Übungsblatt 1

Arbeiten mit Vektoren

Addition, Subtraktion, Multiplikation eines Vektors mit einem Skalar, Betrag

```
a = [ 2; -3; 4 ];      b = [ 3; 5; -1 ];      c = [ 3; 0; 7]
a(3), b(2), c(1)                              % einzelne Koordinaten
d = 2*a  – 3*b + c                            % Linearkombination
a == b                                        % Abfrage auf Gleichheit
ra = sqrt(a(1)^2 + a(2)^2 + a(3)^2)           % Betrag von a
rb = norm(b)                                  % Betrag von b
rc = norm(c)                                  % Betrag von c
norm(d) <= 2*ra + 3*rb + rc                   % Dreiecksungleichung
ea = a / ra                                   % Normierungs-(Richtungs-)Vektor
eb = b / rb
ec = c / rc
```

Richtungkosinus, Richtungwinkel im Gradmaß

```
cal = a(1) / ra, cbe = a(2) / ra, cga = a(3) / ra      % Richtungskosinus a
                                                       % oder einfacher
cal = ea(1), cbe = ea(2), cga = ea(3)
koef = 180 / pi                                        % Bogen in Grad
alfa = koef* acos(cal), beta = koef*acos(cbe), gamma = koef*acos(cga)
```

Skalarprodukt, Vektorprodukt

```
skab = a(1)*b(1) + a(2) * b(2) + a(3) * b(3)      %  Skalarprodukt a·b
skac = dot(a,b)
skbc = dot(b,c)
phi = koef*acos(skab/(ra*rb))                     % Winkel zwischen a und b

n = [ 0 0 0 ]'                                    % Hilfsvektor erzeugen
n(1) = a(2)*b(3) – a(3)*b(2);                     % Vektorprodukt a x b
n(2) = a(3)*b(1) – a(1)*b(3);
n(3) = a(1)*b(2) – a(2)*b(1);
crab = n
crab = cross(a, b)
```

Quelle:
Schott, D.: Ingenieurmathematik mit MATLAB, Fachbuchverlag Leipzig 2004.

Übungsblatt 2

Arbeiten mit Matrizen und Feldern

Addition, Subtraktion, Multiplikation einer Matrix mit einem Skalar, Det., Rang

```
A = [ 2 3 –1; 2 1 –1; 4 2 -1]
B = [ 7 3 –2; -5 1 8; 4 –5 3]

s1 = A( : , 1)                          % 1. Spalte von A
z3 = B( 3, : )                          % 3. Zeile von B

[m, n] = size(A)                        % Anzahl der Zeilen und Spalten
A1 = [ A; [ 2 7 1]]                     % 4. Zeile ergänzen
[m, n] = size(A1)                       % neue Abfrage
A1(4, : ) = [ ]                         % 4. Zeile wieder streichen
E = eye(size(A))                        % Einheitsmatrix wie A

A23 = A(2:3, 1:2)                       % 2x2 - Untermatrix
B12 = B(1:2, 2:3)

C = 3*A – 4*B + E                       % Linearkombination
A == B                                  % Abfrage gleicher Elemente

a = det(A),  b = det(B),        c = det(C)   % Determinantenberechnung

ra = rank(A), rb = rank(B), rc = rank(C)     % Rangbestimmung
```

Die Konditionszahl κ einer Matrix A ist definiert als

$$\kappa = \| A \| \, \| A^{-1} \|, \quad \| A \| = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2} \, .$$

```
konda = cond(A)                         % Konditionszahl von A
kondb = cond(B)                         % Es gilt immer:   kond ≥ 1
kondc = cond(C)                         % nahe 1 – gut konditioniert
                                        % sehr groß – schlecht kondi-
                                        % tioniert (num. singulär)
```

Matrixmultiplikation, Matrixdivision

Zwei Matrizen $A = A^{m,n}$ und $B^{n,p}$ können nur miteinander multipliziert werden, wenn die innere Dimension (hier: n) gleich ist.

```
Prod1 = A*B

A1 = [ A; [2 7 1]]
B1 = [ B'; [ 1 –3 5]]'
Prod2 = A1*B1
```

Ist eine quadratische Matrix A regulär, d.h. ist det(A) nicht gleich Null, so existiert die Inverse X: $A*X = E$ und $X*A = E$. Wir bezeichnen Sie mit $X = A^{-1}$.

Gegeben sei das lineare Gleichungssystem $A*x = b$ mit dem Vektor der rechten Seite b. Um das Gleichungssystem zu lösen, können wir die Gleichung auf beiden Seiten mit A^{-1} multiplizieren. Wir erhalten $A^{-1} * A * x = A^{-1} *b$.

Da aber $A^{-1} * A = E$ und $E *x = x$ ist, erhält man den Lösungsvektor $x = A^{-1} *b$.

Diese Operation wird in MATLAB formal mit einer „Matrixdivision“ realisiert:

```
b = [ 5 0 5]'
x = A \ b          % Achtung: A^-1 wird nicht wirklich berechnet, da das zu
                   % aufwendig wäre (siehe: lineare Gleichungssysteme).
```

Arrayoperationen (Feldoperationen)

```
U1 = A .*B         % elementweise multiplizieren
U2 = A .^2         % elementweise quadrieren
U3 = A ./ B        % elementweise dividieren
U4 = B .\ A        % ist dasselbe
MA = max(A, B)     % elementweises Maximum
MI = min(A, B)     % elementweises Minimum
```

Weitere spezielle Matrizen

```
E1 = eye(m,n)      % rechteckige Einheitsmatrix
E2 = eye(size(A))
O1 = zeros(m,n)    % rechteckige Nullmatrix
O2 = zeros(size(A))
I1 = ones(m,n)     % rechteckige Einsen-Matrix
I2 = ones(size(A)
dA = diag(A)       % Diagonale von A als Vektor
D = diag(dA)       % Diagonalmatrix mit dem Vektor dA
IA = inv(A)        % Inverse von A
```

Quelle:
Schott, D.: Ingenieurmathematik mit MATLAB, Fachbuchverlag Leipzig 2004.

Übungsblatt 3

Lösen linearer Gleichungssysteme

Lösen Sie das lineare Gleichungssystem A x = b:

$$A = \begin{pmatrix} 0.780 & 0.563 \\ 0.913 & 0.659 \end{pmatrix} \qquad b = \begin{pmatrix} 0.217 \\ 0.254 \end{pmatrix}$$

Auf Rechnern mit geringer Genauigkeit können die Eingabewerte verfälscht werden. Es sei der Fall angenommen, dass dabei aus der Matrix A die Matrix A + ΔA und aus dem Vektor der rechten Seite b der Vektor b + Δb entsteht.

Lösen Sie damit wieder das Gleichungssystem. Wie erklären Sie sich die große relative Abweichung der Lösung von der Lösung des ungestörten Systems?

$$A + \Delta A = \begin{pmatrix} 0.780002 & 0.563003 \\ 0.913001 & 0.658997 \end{pmatrix} \qquad b + \Delta b = \begin{pmatrix} 0.217004 \\ 0.253995 \end{pmatrix}$$

Berechnen Sie die Konditionszahl kappa der Matrix A:

```
kappa = cond(A)
rkappa = rcond(A)                    % reziproke Konditionszahl
```

Wie groß ist die Zahl 1.0 + rkappa ?

```
asing = 1.0 + rkappa
```

Schauen Sie sich asing im **format long** an! In diesem Fall ist die Matrix A nahezu numerisch singulär.

