Matlab

Firma: Mathworks, www. mathworks.de bzw. www. mathworks.com

Basiert auf Vektor- und Matrixoperationen, ursprünglich für lineare Algebra
Heute: Softwarepaket zur numerischen Berechnung und Visualisierung ,

- ergänzt mit vielen Zusatzmodulen = Toolboxen
  z.B. für Computeralgebra: Symbolic Math Toolbox
- einer grafischen Entwicklungsumgebung Simulink,
- rechenzeitoptimierten Built-In-Functions für Matrizenoperationen,
- einer eigenen Hochsprache mit interaktiver Umgebung,
- und eigenem Texteditor zur Programmerstellung mit
- einem Debugger zur Prüfung des Quellcodes.

Matlab hat eine weite Verbreitung im Ingenieurbereich, eine Studentenversion ist über die Hochschule erhältlich. Sie wird über die Landeslizenz für Sachsen realisiert. Bitte unter https://www.ncc.hs-mittweida.de/software/matlab.html herunterladen.
(Frei erhältlich ist auch der Matlab-Clone: Octave bei http://www.gnu.org/software/octave/
Dieser enthält zur Erstellung von Grafiken Gnuplot.)

Notwendig vor Arbeitsbeginn:

1. Arbeitsbereich, z.B. I:\matlab, einrichten
2. startup.m muss dorthin kopiert werden und mit den eigenen Pfaden versehen werden:
   path(path, 'I:\matlab');

Befehle werden im Command Window eingegeben und mit der Enter-Taste bestätigt.
Der Befehl path zeigt alle eingestellten Pfade an, zuletzt die eigenen.
Befehle, die in einer .doc oder .docx-Datei geschrieben sind, können Sie mit „Crtl C“ und „Ctrl V“ in das Command Window kopieren.

Desktop

Grundeinstellungen: Desktop – desktop layout – default mit

- Command Window
- Current Folder
- Workspace
- Command History

Auskoppeln eines Fensters: ↗ – einkoppeln: im Fenster zu Desktop gehen: ↘

Hilfe über das Command-Fenster und Online-Hilfe für Funktionen in Matlab:

- help - help name: listet Kommentar im Funktionskopf
- helpwin – helpwin name: listet Hilfe
- doc – doc name: Dokumentation wird geöffnet
- lookfor name: suchen nach name

Demos

help ops oder help ? : Übersicht über alle möglichen Funktionen (dauert!)

Allgemeines

who – whos: Variablenliste – mit Attributen

what: Liste der m-Files und mat-Files im aktuellen Directory

which name: Klassifizierung von name nach variable, File oder Matlabfunktion

exist name: Test auf Existenz von name

Variablen der Sitzung ➔ Workspace, dort editieren möglich und veranschaulichen

clear – clear name

variablenname = besteht aus den Einzelzeichen Buchstabe oder Zahl oder Unterstreichstrich
- Unterscheidung zwischen Groß- und Kleinbuchstaben
- diese wird aufgehoben durch casesen
- keine Leerzeichen, weil Leerzeichen ein Trennzeichen ist

Definition ist nicht erforderlich, nimmt Matrix der erforderlichen Größe mit komplexen Argumenten. Definition erfolgt damit implizit und lokal.
➔ Dimensionsverträglichkeit beachten

Befehlseingabe zeilenweise oder Trennung mit Komma

Enter bewirkt Ausführung
Umsch + Enter ist Zeilenumbruch ohne Ausführung

Befehl kann weitergeführt werden über das Zeilenende durch „nächstes Operationszeichen…“

vorhergehender Befehl: ↑

„!“ ➔ Übergang zur Betriebssystemebene

vordefinierte Variablen: i, j, pi, inf, NaN, eps

Funktionen: Argument ist prinzipiell eine Matrix mit möglicherweise komplexen Elementen,
prinzipiell elementweise Ausführung
help elfun: listet elementare Funktionen
help datafun: listet Funktionen für Datenanalyse

ans : Variable answer = automatische Antwortvariable des Systems bei Berechnungen

quit beendet Matlab-Sitzung
Zuweisungen an Variable gehen bei quit verloren

Ctrl + C unterbricht Programmabarbeitung, z. B. bei Endlosschleife

Datentypen/Zahldarstellung

Zahlen oder Zeichenkette

eps ~ 2.2e-16 (rel. Genauigkeit des Gleitkommas)
realmin
realmax

interne Zahldarstellung: 16 Dezimalen von etwa -2e308 bis +2e308
Punkt statt Komma in Zahl, weil Komma Trennzeichen ist

Datentypen zur Darstellung auf BS oder Drucker: double ➔ 8 Bytes
single ➔ 4 Bytes

Potenzdarstellung: 1.4e-3, 3.456E4 (ohne Leerzeichen dazwischen, sonst Fehler!)

bei komplexen Zahlen zwischen Faktor und i bzw j kein Leerzeichen: 3i oder -5j

Skriptdatei: (Format.m)

```
x=[4/3 1.2345e-6];
format short;disp('format short: ');x
format short e;disp('format short e: ');x
format short g;disp('format short g: ');x
format long;disp('format long: ');x
format long e;disp('format long e: ');x
format long g;disp('format long g: ');x
format bank;disp('format bank: ');x
format rat;disp('format rat: ');x
format hex;disp('format hex: ');x
format compact;disp('format compact: ');x % unterdrückt Leerzeile bei Echo-Ausgabe
format short;
```

Eingabe / Ausgabe

; am Ende eines Befehls unterdrückt das Echo des Systems zu diesem Befehl

x (Standardausgabe mit einer neuen Zeile)

disp(['x = ', num2str(x',3)]) % Variable muss Zeilenvektor sein!, Umwandlung in Zeichenkette

disp([' x = ', sprintf(' %3d',x)]) % Ausgabe mit speziellem Format: 3 Stellen integer mit Vorzeichen

Rundung u.ä.

```
round([pi,exp(1)]) % runden
floor([pi,exp(1)]) % abrunden zur nächsten ganzen Zahl
ceil([pi,exp(1)]) % aufrunden
```

File – New – m-File

```
u=312; v=1.234; w=1.034e-05;
disp([' u = ',sprintf(' %5d ',u)]) % Ausgabeformat 5 Stellen Festkomma

disp([' v = ',sprintf(' %5.4f ',v)]) % Ausgabeformat 5 Stellen Festkomma mit 4
                                     % Nachkommastellen

disp([' w = ',sprintf('\n  %5.4e ',w)]) % Ausgabeformat wie oben aber mit
                                       % Exponentialdarstellung, zusätzlich neue Zeile

disp(sprintf('u=%5d v=%5.4f w=%5.4e \n',u,v,w))
disp(sprintf('u=%5.4f  u=%5.4e ',u,u))
```

File – Save – Im Arbeitsdirectory unter selbst gewählten Namen ablegen: ergibt ein Skriptfile name.m

Skriptfile:

```
%Speichern
clear; % Löschen aller Variablen
a=linspace(0,10,5); % erzeugt Vektor mit 5 Komponenten äquidistant zwischen 0 und 10
a
save; % schreibt alle Daten des Workspace auf das Standardfile matlab.mat
clear; % Löschen aller Variablen
a=1
load; % Lesen der Daten
who
a
```

File– Import/Export Data – Save Workspace as

```
doc save
doc load
help iofun
```

diary on – diary off ➔ Nachbereitung notwendig!

- ➔ evtl. besser: eigenes Sitzungsfile anlegen
- ➔ dort Befehle aufschreiben, zum Test herauskopieren
- ➔ Befehle im Sitzungsfile korrigieren bis sie richtig sind
- ➔ Speichern zwischendurch nicht vergessen!

Vektoren und Matrizen

Befehle, die in einer .doc oder .docx-Datei geschrieben sind, können Sie mit „Crtl C“ und „Ctrl V“ in das Command Window kopieren.

