Programme aus dem Buch: Gerhard Opfer: Numerische Mathematik für Anfänger, 5. Auflage, Vieweg-Teubner, Braunschweig, Wiesbaden, 2008.

Bis auf wenige Ausnahmen sind alle Programme in zwei Teile gegliedert:
der wesentlich Teil hat die Form einer Prozedur `name.m`, die zum Testen in einem Programm mit dem Namen `name_Test.m` aufgerufen wird. Einige direkt aufrufbare Programme existieren nur mit dem Namen `name_Test.m`. Im Regelfall sind also beide Programme `name.m` und `name_Test.m` zu laden. Eine Ausnahme ist das FFT-Programm, das eine zusätzliche Prozedur `p0219_fft_umordnen` benötigt.

Kapitel 1

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 1.6. Zahlbereichsueberlauf, Seite 5.

%%%%%%%%%%%%%%%%%%%%%%%%%Programm p0106_Ueberlauf_Test%%%%%%%%%%%%%%%%%%%%%%%%%%
n=intmax-5; %MATLAB: intmax = 2147483647 = 2^31-1
for j=1:10
  n=n+1; [j,n] %ab j=6 kommt immer intmax heraus
end;
```

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 1.7. Abfrage auf Gleichheit, Seite 6.

%%%%%%%%%%%%%%%%%%%%%%%%%Programm p0107_Gleichheit_Test%%%%%%%%%%%%%%%%%%%%%%%%%
format short e     %Zahlformat in der Form 6.5432e-1
for n=1:10
  x=((1/n)/10+1)*n-n;  %1/10 in exakter Arithmetik
  if x==1/10
    disp(['Gut gegangen bei n = ',int2str(n)]);
    else
    disp(['Schief gegangen bei n = ',int2str(n),', x-1/10 = ',num2str(x-1/10)]);
  end;
end;
format short       %Zahlformat in der Form 0.6543, (Standardformat)
%Ergebnisse (MATLAB 7.2.0.283):
%    n     | x-1/10
%-------------------
% 1 u 2    |  8.3e-17
% 3        |  5.3e-16
% 4 bis 10 | -3.6e-16
```

Kapitel 2

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008. ABSOLUTELY NO WARRANTY.
%Programm 2.9. Horner-Schema fuer alle Ableitungen, Seite 16.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Prozedur p0209_Horner%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%b=p0209_Horner(a,x); Gegeben die Koeffizienten a=(a_1,a_2,...,a_{n+1})
%des Polynoms p in der Anordnung p(x):=a_1+a_2x+...+a_{n+1}x^n. Das Ergebnis
%ist b=(b_1,b_2,...,b_{n+1}) mit b_j=p^(j-1)(x)/(j-1)!, j=1,2,...,n+1.
function b=p0209_Horner(a,x);
  n=length(a)-1;
  for k=0:n-1
    for j=n-1:-1:k
      a(j+1)=a(j+1)+x*a(j+2);
    end;
  end;
  b=a;
```

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008. ABSOLUTELY NO WARRANTY.
%Programm 2.9, Beispiel 2.7. Horner-Schema fuer alle Ableitungen, Seite 16.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%Programm p0209_Horner_Test%%%%%%%%%%%%%%%%%%%%%%%%%%%%
a=[-7,-4,1,2]; x=2;
b=p0209_Horner(a,x) %Ergebnis: b=[5,24,13,2], wie in Tabelle 2.8 vorgerechnet.
```

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008. ABSOLUTELY NO WARRANTY.
%Programm 2.19. FFT mit BIT-Umkehr und Aufrufteil, Seite 25.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Prozedur p0219_fft%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%c=p0219_fft(r,n_akt,c); Schnelle Fourier Transformation angewendet auf den
%Vektor c(r+1),c(r+2),...,c(r+n_akt), geschneidert zur schnellen Auswertung von
%t_k=a_0/2+sum(1 bis n-1)(a_j cos(j*x_k) + b_j sin(j*x_k)) mit aequidistanten
%Knoten x_k:=2pik/n, fuer alle k=0,1,...,n-1. Funktioniert nur, wenn n eine
%Zweierpotenz ist. Das Ergebnis ist noch nicht geordnet.
%Dazu wird die Prozedur p0219_fft_umordnen(n,c) benoetigt.

function c=p0219_fft(r,n_akt,c);
  global n; global w;
    if n_akt>1
      m=floor(n_akt/2); q=floor(n/n_akt);
      for k=r:r+m-1
        h=c(k+1);    c(k+1)=h+c(m+k+1);
        l=mod(k*q,n); %bei grosser Zahl n Fehlerquelle
        c(m+k+1)=w(l+1)*(h-c(m+k+1));
      end; %for k
      c=p0219_fft(r,m,c);   %wird (n-1)mal aufgerufen
      c=p0219_fft(r+m,m,c);
    end; %if n_akt>1
```

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008. ABSOLUTELY NO WARRANTY.
%Programm 2.19. FFT mit BIT-Umkehr und Aufrufteil, Seite 25.

%%%%%%%%%%%%%%%%%%%%%%%%%%Prozedur p0219_fft_umordnen%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function c=p0219_fft_umordnen(n,c);
```

```
  %Der Vektor c der Laenge n wird entsprechend den FFT-Regeln umgeordnet.
  nh=floor(n/2);
  for j=0:n-1
    k=0; l=j; m=nh;
    while l>0
      k=k+mod(l,2)*m; l=floor(l/2); m=floor(m/2);
    end; %while
    if k>j
      h=c(j+1); c(j+1)=c(k+1); c(k+1)=h;
    end; %if
  end; %for j
```