Die Matrix A sei regulär und ΔA sei so klein, dass

$$\kappa_\Delta = \|\Delta A\| \cdot \|A^{-1}\| < 1.$$

Der relative Fehler der Lösung x des gestörten Systems kann dann wie folgt abgeschätzt werden:

$$\frac{\|\Delta x\|}{\|x\|} \le \frac{\|A\| \cdot \|A^{-1}\|}{1 - \kappa_\Delta} \left(\frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta b\|}{\|b\|} \right).$$

Ist κ_Δ sehr nahe Null, dann gilt die vereinfachte Ungleichung

$$\frac{\|\Delta x\|}{\|x\|} \le \mathrm{cond}(A)\left(\frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta b\|}{\|b\|}\right).$$

Lösen Sie die folgenden linearen Gleichungssysteme mit OCTAVE und geben Sie den maximalen absoluten Fehler der Lösung an, wenn der relative Fehler der Matrix A und des Vektors b jeweils maximal $5*10^{-6}$ beträgt:

a)

$$A = \begin{pmatrix} 2 & 3 & -1 \\ 2 & 1 & -1 \\ -4 & 2 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 0 \\ 2 \\ -9 \end{pmatrix}$$

b)

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

c)

$$\begin{aligned} 3x_2 - 5x_3 + x_4 &= 0 \\ -x_1 - 3x_2 - x_4 &= -5 \\ -2x_1 + x_2 + 2x_3 + 2x_4 &= 2 \\ -3x_1 + 4x_2 + 2x_3 + 2x_4 &= 8 \end{aligned}$$

d)

$$\begin{aligned} 5x_1 - 3x_2 + 2x_4 &= 13 \\ 2x_1 + 6x_2 - 3x_3 &= 16 \\ -x_1 + 2x_2 + 4x_3 - x_4 &= -11 \\ -2x_1 - 3x_2 + 2x_3 + 7x_4 &= 10 \end{aligned}$$

Erklären Sie den maximal möglichen Fehler in jeder Lösungskomponente näherungsweise durch die Angabe des möglichen Intervalls der einzelnen Lösungskomponenten.

Literatur: Schwetlick, H., H. Kretschmar: Numerische Verfahren für Naturwissenschaftler und Ingenieure, Fachbuchverlag Leipzig, 1991.

Übungsblatt 4

Grafische Darstellungen

Grafische Darstellung von symbolisch gegebenen Funktionen

```
ezplot( ` exp(-x) `), [ -1, 5 ] )        % e^x im Intervall [ -1, 5 ] zeichnen
syms x, ezplot(exp(-x), [ -1, 5] )       % dasselbe
```

Grafische Darstellung von numerisch gegebenen Funktionen

```
x = -2 : 0.01 : 2.0;                     % Erzeugen des Intervalls [-2; 2]
y1 = sinh(x) ;                           % y = sinh x
y2 = asinh(x);                           % Umkehrfunktion von sinh x
plot(x, y1, `r`, x, y2, `b`)             % Zeichnen beider Grafiken
grid                                     % Gitter und Beschriftung
title( `  Hyperbelfunktion und Areafunktion  `)
xlabel( `x`)
ylabel( `y`)
legend( `sinh x `, `arsinh x `)          % Legende in Reihenfolge y1, y2, ...
hold on
y3 = x;   plot( x, y3 )                  % Zeichnen der Symmetrieachse
hold off

axis square                              % Ausgabe im Quadrat
axis equal                               % und im gleichen Maßstab
```

Mehrere Fenster

```
figure(1), plot(x, y1)                   % 1. Fenster
figure(2), plot(x, y2)                   % 2. Fenster
figure(3), plot(x, y3)                   % 3. Fenster
```

halblogarithmische Darstellung

```
x = 0 : 0.01 : 2.0;                      % Erzeugen des Intervalls [0; 2]
y = exp(x) ;                             % y = e^x
semilogy( x, y )                         %  y - Achse logarithmisch
```

Quellen:

Diercksen, C.: MATLAB / Studentedition, Beuth-Hochschule für Technik
Schott, D.: Ingenieurmathematik mit MATLAB, Fachbuchverlag Leipzig 2004.

Übungsblatt 5

Lösen nichtlinearer Gleichungen

Iterative Bestimmung der Wurzel aus einer nichtnegativen reellen Zahl a

$x_0 = a > 0$, $x_{k+1} = (x_k + a / x_k)/2$, $k = 1, 2, \ldots$

Funktions-m-File: wurzel.m

```
a = input(‘  a =  ‘);
if a < 0;  error(‘ Keine reelle Lösung  ‘); end
tol = 1e-6;                          %  Abbruchschranke
x = a;
while abs(x^2 – a) > tol;
   x = (x + a/x) / 2;
end
x                                    %  Ausgabe
```

Iterative Nullstellenberechnung mit OCTAVE

Gelöst werden soll die transzendente Gleichung $f(x) = 0$,

z. B. $f(x) = \cos x – e^x = 0$ oder $f(x) = e^x + 2e^{-x} – 9 = 0$.

Zur Lösung nichtlinearer Gleichungen (Gleichungssysteme) stehen in OCTAVE zwei Gruppen iterativer Verfahren zur Verfügung:

1. Bisektionsalgorithmus: **[x, fval, exflag, output] = fzero(@f,1)**
2. Optimierungsalgorithmus: **[x, fval, exflag, output] = fsolve(@f,1)**

Dabei bedeuten: **x** Lösung (Lösungsvektor), **fval** Funktionswert (-vektor),
exflag Rückkehrcode (= 1 heißt O.K.), **output** Text/weitere Daten,
@ function handle, **1** Startwert (-vektor).

Dazu muß ein function-m-file mit dem Namen f erstellt werden, z.B.:

```
function y = f(x)
      y = cos(x) – exp(x) ;
end
```

Zur numerischen Lösung wird ein Iterationsverfahren verwendet. Dazu wird ein Startwert benötigt, der möglichst schon in der Nähe der Lösung liegt. Wer nur an der Lösung interessiert ist, kann auch nur schreiben

x = fzero(@f,1) % Startwert x = 1
bzw. **x = fsolve(@f,1)**

Aber Vorsicht! Manchmal wird keine (richtige) Lösung gefunden, z.B. für $\tan(x) - 0.5\,x = 0$ mit dem Bisektionsalgorithmus (Startwert 1)!

Aufgabe:

Lösen Sie die folgenden Gleichungen und nutzen Sie zusätzlich entsprechende grafische Darstellungen.

a) $\cos x = x$, b) $\sin x + 2\,x^{1/2} = 1$, c) $x\,e^{x} = 1$,
d) $1 + \cos x \cosh x = 0$.

Quelle: Schott, D.: Ingenieurmathematik mit MATLAB, Fachbuchverlag Leipzig 2004.

Übungsblatt 6

Dreidimensionale Felder und Strukturen

Dreidimensionale Felder

```
B(2,2,2) = 7                % Zwei 2x2-Matrizen in 2 Schichten
B(1,2,1) = 5                % Element (1,2) der 1. Matrix neu
C = B + 1
D = ones(4,4,4) +3
E(:,:,2) = eye(2,2) + [ 1 2; 3 4]
```

Experimentieren Sie damit und stellen Sie fest, welche Operationen zu Fehlermeldungen führen:

```
A = zeros(10,1) + 1;        B = eye(2) + [ 1 2 3; 4 5 6];
```

Strukturen

In einer Struktur können Variablen verschiedenen Typs zusammengefaßt werden. Wir definieren die Struktur "ADACPost", die folgende Objekte enthalten soll:

Typ		(Zeichenkette)
Anzahl der Sitzplätze		(Skalar)
Gepäckraum		(Skalar)
3D- Maße: Länge, Breite, Höhe		(3D – Vektor).