Eingabe in eckigen Klammern
Elementtrennzeichen: Leerzeichen oder Komma
Zeilenende: Semikolon

Elemente dürfen mathematische Ausdrücke sein

Matrizen können durch Anfügen vergrößert werden: Konkatenation

Zugriff auf Matrixelemente: a(i,j)

Auswahl von Teilen einer Matrix mittels „Colon“-Operator :

Anfangsindex : Schrittweite : Endindex
(Standard=1)

„ : “ ➔ wählt alle definierten Elemente aus

```
A=[1,2,3;4,5,6;7,8,9]
B=[4,1,0;1,4,1;0,1,4]
A(1,3)

x=0:0.1:1
x=pi*(0:0.1:1)
a=1:5
b=3:0.5:7;
x=[a,b]

A([1,2],2)
A(1:2:3,:)
A(1:2:3,1:2:3)
A([1,2],[1,3])
```

Unterscheidung: **Matrix** **- Array**
Matrix-Operationen - Punkt-Operationen, elementweise Ausführung
Dimensionen beachten!!

Nutzung als Array:

```
exp([0:0.5:1])
abs([-5,pi,i])
asin([-1:0.5:1])
sin([-1:0.5:1])
sqrt(a)
sign([-3:3])
```

Matrizenoperationen und Funktionen von Matrizen:

A* = A’ (wenn A reell, gilt dann A^T=A’, Hochkomma über # auf der Tastatur)
A^T = A.’

```
x, y=x’
A+B
A-B
```

```
A*B
A.*B
ans'
```

A^{-1} = inv(A)
det A = det(A)
Eigenwerte der Matrix A : eig(A); [v,d]=eig(A)➔ v=Eigenvektoren, d=diag(Eigenwerten)
Rang(A)=rank(A)

```
det(A)
size(A)
size(A,1), size(A,2)
cond(A)
rank(A)
inv(A)
```

sum(vektor), prod(vektor)
min(vektor), max(vektor), max(matrix(:))
sort(vektor)
find(vektor)

```
a=[3,5,1,9,7];
max(a), min(a)
sum(a)
sort(a)
```

eye(m,n), zeros(m,n), ones(m,n)
.....

```
n=5;m=3;
eye(n)
zeros(n,m)
ones(n,m)

a=[1;2;3]; diag(a)

triu(A)
tril(A)
rand(n)
randi(10,n)
hilb(n)
magic(n)

[L,U]=lu(A)
[VA,DA]=eig(A)
```

Ax=b ➔ $x=A^{-1}b$ ➔x=A\b aber A.\B =(B(i,j)/A(i,j)) für alle i,j

XM=P ➔ $X=PM^{-1}$ ➔ X=P/M

Aber P./M=(P(i,j)/M(i,j)) für alle i,j liefert elementeweise Division!

(Das obere Ende des Divisionszeichens zeigt auf die Matrix^{-1} bzw. auf den Nenner bei Punktoperationen!)

```
b=[4;4;4];
x=A\b, y=B\b
```

Beispiele und Praktikum Matrizen

Geben Sie folgende Matrizen ein: $DU = \begin{pmatrix} 16 & 3 & 2 & 13 \\ 5 & 10 & 11 & 8 \\ 9 & 6 & 7 & 12 \\ 4 & 15 & 14 & 1 \end{pmatrix}, V = \begin{pmatrix} 1-5i \\ 2-i \\ 3+7i \end{pmatrix}.$

1. Erforschen Sie folgende Befehle:

```
v=0:2:8, v1=8:-2:0, v2=1:7
C=[[1:2:5]';[1 4 9]']
D=[[1:2:5]',[1 4 9]']
CS=sum(C)
E=diag(DU)
DIA=diag(E)
Dd=diag([2 4 6 8 ])+diag([-3 -3 -3],1)+diag([-1 -1 -1],-1)
Dd(3,1)=7;Dd(4,2)=12; Dd
diag(Dd,-2)
Dd(:,2)
Dd(4,2:3)
Dd(4,2:3)=[7 0];Dd
DF=fliplr(DU)
```

2. Stellen Sie die transponierten und adjungierten Matrizen von DU und V her und weisen Sie sie neuen Variablen DUT, DUA, VT, VA zu. Bilden Sie die Spalten- und Zeilensummen von DU sowie die Summen der Haupt- und Nebendiagonalen und die Gesamtsumme aller Elemente von DU. Was stellen Sie fest? Bilden Sie DU^{-1} und untersuchen Sie mittels Rang und Determinante, ob die Inverse brauchbar ist.
(Die Matrix DU ist ein magisches Quadrat aus einem Kupferstich von Dürer: Melancholie I)

3. Weisen Sie einer Variablen zu und listen Sie auf:
a) alle Zahlen von 1 bis 10
b) alle Zahlen von 100 abwärts im Abstand von 7 bis zur 50
c) alle Zahlen von 0 bis $2*\pi$ im Abstand von $\pi/4$
d) die Elemente 1 bis 3 der Spalte 4 von DU
Stellen Sie diese Variablen nach Ihrem Geschmack in einem Diagramm dar.

4. a) Sortieren Sie die Spalten von DU in der Reihenfolge 1, 3, 2, 4.
b) Geg: $A = \begin{pmatrix} 2 & 4 \\ 7 & 5 \end{pmatrix}; B = \begin{pmatrix} 4 & 2 \\ 5 & 7 \end{pmatrix}$ Ges.: $Q = \begin{pmatrix} O_{2x2} & E_{2x2} \\ -A^{-1}B & BA^{T} \end{pmatrix}$
c) Löschen Sie in Q die Spalte 2
d) Bilden Sie $B^{-1}A$ und $A\,B^{-1}$

5. Listen Sie die Variablen mit ihren Dimensionen auf und suchen Sie
a) zwei Vektoren, deren Skalarprodukt Sie bilden.
b) zwei Variablen, die Sie mittels Matrizenprodukt multiplizieren können.
c) zwei Variablen, die Sie addieren oder subtrahieren können.

Verwenden Sie gegebenenfalls Teile von Variablen oder definieren Sie neue passender Größe.

d) Lösen Sie das lin. GlS Bx=v(1:2)' mittels Inversion und mittels linsolve
e) Lösen Sie das lin. GlS Bx=0.5v(1:2)' mittels Inversion und mittels linsolve

Grafikanwendungen in MATLAB / 2D und Animation

Termin zur Bearbeitung: 16.04.20

*(Mit *) gekennzeichnete Inhalte sind fakultativ!)*

Für Präsentationen, Kursarbeiten und Ihre Abschlussarbeit ist es oft von Vorteil, wenn man Zusammenhänge durch eine grafische Darstellung untermauern kann. Deshalb beschäftigen wir uns in diesem Praktikum mit der Visualisierung von mathematischen Funktionen u. ä.

Schauen Sie sich zunächst bitte das kleine Einführungsvideo EinführungMATLABInfosim2.mp4 (Laufwerk r.\...\Arbeitsmittel) an, machen Sie sich Notizen zu den dort vorgestellten Möglichkeiten des Plottens einer Funktion und arbeiten Sie danach die unten angegebenen Beispiele ab.

