Rechnerpraktikum Teil 2

3 Einführung in MATLAB

Matthias Knauer
Bremen, 18. April 2007

Achtung!
Diese knappe Zusammenstellung von MATLAB-Befehlen ist nicht als Lehrbuch oder Referenz für MATLAB gedacht. Stattdessen werden werden in Beispielen kompakt Lösungswege für unterschiedliche Probleme dargestellt. Für Hinweise und Korrekturen bin ich dankbar.

Matthias Knauer (`knauer@math.uni-bremen.de`)

Inhaltsverzeichnis

1 Übersicht MATLAB

- MATLAB ist eine kommerzielle mathematische Software der Firma The MathWorks, Inc. zur Lösung diverser mathematischer Probleme und zur grafischen Darstellung der Ergebnisse.
- MATLAB ist für Berechnungen mit Matrizen ausgelegt, woher sich auch der Name ableitet: MATrix LABoratory.
- Programmiert wird MATLAB in einer proprietären, plattformunabhängigen Programmiersprache, die auf der jeweiligen Maschine (Computer) interpretiert wird.
- Kleinere Programme können als so genannte Scripts oder Funktionen zu atomaren Einheiten verpackt werden (M-Files).
- Weitere anwendungsorientierte Werkzeugkisten (Toolboxes) sind auch kommerziell erhältlich.
- Es gibt Schnittstellen, um C-Code einzubinden.
- MATLAB dient im Gegensatz zu Computeralgebrasystemen (z.B. Maple und Mathematica) nicht der symbolischen, sondern primär der numerischen (zahlenmäßigen) Lösung von Problemen.
- Die Software wird in Industrie und an Hochschulen vor allem für numerische Simulation eingesetzt.

2 Eingabe

- Datentypen in Matlab: komplexwertigen Matrizen und Zeichenketten.
- Skalare sind also Matrizen der Dimension 1x1.
- i und j als komplexe Einheit vordefiniert.
- pi ist π.
- eps ist Maschinengenauigkeit.
- Groß- Kleinschreibung wird unterschieden.
- Pfeiltasten für letzte Befehle (mit Starttext!)
- Prozentzeichen für Kommentar
- Ausgabe Standardmäßig in Variable ans (wennn nichts zugewiesen wurde)

- = Zuweisung von Ergebnis an Variable
- ; Unterdrückt Ausgabe
- , mehrere Kommandos in einer Zeile
- ctrl+c Abbruch von Endlosschleife
- Keine Deklaration der Variablen, Matrizen wachsen automatisch mit.
- Variable Anzahl der Ein- und Ausgabeparameter

2.1 Skalare, Vektoren und Matrizen

`n = 7`	$n = 7$	Skalar
`x = [1 2 3 4 5 6]`	$x = (1\ 2\ 3\ 4\ 5\ 6)$	Zeilenvektor
`x = [1, 2, 3, 4, 5, 6]`	$x = (1\ 2\ 3\ 4\ 5\ 6)$	Zeilenvektor
`x = 1:6`	$x = (1\ 2\ 3\ 4\ 5\ 6)$	Zeilenvektor von 1 bis 6, Schrittweite 1
`x = 1:.5:3`	$x = (1\ 1.5\ 2\ 2.5\ 3)$	Zeilenvektor von 1 bis 3, Schrittweite 0.5
`x = 2:-.2:1`	$x = (2\ 1.8\ 1.6\ 1.4\ 1.2\ 1)$	Zeilenvektor mit Schrittweite -.2
`x = linspace(3,4,11)`	$x = (3\ 3.1\ 3.2\ \ldots 3.8\ 4)$	Zeilenvektor mit 11 äquidistanten Einträgen von 3 bis 4.
`x = [1;2;3]`	$x = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$	Spaltenvektor
`x = [1 ret 2 ret 3]`	$x = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$	Spaltenvektor
`x = [1 2 3;4 5 6]`	$x = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$	Matrix

