

509 | Juli 1990

SCHRIFTENREIHE SCHIFFBAU

Axel Busch

Vergleich verschiedener linearer Algorithmen für Spline-Kurven

TUHH

Technische Universität Hamburg-Harburg

Vergleich verschiedener linearer Algorithmen für Spline-Kurven

Axel Busch, Hamburg, Technische Universität Hamburg-Harburg, 1990

ISBN: 3-89220-509-4

© Technische Universität Hamburg-Harburg
Schriftenreihe Schiffbau
Schwarzenbergstraße 95c
D-21073 Hamburg

<http://www.tuhh.de/vss>

Institut für Schiffbau der Universität Hamburg
Bericht Nr. 509

Vergleich verschiedener linearer Algorithmen für Spline-Kurven

von

Dipl.-Ing. Axel Busch

Juli 1990

Zusammenfassung

Die vorliegende Studienarbeit soll durch einen praktischen Vergleich die Brauchbarkeit verschiedener Spline-Algorithmen untersuchen. Alle hier behandelten Splines sind von nur einem Parameter abhängig, d.h. es sind Kurven, die aber teilweise auch frei im Raum liegen können.

Als Maßstab für die Qualität eines Splines werden die folgenden Kriterien berücksichtigt:

- Die entstehende Kurve soll möglichst glatt sein
- Der Rechenaufwand soll gering sein
- Der Bedienungsaufwand soll klein sein, d.h. der Spline soll die vorgegebenen Punkte verwirklichen und keine weiteren steuernden Parameter benötigen. Die Flexibilität in speziellen Anwendungen wird nicht untersucht.

Je nachdem welche der genannten Kriterien man betont, sind unterschiedliche Spline-Algorithmen zu empfehlen. Ziel ist es, den Spline-Algorithmus zu finden, der alle anfallenden "Punkthaufen" zuverlässig, gut und schnell durch einen interpolierenden Spline verbindet.

Spline-Algorithmen, die auf iterativen oder anderen nichtlinearen Verfahren beruhen, werden nicht in diese Betrachtung einbezogen.

Inhaltsverzeichnis

1	Einleitung	1
2	Die mathematische Behandlung	2
2.1	Mathematische Modelle der Splines	2
2.2	Allgemeine Definitionen	3
3	Verschiedene Spline-Algorithmen	4
3.1	Polynom $(n - 1)$ -ten Grades	4
3.2	Kubischer Spline mit verschiedenen Randbedingungen	4
3.3	Hermitesche Spline-Kurve 3. Grades	10
3.4	Hermitescher Spline 5.Grades	11
3.5	Exponentieller Spline	13
3.6	Rationaler Spline	17
3.7	Kubischer Spline mit mitgeführten Koordinaten	22
3.8	Parametrisch kubischer Spline	25
3.9	Kubische Bézier-Kurve	29
3.10	Verbesserter Bézier-Spline	33
3.11	Kubischer Bézier-Spline	36
3.12	B-Spline-Kurve	38
4	Periodische und zyklische Splines	41
4.1	Periodische Splines	41
4.2	Zyklische Splines	42
5	Bewertung der Spline-Algorithmen	43
5.1	Module zum parametrisch kubischen Spline	49
6	Programme	50
6.1	Programmaufruf am Institut für Schiffbau	50
6.2	Übersicht über die verwendeten Routinen	50

6.3	FORTRAN 77 - Source Codes	52
	POLY.FOR	52
	CUB1.FOR	52
	PLCUB.FOR	53
	CUBI.FOR	54
	PLCUBI.FOR	55
	DEFY1.FOR	56
	PLQDEF.FOR	57
	PLSPLEXP.FOR	57
	RAT.FOR	58
	PLRAT.FOR	59
	PARA.FOR	60
	CUBVAR.FOR	61
	PLCUBV.FOR	63
	BEZIER.FOR	63
	VBEZ.FOR	64
	PLBEZ.FOR	66
	BS3.FOR	66
	PLBS3.FOR	67
	BS4.FOR	68
	PLBS4.FOR	69
	DASHAB.FOR	70
	CUBPAR.FOR	82
	CUBPIT.FOR	83
	PLCUBP.FOR	85
	FCUBP.FOR	86
	TCUBP.FOR	87
	DCUBP.FOR	88

1 Einleitung

Diese Arbeit entstand 1986 als kleine Studienarbeit mit einem Vortrag am Institut für Schiffbau in Hamburg. Aufgrund der anhaltenden Nachfrage wurde der Vortrag im Rahmen des schiffbaulichen Kolloquiums im Juli 1990 wiederholt und in einigen Punkten aktualisiert. So wurden die Kapitel "Kubischer Bézier-Spline" und "B-Spline-Kurve" eingefügt. Die "Bewertung der Spline-Algorithmen" bekam ein eigenes Kapitel und es werden die inzwischen existierenden Routinen für den kubisch-parametrischen Spline vorgestellt.

Die Arbeit soll die Brauchbarkeit verschiedener Spline-Algorithmen in der täglichen Anwendung untersuchen. Unter einem "Spline" ist hier eine interpolierende, mindestens zweimal stetig ableitbare Funktion zu verstehen. Nicht interpolierende Kurven werden zur Abgrenzung als "Spline-Kurve" bezeichnet.

Um einen direkten Vergleich zu ermöglichen, habe ich die Spline-Algorithmen programmiert und mit jedem von ihnen dasselbe "Testbild" straken lassen. Dadurch kann man die unterschiedlichen Ausführungen direkt vergleichen und beurteilen. Teilweise wurden mehrere Straks in eine Abbildung geplottet.

Das Testbild (Abb. 1) beinhaltet mehrere Kurvenscharen, die im Schiffbau vorkommen. Zunächst rechts oben einen (ausgedünnten) Spantriß (1), darunter eine Hebelarmkurve (2), darunter dann Wasserlinien (3). Links oben ist ein Diagramm (4), wie es bei der Auswertung von Messungen entstehen kann. Die Punkte dieses Diagramms wurden bewußt so gelegt, daß sie schwer zu straken sind. Diese Schwierigkeit besteht vor allem in der Aueinanderfolge von Abschnitten mit kleiner und großer Sekantensteigung. Im Spantriß (1) müssen sowohl horizontale als auch vertikale Strecken wiedergegeben werden, auch der Kimmradius bereitet Schwierigkeiten. Die Wasserlinien (3) haben viele Punkte, nur die Hebelarmkurve (4) ist wirklich einfach und sollte von jedem Spline-Algorithmen gut bewältigt werden.

Neben diesem Testbild ist teilweise auch noch ein "großes Diagramm", der "Schweineschwanz" und die "Kreise" abgebildet. Diese Figuren werden aber nur gezeigt, wenn dadurch weitere interessante Eigenschaften der Spline-Algorithmen zu erkennen sind.

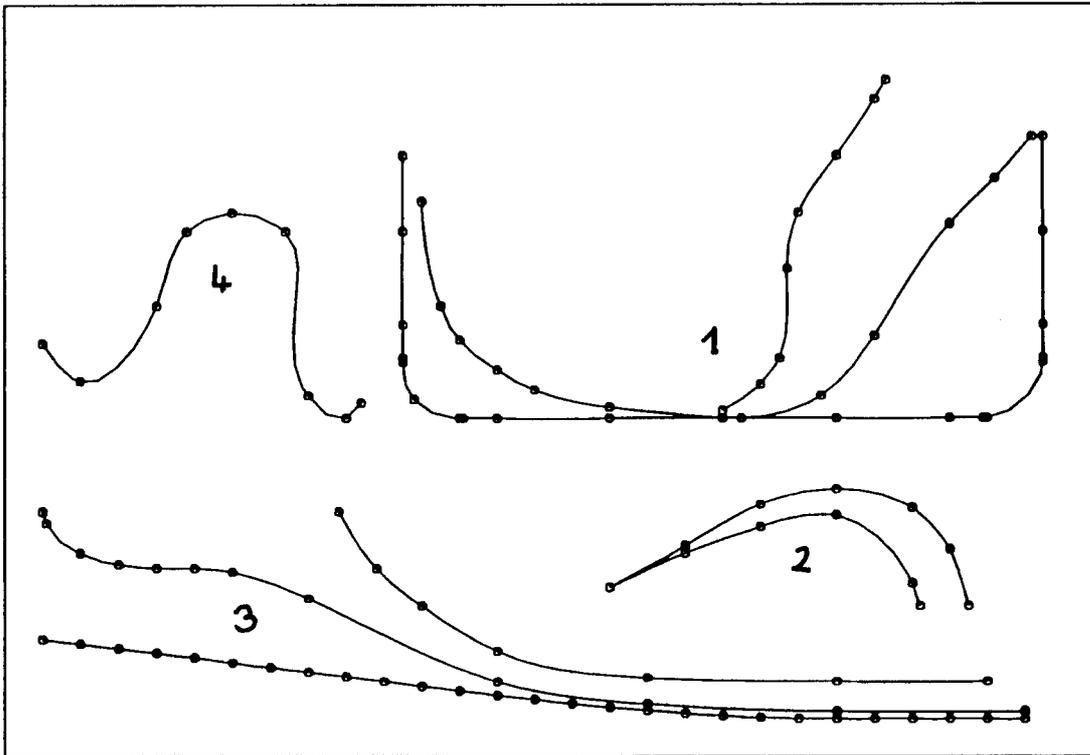


Abbildung 1: Testbild

2 Die mathematische Behandlung

2.1 Mathematische Modelle der Splines

Beim Berechnen von Splines werden eine Anzahl von n Punkten durch eine mathematisch beschreibbare Kurve verbunden. Die Koeffizienten dieser Funktion werden den gegebenen Punkten und den jeweiligen Randbedingungen angepaßt [1, 2]. Um diese Koeffizienten zu berechnen, gibt es zwei grundsätzlich unterschiedliche Ansätze:

- Zum einen kann man den Strak in der Art einer Übertragungsmatrix behandeln, d.h. der Zustand am rechten Rand eines Intervalls wird durch den Zustand am linken Rand beschrieben. Anschließend werden die Matrizen aufmultipliziert, die Zwischenpunkte werden somit kondensiert und am Ende wird ein sehr kleines Gleichungssystem (2×2) gelöst. Dieser Algorithmus ist übersichtlich, er eignet sich für die Handrechnung, da man nur mit wenigen Variablen gleichzeitig arbeiten muß. Bei Splines über viele Stützstellen treten aber zunehmend numerische Ungenauigkeiten auf.
- Die zweite Möglichkeit besteht darin, ein Gleichungssystem aufzustellen, in dem gleichzeitig alle Koeffizienten berechnet werden. Dieses

Verfahren bietet sich insbesondere dadurch an, daß immer (bei nicht-periodischen Splines) tridiagonale Matrizen entstehen, die sehr effizient gelöst werden können. Auch ist der Lösungsalgorithmus insgesamt viel "rechnerfreundlicher".

Bei den folgenden Berechnungen habe ich daher immer den Weg über das Gleichungssystem gewählt, selbstverständlich liefert der erste Lösungsweg die gleichen Kurven.

2.2 Allgemeine Definitionen

Für die Indizes der Punkte und der Intervalle gilt die in Abb. 2 gezeigte Numerierung. Ein Spline über n Punkte hat also $(n-1)$ Intervalle mit $(n-1)$ Funktionsgleichungen $f_i(x)$; mit $1 \leq i \leq n-1$ und $x_i \leq x \leq x_{i+1}$. Die Länge eines Intervalls wird mit $\Delta x_i = x_{i+1} - x_i$ bezeichnet, bzw. $\Delta y_i = y_{i+1} - y_i$.

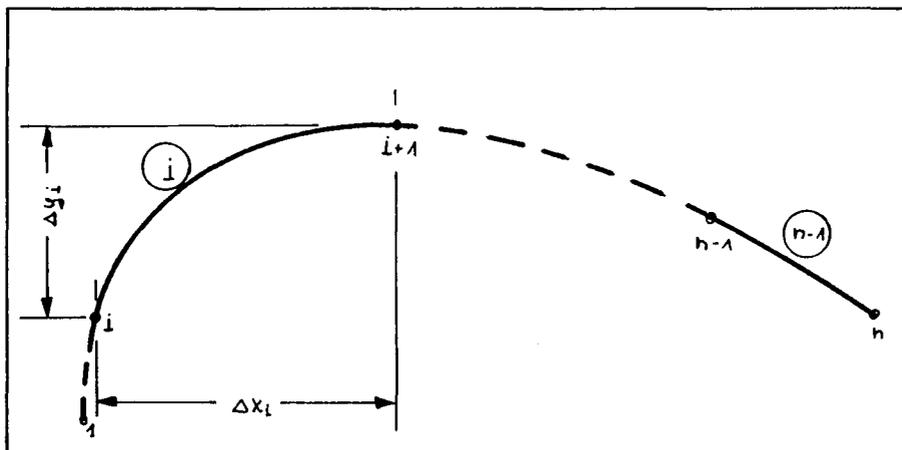


Abbildung 2: Die Indizes der Splines

Soweit nicht weiter angegeben wird immer $x_1 < x_2 < x_3 < \dots < x_n$ angenommen, d.h. die x-Werte sind streng monoton steigend.

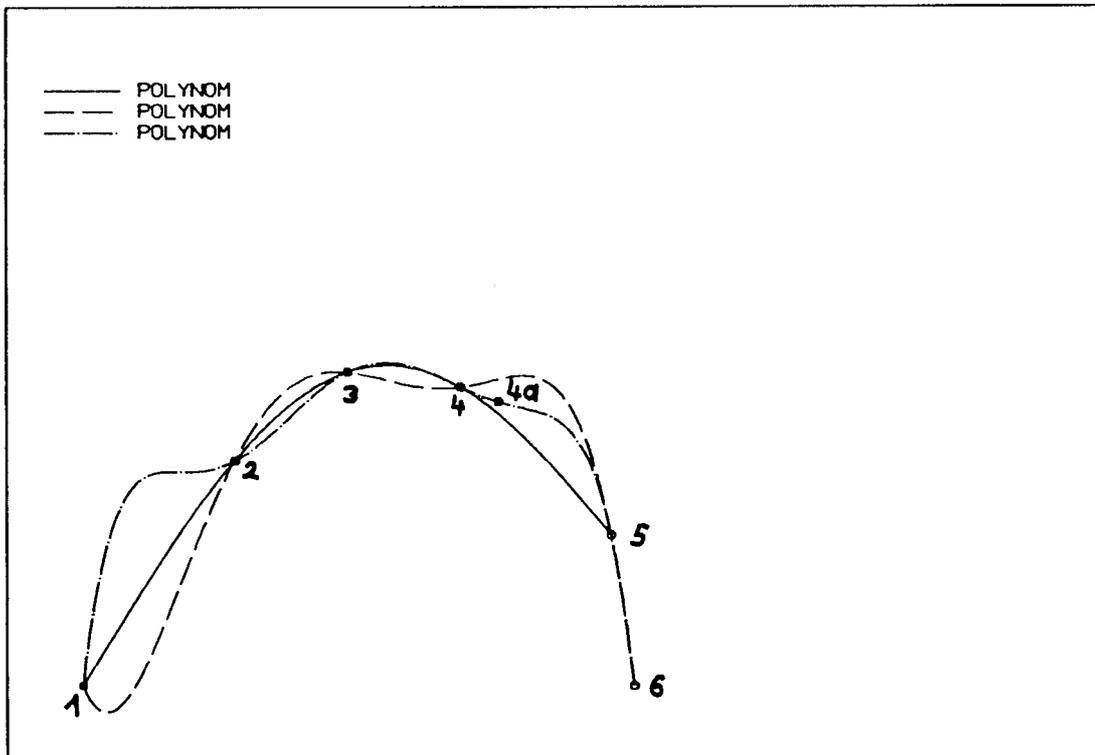


Abbildung 3: Polynom

die erste und zweite Ableitung am Berührungspunkt benachbarter Intervalle gleichsetzt.

Bei den kubischen Splines wird für jedes Intervall ein Ansatz der Form

$$f_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i \quad (3)$$

für das i -te Intervall benutzt. Es treten in dieser Funktion vier Koeffizienten a_i, b_i, c_i und d_i pro Intervall auf. Man kann diese Gleichung so umformen, daß die bisherigen Koeffizienten durch vier neue ersetzt werden. Dafür bieten sich die Funktionswerte y_i und y_{i+1} , sowie die ersten Ableitungen y'_i und y'_{i+1} oder die zweiten Ableitungen [4] y''_i und y''_{i+1} an. In jedem Fall ergibt sich unabhängig von der mathematischen Behandlung die gleiche Kurve.

