Übungen Approximation

Gerhard Opfer SS 1999

Bogen 6: Übungen am 18. Mai 1999

Lösungshinweise

Präsenzaufgaben:

Aufgabe P16. Aus Zeitmangel wurde keine schriftliche Frage gestellt.

Aufgabe P17. Die entscheidende Formel stammt vom Schoenberg & Whitney [1953]. Sind t die $n+k$ Splineknoten und x die n Interpolationsknoten, so kann die entscheidende Bedingung in die Form

$$t_i < x_i < t_{i+k},\ i = 1, 2, \ldots, n \tag{L6.1}$$

gebracht werden. Da der B-Spline B_i auf $]t_i, t_{i+k}[$ positiv ist, muß der Interpolationsknoten x_i gerade in dieses Intervall fallen. Die Größe k bezeichnet dabei die *Ordnung* des B-Splines (Ordnung = Grad + 1). Ist $k = 3$ (quadratische Splines), und $t = [0, 1, 1, 3, 4, 5]$, so ist $n = 3$. Eindeutige Lösbarkeit (unabhängig von den vorgegebenen Werten) bedeutet dann $x_1 \in]t_1, t_4[=]0, 3[$, $x_2 \in]t_2, t_5[=]1, 4[$, $x_3 \in]t_3, t_6[=]1, 5[$. So führt also $x = [1, 2, 3]$ auf ein lösbares und $x = [0, 1, 4]$ auf ein unlösbares Interpolationsproblem.

Aufgabe P18. (a) Die einzige Eigenschaft, die man (ohne weiteres) nicht aus der Rekursionsformel ableiten kann, ist die Erhöhung der Glattheit in jedem Schritt. (b) Die Formel hat große numerische Vorteile gegenüber der direkten Berechnung der dividierten Differenzen, da die Formel die sukzessive Bildung von Konvexkombinationen verlangt, für die keine Auslöschung vorkommen kann. Die B-Splines bilden wie die Bernsteinpolynome eine *Zerlegung der Eins*. Daher sind sie auch im CAD-Gewerbe von großem Interesse. Es ist jedoch zu beachten, daß

$$\sum_{i=1}^{n} B_i(x) = \sum_{i=r+1-k}^{s-1} B_i(x) = 1 \text{ für alle } t_r < x < t_s. \tag{L6.2}$$

Nicht im gesamten Intervall $[t_1, t_{n+k}]$ ist also die Summe gleich 1, denn $r + 1 - k \geq 1$ impliziert $r \geq k$ und $s - 1 \leq n$ impliziert $s \leq n + 1$. Im Beispiel der Aufgabe 11 ist $k = 3, n = 5$, $t = [0, 1, 1, 3, 5, 7, 7, 7]$ und $t_k = t_3 = 1, t_{n+1} = t_6 = 7$, die Summe ist also nur in $[1, 7]$ gleich 1, nicht in $[0, 1]$. Man vgl. die Abb. L6.4.

Hausaufgaben:

Aufgabe 11. Zu beweisen war für $f := gh$ die sogenannte *Leibnizsche Produktformel*

$$f[x_j, x_{j+1}, \ldots, x_{j+k}] = \sum_{s=j}^{j+k} g[x_j, x_{j+1}, \ldots, x_s]h[x_s, x_{s+1}, \ldots, x_{j+k}]. \tag{L6.3}$$

(a) Wir definieren

$$F(x) := \sum_{s=j}^{j+k} (x - x_j)(x - x_{j+1}) \cdots (x - x_{s-1}) g[x_j, x_{j+1}, \ldots, x_s] \cdot$$
$$\sum_{t=j}^{j+k} (x - x_{t+1})(x - x_{t+2}) \cdots (x - x_{j+k}) h[x_t, x_{t+1}, \ldots, x_{j+k}].$$

und vereinbaren, daß ein leeres Produkt den Wert 1 haben soll. Beide Faktoren sind Newtonpolynome, der erste in bezug auf die aufsteigend geordneten Knoten $x_j, x_{j+1}, \ldots, x_{j+k}$, der zweite Faktor in bezug auf dieselben Knoten nur anders herum. Wir erkennen daher, daß $F(x_s) = f(x_s)$, $s = j, j+1, \ldots, j+k$, denn der erste Faktor stimmt mit $g(x_s)$ und der zweite Faktor mit $h(x_s)$ überein. Multiplizieren wir das Produkt aus, so erhalten wir schematisch

$$F = \sum_{s,t=j}^{j+k} = \sum_{s\le t}^{j+k} + \sum_{s>t}^{j+k} \quad \text{und jeder Summand liegt in } \Pi_{s+k-t}.$$