Das konkrete Feld dieser Struktur kann erreicht werden über den Namen

structurename.fieldname

```
ADACPost.type = "Überlandbus_SCANIA_OmniExpress3.6"
ADACPost.passengers = 71
ADACPost.luggage=14.7
ADACPost.measures = [ 15 2.55 3.6 ]
```

Alternativ kann auch eingegeben werden:

```
ADACPost = struct("type", "Überlandbus_SCANIA_OmniExpress3.6",
        "passengers", 71, "luggage", 14.7, "measures", [ 15 2.55 3.6]);
```

Dabei ist die Reihenfolge der Eingabe der Felder frei wählbar.
Möglicherweise gibt es mehrere (vielleicht 3) ADAC-Post – Fahrzeuge.

```
b(1) = ADACPost;
b(2) = ADACPost;
b(3) = ADACPost;
```

Was ist b(3) ? b(3).type

Zwei verschiedene Funktionen dienen in OCTAVE dem Zuweisen von Strukturfeldern und ihrem Wiederfinden:

```
b(3) = setfield( b(3), "type", "Pannenhilfefahrzeug_VW_Passat" )
getfield( b(3), "type" )
```

Geschachtelte Strukturen

```
ADACPost = struct("type1", b(1), "type2", b(3))
ADACPost.type1.type
```

ADACPost besitzt jetzt 2 Felder, type1 und type2. Jedes dieser Felder ist selbst eine Struktur, die durch b(1) und b(3) gegeben ist.

Quelle: Schmidt Hansen, J: GNU Octave Beginner's Guide, Packt Publ., 2011.

Übungsblatt 7

Cell Arrays

Cell-Arrays können numerische Variablen und Zeichenketten enthalten, aber nicht wieder Cell-Arrays (was bei Strukturen möglich ist). Jedes Element (jede Zelle) kann also enthalten

eine skalare Größe, einen Vektor, eine Matrix, eine Zeichenkette usw. Da Cell-Arrays Elemente verschiedenen Typs enthalten können, unterscheiden sie sich von gewöhnlichen Arrays (Feldern) und werden deshalb durch das Mengensymbol { } dargestellt.

Wenn wir beim Beispiel der ADACPost – Fahrzeuge bleiben, können wir eine Instanz eines Cell-Arrays wie folgt bilden:

```
ADACPostbus = {"Überlandbus_SCANIA_OmniExpress3.6",71, 14.7,
                    [ 15 2.55 3.6 ] }
```

Darin sind wieder Typ, Sitzplätze, Gepäckraum und der Vektor „Maße“ enthalten. Eine beliebige Zelle erreicht man so:

```
ADACPostbus{1}
```

Ein zweidimensionales Cell-Array kann wie folgt definiert werden:

```
ADACPostbusse = { "Überlandbus",71, 14.7, [ 15 2.55 3.6 ]; ...
                  "DHL", 2, 15, [ 6 2.1 2.5]; ...
                  "Pannenhilfefahrzeug", 2, 5.2, [ 4.5 1.9 1.7] }
```

Die einzelnen Elemente (Zellen) können über ihre Indizes erreicht werden:

```
ADACPostbusse{ 3, 1 }
```

Variablentyp identifizieren

```
isscalar( ADACPost )     bzw.  isvector( ADACPost )
ismatrix( ADACPost )     bzw.  isstring( ADACPost )
```

Die 1 steht für „wahr“, die 0 für „falsch“. Alles falsch, was ist es dann?

```
typeinfo( ADACPost )
typeinfo( ADACPostbus )
typeinfo( ADACPostbusse )
```

Alle Objekte, die mit B beginnen, können so gelöscht werden:

```
clear B*
```

Das geht natürlich auch mit A (bitte nur durchführen, wenn es wirklich gewollt ist!) : `clear A*`

Operatoren für Strukturen und Cell-Arrays

Die Struktur s enthalte eine Matrix und einen Vektor:

```
s = struct("A", [ 1 2; 3 4 ], "b", [ 5  11]')
x = s.A \ s.b                                   % Lösen des lin. GLS
```

Lösen eines linearen Gleichungssystems mit Cell-Arrays:

```
c = { [ 1 2; 3 4 ],  [ 8 18 ]' }
y = c{1}\c{2}
```

Operationen mit vollständigen Strukturen oder Cell-Arrays sind jedoch in OCTAVE nicht definiert.

Quelle: Schmidt Hansen, J: GNU Octave Beginner's Guide, Packt Publ., 2011.

Übungsblatt 8

Umgang mit Polynomen

Das Polynom $y = P(x) = x^3 - 15x - 4$ wird mit Hilfe des Zeilenvektors seiner Koeffizienten dargestellt:

```
p = [ 1 0 −15 −4 ]
```

Funktionswertberechnung

```
polyval( p, 2)          % Funktionwert bei x = 2
polyval( p, 1:3)        % dasselbe für x = 1, 2 und 3
x = −5 : 0.1 : 5;
y = polyval(p, x)
```

Nullstellen von Polynomen

```
r = roots(p)            % Spaltenvektor der Nullstellen
pr = poly( r )          % Polynomkoeffizienten aus den Nullstellen
```

Multiplikation von Polynomen

```
p1 = [1 3 2]; p2 = [ 1 0 1];
p3 = conv(p1, p2)       % Convolution -- Produkt beider Polynome
```

Polynomdivision

```
[ p4, r4 ] = deconv( p1, p2)    % Deconvolution -- p4 ganzrationale F.,
                                % r4 Divisionsrest
```

Partialbruchzerlegung einer echt gebrochen-rationalen Funktion

Zählerpolynom: $p_1(x) = 4x^2 + 4x + 1$
Nennerpolynom: $p_2(x) = x^3 + x^2$

```
p1 = [ 4 4 1];  p2 = [1 1 0 0];
[ r, xP, g ] = residue( p1, p2)   % r   Zähler der Partialbrüche
                                  % xP Polstellen der Partialbrüche
                                  % g ganzrationaler Teil
```

Das Ergebnis der Partialbruchzerlegung ist

$$\frac{4x^2+4x+1}{x^3+x^2} = \frac{r_1}{x-x_{P1}} + \frac{r_2}{x-x_{P2}} + \frac{r_3}{x-x_{P3}} = \frac{1}{x+1} + \frac{3}{x} + \frac{1}{x^2}.$$

1. Aufgabe:

Zählerpolynom: zp = 4 x^4 – 4,
Linearfaktoren des Nenners: np1 = x – 2, np2 = x + 2.
Berechnen Sie mit Hilfe einer Polynommultiplikation den Nenner np.
Bestimmen Sie die Nullstellen des Zählerpolynoms.
Zerlegen Sie die unecht gebrochen-rationale Funktion zp / np in eine ganzrationale Funktion qp und eine echt gebrochen-rationale Funktion rp.
Führen Sie für rp eine Partialbruchzerlegung durch.

2. Aufgabe: Führen Sie eine Partialbruchzerlegung folgender Funktionen durch:

a) $\frac{2}{x^2-1}$ b) $\frac{2x+3}{x^2+3x+2}$

c) $\frac{x^3+x^2+x+2}{x^4+3x^2+2}$ d) $\frac{2x^3-14x^2+14x+30}{x^2-4}$

<u>Darstellung komplexer Nullstellen</u>

Das Polynom K(x) hat komplexe Koeffizienten:

```
k = [ 1  3–j  j–5]
kn = roots(k)          % komplexe Nullstellen
compass(kn)            % grafische Darstellung in der komplexen Ebene
```

3. Aufgabe:
Lösen Sie die Gleichungen im Bereich der komplexen Zahlen und stellen Sie die Lösungen grafisch dar.

a) $z^7 - z = 0$, b) $z^5 + z^4 + z^3 + z^2 + z + 1 = 0$, c) $z^3 - j = 0$.

