M-Files und Skript-Files

- Selbstdefinierte Funktionen (proceduren), die über ihren Namen aufgerufen werden.
- M-Files sind im ASCII-Format zu schreiben und können im MATLAB-Editor erzeugt werden.
- <filename>.m

  filename muss dem Funktionsnamen entsprechen.
- **Vereinbarung:** Namen von selbsterzeugten M-Files beginnen mit den Initialen des Bearbeiters.

  z.B.: Klaus Schulze schreibt einen M-File zum Gauss-Algorithmus mit dem Filenamen: ksgauss.m

M-Files

- Alle selbst erzeugten M-Files werden im Verzeichnis I:\matlab\ *.* abgelegt.
- Werden M-Files in einem Unterverzeichniss von I:\matlab\ abgelegt, so ist dieses im startup-File anzugeben.

  z.B.: Unterverzeichnis I:\matlab\lingleich\ *.*

  Eintrag im startup-File:

  path(path ‘I:\matlab\lingleich\‘);

  Achtung: startup-File nicht löschen !

Beispiel eines M-Files

```
function x = hghilbert(n)
% Lösen der Gleichung A*x = b mit der Hilbert-Matrix A
% und der rechten Seite b=A*x0 (x0=(1 .... 1)) nach dem
% Gauss Algorithmus.
if nargin < 1  % number of function input arguments
    n = input(' Eingabe der Gleichungsanzahl n: ');
end;
A = hilb(n);
b = A*ones(n,1);
x = A\b;
```

Aufruf (Beispiel):

```
>> hghilbert(5)
>> n = 5
```

Control Flow (1)

for – Loop:

```
for x = array
      Anweisungen
end
```

while – Loop:

```
while Ausdruck
      Anweisungen
end
```

Beispiel 1:

```
for n = 1:2:10
      x(n) = sin(n*pi/10);
end
```

Beispiel 2:

```
for k = 1:5
      kk = k^2;
end
```

Beispiel 3:

```
t = 0;
while t < 100;
      t = t+1;
end
```

Eine break-Anweisung schließt beide Schleifen… !

Control Flow (2)

if – else – end Konstruktionen:

```
if  Ausdruck
   Anweisungen
end
```

```
if  Ausdruck
   Anweisungen
else
   Anweisungen
end
```

```
if  Ausdruck
   Anweisungen
elseif  Ausdruck
   Anweisungen
elseif  Ausdruck
   Anweisungen
end
```

```
if  x > 0
    y = 1;
elseif  x < 0
   y = -1;
else
   y = 0;
end
```

Beispiel zur Codierung von y = sign(x)

Control Flow (3)

Relationale Operatoren:

< ; > ; <= ; >=
== „gleich“
~= „ungleich“

Logische Operatoren:

& „und“
| „oder“
~ Negation

Beispiel:

```
if  ~a
   x = 1;   % wenn a = 0
else
   x = 1;   % wenn a <> 0
end
```

Control Flow (4)

switch – case Konstruktionen

```
switch  Ausdruck
    case  Test-Ausdruck 1
        Anweisungen
    case  Test-Ausdruck 2
        Anweisungen
    ……..……
    case  Test-Ausdruck n
        Anweisungen
    otherwise
        Anweisungen
end
```

Beispiel:

```
switch  k
    case  3
        x = 5;
    case  27
        x = 100;
    otherwise
        x = 0;
end
```

Quadratische Gleichung (1)

$$x^2 - 2px + q = 0$$

```
function QUADGLEI
% Berechnung der NST einer quadratischen Gleichung.
char = 'y';
while char == 'y' | char == 'Y'
    % Eingabe der Parameter
    ……..
    % Berechnung der Nullstellen
    ……..
   disp('  ')
   char = input(' Noch eine quadratische Gleichung? ', 's');
end
```

Quadratische Gleichung (2)

Eingabe der Parameter:

```
disp(' Loesung der quadratischen Gleichung x^2 - 2px + q = 0 ')
  disp('  ')
  p = input(' Eingabe des Parameters p:  ');
  disp('  ')
  q = input(' Eingabe des Parameters q:  ');
  disp('  ')
  disp(' Nullstellen x1 und x2 der quadratischen Gleichung: ')
  disp('  ')
```