Bei Auflösung nach der ersten Ableitung ergibt sich:

$$a_i = \frac{1}{\Delta x_i^2} \left(-2 \cdot \frac{\Delta y_i}{\Delta x_i} + y'_i + y'_{i+1} \right) \quad (4)$$

$$b_i = \frac{1}{\Delta x_i} \left(3 \cdot \frac{\Delta y_i}{\Delta x_i} - 2 \cdot y'_i - y'_{i+1} \right) \quad (5)$$

$$c_i = y'_i \quad (6)$$

$$d_i = y_i \quad (7)$$

Als Kopplung der einzelnen Intervalle stehen einem die Gleichheit der ersten (Gl. 8) und der zweiten Ableitung (Gl. 9) zur Verfügung.

$$f'_{i-1}(x_i) = f'_i(x_i) = y'_i \quad (8)$$

$$f''_{i-1}(x_i) = f''_i(x_i) = y''_i \quad (9)$$

$$f''_{i-1}(x_i) = \frac{2}{\Delta x_{i-1}} \left(-3 \frac{\Delta y_{i-1}}{\Delta x_{i-1}} + y'_{i-1} + 2y'_i \right) \quad (10)$$

$$f''_i(x_i) = \frac{2}{\Delta x_i} \left(3 \frac{\Delta y_i}{\Delta x_i} - 2y'_i - y'_{i+1} \right) \quad (11)$$

$$\frac{1}{\Delta x_{i-1}} y'_{i-1} + 2 \left(\frac{1}{\Delta x_{i-1}} + \frac{1}{\Delta x_i} \right) y'_i + \frac{1}{\Delta x_i} y'_{i+1} = 3 \frac{\Delta y_{i-1}}{\Delta x_{i-1}^2} + 3 \frac{\Delta y_i}{\Delta x_i^2} \quad (12)$$

Der gesamte Spline wird damit durch ein lineares Gleichungssystem (Gl. 13) mit einer symmetrischen, tridiagonalen Matrix repräsentiert.

$$\left[\begin{array}{ccccccc} 2 \left(\frac{1}{\Delta x_1} + \frac{1}{\Delta x_2} \right) + \frac{1}{\Delta x_2} & & & & & & \\ \frac{1}{\Delta x_2} & +2 \left(\frac{1}{\Delta x_2} + \frac{1}{\Delta x_3} \right) + \frac{1}{\Delta x_3} & & & & & \\ & \frac{1}{\Delta x_3} & +2 \left(\frac{1}{\Delta x_3} + \frac{1}{\Delta x_4} \right) + \frac{1}{\Delta x_4} & & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \frac{1}{\Delta x_{n-2}} & +2 \left(\frac{1}{\Delta x_{n-2}} + \frac{1}{\Delta x_{n-1}} \right) & & \end{array} \right]$$

$$\left\{ \begin{array}{c} y'_2 \\ y'_3 \\ y'_4 \\ \vdots \\ y'_{n-1} \end{array} \right\} = \left\{ \begin{array}{c} 3 \frac{\Delta y_1}{\Delta x_1^2} + 3 \frac{\Delta y_2}{\Delta x_2^2} - \frac{1}{\Delta x_1} y'_1 \\ 3 \frac{\Delta y_2}{\Delta x_2^2} + 3 \frac{\Delta y_3}{\Delta x_3^2} \\ 3 \frac{\Delta y_3}{\Delta x_3^2} + 3 \frac{\Delta y_4}{\Delta x_4^2} \\ \vdots \\ 3 \frac{\Delta y_{n-2}}{\Delta x_{n-2}^2} + 3 \frac{\Delta y_{n-1}}{\Delta x_{n-1}^2} - \frac{1}{\Delta x_{n-1}} y'_n \end{array} \right\} \quad (13)$$

Die Gleichung (Gl. 13) zeigt das Gleichungssystem bei gegebenen ersten Ableitungen am Anfang y'_1 und am Ende y'_n des Splines.

Wird als Randbedingung die Krümmung y''_1 und y''_n vorgegeben, so lautet die erste und die letzte Gleichung (vgl. Gl. 62) :

$$\frac{2}{\Delta x_1} y'_1 + \frac{1}{\Delta x_1} y'_2 = 3 \frac{\Delta y_1}{\Delta x_1^2} - \frac{y''_1}{2} \quad (14)$$

$$\frac{1}{\Delta x_{n-1}} y'_{n-1} + \frac{2}{\Delta x_{n-1}} y'_n = 3 \frac{\Delta y_{n-1}}{\Delta x_{n-1}^2} + \frac{y''_n}{2} \quad (15)$$

Wird als Randbedingung eine konstante Krümmung angenommen (Kreis) $y''_1 = y''_2$ und $y''_{n-1} = y''_n$, so ergibt sich:

$$\frac{1}{\Delta x_1} y'_1 + \frac{1}{\Delta x_1} y'_2 = 2 \frac{\Delta y_1}{\Delta x_1} \quad (16)$$

$$\frac{1}{\Delta x_{n-1}} y'_{n-1} + \frac{1}{\Delta x_{n-1}} y'_n = 2 \frac{\Delta y_{n-1}}{\Delta x_{n-1}} \quad (17)$$

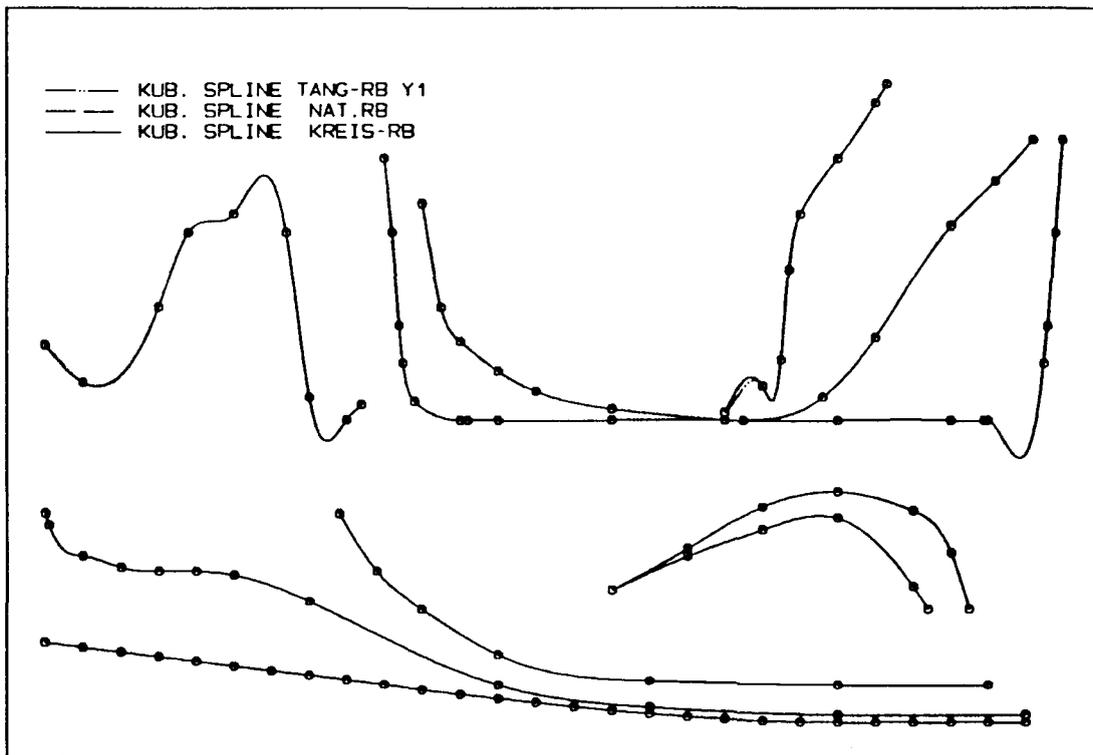


Abbildung 4: Kubische Splines, Testbild

In Abb. 4 und Abb. 5 sind verschiedene kubische Splines zu sehen. Sie unterscheiden sich nur durch ihre Randbedingungen.

- Kreis Randbedingung : $y_1'' = y_2''$; $y_{n-1}'' = y_n''$
 natürliche Randbedingung : $y_1'' = y_n'' = 0$
 Tangenten Randbedingung : $y_1' = y_n' = 0$
 Tangenten Randbedingung Y1 : $y_1' = \frac{\Delta y_1}{\Delta x_1}$; $y_n' = \frac{\Delta y_n}{\Delta x_n}$
 : (Differenzenquotient)

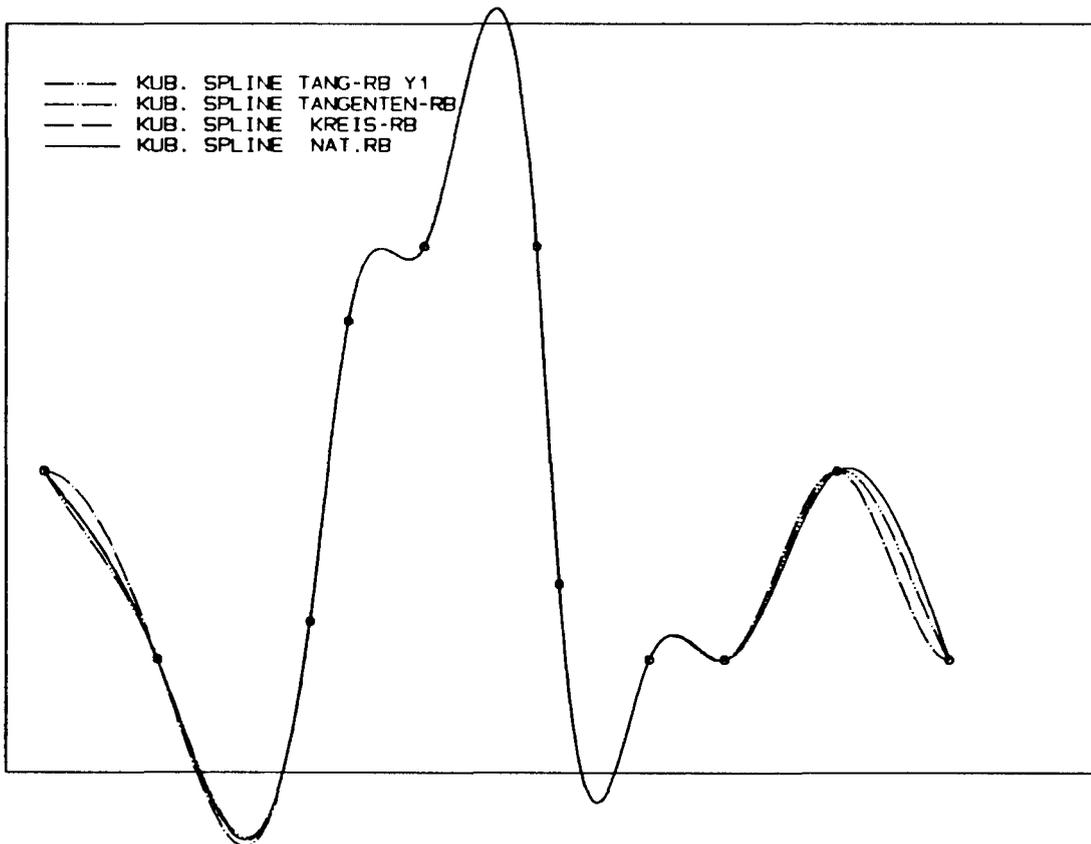


Abbildung 5: Kubische Splines, Diagramm

Es ist deutlich zu sehen, daß sich die Randbedingungen nur auf die Randintervalle auswirken. Auf das Innere des Splines haben sie keinen Einfluß.

Allen kubischen Splines ist gemeinsam, daß sie in den Bereichen mit geringen Steigungsänderungen gut straken, aber immer, wenn Intervalle mit stark unterschiedlichen Sekantensteigungen aneinanderstoßen, schwingt die Kurve über. Dabei ist das Intervall mit kleiner Abszissendifferenz Δx stets steifer als das Intervall mit großem Δx . Besonders beim Kimmradius Abb. 4 des Spantrisses und beim Diagramm kann man dies sehen.

Ein weiterer Mangel ist, daß senkrechte Linien nicht darstellbar sind. Aus diesem Grund wurde der Spantriß oben aufgebogen. Der Fehler läßt sich manchmal, wie in Abb. 6 gezeigt, durch Drehen des Koordinatensystems umgehen.

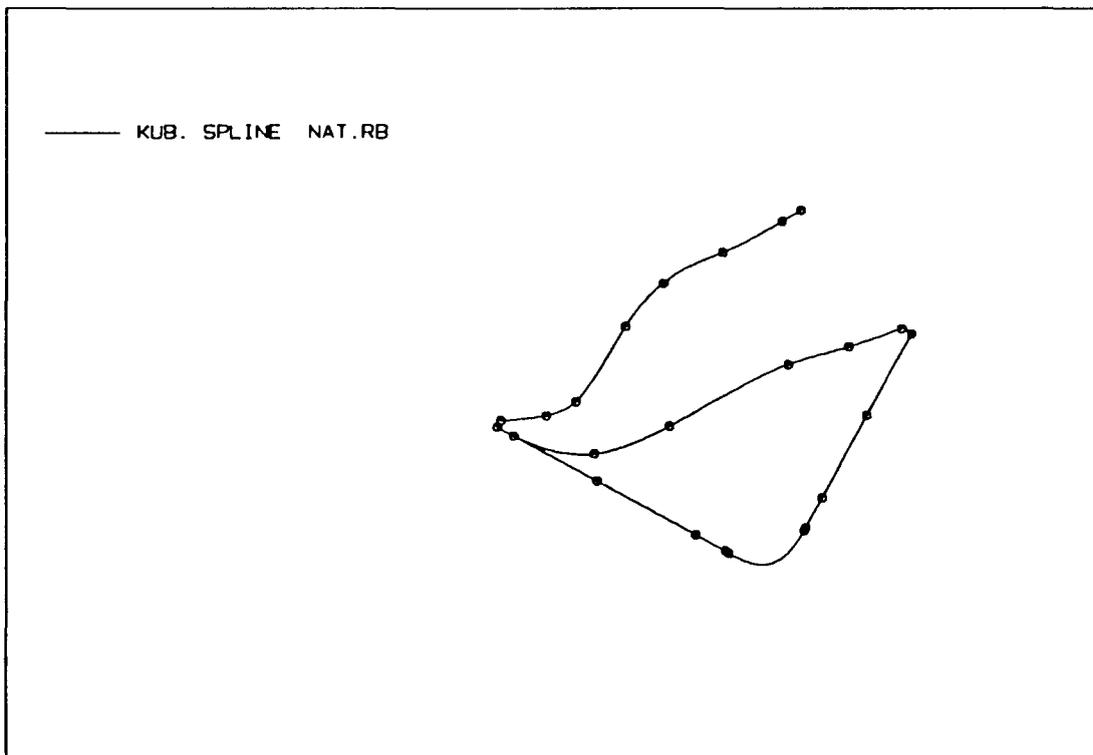


Abbildung 6: Kubischer Spline mit gedrehtem Spantrieb

3.3 Hermitesche Spline-Kurve 3. Grades

Bei der hermiteschen Spline-Kurve wird ebenfalls für die einzelnen Intervalle ein kubischer Ansatz gemacht. Allerdings wird nun nicht mehr die erste Ableitung gekoppelt. Sondern sie wird in einem Vorlauf berechnet, indem durch je drei benachbarte Punkte ein quadratisches Polynom gelegt wird, das dann die Steigung im mittleren Punkt bestimmt (Gl. 18). Die 2. Ableitung ist damit nicht mehr gekoppelt, da keine freien Variablen mehr zur Verfügung stehen.

$$y'_i = \frac{\Delta x_i \frac{\Delta y_{i-1}}{\Delta x_{i-1}} + \Delta x_{i-1} \frac{\Delta y_i}{\Delta x_i}}{\Delta x_{i-1} + \Delta x_i} \quad (18)$$

Für die Randpunkte wird die Sekantensteigung angenommen:

$$y'_1 = \frac{y_2 - y_1}{x_2 - x_1} \quad \text{und} \quad y'_n = \frac{y_n - y_{n-1}}{x_n - x_{n-1}} \quad (19)$$

Somit sind für jeden Abschnitt y und y' an den beiden Seiten eines jeden Intervalls gegeben und es muß nun kein Gleichungssystem mehr gelöst werden. Praktisch handelt es sich also nach der Vorarbeit von Gl. 18 nur noch um eine hermitesche Interpolation.