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008. ABSOLUTELY NO WARRANTY.
%Programm 2.19. FFT mit BIT-Umkehr und Aufrufteil, Seite 25.
%Gebraucht werden  c=p0219_fft(r,n_akt,c);  und  c=p0219_fft_umordnen(n,c);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Programm p0219_fft_Test%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear
global n; global w;
n=8;
%1. Einige Datenueberpruefungen=================================================
%1a. Ist n zu gross?
if n>=2^16
  warning('In p0219_fft ist n moeglicherweise zu gross!');
end; %if
%1b. Ist n eine Zweierpotenz?
ja=0;
if n>0 & n==round(n) & round(log2(n))==log2(n)
  ja=1;
end; %if
if ~ja
  error('n ist in p0219_fft keine Zweierpotenz!');
end; %if=======================================================================
%2. Datenversorgung nach Beispiel 2.16, Seite 24=================================
J=[0:n-1];
w=exp(i*2*pi*J/n); %w ist Vektor der Laenge n
a(1)=8*pi*pi/3;
b(1)=0; %wird nicht gebraucht
for j=1:n
  a(j+1)=4/(j^2); %4/1, 4/4, 4/9, 4/16, ...
  b(j+1)=-4*pi/j; %b(n+1) wird nicht gebraucht
end; %for j
c(1)=a(1)/2 + a(n+1);
for j=1:n-1
  c(j+1)=0.5*(a(j+1)+a(n-j+1)+i*(-b(j+1)+b(n-j+1)));
end; %for
%Ergebnis fuer n=4: c=[13.4095, 2.2222 + 4.1888i, 1.0000, 2.2222 - 4.1888i].
%n=8:               c=[13.2220, 2.0408 + 5.3856i, 0.5556 + 2.0944i, 0.3022 + 0.8378i,
%                      0.2500, 0.3022 - 0.8378i, 0.5556 - 2.0944i, 2.0408 - 5.3856i].
%3. Aufrufteil, enthaelt Umordnung==============================================
c=p0219_fft(0,n,c);
c=real(c);        %imaginaere Rundungsfehler abschneiden
c=p0219_fft_umordnen(n,c); %Umordnen nach den FFT-Regeln
%Ergebnis fuer n=4: c=[18.8539, 4.0319, 9.9650, 20.7871].
%              n=8: c=[19.2692, 2.4408, 3.2652, 5.9009, 9.8970, 15.1256, 21.4565, 28.4206].
```

▼

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 2.25. Herstellung eines Kettenbruchs, Seite 31/32.

%%%%%%%%%%%%%%%%%%%%%%%%%%Prozedur p0225_Kettenbruch%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%[quotient,rest_polynom]=p0225_kettenbruch(zeler_polynom,nenner_polynom);
%Ausgehend von Grad(zeler_polynom) >= Grad(nenner_polynom) werden zwei
%Polynome quotient und rest_polynom nach der Formel
%zeler_polynom=nenner_polynom*quotient+rest_polynom berechnet.
%Alle Polynome p haben die Darstellung p(x)=a_1+a_2x+...+a_{n+1}x^n, werden also
%durch den Vektor (a_1,a_2,...,a_{n+1}) dargestellt. Ist a_{n+1} nicht Null,
%so ist n der Grad von p. Ist andernfalls a_{j+1} nicht Null aber
%a_{j+2}=a_{j+3}=...=a_{n+1}=0, so ist j der Grad von p.
%Enthaelt die Prozedur [g,polynom_neu]=grad(polynom);

function [quotient,rest_polynom]=p0225_kettenbruch(zeler_polynom,nenner_polynom);
  set(0,'RecursionLimit',10); %Vorsichtsmassnahme bei rekursivem Aufruf
  [gzeler,zeler_polynom]=grad(zeler_polynom);
  [gnenner,nenner_polynom]=grad(nenner_polynom);
  if gzeler >= gnenner
    rest_polynom=zeler_polynom;
    for k = gzeler-gnenner:-1:0
      quotient(k+1) = rest_polynom(gnenner+k+1)/nenner_polynom(gnenner+1);
      for j=gnenner+k:-1:k
        rest_polynom(j+1)=rest_polynom(j+1)-quotient(k+1)*nenner_polynom(j-k+1);
      end; %for j
    end; %for k
    [g_rest,rest_polynom]=grad(rest_polynom);
  end; %if

function [g,polynom_neu]=grad(polynom);
  %Ergibt eine Fehlermeldung, wenn polynom = [0,0,...,0].
  %polynom_neu ist dasselbe wie polynom, aber ohne die Nullen am Ende.
  Schranke=1e-10; %Sollte in gewissen Faellen modifiziert werden
  l=length(polynom);    g=l-1;   ell=l;
  %while ell >= 1 & polynom(ell)==0               %ideelle Form
  while  ell >= 1 & abs(polynom(ell)) < Schranke  %numerische Form
    g=g-1; ell=ell-1;
  end; %while
  if nargout==2
    polynom_neu=polynom; polynom_neu(g+2:l)=[];
  end;
```

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 2.25. Herstellung eines Kettenbruchs, Seite 31/32.

%%%%%%%%%%%%%%%%%%%%%%%%%%Programm p0225_Kettenbruch_Test%%%%%%%%%%%%%%%%%%%%%%%%
zeler=[-2,3,4];   %Daten aus Beispiel 2.22, Seite 28.
nenner=[5,-4,2];  %Ergebnisse dazu in Tabelle 2.25, Seite 33.
le=2; z=1;%
ze{1}=zeler;
```

```
while le>0
  z=z+1;
  [quotient,rest]=p0225_kettenbruch(zeler,nenner);
  qu{z}=quotient;
  ze{z}=nenner;
  zeler=nenner;
  nenner=rest;
  le=length(rest);
end; %while
%Ergebnisse zeigen:
format rat %rationales Anzeigeformat
[0,ze{1}]
for j=2:z
  [j-1,ze{j}]
  [j-1,qu{j}]
end;
format short
%Einige Tests: r(0)=-2/5; r(1)=5/3;
wertbeinull=qu{2}(1)+1/(qu{3}(1)+1/qu{4}(1))        %-0.4000
wertbeieins=sum(qu{2})+1/(sum(qu{3})+1/sum(qu{4}))  % 1.6667
```

Kapitel 3

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 3.13. Dividierte Differenzen mit Neville-Reihenfolge, Seite 42.

%%%%%%%%%%%%%%%%%%%%%%Prozedur p0313_dividierte_Diff_Neville%%%%%%%%%%%%%%%%%%%%%%%%%
%c=p0313_divierte_Diff_Neville(x,f); %Aus den Daten (x_j,f_j), j=0,1,...,n werden
%die dividierten Differenzen c=(c_0,c_1,...,c_n) in Neville-Reihenfolge berechnet.
function c=p0313_dividierte_Diff_Neville(x,f);
  c=f;   n=length(f)-1;
  for j=1:n
    for k=n:-1:j
      c(k+1)=(c(k+1)-c(k))/(x(k+1)-x(k-j+1));
    end; %for k
  end; %for j
```

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 3.18. Dividierte Differenzen mit Aitken-Reihenfolge, Seite 43.