1. Zeichnen der Normalparabel: x=0:0.1:2;y=x.^2;plot(x,y)

Ein neuer plot-Befehl löscht das bestehende Grafikfenster und erzeugt ein neues Bild:
z=sqrt(x);plot(x,z)

"hold on" hält das Grafikfenster für weitere plot-Befehle offen => Abschluss mit "hold off" beendet die Möglichkeit etwas dazu zu zeichnen:
plot(x,y),hold on,plot(x,z), hold off

2. Mehrere Kurven sofort in einer Zeichnung: plot(x,y,'r',x,z,'--')
(1. Kurve: rot durchgezogen, 2. Kurve: Standardfarbe blau, gestrichelt)

3. Sinnvolle Ergänzungen (wirken auch ohne „hold on“ auf die aktuelle Zeichnung):
grid %Gitter einzeichnen
title('Wurzel- und Quadratfunktion') % Überschrift
xlabel('x');ylabel('y') % Achsenbeschriftung
axis([0 2 0 9]) % eigener x-Bereich: [0,2], y-Bereich: [0,9]
gtext({'Wurzel'; 'Quadrat'; }) % Platzierung von Text an der Kursurposition
legend

Informieren Sie sich über den Befehl gtext und die möglichen Farben und Stricharten beim plot-Befehl in der Dokumentation und/oder im File MatlabUebersicht.pdf, das Sie im Laufwerk Arbeitsmittel finden.

Erkunden Sie die Möglichkeiten, eine bereits bestehende Zeichnung im Zeichnungseditor nachzubereiten!

Schließen Sie das Bild.

4. Suchen Sie die mathematische Definition der Funktion "humps"
doc humps; dort: view code ➔ y = 1 ./ ((x-.3).^2 + .01) + 1 ./ ((x-.9).^2 + .04) - 6;
und stellen Sie sie
a) im bereits benutzten x-Bereich [-1,2] mit Standardeinstellungen dar: y=humps(x);plot(x,y)
b) im x-Bereich [-1,2] mit 30 Stützwerten: x=linspace(-1,2,30); y=humps(x); plot(x,y)

„Knicke“ im Funktionsgraphen weisen auf eine zu geringe Anzahl von Stützwerten im x-Bereich hin!

5. Zeichnen Sie die ausgefüllte Sinusfunktion: x=0:pi/100:2*pi;y=sin(x);area(x,y)

6. Zeichnen Sie die Sinusfunktion punktweise im Intervall [0, 2*pi], Symbol für den Funktionswertpunkt: “ *„

```
x=linspace(0,2*pi,200); plot(x,sin(x),'*')
oder    i=0:200; x=i*pi/100; plot(x,sin(x),'*')
```

7. **Zeichnen von Polynomen:** $P_3(x)=1x^3 + 2x^2 - 1x + 5$
Achtung: In dieser Form ist die Polynomwertberechnung numerisch instabil!==> Deshalb:
x=linspace(-100,70,200);c=[1 2 -1 5];y=polyval(c,x);plot(x,y)
Die Funktion polyval(c,x) berechnet den Funktionswert des Polynoms rundungsfehlerstabil mittel Hornerschema an der Stelle x (kann auch ein x-Vektor sein), wenn in c die Koeffizienten des Polynoms in absteigender Reihenfolge der Potenzen von x stehen.
Achtung „0“ einfügen, wenn eine x-Potenz nicht im Polynom auftritt!

8. Zeichnen Sie nacheinander die Funktion y=cos(tan(pi*x)) im Bereich [0,1]

a) mittels einer möglichst einfachen Anweisung:

```
x=0:0.001:1; y=cos(tan(pi*x));plot(x,y)
```

b*) über eine Funktionsdatei:

```
function y=cota(x); y=cos(tan(pi*x)); %==> als cota.m abspeichern
x=0:0.001:1; y=cota(x);plot(x,y); % im Command Window eingeben
```

9. Zeichnen Sie in eine Grafik y=sin(x), y1=sin(x+.5), y2=sin(x+1) mit verschiedenen Farben und evtl. verschiedenen Stricharten im Intervall [0, 2*pi]

a) mittels einer möglichst einfachen Anweisung:

```
x=0:pi/100:2*pi; y=sin(x); y1=sin(x+.5); y2=sin(x+1);plot(x,y,x,y1,'r-',x,y2,'g*')
```

b) durch Hinzufügen zu der 1. Funktion,

```
x=0:pi/100:2*pi; y=sin(x); y1=sin(x+0.5);y2=sin(x+1);plot(x,y)
hold on;plot(x,y1,'r*'),plot(x,y2,'g--')
hold off
```

c) über den Ausdruck einer Matrix: W=[y;y1;y2];plot(x,W)

d) in einer Grafikpalette, 3 Bilder nebeneinander

```
x=0:pi/100:2*pi; y=sin(x); y1=sin(x+0.5);y2=sin(x+1);
subplot(1,3,1),plot(x,y), subplot(1,3,2),plot(x,y1,'r-'), subplot(1,3,3),plot(x,y2,'g--')
```

e) in 3 verschiedene Grafiken, die gleichzeitig existieren

```
Schließen Sie das vorhandene Bild.
x=0:pi/100:10; y=sin(x); y1=sin(x-0.5);y2=sin(x+1);
figure(1),plot(x,y); figure(2),plot(x,y1,'r-'), figure(3),plot(x,y2,'g--')
```

10. Erzeugen Sie in einem Fenster eine Funktionenschar der Exponentialfunktion y=exp(k*0.1*x) für k=1, … , 5

```
x=(0:0.01:3)';k=(0.1:0.1:0.5);Y=exp(x*k);plot(x,Y)
```

(oder x=(0:0.01:3)';F=zeros(length(x),5);
for k=1:5, F(:,k)=exp(k*0.1*x); end; plot(x,F)
oder x=(0:0.01:3)';
for k=1 : 5, y=exp(k*0.1*x); plot(x,y), hold on, end;
hold off

11. Stellen Sie fest, welche Funktionsbeschreibung die programmierten Funktionen haben:

a) t=0:pi/200:2*pi;x=sin(t);y1=sin(t+.25);y2=sin(t+0.5);plot(x,y1,'r-',x,y2,'g--')
%Lissajous-Figuren, Parameterdarstellung, keine explizite Darstellung möglich

b) t=0:pi/200:2*pi;x=3*cos(t);y1=3*sin(t);plot(x,y1,'r-')
%Kurve in Parameterdarstellung (=Kreis) ➔ axis equal

c) t=0:pi/200:2*pi;x=2*cos(t);y1=3*sin(t);plot(x,y1,'r-')
% Parameterdarstellung (=Ellipse)
Schließen Sie den Befehl "axis equal" an. Was stellen Sie fest?

f) Kurve in Polarkoordinatendarstellung, archimedische Spirale

phi=0:.1:6*pi; r=2*phi;polar(phi,r)
oder
*) syms phi r;r=2*phi; ezpolar(r,[0,2*pi]) % Arbeit mit Computeralgebra, kommt noch

g) run('R:\CB\Bernert\Infosim\Programme\fktmitFall.m')
Erschließen Sie sich aus dem Bild die Bedeutung der for- und der if-Anweisung entsprechend dem folgenden Programmtext:

```
%fktmitfall
clear
x=linspace(0,3,300);
for k=1:length(x)
    if x(k)<=1
       y(k)=x(k);
    elseif x(k)<=2
       y(k)=1-(1-x(k))^2;
    else
       y(k)=sqrt(x(k)-2);
    end
end
plot(x,y)
```

12*) Beispiel zur Animation – Federschwinger
Im Ruhezustand hat die Feder die Gestalt einer Schraubenlinie:
x = a*cos(u), y = a*sin(u), z = u, $0 \le u \le 8*pi$

Um die Oszillation mit der Amplitude A und der Kreisfrequenz ω zu simulieren, wird z in Abhängigkeit von der Zeit t variiert:

z = (1 + A*cos(ω*(t - 1))) * u
Es werden Bilder der Federposition zu den Zeiten t = 1, 2, … , 6 erzeugt und mehrfach gezeigt.
Speichern Sie die unten stehende Skriptdatei ab, geben Sie ihren Namen im Command Window ein und lassen sie sich überraschen.

```
%Skriptdatei Federoszillation
u=0:pi/60:8*pi;
a=1; A=0.2;om=2*pi/5;
M=moviein(6); % Reservierung von 6 Bildern
for t=1:6
    x=a*cos(u);y=a*sin(u);
    z=(1+A*cos(om*(t-1)))*u;
    plot3(x,y,z)
    axis([-a a -a a 0 10*pi]);
    M(:,t)=getframe; %Speicherung der Bilder
end
movie(M,15) %15-maliges Abspielen der Bilder
```

Damit haben wir alles ausprobiert und können nun zu einem Anwendungsbeispiel aus Ihrem Studiengang gehen. Dieses ist, wie immer in der Praxis, etwas komplizierter als die akademischen Beispiele. Lassen Sie sich davon aber nicht abschrecken. Durch das Einsetzen der bekannten Größen reduziert es sich auf eine quadratische Funktion $f(r)=\alpha r^2;\quad \alpha=const.$ **bzw. eine Funktion der Art**

$$f(r)=\frac{\alpha}{r^2};\quad \alpha=const.$$ **1/(x²)**

Übungsbeispiel aus der Technischen Akustik

Stellen Sie den Schalldruck p eines Punktstrahlers auf ein Probestück der Größe 10cm x 10cm in Abhängigkeit von der Frequenz und in Abhängigkeit des Abstandes r zur Schallquelle grafisch dar. Zeichnen Sie ein Bild in logarithmischer Darstellung.

