Approximation

Lösungshinweise

Gerhard Opfer SS 2000

Bogen 12: Ausgabetermin: am 6. Juli 2000

Abgabetermin: am 13. Juli 2000 vor den Übungen

Begriffe aus der Differentialgeometrie (aus FARIN [1994, S. 148–149]) (alle vorkommenden Ableitungen seien vorhanden und stetig):

Kurve :	$\mathbf{x} : [a,b] \to \mathbb{R}^d$, stetig,
Spur oder *Pfad* einer Kurve $\mathbf{x}$:	$S := \{\mathbf{x}(t) : t \in [a,b]\}$,
Geschwindigkeitsvektor von $\mathbf{x}$:	$\dot{\mathbf{x}}$,
reguläre Kurve :	$\dot{\mathbf{x}}(t) \neq \mathbf{0}$, für alle $t \in [a,b]$,
Kreuzprodukt $(d=3)$:	$\mathbf{x} := (x_1, x_2, x_3)^{\mathrm{T}}$, $\mathbf{y} := (y_1, y_2, y_3)^{\mathrm{T}}$, $\mathbf{x} \times \mathbf{y} =: \mathbf{z} =: (z_1, z_2, z_3)^{\mathrm{T}}$ mit $z_1 = x_2y_3 - x_3y_2$, $z_2 = x_3y_1 - x_1y_3$, $z_3 = x_1y_2 - x_2y_1$,
Bogenlänge :	$\sigma(t) := \int_a^t \lVert\dot{\mathbf{x}}(\tau)\rVert_2 \, \mathrm{d}\tau.$

Ist $\sigma(t) = t$, so sagen wir, daß $\mathbf{x}$ nach der Bogenlänge parametrisiert ist. Wir schreiben dann $\mathbf{x}(s)$ und $\mathbf{x}'(s) := \frac{\mathrm{d}}{\mathrm{d}s}\mathbf{x}(s)$. Für reguläre Kurven ist $\sigma(t)$ stetig differenzierbar und strikt monoton wachsend, also umkehrbar. Jede reguläre Kurve kann also nach der Bogenlänge parametrisiert werden. Man setzt $\sigma(t) = s$ und $t = \sigma^{-1}(s)$. Im folgenden ist immer $d = 3$.

Tangentenvektor :	$\mathbf{t} := \dfrac{\dot{\mathbf{x}}}{\lVert\dot{\mathbf{x}}\rVert_2}$,
Normalenvektor :	$\mathbf{m} := \mathbf{b} \times \mathbf{t}$,
Binormalenvektor :	$\mathbf{b} := \dfrac{\dot{\mathbf{x}} \times \ddot{\mathbf{x}}}{\lVert\dot{\mathbf{x}} \times \ddot{\mathbf{x}}\rVert_2}$,
Krümmung :	$\kappa(t) := \dfrac{\lVert\dot{\mathbf{x}} \times \ddot{\mathbf{x}}\rVert_2}{\lVert\dot{\mathbf{x}}\rVert_2^3}$,
	$\kappa(s) := \lVert\mathbf{x}''\rVert_2$,
Torsion :	$\tau(t) := \dfrac{\det(\dot{\mathbf{x}}, \ddot{\mathbf{x}}, \dddot{\mathbf{x}})}{\lVert\dot{\mathbf{x}} \times \ddot{\mathbf{x}}\rVert_2^2}$,
	$\tau(s) := \dfrac{\det(\mathbf{x}', \mathbf{x}'', \mathbf{x}''')}{\lVert\mathbf{x}''\rVert_2^2}$,
Schmiegebene :	$\mathbf{O}(u,v) := \mathbf{x} + u\dot{\mathbf{x}} + v\ddot{\mathbf{x}}, \quad (u,v) \in \mathbb{R}^2$ (Parameterdarstellung).

Präsenzaufgaben:

Aufgabe P34. Bestimmen Sie die Splineknoten in Aufgabe 23 unter Beachtung des Satzes von Schoenberg & Whitney.

Lösung. Lesen Sie die Lösung von Aufgabe 23.

Aufgabe P35. (a) Berechnen Sie die verschiedenen Kenngrößen der Kurve $\mathbf{x} := (\cos t, \sin t, t), t \in [0, 10\pi]$. (b) Wie sieht die Kurve aus? (c) Wie lang ist die Kurve etwa?

Lösung. Wir berechnen prophylaktisch alle Ableitungen bis zur Ordnung Drei:

$$\begin{aligned}
\dot{\mathbf{x}} &= (-\sin t, \cos t, 1),\\
\ddot{\mathbf{x}} &= (-\cos t, -\sin t, 0),\\
\dddot{\mathbf{x}} &= (\sin t, -\cos t, 0)
\end{aligned}$$

und weiter

$$\begin{aligned}
||\dot{\mathbf{x}}||_2 &= \sqrt{2},\\
\dot{\mathbf{x}} \times \ddot{\mathbf{x}} &= (\sin t, -\cos t, 1),\\
||\dot{\mathbf{x}} \times \ddot{\mathbf{x}}||_2 &= \sqrt{2},\\
\det(\dot{\mathbf{x}}, \ddot{\mathbf{x}}, \dddot{\mathbf{x}}) &= 1.
\end{aligned}$$

Damit ist $\sigma(t) = \sqrt{2}\int_0^t \,\mathrm{d}\tau = \sqrt{2}t = s$ und die Gesamtlänge $l := \sigma(10\pi) = \sqrt{2}\cdot 10\pi \approx 44.4288$. Die Kurve ist also bis auf den Faktor $\sqrt{2}$ nach ihrer Bogenlänge parametrisiert. Wir erhalten weiter $\mathbf{y}(s) := \mathbf{x}(s/\sqrt{2}) = \mathbf{x}(t)$ mit $s \in [0, l]$ und

$$\begin{aligned}
\mathbf{t} &= (-\sin t, \cos t, 1)/\sqrt{2},\\
\mathbf{b} &= (\sin t, -\cos t, 1)/\sqrt{2},\\
\mathbf{m} &= (\cos t, \sin t, 0),\\
\kappa(t) &= \frac{1}{2},\\
\kappa(s) &= \frac{1}{2},\\
\tau(t) &= \frac{1}{2},\\
\tau(s) &= \frac{1}{2}.
\end{aligned}$$

(b) Eine zylindrische Spirale mit Ganghöhe 2π, sie läuft *weinwendig* um den Mittelpunkt, denn ihre Torsion ist positiv. Kurven mit negativer Torsion heißen *hopfenwendig*. (c) Eine Umdrehung hat, projiziert auf die x-y-Ebene, die Länge 2π, die Gesamtlänge l ist also mindestens $5\cdot 2\pi \approx 31.4159$.