Die Summe $\sum_{s>t}^{j+k}$ verschwindet an den Knoten $x_j, x_{j+1}, \ldots, x_{j+k}$, daher muß auch die Summe $\sum_{s\le t}^{j+k}$ an den Stellen $x_j, x_{j+1}, \ldots, x_{j+k}$ mit f übereinstimmen. Diese Summe ist aber ein Polynom vom Grad k. Daher muß der Höchstkoeffizient (das ist per definitionem die dividierte Differenz!)

$$\sum_{s=t} g[x_j, x_{j+1}, \ldots, x_s]h[x_t, x_{t+1}, \ldots, x_{j+k}]$$

mit $f[x_j, x_{j+1}, \ldots, x_{j+k}]$ übereinstimmen. (Man vgl. auch die Bemerkungen in de Boor*), S. 6,).

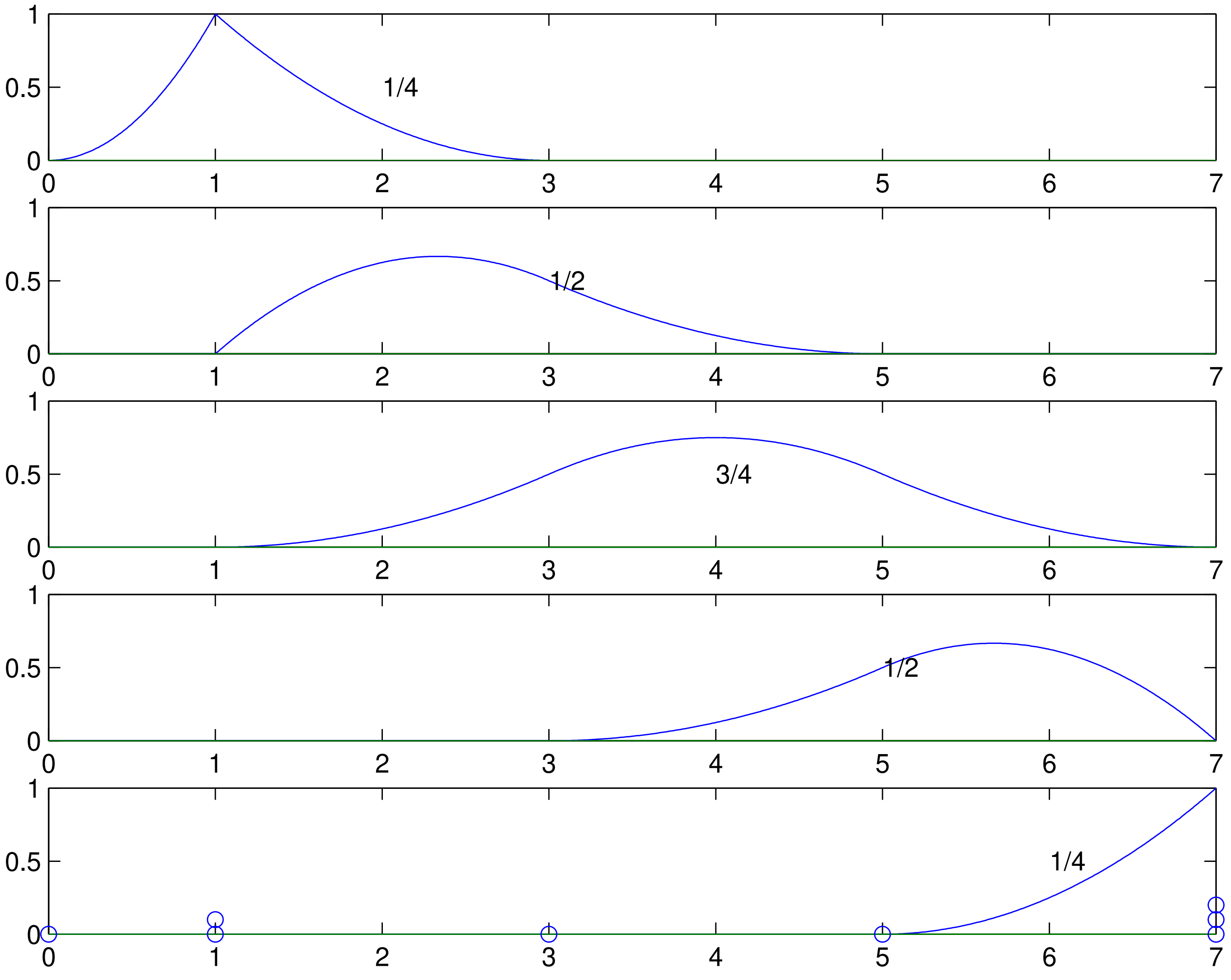

Abbildung L6.4. Alle fünf quadratischen B-Splines zu den Knoten $[0,1,1,3,5,7,7,7]$

*) C. de Boor: A practical guide to splines, Springer, New York, Heidelberg, Berlin, 1978, 392 S.

(b) Sind alle Knoten gleich x, so ist definitionsgemäß

$$gh[x_j, x_{j+1}, \ldots, x_{j+k}] = \frac{(gh)^{(k)}(x)}{k!} = \sum_{s=j}^{j+k} \frac{g^{(s-j)}(x)}{(s-j)!} \frac{h^{(j+k-s)}(x)}{(j+k-s)!} = \sum_{\ell=0}^{k} \frac{g^{(\ell)}(x)}{\ell!} \frac{h^{(k-\ell)}(x)}{(k-\ell)!}.$$

Wegen $\frac{k!}{\ell!(k-\ell)!} = \binom{k}{\ell}$ ist das die bekannte Formel zur Ableitung eines Produkts.