Gegeben: $|p|=\left|\frac{i\omega\rho_0}{4\pi r}qe^{-ikr}\right|=\left|\frac{i\omega\rho_0}{4\pi r}q\right|\left|e^{-ikr}\right|=\frac{\omega\rho_0}{4\pi r}|q|;\qquad |q|=v\cdot s$: Schallschnelle

s : Fläche des Probestückes, s = $100\text{cm}^2=0.01\text{m}^2$
v: Schwinggeschwindigkeit, v = 1m/s

ρ_0 : mittlere Dichte der Umgebungsluft, $\rho_0=1.2041\frac{kg}{m^3}$ bei 20° C = 293,15K

ω : Kreisfrequenz, $\omega=\frac{2\pi}{T}=2\pi f,\quad Maßeinheit:\left[\frac{1}{s}\right]$

Damit ergibt sich:

1. in *r = 1m* Abstand zur Schallquelle

$$|p|=\frac{\omega\rho_0}{4\pi r}|q|=\omega\frac{1}{s}\cdot 1,2041\frac{kg}{m^3}\cdot\frac{1}{4\pi\cdot 1m}\cdot 1\frac{m}{s}\cdot 0.01m^2=2\pi f\,\frac{0,012041}{4\pi}\frac{kg}{ms^2}=6,0205\cdot 10^{-3}\,f\,Pa$$

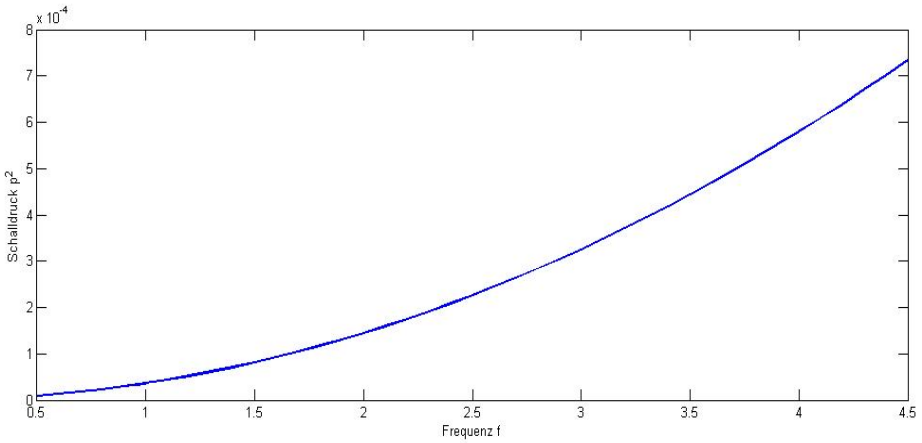
$$\text{bzw. } |p|^2=p\cdot\overline{p}=\left(\frac{\omega\rho_0}{4\pi r}|q|\right)^2=\left(6,0205\cdot 10^{-3}\,f\,Pa\right)^2=36,246\cdot 10^{-6}\,f^2\,(Pa)^2$$

Der Grund für das Plotten von p^2 liegt darin, dass p eine komplexwertige Funktion ist. Darüber wurde bisher nicht gesprochen. (In der Akustik passiert das auch nicht.) Es wird deshalb mit dem Betrag von p gearbeitet oder einfacher mit p^2. Aus dieser Funktion erhält man dann durch Wurzelziehen ja sehr schnell den Betrag, was aber meist nicht nötig ist.

Versuchen Sie nun, selbstständig die Zeichenbefehle zu finden. Die Lösung finden Sie auf der nächsten Seite

Zeichnen dieser Funktion in Matlab:

f=10:0.1:20; p2f=3.6246e-5.*f.^2; plot(f,p2f)

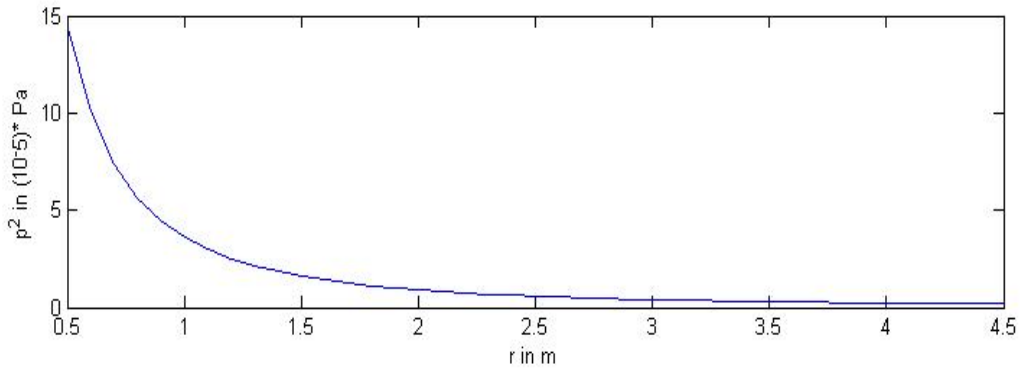


2. bei $f = 2\pi \cdot 1\frac{1}{s}$ im Abstand *r* zur Schallquelle:

$$|p| = \frac{\omega\rho_0}{4\pi r}|q| = 2\pi\frac{1}{s}\cdot 1{,}2041\frac{kg}{m^3}\cdot\frac{1}{4\pi\cdot r}\frac{1}{m}\cdot 1\frac{m}{s}\cdot 0.01m^2 = 2\pi\frac{0{,}012041}{4\pi}\frac{1}{r}\frac{kg}{ms^2} = 6{,}0205\cdot 10^{-3}\frac{1}{r}Pa$$

bzw. $|p|^2 = p\cdot\overline{p} = \left(\frac{\omega\rho_0}{4\pi r}|q|\right)^2 = \left(6{,}0205\cdot 10^{-3}\frac{1}{r}Pa\right)^2 = 36{,}246\cdot 10^{-6}\frac{1}{r^2}(Pa)^2$

Zeichnen: r=0.5:0.1:4.5; p2r=3.6246./(r.^2); plot(r,p2r)



3. Doppelt logarithmische Darstellung

Logarithmieren wir nun die letzte Gleichung:

