu = 7.3;  % Parameter Grundgesamtheit
rng(1234567); xd =  mudu + sddu*randn(1,500);
muap = 174 ; sdap = 7;  % Parameter Vergleichsgruppe
% Mit Zufallsgenerator ezeugte Experimentierdaten
rng(2345671); xap1 = muap + sdap*randn(1,15);
rng(3456712); xap2 = muap + sdap*randn(1,30);
rng(4567123); xap3 = muap + sdap*randn(1,100);
[h1,p1,ci1,stat3] = ttest(xap1,mudu)
%  h1 =  0  , p1 =  0.1294
[h2,p2,ci2,stat3] = ttest(xap2,mudu)
%  h2 =  0  , p2 =  0.0765
[h3,p3,ci3,stat3] = ttest(xap3,mudu)
%  h3 =  1  , p3 =  1.8974e-06
[hd1,pd1,cid1,statd3] = ttest2(xap1,xd)
%  hd1 = 0  , pd1 = 0.1268
[hd2,pd2,cid2,statd3] = ttest2(xap2,xd)
%  hd2 = 1  , pd2 = 0.0358
[hd3,pd3,cid3,statd3] = ttest2(xap3,xd)
%  hd2 = 1  , pd2 = 0.0358
```