%%%%%%%%%%%%%%%%%%%%%%Prozedur p0318_dividierte_Diff_Aitken%%%%%%%%%%%%%%%%%%%%%%%%%
%c=p0318_dividierte_Diff_Aitken(x,f); %Aus den Daten (x_j,f_j), j=0,1,...,n werden
%die dividierten Differenzen c=(c_0,c_1,...,c_n) in Aitken-Reihenfolge berechnet.
function c=p0318_dividierte_Diff_Aitken(x,f);
  c=f;
  n=length(f)-1;
  for j=1:n
    cj=c(j); xj=x(j);
    for k=j:n
      c(k+1)=(c(k+1)-cj)/(x(k+1)-xj);
    end; %for k
  end; %for j
```

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 3.15. Berechnung des Wertes y:=p(xi) mit Neville-Reihenfolge.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Prozedur p0315_Neville%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%y=p0315_Neville(x,f,xi); Fr die Daten (x_j,f_j), j=1,2,...,n+1, x_j<x_{j+1}
%wird mit der Methode von Neville der Wert y=p(xi) des Interpolationspolynoms p
%an der Stelle xi berechnet. Ist xi ein Vektor, so auch y=p(xi).
function y=p0315_Neville(x,f,xi);
    n=length(x)-1; m=length(f)-1;
    y=zeros(size(xi));
    if m~=n
      error('In p0315_Neville haben x und f nicht die gleiche Laenge!');
    end;
    Add=0; Mult=0; Div=0; %Statiskik der Operationszahlen, kann auch entfallen.
    for j=1:length(xi)
      for k=0:n
        p(k+1)=f(k+1); sk=xi(j)-x(k+1); %1 Add
       Add=Add+1;
        for l=k-1:-1:0
          p(l+1)=p(l+2)+(p(l+2)-p(l+1))*sk/(x(k+1)-x(l+1)); %3 Add, 1 Mult, 1 Div
      Add=Add+3; Mult=Mult+1; Div=Div+1;
        end; %for l
      end;  %for k
      y(j)=p(1); %[n,Add,Mult,Div]
    end; %for j
```

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 3.15. Berechnung des Wertes y:=p(xi) mit Aitken-Reihenfolge.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Prozedur p0316_Aitken%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%y=p0316_Aitken(x,f,xi); Fr die Daten (x_j,f_j), j=1,2,...,n+1 x_j<x_{j+1},
%wird mit der Methode von Aitken der Wert y=p(xi) des Interpolationspolynoms p
%an der Stelle xi berechnet. Ist xi ein Vektor, so auch y=p(xi).
function y=p0316_Aitken(x,f,xi);
  n=length(x)-1; m=length(f)-1;
  y=zeros(size(xi));
  if m~=n
    error('In p0316_Aitken haben x und f nicht die gleiche Laenge!');
  end;
  Add=0; Mult=0; Div=0; %Statiskik der Operationszahlen, kann auch entfallen.
  for j=1:length(xi)
    q=f;
    for k=1:n
      for l=k:n
        q(l+1)=q(l+1)+(q(l+1)-q(k))*(xi(j)-x(l+1))/(x(l+1)-x(k));
        % 4 Add, 1 Mult, 1 Div
        Add=Add+4; Mult=Mult+1; Div=Div+1;
      end; %for l
    end; %for k
    y(j)=q(n+1);  %[n,Add,Mult,Div]
  end; %for j
```

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 3.13 und 3.18. Dividierte Differenzen Neville und Aitken, Seite 42/43.

%%%%%%%%%%%%%%%%%Programm p0313u18_dividierte_Diff_Test%%%%%%%%%%%%%%%%%%%%%%%%
x1=linspace(0,15,16)';  f1=x1.^5;  %f1=rand(size(x1));
%c_neville1=p0313_dividierte_Diff_Neville(x1,f1);
%c_aitken1=p0318_dividierte_Diff_Aitken(x1,f1);
%norm(c_aitken1-c_neville1)
%-------------------------------------------------------------------------------
x2=[0,1,3,4]';  f2=[1,2,3,2]';  %Beispiel aus Tabelle 3.23, S. 45
c_neville2=p0313_dividierte_Diff_Neville(x2,f2); %Erste Zeile aus Tabelle 3.23
c_aitken2=p0318_dividierte_Diff_Aitken(x2,f2);
no=norm(c_aitken2-c_neville2) %ist Null
```

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 3.15 und 3.16. Berechnung von y:=p(xi) nach Neville und Aitken, S. 42

%%%%%%%%%%%%%%%%%%%%%%Programm p0315u16_Neville_Aitken_Test%%%%%%%%%%%%%%%%%%%%
%x=rand(10,1);          %zufaellige Verteilung, gibt schnell Fehler
                        %fuer groessere n (etwa ab n = 15)
x=linspace(0,1,12)';    %aequidistante Verteilung
x=unique(x); Polynomgrad=length(x)-1
%Die x-Werte werden aufsteigend sortiert und moegliche Dubletten weggelassen.
f=x.*x; %Das Interpolationspolynom p stimmt in diesem Fall
        %(theoretisch) mit p(x)=x^2 ueberein.
y1=p0315_Neville(x,f,pi);
y2=p0316_Aitken(x,f,pi);
disp('Fehler an der Auswertungsstelle pi:');
Fehler_Neville=pi^2-y1
Fehler_Aitken=pi^2-y2

%Anwendung auf mehrere Auswertungen simultan.
xi=rand(25,1);
xi=unique(xi);
z1=p0315_Neville(x,f,xi);
z2=p0316_Aitken(x,f,xi);
Fehler_Neville=norm(xi.^2-z1)
Fehler_Aitken=norm(xi.^2-z2)
Unterschied=norm(z1-z2)
```

Kapitel 4

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 4.36. de Casteljau-Algorithmus, Seite 118.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Prozedur p0436_Casteljau%%%%%%%%%%%%%%%%%%%%%%%%
%[A]=p0436_Casteljau(a,t); Gegeben Kontrollpunkte a(1) bis a(k) in komplexer Form
%und ein Gitter t mit 0<=t(j)<t(j+1)<=1 ueber dem alle Kurven berechnet werden.
%Ist n die Laenge des Gitters, so wird eine (kxn) Matrix A berechnet, und A(j,:)
%ist fuer jedes 1<=j<=k ein Polygon. Die endgueltige Kurve ist A(1,:);
```

```
function [A]=p0436_Casteljau(a,t);
  k=length(a);
  for j=1:k
    A(j,:)=a(j)*ones(size(t));
  end;
  for r=k-1:-1:1
    for j=1:r
      A(j,:)=(1-t).*A(j,:)+t.*A(j+1,:); %wiederholte Konvexkombinationen.
    end; %for j
  end; %for k %Das Endergebnis steht auf A(1,:).
```

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 4.36. de Casteljau-Algorithmus, Seite 118.