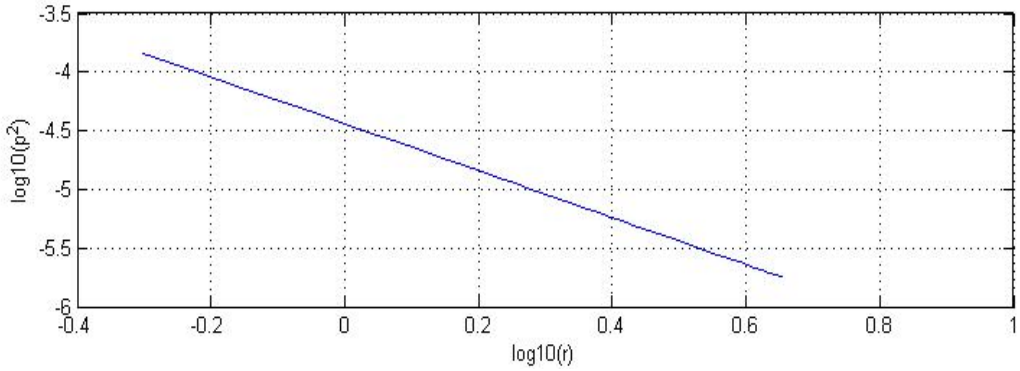
$$\log|p|^2 = \log\left(36{,}246\cdot 10^{-6}\frac{1}{r^2}\right) = \log(36{,}246) + \log(10^{-6}) + \log\frac{1}{r^2} = \log(36{,}246) - 6 - 2\log(r)$$

Wird log(r) = x als Variable betrachtet, ergibt sich eine Geradengleichung der Art

$y = \log|p|^2 = \log(36{,}246) - 6 - 2\log(r) = -4.4407 - 2x$

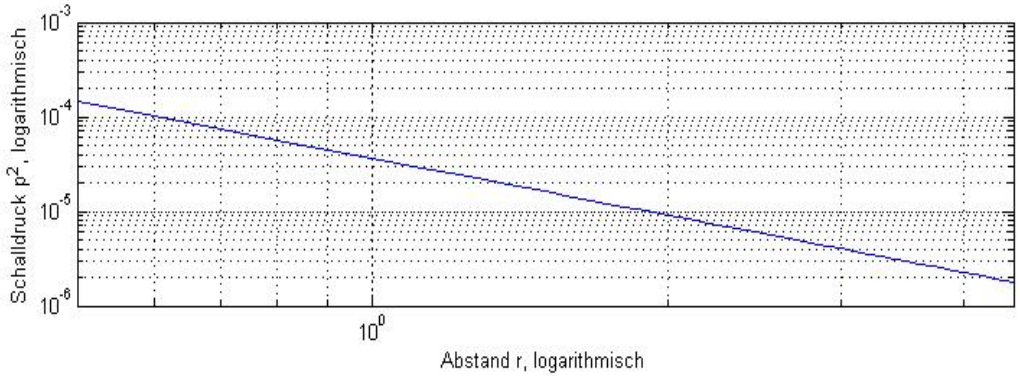
Zeichnen:

r=0.5:0.1:4.5; plog=log10(36.246)-6-2*log10(r); plot(log10(r),plog), grid



Mit der Matlabfunktion loglog wird die Darstellung automatisch in beiden Achsen logarithmisch vorgenommen:

r=0.5:0.1:4.5; p2r=36.246e-6./r.^2; loglog(r,p2r), grid



***) 3D- Grafik**

Parameterkurven:

1.Strecke: t=0:0.1:1; x=3*t+1; y=t-1; z=2*t+3; plot3(x,y,z)
- ➔ grid
- ➔ axis('ij')

2. Schraubenlinie: t=linspace(0,10*pi); plot3(sin(t),cos(t),t)

oder syms t; ezplot3(sin(t),cos(t),t)

➔ title('Helix'), xlabel('sint'),ylabel('cos(t)'), zlabel('t')

3. Betrachtungspunkt ändern (Standard view(37.7,30)):
view(30,40) % 30 Grad Seitenwinkel: negative x- Achse vom Betrachter weg,
% 40 Grad Höhenwinkel, mit dem der Betrachter auf die x-y-Ebene sieht.

4. Mehrere Kurven in ein Bild: ezplot analog zu plot verwenden

5. Raumflächen: Informieren Sie sich über die Funktion peaks

```
peaks(10) % Raster 10x10 z-Werte über der x-y-Ebene im dargestellten Bereich
peaks     % Raster 49x49 z-Werte über der x-y-Ebene im dargestellten Bereich
```

Probieren Sie die folgenden Darstellungen aus:

```
F=peaks;
subplot(3,2,1), mesh(F)
subplot(3,2,2), surf (F)
subplot(3,2,3), contour(F)   %  2-D-Höhenlinien
subplot(3,2,4), contour3(F,15) %  15 3-D-Höhenlinien
subplot(3,2,5), plot(F),  %Parallelschnitte des Standardrasters
subplot(3,2,6), pcolor(F) % Umsetzung der z-Werte in Farben
```

Bei nutzerdefinierten Funktionen muss zuerst ein Gitter in der Definitionsebene hergestellt werden, auf dem dann die Funktionswerte ausgerechnet und dargestellt werden können.

```
x=-2:0.2:2; y=x; [X,Y]=meshgrid(x,y);
Z=X.*exp(-X.^2-Y.^2);
surf(Z)
```

Übungsaufgaben:

1. Zeichnen Sie die Funktion $|p| = \left|\frac{A}{r}e^{-ikr}\right| = \frac{A}{r}\left|e^{-ikr}\right| = \frac{A}{r}$ analog zu Punkt 9 a) bis 9 e) auf verschiedene Art und Weisen, unabhängige Veränderliche: $1 \le r \le 20, \quad A = const. = 10$

2. Erstellen Sie in Analogie zum Übungsbeispiel aus der Akustik einfach und doppelt logarithmische für die obige Funktion aus Aufgabe 1.

3. Stellen Sie die Eigenformen $ev_n = \cos(\frac{n\pi x}{l}), l = 2, 1 \le n \le 4$ im Intervall [0;2] in einem Bild und in einer Bildpalette dar.

Programmierung

Elemente der Programmierung:

1. if-Anweisung:

```
if  bedingung 1
        {anweisungen 1}
   elseif bedingung 2              % wahlfrei
        {anweisungen 2}            % wahlfrei
   …..
   else                            % wahlfrei
        {anweisungen}              % wahlfrei
end
```

„bedingung 1“, „bedingung 2“ sind Fragen mit den Vergleichsoperatoren <, >, = =, <=, >=, ~ =
oder eine logische Verknüpfung solcher Fragen mit den Operatoren ||, &&, ~
(bei bitweisen logischen Operationen entfällt die Verdopplung des Operationszeichens, und es gibt noch einige Operationen mehr ➔ Hilfe-File)

Beispiel: A) x=input(' x = '); if x<0; x=-x;end; x
B) x=input(' x = '); if x<0; s=-1; elseif x==0;s=0; else s=1;end; s
% Welche Funktionen wurden hier programmiert?

2. case-Anweisung:

```
switch ausdruck
case ausdruck1
        {anweisungen1}
case ausdruck2                     % wahlfrei
        {anweisungen2}             % wahlfrei
…..
otherwise {anweisungen}            % wahlfrei
end
```

„ausdruck“ ist ein Term, der beobachtet wird,
„ausdruck1“, „ausdruck2“…. sind Zeichenketten mit dem Inhalt des zu überprüfenden Wertes des beobachteten Terms

Beispiel 1:

```
z = input('Gib eine Zahl ein! ');
switch z
    case {-1,-2}
        disp('negativ')
    case 0
        disp('null')
    case {1,2,3,4,5}
        disp('positiv')
    otherwise
        disp('nicht entscheidbar')
end
```

Beispiel 2:

```
wahrheitswert=input('Wahrheitswert als Zeichenkette eingeben!');
switch wahrheitswert;
        case 'wahr';w=1,
        case 'falsch'; w=0,
        otherwise disp('unzulässiger Wert'),
end
```

3. for-Schleife:

```
        for name=bereich
                {anweisungen}
        end
```

„name“ bezeichnet die Laufvariable
„bereich“ beschreibt deren Wertebereich ➔ Colonanweisung

Beispiel 1: Berechnung von $2^5 \cdot x$

```
        x=input('x = ');
        for i =1:5;
                x=2*x;
        end
        x
```