Aufgabe 12. Das Programm zur Erzeugung aller B-Splines vorgegebener Ordnung k zu einem festen x und vorgegebenen Splineknoten steht in de Boor, S. 134. Eine MATLAB-Version (mit Namen `106_deBoor.m`) kann unter der unten angegebenen Adresse in den web-Seiten gefunden werden. Es sind einige weitere Programme zu finden, die von diesem Programm Gebrauch machen. Das Programm `106_bsplines.m` erzeugt zu noch einzutragenden Daten alle B-Splines und zeichnet sie in eine einzige Figur. Will man gewisse Daten mit B-Splines interpolieren, so wird ein Programm `106_bsplinesinterpol.m` bereitgestellt, das zusammen mit einem weiteren Programm `106_callinterpol.m` (zur Datenversorgung und zum Aufrufen) diese Aufgabe löst. Diese Programme sind auch hinten abgedruckt. **Hinweis:** Beim Kopieren dieser Programme in windows-Umgebungen hinein, oder auf Disketten im DOS-Format ist zu beachten, daß die Programmnamen (i. e. die Namen der Dateien) verkürzt (also verhunzt) werden.
Die fünf zu zeichnenden Figuren stehen in Abbildung L6.4. Sie wurden mit dem Programm `106_bsplines.m` hergestellt.