Quadratische Gleichung (3)

Berechnung der Nullstellen:

```
r = p*p - q;          % Berechnung der Diskriminante

if r < 0              % Komplexwertige NST
    r = sqrt(-r);
                      %      x1 = p + i*r;
                      %      x2 = p -  i*r;
    str1 = ' Nullstelle x1: %5g + %5g i\n';
    str2 = ' Nullstelle x2: %5g -  %5g i\n';
    fprintf(str1,p,r)
    fprintf(str2,p,r)
```

Quadratische Gleichungen (4)

```
else                    % Reellwertige NST
    r = sqrt(r);
    if p < 0
       r = -r;
    end
    x2 = p + r;
    if x2 == 0
       x1 = 0;
    else
       x1 = q/x2;
    end
    fprintf(' Nullstelle x1:  %5g\n Nullstelle x2:  %5g\n', x1, x2)
end
```

Eingabe von Punkten (1)

```
function xy = NNeingabe(ptmin,ptmax);
% ========================================
% Grafische Eingabe von Punkten in der Ebene per Mouse.
% ptmin: Mindestanzahl der einzugebenden Punkte
% ptmax: Hoechstanzahl der einzugebenden Punkte
% ========================================
hold on
…. Definition von Parametern: Eingabe von Punkten (2)
…. Schleife zur Punkteingabe: Eingabe von Punkten (3)
hold off
```

Eingabe von Punkten (2)

```
% Definition von Parametern
KMIN = 2;
KMAX = 10000;
if nargin < 1,
   ptmin = KMIN;
   ptmax = KMAX;
elseif nargin < 2
   ptmax = KMAX;
end;
% axis([-1 1 -1 1]);
xy = [];
n = 0;
```

Eingabe von Punkten (3)

```
%  Schleife zur Punkteingabe
but = 1;
while (but == 1 | n < ptmin) & (n < ptmax)
   [xi,yi,but] = ginput(1);
   if length(but) == 0 | but ~= 1,
      but = 2;
   else
      plot(xi,yi,'ro')
      n = n + 1;
      text(xi,yi,[' ' int2str(n)]);
      xy = [xy; xi,yi];
   end;
end;
```

Spline Kurven (1)

```
function NNtestspl
% Zeichnen zweidimensionaler kubischer Spline-Kurven
axis([-1 1 -1 1]);          % Darstellungsbereich der Kurve
nstep = 200;                % Parameter zur Kurvendarstellung
xyp = NNeingabe(5);  % Eingabe der Kurvenpunkte
n = size(xyp,1);            % Anzahl der Kurvenpunkte
t = 0.0;                    % Startwert des Kurvenparameters
for k = 2:n
   % Berechnung diskreter Kurvenparameter
   t = [t; t(k-1) + norm((xyp(k,:)-xyp(k-1,:)),2)];
end
```

Spline Kurven (2)

```
px = spline(t,xyp(:,1));              % Parameter der Kurve x(t)
py = spline(t,xyp(:,2));              % Parameter der Kurve y(t)
ti = linspace(t(1),t(n),nstep);       % äquidist. Stützst. auf [t(1),t(n)]
xi = ppval(px,ti);                    % xi = x(ti)  i = 1,…,nstep
yi = ppval(py,ti);                    % yi = y(ti)  i = 1,…,nstep
hold on
plot(xi,yi,'g')                       % Zeichnen der Kurve (grün)
xlabel('x - Achse')                   % Beschriftung der x - Achse
ylabel('y - Achse')                   % Beschriftung der y - Achse
title('Kubische Spline Kurven')       % Beschreibung der Kurve
```

3D – Funktionsplots (1)

```
function testplot
% Verschiedene grafische Darstellungen einer Funktion z = z(x,y)
xbeg = -3.0;  xend =  3.0;
ybeg = -3.0;  yend =  3.0;
nx = 60;  ny = 60;
xpoint = linspace(xbeg,xend,nx);
ypoint = linspace(ybeg,yend,ny);
[x,y] = meshgrid(xpoint,ypoint);
% Funktionswerte z(i,j) der darzustellenden Funktion definieren
z =  3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) ...
   - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) ...
   - 1/3*exp(-(x+1).^2 - y.^2);
```