Aufgabe P36. Was hätten Sie in der Vorlesung oder Übung gern anders gesehen, Behandlung anderer Themen, ausführlichere oder weniger ausführliche Behandlung bestimmter Themen. Offensichtlich sind numerische Aufgaben nicht gern gesehen, warum ist das so?

Lösung. Dieses Thema wurde leider (aus Zeitmangel) nicht diskutiert.

Hausaufgaben:

Aufgabe 23. Lösen Sie das Interpolationsproblem $s(\tau_{2j-1}) = s_{2j-1}$, $s'(\tau_{2j}) = s_{2j}, j = 1, 2, \ldots, 7$ auf dem Intervall $[a, b] := [0.05, 0.4]$ mit $\tau_1 := a, \tau_3 := 0.06, \tau_5 := 0.075, \tau_7 := 0.09, \tau_9 := 0.13, \tau_{11} = 0.21, \tau_{13} := b$, mit $\tau_{2j} = \tau_{2j-1}$, $j = 1, 2, \ldots, 7$. Die sieben Interpolationsknoten τ_j sind also als doppelt aufzufassen. Die Werte und Ableitungen sind zu berechnen aus $s_{2j-1} := f(\tau_{2j-1})$, $s_{2j} := f'(\tau_{2j})$, $j = 1, 2, \ldots, 7$ mit $f(t) := t\sin(1/t)$. Verwenden Sie zur Interpolation einen kubischen Spline $s \in \langle B_{1,4}, B_{2,4}, \ldots, B_{14,4}\rangle$ mit den Splineknoten $t_1 = t_2 = t_3 = t_4 = a < t_5 \le \cdots \le t_{14} < t_{15} = t_{16} = t_{17} = t_{18} = b$ und verwenden Sie die Interpolationsknoten soweit wie möglich auch als (doppelte) Splineknoten $t_j, j = 1, 2, \ldots, 18 = n + 4$. Beachten Sie dabei die Bedingungen des Satzes von Schoenberg & Whitney für die Ordnung $k = 4$, nämlich

$$t_j < \tau_j < t_{j+4}.$$

Als Lösung zeichnen Sie s und f über $[a, b]$ in eine einzige Zeichnung. Von der Matrix $M := B_{j,4}(\tau_k)$, $j, k = 1, 2, \ldots, 14$ geben Sie bitte den oberen linken 4×4-Ausschnitt an und die ersten vier Koeffizienten $a_1, \ldots, a_4$ in der Darstellung $s = \sum_{j=1}^{14} a_j B_{j,4}$, in MATLAB-Format *short* (4 Stellen nach dem Komma).

Lösung. Die Interpolationsknoten sind

$$a = \tau_1 = \tau_2 < \tau_3 = \tau_4 < \cdots < \tau_{n-1} = \tau_n = b \text{ mit } n = 14 \tag{L12.1}$$

mit den oben angegebenen Werten. Aufgrund der gegebenen Daten ist klar, daß der gesuchte, kubische Interpolationsspline s intervallweise ausgerechnet werden kann, da s auf $[\tau_{2j-1}, \tau_{2j+1}]$ durch die vier vorgegebenen Werte $s(\tau_{2j-1}) = s_{2j-1}, s'(\tau_{2j}) = s_{2j}, s(\tau_{2j+1}) = s_{2j+1}, s'(\tau_{2j+2}) = s_{2j+2}$ bereits eindeutig bestimmt ist. Man kann also in jedem Intervall das entsprechende hermitesche Interpolationspolynom in Newtonscher Form berechnen. Formeln in Opfer [1994, S. 60]. Das Ergebnis ist in Abb. L12.1 skizziert. Setzen wir $t_{j+2} = \tau_j$, $j = 1, 2, \ldots, 14$ und $t_1 = t_2 < a$, so sind die Bedingungen von Schoenberg und Whitney erfüllt. Tatsächlich benutzen wir jedoch $t_1 = t_2 = t_3 = t_4 = a$. Man vgl. Formel (L12.13′) am Schluß dieses Bogens.

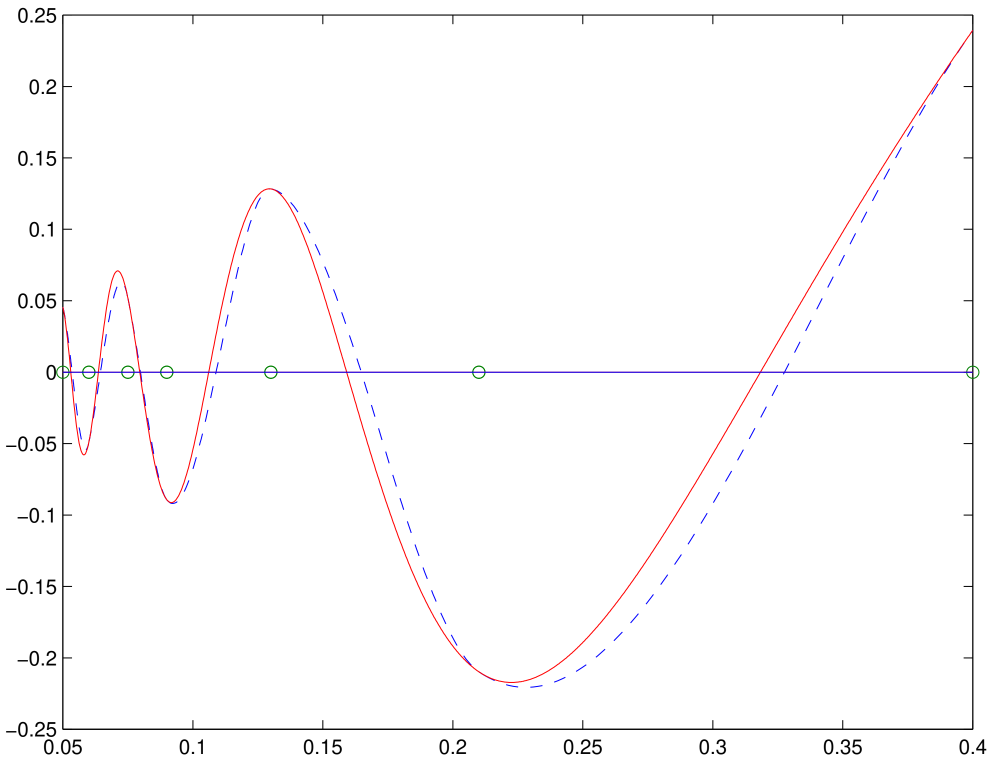

Abb. L12.1. Kubischer Interpolationsspline (gestrichelt) mit Doppelknoten

Um zu einer B-Spline-Darstellung zu kommen, benötigt man Algorithmen zur Berechnung von $s = \sum_j a_j B_{j,k}$ und $s' = \sum_j a_j B'_{j,k} = \sum_j a'_j B_{j,k-1}$ bei gegebenen Koeffizienten a_j. Diese Formeln sind aber bekannt (s. Bogen 11). Da wir in dieser Aufgabe nur erste Ableitungen benötigen, ist es zweckmäßig, die Formel dafür explizit aufzuschreiben:

$$B'_{j,k} = \frac{k-1}{t_{j+k-1} - t_j} B_{j,k-1} - \frac{k-1}{t_{j+k} - t_{j+1}} B_{j+1,k-1}.$$