`Programm L6.5.` MATLAB-Programm `106_deBoor.m` zur Herstellung von B-Splines

```
%=====================Vorspann und Kommentar================================
%Zu gegebener Knotenverteilung Knoten und f\"ur gegebene Werte X werden
%alle B-Splines der Ordnung k rekursiv ausgerechnet. Das Ergebnis ist
%eine Matrix B mit length(Knoten)-k Zeilen und mit length(X) Spalten.
%Die Spalten entsprechen den Werten von X, die Zeilen enthalten die
%verschiedenen B-Splines.

%Wird zum Zeichnen verwendet in: bsplines.m
%Wird zum Interpolieren und Zeichnen verwendet in: bsplinesinterpol.m

%Aufruf:                [B]=106_deBoor(k,Knoten,X)

%Dies ist i.w. der Algorithmus von de Boor, S. 134,
%geschneidert nach der bekannten Rekursionsformel f\"ur B-Splines.
%Das Ergebnis ist f\"ur jeden reellen Wert x ein k-Vektor b,
%in dem die Werte aller k B-Splines an der Stelle x, immer zugeordnet
%zum Knoten Knoten(i), gespeichert werden. Also
%b(1)=B_{i-k+1,k}(x), b(2)=B_{i-k+2,k}(x), ..., b(k)=B_{i,k}(x).
%Um die B-Splines zu zeichnen, w\"ahle man geeignet viele x.
%Will man Interpolationsprobleme (i.e. Gleichungssysteme) l\"osen, so mu\ss\ der
%Vektor Knoten die L\"ange n+k und X die L\"ange n haben. Die resultierende
%Matrix B ist dann quadratisch mit Ordnung n.

%Man vgl. auch den Kommentar in 106_indexlauf.doc
%Programm: Gerhard Opfer, 13. Mai 1999

%=================================Kopf====================================
function [B]=deBoor(k,Knoten,X)

%================================Vorspann===================================
n=length(Knoten);dis=length(X);
B=zeros(n+k-2,dis); %In B werden die B-Splines gesammelt.
%Test 1, ob gen\"ugend Knoten vorhanden sind.
if n<=k
  error(['Ordnung = ',int2str(k),', Anzahl der Knoten = ',int2str(n),...
         ', zu wenig  Knoten!']);
end; %if
%Test 2, ob die Vielfachheit der Knoten nicht \"uberschritten wird.
for j=1:n-k
  if Knoten(j)>=Knoten(j+k)
    error(['Knoten ',int2str(j),' hat zu hohe Vielfachheit']);
  end; %if
end; %for j

%============================Vorspann Ende================================

%=============================B-Splines==================================
sp=0;
for x=X;
```

```
  sp=sp+1; %wird unten gebraucht.
  b=zeros(k,1); %Platz f\"ur die B-Splines f\"ur ein einziges x
  deltaR=b;deltaL=b;
  %x (ungleich letzter Knoten) definiert eindeutig ein Intervall
  %x in   [Knoten(i),Knoten(i+1)[.
  %Ist x der letzte Knoten, so ist i=n-Vielfachheit des letzten Knotens.
  ih=find(x>=Knoten);
  i=max(ih);
  if i==n, %wenn x der letzte Knoten ist
    ih=find(Knoten==x);
    i=n-length(ih);
  end; %if
  b(1)=1;
  for j=1:k-1
    if i+j>n
      deltaR(j)=-x; else %Sonderma\ss nahme f\"ur nicht vorhandene Knoten
      deltaR(j)=Knoten(i+j)-x;
    end; %if
    if i+1-j<=0
      deltaL(j)=x;  else %Sonderma\ss nahme f\"ur nicht vorhandene Knoten
      deltaL(j)=x-Knoten(i+1-j);
    end; %if
    jp1=j+1;
    saved=0;
    for r=1:j
      term=b(r)/(deltaR(r)+deltaL(jp1-r));
      b(r)=saved+deltaR(r)*term;
      saved=deltaL(jp1-r)*term;
    end; %for r
    b(jp1)=saved;
  end; %for j
  B(i:i+k-1,sp)=b;
end; %for x
B([1:k-1,n:n+k-2],:)=[]; %\"Uberschu\ss\ wird gestrichen.
%Die Matrix B hat schlie\ss lich n-k Zeilen und dis Spalten.

%=======================Ende B-Splines erzeugen==============================
```