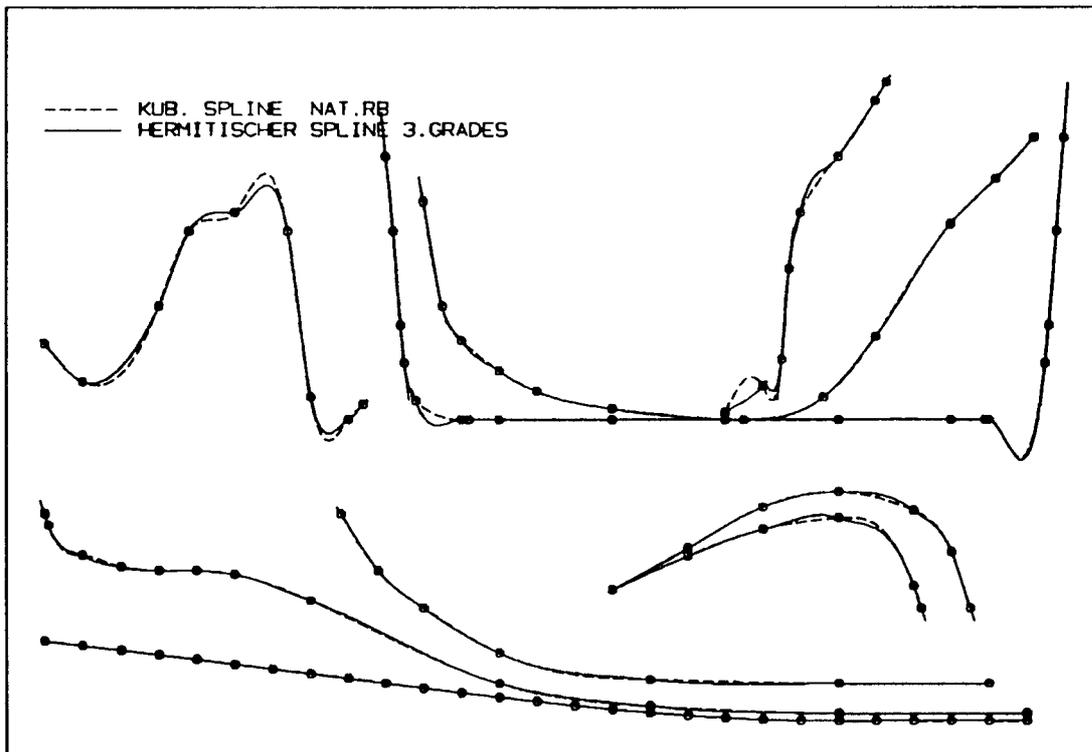


Abbildung 7: Hermitesche Spline-Kurve 3. Grades - Testbild

In Abb. 7 ist die Hermitesche Spline-Kurve 3.Grades zu sehen, gleichzeitig ist als Vergleich der normale kubische Spline eingetragen und es ist an jedem Punkt die Vorgabe der Steigung y' durch einen kurzen Strich markiert. Man sieht, daß die vorgegebenen Steigungen eine Verbesserung gegenüber dem kubischen Splines ist, aber dennoch schwingt auch diese Funktion über und an einigen Stellen ist auch zu sehen, daß die 2. Ableitung nun nicht mehr gekoppelt ist. Beim Diagramm treten beispielsweise starke Krümmungsunterschiede in den beiden letzten Abschnitten auf.

Da dieser Spline in seiner Qualität nicht vorhersehbar und unzuverlässig ist, sollte man von der Anwendung absehen.

3.4 Hermitescher Spline 5.Grades

Als Konsequenz aus dem hermiteschen Spline 3. Grades ist es naheliegend, auch noch die höheren Ableitungen zu koppeln. Dazu sind weitere Variablen notwendig. Man gewinnt sie, indem man den Grad des Polynoms innerhalb eines Intervalls erhöht. Im vorliegenden Fall sind die Polynome dreimal stetig differenzierbar aneinandergesetzt.

$$f_{i-1}^{(k)} = f_i^{(k)} = y_i^{(k)} \quad ; \quad k = 0, 1, 2, 3 \quad (20)$$

Die erste Ableitung wird wie beim hermiteschen Spline 3. Grades vorgegeben.

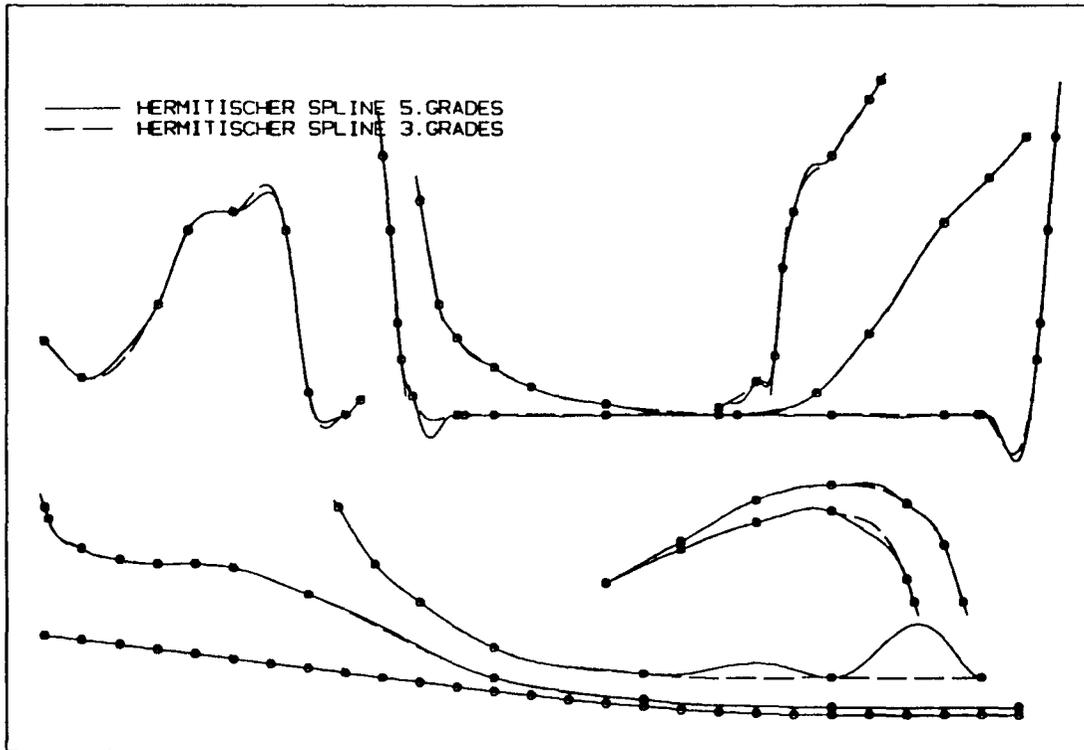


Abbildung 8: Hermitescher Spline 5. Grades - Testbild

In Abb. 8 ist sowohl der hermitesche Spline 3. als auch 5. Grades eingetragen. Man kann sehen, daß der hermitesche Spline 5. Grades beim Diagramm besser ist, die Krümmungen sind jetzt gekoppelt. Aber bei der Hebelarmkurve und bei den Wasserlinien entstehen zusätzliche Beulen, die überhaupt nicht straken.

Somit ist dieser Spline-Algorithmus völlig unbrauchbar.

3.5 Exponentieller Spline

Bei den bisher betrachteten Splines handelte es sich immer um Abwandlungen des kubischen Splines. Natürlich ist aber auch jede andere Funktion $g(x)$ möglich, sofern sie sich nur zweimal stetig ableiten läßt.

$$f_i(x) = a_i g_1(x - x_i) + b_i g_2(x - x_i) + c_i g_3(x - x_i) + d_i g_4(x - x_i) \quad (21)$$

Besonders geeignet sind solche Funktionen, die sich aus zwei Teilen zusammensetzen.

$$f_i(x) = a_i u + b_i t + c_i g(u) + d_i g(t) \quad (22)$$

mit $t = \frac{x-x_i}{\Delta x_i}$; $u = 1 - t = \frac{x_{i+1}-x}{\Delta x_i}$

$$g(0) = 0 \quad ; \quad g(1) = 0 \quad (23)$$

Der erste Teil mit den Koeffizienten a und b erzeugt einen Polygonzug, während der zweite mit den Koeffizienten c und d den stetigen Übergang der ersten und zweiten Ableitung bewirkt.

Die erste Ableitung ist dann :

$$f'_i(x) = \frac{1}{\Delta x_i} (-a_i + b_i - c_i \cdot g'(u) + d_i \cdot g'(t)) \quad (24)$$

Auch hier lassen sich die Koeffizienten a, b, c und d durch y und y' ersetzen

$$a_i = y_i \quad (25)$$

$$b_i = y_{i+1} \quad (26)$$

$$c_i = \frac{1}{g'^2(1) - g'^2(0)} \cdot (\Delta y_i (g'(1) - g'(0)) + \Delta x_i (g'(0) \cdot y'_{i+1} - g'(1) \cdot y'_i)) \quad (27)$$

$$d_i = \frac{1}{g'^2(1) - g'^2(0)} \cdot (\Delta y_i (g'(0) - g'(1)) + \Delta x_i (g'(1) \cdot y'_i - g'(0) \cdot y'_{i+1})) \quad (28)$$

mit $g'^2(1) \neq g'^2(0)$

Die Splinebedingung $f''_{i-1}(x_i) = f''_i(x_i) = y''_i$ ergibt dann Gl. 29 für das symmetrische, tridiagonale Gleichungssystem (vgl.:Gl. 12).

$$\frac{\alpha_{i-1}}{\Delta x_{i-1}} y'_{i-1} + \left(\frac{\beta_{i-1}}{\Delta x_{i-1}} + \frac{\beta_i}{\Delta x_i} \right) y'_i + \frac{\alpha_i}{\Delta x_i} y'_{i+1} = \gamma_{i-1} \frac{\Delta y_{i-1}}{\Delta x_{i-1}^2} + \gamma_i \frac{\Delta y_i}{\Delta x_i^2} \quad (29)$$

$$\alpha_i = -\frac{g'(0)g''(1) + g'(1)g''(0)}{g'^2(1) - g'^2(0)} \quad (30)$$

$$\beta_i = \frac{g'(0)g''(0) + g'(1)g''(1)}{g'^2(1) - g'^2(0)} \quad (31)$$

$$\gamma_i = \frac{g''(1) - g''(0)}{g'(0) + g'(1)} \quad (32)$$

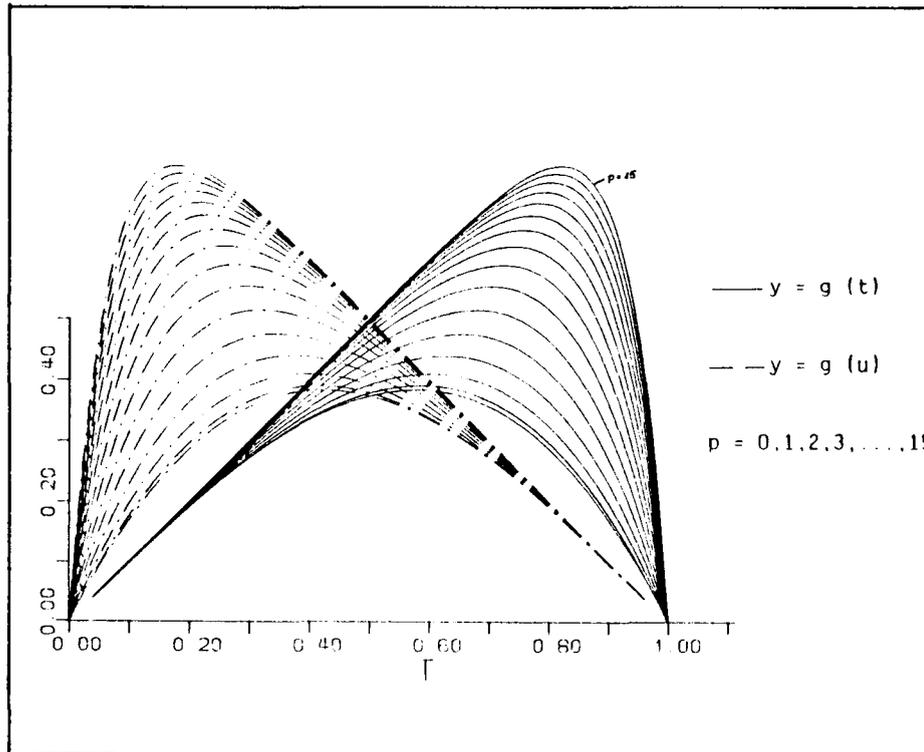


Abbildung 9: Exponentielle Funktion

Nun muß nur noch eine geeignete Funktion $g(t)$ gefunden werden, die folgende Bedingungen erfüllt :

$$g(0) = 0 \quad ; \quad g(1) = 0 \quad ; \quad g'^2(0) \neq g'^2(1) \quad (33)$$

Natürlich soll $g(t)$ auch möglichst einfach zu berechnen sein und man sollte $g(t)$ so erweitern können, daß man die Funktion durch einen Parameter p ergänzen kann, für den gilt :

$$\lim_{p \rightarrow \infty} g(t, p) = \pm t \quad (34)$$

d.h. für $p = \infty$ wird die Kurve zu einem Polygonzug

$$f_i(x) = (a_i + c_i)u + (b_i + d_i)t \quad (35)$$

Eine geeignete Funktion in diesem Sinn ist

$$g_i(p_i, t) = \frac{\sinh(p_i t) - t \cdot \sinh(p_i)}{\sinh(p_i) - p_i} \quad (36)$$

Diese Funktion wurde erstmals von Schweikert [7] aus der Biegelinie einer Straklatte unter Zugspannung "spline in tension" gefunden und als exponentieller Spline bezeichnet.

Vorausgesetzt der $\sinh(x)$ wird durch eine Reihenentwicklung angenähert, wird diese Funktion zusätzlich für den Grenzfall $p \rightarrow 0$ identisch mit dem kubischen Polynom.

Analog zum kubischen Spline läßt sich aus Gleichung 29 ein Gleichungssystem aufbauen. Die Randbedingungen sind z.B. wieder gegebene erste Ableitungen y'_1 und y'_n . Der Parameter p_i kann dabei für jedes Intervall verschieden sein.

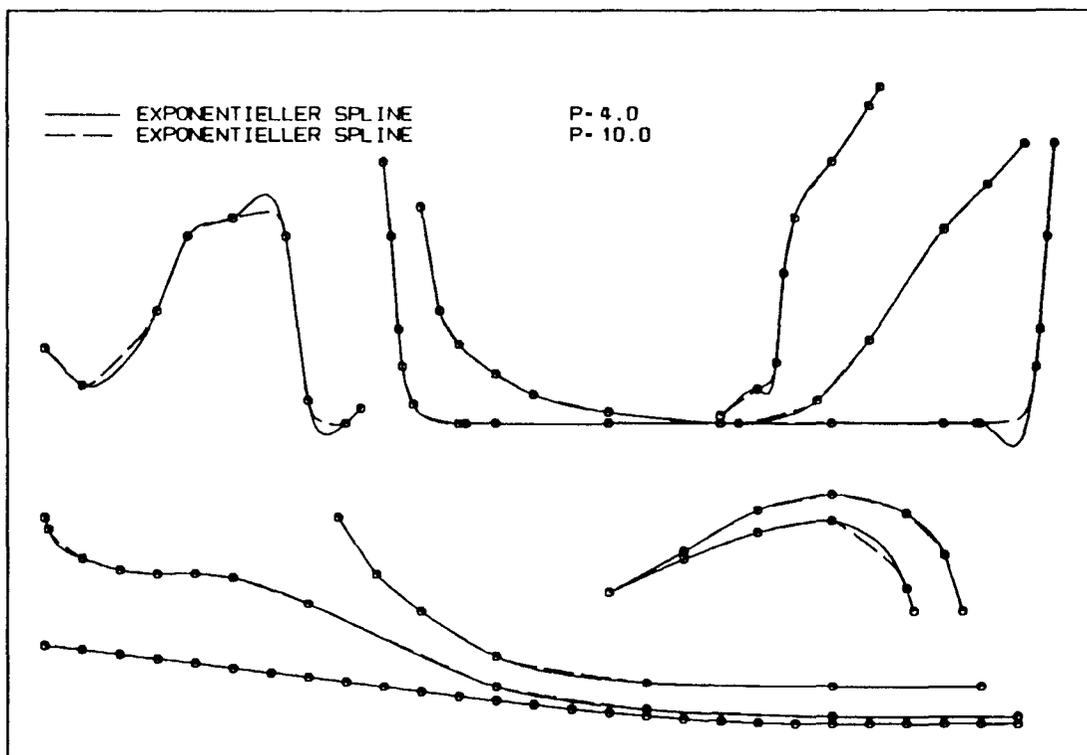


Abbildung 10: exponentieller Spline - Testbild

Abb. 10 zeigt das Testbild gestrakt mit dem exponentiellen Spline, dabei ist der Parameter für alle Intervalle $p = p_i = 4$ bzw. $p = 10$. Der Strak für $p = 4$ enthält noch deutlich - aber abgeschwächt - alle Fehler, die auch der

kubische Spline hatte. Für $p = 10$ ist die Annäherung an den Polygonzug bereits so stark, daß die Kurve eckig wirkt.