3D – Funktionsplots (2)

```
% 2D Contour Plot von z(x,y)
contour(x,y,z)
hold on
pause

% Darstellung des Gradienten von z(x,y) durch Pfeile
dx = x(1,2) - x(1,1);     % Spacing in x - direction
dy = y(2,1) - y(1,1);     % Spacing in y - direction
[dzdx,dzdy] = gradient(z,dx,dy);
quiver(x,y,dzdx,dzdy)  % Gradientendarstellung durch Pfeile
hold off
pause
```

3D – Funktionsplots (3)

```
% Farbliche räumliche Darstellung der Oberfläche
% Abstufung gibt die Krümmung der Oberfläche wieder
L = del2(z,dx,dy); % Approximation des LAPLACE-Operators
surf(x,y,z,L)          % 3-D shaded surface plot
shading interp         % Interpolation der Farbtabelle
pause
% 3D Netzoberfläche der Funktion z(x,y)
mesh(x,y,z)
pause
% 3D Netzoberfläche mit hidden off von z(x,y)
mesh(x,y,z)
hidden off
pause
```

3D – Funktionsplots (4)

```
% Mesh plot with underlying contour plot
hidden on
meshc(x,y,z)
pause
% Mesh plot with zero plane
meshz(x,y,z)
pause
% Filled Contourplot with 12 colours
contourf(x,y,z,12)
colorbar
pause
```

3D – Funktionsplots (5)

```
% Generate pseudocolor plot
% 12 contour lines in black
pcolor(x,y,z)
shading interp
hold on
c = contour(x,y,z,12,'k');
pause
clabel(c)
pause
hold off
close all
```

NEWTON – Verfahren (1)

```
function [str1,str2] = NNnewton(fktstr,TOL,kmax)
% Newton Verfahren zur Lösung skalarer
% nichtlinearer Gleichungen
% fktstr        Funktionsstring zur NST-Bestimmung
% TOL           Abbruchfehler der NEWTON - Iteration
% kmax          max. Iterationsanzahl
% str1, str2    Ergebnisstrings
    …. Parameter setzen: NEWTON - Verfahren (2)
    …. Vorbereitung des NEWTON - Verfahrens (3)
    …. NEWTON – Iteration (4)
    …. Irregulärer Ausgang des NEWTON – Verfahrens (5)
```

NEWTON – Verfahren (2)

```
if nargin < 2, TOL = 1.0e-10; end;
if nargin < 3, kmax = 100; end;
% Bestimmen eines NST - Bereiches
maxint = 10;
a = -1.0;     b =  1.0;
for k = 1:maxint
   fab = feval(fktstr,a)*feval(fktstr,b);
   if fab < 0, break; end;
   a = a + a;   b = b + b;
end
if k >= maxint
    disp(' Warning: No zeros found');
end;
```

NEWTON – Verfahren (3)

```
% Zeichnen der zu lösenden NST-Funktion
fplot(fktstr,[a b])
hold on
% Zeichnen einer “roten” x-Achse
plot([a b],[0 0], 'r')
% Eingabe eines Startwertes zur NEWTON – Iteration per
% mouse
xy = qeingabe(1,1);
hold on
% Startwert setzen
x = xy(1,1);
```

NEWTON – Verfahren (4)

```
for k = 1:kmax
  [f,df] = feval(fktstr,x);   % Funktionswert und Ableitung berechnen
   deltax = f/df;
   x = x - deltax;            % NEWTON - Verfahren
   plot(x,0,'ro')             % Zeichnen der Näherungswerte
   if abs(deltax) < (abs(x) + 1)*TOL                    % Abbruchfehler
      str1 = ['  number of iterations: ',num2str(k)];  % Ergebnisstr. 1
      str2 = ['  zero of ',fktstr,': ',num2str(x)];    % Ergebnisstr. 2
      break                    % Iteration benden
   end
end
```

NEWTON – Verfahren (5)

```
% Ergebnisstrings bei Überschreiten der
% Iterationsanzahl kmax
if k == kmax
   str1 = ['  max. number of iterations: ',num2str(k)];
   str2 = '  Convergence is failed';
end
hold off
```

NEWTON – Verfahren (6)