Programm L6.6. MATLAB-Programm `l06_bsplines.m` zum Zeichnen von B-Splines

```
%============================Kommentar=====================================
%Wir zeichnen B-Splines. Vorgegeben:
% (1) Ordnung k, (Grad =  k-1)
% (2) Splineknoten Knoten.
%Gezeichnet wird automatisch \"uber [aa,bb] mit
%aa=Knoten(1), bb=Knoten(length(Knoten)).
%Ergebnis ist intern eine Matrix B, die spaltenweise f\"ur jedes x in [aa,bb]
%die entsprechenden B-Spline-Werte und zeilenweise die B-Splines enth\"alt.
%Diese Matrix wird schlie\ss lich zeilenweise mit plot gezeichnet.
%Einzugeben hinter "clear" sind:
%(1) die Ordnung k der zu zeichnenden Splines, eine nat\"urliche Zahl k,
%(2) der Vektor der Splineknoten in der Form Knoten=[k_1,k_2,...,k_n]
%mit n >= k+1 nicht absteigend geordneten Komponenten.

%Programm: Gerhard Opfer, 13. Mai 1999
%Wesentlicher Programmteil:      l06_deBoor.m
%Bild aus de Boor, S. 112 und viele weitere Beispiele werden richtig
%reproduziert.

%==============================Vorspann====================================
%Auswahl des Splineordnung k und der Splineknoten Knoten.

%=========================Eingabedaten hier=================================
%Hier k\"onnen eigene Daten (k und Knoten) eingetragen werden.
clear
k=3;  %k=3: quadratische Splines
Knoten=[0,1,1,3,5,7,7,7];           %Beispiel \"Ubungsaufgabe
%k-fache Knoten im Inneren produzieren eine Unstetigkeitsstelle!
%========================Eingabedaten Ende===================================

%==============Vorspann Ende, Vorbereitung zum Zeichnen=====================
n=length(Knoten);
aa=Knoten(1); bb=Knoten(n);
dis=100*(bb-aa)+1;
X=linspace(aa,bb,dis);
B=l06_deBoor(k,Knoten,X); %wesentlicher Programmteil

%===============================Zeichenteil================================
%Um mehrfache Knoten besonders zu markieren, werden die mehrfachen Knoten
%extrahiert mit der folgenden am Beispiel beschriebenen Technik:
```

```
%        [122345555666]---->[255566]---->[556]---->[5]---->[].
%mindestens: einfach        doppelt    dreifach vierfach
Hilfsvektor=Knoten;
Knotenmehrfach{1}={Hilfsvektor}; %Konstruktion mit sog. cell arrays.
j=1;
while ~isempty(Hilfsvektor)
  [C,I]=unique(Hilfsvektor);
  Hilfsvektor(I)=[];
  Knotenmehrfach{j+1}={Hilfsvektor};
  j=j+1;
end; %while
jmehrfachmax=j-1; %die h\"ochste auftretende Vielfachheit eines Knotens

%Tabelle=['1/4','1/2','3/4','1/2','1/4']; %Daten aus \"Ubungsbeispiel
for p=1:n-k
  subplot(n-k,1,p);
  plot(X,B(p,:),[aa bb],[0 0]);hold on
  %text(p+1,0.5,Tabelle(3*(p-1)+1:3*(p-1)+3));
        %Nur f\"ur \"Ubungsaufgabe sinnvoll
  if p==n-k %die mehrfachen Knoten werden nur einmal, im letzten Bild markiert
    for j=1:jmehrfachmax
      Hilfsvektor=[Knotenmehrfach{j}];
      Hilfsvektor=[Hilfsvektor{1}(1,:)];
      plot(Hilfsvektor,zeros(size(Hilfsvektor))+0.1*(j-1),'o');
    end; %for j
  end; %if p
hold off
end; %for p
%print -dpsc bspluebg.ps

%============================Ende Zeichenteil================================
%========================Ende B-Splines zeichnen=============================

%==============================Service-Teil=================================

%=============Werte \"Ubungsaufgabe an den ganzzahligen Stellen=============
%x =   0    1     2         3         4         5         6       7
%==========================================================================
%      0    1   0.2500         0         0         0         0    0
%      0    0   0.6250    0.5000    0.1250         0         0    0
%      0    0   0.1250    0.5000    0.7500    0.5000    0.1250    0
%      0    0        0         0    0.1250    0.5000    0.6250    0
%      0    0        0         0         0         0    0.2500    1
%============================Service-Teil Ende==============================
```