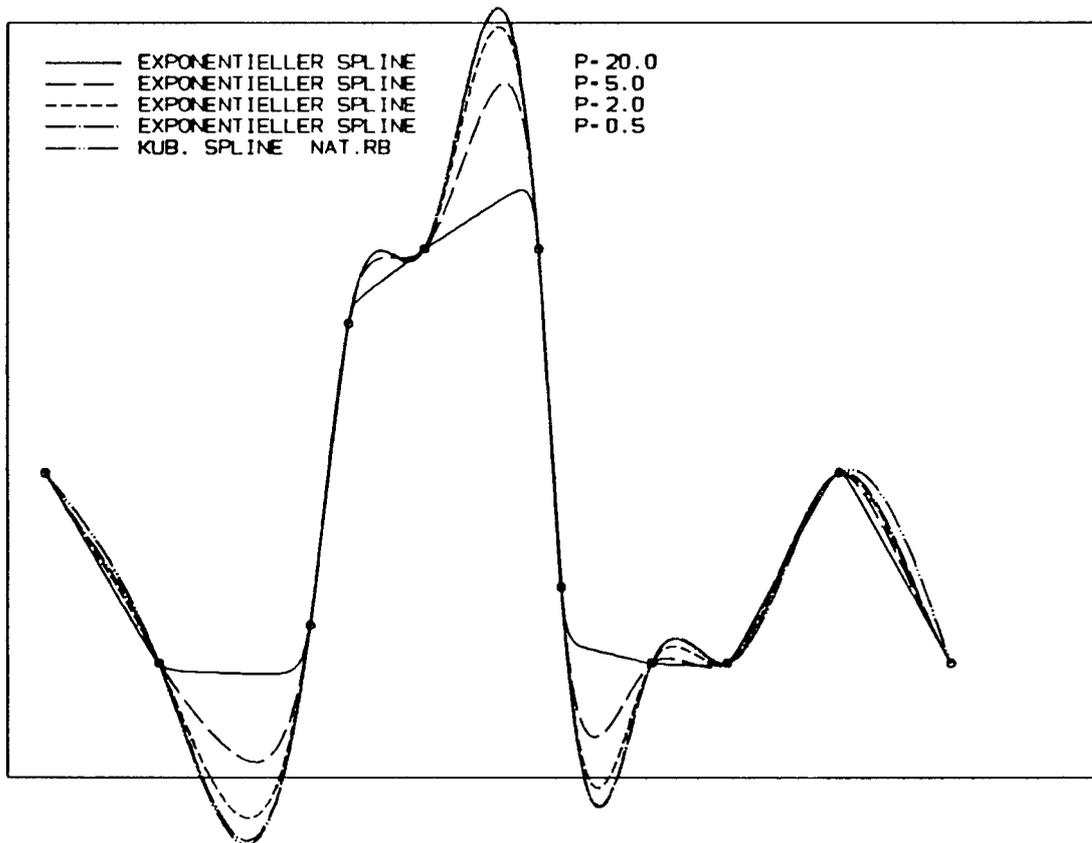


Abbildung 11: exponentieller Spline - Diagramm

Abb. 11 zeigt das große Diagramm mit verschiedenen Parametern p . Die Fehler des kubischen Splines lassen sich zwar beheben, aber dann wird der Spline sehr kantig. Mit einem relativ hohem Aufwand läßt er sich durch Versuche mit intervallweise unterschiedlichem Parameter p_i noch etwas verbessern, aber die Neigung dieses Splines zu einem "Knick" in der Mitte der Intervalle bleibt bestehen.

Somit ist dieser Spline etwas besser als der kubische Spline, aber das rechtfertigt den notwendigen, hohen Arbeitsaufwand nicht. Rechnerseitig verlangsamten die hyperbolischen Funktionen $\sinh(t)$ und $\cosh(t)$ die Ausführung und selbst, wenn sie in einer Tabelle abgelegt werden, bleibt noch sehr viel Rechnerarbeit und damit viel Rechenzeit.

3.6 Rationaler Spline

Eine weitere Verbesserung des Splines lässt sich durch eine andere Funktion $g(x)$ erreichen (Abb. 12)

$$g_i(p_i, t) = \frac{t^3}{p_i(1-t) + 1} - t \quad (37)$$

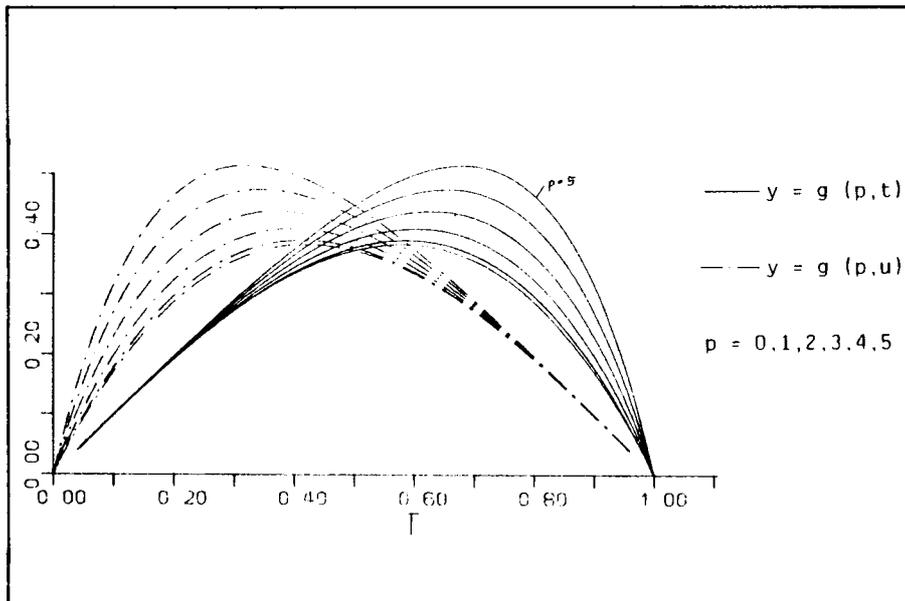


Abbildung 12: Rationale Funktion

Man kann hier den Term $-t$ auch weglassen, da er schon in a, b enthalten ist. Eingesetzt in Gl. 22 ergibt sich :

$$f_i(x) = a_i u + b_i t + c_i \frac{u^3}{p_i t + 1} + d_i \frac{t^3}{p_i u + 1} \quad (38)$$

Zusätzlich kann man nun auch noch den Parameter p für $g(q, t)$ und $g(p, u)$ getrennt definieren, um die rechte und linke Seite eines Intervalls unterschiedlich hart zu straken.

Setzt man wieder y'_i als unbekannte Variable ein, so führt die Spline-Bedingung $f''_{i-1}(x_i) = f''_i(x_i) = y'_i$ auf die Gleichung (39). Das Gleichungssystem ist nun nicht mehr symmetrisch wie beim exponentiellen Spline (vgl. Gl. 29), dies entsteht aber nur durch die ungleichen Parameter $p_i \neq q_i$.

$$\frac{\alpha_{i-1}}{\Delta x_{i-1}} y'_{i-1} + \left(\frac{\beta_{i-1}}{\Delta x_{i-1}} + \frac{\tilde{\beta}_i}{\Delta x_i} \right) y'_i + \frac{\tilde{\alpha}_i}{\Delta x_i} y'_{i+1} = \gamma_{i-1} \frac{\Delta y_{i-1}}{\Delta x_{i-1}^2} + \tilde{\gamma}_i \frac{\Delta y_i}{\Delta x_i} \quad (39)$$

$$\alpha_i = \frac{q_{i-1}^2 + 3q_{i-1} + 3}{(2 + p_{i-1}) \cdot (2 + q_{i-1}) - 1} \quad (40)$$

$$\beta_i = \frac{(q_{i-1}^2 + 3q_{i-1} + 3) \cdot (2 + p_{i-1})}{(2 + p_{i-1}) \cdot (2 + q_{i-1}) - 1} \quad (41)$$

$$\tilde{\beta}_i = \frac{(p_i^2 + 3p_i + 3) \cdot (2 + q_i)}{(2 + p_i) \cdot (2 + q_i) - 1} \quad (42)$$

$$\tilde{\alpha}_i = \frac{(p_i^2 + 3p_i + 3)}{(2 + p_i) \cdot (2 + q_i) - 1} \quad (43)$$

$$\gamma_i = \frac{(q_{i-1}^2 + 3q_{i-1} + 3) \cdot (3 + p_{i-1})}{(2 + p_{i-1}) \cdot (2 + q_{i-1}) - 1} \quad (44)$$

$$\tilde{\gamma}_i = \frac{(p_i^2 + 3p_i + 3) \cdot (3 + q_i)}{(2 + p_i) \cdot (2 + q_i) - 1} \quad (45)$$

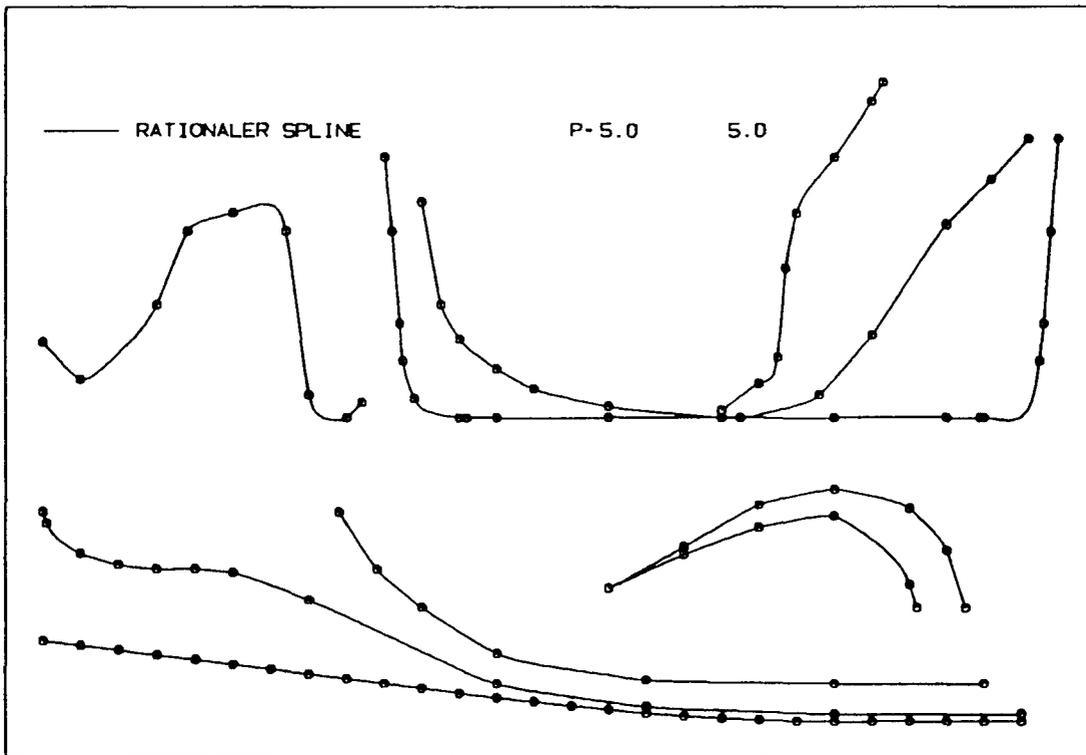


Abbildung 13: rationaler Spline - Testbild

Abb. 13 zeigt das Testbild mit dem rationalen Spline gestrakt. Der Kurvenparameter wurde für alle Intervalle auf $p_i = q_i = 5$ gesetzt. Wesentliche Fehler des kubischen Splines sind nun verschwunden, allerdings sind z.B. beim Spantriß im Vorschiffbereich neue, andere hinzugekommen.

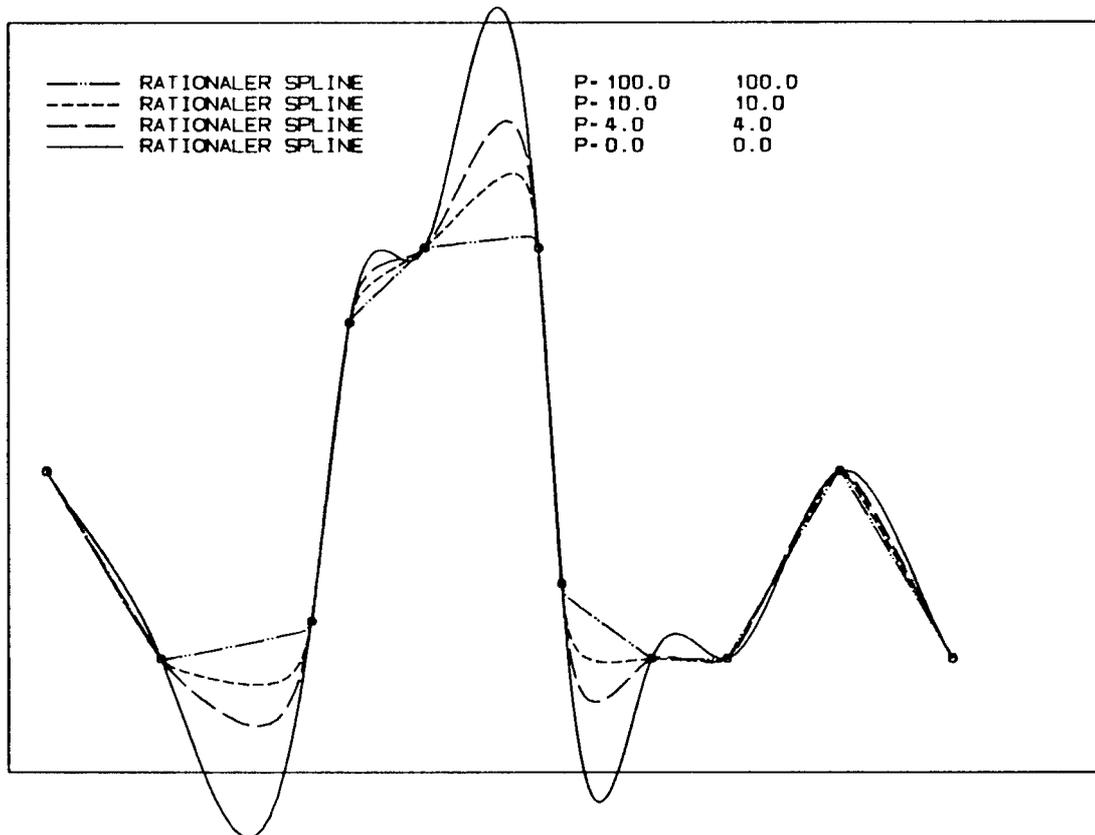


Abbildung 14: rationaler Spline - Diagramm

Der Strak läßt sich, analog zum exponentiellen Spline, durch intervallweise verschiedene Parameter p und q noch verbessern. Abb. 14 und Abb. 15 zeigen eine Parameterstudie am großen Diagramm.

In Abb. 15 ist die Kurve sehr gut gestrakt, es treten keinerlei Fehler mehr auf und man könnte meinen, der Strak sei per Hand erzeugt. Leider ist der Arbeitsaufwand beim Ausprobieren der Parameter unverträglich hoch, so daß dieser Algorithmus nur für Geübte schnell zu einem guten Ergebnis führt.