Die Einzelheiten inkl. der numerischen Ergebnisse sind in dem nachfolgenden Programm L12.5 zusammengefaßt. Davor in den Programmen L12.2 bis L12.4 stehen einige Hilfsprogramme zur Berechnung von B-Splines und Splines in der Form $s = \sum_{j=1}^{n} a_j B_{j,k}$ und ein Programm zur Berechnung von a' in $s' = \sum_{j=1}^{n} a'_j B_{j,k-1}$. Das letzte Programm wurde allerdings nicht benutzt, da es einfacher ist, die ersten Ableitungen der B-Splines direkt zu berechnen. Diese MATLAB-Programme sind auch zum Probieren abrufbar in

`http://www.math.uni-hamburg.de/home/opfer/aufgaben00.html`

Programm L12.2. Berechnung aller B- Splines der Ordnung k

```
%==============================================================================
%(0) Vorbemerkungen
%Alle B-Splines k-ter Ordnung (Grad k-1) werden zu vorgegebener Knotenfolge
%Knoten (ein Vektor beliebiger L\"ange, aufsteigend geordnet, mehrfache Knoten
%erlaubt), berechnet. Wir vergr\"o\ss ern (oder verkleinern) ggf. die
%Knotenfolge an beiden Enden so, da\ss\ die beiden Endknoten die gleichen
%Vielfachheiten k=Ordnung haben. Beginnt man also nur mit zwei Knoten,
%so erh\"alt man die Bernstein-Polynome. Ist i die Anzahl der gegebenen,
%inneren Knoten (mehrfache mehrfach gez\"ahlt), so ist die Gesamtanzahl
%der verwendeten Knoten lenge:=i+2k. Jeder Spline braucht k+1 Knoten.
%Das Ergebnis ist eine Matrix BB(j,s). Mit j werden die ausgerechneten
%B-Splines der Ordnung k numeriert, s steht f\"ur die Anzahl der angegebenen
%Funktionswerte x. Das Programm hat die Form
%
%             function BB=bsplinesfct(Ordnung,Knoten,x);
%
%Die Gre x enthlt die Argumente, fr die die B-Splines ausgerechnet
%werden sollen. Sind die Knoten nicht geordnet, werden sie sortiert,
%ist die Anzahl der Knoten zu klein, wird die Rechnung abgebrochen.
%Die individuellen Splines sind (mindestens) rechtsseitig stetig, der letzte
%Spline (mit Nummer lenge-Ordnung) ist am rechten Rand auch linksseitig stetig.
%Der erste, bei a:=min Knoten beginnende Spline B(1,:) mit Tr\"ager
%[a,Knoten(k+1)] verschwindet nicht, da a vom Programm her die Vielfachheit
%k bekommt und a<Knoten(k+1). Das kann aber ausgeschaltet werden.
%Man vergleiche weiter unten unter Punkt (2).
%Programm: Gerhard Opfer, Juli 2000.
%Quelle: Carl de Boor: B(asic)-Spline Basics, ftp-server Madison, 1996.

function BB=bsplinesfct(Ordnung,Knoten,x);
%==============================================================================
%(1) Vorbesetzungen
t=Knoten;
if length(t)<2, error('Zu wenig Knoten'); end; %Schlu\ss
talt=t;
t=sort(t);
if talt  ~= t
  disp('Knoten nicht aufsteigend geordnet, Umordnung vorgenommen!');
end; %if
talt=[];
%==============================================================================
%(2) Randknoten auf jeder Seite auf genau Ordnung St\"uck erg\"anzen oder
%reduzieren. Hinweis: Dieser Teil kann in gewissen Fllen unerwnscht sein.
%Man kann ihn problemlos weglassen (auskommentieren).
li=t(1);re=t(length(t));
links=find(t==li);
rechts=find(t==re);
t([links,rechts])=[];
rnd=ones(1,Ordnung);
t=[li*rnd,t,re*rnd];
%Randknoten angepa\ss t.
%==============================================================================
%(3) Vorbesetzungen
lenge=length(t); %lenge = Anzahl der Knoten
l=length(x);     %l = Anzahl der Funktionswerte
tt=x;
B=zeros(lenge-1,l);
n=zeros(1,lenge);
n(1)=1;
%Vorbesetzungen zu Ende
%==============================================================================
%tic;  %Die Stoppuhr wird gestartet.
%==============================================================================
%(4) Zuerst st\"uckweise konstante Splines, das ist der komplizierteste Teil.
for j=1:lenge-1
  n(j+1)=n(j);
  if t(j+1)>t(j)
    if j<lenge-Ordnung
      ax=find(x>=t(j) & x<t(j+1));  %Anzahl der x-Werte in [t_j,t_{j+1}[
    elseif j==lenge-Ordnung
      ax=find(x>=t(j) & x<=t(j+1)); %Anzahl der x-Werte in
                          %[t_{lenge-Ordnung},t_{lenge-Ordnung+1}]
                          %linkssseitige Stetigkeit am rechten Intervallpunkt
    end; % if
    n(j+1)=n(j)+length(ax);
    J=n(j):n(j+1)-1;
    if isempty(ax)==0  %nicht leer
      B(j,J)=ones(1,length(J));
    end; %if is
```

```
  end; %if t(...
end; %for j
%Stckweise konstante Splines B_1,...,B_{lenge-1} berechnet.
%==========================================================================
%(5) Globalschleife f\"ur alle Ordnungen
for k=2:Ordnung
  %========================================================================
  %(5a) Bestimmung der Omegas
  for j=1:lenge-k+1
    if t(j+k-1)>t(j)
      omega(j,:)=(tt-t(j))/(t(j+k-1)-t(j));
      else
      omega(j,:)=zeros(1,l);
    end; %if t(
  end; %for j
    %========================================================================
  %(5b) Bestimmen der Splines: Rekursionsformel
  for j=1:lenge-k
    B(j,:)=omega(j,:).*B(j,:)+(1-omega(j+1,:)).*B(j+1,:);
  end; %for j
%==========================================================================
end; %for k
%Globalschleife f\"ur alle Ordnungen zu Ende
%==========================================================================
%toc;  %Die Stoppuhr wird gestoppt.
%==========================================================================
%(6) Weglassen berflssiger Teile
%Da zuerst lenge - 1  stckweise konstante Splines hergestellt werden, werden
%insgesamt lenge-1 B-Splines berechnet. Tatschlich werden am Schlu aber nur
%lenge-Ordnung B-Splines der gegebenen Ordnung berechnet. Die auch berechneten
%B-Splines der Ordnungen 1,2,...,Ordnung - 1 werden jetzt weggelassen.
B(lenge-Ordnung+1:lenge-1,:)=[];
%==========================================================================
%(7) Schlus\ss
BB=B;
return
%==========================================================================
```