Beispiel 2: Zeichnung über spaltenweise Matrixbelegung

```
        x=(0:0.01:3)';F=zeros(length(x),5);
        for j=1:5,
                F(:,j)=exp(j*0.1*x);
        end
        plot(x,F)
```

oder einzelne Zeichnungen:

```
        x=(0:0.1:3)';
        for i=1 : 5,
                y=exp(i*0.1*x); plot(x,y), hold on,
        end;
        hold off
```

4. while-Schleife:

```
        while bedingung
                {anweisungen}
        end
```

Beispiel:

```
format long e; x=input('x = ');
  while x <1.0-eps;
        x=x+0.1
  end % Sukzessive Addition: Bedeutung von eps wird gezeigt!
```

Beachte: normalerweise steht hinter jeder Zeile der angegebenen Befehle ein Semikolon zur Unterdrückung des Echos des Systems. – Ausnahme: end-Befehl

Skriptfiles: Abarbeitung einer Befehlsfolge
keine Parameterübergabe
Aufruf durch name;
Variablen sind global

Beispiel 1: (s. Übung 1)
File – New – m-File

```
u=312; v=1.234; w=1.034e-05;
disp([' u = ',sprintf(' %5d ',u)])
disp([' v = ',sprintf(' %5.4f ',v)])
disp([' w = ',sprintf('\n  %5.4e ',w)])
disp(sprintf('u=%5d v=%5.4f w=%5.4e \n',u,v,w))
disp(sprintf('u=%5.4f  u=%5.4e ',u,u))
```

speichern unter einem Namen, z.B. Formate.m

Beispiel 2: (s. Übung 1)

```
%Speichern
clear; % Löschen aller Variablen
a=linspace(0,10,5); % erzeugt Vektor mit 5 Komponenten äquidistant zwischen 0 und 10
a
save; % schreibt alle Daten des Workspace auf das Standardfile matlab.mat
clear; % Löschen aller Variablen
load; % Lesen der Daten
who
a
```

Beispiel 3:

```
% (umständliche) Matrixeingabe von a,
%Achtung: a sollte noch nicht belegt sein oder clear a

% clear a

m=input(' Anzahl der Zeilen: m = ');
n=input(' Anzahl der Spalten: n = ');
disp(' ') % Leerzeile
for i =1 : m %zeilenweise Eingabe
      for j=1 : n %Elemente der Zeile
         b=num2str(i);
         c=num2str(j);
         d=[' a( ', b, ' ,' ,c,' ) ='];disp(d);
         a(i,j)=input('  ');
   end
end
```

Beispiel 3: (s. Übung 2)

```
%fktmitfall
clear
x=linspace(0,3,300);
for k=1:length(x)
    if x(k)<=1
       y(k)=x(k);
    elseif x(k)<=2
       y(k)=1-(1-x(k))^2;
    else
       y(k)=sqrt(x(k)-2);
    end
end
plot(x,y)
```

Beispiel 4: Animation (s. Übung 2)

```
%Skriptdatei Federoszillation
u=0:pi/60:8*pi;
a=1; A=0.2;om=2*pi/5;
M=moviein(6); % Reservierung von 6 Bildern
for t=1:6
    x=a*cos(u);y=a*sin(u);
    z=(1+A*cos(om*(t-1)))*u;
    plot3(x,y,z)
    axis([-a a -a a 0 10*pi]);
    M(:,t)=getframe; %Speicherung der Bilder
end
movie(M,15) %15-maliges Abspielen der Bilder
```

Beispiel 5: Erzeugen Sie eine Animation der Sinusschwingung

```
%Skriptdatei Sinusschwingungen

u=0:pi/60:2*pi;i=0;
M=moviein(20);
for t=1:6
   x=u;y=(t-1)*0.2*sin(u);
   plot(x,y)
   axis([0 2*pi -1.2 1.2]);
   M(:,t)=getframe;
end
for t=1:4
  x=u;y=(1-t*0.2)*sin(u);
  plot(x,y)
  axis([0 2*pi -1.2 1.2]);
  M(:,6+t)=getframe;
end
for t=1:6
   x=u;y=-(t-1)*0.2*sin(u);
   plot(x,y)
   axis([0 2*pi -1.2 1.2]);
   M(:,10+t)=getframe;
end
for t=1:4
  x=u;y=-(1-t*0.2)*sin(u);
  plot(x,y)
  axis([0 2*pi -1.2 1.2]);
  M(:,16+t)=getframe;
end
movie(M,10)
```

Funktionsdateien: Beginn mit Schlüsselwort „function“
Parameterübergabe in das UP: in Klammern hinter dem Programmnamen
liefert Funktionswert(e) zurück
Aufruf durch name(Parameter); oder durch Zuweisung auf eine andere Variable: name1=name(Parameter);

Kommentare: beginnen mit %
Kommentare nach function-Zeile werden bei help functionname angezeigt
Kommentare nutzen!!! (Vergesslichkeit)

Gültigkeit der Variablen:
ans und der Funktionsname sind global, d.h. in der Funktion und außerhalb verfügbar
alle anderen Variablen sind lokal, d.h. nur dort verfügbar, wo man sie benutzt
(Variablen können vor der erstmaligen Verwendung mit dem Befehl „global name“ überall dort verfügbar gemacht werden, wo man sie benötigt - widerspricht aber dem Unterprogrammkonzept!)

Übergabe von einem Funktionsnamen erfolgt in einer Zeichenkette. Im Unterprogramm, in dem diese Funktion mit ihren Argumenten aufgerufen werden soll, muss diese Zeichenkette mit dem Namen zk mit Hilfe von f=fcnchk(zk) zu einem Funktionsnamen gemacht werden. Dann kann die Funktion aufgerufen werden: a=f(argument).

Beispiel 1: (s. Übung 1)

```
function  y=cota(x);
% Beispielfunktion
y=cos(tan(pi*x));
```

Aufruf:
x=0:0.001:1; y=cota(x);plot(x,y);

Beispiel 2:

Fibonacci-Folge: (0), 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377,….

$$x_k = x_{k-1} + x_{k-2}; \quad x_0 = 1; \quad x_1 = 1$$

$$\begin{pmatrix} x_k \\ x_{k-1} \end{pmatrix} = A \begin{pmatrix} x_{k-1} \\ x_{k-2} \end{pmatrix} \quad \text{mit} \quad A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$$

Laden Sie die m-Dateien Fibo.m, Fibofolge.m, fibofkt.m, FibomitFkt.m und Fibogesmitfkt.m, Fibogesmitfktuebergabe auf Ihren Arbeitsbereich und erforschen Sie deren Arbeitsweise durch Anzeige im Editor und den entsprechenden Funktionsaufruf.

```
Fibo
Fibofolge
fibofkt(10,1,1)
x=fibofkt(10,1,1)
FibomitFkt
Fibogesmitfkt
a=Fibogesmitfktuebergabe('fibofkt')
```

Aufgaben:

1. Schreiben Sie ein Programm zur numerisch stabilen Berechnung der Nullstellen der quadratischen Gleichung $x^2 + px + q=0$. Die Parameter p und q sollen dabei im Dialog eingegeben werden. Testen Sie das Programm anhand von Beispielen.
2. Schreiben Sie sowohl eine Skriptdatei als auch ein Unterprogramm zur Lösung der Aufgaben 1 aus der Übung "Fehler".
3. Schreiben Sie ein Unterprogramm zur Lösung der Aufgabe 3 aus der Übung "Fehler".
4. Entnehmen Sie aus einem anderen Modul ein kleines rechnerisches Problem und stellen Sie dafür ein Unterprogramm oder eine Skriptdatei auf.