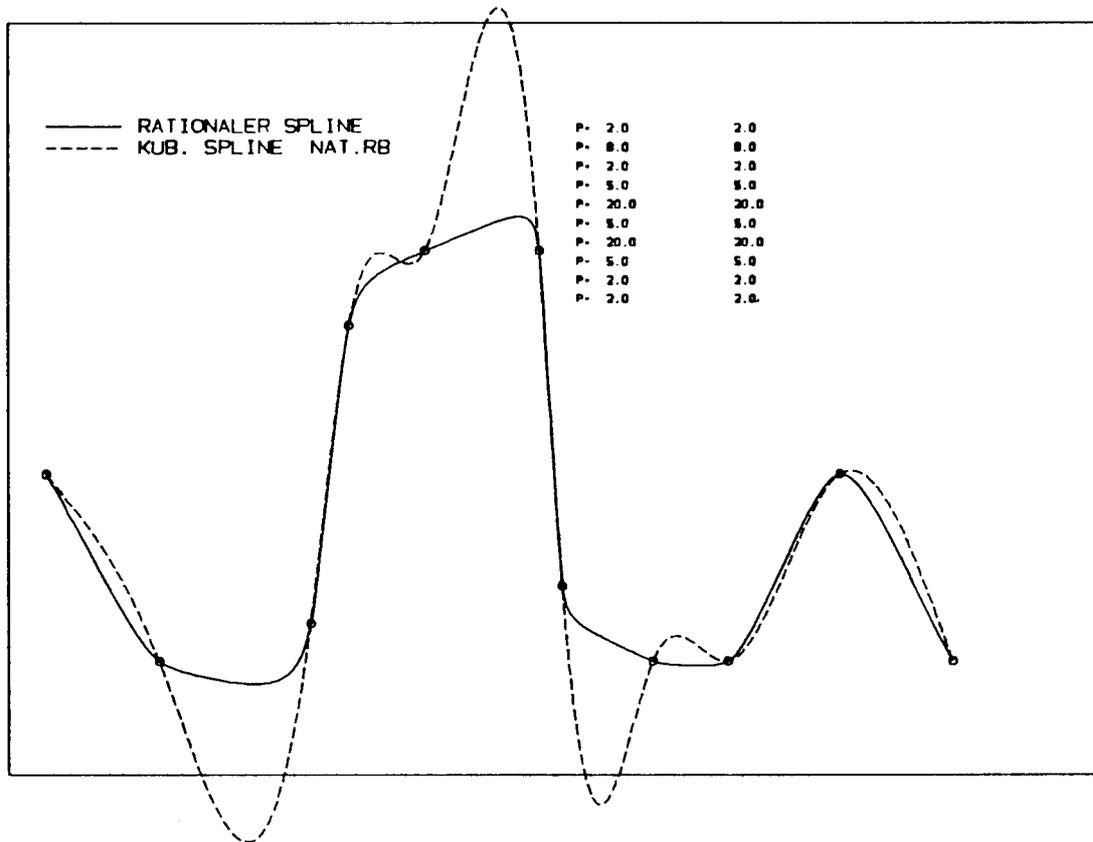


Abbildung 15: rationaler Spline - Diagramm

Da die Wahl der Parameter im wesentlichen von der Sekantensteigung der aneinanderstoßenden Intervalle abhängt, liegt es nahe, dies mathematisch zu formulieren und die Parameter p_i und q_i in einem Unterprogramm automatisch berechnen zu lassen.

$$p_i = |\Delta y'_i - \Delta y'_{i-1}| \cdot 1.2 \quad , \quad q_i = |\Delta y'_i - \Delta y'_{i+1}| \cdot 1.2 \quad (46)$$

$$\text{mit } \Delta y'_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \quad (47)$$

In Abb. 16 und Abb. 17 wurden die Parameter durch die Gleichungen Gl. 46 berechnet. Der Strak ist zwar gut, aber der Rechenaufwand war schon für den normalen rationalen Spline sehr hoch.

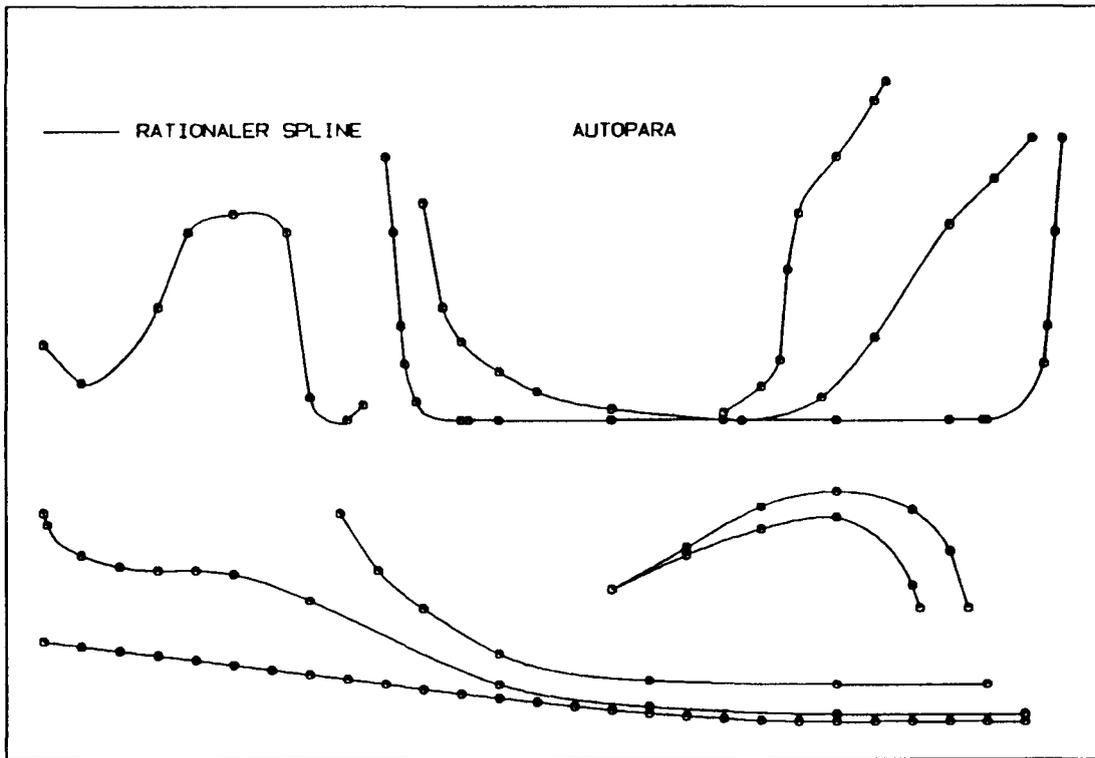


Abbildung 16: auto. rationaler Spline - Testbild

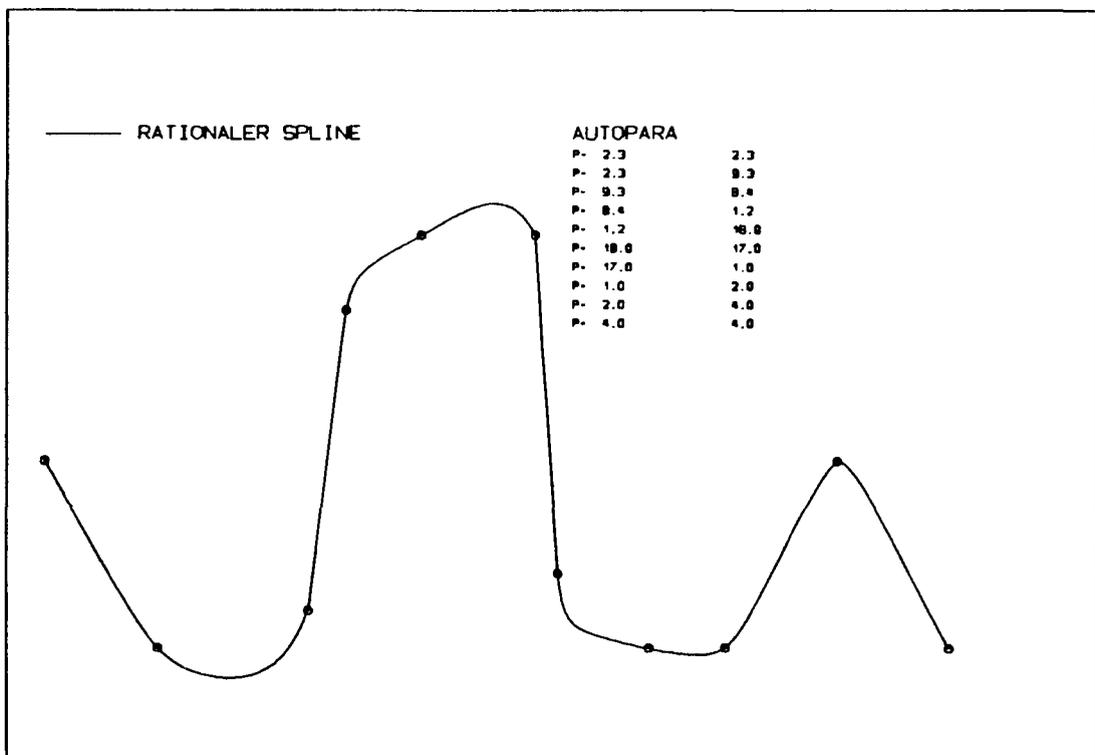


Abbildung 17: auto. rationaler Spline - Diagramm

3.7 Kubischer Spline mit mitgeführten Koordinaten

Die gesamte bisher gezeigte Palette der Splines hatte den erheblichen Nachteil, daß sie an Stellen mit sehr großer Steigung entweder gar nicht funktioniert oder schwerwiegende Mängel aufgewiesen hat.

Dieses Problem läßt sich aber dadurch beheben, daß man den Kurvenverlauf jeweils innerhalb eines Intervalls in einem eigenen Koordinatensystem beschreibt, das durch die beiden Endpunkte des Intervalls verläuft. Gleichzeitig wird die örtliche Abszisse η auch normiert.

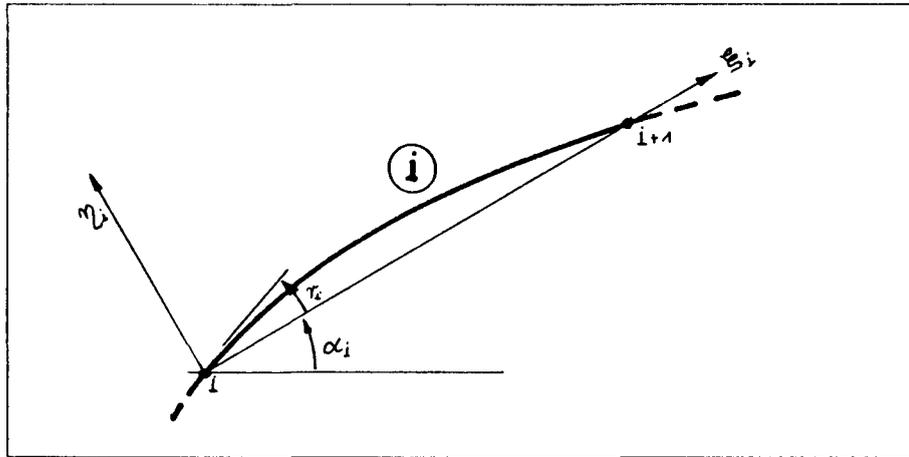


Abbildung 18: Das mitgeführte Koordinatensystem

$$x = x_i + \xi_i \cos \alpha_i - \eta_i \sin \alpha_i \quad (48)$$

$$y = y_i + \xi_i \sin \alpha_i + \eta_i \cos \alpha_i \quad (49)$$

$$\alpha_i = \arctan \left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i} \right) \quad (50)$$

$$(51)$$

Auch hier wird ein kubischer Ansatz gemacht:

$$\eta = t u \cdot (f'_i(0) u - f'_i(1) t) \cdot l_i \quad (52)$$

$$\text{mit } l_i = \sqrt{\Delta x_i^2 + \Delta y_i^2} \quad \text{und } t = \frac{\xi_i}{l_i} \quad \text{und } u = 1 - t \quad (53)$$

Bei der Kopplung benachbarter Abschnitte muß dann zusätzlich zur 1. Ableitung auch noch der Winkel des örtlichen Koordinatensystems berücksichtigt werden.

$$\arctan(f'_i(1)) + \alpha_i = \arctan(f'_{i+1}(0)) + \alpha_{i+1} \quad (54)$$

Aus der Kopplung der 2. Ableitung ergibt sich dann :

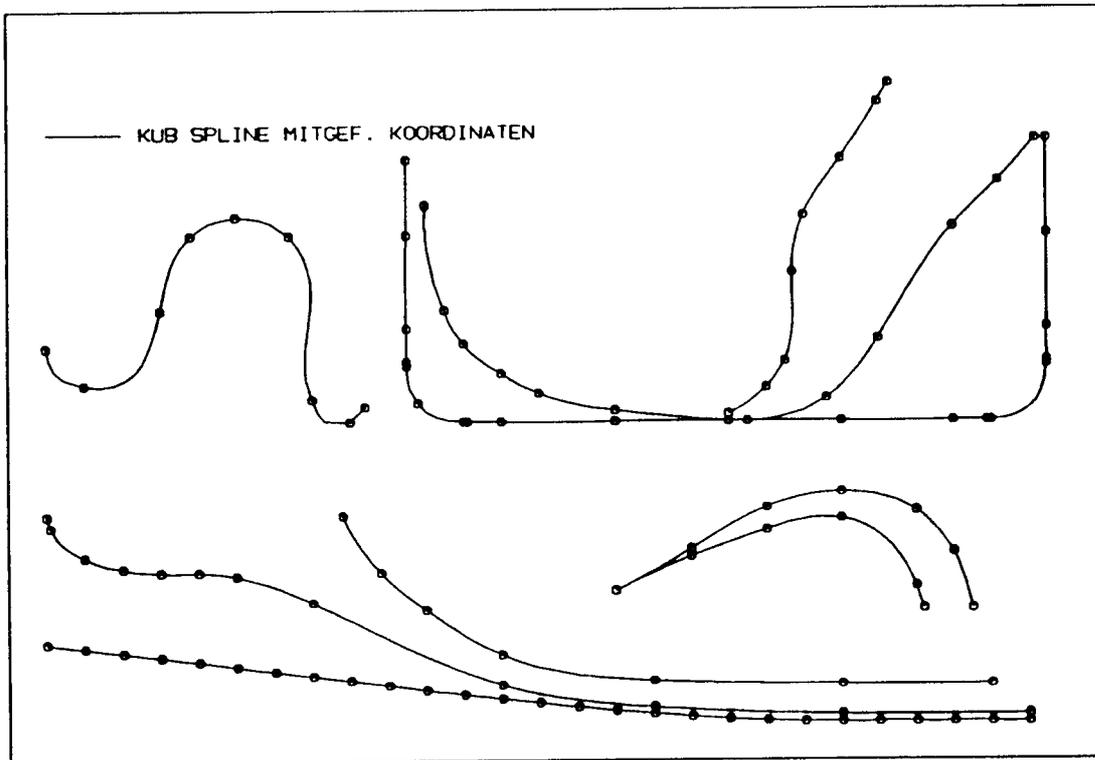


Abbildung 19: kub. Spline mitgef. Kood.- Testbild

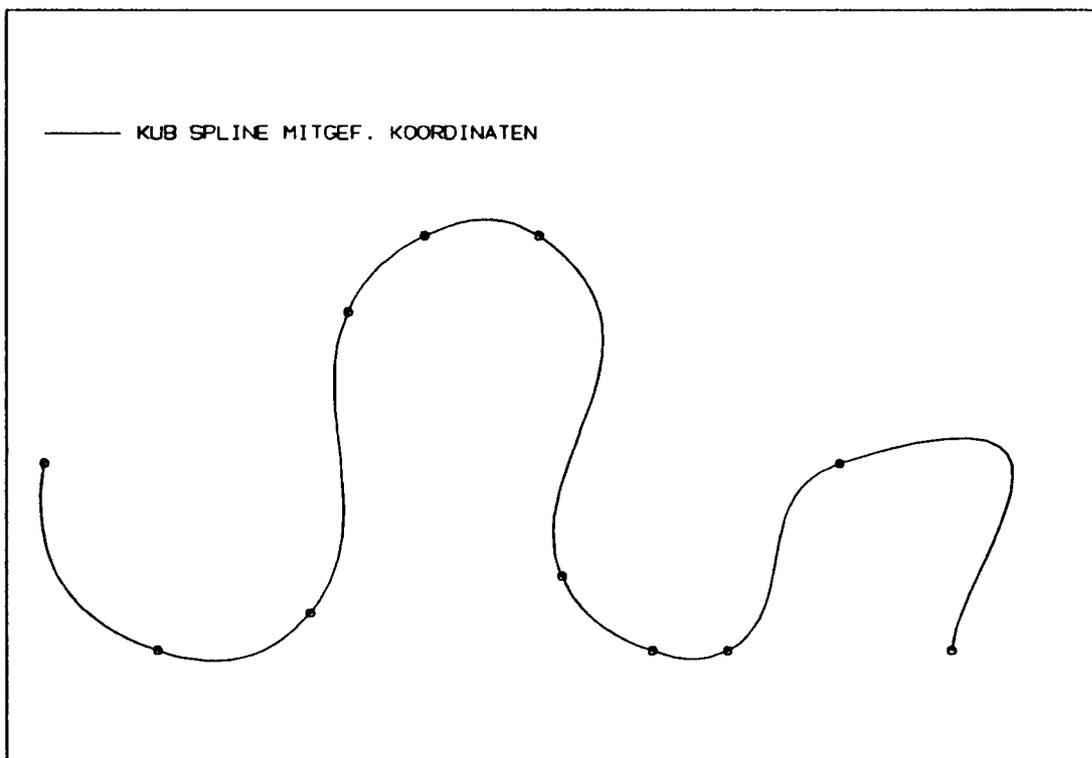


Abbildung 20: kub. Spline mitgef. Kood.- Diagramm

$$A \cdot \begin{pmatrix} f'_{x1} \\ f'_{x2} \\ f'_{x3} \\ f'_{x4} \\ \vdots \\ f'_{x,n-1} \\ f'_{xn} \end{pmatrix} = \begin{pmatrix} -\frac{f''_{x1}}{2} + 3\frac{\Delta x_1}{\Delta t_1^2} \\ 3\frac{\Delta x_1}{\Delta t_1^2} + 3\frac{\Delta x_2}{\Delta t_2^2} \\ 3\frac{\Delta x_2}{\Delta t_2^2} + 3\frac{\Delta x_3}{\Delta t_3^2} \\ 3\frac{\Delta x_3}{\Delta t_3^2} + 3\frac{\Delta x_4}{\Delta t_4^2} \\ \vdots \\ 3\frac{\Delta x_{n-2}}{\Delta t_{n-2}^2} + 3\frac{\Delta x_{n-1}}{\Delta t_{n-1}^2} \\ \frac{f''_{xn}}{2} + 3\frac{\Delta x_{n-1}}{\Delta t_{n-1}^2} \end{pmatrix} \quad (62)$$