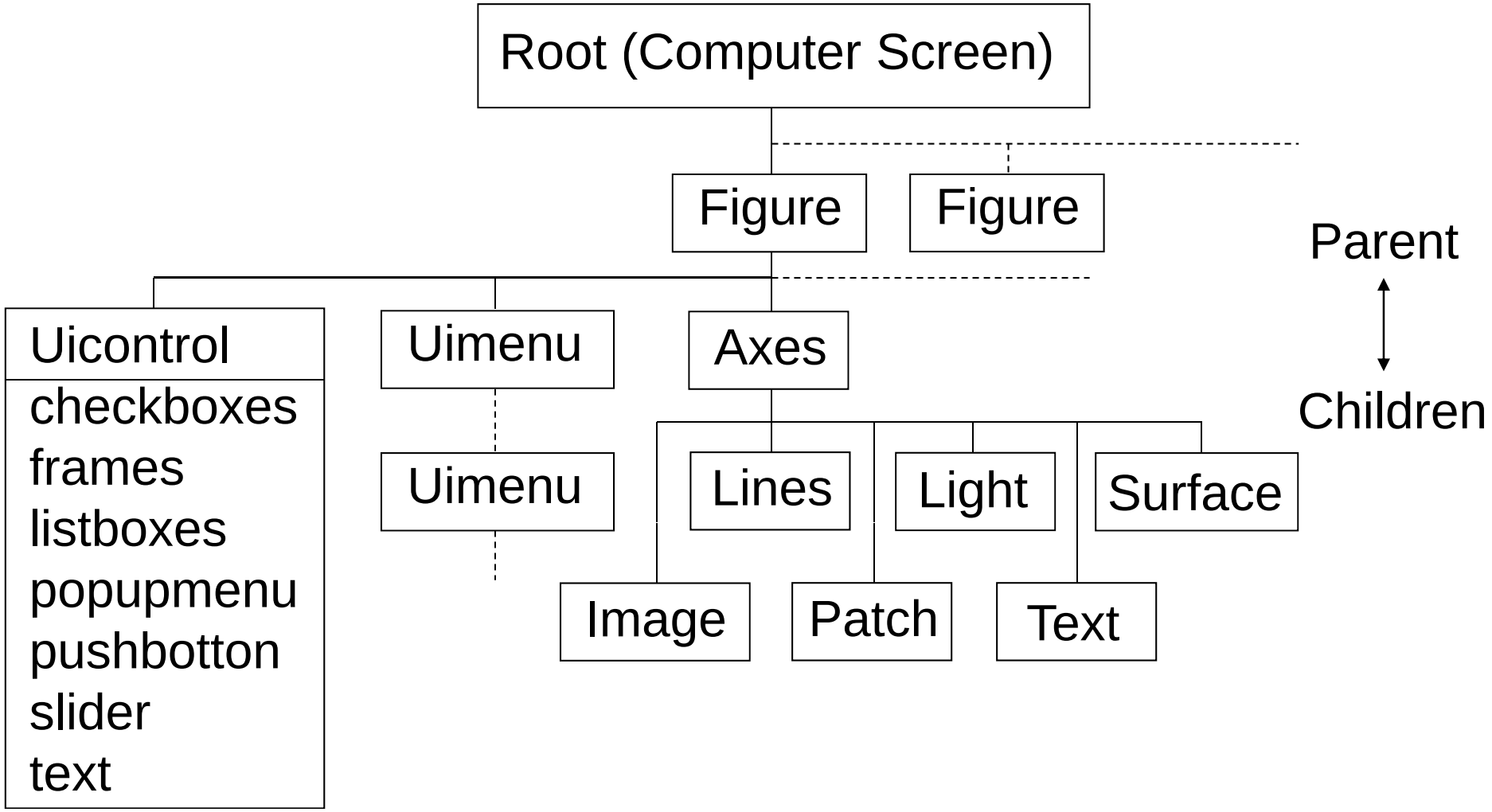
Beispielfunktion zum NEWTON - Verfahren

```
function [f,df] = NNcosxx(x)
% Berechnung des Funktionswertes f und der ersten
% Ableitung df der Funktion f(x) = cos(x) - x.
f = cos(x) - x;
if nargout > 1
   df = -sin(x) - 1;
end
```