Programm L12.3. Berechnung eines Splines aus B-Splinekoeffizienten

```
%=========================================================================
%(0) Vorbemerkungen
%Berechnung von s(t)=\sum_ja_jB_{j,k}(t) bei
%gegebenen Knoten t_j, Koeffizienten a_j, gegebener Ordnung k
%und gegebenen Argumenten t in [t_ell,t_{ell+1}[. Der Aufruf hat die Form:

%         function s=splinesfct(Knoten,Koeff,Ordnung,Argumente);

%Liegen die Argumente nicht in einem Intervall [t_ell,t_{ell+1}[,
%so wird das Programm mit einer Fehlermeldung abgebrochen.
%Fr a=(0,0,...0,1,0,...,0,0), mit a_ell=1 ist s=B_{ell,k}.
function  s=splinesfct(Knoten,Koeff,Ordnung,Argumente);
%=========================================================================
%(1) Umbenennungen
t=Knoten;k=Ordnung;tt=Argumente;
%la=length(tt);lk=length(Knoten);
%=========================================================================
%(2) Intervall finden
%Zuerst wird geprft, in welchem Intervall die gegebenen Argumente tt liegen:
mi=min(tt);ma=max(tt);
if mi==t(length(t))
  ell=length(Knoten)-k-1;
  else
  K=find(t<=mi);
  ell=max(K);
  if t(ell+1)<=ma
    error('Argumente nicht in einem einzigen Intervall');
  end; %if
end; %if
%Prfung zu Ende, Intervall ist [t_ell,t_{ell+1}[.
%=========================================================================
%(3) Vorbesetzung
for j=ell-k+1:ell
  a(j,:)=Koeff(j)*ones(size(tt));
end; %for j
%=========================================================================
%(4) eigentliche Rekursion
for r=0:k-2
  for j=ell:-1:ell-k+r+2   %rckwrts!
    if t(j+k-r-1)>t(j)
```

```
      omega(j,:)=(tt-t(j))/(t(j+k-r-1)-t(j));
      else
      omega(j,:)=zeros(1,l);
    end; %if
    a(j,:)=(1-omega(j,:)).*a(j-1,:)+omega(j,:).*a(j,:);
  end; %for j
end; %for k
%==========================================================================
(5) Schlu\ss
s=a(ell,:);
return;
%==========================================================================
```

Programm L12.4. Berechnung von a' in $s' = \sum a'_j B_{j,k-1}$

```
%===========================================================================
%(0) Vorbemerkungen
%Ist s=sum_ja_jB_{j,k} ein Spline der Ordnung k, so werden die Koeffizienten
%a' von s'=sum_ja'_jB_{j,k-1} berechnet. Gegeben ist eine Knotenfolge t_j,
%nicht fallend (das wird vom Programm nicht geprft), mehrfache Knoten erlaubt,
%die die B-Splines B_{j,k} und B_{j,k-1} eindeutig definiert.
%Die Knoten drfen, damit der Splines s differenzierbar ist,
%hchstens (k-2)-fache Vielfachheit haben, im Programm
%wird jedoch nur t(j+k-1)-t(j)>0 fr alle j verlangt. Ist t(j+k-1)-t(j)=0,
%so ist B_{j,k-1}=0 und B_{j,k} ist mglichweise noch nicht einmal
%stetig bie t_j. Im Programm ist a'=aableitg. Geht die Summe fr s von 1 bis n,
%so mssen die Knoten von 1 bis n+k laufen. Das Ergebnis ist
%a'=(0,a_2',a_3',...,a_n'). Will man durch Wiederholung hhere Ableitungen
%ausrechnen, so ist jeweils, a'(1) und Knoten(1) wegzulassen. Die Koeffizienten
%werden nach der folgenden Formel berechnet:
%a_j':=(k-1)(a_j-a_{j-1})/(t_{j+k-1}-t_j) sofern t_{j+k-1}-t_j>0 und j>1.
%Im Falle t_{j+k-1}-t_j=0 ist a_j'=Inf (bei a_j-a_{j-1} nicht Null)
%oder a_j'=NaN (bei a_j-a_{j-1}=0). Will man nur die Ableitungen der B-Splines
%bestimmen, ist es einfacher, diese direkt zu bestimmen.
%Das Programm (Gerhard Opfer, 17. Juli 2000) hat die Form:
%
%          function aableitg=astrich(Knoten,Koeff,Ordnung);

function aableitg=astrich(Knoten,Koeff,Ordnung);
%===========================================================================
%(1) Umbenennungen
k=Ordnung;t=Knoten;a=Koeff;
%===========================================================================
%(2) Berechnung der Knotendifferenzen
n=length(a);
Knotenrechts=t(k+1:n+k-1);
Knotenlinks=t(2:n);
deltat=Knotenrechts-Knotenlinks;
%===========================================================================
%(3) Prfen auf nichtdifferenzierbare Stellen (Rand nicht mitgerechnet)
J=find(deltat==0);
if isempty(J)==0
  Kn=unique(Knotenrechts(J)); %(mehrfache Knoten werden eliminiert)
  disp(['Spline an den Knoten ',num2str(Kn),' nicht differenzierbar']);
  %An den Randknoten werden Unstetigkeiten nicht bemerkt.
end; %if
%===========================================================================
%(4) Berechnung der Koeffizientendifferenzen
Koeffrechts=a(2:n);
Koefflinks=a(1:n-1);
deltaa=Koeffrechts-Koefflinks;
%===========================================================================
%(5) Berechnung der Koeffizienten der Ableitung
aableitg(1)=0;
aableitg(2:n)=(k-1)*deltaa./deltat;
%===========================================================================
%(6) Schlu\ss
return;
%===========================================================================
```