Programm L6.7. MATLAB-Programm `l06_bsplinesinterpol.m` zum Interpolieren von Daten mit B-Splines

```
%============================Kommentar=====================================
%L\"osung von Interpolationsproblemen mit Hilfe von B-Splines. Gegeben
%  (1) Ordnung k>=2, (Grad = k-1)
%  (2) Interpolationsknotenfolge X strikt aufsteigend, L\"ange n,
%  (3) Wertefolge Y, L\"ange n,
%  (4) Splineknotenfolge Knoten, nicht absteigend, mit n+k Knoten.

%Die Angabe (4) der Splineknoten kann entfallen, dann werden die
%Splineknoten mit Mittelbildung aus den Knoten X wie in (7.2) beschrieben,
%generiert. Diese Methode ist zu empfehlen. In diesem Fall sind die
%Splineknoten Ausgabeparameter.

%Berechnet wird ein Spline S der Ordnung k mit den vorgegebenen
%oder berechneten Splineknoten, so da\ss\
%  (5) S(X)=Y.

%Wir adaptieren einen Algorithmus von de Boor, S. 134
%zur Berechnung der B-Splines. Ein Zwischenresultat ist eine n mal n-Matrix B,
%die die n Werte aller B-Splines zeilenweise enth\"alt und aus der
%der L\"osungsvektor (Spaltenvektor, L\"ange n) berechnet wird.
%Aufruf:

                 %[Loesung]=l06_bsplineinterp(k,X,Y,Knoten); oder
          %[Loesung,Knoten]=l06_bsplineinterp(k,X,Y);

%Programm: Gerhard Opfer, 13. Mai 1999
%Wesentlicher Programmteil: l06_deBoor.m

%==============================Vorspann====================================
function [Loesung,Knoten]=bsplineinterp(k,X,Y,Knoten);
n=length(X);
if nargin==3 %Splineknoten werden generiert:
  Knoten(1:k)=X(1)*ones(1,k);
```

```
  Knoten(n+1:n+k)=X(n)*ones(1,k);
  Knoten(1)=Knoten(1)-0.01;Knoten(n+k)=Knoten(n+k)+0.01;
  for j=1:n-k
    Knoten(k+j)=sum(X([j+1:j+k-1]))/(k-1);
  end; %for j
  if nargout==2
    Knoten=Knoten;
  end; %if nargout
end; %if nargin
if length(Knoten)-n~=k
  error('Diskrepanz zwischen Anzahl von Spline- und Interpolationsknoten');
end; %if
%Test: Sind die Bedingungen von Schoenberg-Whitney erf\"ullt.
for j=1:n
links=Knoten(j);rechts=Knoten(j+k);mitte=X(j);
  if links>=rechts
    error(['Knoten ',int2str(j),' hat zu hohe Vielfachheit']);
      elseif links >= mitte | mitte >= rechts
      error(['Am Knoten ',int2str(j),' Schoenberg-Whitney verletzt!']);
  end; %if
end; %for j
%============================Vorspann Ende==================================

%==============================B-Splines===================================

B=l06_deBoor(k,Knoten,X);
   %Die Matrix B hat n Zeilen und n Spalten mit einer
   %Bandbreite unterhalb k: B(j,k)=0, wenn |j-k|>=k.

%==============================Ende B-Splines==============================
Loesung=B'\Y';
%=============================L\"osung gefunden============================
```