NEWTON – Verfahren (7)

```
function NNnullstellen
% Treiberfunktion zur Berechnung von Nullstellen
% skalarer Funftionen.
datafile = uigetfile('*.m','Choose a File');
if datafile
   fktstr = strtok(datafile,'.');
else
   error('You have to choose a file');
end;
[str1, str2] = NNnewton(fktstr);
disp(str1)
disp(str2)
```

Objekte


Root (Computer Screen)
Figure
Figure
Parent
Children
Uicontrol
checkboxes
frames
listboxes
popupmenu
pushbotton
slider
text
Uimenu
Uimenu
Axes
Lines
Light
Surface
Image
Patch
Text

Objekteigenschaften

Abfrage von Objekteigenschaften

```
get(handl, ´Property Name´)
```

handl : Name des Objektes
´Property Name´ : Objekteigenschaft

Beispiel:
```
Hf_1 = gcf(gca,gco)    % get currand figure
c = get(Hf_1, ´Color´)  % get figure color
```

Setzen von Objekteigenschaften

```
set(handl, ´Prop.Name´, ´Prop.Value´)
```

Beispiel: set(Hf_1, ´Color´, ´r´)

Benutzeroberflächen (1)

```
function NNtestmenu(action)
% Testen von Callback Routinen und Menugestaltung
global fktstr datstr str1 str2   % stets verfügbare Variable
if nargin < 1,
  action = 'initialize';
end;
if strcmp(action,'initialize');
    …. Figure-Initialisierung: Benutzeroberflächen (2a) und (2b)
elseif strcmp(action,'open')
    …. File öffnen: Benutzeroberflächen (3)
elseif strcmp(action,'save')
    …. File schreiben: Benutzeroberflächen (4)
elseif strcmp(action,'analysis')
    …. Ausführung der Berechnungen: Benutzeroberflächen (5)
elseif strcmp(action,'exit')
    …. Beenden des Programms:  Benutzeroberfläche (6)
end
```

Benutzeroberflächen (2a)

```
  % Initialisierung des Benutzerwindows (figure)
  pos = get(0,'DefaultFigurePosition');
  pos = [pos(1), pos(2)-100, 560, 540];
  figNumber = figure('Name','Newton - Verfahren', ...
                     'NumberTitle','off', ...
                     'Visible','off', ...
                     'Position',pos,...
                     'BackingStore','off');
%                      'menuBar','none');
% Hm_file = uimenu(figNumber,'label','File');
  Hm_file = uimenu(figNumber,'label','Projekt','Position',1);
```

Benutzeroberflächen (2b)

```
  uimenu(Hm_file,'Label','Open File',...
                 'Callback','testmenu open');
  uimenu(Hm_file,'Label','Save File',...
                 'Callback','testmenu save');
  uimenu(Hm_file,'Label','Analysis',...
                 'Separator','on',...
                 'Callback','testmenu analysis');
  uimenu(Hm_file,'Label','Exit',...
                 'Separator','on',...
                 'Callback','testmenu exit');
%  Hm_analysis = uimenu(figNumber,'label','Analysis',...
%                   'Callback','testmenu analysis');
  set(figNumber, 'Visible','on');
```

Benutzeroberflächen (3)

```
%  File öffnen
%
datafile = uigetfile('*.m','Choose a data file');
   if datafile
      fktstr = strtok(datafile,'.');
   end;
```

Benutzeroberflächen (4)

```
%  Ergebnisse in einen File schreiben
%
[datafile,datapath] = uiputfile('*.m','Save as');
   if datafile
      datafile = strtok(datafile,'.');
      datstr = [datapath datafile '.m'];
      fid = fopen(datstr, 'wb');
      fwrite(fid, [str1, sprintf('\n'), str2]);
      fclose(fid);
```

Benutzeroberflächen (5)

```
% Durchführung der Berechnungen zum NEWTON-Verfahren
%
[str1, str2] = NNnewton(fktstr);
   disp(str1)
   disp(str2)
```

Benutzeroberflächen (6)

```
% Beenden des Programms
%
reply = questdlg('Really close - Analysis ?',...
                 'Closure','YES','NO','Cancel','default');
   if strcmp(reply,'YES')
      close all;
   end;
```