2.2 Zugriff auf Elemente

`z = x(1,2)`	$z = 2$	Element auslesen
`x(1,1) = 9`	$x = \begin{pmatrix} 9 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$	Element setzen
`x(2,:)`	$ans = \begin{pmatrix} 4 & 5 & 6 \end{pmatrix}$	2. Zeile
`x(:,3)`	$ans = \begin{pmatrix} 3 \\ 6 \end{pmatrix}$	3. Spalte
`x(1,1:2)`	$ans = \begin{pmatrix} 9 & 2 \end{pmatrix}$	Spalten 1–2 von 1. Zeile

2.3 Matrixfunktionen

`y = x'`	$y = \begin{pmatrix} 9 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$	Transposition
`N = zeros(3,2)`	$N = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$	3x2 Nullmatrix
`M = ones(3,2)`	$M = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix}$	3x2 Einsmatrix
`I = eye(2)`	$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	2x2 Einheitsmatrix
`D = diag([1 2])`	$I = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$	Vektor in Diagonalmatrix
`D = diag([1;2])`	$I = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$	Vektor in Diagonalmatrix
`R = rand(3)`	$R = \begin{pmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{pmatrix}$	3x3 Zufallsmatrix
`R = randn(3)`	$R = \begin{pmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{pmatrix}$	3x3 Zufallsmatrix, normalverteilt
`NM = [N M]`	$NM = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$	Zusammensetzen von Blockmatrizen
`R = magic(3)`	$R = \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix}$	Magisches Quadrat

3 Hilfe

```
help help
help
help sin
doc sin
demo
lookfor eigenvalue
```

Anzeige der Befehle, die das Wort *eigenvalue* in Befehlserklärung haben

4 Variablen

`who`	Anzeigen aller Variablen
`whos`	ausführliches Anzeigen aller Variablen
`clear M`	Variable M löschen
`clear`	Alle Variablen löschen
`format long/short`	Ausgabegenauigkeit setzen

5 Operationen

`x+y`	Elementweise Addition, wenn Dimensionen übereinstimmen
`x*y`	Matrixmultiplikation
`x.'`	nur Transposition
`x'`	Transposition und komplexe Konjugation = Adjungierte
`x.*y`	Elementweise Multiplikation
`3+x`	Addition zu jedem Matrixelement
`x^2`	Potenz der Matrix
`x^-1`	Inverses der Matrix
`x.^2`	Punktweise Potenz
`x=A\b`	Lösen von Glsys A*x=b
`x=b/c`	Lösen von Glsys x*c=b
`x=b./c`	Elementweise Division
`max(x)`	Größtes Element
`size(x)`	Dimension
`[x,y] = size(A)`	Dimension
`length(x)`	Max. Dimension

Beispiel:

$$A = \begin{pmatrix} 1 & 1 & -3 \\ 2 & 0 & 2 \\ -1 & 1 & 3 \end{pmatrix}, \quad b = \begin{pmatrix} 2 \\ -6 \\ 0 \end{pmatrix}$$

$$c = A\backslash b = \begin{pmatrix} -2 \\ 1 \\ -1 \end{pmatrix}$$

$$b/c = \begin{pmatrix} -1 & 0 & 0 \\ 3 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

6 Datenanalyse

`sum(A)`	Summe der Zeilenspalten
`prod(A)`	Produkt der Zeilenspalten
`mean(A)`	Mittelwert
`std(A)`	Standardabweichung
`sort(A)`	Sortieren der Spalten

7 Lineare Algebra

`[V,D] = eig(A)`	Eigenwerte D und Eigenvektoren V
`det(A)`	Determinante
`inv(A)`	Inverse
`poly(A)`	Koeffizienten des char. Polynoms

8 Dateien

Textdatei erstellen mit Matrix als Inhalt:

8.1 Matrizen laden und speichern

`matrix.dat`: Daten

```
1 2 3
3 4 5
4 5 6
```

`load('matrix.dat')`	Variable `matrix` hat jetzt diesen Wert.
`D=load('matrix.dat')`	`D` hat diesen Wert
`save test.dat D -ASCII`	Variable `D` lesbar abspeichern
`save sitzung.mat`	Komplette Sitzung abspeichern
`load sitzung.mat`	Sitzung laden

8.2 Skriptdateien

MATLAB-Editor öffnen mit `edit` oder über das Menu mit *File →M-File*

`programm.m`: Skriptdatei

```
A = [1 2 3; 4 5 6; 7 8 9];
d = det(A)
```

Aufruf dieses MATLAB-Skripts mit `programm`

`pwd`	aktuelles Arbeitsverzeichnis anzeigen
`cd /home/knauer/matlab`	in anderes Verzeichnis wechseln

- `load('xxx')` und `load xxx` sind äquivalent. Bei Zeichenketten als Parameter kann Klammer und ” weggelassen werden!
- Zeichenketten werden wie Vektoren behandelt.