%%%%%%%%%%%%%%%%%%%%%%%%%%Programm p0436_Casteljau_Test%%%%%%%%%%%%%%%%%%%%%%%%%%
close all;
a(1)=-5+3i; a(2)=-2-2.5i; a(3)=0; a(4)=2-2.5i; a(5)=5+3i;   %Wie im Buch, S. 118.
t=linspace(0,1,41); geschlossen=0;
[A]=p0436_Casteljau(a,t);
k=length(a);
for j=1:k
  if j<k
    text(real(a(j))+0.08,imag(a(j))+0.08,int2str(j));
  end; %if
  if j==k & ~geschlossen
    text(real(a(j))+0.08,imag(a(j))+0.08,int2str(j));
  end; %if
  if j==1, hold on;
  end; %if
end; %for j
plot(a(:),'--'); plot(a(:),'s'); plot(a(:),'*'); %dicke Punkte
%Kontrollpolygon ist jetzt fertig.
set(gcf,'DefaultLineLineWidth',2);  %dickere Linie
plot(A(1,:),'b'); hold off;  %Die fertige Kurve ist blau und dick.
set(gcf,'DefaultLineLineWidth',0.5); %wieder normaldicke Linie
%-----------------------------------------------------------------
%Dasselbe nochmal mit geschlossener Kurve
figure(2);
a(6)=a(1); geschlossen=1;
[A]=p0436_Casteljau(a,t);
k=length(a);
for j=1:k
  if j<k
    text(real(a(j))+0.08,imag(a(j))+0.08,int2str(j));
  end; %if
  if j==k & ~geschlossen
    text(real(a(j))+0.08,imag(a(j))+0.08,int2str(j));
  end; %if
  if j==1, hold on;
  end; %if
end; %for j
plot(a(:),'--'); plot(a(:),'s'); plot(a(:),'*'); %dicke Punkte
%Kontrollpolygon ist jetzt fertig.
set(gcf,'DefaultLineLineWidth',2);  %dickere Linie
plot(A(1,:),'b'); hold off;  %Die fertige Kurve ist blau und dick.
```

```
set(gcf,'DefaultLineLineWidth',0.5); %wieder normaldicke Linie
%-----------------------------------------------------------------
%Vorschlag fuer weitere Tests:
figure(3)
k=5; geschlossen=0;
a=round(10*(rand(1,k)-0.5))+i*round(10*(rand(1,k)-0.5)); %Gibt interessante Bilder.
%Kontrollpunkte a sind hier ganzzahlig in [-5,5]^2. Jeder Aufruf gibt ein neues
%Bild. Will man geschlossene Kurven, so fuege man a(k+1)=a(1) und k=k+1 hinzu.
%a(k+1)=a(1); k=k+1; geschlossen=1;
[A]=p0436_Casteljau(a,t);
for j=1:k
  if j<k
    text(real(a(j))+0.08,imag(a(j))+0.08,int2str(j));
  end; %if
  if j==k & ~geschlossen
    text(real(a(j))+0.08,imag(a(j))+0.08,int2str(j));
  end; %if
  if j==1, hold on;
  end; %if
end; %for j
plot(a(:),'--'); plot(a(:),'s'); plot(a(:),'*'); %dicke Punkte
%Kontrollpolygon ist jetzt fertig.
set(gcf,'DefaultLineLineWidth',2);  %dickere Linie
plot(A(1,:),'b'); hold off;  %Die fertige Kurve ist blau und dick.
set(gcf,'DefaultLineLineWidth',0.5); %wieder normaldicke Linie
%-----------------------------------------------------------------
disp('Einige CAD-Bilder werden gezeichnet');
```

Kapitel 5

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 5.17, adaptive Simpson-Integration, Seite 135.

%%%%%%%%%%%%%%%%%%%%%%%%%Prozedur p0517_Simpson_adaptiv%%%%%%%%%%%%%%%%%%%%%%%%%%%
%[s_ad,wie_oft]=p0517_Simpson_adaptiv(a,b,alte_schetzung,f);
%adaptives Simpson-Verfahren zur Integration von f auf [a,b].
%wie_oft ist die Anzahl der Adaptionen, eingeschraenkt auf 50.
%f ist ein String, ein Name fuer eine Funktion, z. B. f='sin'.
function [s_ad,wie_oft]=p0517_Simpson_adaptiv(a,b,alte_schetzung,f);
  global z global_eps;
  z=z+1;
  set(0,'RecursionLimit',50);
  if z > 50
    error('Too many adaption steps in p0517_Simpson_adaptiv');
  end; %Notausgang
  c=(a+b)/2;
  h=b-a;
  ac_integral=p0517_Simpson_elementar(a,c,f);
  cb_integral=p0517_Simpson_elementar(c,b,f);
  neue_schetzung=ac_integral+cb_integral;
  if abs(neue_schetzung-alte_schetzung)>h*global_eps
    %Konvergenzbedingung nicht erfuellt
    s_ad=p0517_Simpson_adaptiv(a,c,ac_integral,f)+...
    p0517_Simpson_adaptiv(c,b,cb_integral,f);
```

```
    else
    s_ad=neue_schetzung+(neue_schetzung-alte_schetzung)/15;
  end; %if
  if nargout==2
    wie_oft=z;
  end; %if
```

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 5.17. adaptive Simpson-Integration, S. 135