$$A \cdot \begin{pmatrix} f'_{y1} \\ f'_{y2} \\ f'_{y3} \\ f'_{y4} \\ \vdots \\ f'_{y,n-1} \\ f'_{yn} \end{pmatrix} = \begin{pmatrix} -\frac{f''_{y1}}{2} + 3\frac{\Delta y_1}{\Delta t_1^2} \\ 3\frac{\Delta y_1}{\Delta t_1^2} + 3\frac{\Delta y_2}{\Delta t_2^2} \\ 3\frac{\Delta y_2}{\Delta t_2^2} + 3\frac{\Delta y_3}{\Delta t_3^2} \\ 3\frac{\Delta y_3}{\Delta t_3^2} + 3\frac{\Delta y_4}{\Delta t_4^2} \\ \vdots \\ 3\frac{\Delta y_{n-2}}{\Delta t_{n-2}^2} + 3\frac{\Delta y_{n-1}}{\Delta t_{n-1}^2} \\ \frac{f''_{yn}}{2} + 3\frac{\Delta y_{n-1}}{\Delta t_{n-1}^2} \end{pmatrix} \quad (63)$$

Die Ableitungen verstehen sich immer als $\frac{\delta}{\delta t}$. Als Randbedingung wird $f''_1 = f''_n = 0$ verwendet.

In Abb. 21 und Abb. 22 ist dieser parametrische Spline zu sehen. Alle schwierigen Stellen werden sehr gut gestrakt, der Kimmradius (zum Vergleich ist ein Kreis eingezeichnet) wird ausgezeichnet wiedergegeben. Auch bei diesem Algorithmus entstehen nicht immer eindeutige Funktionen. Abb. 23 zeigt wie man diesen Effekt auch bewußt nutzen kann.

Da dieser Spline die beiden Koordinaten x und y mit einem kubischen Spline über der (Ersatz-)Bogenlänge ausstrakt, benötigt er doppelt so viel Rechenzeit wie der einfach kubische Spline. In der praktischen Anwendung müßte man aber bei dem einfach kubischen Spline zunächst prüfen, ob die Abszissen auch wirklich streng monoton ansteigen, bevor man den eigentlichen Strak starten kann. Beim parametrisch kubischem Spline droht einem hier kein Rechnerabsturz, die Abfrage und Fehlerbehandlung kann man sich sparen.

Analog zum kubischen Spline läßt sich auch jeder andere Spline-Algorithmus zu einem parametrischen Spline umbauen. Der rationale Spline bringt hier aber kaum noch Vorteile gegenüber dem kubischen Spline, da systembedingt die Steigung der Kurve auf den Bereich $-1 < f'(x) < 1$ begrenzt ist und der rationale Spline erst bei sehr großen Steigungen vom kubischen abweicht.

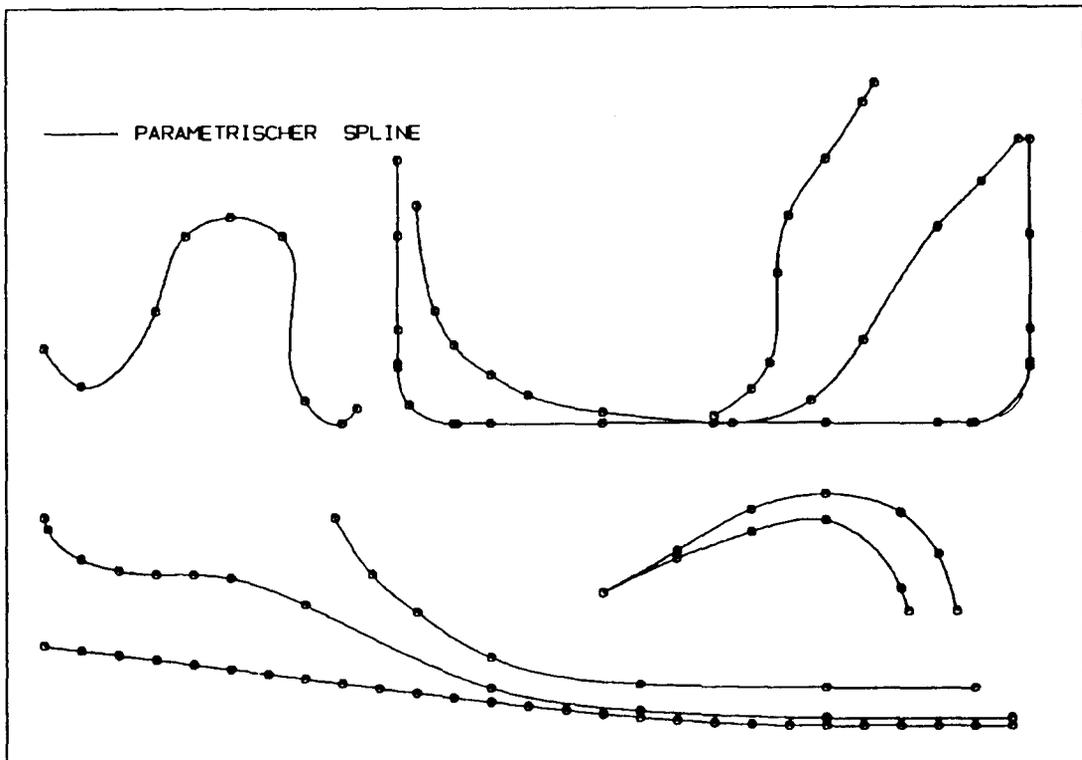


Abbildung 21: Parametrisch kubischer Spline - Testbild

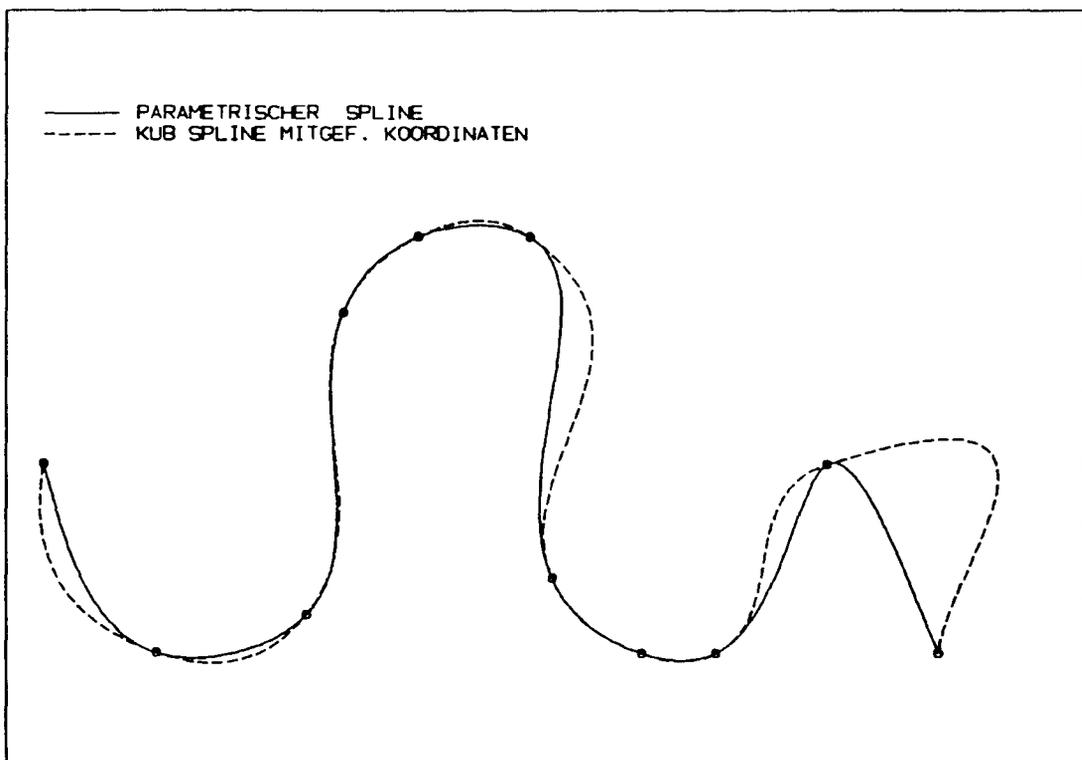


Abbildung 22: Parametrisch kubischer Spline - Diagramm

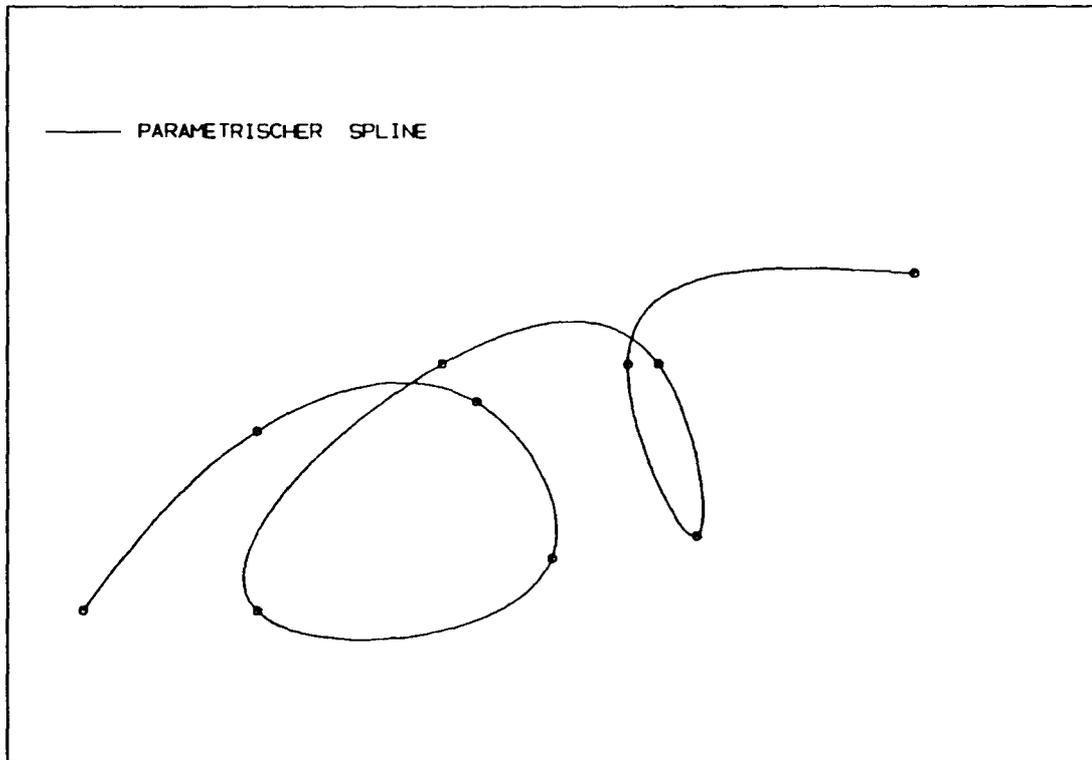


Abbildung 23: Parametrisch kubischer Spline - Schweineschwanz

3.9 Kubische Bézier-Kurve

Ein von den bisher vorgestellten Spline-Algorithmen verschiedenes Verfahren ist die Bézier-Kurve [2, 5]. Hierbei ist es nicht notwendig ein Gleichungssystem über den gesamten Spline zu lösen. Dieser Vorteil wird dadurch erkauft, daß bei der Bézier-Kurve die vorgegebenen Punkte nicht erfüllt werden, sondern sie werden wie bei einer Approximation nur als Gewichtung genommen. Die Wirkung eines einzelnen Knotens ist auf die beiden angrenzenden Intervall beschränkt.

Zu den vorgegebenen Knoten, den Kontrollpunkten $\vec{t}_n = (x_n, y_n)$, werden nach einem sehr einfachen Algorithmus sog. Bézier-Punkte $\vec{b}_{j,i}$ mit $j = 0, 1, 2, 3$ und $i = 1, 2, 3, \dots, (n-1)$ gebildet (Abb. 24). Zu jedem Intervall gehören jeweils vier Bézier-Punkte. Der letzte Punkt $\vec{b}_{3,i}$ des Intervalls i ist mit dem ersten Punkt $\vec{b}_{0,i+1}$ des folgenden Intervalls ($i+1$) identisch.

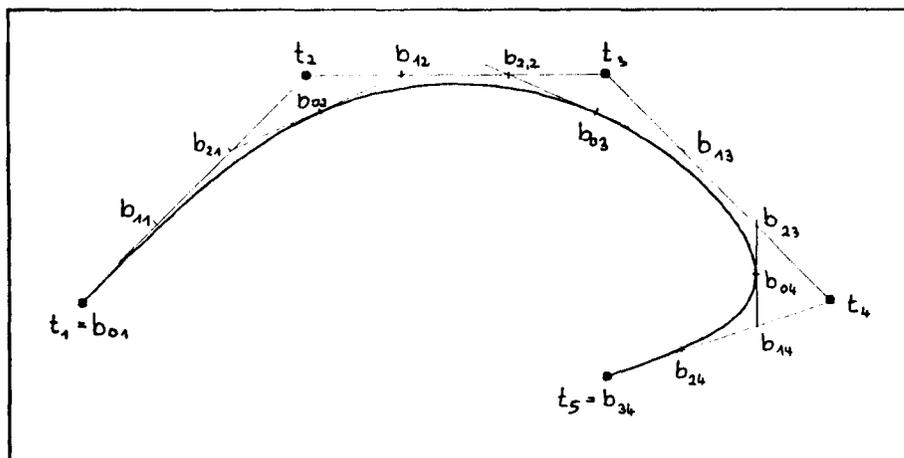


Abbildung 24: Bézier-Punkte bei der Bézier-Kurve

$$\vec{b}_{0,i} = \frac{1}{2}(\vec{b}_{2,i-1} + \vec{b}_{1,i}) \quad (64)$$

$$\vec{b}_{1,i} = \frac{1}{3}(\vec{t}_{i+1} + 2\vec{t}_i) \quad (65)$$

$$\vec{b}_{2,i} = \frac{1}{3}(2\vec{t}_{i+1} + \vec{t}_i) \quad (66)$$

$$\vec{b}_{3,i} = \frac{1}{2}(\vec{b}_{2,i} + \vec{b}_{1,i+1}) = \vec{b}_{0,i+1} \quad (67)$$

mit $i = 1, 2, \dots, (n-1)$. Hinzu kommen die natürlichen Randbedingungen:

$$\vec{b}_{0,1} = \vec{t}_1 \quad ; \quad \vec{b}_{3n} = \vec{t}_n \quad (68)$$

Innerhalb des Intervalles i berechnet sich die Kurve $\vec{P}_i(v)$ dann durch

$$\vec{P}_i(v) = \vec{b}_{0,i}(1-v)^3 + 3\vec{b}_{1,i}(1-v)^2v + 3\vec{b}_{2,i}(1-v)v^2 + \vec{b}_{3,i}v^3 \quad (69)$$

mit dem lokalen, äquidistanten Parameter $v \in [0, 1]$.