Programm L12.5. Lösung von Aufgabe 23 auf Bogen 12

```
%===========================================================================
%(1) Vorbesetzungen
clear
Ordnung=4;
aa=0.05;b=0.4; %Intervall
tau=[aa,aa,0.06,0.06,0.075,0.075,0.09,0.09,0.13,0.13,0.21,0.21,b,b]; %14 Stck
taueinfach=[aa,0.06,0.075,0.09,0.13,0.21,b]; %Interpolationsknoten einfach
t=[aa aa tau b b]; %alle 18 Splineknoten
```

```
%B_1 ist auf [t_1,t_5[, B_2 auf [t_2,t_6[, ..., B_{14} auf [t_{14},t_{18}] definiert.
%auszurechnen B_1(tau_1),   B_2(tau_1), ... ,   B_14(tau_1)
%             B_1'(tau_2),  B_2'(tau_2), ... ,  B_14'(tau_2)  %tau_2=tau_1,...
%             .
%             .
%             .
%             B_1(tau_13),  B_2(tau_13), ... ,  B_14(tau_13)
%             B_1'(tau_14), B_2'(tau_14), ... , B_14'(tau_14)  %tau_14=tau_13
%             Im Programm heit diese die Matrix Bneu
%Bentigt werden die folgenden Unterprogramme:

                  %B=bsplinesfct(k,Knoten,Knoteneinfach);
                  %alle B-Splines der Ordnung k zur Knotenfolge Knoten,
                  %werden an den Stellen Knoteneinfach ausgerechnet.

                  %s=splinesfct(Knoten,a,k,x);
                  %Zu gegebenen Koeffizienten a, wird der Spline
                  %s = sum a_jB_j(x) der Ordnung k an den Stellen x ausgerechnet.
                  %Die Stellen x mssen in einem Intervall
                  %[Knoten(j),Knoten(j+1)[  liegen, sonst Fehlermeldung.
%============================================================================
%(2a) Berechnung der Matrix der Werte
B=bsplinesfct(Ordnung,t,taueinfach);      %B hat  die Gre 14 x 7.
%(2b) Berechnung der Matrix der Ableitungen
Bkm1=bsplinesfct(Ordnung-1,t,taueinfach); %B-Splines mit einer um 1 niedrigeren
                                          %Ordnung. Gibt Splines Bkm1 mit Nummern
                                          %von 1 bis 13. Das Progrann bsplinesfct
                                          %reduziert die Anzahl der Endknoten
                                          %immer auf Ordnung, in t sind dann
                                          %also nur noch 16 Knoten, und es gibt
                                          %daher 16 - (Ordnung - 1) = 13 Zeilen.
Bkm1=[zeros(1,length(taueinfach));Bkm1];  %vorne wird eine Zeile mit Nullen
                                          %davorgehngt, da die vierfachen Knoten
                                          %B_{1,3}=0 produzieren.
                                          %Dieser Spline wird aber vom Programm
                                          %bsplinesfct nicht ausgerechnet.
Bstrichh(1,:)=(Ordnung-1)*(-Bkm1(2,:)/(t(5)-t(2)));
for j=2:13
  Bstrichh(j,:)=(Ordnung-1)*(Bkm1(j,:)/(t(j+3)-t(j))-Bkm1(j+1,:)/(t(j+4)-t(j+1)));
end; %for j
Bstrichh(14,:)=(Ordnung-1)*(Bkm1(14,:)/(t(17)-t(14)));
Bneu=zeros(14,14); %Bneu wird aus B und Bstrichh zusammengesetzt
Bneu(1:2:13,:)=B';
Bneu(2:2:14,:)=Bstrichh';  %Die Matrix zerfllt in 4er Blcke
%============================================================================
%Alternativ zur Berechnung der Ableitung: %wird aber nicht gebraucht.
%I=eye(14);
%ast=[];
%for j=1:14
%  a=I(j,:);
%  as=astrich(t,a,4);
%  ast=[ast;as]; %Koeffizientenmatrix
%end; %for
%============================================================================
%(3) Jetzt Definition der rechten Seite
y=taueinfach.*sin(1./taueinfach);
ystrich=sin(1./taueinfach)-cos(1./taueinfach)./taueinfach;
yneu=zeros(1,14);
yneu(1:2:13)=y;
yneu(2:2:14)=ystrich;
yneu=yneu';
%============================================================================
%(4) L\"osung des Gleichungssystems, L\"osung ist a
a=Bneu\yneu;
%============================================================================
%(5) Zeichnen des Splines (geht nur Intervallweise)
for j=1:6
  x=linspace(taueinfach(j),taueinfach(j+1));
  x(length(x))=[]; %halboffenes Intervall
  s=splinesfct(t,a,Ordnung,x);
  plot(x,s,'b--'); if j==1, hold on, end; %if
end; %for
%============================================================================
%(6) Zeichnen der interpolierten Funktion t*sin(1/t) in [0.05,0.4]
tt=linspace(aa,b,201);
yy=tt.*sin(1./tt);
plot(tt,yy,'r',[aa b],[0 0],'b',taueinfach,zeros(size(taueinfach)),'go')
hold off
%print -dpsc intspl.ps  %postscript-Version des (farbigen) Bildes herstellen
return
```

```
%===========================================================================
% (7) Ergebnisse:
% rechte Seite        Koeffizienten   lfd. Nr.
%=========================================
%    0.0456              0.0456          1
%   -7.2487              0.0215          2
%   -0.0491             -0.0783          3
%    8.7579             -0.0053          4
%    0.0520              0.0966          5
%
%   -8.9063              0.0075          6
%   -0.0894             -0.0780          7
%   -2.2742             -0.1197          8
%    0.1283              0.1317          9
%   -0.2513              0.1216         10
%
%   -0.2097             -0.1768         11
%   -1.2345             -0.2879         12
%    0.2394              0.0746         13
%    2.6013              0.2394         14

Matrix 14 x 14
%===============================================================================================
%    1    0    0    0    0    0        0        0        0        0        0        0        0        0
% -300  300    0    0    0    0        0        0        0        0        0        0        0        0
%    0    0  0.6  0.4    0    0        0        0        0        0        0        0        0        0
%    0    0 -120  120    0    0        0        0        0        0        0        0        0        0
%    0    0    0    0  0.5  0.5        0        0        0        0        0        0        0        0
%    0    0    0    0 -100  100        0        0        0        0        0        0        0        0
%    0    0    0    0    0    0   0.7273   0.2727        0        0        0        0        0        0
%    0    0    0    0    0    0 -54.5455  54.5455        0        0        0        0        0        0
%    0    0    0    0    0    0        0        0   0.6667   0.3333        0        0        0        0
%    0    0    0    0    0    0        0        0 -25.0000  25.0000        0        0        0        0
%    0    0    0    0    0    0        0        0        0        0   0.7037   0.2963        0        0
%    0    0    0    0    0    0        0        0        0        0 -11.1111  11.1111        0        0
%    0    0    0    0    0    0        0        0        0        0        0        0        0   1.0000
%    0    0    0    0    0    0        0        0        0        0        0        0 -15.7895  15.7895
%===============================================================================================
```

Der mit diesen Daten berechnete Spline s stimmt mit dem in Abb. L12.1 angegebenen vollständig überein.