8.3 Funktionsdateien

`quadratflaeche.m`: Funktionsdatei

```
function I = quadratflaeche(q)
% QUADRATFLAECHE. Flaeche eines Quadrats.
%    QUADRATFLAECHE(Q) ist die Flaeche eines
%    Quadrats mit Seitenlänge Q.

I = q*q;

% Ende der Funktion quadratflaeche
```

- Name der Datei = Name der Funktion + `.m`
- Aufruf mit `flaeche = quadratflaeche(2.0);`
- Aufruf der Hilfe mit `help quadratflaeche`
- Hilfe soll Semantik und Syntax klarmachen
- auch mehrere Rückgabewerte möglich: `function [I,J] = name(...)`
- `nargin` speichert Anzahl der bereitgestellten Eingabewerte
- Beispiele: `/home/matlab/7.2/toolbox/matlab/elmat`

9 Visualisierung

```
x = 0:0.1:2*pi
y = sin(x)        % MATLAB kennt alle trig. Funktionen
plot(x,y)
```

9.1 2d-Plots

`plot(x1,y1, x2, y2, ...)`	Mehrere Plots
`plot(x,Y)`	Matrix Y wird zeilenweise gegen x geplottet
`plot([0 1 4],[5 3 7])`	Plot durch Punkte (0,5), (1,3), (4,7)
`plot([0 1 4;5 3 7])`	drei Plots durch je zwei Punkte

Zugriff auf Figures

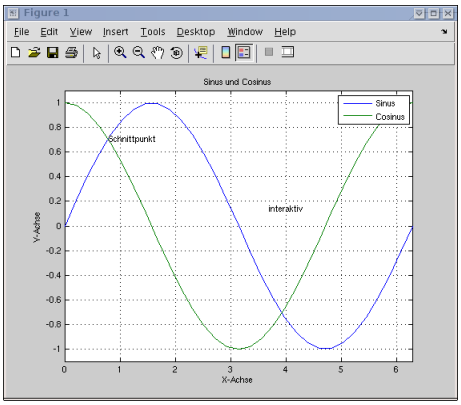
`hold on`	hold auf on oder off stellen, mehrere Bilder in einer Figur
`close`	Aktuelles schliessen
`close all`	Alles schliessen
`figure(3)`	Neuanlegen oder Auswahl von bestehender Figure 3

Änderung der Linienstile

`plot(x,2*y,':r');`	rote gepunktete Linie verwenden
`doc LineSpec`	Komplette Dokumentation

9.2 Beschriftung

```
x=linspace(0,2*pi,31);
plot(x,[sin(x);cos(x)]);
xlabel('X-Achse')
ylabel('Y-Achse')
legend('Sinus','Cosinus')
axis([0,2*pi,-1.1,1.1]);
title('Sinus und Cosinus')
text(pi/4,sqrt(1/2),
    'Schnittpunkt')
grid on
gtext('interaktiv')
```



9.3 Spezielle Plots

`subplot(2,2,3)`	Auswahl des 3. Unterfensters einer 2x2-Figure
`loglog`	Skalen beider Achsen logarithmisch
`semilogx`	Skala der x-Achse logarithmisch
`semilogy`	Skala der y-Achse logarithmisch