Diese Funktion läßt sich auch straffer und unübersichtlicher mit Hilfe der kubischen Bernstein-Polynome darstellen:

$$\vec{P}_i(v) = \sum_{j=0}^3 \vec{b}_{j,i} B_j^3(v) \quad (70)$$

$$B_j^3(v) = \binom{3}{j} (1-v)^{3-j} v^j \quad ; \quad j = 0, 1, 2, 3 \quad (71)$$

Die Parameterdarstellung Gl. 69 ist so gewählt, daß die erste Ableitung (Gl. 72) an der Stelle $v = 0$ und $v = 1$ gerade gleich dem Differenzvektor der benachbarten Bézier-Punkte ist.

$$\vec{P}'_i(v) = -3\vec{b}_{0,i}(1-v)^2 + 3\vec{b}_{1,i}(1-4v+3v^2) + 3\vec{b}_{2,i}(2v-3v^2) + 3\vec{b}_{3,i}v^2 \quad (72)$$

$$\vec{P}'_i(0) = -3\vec{b}_{0,i} + 3\vec{b}_{1,i} = 3\vec{a}_{1,i} \quad (73)$$

$$\vec{P}'_i(1) = -3\vec{b}_{2,i} + 3\vec{b}_{3,i} = 3\vec{a}_{3,i} = 3\vec{a}_{1,i+1} = \vec{P}'_{i+1}(0) \quad (74)$$

Die zweite Ableitung Gl. 75 ist in den Übergängen der Intervalle stetig.

$$\vec{P}''_i(v) = 6\vec{b}_{0,i}(1-v) + 3\vec{b}_{1,i}(-4+6v) + 3\vec{b}_{2,i}(2-6v) + 6\vec{b}_{3,i}v \quad (75)$$

$$\vec{P}''_i(0) = 6(\vec{b}_{0,i} - 2\vec{b}_{1,i} + \vec{b}_{2,i}) = 6(\vec{b}_{0,i} - \vec{t}_{i-1}) \quad (76)$$

$$\vec{P}''_i(1) = 6(\vec{b}_{1,i} - 2\vec{b}_{2,i} + \vec{b}_{3,i}) = 6(\vec{b}_{3,i} - \vec{t}_i) = \vec{P}''_{i+1}(0) \quad (77)$$

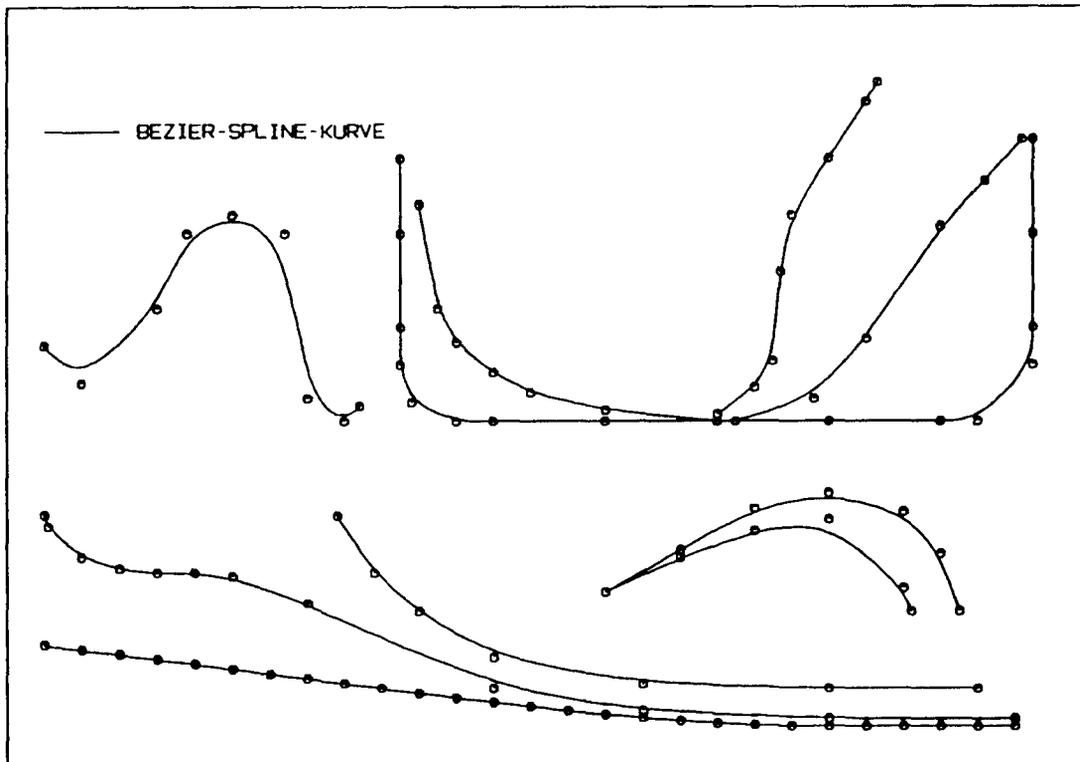


Abbildung 25: Bézier-Kurve - Testbild

Der Nachteil dieses Splines ist, daß nunmehr nicht die Kontrollpunkte sondern nur noch die Interpolationsstellen verwirklicht werden. Die Kurve läuft also neben den vorgegeben Punkten. Nur der erste und der letzte Punkt werden eingehalten, sowie der mittlere Punkt, wenn drei Punkte auf einer Geraden liegen. Die Abszissen müssen auch nicht streng monoton steigend angeordnet sein. Wird ein Punkt doppelt eingegeben, so wirkt er wie der Berührungspunkt zweier getrennter Bézier-Kurven. Fügt man später einen weiteren Punkt in einen bestehenden Spline ein, so bewirkt dies nur Veränderungen in den beiden Nachbarintervallen. Die Kurve zwischen drei beliebigen Punkten eines Splines, liegt stets in dem Dreieck, das diese Punkte aufspannen.

Der Rechenaufwand der Bézier-Kurve ist sehr gering, da kein Gleichungssystem gelöst werden muß. Sie ist von allen Spline-Algorithmien der schnellste.

Die Abb. 25 und Abb. 26 zeigen mit Bézier-Kurven gestraakte Kurven. Offensichtlich verläuft die Kurve neben den Punkten, sie ist aber auf jeden Fall glatt.

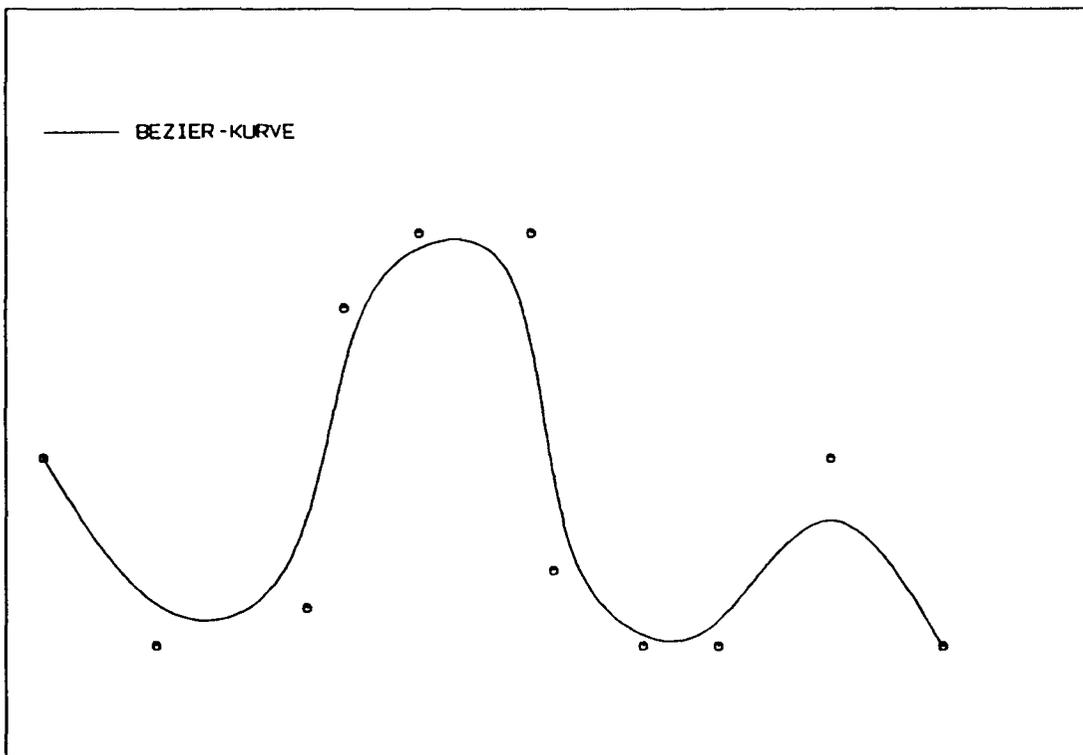


Abbildung 26: Bézier-Kurve - Diagramm

3.10 Verbesserter Bézier-Spline

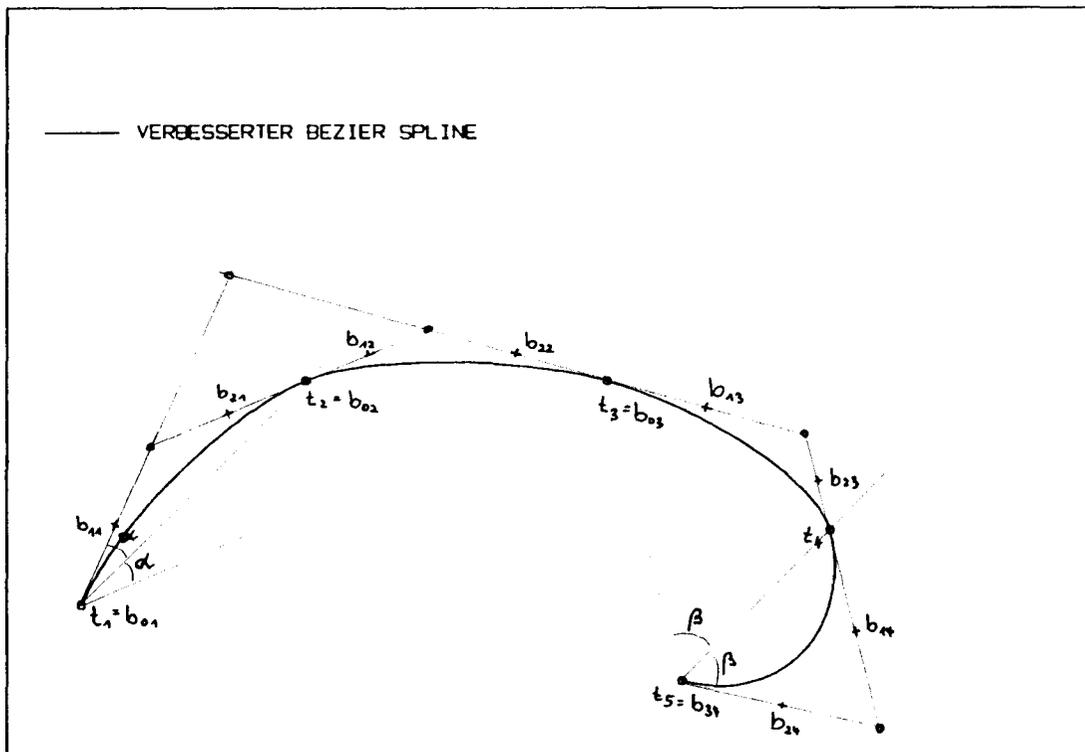


Abbildung 27: Konstruktion des verbesserten Bézier-Splines

Durch eine kleine Änderung des Algorithmuses läuft die Bézier-Kurve durch die vorgegeben Punkte. Dazu wird die Steigung an den Punkten gleich der Sekantensteigung der beiden benachbarten Punkten gesetzt (Abb. 27; es handelt sich also auch hier um keinen "echten" Spline. Die Schnittpunkte der Steigungsgeraden entsprechen dann den normalen Bézier-Punkten. Leider geht damit der Vorteil der kurzen Rechenzeit teilweise verloren, aber es muß auch kein Gleichungssystem gelöst werden. Als Randbedingung wurde hier einmal konstante Krümmung in den Randintervallen verwirklicht. In allen anderen Eigenschaften entspricht der verbesserte Bézier-Spline aber der normalen.

Die Abb. 28 bis Abb. 30 zeigen entsprechende Splines. Der Strak ist nur bei dem Übergang von einer Geraden zum Kreis (Kimmradius im Spantrieb) zu beanstanden, für diese Splineart müßte man die Punkte anders vorgeben. Alle anderen Kurven, besonders das Diagramm, sind sehr gut gestrakt.

Bei großen Unterschieden in der Intervalllänge nähert sich der Kurvenverlauf dem Polygonzug an.