Quellen: Schott, D.: Ingenieurmathematik mit MATLAB, Fachbuchverlag Leipzig 2004.
Kutzner, R., S. Schoof: MATLAB/Simulink, RRZN Hannover, 2012.

Übungsblatt 9

Kombinatorik und diskrete Wahrscheinlichkeitsverteilungen

Für die Formeln der Kombinatorik wird der Begriff der Fakultät benötigt.

```
f5 = factorial(5)
```

Wahrscheinlichkeitsfunktionen von diskreten Zufallsgrößen können in einem Balkendiagramm, auch Stabdiagramm genannt, dargestellt werden.

```
x = [0:4];
f = [ 0.1 0.25 0.5 0.15 ];
bar(x,f)
```

Aufgabe 1:

In einer Lieferung von 20 Elektrogeräten sind 5 fehlerhafte Geräte enthalten. Berechnen Sie mit OCTAVE die Wahrscheinlichkeitsfunktion für die Ziehung von 0, 1, 2, 3, 4 und 5 fehlerhaften Geräten bei 5 zufälligen Entnahmen ohne Zurücklegen. Stellen Sie diese im Balkendiagramm grafisch dar.

Aufgabe 2:

Die Zufallsgrößen X_i, i = 1, 2, ... n, seien die Augenzahlen, die beim i-ten Wurf eines homogenen Würfels angezeigt werden. Berechnen Sie die Wahrscheinlichkeitsfunktion der Zufallsgröße $Y_n = X_1 + X_2 + ... + X_n$ und der normiereten ZG $Z_n = (Y_n - \mu_n) / \sigma_n$ und stellen Sie diese in einem Balkendiagramm grafisch dar für n = 1, 2, 3, 4, 5 und 6.

Aufgabe 3:

Bei der Herstellung von Gewindeschrauben entsteht ein Ausschußanteil von 2%. Berechnen Sie Erwartungswert und Standardabweichung der Zahl der in 250, 500, 1000 und 2000 Stück enthaltenen unbrauchbaren Schrauben.

Aufgabe 4:

Ein Sportschütze trifft mit einer Wahrscheinlichkeit p = 1/3 die Zielscheibe. Wie groß ist die Wahrscheinlichkeit, daß er bei 5 Versuchen die Scheibe k mal trifft, k = 0, 1, 2, 3, 4, 5. Erstellen Sie ein Balkendiagramm.

Quelle: Papula, L.: Mathematik für Ing. und Naturwis., Bd. 3., Vieweg.

Übungsblatt 10

Stetige Wahrscheinlichkeitsverteilungen

Wahrscheinlichkeitsdichten und Verteilungsfunktionen von stetigen Zufallsgrößen werden grafisch mit dem plot-Befehl dargestellt.

```
x = [0:0.01:10];
f = 0.2*exp(-0.2*x);
F = 1 – exp(-0.2*x);
subplot(1,2,1), plot(x,f), title('  Dichte der Exponentialverteilung  ')
subplot(1,2,2), plot(x,F), title('  Verteilungsfunktion der
                                    Exponentialverteilung  ')
```

Aufgabe 1:

Für die Ausführungszeit X in Stunden einer bestimmten Arbeit wird die folgende Dichtefunktion angenommen:

$$f(x) = C\,(9 - x)^2,\quad 0 \le x \le 9, \text{ und } f(x) = 0 \text{ sonst.}$$

a) Berechnen Sie den Wert, den die Konstante C annehmen muß, und
b) die Verteilungsfunktion der Zufallsgröße X.
c) Stellen Sie die Dichtefunktion und die Verteilungsfunktion grafisch dar.
d) Berechnen Sie Erwartungswert und Standardabweichung der Zeitdauer X.
e) Für die Ausführung der Arbeit wird ein Grundbetrag von 30 Euro und ein Stundenlohn von 80 Euro berechnet. Zusätzliche Materialkosten fallen nicht an. Wie groß sind Erwartungswert und Standardabweichung des Rechnungsbetrages?

Aufgabe 2:

Die Feuerwehr einer Stadt wird ca. aller 2 Tage zum Löschen eines Brandes eingesetzt. Die Zahl der Einsätze X pro Woche wird als ***Poisson***-verteilt angenommen.

a) Berechnen Sie die Wahrscheinlichkeitsfunktion der Anzahl der Einsätze pro Woche bis X = 8 und stellen Sie diese grafisch dar.
b) Mit welcher Wahrscheinlichkeit muß die Feuerwehr mehr als 2 mal pro Woche ausrücken?
c) Berechnen Sie die die Dichte und die Verteilungfunktion der Zeit T, die zwischen zwei Bränden vergeht. Stellen Sie beide Funktionen grafisch dar.
d) Mit welcher Wahrscheinlichkeit muß die Feuerwehr innerhalb von zwei Tagen nach dem letzten Einsatz wieder ausrücken?
e) Mit welcher Wahrscheinlichkeit findet mindestens 5 Tage nach dem letzten Einsatz kein neuer Einsatz statt?

Aufgabe 3:

Die Längenmessung von 10 Schrauben ergibt folgende Ergebnisse (in mm):

10 8 9 10 11 11 9 12 8 12.

Die Länge der Schrauben ist normalverteilt mit einer Varianz von $\sigma^2 = 4$ mm². Der Erwartungswert μ der Länge der Schrauben ist unbekannt. In welchem um den Mittelwert herum symmetrischen Intervall befindet sich mit einer Wahrscheinlichkeit von 95% der unbekannte Erwartungswert μ?
(Dieses Intervall heißt 95%iges Vertrauensintervall des Erwartungswertes μ.)

Aufgabe 4:

Eine Energiesparlampe leuchtet durchschnittlich insgesamt 10000 Stunden, bevor sie ausfällt, bei einer Standardabweichung von 800 Stunden.
a) Mit welcher Wahrscheinlichkeit leuchtet sie weniger als 9000 Stunden?
b) Mit welcher Wahrscheinlichkeit liegt die Leuchtdauer zwischen 9000 und 10500 Stunden?
c) Welche Leuchtdauer wird von 80% der Lampen nicht überschritten?
d) Welche Leuchtdauer wird von 90% der Lampen mindestens erreicht?
e) Mit welcher Wahrscheinlichkeit leuchtet eine Lampe nach 10500 Stunden immer noch, wenn sie 9000 Stunden bereits erreicht hat?

Aufgabe 5:

Eine Sorte Autoreifen erreicht im Mittel 40 000 km mit einer Standardabweichung von 4310 km.
Mit welcher Wahrscheinlichkeit
a) muß ein Reifen bereits nach 37 500 km gewechselt werden?
b) übersteht ein Reifen 45 000 km?
c) können alle vier Reifen mindestens 42 000 km am Fahrzeug verbleiben?
d) muß der erste Reifen spätestens nach 38 000 km gewechselt werden?
e) Welche Lebensdauer erreichen 80% der Reifen mindestens?

Quellen: Ueckerdt, B: Wahrscheinlichkeitsrechnung, Aufgabensammlung der HWR.
Papula, L.: Mathematik für Ing. und Naturwis., Bd. 3., Vieweg.

Übungsblatt 11

Numerische Differentiation und Integration

1. Berechnen der 1.Ableitung von y = f(x) = sin x im Intervall [0,π/2]

```
>> x = 0 : 0.01 : pi/2;  y = sin(x);
>> dyx = diff(y) ./ diff(x)                % Differenzenquotienten bilden
>> xr = x; xr(length(x)) = [];             % letzte Koordinate entfernen
>> plot(xr,dyx)                            % Grafik
```

Aufgabe 1:

Berechnen Sie numerisch die 1. und 2. Ableitung folgender Funktionen im angegebenen Intervall:

a) $y = f(x) = \cos(x)$, $[-\pi, \pi]$;
b) $y = g(x) = x / (1 + x^2)$, $[-10, 10]$;
c) $y = h(x) = (e^x – e^{-x}) / (e^x + e^{-x}) = \tanh(x)$, $[-10, 10]$;
d) $y = k(x) = x e^x$, $[-10, 10]$;
e) $y = l(x) = x \ln(x)$, $[0.1, 5.1]$.