%%%%%%%%%%%%%%%%%%%%%%%%Prozedur p0517_Simpson_integrand%%%%%%%%%%%%%%%%%%%%%%%%
function wert=p0517_Simpson_integrand(x);
  %Bei adaptiver Simpson-Integration ist die zu integrierende
  %Funktion f='p0517_Simpson_integrand';

  %Beispiel 1:
  %wert=1./x; return   %Das exakte Integral ueber [1,e] ist Eins

  %Beispiel 2:
  %wert=sin(x); return %Das exakte Integral ueber [0,pi/2] ist Eins

  %Beispiel 3: eine stetige, aber nicht differenzierbare Funktion:
  alpha=0.1;
  I=find(x>=alpha);
  J=find(x<alpha);
  x(I)=1./x(I);
  x(J)=1/alpha;
  wert=x;  return
  %Nach 32 Schritten erhaelt man Integral  = 3.3025 8658 1536 93
  %Exakter Wert ist 1 + log(10)            = 3.3025 8509 2994 05
  %mit einem Fehler von 1+log(10)-Integral = 1.4885e-06

  %Beispiel 4:
  %wert=cos(4*x);
  %Das exakte Integral ueber [0,2pi] ist Null.
  %Es kommt bei der Adaption aber 2*pi heraus.
```

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 5.17, adaptive Simpson-Integration, S. 135.

%%%%%%%%%%%%%%%%%%%%%%%%Prozedur p0517_Simpson_elementar%%%%%%%%%%%%%%%%%%%%%%%%
%s=p0517_Simpson_elementar(a,b,f); Simpson Integration im Intervall [a,b] ueber f.
%f ist ein String, also ein Name fuer eine Funktion, z. B. f='sin'. In dieser
%Umgebung ist f='p0517_Simpson_integrand'.
function s=p0517_Simpson_elementar(a,b,f);
  s=(feval(f,a)+4*feval(f,(a+b)/2)+feval(f,b))*(b-a)/6; %Simpson klassisch
```

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 5.17, adaptive Simpson-Integration, S. 135

%%%%%%%%%%%%%%%%%%%%%%%%Programm p0517_Simpson_adaptiv_Test%%%%%%%%%%%%%%%%%%%%%%%%
%Verwendet werden die Unterprogramme:
```

```
%1. [s,z]=p0517_Simpson_adaptiv(a,b,alte_schetzung,f);%wesentliches Programm
%2. s=p0517_Simpson_elementar(a,b,f);                 %elementare Simpsonformel
%3. wert=p0517_Simpson_integrand(x);                  %Die zu integrierende Funktion

global z global_eps;
z=0;
f='p0517_Simpson_integrand';
a=0; b=1;       %Beispiel 3
%a=1; b=exp(1); %Beispiel 1
%a=0;  b=pi/2;  %Beispiel 2
%a=0; b=2*pi;   %Beispiel 4
global_eps=(15*5e-5)/(b-a);
[Integral,Schritte]=p0517_Simpson_adaptiv(a,b,p0517_Simpson_elementar(a,b,f),f)
```

▼

Kapitel 6

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 6.4, Gauss-Elimination, Seite 168.

%%%%%%%%%%%%%%%%%%%%%%%%%Prozedur p0604_Gauss_elimination%%%%%%%%%%%%%%%%%%%%%%%
%[L,R]=p0604_Gauss_elimination(A); Die quadratische Matrix A der Ordnung n wird
%zerlegt: A=LR. L ist eine linke Dreiecksmatrix mit Einsen in der Diagonalen,
%R ist eine rechte Dreiecksmatrix. Die Zerlegung wird ohne Pivotsuche hergestellt,
%funktioniert also nur, wenn alle Hauptabschnitte von A nicht singulaer sind. Die
%Loesung von Ax=LRx=b erfolgt dann mittels  Ly=b;  Rx=y; y wird durch Vorwaerts-
%einsetzen, x durch Rueckwaertseinsetzen berechnet. In diesem Programm ist L auch
%in der linken Haelfte von A (ohne Diagonalelemente) und R auch in der rechten
%Haelfte (inkl. Diagonalelemente) von A gespeichert.
function [L,R]=p0604_Gauss_elimination(A);
  [m,n]=size(A);
  if m~=n, error('A in p0604_Gauss_elimination ist nicht quadratisch'); end;
  L=eye(n); R=zeros(n,n);
  for j=1:n-1
    for k=j+1:n
      if A(j,j)==0,
        error(['Hauptabschnitt ',int2str(j),...
               ' von A in p0604_Gauss_elimination ist singulaer']);
      end; %if
      if abs(A(j,j))<1e-10,
        warning(['Hauptabschnitt ',int2str(j),...
                 ' von A in p0604_Gauss_elimination ist fast singulaer']);
      end; % if
      c=A(k,j)/A(j,j); A(k,j)=c;
      for l=j+1:n
        A(k,l)=A(k,l)-c*A(j,l);
      end; %for l
    end; %for k
  end; %for j
  for l=1:n
    L(l,1:l-1)=A(l,1:l-1);   R(l,l:n)=A(l,l:n);
  end;
```

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 6.4, Gauss-Elimination, Seite 168.

%%%%%%%%%%%%%%%%%%%%%%%%Programm p0604_Gauss_elimination_Test%%%%%%%%%%%%%%%%%%%%%
N=10;
A1=rand(N,N);      %(NxN)-Matrix mit zufaelligen Elementen in [0,1].
A2=hilb(N);        %Hilbertmatrix der Ordnung N;
A3=vander([1:N]); %Vandermondematrix der Ordnung N (s. help vander in MATLAB)
[L1,R1]=p0604_Gauss_elimination(A1); n1=norm(A1-L1*R1);
[L2,R2]=p0604_Gauss_elimination(A2); n2=norm(A2-L2*R2);
[L3,R3]=p0604_Gauss_elimination(A3); n3=norm(A3-L3*R3);
format short e
[n1,n2,n3]  %%1.4520e-15   3.0714e-17   1.1295e-07 bei N=10.
format short
```

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 6.14, Vorwaerts- und Rueckwaertseinsetzen beim Eliminationsverfahren,
%Seite 179.

%%%%%%%%%%%%%%%%%%%%Prozedur p0614_vorwaerts_rueckwarts_einsetzen%%%%%%%%%%%%%%%%%%
%[x,y]=p0614_vorwaerts_rueckwaerts_einsetzen(L,R,b)
%Das Gleichungssystem LRx=b wird ber (a) Ly=b, (b) Rx=y geloest.
%L ist eine linke Dreiecksmatrix mit Einsen in der Diagonalen,
%und R ist eine rechte Dreiecksmatrix.
function [x,y]=p0614_vorwaerts_rueckwaerts_einsetzen(L,R,b);
  [n]=length(b);
  x=zeros(n,1); y=x; %Damit sind x,y Spaltenvektoren.
  for j=1:n
    s=b(j);
    for k=1:j-1
      s=s-L(j,k)*y(k);
    end;
  y(j)=s;
  end; %Vorwaertseinsetzen zu Ende.

  for j=n:-1:1
    s=y(j);
    for k=j+1:n
      s=s-R(j,k)*x(k);
    end;
    x(j)=s/R(j,j);
  end; %Rueckwaertseinsetzen zu Ende.
```