9.4 3d-Plots

```
z = peaks(25);
surf(z);
colormap(hsv)
```

oder

```
L = linspace(0,3*pi,31)
[X,Y] = meshgrid(L,L)
[X,Y] = meshgrid(L)
Z = sin(X)+cos(Y)+ X/3+Y/3;
mesh(X,Y,Z) % Gitter
```

andere Darstellungen

`mesh(Z)`	Gitter, aber mit Standardachsen
`contour(Z)`	Konturlinien
`contourf(Z)`	gefüllte Konturlinien
`contour3(z)`	Konturlinien, dreidimensional
`imagesc(Z)`	Farbwerte
`surf(Z)`	Oberfläche
`surface(Z)`	Ansicht von oben
`plot3(X,Y,Z)`	Kurve im $\mathbb{R}^3$

Anzeigeoptionen

`shading interp|flat|faceted`	Glättung der Oberfläche
`colormap hsv|flag|gray|jet|prism|winter`	Farbtabelle
`doc colormap`	Übersicht über Farbtabellen

Erstellen einer eigenen colormap

```
colormap([0 0 1;1 0 0])  % nur rot und blau
x=linspace(0,1,20);
r=linspace(0,1,20);
g=linspace(0,0,20);
```

```
b=linspace(1,0,20);
colormap([r;g;b]');       % interpoliert rot->blau
```

10 Bedingungen

```
if b ~= 0
   c = a/b;
else
   c = 0;
end
```

auch `elseif` für geschachtelte `if`-Anweisung

Vergleichsoperatoren

`~=`	ungleich
`==`	gleich
`<, >,<=, >=`	wie gewohnt.

Ergebnis eines Vergleichs: 0 oder 1

logische Verknüpfungen

`a & b`	a und b
`a | b`	a oder b
`and(a,b)`	a und b
`or(a,b)`	a oder b
`xor(a,b)`	genau a oder b
`~a`	nicht a

11 Schleifen

Berechnung der Fakultät:

```
x=1;
for k=1:11
   x = x*k;
end
x
```

```
x=1; k=11;
while k>1
    x = x*k;
    k = k-1;
end
x
```

- Manueller Abbruch der Schleife mit CTRL+C.
- Programminterner Abbruch mit Schlüsselwort break

12 Drucken

`print`	Standarddrucker
`printopt`	Genaue Anzeige des Befehls
`print -deps myplot.eps`	Ausgabe in eps-Datei
`print -depsc colorplot.eps`	farbige Ausgabe
`print -djpeg80 myplot.jpg`	Ausgabe als jpg
`orient portrait | landscape`	Einstellung der Papierausrichtung
`print('-djpeg80'text);`	Alternativer Aufruf

13 inline-Funktionen

```
g = inline('t^2')              % g(t) = t^2
g(2)
g = inline('x^2*y')            % g(x,y) = x^2*y
g(2,3)
g = inline('x^2*y','y','x')    % g(y,x) = x^2*y
g(2,3)
```

14 Funktionen auswerten

Übergabe von Funktionen als Parameter

myplot.m: Parameter für Funktion

```
function y = myplot(fun,a,b)

x = linspace(a,b,101);
y = feval(fun,x);
plot(x,y)
```

Aufruf der Funktion mit:

```
myplot(@sin,0,2*pi)
myplot(@gg,0,2*pi)
```

Dabei kann auch eine eigene Funktion übergeben werden:

gg.m: Eigene Funktion

```
function A = gg(x)

A = x.*sin(x)
```

15 Formatierte Ausgabe

Formatierungsanweisungen wie in C

Schreiben in Datei

```
file = fopen('demo.txt','w')
for i=1:10
    fprintf(file,'Kehrwert␣von␣%d␣ist␣%6.3f\n',i,1/i);
end
fclose(file)
```

Schreiben in Zeichenkette

```
j=21;
text = sprintf('file%03d.jpg',j);    % "file021.jpg"
```