CAS in Matlab

Matlab startete als reines Numerik-Programm-System, hat aber schon seit vielen Jahren auch einen Computeralgebrabaustein, die symbolische Toolbox, die mit zur Grundausstattung des Systems zählt.
Vom Taschenrechner her ist die Funktionalität und teilweise auch die Bedienung bereits bekannt. Allerdings hat Matlab gegenüber dem TR eine mehr Rechenpower im Hintergrund sowie einen größeren Bildschirm zur Verfügung und kann auch Formeln die mehrere A4-Seiten umfassen, bearbeiten und ausgeben.
Im Gegensatz zum numerischen Rechnen bietet die Symbolic Math Toolbox die Arbeit mit Variablen analog zum bekannten Rechnen mit Papier und Stift, im Wesentlichen ohne Zahlen, d.h. analytisch. Dabei ist aber zu beachten, dass manche mathematischen Probleme analytisch nicht lösbar sind. Insofern ist die Kombination aus Numerik und CAS günstig, was auch andere Hersteller solcher Software festgestellt und realisiert haben. Z.B. enthält das ursprünglich als CAS-System konzipierte Programmsystem Mathematica auch numerische Berechnungsmöglichkeiten, ist aber nicht modular aufgebaut und damit sehr teuer.

Symbolische Toolbox:

- angekoppelter Baustein an Matlab, entspricht dem Computeralgebrasystems MuPAD,
  (Matlab bietet auch direkten Zugriff darauf in Form des MuPAD-Notebook-Interface mit dem Befehl: mupadwelcome, Hilfe über Hilfe zur Symbolic Math Toolbox)*

- enthält > 100 Funktionen, die die Matlab- Sprache erweitern z.B. für
  - Differentiation, Integration, Grenzwertbildung
  - Summen, Reihen
  - Vereinfachung von Reihen
  - Symbolische und Numerische Lösung von algebraischen und Differentialgleichungen
  - Spezielle Funktionen
  - Matrizeninversion, Determinante, Eigenwerte, Singulärwertzerlegung und kanonische Form von symbolischen Matrizen
  - Fouriertransformation, Laplacetransformation, Z-Transformation und ihre Inversen

Für die Arbeit mit symbolischen Variablen/Zahlen ist die **Deklaration symbolischer Variabler notwendig**:

```
syms a b c; % Achtung nur Leerzeichen zwischen den Variablen!
```

Umwandlung von Zahlen/Zeichenketten/numerischen Variablen in symbolische und zurück:

```
sym(2), x=sym('x'), b=sym(2)/sym(5), n=1.2, sym(n)

sym(0.1,'e')      %liefert Wert + mögl. Fehler durch Maschinendarstellung
sym(0.1,'d')      %liefert alle signifikanten Stellen
digits(7); sym(0.1,'d') %liefert 7 Stellen
sym(0.1,'r')      %liefert rationale Zahldarstellung
sym(0.1,'f')      %liefert Quotientendarstellung
double(ans)       %Wandlung in eine Gleitpunktdarstellung
```

Verwendung von symbolischen Größen in numerischen Rechnungen führt zum Verlust der symbolischen Eigenschaft: b=1/n
Mit einem Blick in den Workspace können Sie sofort sehen, wie der Status Ihrer Variable ist.

Löschen von symbolischen Variablen:

- Löschen der symbolischen Eigenschaft bei einer Variablen ➔ numerische Neuverwendung oder clear x
- syms x erneuert symbolische Eigenschaft

Übungsaufgaben Komplex 1:
Bitte versuchen sie es zuerst selbst. Wenn es nicht funktioniert, dann finden Sie die Lösungen am Ende des Praktikums.

- Berechnen Sie $\frac{1}{3}+\frac{2}{5}$ als Bruch und anschließend als Dezimalzahl.

- Erstellen Sie die Hilbertmatrix der Dimension (3,3) und wandeln Sie diese in eine symbolische Matrix um. Wo liegt der Unterschied? Berechnen Sie dazu jeweils die Eigenwerte und vergleichen Sie die dazugehörigen double-Zahlen.

- Mit symbolischen Matrizen können alle Matrizenoperationen symbolisch ausgeführt werden. Bilden Sie von der symbolischen Hilbertmatrix die Summe der 1. Zeile in vereinfachter Form und die Determinante.

- Erstellen Sie die Matrix $B = \begin{bmatrix} a & b & c \\ b & c & a \\ c & a & b \end{bmatrix}$. Wiederholen Sie alle Operationen, die Sie mit der Hilbertmatrix durchführten (EW, det, Summe 1. Spalte zur Abwechslung) und das, was Ihnen Spaß macht.

- Erstellen Sie eine komplexen Variable: syms a b real, z=a+b*i,
und berechnen Sie die konjugiert komplexe Zahl und den Betrag: conj(z), z*conj(z), expand(ans)

- Kreieren Sie eine symbolische Variable namens rho mit dem Inhalt $\frac{1+\sqrt{5}}{2}$, bilden Sie den symbolischen Term f=rho^2 - rho - 1 und vereinfachen Sie ihn mit simplify(f)

Arbeiten mit symbolischen Variablen

Symbolische Terme können auch **vordefinierte Funktionen enthalten**, z.B.:
syms x y z, r=sqrt(x^2+y^2+z^2), t=atan(y/x), f1=sin(x*y)/(x*y)

Vereinfachen komplizierter Terme durch simplify(term) oder expand(term)

Ersetzen von symbolischen Variablen durch Werte: subs(f,1) ersetzt die x am nächsten liegende Variable

Bsp.: syms a b c x; f=a*x^2+b*x+c
subs(f,1) ➔ ans = a+b+c
g1= subs(f,c,1); g2=subs(g1,b,3);g3=subs(g2,a,-2), subs(g3,100) (=g3(100))

Übung 2: Substituieren Sie in der Matrix B das Element c durch x.

Suchen von symbolischen Variablen in symbolischen Termen: symvar(f)

Darstellen von symbolischen Funktionen

Informieren Sie sich über die Befehle fplot und ezplot in der Dokumentation

Bsp.: fplot(@humps,[0 2])

ezplot(g3)

syms x y;y=sqrt(1-x^2);ezplot(y,[-1,1]) %symbolisch definierte Kurve (Halbkreis)
➔ axis equal

syms x y z; z=x^2/4+y^2/9-1;ezplot(z,[-2,2,-3,3]) % symbolisch definierte Kurve (Ellipse)
Schließen Sie den Befehl "axis equal" an. Was stellen Sie fest?

Zeichnen der Funktion y=cos(tan(pi*x)) im Bereich [0,1] mittels Erfassung der Funktion als Zeichenkette und numerischer Auswertung des Funktionsausdruckes:
y='cos(tan(pi*x))';x=0:0.001:1;z=eval(y);plot(x,z)

oder durch Nutzung der internen Funktionsdarstellung : fplot(@(x)cos(tan(pi.*x)) ,[0 1])

Symbolische Unterprogramme für Funktionen:

```
function z=sinc1(x)
%Die symbolische sinc-Funktion braucht eine symbolische Eingangsvariable
if isequal (x,sym(0))
   z=1;
else
   z=sin(x)/x;
end
```

Abspeichern unter dem Namen „sinc1“
Aufruf im Kommandofenster: z=sinc1(x) (wobei x eine symbolische Variable sein muss: clear x, syms x);
Anschließend: ezplot(z)

Differentiation:

```
h=sin(x^2), k=exp(x)*(cos(x))^2
h1=diff(h),k1=diff(k)
simplify(h1)
simplify(k1)
h2=diff(h1),h22=diff(diff(h))
diff('5'), diff(sym(5))
syms n,diff(x^n),
simplify(ans)
syms r l f, x=r*cos(l)*cos(f), diff(x,l),diff(x,r),diff(diff(x,f),r)
```

Matrizen werden elementweise differenziert:

```
syms A a x, A=[cos(a*x) sin(a*x);exp(x^3) log(3*x)], diff(A)
```

Integration:

Voraussetzung: syms var

int(f)	
int(f,var)	Integration der Funktion f bezüglich der Variable var
int(f,var,a,b)	Integration der Funktion f bezüglich der Variable var in den Grenzen von a bis b

Bei Nichtberechenbarkeit des Integrals kommt als Matlab-Antwort die Eingabe zurück oder ein Term mit der Fehlerfunktion erf oder anderen nicht berechenbaren Termen.

```
syms x
int(sin(x))      ➔      ans = - cos(x)
erg=int (sin(x),0,pi)    ➔    erg = 2, symbolische Größe

syms a b t
 g=cos(a*t+b),  int(g,t)    ➔    ans=sin(a*t+b) /a

int(1./(x.^3-2*x-5))    ➔    ans=…, kein expliziter symbolischer Term, der verwendbar wäre
int(1./(x.^3-2*x-5),x,10,20)    ➔    ans= Echo der Aufgabe, nicht berechnet
```

double(ans) ➔ Bei bestimmten Integralen kann double die Berechenbarkeit bringen, aber symbolische Eigenschaft geht verloren!