Programm L6.8. MATLAB-Programm `l06_callinterpol.m` zum Datenversorgen und Aufrufen von `l06_bsplinesinterpol.m`

```
%Wir rufen das Interpolationsprogramm l06_bsplineinter.m mit verschiedenen Daten
%auf. Die Daten sind (X,fu(X)), fu enth\"alt den Funktionsnamen als String. Nur
%vor der Zeile "Beginn des Programms" sind Eintragungen zu machen und ggf. ist
%der Aufruf durch "[Loesung]=l06_bsplinesinterpol(k,X,Y,Knoten);" zu ersetzen.
%Programm: Gerhard Opfer, 13. Mai 1999.  Wesentlicher Programmteil: l06_deBoor.m
clear

%=================Auswahl der zu interpolierenden Funktion====================
fu='l06_Schwarz';
          %Ansonsten: fu='tan', fu ='sqrt', etc, einsetzen

%=======================Auswahl der Splineordnung=============================
k=4; %k=4: kubische Splines

%===========================Auswahl der Knoten================================
X=[1,2.5,5.25,9.5,12,14.5,17]; %7Daten, aus Schwarz, 2. Aufl., S. 137

%========================Beginn des Programms================================
n=length(X);   Y=feval(fu,X);
[Loesung,Knoten]=l06_bsplinesinterpol(k,X,Y);
%Da wir die L\"osung gern sehen wollen, ist es zweckm\"a\ss ig, die B-Splines
%f\"ur die ausgerechneten oder gew\"ahlten Knoten an vielen x-Werten
%auszuwerten. Wir rechnen dabei die fr\"uheren X-Werte ggf. nochmal aus.
%Wir nehmen das in Kauf, da das das Programm sehr vereinfacht. Will man auch
%die Splineknoten vorgeben, ersetze man den obigen Aufruf durch
%[Loesung]=l06_bsplinesinterpol(k,X,Y,Knoten);

dis=161; %die reine Willk\"ur
Intervall=linspace(X(1),X(length(X)),dis);
Bfein=l06_deBoor(k,Knoten,Intervall); %Bfein hat n Zeilen, dis Spalten
Bild=Bfein'*Loesung;

figure(1)
%Fehlerfunktion wird gezeichnet
Bildexakt=feval(fu,Intervall);
maxx=max(abs(Bildexakt'-Bild));
plot(Intervall,Bildexakt-Bild',[X(1),X(n)],[0 0]);hold on
plot(X,zeros(size(X)),'ro'); %Interpolationsknoten markiert in rot mit "o".
plot(Knoten,zeros(size(Knoten)),'gx'); %Splineknoten in gr\"un mit "x".
xlabel(['Fehlerfunktion f\"ur ',fu,', Ordnung = ',int2str(k)])
hold off

%Bei gr\"o\ss erer Abweichung werden Funktion und Spline gezeichnet
if maxx>0.1
```

```
  figure(2)
  %Funktion in rot, und Interpolierende in gr\"un, nur sinnvoll,
  %wenn der Fehler noch sichtbar ist, hier > 0.1
  plot(Intervall,feval(fu,Intervall),'r',[X(1),X(n)],[5 5]);hold on
  plot(Intervall,Bild','g');
  plot(X,5*ones(size(X)),'ro');
  plot(Knoten,zeros(size(Knoten)),'gx');
  xlabel(['Funktion ',fu,' und Spline mit Ordnung = ',int2str(k),...
          ', Fehlermax = 'num2str(maxx),]);
  hold off
end; %if maxx

%Bei vorgegebenen Interpolationsknoten und automatischen Splineknoten:
%klappt gut f\"ur tangens in [0,pi/4], k=3 und k=4
%dito Wurzel in [0.25,1]
%Beispiel Schwarz: leidlich, aber nicht schlechter als bei Schwarz.
```

Programm L6.9. MATLAB-Programm `l06_Schwarz.m` als Testfunktion in `l06_callinterpol.m`

```
%Enth\"alt eine Testfunktion von Schwarz zum Testen der
%B-Splineinterpolation in der Form
%
%                            [y]=l06_Schwarz(x);

%Beispiel aus H. R. Schwarz: 2. Aufl., S. 137, 4. Aufl., S. 159
%mit Interpolationsknoten X=[1, 2.5, 5.25, 9.5, 12, 14.5, 17]

function [y]=l06_Schwarz(x);
y=2.5*(cos(2*pi*x/16)-sin(4*pi*x/16))+5;
```

Die Aufgaben und mit einiger Verzögerung auch die Lösungen können von den web-Seiten kopiert werden:

`http://www.math.uni-hamburg.de/home/opfer/aufgaben99A.html`