16 Differenzieren und Integrieren

Differenzen berechnen

```
x=linspace(0,2*pi,51);
y=sin(x);
z=diff(y)/(x(2)-x(1)); % Vorwärtsdifferenzenquotient
z(51)=z(50);
plot(x,[y;z]);
```

numerische Differentiation

```
w = gradient(y,x);
plot(x,[y;z;w]);
```

numerische Integration von Messdaten

```
for i=2:51
    v(i) = trapz(x(1:i),w(1:i));
end
plot(x,[y;z;w;v]);
```

oder kürzer

```
v2 = cumtrapz(x,w);
plot(x,[v2;v]);
```

numerische Integration von Funktionen

`quad(@sin,0,pi)`	$\int_0^{\pi} \sin x \, dx$
`quad(@(x)sin(x),0,pi)`	Integration von $x \mapsto sin(x)$
`quad1(@(x)sin(x),0,pi)`	genauere Berechnung
`dblquad(@(x,y)x.^2*y,0,1,0,2)`	$\int_0^2 \int_0^1 x^2 y \, dx \, dy$
`triplequad`	$\int \int \int$

17 Mathematisches Pendel

$$\ddot{\varphi} = -\frac{\sin\varphi}{2}$$

oder als DGL-System:

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -\frac{\sin x_1}{2}$$

mit Anfangswerten $x(0) = \begin{pmatrix} 2.9 \\ 0 \end{pmatrix}$

17.1 DGL-Systeme integrieren

`pendel.m`: DGL für das Pendel

```
function out = pendel(t,x)
% pendel
out = [x(2); -0.5*sin(x(1))];
```

Aufruf mit...

```
tspan = linspace(0,20,101);
[T,Y] = ode45(@pendel,tspan,[2.9,0],1.e-5);
plot(T,Y)              % Plot von x_1,x_2 gegen die Zeit
pause
plot(Y(:,1),Y(:,2))    % Phasenportrait
```

17.2 Animation

```
tspan = linspace(0,20,101);
[T,Y] = ode45(@pendel,tspan,[2.9,0],1.e-5);

for j=1:101  % in jedem Bild ...
        % ... das Pendel durch eine Gerade andeuten
        plot([0,sin(Y(j,1))],[0,-cos(Y(j,1))])
        axis([-1,1,-1,1])
        F(j) = getframe;
end
movie(F,20)
```

Animation als AVI-Datei

```
tspan = linspace(0,20,101);
[T,Y] = ode45(@pendel,tspan,[2.9,0],1.e-5);

% AVI-File initialisieren
fig=figure;
set(fig,'DoubleBuffer','on');
mov = avifile('example.avi')

for j=1:101
    plot([0,sin(Y(j,1))],[0,-cos(Y(j,1))])
    axis([-1,1,-1,1])
    % Bild hinzufuegen
    mov = addframe(mov,getframe);
end

% AVI-File schliessen
mov = close(mov);
```

17.3 Vektorfelder

In jedem Punkt des Gitters gibt ein Vektorpfeil an, in welche Richtung sich das System von diesem Punkt aus weiterentwickelt.

```
lx = linspace(-4,4,21);
ly = linspace(-2,2,21);
[X,Y] = meshgrid(lx,ly);
[ax,ay]=size(X);
U=zeros(ax,ay);
V=zeros(ax,ay);
for i=1:ax
    for j=1:ay
        hilf = pendel(0,[X(i,j);Y(i,j)]);
        U(i,j) = hilf(1);
        V(i,j) = hilf(2);
    end
end
quiver(X,Y,U,V);
```

Vergleich mit Bahnkurve

```
pause;hold on
tspan = linspace(0,20,101);
[T,Y] = ode45(@pendel,tspan,[2.9,0],1.e-5);
plot(Y(:,1),Y(:,2),'--r')
```

18 Lineare Gleichungssysteme

```
A = eye(3)+1
b = [1;2;-1]
x = A\b              % Löse A * x = b
                     % Lösung ist eindeutig, da A regulär
x2 = inv(A)*b        % x = A^-1 * b
```

Singuläre Matrix

```
C = [1 0 1; 1 1 0; 0 -1 1]
b = [1;2;-1]
x = C\b              % C ist singulär
                     % Lösung nicht eindeutig
```

Reduzierte Form

```
rref([A b])          % [A b] inreduzierter Form
rref([C b])
```

Interpretation der Lösung:

```
1     0     1     1
0     1    -1     1
0     0     0     0
```

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

Probe

```
% Lsg ist [1;1;0] + lambda*[-1;1;1]
C*[1;1;0]                    % = [1;2;-1]
C*([1;1;0] + 5*[-1;1;1])     % = [1;2;-1]
```