Mehrfache Integrale werden geschachtelt:

```
syms x y z a;

int(int(sin(x+y),x,0,y),y,0,1)    ➔    ans = sin(1) - sin(2)/2

int (int(int( x^2+y^2,x,0,a),y,0,a),z,0,a)    ➔    ans = (2*a^5)/3
```

Informieren Sie sich in der Funktionsliste über mögliche weitere Operationen mit symbolischen Termen und testen Sie einige Befehle aus:
Home ➔ Help ➔ Documentation➔ Symbolic Math Toolbox ➔ oben quer: Functions ➔ nach unten scrollen ➔ Funktionen sind nach Einsatzgebieten gelistet und beschrieben

Übungsbeispiel: Kurvendiskussion
Arbeiten Sie diese Beispiel aufmerksam durch. In der letzten Zusatzaufgabe des Belegs gewinnt es an Bedeutung.

Im 1. Semester führten Sie Kurvendiskussionen mit folgendem Inhalt aus:

1. Definitionsgebiet
2. Symmetrie, Periodizität
3. Verhalten an den Randpunkten des Definitionsgebietes
4. Achsenschnittpunkte
5. Unstetigkeitsstellen
6. Extremwerte
7. Wendepunkte
8. Skizze
9. Wertevorrat

Bis auf die Angabe des Definitionsgebietes, der Periodizität, der Klassifikation der Unstetigkeitsstellen, und des Wertebereiches, die von Hand gestimmt werden müssen, können Sie nun mittels Matlab eine Kurvendiskussion ausführen, z. B. für die Funktion

$$f(x) = \frac{3x^2 + 6x - 1}{x^2 + x - 3}$$

in Matlab: syms x, f=(3*x^2+6*x-1)/(x^2+x-3), aber ich nutze Zähler und Nenner einzeln für einige Teilaufgaben und definiere die Funktion f deshalb über Zähler/Nenner (s. 2.)

1. Definitionsgebiet:
   Suche Nennernullstellen: syms x, zae=3*x^2+6*x-1,nen=x^2+x-3
   nst=solve(nen) liefert die Nennernullstellen ➔D_f=**R**\{ 1.3028, -2.3028} , double nutzen!

2. Symmetrie, Periodizität
   f=zae/nen, subs(f,-x), simplify(ans), simplify(f)
   Der Vergleich der Terme ergibt keine Symmetrie für f(x). Es ist keine periodische Funktion.
   Wir zeichnen nun zuerst die Funktion, um die problematischen Punkte zu sehen: ezplot(f), grid

3. Verhalten an den Randpunkten des Definitionsgebietes
   limit(f,inf), limit(f,-inf) liefert die horizontale Asymptote y = 3 der Funktion

4. Achsenschnittpunkte
   subs(f,0) liefert den Funktionswert an der Stelle x=0 ➔ SP mit y-Achse
   nstx=solve(zae) liefert die Zaehlernullstellen ➔ SP mit der x-Achse

5. Unstetigkeitsstellen
   nst=solve(nen) liefert die Nennernullstellen (s.o.) ➔ senkrechte Asymptoten oder Stellen einer Lücke, wenn dieser Wert auch als Nullstelle des Zählers auftritt.
   Einzeichnen aller dieser Asymptoten in den Graph mittels der Skriptdatei asymtoten.m auf r:\CB\Bernert\InfoSim\Programme:

```
%zeichne die horizontale und die vertikalen Asymptoten
ezplot(f)
grid
hold on %Graf soll erhalten bleiben
plot([-2*pi 2*pi],[3 3], 'g') % horizontale A.
plot(double(nst(1))*[1 1],[-5 10],'r') %vertikale A. Nr. 1
plot(double(nst(2))*[1 1],[-5 10],'r') %vertikale A. Nr. 2
title('Horizontale und vertikale Asymptoten')
```

Danach werden die einseitigen Grenzwerte an den Stellen der Asymptoten bestimmt, um numerisch zu entscheiden, ob es Polstellen gerader oder ungerader Ordnung sind:

limit(f,x,nst(1),'left'), limit(f,x,nst(1),'right')
limit(f,x,nst(2),'left'), limit(f,x,nst(2),'right')

liefern die links- bzw. rechtsseitigen GW an den Nennernullstellen. Damit liegen dort, wie schon aus der Grafik ersichtlich, einfache Polstellen vor.

6. Extremwerte
   Zur Bestimmung der relativen Extrema benötigen wir die 1. Ableitung und deren Nullstellen:
   f1=diff(f), simplify(f1)
   kritpkt=solve(f1)
   Aus dem Bild ist ersichtlich, dass beim kleineren x-Wert das relative Minimum und beim größeren das relative Maximum vorliegen muss. Bewiesen wird es über den Wert der 2. Ableitung an diesen Stellen:

   f2=diff(f1), simplify(f2)
   w1=double(subs(f2,kritpkt(1))), w2=double(subs(f2,kritpkt(2)))

   Das folgende Skriptfile (BildExtrema.m) trägt die Extrema in das Bild ein:

```
%Einzeichnen der Extrema
hold on
plot(double(kritpkt),double(subs(f,kritpkt)),'ro')
title('Maximum und Minimum von f')
text(-5.5,3.2,'Lokales Minimum')
text(-2.5,2,'lokales Maximum')
```

7. Wendepunkte
   Analoges Vorgehen liefert die Wendepunkte. Führen Sie diese Rechnung selbstständig aus. Es gibt nur einen reellen Wendepunkt, komplexe Werte werden nicht betrachtet.
8. Die Skizze wurde parallel zur Rechnung erstellt.
9. Wertebereich: $W_f = R \setminus (\min, \max)$

Übungsaufgabe:
Führen Sie analog eine vollständige Kurvendiskussion oder Teile davon durch für eine für Sie interessante Funktion oder für f(x)=((4x-2)/(x²+4x)) oder f(x)=1/(5+4cosx)).

Lösungen zu den Aufgaben im Text :

```
% Lösungen zu Komplex 1:
z=sym(1/3) +sym(2/5)
z1= double(z)

H=hilb(3),Hs=sym(H),E=eig(H),Es=eig(Hs), double(Es)
%Unterschied: Die numerisch bestimmten Eigenwerte sind genauer, da eine
%symmetrische Matrix wie die Hilbertmatrix reelle Eigenwerte hat. In der
%Bruchdarstellung der Eigenwerte der symbolischen Matrix sind aber
%Imaginärteile enthalten.... - Also seien Sie vorsichtig mit der
%symbolischen Rechnung!

sum(Hs(1,:)),det(Hs)

syms B a b c
B=[a b c;b c a; c a b]
eig(B), D=det(B),....

syms rho
rho= sym(1+sym(sqrt(5)))/sym(2)
f=rho^2 - rho - 1
simplify(f)
```