Stellen Sie jeweils die drei Funktionen im angegebenen Intervall grafisch dar!

2. Berechnen bestimmter und unbestimmter Integrale von $y = f(x) = x^2 e^{-x^2}$

Erstellen eines M-Files: integrand.m

```
function  y = integrand(x)
y = x.^2.*exp(-x.^2) ;
```

```
>> x = linspace(0,1,101); y = integrand(x);
>> I0 = trapz(x,y)                  % Trapezregel        (1. Grades)
>> I1 = quad(`integrand`,0,1)       % adaptive Simpson-Quadratur (2. Grades)
>> I2 = quadl(`integrand`,0,1)      % adaptive Lobatto-Quadratur
```

Dazu die Stammfunktion numerisch berechnen und grafisch darstellen:

```
>> integral = zeros(1,101);
>> for i = 2 : 101 integral(i) = quad(`integrand`,0,x(i)); end;
>> plot(x,integral)                 % Grafik
```

3. Trapezregel, Simpson-Regel, Newtonsche 3/8-Regel, Milne-Regel, …

	Polynom	Teilintervalle	Schritt-weite h	Globale Fehlerordnung	Gewichte
Trapezregel	1. Grades	n	b-a	2	1/2, 1/2
Simpson-Regel (Keplersche Faßregel)	2. Grades	2 n	b-a	4	1/6, 4/6, 1/6
Newtonsche 3/8-Regel	3. Grades	3 n	b-a	4	1/8, 3/8, 3/8, 1/8
Milne-Regel	4.Grades	4 n	b-a	6	7/90, 32/90, 12/90, 32/90, 7/90
Weddle-Regel	6. Grades	6 n	b-a	8	...

Aufgabe 2:

Implementieren Sie die summierte Simpson-Regel selbständig und berechnen Sie die bestimmten Integrale für folgende Funktionen numerisch:

1. $f(x) = \sin x$, von 0 bis π,
2. $f(x) = \cos x$, von 0 bis $\pi/2$,
3. $f(x) = \mathrm{sqrt}(1 + \exp(0.5x^2))$, von 1 bis 2.6,
4. $f(x) = \tan x$, von -1 bis 0,
5. $f(x) = (1 - e^{-x}) / x$, von 1 bis 2,
6. $f(x) = \mathrm{sqrt}(1 + 2 x^4)$, von 1 bis 4,
7. $f(x) = x^3 / (e^x - 1)$, von 0.5 bis 1,
8. $f(x) = e^x / x^2$, von 1 bis 3.

Berechnen Sie zum Vergleich die entsprechenden Integrale mit den Programmen **trapz** und **quad**!

4. Monte-Carlo–Methode zur Berechnung bestimmter Integrale

Eine Funktion y = f(x) sei im Intervall [a,b] nichtnegativ und besitze keine Unstetigkeitsstellen zweiter Art. Zur Berechnung des bestimmten Integrals kann man dann auch wie folgt vorgehen:

1. Bestimmung des Funktionsmaximums M im Intervall [a, b].
2. Das Rechteck der Höhe M über dem Intervall [a,b] besitzt den Flächeninhalt A = M(b-a).
3. Mit Hilfe eines Zufallszahlengenerators werden n gleichmäßig verteilte 2D-Zufallsvektoren z = (x,y) mit $a \le x \le b$, $0 \le y \le M$ erzeugt. Die grafische Darstellung jedes dieser Zufallsvektoren ist ein Punkt im Rechteck.
4. Wir bezeichnen mit n die Gesamtzahl der Punkte im Rechteck und mit m die Anzahl derjenigen Punkte, die zwischen der x-Achse und der Grafik von f(x) (also unterhalb von f(x)) liegen. Dann gilt näherungsweise $\frac{I}{A} = \frac{m}{n}$

und I = A * m / n ist ein Näherungswert für das gesuchte Integral I.
Realisierung in OCTAVE: I = mcintgr('funktion', a, b, n)
Quelltext des Programms **mcintgr** (aus GNU Octave):

```
function I = mcintgr(fun, a, b, mcloops)                                #1
                                                                        #2
# Check input args                                                      #3
if ( nargin != 4 | nargout> 1 )                                         #4
usage("mcintgr is called with 4 inputs and 1 output");                  #5
endif                                                                   #6
                                                                        #7
# Check if user supplied function exists                                #8
if !exist(fun)                                                          #9
usage("mcintgr: Sure about the function name?");                        #10
elseif ( length(feval(fun,a)) != 1 )                                    #11
usage("Function passed to mcintgr must be a scalar function");          #12
                                                                        #13
endif                                                                   #14
                                                                        #15
# Find maximum value of f                                               #16
x = linspace(a,b);                                                      #17
y = feval(fun,x);                                                       #18
                                                                        #19
# Check if f is positive                                                #20
if ( min(y) < 0 )                                                       #21
usage("mcintgr: the function must be positive in the interval");        #22
                                                                        #23
endif                                                                   #24
                                                                        #25
# Set max of m                                                          #26
maxy = max(y);                                                          #27
                                                                        #28
# Calculate the interval                                                #29
l = b - a;                                                              #30
                                                                        #31
# Initialize the counters                                               #32
counter = 0;                                                            #33
nloops = 0;                                                             #34
                                                                        #35
# Main mc loop                                                          #36
while ( nloops<= mcloops )                                              #37
r1 = a + l*rand;                                                        #38
r2 = maxy*rand;                                                         #39
                                                                        #40
fr1 = feval(fun,r1);                                                    #41
if ( r2<fr1 )                                                           #42
counter++;                                                              #43
endif                                                                   #44
                                                                        #45
nloops++;                                                               #46
endwhile                                                                #47
                                                                        #48
# The integral                                                          #49
I = counter/mcloops*maxy*l;                                             #50
                                                                        #51
endfunction                                                             #52
```

Berechnen Sie die Integrale aus Aufgabe 2 mit Hilfe der Monte-Carlo-Methode!

Quellen:
Deuflhard, P., A. Hohmann: Numerische Mathematik I, de Gruyter, 2002
Lothar Papula: Mathematik für Ingenieure und Naturwissenschaftler, Bd. 1 Vieweg.
Schmidt Hansen, J: GNU Octave Beginner's Guide, Packt Publ., 2011.
Dieter Schott, Ingenieurmathematik mit MATLAB, Fachbuchverlag Leipzig im Carl Hanser Verlag, 2004.

Übungsblatt 12

Kodierung

Ein Ereignis A trete mit der Wahrscheinlichkeit p auf. Wir definieren ein Maß der Unsicherheit oder, mit anderen Worten, ein Maß für die Information, die die Beobachtung des Ereignisses A liefert:

$$\mathbf{I(A) = - ld(p) = - \log_2(p)}$$

Der Erwartungswert der Information, der bei der Beobachtung einer diskreten ZG X mit den Wahrscheinlichkeiten p_i, i = 1, …n, $p_1 + \ldots + p_n = 1$, erhalten wird, heißt **Entropie** der ZG X (*C. E. Shannon*):

$$\mathrm{Entr}(X) \;=\; -\sum_{i=1}^{n} p_i \,\mathrm{ld}(p_i).$$

Bei der Nachrichtenübertragung wird eine Information X gesendet und eine Information Y empfangen. Damit kann man verschiedene Entropien definieren:

Entr(X) – Erwartungswert der Information am Sender
Entr(Y) – Erwartungswert der Information am Empfänger
Entr(XY) – Charakterisierung der Übertragung
Entr(X | Y) – Bedingte Entropie unter der Bedingung, das Signal Y empfangen zu haben (charakterisiert die Möglichkeit, mit Hilfe des erhaltenen Signals Übertragungsfehler zu korrigieren).