Aufgabe 24. Berechnen Sie (a) das begleitende Dreibein, (b) die Bogenlänge, (c) die Gesamtlänge, (d) die Krümmung und (e) die Torsion der folgenden Kurve (Spiralfeder, s. Abb. 12.1) $\mathbf{x} : [-a, a] \to \mathbb{R}^3$ mit $a = 10\pi$ und

$$\mathbf{x} := \frac{1}{a}\begin{cases} (a-t)\cos t, \\ (a-t)\sin t, \\ t. \end{cases} \tag{12.5}$$

Lösung. Wir berechnen zuerst

(L12.2)
$$\dot{\mathbf{x}} = \frac{1}{a}\begin{cases} -\cos t - (a-t)\sin t, \\ -\sin t + (a-t)\cos t, \\ 1, \end{cases} \quad \ddot{\mathbf{x}} = \frac{1}{a}\begin{cases} 2\sin t - (a-t)\cos t, \\ -2\cos t - (a-t)\sin t, \\ 0, \end{cases} \quad \dddot{\mathbf{x}} = \frac{1}{a}\begin{cases} 3\cos t + (a-t)\sin t, \\ 3\sin t - (a-t)\cos t, \\ 0. \end{cases}$$

Dann ist $||\dot{\mathbf{x}}||_2 = \frac{1}{a}\sqrt{2+(a-t)^2}$. Also ist (b) die Bogenlänge (wir benutzen für das Integral eine Tabelle)

$$\begin{aligned}\sigma(t) &= \int_{-a}^{t} ||\dot{\mathbf{x}}(\tau)||_2 \,\mathrm{d}\tau = \frac{\sqrt{2}}{a}\int_{-a}^{t}\sqrt{1+(\frac{\tau-a}{\sqrt{2}})^2}\,\mathrm{d}\tau = \frac{2}{a}\int_{-\sqrt{2}a}^{\frac{t-a}{\sqrt{2}}}\sqrt{1+x^2}\,\mathrm{d}x \\ &= \frac{1}{a}\Big\{x\sqrt{1+x^2} + \log(x+\sqrt{1+x^2})\Big\}\Big|_{-\sqrt{2}a}^{\frac{t-a}{\sqrt{2}}}.\end{aligned}$$

Damit ist (c) die Gesamtlänge

$$\sigma(a) = \sqrt{2}\sqrt{1+2a^2} - \frac{1}{a}\log(-\sqrt{2}a + \sqrt{1+2a^2}\,) \approx 62.9906 \text{ für } a = 10\pi.$$

Um das Dreibein und die weiteren Größen berechnen zu können, bestimmen wir mit $a - t =: T$

$$\dot{\mathbf{x}} \times \ddot{\mathbf{x}} = \frac{1}{a^2}(2\cos t + T\sin t, 2\sin t - T\cos t, 2 + T^2),$$

$$||\dot{\mathbf{x}} \times \ddot{\mathbf{x}}||_2 = \frac{1}{a^2}\sqrt{T^4 + 5T^2 + 8},$$
$$\det(\dot{\mathbf{x}}, \ddot{\mathbf{x}}, \dddot{\mathbf{x}}) = \frac{1}{a^2}(6 + T^2).$$

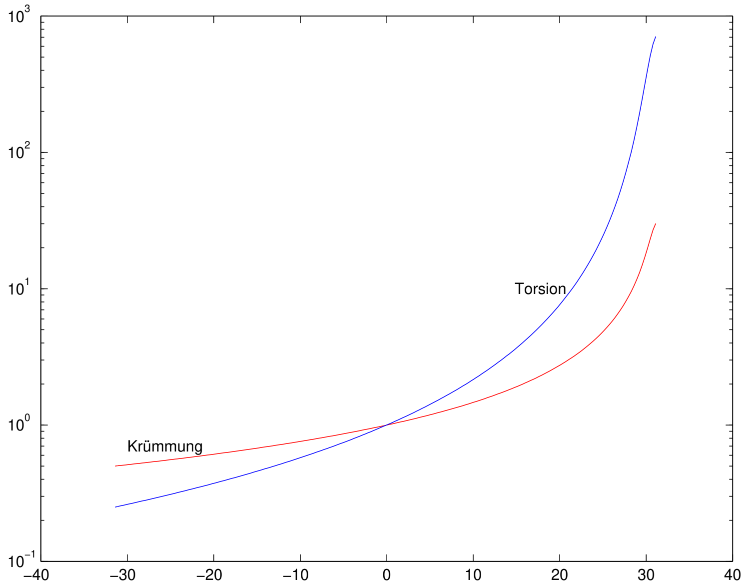

Abb. L12.6. Krümmung und Torsion der Spiralfeder (12.5)

Damit ist dann

$$\mathbf{t} = \frac{(-\cos t - T\sin t, -\sin t + T\cos t, 1)}{\sqrt{2 + T^2}},$$
$$\mathbf{b} = \frac{(2\cos t + T\sin t, 2\sin t - T\cos t, 2 + T^2)}{\sqrt{T^4 + 5T^2 + 8}},$$
$$\mathbf{m} = \frac{(4\sin t - 3T\cos t + T^2\sin t - T^3\cos t, -4\cos t - 3T\sin t - T^2\cos t - T^3\sin t, T)}{\sqrt{(2 + T^2)(T^4 + 5T^2 + 8)}},$$
$$\kappa(t) = \frac{a\sqrt{T^4 + 5T^2 + 8}}{(\sqrt{2 + T^2}\,)^3},$$
$$\tau(t) = \frac{a^2(6 + T^2)}{T^4 + 5T^2 + 8}.$$

Die Krümmung und die Torsion sind in Abhängigkeit von $t \in [-a, a]$ in Abb. L12.6 gezeichnet. Die drei Vektoren $\mathbf{t}, \mathbf{m}, \mathbf{b}$ definieren eine orthogonale Matrix. Das kann man prüfen, um Rechenfehler zu entdecken. Die Spirale hat in der Mitte bei $t = 0$ etwa den Radius Eins, also dort etwa die Länge 2π. Eine grobe Schätzung für die Gesamtlänge ist also $10 \cdot 2\pi \approx 62.8319$, eine bereits recht gute Schätzung, der relative Fehler ist nur 2.5‰. (Wer macht ein schöneres Promille-Zeichen?)