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 6.14, Vorwaerts- und Rueckwaertseinsetzen beim Eliminationsverfahren,
%Seite 179.

%%%%%%%%%%%%%%%%%%%%Programm p0614_vorwaerts_rueckwaerts_einsetzen_Test%%%%%%%%%%%
N=5;    A=rand(N,N);
[L,R]=p0604_Gauss_elimination(A);
```

```
x=[1:N]';   b=A*x;
[x,y]=p0614_vorwaerts_rueckwaerts_einsetzen(L,R,b);
x,  norm(L*y-b) %ist Null (bis auf Rundungsfehler)
```

Kapitel 8

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 8.55, Berechnung von Householder-Vektoren, S. 255.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Prozedur p0855_househ%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%[omega,c]=p0855_househ(x); Es wird ein Vektor omega und eine Zahl c ausgerechnet,
%so dass  H:=I-c*omega*omega' eine Householder-Matrix ist, also die Eigenschaft
%Hx=fe_1, |f|=||x|| (auch im komplexen Fall) hat.
function [omega,c]=p0855_househ(x);
  h=norm(x); %bei h=0 geht die Prozedur nicht! Siehe letzte und vorletzte Zeile.
  omega=x;
  if x(1)==0
    omega(1)=h; else
    omega(1)=x(1)+sign(x(1))*h;
  end; %if
  omega=omega/omega(1); %erste Komponente Eins
  c=1+abs(x(1))/h;  %=2/||omega||^2
```

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 8.56, Bestimmung einer OR-Zerlegung mit dem Householder-Verfahren, S. 255.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Prozedur p0856_householder%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%[Q,R]=p0856_householder(A); A ist eine reelle oder komplexe (mxn)-Matrix mit m>=n.
%Es wird die Zerlegung A=QR bestimmt. Q ist eine quadratische
%orthogonale (mxm)-Matrix (auch im komplexen Fall: QQ'=I), und R ist eine rechte
%(mxn)-Dreiecksmatrix. Es wird die Prozedur p0855_househ verwendet.
function [Q,R]=p0856_householder(A);
  [m,n]=size(A);
  Q=eye(m); R=zeros(size(A));
  for j=1:min(m-1,n)
    [u,c]=p0855_househ(A(j:m,j));
    A(j:m,j:n)=A(j:m,j:n)-c*u*(u'*A(j:m,j:n));%<===wesentlicher Teil
    R(j:m,j:n)= A(j:m,j:n);
    Q=Q-[zeros(m,j-1),[c*(Q(1:j-1,j:m)*u)*u';c*(Q(j:m,j:m)*u)*u']];
    A(j+1:m,j)=u(2:length(u)); %Householder-Vektoren
  end; %for j
```

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 8.56, Bestimmung einer OR-Zerlegung mit dem Householder-Verfahren, S. 255.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Programm p0856_householder_Test%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Beispiele aus dem Buch (Beispiele 8.52, 8.54)
A1=[0,1,2,3]';
A2=[1,2,3,4+i].';  %(im Komplexen ist MATLAB A' wie A*, aber A.' ist Transposition).
```

```
A3=[1,0,0,0;1,1,0,0;1,1,1,0;1,1,1,1];
[Q1,R1]=p0856_householder(A1);
[Q2,R2]=p0856_householder(A2);
[Q3,R3]=p0856_householder(A3);

b=[1,2,3,4]'; b=[b,2*b]; %zwei rechte Seiten
A4=[A3,b];
[mb,nb]=size(b);
[m,n]=size(A4);
[Q4,R4]=p0856_householder(A4);
x=R4(1:n-nb,1:n-nb)\R4(1:n-nb,n-nb+1:n); %Rueckwaertseinsetzen
x  %Loesung
%In MATLAB wird Ax=b durch x=A\b geloest.
```


Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 8.57, Bidiagonalisierung mit dem Householder-Verfahren, S. 256.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Prozedur p0857_bidiag%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%[B,U,V]=p0857_bidiag(A);
%Programm benoetigt p0855_househ.
%Ist B die resultierende obere Bidiagonalmatrix, so ist A=U*B*V'
%mit den hier ausgerechneten U,V.
%A wird ueberschrieben mit Householdervektoren.
function [B,U,V]=p0857_bidiag(A);
  [m,n]=size(A);    if m<n, A=A.'; disp('Matrix gestuerzt'); [m,n]=size(A); end;
  %Also ist m >= n
  U=eye(m);  V=eye(n-1); B=zeros(size(A));
  for j=1:n %Beginn des wesentlichen Teils
    [u,c]=p0855_househ(A(j:m,j)); %Anwendung auf die Spalten
    A(j:m,j:n)=A(j:m,j:n)-c*u*(u'*A(j:m,j:n));
    A(j+1:m,j)=u(2:m-j+1); %Speicherung von u in Spalte j
    U=U-[zeros(m,j-1),[c*(U(1:j-1,j:m)*u)*u';c*(U(j:m,j:m)*u)*u']];
    if j<=n-2
      [v,c]=p0855_househ(A(j,j+1:n)'); %Anwendung auf die Zeilen
      A(j:m,j+1:n)=A(j:m,j+1:n)-c*(A(j:m,j+1:n)*v)*v';
      A(j,j+2:n)=v(2:n-j)'; %Speicherung von v in Zeile j
      V=V-[zeros(n-1,j-1),...
                [c*(V(1:j-1,j:n-1)*v)*v';c*(V(j:n-1,j:n-1)*v)*v']];
    end; %if
  end; %for %Ende des wesentlichen Teils
  V=[1,zeros(1,n-1);zeros(n-1,1),V];
  B(1,1)=A(1,1);
  for j=2:n
    B(j-1:j,j)=A(j-1:j,j);
  end;
```

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 8.57, Bidiagonalisierung mit dem Householder-Verfahren, S. 256.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Programm p0857_bidiag_Test%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
A1=[0 4 8;7 8 0;4 5 7;9 2 4;5 7 8]; %Beispiel 5.58 aus dem Buch, S. 257.
```

```
[B1,U1,V1]=p0857_bidiag(A1);
n1=norm(A1-U1*B1*V1') %ist Null bis auf Rundungsfehler
%-------------------------------------------------------------------------
A2=[0 4 8;7 8 0;4 5 7;9 2 4;5 7 8+2i]; %komplexes Beispiel
[B2,U2,V2]=p0857_bidiag(A2);
n2=norm(A2-U2*B2*V2') %ist Null bis auf Rundungsfehler
%-------------------------------------------------------------------------
A3=rand(30,20)+i*rand(30,20);
[B3,U3,V3]=p0857_bidiag(A3);
n3=norm(A3-U3*B3*V3') %ist Null bis auf Rundungsfehler
%Matrix enthaelt viele Nullen-----------------------------------------------
m=20; n=10; A4=round((rand(m,n)-0.47));
Anzahl_der_Nullen=m*n-sum(sum(A4)), von=m*n
[B4,U4,V4]=p0857_bidiag(A4);
%Hier erkennt man, dass der Bidiagonalisierungsprozess so nicht funktioniert,
%da A4 nicht den Maximalrang hat.
```

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 8.104, Zeichnen einer Skalierungsfunktion, S. 299.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Prozedur p08104_skl_fkt%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Y = p08104_skl_fkt(T,H,level);  Berechnung von Skalierungsfunktionen mittels
%einer Fixpunktgleichung. Dabei ist H die Maske der Skalierungsgleichung und T
%ist ein aequidistantes Gitter auf dem die Funktion ausgerechnet werden soll.
function Y = p08104_skl_fkt(T,H,level);
  if (level == 0)
    Y=Iterationsbeginn2(T);
    else
    n=length(H);
    su=0;
    for j=0:n-1
      su=su+H(j+1)*p08104_skl_fkt(2*T-j,H,level-1);
    end;
    Y=sqrt(2)*su;
  end

function Y2=Iterationsbeginn2(T);
  %Anfangswert mit Spline der Ordnung Zwei.
  if abs(T-1)>=1
    Y2=0;
    else
    if T<=1
      Y2=T;
      else
      Y2=2-T;
    end;
  end;

function Y1=Iterationsbeginn1(T);
  %Anfangswert mit Spline der Ordnung Eins.
  if T>=0 & T<1
    Y1=1;
    else
    Y1=0;
  end;
```

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 8.104, Zeichnen einer Skalierungsfunktion, S. 299.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%Programm p08104_skl_fkt_Test%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Insgesamt zehn Beispiele. Die Masken sind h{1} bis h{10}.
close all
%=============================================================================
%1. Daubechies-Wavelet mit 2 Parametern
h{1}=[1,1]'/sqrt(2);
text1{1}='Daubechies-Skalierungsfunktion mit 2 Parametern';
text2{1}='Daubechies-Wavelet mit 2 Parametern';
%2. Daubechie-Wavelet mit 4 Parametern=======================================
%   (Chui, S. 129; Daubechies, S. 195,
h{2}=[(1+sqrt(3)),(3+sqrt(3)),(3-sqrt(3)),(1-sqrt(3))]'/(4*sqrt(2));
text1{2}='Daubechies-Skalierungsfunktion mit 4 Parametern';
text2{2}='Daubechies-Wavelet mit 4 Parametern';
%3. Daubechie-Wavelet mit 6 Parametern ======================================
h{3}=[0.3326705529500825
      0.8068915093110924
      0.4598775021184914
     -0.1350110200102546
     -0.0854412738820267
      0.0352262918857095];  %Blatter, S. 135, Daubechies, S. 195,
                            %auch Geiger-Mskr. S. 142
text1{3}='Daubechies-Skalierungsfunktion mit 6 Parametern';
text2{3}='Daubechies-Wavelet mit 6 Parametern';
%4. Daubechie-Wavelet mit 8 Parametern ======================================
h{4}=[0.2303778133088964
      0.7148465705529154
      0.6308807679398587
     -0.0279837694168599
     -0.1870348117190931
      0.0308413818355607
      0.0328830116668852
     -0.0105974017850690];  %Geiger-Mskr. S. 142, Daubechies, S. 195
text1{4}='Daubechies-Skalierungsfunktion mit 8 Parametern';
text2{4}='Daubechies-Wavelet mit 8 Parametern';
%4. B-Splines der Ordnung zwei bis sieben. ==================================
h{5}=[1,2,1]'/(2^1.5); %B-Spline der Ordnung 2 (linear)
text1{5}='B-Spline der Ordnung 2 (linear)';
%=============================================================================
h{6}=[1,3,3,1]'/(2^2.5);  %B-Spline Ordnung 3 (quadratisch)
text1{6}='B-Spline der Ordnung 3 (quadratisch)';
%=============================================================================
h{7}=[1,4,6,4,1]'/(2^3.5);   %B-Spline der Ordnung 4 (kubisch)
text1{7}='B-Spline der Ordnung 4 (kubisch)';
%=============================================================================
h{8}=[1,5,10,10,5,1]'/(2^4.5);   %B-Spline der Ordnung 5 (quartisch)
text1{8}='B-Spline der Ordnung 5 (quartisch)';
%=============================================================================
h{9}=[1,6,15,20,15,6,1]'/(2^5.5);   %B-Spline der Ordnung 6 (quintisch)
text1{9}='B-Spline der Ordnung 6 (quintisch)';
%=============================================================================
h{10}=[1,7,21,35,35,21,7,1]'/(2^6.5);   %B-Spline der Ordnung 7 (sixtisch)
text1{10}='B-Spline der Ordnung 7 (sixtisch)';
%=============================================================================
Beispiele=[2:4,7];  %Beispiele 2 bis 4 und 7 werden bearbeitet.
                    %Die Skalierungsfunktion aus Beispiel 1 ist unstetig,
```

```
60                      %daher schlechte Konvergenz.
61 Beispiele=[2];       %nur Beispiel 2
62 Beispiele=[1:10]     %alle Beispiele
63 nr=0;
64 for k=Beispiele
65   %=============================================================================
66   %Zuerst kommen die Skalierungsfunktionen:
67   nr=nr+1;
68   n=length(h{k});
69   t=linspace(0,length(h{k})-1,101);
70   set(0,'RecursionLimit',20);
71   Stufen=4; %Anzahl der Iterationen, sollte immer kleiner gleich 6 gewaehlt werden.
72   %Bei zu grosser Stufenzahl besteht die Gefahr, dass der gesamte Speicher ueberschrieben
73   %wird inkl. wesentlicher Teile des Betriebssystems.
74   lent=length(t);
75   for j=1:lent
76     y(j)=p08104_skl_fkt(t(j),h{k},Stufen);
77   end;
78   skl{k}=y; %Werte der Skalierungsfunktion k
79   figure(nr)
80   set(gcf,'DefaultLineLineWidth',1.5); %dickere Linien
81   plot(t,y); hold on
82   set(gcf,'DefaultLineLineWidth',0.5); %wieder normaldicke Linien
83   plot([t(1),t(lent)],[0,0]); %x-Achse
84   xlabel([text1{k}]);
85   hold off;
86
87   %=============================================================================
88   %Jetzt kommen die Wavelets:
89   if k<5 %fuer k>=5 (B-Splines) gibt es keine Wavelets,
90     nr=nr+1;
91     wlt=zeros(size(t));
92     w=cell(n,1);
93     for l=1:n
94       g{k}(l)=(-1)^(l-1)*h{k}(n+1-l);
95       for j=1:lent
96         w{l}(j)=p08104_skl_fkt(2*t(j)-(l-1),h{k},Stufen);
97       end; % for j
98       wlt=wlt+g{k}(l)*w{l};
99     end; %for l
100    wlt=sqrt(2)*wlt;
101    figure(nr);
102    set(gcf,'DefaultLineLineWidth',1.5); %dickere Linien
103    plot(t,wlt); hold on;
104    set(gcf,'DefaultLineLineWidth',0.5); %normal dicke Linien
105    plot([t(1),t(lent)],[0,0]); %x-Achse
106    xlabel([text2{k}]);
107    hold off
108  end; %if k
109 end; %for k
110 disp('Skalierungsfunktionen und Wavelets wurden gezeichnet');
111 disp('Zur Fortsetzung ENTER druecken!'); pause
```