Entr(Y | X) – Charakterisierung des Rauschens der Übertragung.

Damit kann man die wechselseitige Information der Übertragungsstrecke definieren (Information über X, die in Y enthalten ist)

$$\mathbf{I(X \mid Y) = Entr(X) - Entr(X \mid Y),}$$

die noch von der Verteilung der ZG X abhängt. Als Durchlaßfähigkeit der Übertragungsstrecke definieren wir jetzt das Maximum der wechselseitigen Information über alle möglichen Verteilungen der ZG X

$$C \;=\; \max_{p(x)} I(X \mid Y).$$

Shannon hat gezeigt, dass bei der Verwendung derjenigen Verteilung, die das Maximum liefert, der Rauscheinfluß minimiert bzw. sogar beseitigt werden kann.

Je näher man an diese Grenzverteilung herankommt, desto geringer ist der Rauscheinfluß und desto geringer ist die Fehlerwahrscheinlichkeit bei der Übertragung.
Solange die Entropie des Senders die Durchlaßfähigkeit der Übertragungsstrecke nicht übersteigt, also solange

$$\mathbf{Entr(S) \leq C}$$

gilt, solange gibt es die Möglichkeit, die Information des Senders mit Hilfe eines Eingangsalphabets zu kodieren. Die Code-Wörter müssen dabei so gewählt werden, dass das Verteilungsgesetz am Eingang der Grenzverteilung möglichst nahe kommt. Die Menge aller Code-Wörter wird mit **Code** bezeichnet. Bei einem ungleichmäßigen Code (verschiedener Längen) muß folgende Bedingung erfüllt sein, damit Anfang und Ende eines Zeichens eindeutig erkennbar sind:
Kein Code darf am Anfang eines anderen längeren Codes auftreten.

Fano - Code
Für eine symmetrische digitale Übertragung führt die 0-1-Verteilung mit p = q = 0.5 zu einem Maximum von I(X|Y). Die Übertragung jedes Bits liefert dann im Mittel die Information

$$\mathbf{Entr(1\ Bit) = -\ (\ 0.5\ \log_2(0.5) + 0.5\ \log_2(0.5)\ ) = 1}.$$

Fano hat folgendes Vorgehen bei der Kodierung vorgeschlagen:

1. Anordnen der Zeichen in der Reihenfolge fallender Wahrscheinlichkeiten;
2. Zerlegung in zwei möglichst gleichwahrscheinliche Teilmengen;
3. Jeder der beiden Teilmengen 0 bzw. 1 zuordnen;
4. Teilschritte 2. – 3. solange durchführen, bis alle Zeichen kodiert sind.

Aufgabe 1: Die Zeichen A, B, C, D, E und F treten mit folgenden Häufigkeiten auf:

A	B	C	D	E	F
0.5	0.25	0.10	0.08	0.05	0.02

Kodieren Sie die Zeichen nach der Methode von *Fano*.

Der Erwartungswert der Anzahl N der Bits pro kodiertes Zeichen ist

$$E(N) = \sum_{i=1}^{6} n_i p_i = 1.97.$$

Ideal wäre ein Wert von

$$Entr_{ideal} = -\sum_{i=1}^{6} p_i \log_2(p_i) = 1.9527$$

Informationseinheiten pro Zeichen. Am Verhältnis

$$\frac{\mathrm{Entr}_{\mathrm{ideal}}}{\mathrm{E(N)}} = \frac{1.952}{1.97} = 0.99$$

kann die Effektivität der Kodierung gemessen werden. Je näher das Verhältnis an der 1 ist, desto besser ist die Kodierung.

Aufgabe 2: Zur Zufallsgröße X am Eingang gehören folgende Wahrscheinlichkeiten:

X	x_1	x_2	x_3	x_4	x_5	x_6
P(X)	0.30	0.28	0.12	0.12	0.10	0.08.

Stellen Sie einen Fano-Code auf und schätzen Sie die Effektivität des Codes ein!

Aufgabe 3: Die ZG X am Eingang der Übertragungsstrecke genügt folgender Verteilung:

X	x_1	x_2	x_3	x_4
P(X)	0.1	0.2	0.3	0.4

Die ZG Y am Ausgang der Übertragungsstrecke nimmt nur drei Werte an (was zugegebenermaßen recht schlecht ist).
Die Übertragungsmatrix **T** mit $p^i_j = P(Y = y_j \mid X = x_i)$ ist

$$T = \left(p^i_j\right) = \begin{pmatrix} 0.50 & 0 & 0.50 \\ 0.20 & 0.40 & 0.40 \\ 0.50 & 0.25 & 0.25 \\ 0 & 0.50 & 0.50 \end{pmatrix}.$$

Berechnen Sie
a) die Verteilung der ZG Y am Ausgang;
b) die zweidimensionale Verteilung des Vektors (X,Y);
c) die Matrix Q der bedingten Wahrscheinlichkeiten

$$Q = q^j_i = P(X = x_i \mid Y = y_j)!$$

d) Schätzen Sie den Informationsverlust ein.

Aufgabe 4: Die ZG X am Eingang der Übertragungsstrecke genügt folgender Verteilung:

X	x_1	x_2	x_3	x_4	x_5	x_6
P(X)	0.30	0.28	0.12	0.12	0.10	0.08

Die ZG Y am Ausgang nimmt ebenfalls sechs Werte an.

Die Übertragungsmatrix **T** mit $p^i_j = P(Y = y_j \mid X = x_i)$ ist

$$T = \left(p^i_j\right) = \begin{pmatrix} 0.60 & 0.35 & 0.03 & 0.02 & 0 & 0 \\ 0.10 & 0.45 & 0.20 & 0.15 & 0.10 & 0 \\ 0.10 & 0.15 & 0.50 & 0.15 & 0.05 & 0.05 \\ 0 & 0.05 & 0.15 & 0.60 & 0.15 & 0.05 \\ 0 & 0 & 0.07 & 0.18 & 0.55 & 0.20 \\ 0 & 0 & 0.08 & 0.12 & 0.25 & 0.55 \end{pmatrix}.$$

Berechnen Sie
a) die Verteilung der ZG Y am Ausgang;
b) die zweidimensionale Verteilung des Vektors (X,Y);
c) die Matrix Q der bedingten Wahrscheinlichkeiten

$$Q = q^j_i = P(X = x_i \mid Y = y_j)!$$

d) Schätzen Sie den Informationsverlust ein.