Inhalt der letzten drei Stunden: Kontrollpolygon $K_{\mathbf{a},\mathbf{t}}$ durch die Punkte $(t_j^*, a_j) \in \mathbb{R}^2$ zum Spline $s := \sum_j a_j B_{j,k}$ wird definiert, mit $\mathbf{a} = \{a_j\}$, $t_j^+ := (t_j + t_{j+1} + \cdots + t_{j+k-1})/(k-1)$. Satz: Der Abstand zwischen s und $K_{\mathbf{a},\mathbf{t}}$ hat die Größenordnung $O(h^2)$ mit $h = \max_j(t_{j+1} - t_j)$, Algorithmus zur Gitterverfeinerung angegeben, für den Fall daß die gegebene Knotenfolge $\mathbf{t}$ durch Hinzufügen eines einzigen Punktes $\hat{t}$ zum Gitter $\hat{\mathbf{t}}$ erweitert wird. Genauer: Angabe der Darstellung von $s = \sum \hat{a}_j B_{j,k}$ bezüglich $\hat{\mathbf{t}}$, wenn s zur Knotenfolge $\mathbf{t}$ gegeben ist, Bilder gezeigt.
Neues Tema: Wavelets (W. Cheney & W. Light [2000, Ch. 34]), zuerst Erinnerung an Fourieranaylse für periodische Funktionen f in $L_2[0, 1]$, also $f(x+n) = f(x), x \in [0, 1], n \in \mathbb{Z}$, Zerlegung von f in Spektralanteile

in der Form $f(x) \sim \sum_{j\in\mathbb{Z}} c_j \mathrm{e}^{2\pi \mathrm{i} jx}$, Erzeugung aller Funktionen $e_j(x) := \mathrm{e}^{2\pi\mathrm{i}jx}$ aus einer einzigen Funktion $e(x) := \mathrm{e}^{2\pi\mathrm{i}x}$ durch *Skalierung* in der Form $e_j(x) = e(jx)$, die Funktionen e_j bilden eine orthonormale Basis im Hilbertraum $L_2[0,1]$, Einführung eines *Urwavelets* ψ in $L_2(\mathbb{R})$ (Periodizität wird aufgegeben) und Bildung neuer Funktionen

$$\psi_{jk}(x) := 2^{j/2}\psi(2^j x - k), \quad j,k \in \mathbb{Z},\ x \in \mathbb{R} \tag{L12.3}$$

durch Skalierungen *und* Verschiebungen, Verwendung des Skalarprodukts $< u, v >:= \int_{\mathbb{R}} u(x)\overline{v(x)}\,\mathrm{d}x$, Beweis von $< f_{mj}, g_{mk} >=< f_{nj}, g_{nk} >$ für $f,g \in L_2(\mathbb{R})$, wobei f_{jk}, etc. genauso gebildet werden wie ψ_{jk}, also $||\psi||_2 = 1 \Rightarrow ||\psi_{jk}||_2 = 1$, die Leitidee ist, das Urwavelet ψ so zu finden, daß ψ_{jk} eine orthonormale Folge in $L_2(\mathbb{R})$ bildet, also $< \psi_{jm}, \psi_{kn} >= \delta_{jm,kn}$ also $\delta_{jm,kn} = 1$ nur für $(j,m) = (k,n)$ und Null in allen anderen Fällen, in diesem Fall heißt das Urwavelet ψ *orthogonal*. Beispiel: *Haarwavelet*

$$h(x) := \begin{cases} 1 & 0 \le x < \frac{1}{2}, \\ -1 & \frac{1}{2} \le x < 1, \\ 0 & \text{sonst.} \end{cases} \tag{L12.4}$$

Beweis, daß h ein orthogonales Urwavelet ist. Neue Idee zur Konstruktion von Urwavelets: Ausgehend von $\phi \in L_2(\mathbb{R})$ betrachtet man bei fester Skalierung (also festes j) die Mengen $\{\phi_{jk} : k \in \mathbb{Z}\}$ (Bildung wie in (L12.3)) und definiert die Räume

$$V_j := \overline{< \ldots, \phi_{j,-1}, \phi_{j,0}, \phi_{j,1}, \ldots >}, \tag{L12.5}$$

also den Abschluß der linearen Hülle dieser Elemente. Ist $\phi \in V_1$, so gibt es also $\mathbf{a} = \{a_j\}_{j\in\mathbb{Z}} \in \ell_2$ mit $\phi(t) = \sum_{j\in\mathbb{Z}} a_j \phi(2t - j)$. Diese Gleichung heißt *Skalengleichung*, Konstruktion von orthogonalen Wavelets mit Hilfe der Skalengleichung, weitere Eigenschaften für den Fall, daß $\bigcup V_j$ dicht in $L_2(\mathbb{R})$ liegt.

Anhang: Der Satz von Schoenberg und Whitney

Ausgehend von einer B-Spline-Basis $B_{1,k}, B_{2,k}, \ldots, B_{n,k}$, bestehend aus B-Splines der Ordnung k gibt dieser Satz Auskunft über die mögliche Lage von *Interpolationsknoten*,

$$\boldsymbol{\tau} := \{\tau_1, \tau_2, \ldots, \tau_n\}, \quad \tau_j \le \tau_{j+1}, \quad j = 1, 2, \ldots, n-1, \tag{L12.6}$$

die zu einem eindeutig lösbaren Interpolationsproblem führen. Wir folgen bei leichter Abänderung der (Spline-)Knotennumerierung Nürnberger [1989, S. 108–110] und Schumaker [1981, S. 171]. Die Splines seien durch die *Splineknoten*

$$\mathbf{t} := \{t_1, t_2, \ldots, t_{n+k}\}, \quad t_j \le t_{j+1}, \quad j = 1, 2, \ldots, n-1, \tag{L12.7}$$

eindeutig bestimmt. Wir definieren den Raum

$$S_n := \langle B_{1,k}, B_{2,k}, \ldots, B_{n,k} \rangle \tag{L12.8}$$

und setzen

$$t_j < t_{j+k}, \quad j = 1, 2, \ldots, n \tag{L12.9}$$

voraus. Damit ist $\dim S_n = n$. Sowohl die Splineknoten als auch die Interpolationsknoten dürfen mehrfach vorkommen. Wir definieren einen Vektor $\mathbf{d} := (d_1, d_2, \ldots, d_n)$ folgendermaßen: Ist $\tau_{\kappa-1} < \tau_\kappa = \tau_{\kappa+1} = \cdots = \tau_{\kappa+l} < \tau_{\kappa+l+1}$, ist also τ_κ ein $(l+1)$-facher Interpolationsknoten, $l \ge 0$, so ist $d_{\kappa+r} := r, r = 0, 1, \ldots, l$. Sind alle Interpolationsknoten $\boldsymbol{\tau}$ einfach, so ist $\mathbf{d}$ der Nullvektor, sind alle Interpolationsknoten $\boldsymbol{\tau}$ doppelt, so ist $\mathbf{d} = (0, 1, 0, 1, \ldots, 0, 1)$. Bei vorgegebenen Zahlen $s_\kappa, \kappa = 1, 2, \ldots, n$ lautet das Interpolationsproblem

$$s^{(d_\kappa)}(\tau_\kappa) = s_\kappa, \quad \kappa = 1, 2, \ldots, n, \quad s \in S_n. \tag{L12.10}$$

Ist also τ_κ ein $(l+1)$-facher Interpolationsknoten und $\tau_{\kappa-1} < \tau_\kappa$, so wollen wir $s(\tau_\kappa) = s_\kappa$, $s'(\tau_{\kappa+1}) = s_{\kappa+1}$, $\ldots$, $s^{(l)}(\tau_{\kappa+l}) = s_{\kappa+l}$ erreichen, wobei $\tau_\kappa = \tau_{\kappa+1} = \cdots = \tau_{\kappa+l} < \tau_{\kappa+l+1}$. Ist g_j die Glattheit von s am

Splineknoten t_j (so oft kann man s dort stetig differenzieren), und ist $\#t_j$ die Vielfachheit des Splineknotens t_j, so haben wir die bekannte Formel ($g_j = -1$ bedeutet Unstetigkeit)

(L10.1) Ordnung $k = g_j + 1 + \#t_j$ an jedem Splineknoten t_j.