Kapitel 9

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 9.28, Singulaerwertberechnung, S. 321.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Prozedur p0928_sv%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%[sigma,Fehler,Anzahl]==p0928_sv(a,b); A ist eine Bidiagonalmatrix
%mit den n Diagonalelementen a und den n-1 Superdiagoalelementn b.
%Davon ausgehend werden die Singulaerwerte sigma von A berechnet.
%Funktioniert nur, wenn A vollen Rang hat.
function [sigma,Fehler,Anzahl]=p0928_sv(a,b);
  p=real(a).^2+imag(a).^2;    q=real(b).^2+imag(b).^2;
  n=length(a);    delta=0;    zeler=0;
  while norm(q,inf)>1e-14 & zeler<5000   %(Notbremse)
    zeler=zeler+1;
    %============Beginn wesentlicher Teil==============
    d=p(1)-delta;
    for j=1:n-1
      p(j)=d+q(j);   h=p(j+1)/p(j);
      q(j)=q(j)*h;   d=d*h-delta;
    end; %for j
    p(n)=d;
    %=============Ende wesentlicher Teil===============
  end; %while
  sigma=sqrt(p); Fehler=norm(q,inf); Anzahl=zeler;
```

Hier klicken!

```
%Gerhard Opfer: Numerische Mathematik fuer Anfaenger, 5. Auflage,
%Vieweg-Teubner, Wiesbaden, 2008.  ABSOLUTELY NO WARRANTY.
%Programm 9.28, Singulaerwertberechnung, S. 321.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Programm p0928_sv_Test%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Wir behandeln die Beispiele 9.33 und 9.34 aus dem Buch (S. 329).
A1=[0 4 8;7 8 0;4 5 7;9 2 4;5 7 8]; %Beispiel 5.58 aus dem Buch, S. 257.
[B1,U1,V1]=p0857_bidiag(A1);
a1=diag(B1); b1=[B1(1,2),B1(2,3)]';
[sigma1,Fehler1,Anzahl1]=p0928_sv(a1,b1);
%-------------------------------------------------------------------------
A2=[5i,7+7i,4+4i;9+3i,4+i,4+i;8+6i,5+3i,2+5i;6+i,8+6i,3i;6+8i,3+8i,8+5i];
[B2,U2,V2]=p0857_bidiag(A2);
a2=diag(B2); b2=[B2(1,2),B2(2,3)]';
[sigma2,Fehler2,Anzahl2]=p0928_sv(a2,b2)
%sigma2=[27.07193566077686, 8.41597378316029, 6.80306437276005]';
%Beispiel S. 320 unten---------------------------------------------------
klein=1e-10; A4=[1,1;0,klein]; n=2;
[B4,U4,V4]=p0857_bidiag(A4);
a4=diag(B4); b4=[];
for j=1:n-1
  b4=[b4;B4(j,j+1)];
end;
[sigma4,Fehler4,Anzahl4]=p0928_sv(a4,b4);
%neues Beispiel----------------------------------------------------------
m=30; n=20; A3=rand(m,n);
[B3,U3,V3]=p0857_bidiag(A3);
a3=diag(B3); b3=[];
```

```
29 for j=1:n-1
30   b3=[b3;B3(j,j+1)];
31 end;
32 [sigma3,Fehler3,Anzahl3]=p0928_sv(a3,b3);
33 %-------------------------------------------------------------------------
```