Dekodierung und Fehlerkorrektur

1. Lineare Codes 2. Zyklische Codes

Es sei $\mathbf{e_i}$ der Zeilenvektor $\mathbf{e_i} = (0\ 0\ \ldots 0\ 1\ 0\ \ldots\ 0)$. Alle Vektoren $\mathbf{e_i}$, i = 1, 2,…n,
| i-te Stelle
bilden eine Basis im $\mathbb{R}^n$. Es sei $\mathbf{a} = (a_1, a_2, \ldots a_n)$ ein Vektor im $\mathbb{R}^n$. Dieser Vektor wird von der nxn-Einheitsmatrix **E** in den Vektor $\mathbf{v} = (v_1, v_2, \ldots v_n)$ überführt. Dann gilt auch **a** = **v**, was einer sicheren Übertragung entspricht. Die Übertragung kann aber auch durch eine nxn-Matrix **T** gekennzeichnet sein, die nicht gleich der Einheitsmatrix ist. Dann ist **v** ≠ **a**, was einer fehlerhaften Übertragung entspricht. Wir wollen jetzt ein m-stelliges digitales Zeichen durch ein n-stelliges digitales Zeichen (n > m) ersetzen, so dass eine Redundanz entsteht. Die zugehörige Übertragungsmatrix vom Rang m nennen wir $\mathbf{G^{\#}}$:

$$G^{\#} = \begin{pmatrix} 1 & 0 & \ldots & 0 & \\ 0 & 1 & \ldots & 0 & \\ . & . & & & a_{ij} \\ 0 & 0 & \ldots & 0 & \\ 0 & 0 & \ldots & 1 & \end{pmatrix} \begin{matrix} \\ \\ \\ \\ \\ \end{matrix} \text{m Zeilen}$$

m Spalten n-m Spalten

Die ersten m Stellen dieses **systematischen Codes** sind die **Informationsstellen**, die anderen n-m Stellen die **Kontroll- oder Prüfstellen**. Im Fall

digitaler Zeichen sind also 2^m verschiedene Worte möglich. Wir ergänzen jetzt die mxn-Matrix $\mathbf{G}^{\#}$ zu einer neuen nxn-Matrix

$$
\begin{array}{l}
\\
G^{\#} \rightarrow \\
\\
\\
\\
H^{\#} \rightarrow \\
\\
\\
\\
\end{array}
\begin{pmatrix}
1 & 0 & \ldots & 0 & & & & \\
0 & 1 & \ldots & 0 & & & & \\
\cdot & \cdot & \ldots & \cdot & & a_{ij} & & \\
0 & 0 & \ldots & 0 & & & & \\
0 & 0 & \ldots & 1 & & & & \\
 & & & & 1 & 0 & \ldots & 0 \\
 & & & & 0 & 1 & \ldots & 0 \\
b_{ji} & = & -a_{ij} & & & & & \\
 & & & & 0 & 0 & \ldots & 0 \\
 & & & & 0 & 0 & \ldots & 1
\end{pmatrix}
$$

m Spalten n-m Spalten

Für die beiden Matrizen $\mathbf{G}^{\#}$ und $\mathbf{H}^{\#}$ gilt die folgende Beziehung, die wir zur Fehlersuche ausnutzen:

$$\underset{mxn}{\mathbf{G}^{\#}} * \underset{nx(n-m)}{\mathbf{H}^{\#\prime}} = \underset{mx(n-m)}{(\underset{(n-m)xn}{\mathbf{H}^{\#}} * \underset{nxm}{\mathbf{G}^{\#\prime}})'} = \mathbf{0}.$$

Es sei jetzt $\mathbf{v} = \mathbf{a} * \mathbf{G}^{\#}$, $\mathbf{a}$ ist der m-dimensionale Vektor, der in den n-dimensionalen (redundanten) Vektor $\mathbf{v}$ übergeht. Dann ist

$$\underset{1xn}{\mathbf{v}} * \underset{nx(n-m)}{\mathbf{H}^{\#\prime}} = \underset{1xm}{\mathbf{a}} * \underset{mxn}{\mathbf{G}^{\#}} * \underset{nx(n-m)}{\mathbf{H}^{\#\prime}} = \underset{1x(n-m)}{\mathbf{0}},$$

d.h. $\mathbf{H}^{\#\prime}$ überführt jeden Vektor $\mathbf{v}$ in einen Nullvektor. Dasselbe kann in anderer Form geschrieben werden, nämlich

$$\underset{(n-m)xn}{\mathbf{H}^{\#}} * \underset{nx1}{\mathbf{v}'} = \underset{(n-m)x1}{\mathbf{0}}.$$

Diese Gleichung stellt ein homogenes (n-m)-dimensionales lineares Gleichungssystem mit (n-m) Variablen dar. Die (n-m)-dimensionalen Vektoren, deren Komponenten Zeichen der Code-Worte auf den Prüfstellen sind, sind Lösungen des homogenen Gleichungssystems.

Fehlererkennung:

Wenn jetzt am Ausgang der Übertragungsstrecke ein Vektor $\mathbf{w}$ erscheint, für den

$$\mathbf{H}^{\#} * \mathbf{w}' \neq \mathbf{0}$$

ist, so wird ein Fehler entdeckt.

Bemerkung: Immer dann, wenn $\mathbf{H^{\#} * w' = 0}$ ist, kann also davon ausgegangen werden, dass kein Fehler des vorher definierten Typs aufgetreten ist. Eine Garantie für das Fehlen anderer Fehler gibt es jedoch nicht.

Aufgabe 5: Implementieren Sie die folgenden 3 Beispiele und überprüfen Sie die Fehlererkennung und die Möglichkeit der Fehlerkorrektur!

1. Beispiel eines linearen digitalen Codes, der einfache Fehler erkennt und korrigiert

(*Hamming*-Code zur Erkennung und Korrektur einfacher Fehler ($\mathbf{e = 1}$))

Um in einem kodierten Wort der Länge n einen einfachen Fehler zu erkennen, muß die Bedingung $n \leq 2^{n-m} - 1$ erfüllt sein. Für ein Wort der Länge m = 4 ist $n = 2^{7-4} - 1 = 7$.
Als *Hamming*-Abstand wird die Anzahl der nicht übereinstimmenden Bits zweier Worte bezeichnet. Es sei **d** der *Hamming*-Abstand einer Spalte einer Matrix zum Null-Vektor. Damit können weitere Bedingungen an die Matrix $\mathbf{-A'}$ angegeben werden, z. B. muß in $-\mathbf{A'}$

- in jeder Spalte mindestens 2 mal eine Eins vorkommen ($\mathbf{d - 1 \geq 2e = 2}$);
- das array-Produkt je zweier verschiedener Spalten mindestens ein nicht-null-Element enthalten ($\mathbf{d - 2 \geq 1}$);
- das array-Produkt je dreier verschiedener Spalten mindestens null nicht-null-Elemente enthalten ($\mathbf{d - 3 \geq 0}$).

Die Matrix

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

erfüllt diese Bedingungen. Damit kann die Prüfmatrix $\mathbf{H^{\#}}$ gebildet werden:

$$H^{\#} = \begin{pmatrix} -A' & E \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Wir stellen die Matrix $\mathbf{H^{\#}}$ nochmals so um, dass die Spalten der Prüfmatrix **H** der Reihe nach die digitalen Codes der Zahlen von 1 bis 7 darstellen:

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

Dann erhält man aus dem Produkt **z = H * v'** den Spaltenvektor **z**, der bei Auftreten eines (einfachen) Fehlers die digitale Stelle des aufgetretenen Fehlers anzeigt!
Die Informationsmatrix $\mathbf{G^{\#} = (E \mid A)}$ wird entsprechend $\mathbf{H^{\#}}$ ebenfalls umgestellt:

$$G = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Mit digitalen Worten der Länge 4 kann ein Alphabet mit $2^4 = 16$ Zeichen übertragen werden. Diese 16 Zeichen können mit Hilfe der Informationsmatrix **G** in die entsprechenden 16 Code-Zeichen der Länge 7 umgewandelt werden.

Beispiel: Das Zeichen **a** = (1 0 1 1) erhält die Codierung **c** = (0 1 1 0 0 1 1). Wird jetzt z. B. das dritte Bit verändert, also **w** = (0 1 **0** 0 0 1 1), so liefert der Prüfvorgang

$$H \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}.$$

Damit wird angezeigt, dass ein Fehler im dritten Bit aufgetreten ist.
Hinweis: Bei der Arbeit mit OCTAVE müssen alle Matrixoperationen unter Benutzung der Funktion mod durchgeführt werden, also aus a*G = c wird

c = mod(a*G,2).