Ist t kein Splineknoten, so ist $s^{(j)}(t) = 0$ für alle $j \geq k$. Die Vielfachheit $\#\tau_\kappa$ der Interpolationsknoten ist daher eingeschränkt:

$$\#\tau_\kappa \leq \begin{cases} k - \#t_j & \text{für } \tau_\kappa \in \mathbf{t}, \tau_\kappa = t_j, \\ k-1 & \text{für } \tau_\kappa \notin \mathbf{t}. \end{cases} \tag{L12.11}$$

Diese Bedingung kann auch als Einschränkung an die Splineknoten verstanden werden. Will man also an einem Splineknoten $t_j = \tau_\kappa$ interpolieren, so muß (L12.9) zu

$$t_j < t_{j+k-\#\tau_\kappa} \tag{L12.12}$$

abgeschwächt werden. Wir benutzen, daß S_n ein schwacher Haarscher Raum ist, d. h. jedes $s \in S_n$, das auf $]t_1, t_{n+k}[$ nur endlich viele Nullstellen hat, hat dort höchstens $n-1$ Nullstellen. Mehrfache Nullstellen werden entsprechend auch mehrfach gezählt.

Satz 12.1. (Schoenberg & Whitney, 1953) Das Interpolationsproblem (L12.10) ist unter den Einschränkungen (L12.11) genau dann lösbar, wenn

$$t_j < \tau_j < t_{j+k} \text{ für alle } j = 1, 2, \ldots, n. \tag{L12.13}$$

Beweis: Entscheidend ist die Nichtsingularität der Matrix

$$\mathbf{M} := (M_{\kappa,\ell}) := B_{\ell,k}^{(d_\kappa)}(\tau_\kappa), \quad \kappa, \ell = 1, 2, \ldots, n. \tag{12.3}$$

Für alle Ableitungen der B-Splines gilt $B_{j,k}^{(r)}(t) = 0$ für $t \notin [t_j, t_{j+k}[$, da die Splines selbst dort verschwinden. (a) Sei die Bedingung (L12.13) nicht erfüllt. Es existiert also ein $1 \leq j_0 \leq n$ mit $\tau_{j_0} \notin]t_{j_0}, t_{j_0+k}[$. Also ist entweder (a1) $\tau_{j_0} \notin]t_j, t_{j+k}[$ für alle $j \geq j_0$ oder (a2) $\tau_{j_0} \notin]t_j, t_{j+k}[$ für alle $j \leq j_0$. Im Falle (a1) ist $B_{\ell,k}^{(d_\kappa)}(\tau_\kappa) = 0$, $\kappa \leq j_0 \leq \ell$, im Falle (a2) ist $B_{\ell,k}^{(d_\kappa)}(\tau_\kappa) = 0$, $\ell \leq j_0 \leq \kappa$. Die Matrix $\mathbf{M}$ aus (12.3) hat also in keinem Fall den vollen Rang, das Interpolationsproblem ist also i. a. nicht lösbar. (b) Sei die Bedingung (L12.13) erfüllt. Wir nehmen an, daß (L12.10) nicht erfüllt werden kann, das homogene System $\mathbf{Ma} = \mathbf{0}$ hat also eine Lösung $\mathbf{a} := (a_1, a_2, \ldots, a_n)^{\mathrm{T}} \neq \mathbf{0}$. Mit diesem $\mathbf{a}$ definieren wir den Spline $s := \sum_{\ell=1}^n a_\ell B_{\ell,k}$. Sei ℓ_0 der kleinste Index mit $a_{\ell_0} \neq 0$, und sei t_{ℓ_1+k} der Splineknoten mit der kleinsten Nummer mit $\ell_1 \geq \ell_0$, so daß s auf $[t_{\ell_1+k}, \hat{t}[$ mit $\hat{t} > t_{\ell_1+k}$ verschwindet. Dann hat s auf $]t_{\ell_0}, t_{\ell_1+k}[$ höchstens endlich viele Nullstellen und nach (L12.13) verschwindet s an den $\ell_1 - \ell_0 + 1$ Stellen $\tau_{\ell_0}, \tau_{\ell_0+1}, \ldots, \tau_{\ell_1}$. Der $\ell_1 - \ell_0 + 1$ dimensionale Raum $\langle B_{\ell_0,k}, B_{\ell_0+1,k}, B_{\ell_1,k}\rangle$ enthält s auf $]t_{\ell_0}, t_{\ell_1+k}[$, d. h. s kann nur $\ell_1 - \ell_0$Nullstellen haben, ein Widerspruch. □

Bemerkung L12.7. Interpoliert man einen Wert oder eine Ableitung an einem Splineknoten, so kann man diese Größen auch als rechsseitige Grenzwerte auffassen, da wir die B-Splines immer als rechtseitig stetig definiert haben, d. h. die obige Theorie läßt sich ohne Änderung auf diesen Fall übertragen, wobei dann allerdings die erste Bedingung (L12.13) von Schoenberg & Whitney zu

$$t_1 \leq \tau_1 < t_{k+1} \tag{L12.13$'$}$$

erweitert werden muß. Einzelheiten bei Schumaker [1981, S. 171]. Das ist insbesondere anzuwenden, wenn der erste Splineknoten k-fach ist und dort interpoliert werden soll.

Literatur

W. Cheney & W. Light, A course in approximation theory, Brooks/Cole, Pacific Grove, 2000, 359 S.

G. Farin, Kurven und Flächen im Computer Aided Geometric Design, 2. Aufl., Vieweg, Braunschweig, 1994, 365 S.

G. Nürnberger, Approximation by spline functions, Springer, Berlin, 1989, 243 S.

G. Opfer: Numerische Mathematik für Anfänger, 2. Aufl., Vieweg, Braunschweig, 1994, 295 S.

L. L. Schumaker, Spline functions, basic theory, Wiley, New York, 1981, 553 S.

(version of 21. 8. 2000 at 1130 minutes since midnight)