19 Lineare Optimierung

$$\begin{array}{lrcl} \min\limits_{x} & c^T \cdot x & & \\ \text{unter} & Ax & \leq & b \\ & x & \geq & l \end{array}$$

mit $x \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m, c \in \mathbb{R}^n, l \in \mathbb{R}^n$.

```
c = [-2;-3];
A = [1 2; 1 1];
b = [10;6];
l = [0;0];

[x, fWert] = linprog(c,A,b,[],[],l)
```

20 Polynome

`p = [1 -8 2 1 -12]`	$x^4 - 8x^3 + 2x^2 + x - 12$
`roots(p)`	Nullstellen von p
`poly([1 2 3])`	Polynom mit den Nullstellen $1, 2, 3$
`conv([1 1],[1 1])`	$(x+1) \cdot (x+1)$
`[1 2 1] + [3 0 0]`	$(x^2 + 2x + 1) + (x^3)$
`[1 2 3 4 5 6] + [0 p]`	Polynome müssen gleiche Länge haben
`p = [1 5 4 -4]`	
`x = linspace(-3,2,20);`	
`y = polyval(p,x);`	Auswertung des Polynoms
`plot(x,y)`	wohl Nullstelle bei $x = -2$
`deconv(p,[1 2])`	Polynomdivision durch $x + 2$
`polyder(p)`	Ableiten des Polynoms
`polyint(p)`	Integration des Polynoms

20.1 Interpolation mit Polynomen

```
t = 0:.5:5;
x = .5*t.*t - t +2 +rand(size(t));
plot(t,x)                % Messdaten mit "Störung"
pause

p = polyfit(t,x,2);      % Polynom vom Grad 2
pp = polyval(p,t);

q = polyfit(t,x,1);      % Gerade durch Punkte
qq = polyval(q,t);

plot (t,x,'r.',t,x,'-r',t,pp,'-g',t,qq,'-b')
legend('Messdaten','Messdaten','quadr.Interpol',
```

```
        'lin.Interpol')
```

20.2 Spline-Interpolation

```
x = [0 5 4 3 1 2]
t = 1:6;

ti = linspace(1,6,101);
xx = spline(t,x,ti)

plot(ti,xx)
hold on
plot(t,x,'ro')
```

Zweidimensional

```
x = [0 5 4 3 1 2]
y = [4 2 -1 3 5 0]
t = 1:6;
ti = linspace(1,6,101);

xx = spline(t,x,ti)
yy = spline(t,y,ti)

plot(xx,yy)
hold on
plot(x,y,'ro')
hold off
```

21 Function functions

Funktionen, die Funktionen als Argumente entgegennehmen

`fplot(@sin,0,2*pi)`	Plot von sinus im Intervall $[0, 2\pi]$
`fzero(@sin,2)`	Bestimmen der Nullstelle mit Startschätzung $x = 2$
`feval(@sin,2)`	Auswerten der Funktion an der Stelle $x = 2$

22 Komplexe Zahlen

`z = 1 + 2i`	Komplexe Zahl $1 + 2i$
`real(z)`	Realteil
`imag(z)`	Imaginärteil
`conj(z)`	Konjugiert komplexes
`abs(z)`	Absolutbetrag
`angle(z)`	Winkel für Polarkoordinaten

Apfelmännchen

Für welche $c \in \mathbb{C}$ konvergiert diese Folge?