2. Beispiel eines linearen digitalen Codes, der einfache Fehler korrigiert und zweifache Fehler erkennt

(*Hamming*-Code zur Erkennung zweifacher Fehler)

Die Code-Worte der Länge n müssen jetzt um ein Bit erweitert werden, auf dem jeweils eine 1 steht. Wir betrachten den Fall m = 4 und n = 8.
Für jedes Code-Wort muß zusätzlich die Bedingung gelten:

Summe aller Bits = 0 (modulo 2).

Ähnlich dem vorherigen Beispiel erhält man die Prüfmatrix **H**

$$H = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Die Informationsmatrix **G** lautet dann

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

Beispiel: Das Zeichen **a** = (1 0 1 1) erhält die Codierung **c** = (1 0 1 0 0 1 0 1). Wird jetzt z. B. das vierte Bit verändert, also **w** = (1 0 1 **1** 0 1 0 1), so liefert der Prüfvorgang

$$H \cdot \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}.$$

Es sei **s** die Zahl, die durch die ersten drei Bits gegeben ist ($0 \leq \mathbf{s} \leq 7$).
Ist das vierte Bit gesetzt (also = 1), handelt es sich um einen einfachen Fehler. Die Stelle des Fehlers ist dann **s + 1**. Ist s > 0 und das vierte Bit gelöscht (= 0), handelt es sich um einen zweifachen Fehler.

3. Beispiel eines zyklischen digitalen Codes, der einfache Fehler erkennt und korrigiert

Es soll das Zeichen **a** = (1 0 0 1) kodiert werden. Wir konstruieren wieder einen Code für m = 4 mit n = 7 und k = n – m = 3. Wir betrachten das Polynom

$$p(z) = z^3 \# z \# 1,$$

das keine Nullstellen 0 oder 1 besitzt. Weiter sei

$$M(z) = z^3 \# 1$$

das zu **a** gehörende Polynom. Wir führen eine Polynomdivision von $z^k \cdot M(z)$ durch p(z) durch. Der Divisionsrest ist $r(z) = z^2 \# z$:

$$z^3 M(z) = z^6 \# z^3 = p(z) \cdot (z^3 \# z) \# (z^2 \# z)$$

Wir berechnen

$$C(z) = z^k \cdot M(z) – r(z) = z^6 \# z^3 \# z^2 \# z.$$

Der erhaltene Code ist **c** = (1 0 0 1 1 1 0). Die ersten vier Bits sind die Informationsbits, die weiteren drei die Prüfbits. Eine fehlerhafte Übertragung führte zum fehlerhaften Code **w** = (1 0 **1** 1 1 1 0), der dem Polynom

$$C^*(z) = z^6 \# \mathbf{z^4} \# z^3 \# z^2 \# z$$

entspricht. Die Polynomdivision von C*(z) durch $p(z) = z^3 \# z \# 1$ ergibt einen Rest $z^2 \# z$, der nicht Null ist und somit auf einen Fehler hinweist, denn bei der Division von C(z) durch p(z) bleibt kein Rest.

Quelle: Kaufmann, A: Introduction a la combinatorique en vue des applications, Dunod, Paris, 1968.

Übungsblatt 13

Lineare Regressionsrechnung

Aufgabe 1: Gegeben seien folgende Messwerte für die mittlere Stammhöhe von Kiefern [m] in Abhängigkeit von ihrem mittleren Stammdurchmesser [cm]:

i	1	2	3	4	5	6	7	8	9	10
x_i	10,6	14,0	18,1	23,2	25,0	26,4	30,5	32,5	36,6	40,1
y_i	8,6	11,5	12,4	15,6	15,1	17,7	18,9	18,6	21,3	24,3

Datenquelle: Kleine Enzyklopädie Mathematik, Leipzig 1969.

Gesucht ist eine **lineare Funktion y = f(x) = a + b x**, die diese Daten möglichst gut approximiert.

Lösungsweg mit OCTAVE:

1. Abspeichern der y-Meßwerte in einem Spaltenvektor **ym**;
2. Abspeichern der x-Meßwerte in einem Spaltenvektor **xm**;
3. Erstellen einer Matrix **A**, in deren 1. Spalte nur Einsen und deren 2. Spalte die x-Meßwerte stehen;
4. Der gesuchte Vektor der Regressionskoeffizienten sei **p = [a b]'**.
   Das Gleichungssystem **A*p = ym** ist überbestimmt und deshalb im üblichen Sinne (normalerweise) nicht lösbar, da (normalerweise) nicht alle Meßpunkte $P_i = (x_i, y_i)$, i = 1, … n, auf einer Geraden liegen.
   Vielmehr gilt eine Gleichung der Art **A*p – ym = r** mit dem Residuumsvektor **r**, der minimiert werden soll.
5. Wir multiplizieren beide Seiten der Gleichung mit **A'**, d.h. wir bilden eine neue 2x2-Matrix **B = A'*A** und einen 2-dimensionalen Vektor **d = A'*ym**, so daß das **Normalgleichungssystem B*p – d = 0** gelöst werden kann.
   Wenden Sie dazu nacheinander folgende mögliche Verfahren an:
   a) Normalgleichungen, LU-Zerlegung **[L,U] = lu(B)**
   b) Normalgleichungen, Cholesky-Zerlegung **U = chol(B)**
   für die positiv definite symmetrische Matrix **B = U'*U**.
6. Eine weniger fehlerbehaftete Möglichkeit besteht darin, anstelle des Normalgleichungssystems das **Fehlergleichungssystem A*p – ym = r** direkt zu lösen, indem mit der Matrix **A** eine **Orthogonalisierung** nach *Householder* (QR-Zerlegung der Matrix A) durchgeführt wird:
   c) Fehlergleichungssystem, QR-Zerlegung **[Q,R] = qr(A)**.
7. Mit der Befehlszeile **p = A \ ym** wird automatisch eine *Householder*-Transformation der Matrix **A** durchgeführt und das entsprechende Quadratmittelproblem gelöst. Das Ergebnis ist der gesuchte Vektor p = [a b]'.
8. Ist nicht bekannt, ob y überhaupt von x abhängt, kann auch
   d) eine **Singulärwertzerlegung** von A = U*S*V durchgeführt werden:
   [U,S,V] = svd(A).

9. Grafische Darstellung der Messdaten und der Regressionsfunktion.

Aufgabe 2: Bestimmen Sie eine lineare Regressionsfunktion!

x_i	1.3	2.2	2.9	3.1	4.5	5.7	7.1	8.0	8.7	8.9	9.3	9.9
y_i	1.3	1.0	0.85	0.80	0.33	0	-0.4	-0.7	-0.9	-1.0	-1.1	-1.2

Aufgabe 3: Bestimmen Sie eine quadratische Regressionsfunktion!

x_i	1	2	3	3.5	4.5	5.5	6	7	8	8.5	9.5	10
y_i	11	7.3	4.8	4.1	1.0	0	0.6	2	3.7	4.2	5.6	8

Aufgabe 4: Bestimmen Sie eine quadratische Regressionsfunktion!

x_i	0.04	0.32	0.51	0.73	1.03	1.42	1.60
y_i	2.63	1.18	1.16	1.54	2.65	5.41	7.67

Datenquelle: Schwarz, H. R.: Numerische Mathematik, Teubner 1997.

Aufgabe 5: Gegeben seien folgende Messwerte:

x_i	1	2	3	4	5
y_i	1.8395	0.6765	0.2490	0.0915	0.0335

Finden Sie eine nichtlineare Funktion $y = f(x) = a\, e^{bx}$, die diese Daten möglichst gut approximiert.

Hinweis: Logarithmieren Sie dazu beide Seiten der Gleichung.