$$z_1 = c$$

$$z_{n+1} = z_n^2 + c$$

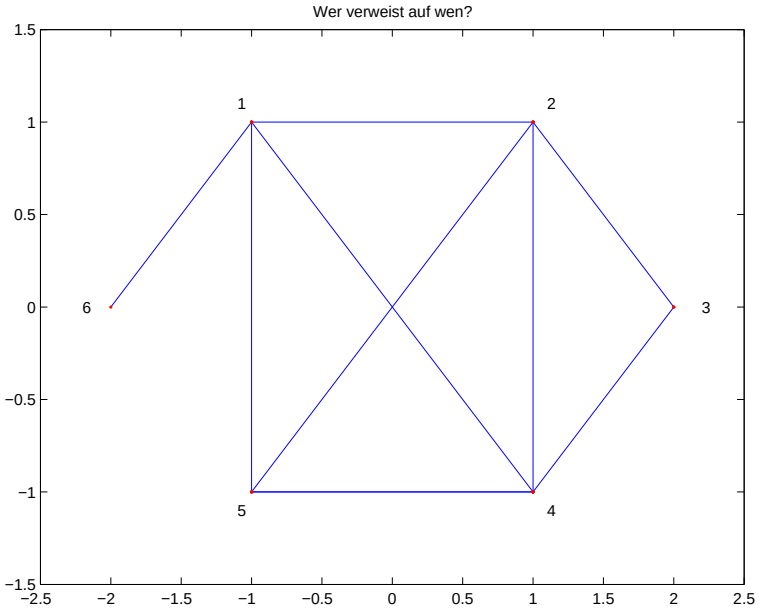
```
m=201;
l1 = linspace(-2.5,1.5,m);
l2 = linspace(-1.5,1.5,m);

% Speichere hier ab, wie oft an jedem Punkt
% iteriert wurde.
A = zeros(m);

for x=1:m
    for y=1:m
        c = l1(x)+l2(y)*i;
        z = c;
        for j=1:100
            z = z^2+c;
            if (abs(z)>100)
                A(y,x)=j;
                break;
            end
        end
    end
end

image(A)
colormap(gray(50))
colormap(hsv)
```

23 Wie funktioniert Google?

Die Spalten der Übergangmatrix M geben an, zu welchen anderen Seiten verlinkt wird. $M_{2,1} = 1 \Longleftrightarrow$ Link von Seite 1 nach 2

`GOOGLE.dat`: Übergangsmatrix

```
0     0     0     0     0     1
1     0     1     1     0     0
0     0     0     1     0     0
1     0     0     0     1     0
1     1     0     1     0     0
0     0     0     0     0     0
```



Laden der Matrix M und Plotten

```
M = load('GOOGLE.dat')

% Plotten von M
xy = [ -1 1; 1 1; 2 0; 1 -1; -1 -1; -2 0];
gplot(M,xy,'b-');
hold on
gplot(M,xy,'r.');
axis([-2.5 2.5 -1.5 1.5]);
text(xy(:,1)*1.1,xy(:,2)*1.1,['123456'])
hold off
title(['Wer verweist auf wen?'])
```

Bestimmung der Wahrscheinlichkeitsmatrix P

```
n = length(M)
c = .85

% Matrix Q der Uebergangswahrscheinlichkeiten i->j
g = sum(M)
G = ones(n,1)*g
Q = M./G

% 15% Wahrscheinlichkeit für Eingabe einer bel. URL
N = ones(n)/n
P = c*Q + (1-c)*N
```

Beginne mit „beliebigem“ Startvektor S. Z.B. sei jede Seite am Anfang gleichbeliebt. Nach k Links erreicht man die Seite j mit der Wahrscheinlichkeit p_j.

```
% Startvektor
S = ones(n,1)/n

% nach k "Links": Wahrscheinlichkeitsverteilung p
p = P*P*P*P*P*P*S
sum(p)              % Wahrscheinlichkeitsvektor
```

Oder löse für stationären Fall $Px = x$ bzw. $Px = 1 \cdot x$, d.h. suche Eigenvektor zum Eigenwert $\lambda = 1$.

```
[D,V] = eig(P)
pg = D(:,1)         % der erste EW ist 1
pg2 = pg/sum(pg)    % Normierung
```

24 Einzellizenzen der Toolboxes

- Welche Toolboxes sind an der Uni installiert?
- Wieviele Lizenzen gibt es?
- Wieviele Lizenzen sind gerade frei?
- Wer blockiert eine Lizenz (eventuell unbeabsichtigt)?

```
$ cd /home/matlab/flexlm
$ lmstat -a                          % Alle Infos
$ lmstat -f Optimization_Toolbox     % Wer benutzt die Toolbox?
$ lmstat -i Optimization_Toolbox     % Allgemeines über Toolbox
```