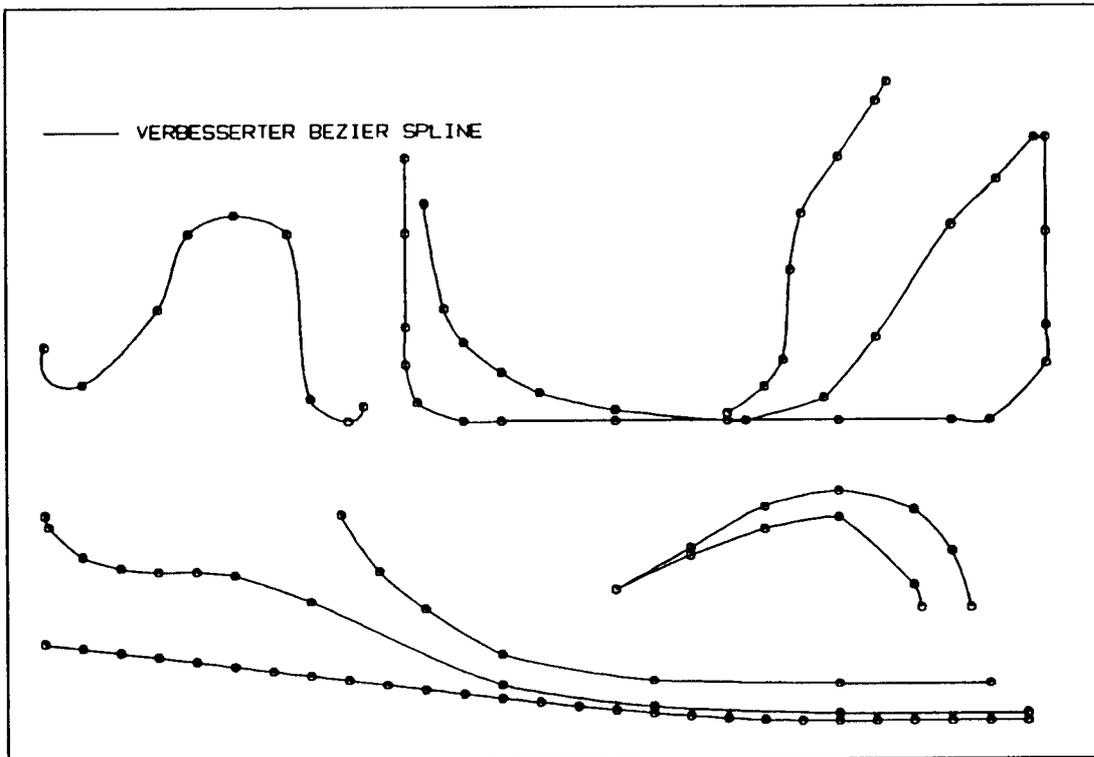


Abbildung 28: Verbessertes Bézier-Spline - Testbild

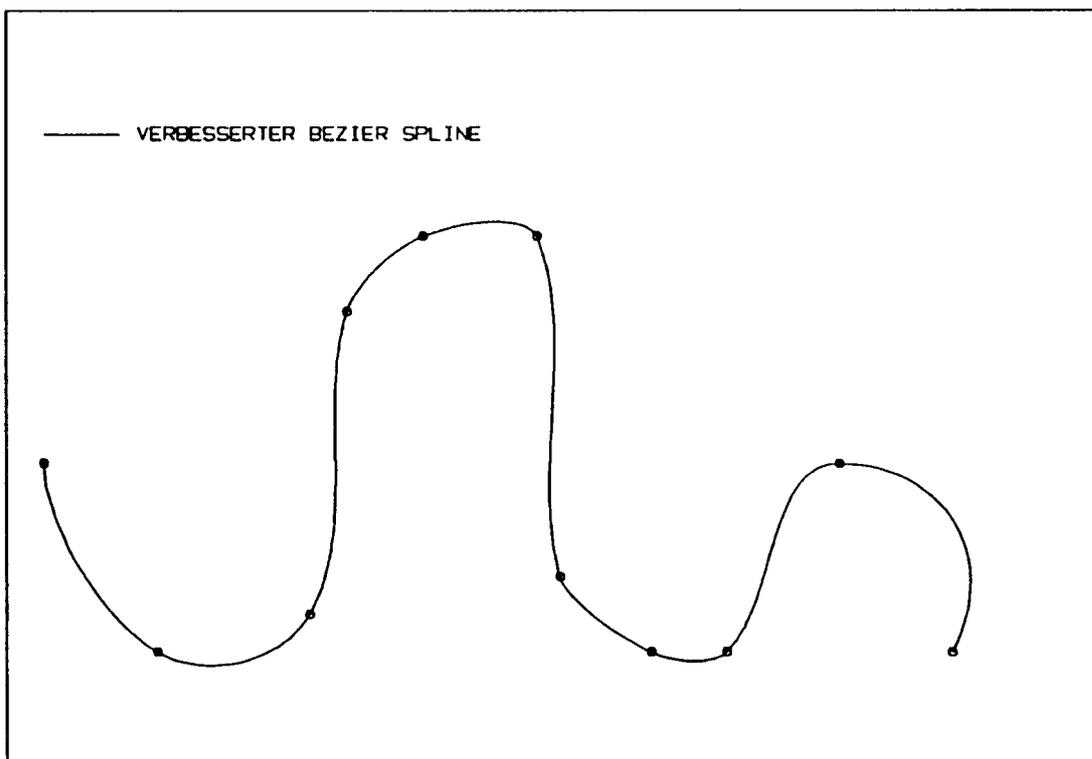


Abbildung 29: Verbessertes Bézier-Spline - Diagramm

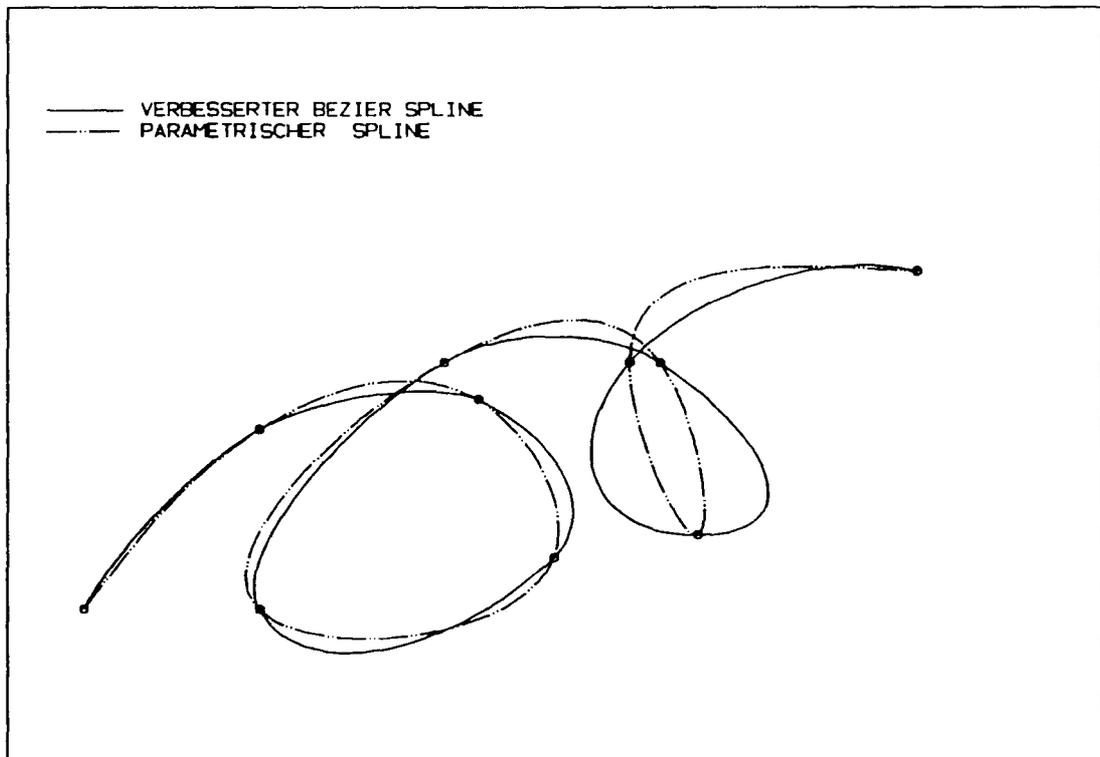


Abbildung 30: Verbesserter Bézier-Spline - Schweineschwanz

3.11 Kubischer Bézier-Spline

Eine Verbesserung der Bézier-Kurve läßt sich nur erreichen, indem die vorgegebenen Knoten auch wirklich erfüllt werden und trotzdem die erste und zweite Ableitung frei gekoppelt ist. Da mehr Forderungen an den Spline auch das Lösen eines Gleichungssystems erfordern, ist damit der Vorteil der geringen Rechenzeit natürlich verloren.

Hierzu formuliert man die Gl. 70 noch einmal neu und verwendet nicht die lokalen, äquidistanten Parameter v , sondern einen globalen Parameter t_n . Dabei ist jedem Knoten n genau ein Parameter t_n zugeordnet und es gilt $t_1 < t_2 < \dots < t_n$. Mit dem gleichen Bernstein-Polynom 3-ten Grades (Gl. 71) gilt:

$$\vec{P}_i(t) = \sum_{j=0}^n \vec{b}_{j,i} B_j^n \left(\frac{t - t_i}{\mu_i} \right) \quad ; \quad \mu_i = t_{i+1} - t_i \quad (78)$$

Die Knoten \vec{P}_i sollen nun erfüllt werden, sowie die Forderung nach stetiger erster und zweiter Ableitung :

$$\vec{P}_{i-1}(t_i) = \vec{P}_i(t_i) = \vec{P}_i = \quad (79)$$

$$\vec{b}_{3,i-1} = \vec{b}_{0,i} \quad (80)$$

$$\vec{P}'_{i-1}(t_i) = \vec{P}'_i(t_i) = f'_i = \quad (81)$$

$$\frac{3}{\mu_{i-1}} (\vec{b}_{3,i-1} - \vec{b}_{2,i-1}) = \frac{3}{\mu_i} (\vec{b}_{1,i} - \vec{b}_{0,i}) \quad (82)$$

$$\vec{P}''_{i-1}(t_i) = \vec{P}''_i(t_i) = \quad (83)$$

$$\frac{3(3-1)}{\mu_{i-1}^2} (\vec{b}_{3,i-1} - 2\vec{b}_{2,i-1} + \vec{b}_{1,i-1}) = \frac{3(3-1)}{\mu_i^2} (\vec{b}_{2,i} - 2\vec{b}_{1,i} + \vec{b}_{0,i}) \quad (84)$$

Gleichung 82 formuliert man vorteilhaft in den Ableitungen f'_i (Gl. 85) und erhält - eingesetzt in Gl. 84 - ein Gleichungssystem (Gl. 86), um die erste Ableitung zu berechnen.

$$\vec{b}_{1,i} = \frac{\mu_i}{3} f'_i + b_{0,i} \quad ; \quad \vec{b}_{2,i} = \frac{-\mu_i}{3} f'_{i+1} + b_{0,i+1} \quad (85)$$

$$\begin{aligned}
& \left[\begin{array}{cccc}
2\frac{1}{\mu_1} + & \frac{1}{\mu_1} & & \\
\frac{1}{\mu_1} + 2\left(\frac{1}{\mu_1} + \frac{1}{\mu_2}\right) + & \frac{1}{\mu_2} & & \\
& \frac{1}{\mu_2} & + 2\left(\frac{1}{\mu_2} + \frac{1}{\mu_3}\right) + & \frac{1}{\mu_3} \\
& \dots & \dots & \dots \\
& & \frac{1}{\mu_{n-1}} & + 2\left(\frac{1}{\mu_{n-1}} + \frac{1}{\mu_n}\right) + \frac{1}{\mu_n} \\
& & & + \frac{1}{\mu_n} & + 2\frac{1}{\mu_n}
\end{array} \right] \cdot \left\{ \begin{array}{c} f'_1 \\ f'_2 \\ f'_3 \\ f'_4 \\ \vdots \\ f'_{n-1} \\ f'_n \end{array} \right\} = \\
& = \left\{ \begin{array}{c} \frac{3}{\mu_1^2} (\vec{b}_{0,2} - \vec{b}_{0,1}) \\ \frac{3}{\mu_1^2} (\vec{b}_{0,2} - \vec{b}_{0,1}) + \frac{3}{\mu_2^2} (\vec{b}_{0,3} - \vec{b}_{0,2}) \\ \frac{3}{\mu_2^2} (\vec{b}_{0,3} - \vec{b}_{0,2}) + \frac{3}{\mu_3^2} (\vec{b}_{0,4} - \vec{b}_{0,3}) \\ \frac{3}{\mu_3^2} (\vec{b}_{0,4} - \vec{b}_{0,3}) + \frac{3}{\mu_4^2} (\vec{b}_{0,5} - \vec{b}_{0,4}) \\ \vdots \\ \frac{3}{\mu_{n-2}^2} (\vec{b}_{0,n-1} - \vec{b}_{0,n-2}) + \frac{3}{\mu_{n-1}^2} (\vec{b}_{0,n} - \vec{b}_{0,n-1}) \\ \frac{3}{\mu_{n-1}^2} (\vec{b}_{0,n} - \vec{b}_{0,n-1}) \end{array} \right\} \quad (86)
\end{aligned}$$

Ein Vergleich mit dem Gleichungssystem Gl. 62 zeigt, daß der kubische Bézier-Spline mit dem kubisch parametrischen Splines identisch ist, wenn der Parameter t_i analog (z.B. als Polygonlänge) angesetzt wird.

Somit kann man durch Gleichung 85 die Bézier-Punkte eines kubisch parametrischen Splines errechnen und ggf. die geometrisch anschaulichen Eigenschaften nutzen.

3.12 B-Spline-Kurve

Ebenso wie bei den Bézier-Kurven muß man bei den B-Splines (Basis-Splines) genau unterscheiden, ob von B-Spline-Kurven oder B-Splines die Rede ist. Die B-Spline-Kurven haben fast die gleichen Eigenschaften wie die Bézier-Kurven. Sie straken eigentlich immer, es muß kein Gleichungssystem gelöst werden, aber die Kurve läuft auch nicht durch die vorgegebenen Knoten. Das mathematische Konzept ist formal mit dem der Bézier-Kurven identisch; die Bézier-Kurven bilden genaunommen eine Untermenge der B-Spline-Kurven. Der Vorteil der B-Spline-Kurve ist, daß sie weniger Kontrollpunkte benötigt als die Bézier-Kurve. In der Rechengeschwindigkeit sind B-Spline-Kurve und Bézier-Kurve ungefähr gleichschnell.

Auch der B-Spline-Kurve liegt eine Parametrisierung zugrunde. Diese Parameter der Kontrollpunkte werden zu einem Trägervektor $T = (t_0, t_1, \dots, t_n)$ zusammengefaßt. Dabei ist eine Mehrfachnennung möglich und für die Randpunkte sogar notwendig, wenn man den Randknoten erfüllen will.

Die B-Spline-Kurve k -ter Ordnung wird durch die Gleichung 87 berechnet.

$$\vec{P}(t) = \sum_{i=0}^n \vec{d}_i N_{i,k}(t) \quad ; \quad n > k - 2 \quad ; \quad t \in [t_2, t_{n+1}] \quad (87)$$

Darin ist die B-Spline-Funktion durch Gl. 88 rekursiv zu ermitteln.

$k = 1$:

$$N_{i,1}(t) = \begin{cases} 1 & \text{für } t_i \leq t \leq t_{i+1} \\ 0 & \text{sonst} \end{cases} \quad (88)$$

$k > 1$:

$$N_{i,k}(t) = \frac{t - t_i}{t_{1+k-1} - t_i} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t) \quad (89)$$

mit $i = 0, 1, 2, 3, \dots, n$

Die B-Spline-Funktion wird im vorliegenden Fall mittels des Algorithmus von de Boor [5, 8] errechnet. Dabei wird ausgenutzt, daß für einen gegebenen Parameter $t \in [t_m, t_{m+1}]$ nur die Funktionen $N_{i,k}$ mit dem Index $i = m - (k-1), \dots, j$ ungleich Null sind.

$$\vec{P}(t) = \sum_{i=0}^{n+j} \vec{D}_i^j N_{i,k-j}(t) \quad ; \quad j = 0, 1, 2, \dots, (k-1) \quad (90)$$

$$\vec{D}_i^j = (1 - \alpha_i^j) \vec{D}_{i-1}^{j-1} + \alpha_i^j \vec{D}_i^{j-1} \quad ; \quad j > 0 \quad (91)$$

$$\vec{D}_i^0 = \vec{d}_i \quad (92)$$

$$\text{mit } \alpha_i^j = \frac{t - t_i}{t_{i+k-j} - t_i} \quad (93)$$

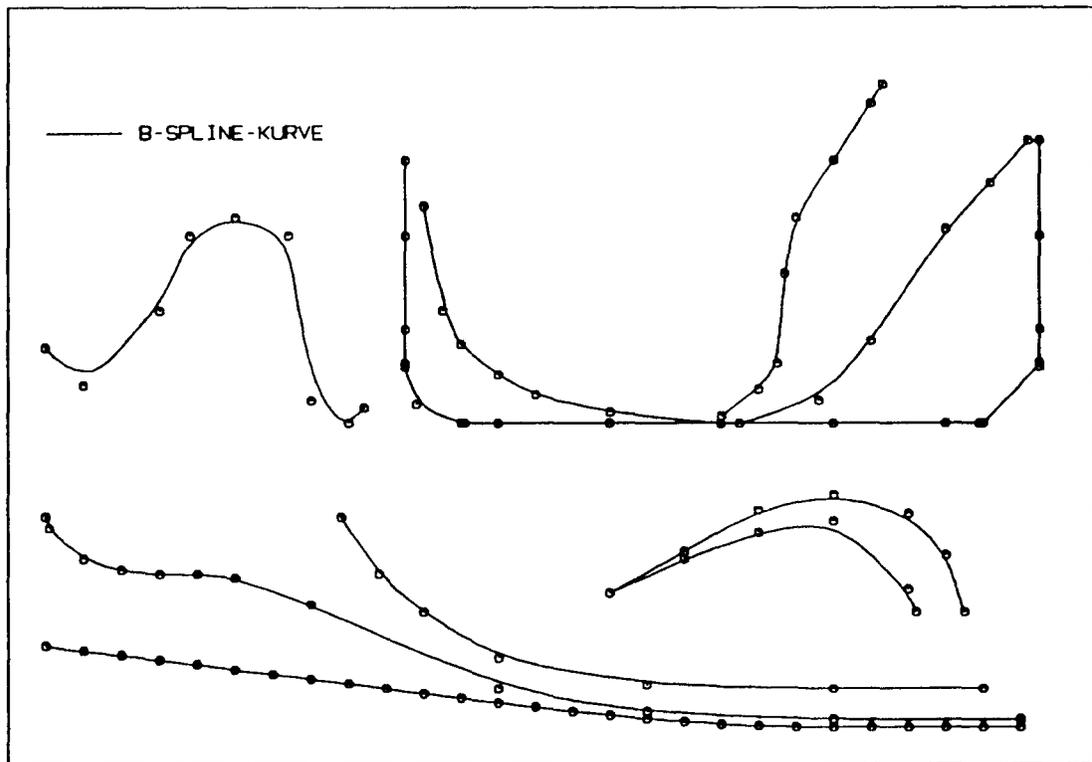


Abbildung 31: B-Spline-Kurve - Testbild

Das Testbild Abb. 31 ist der Bézier-Spline-Kurve sehr ähnlich. Es wurde mit der B-Spline-Kurve 3. Ordnung gestrakt. Eine Erhöhung der Ordnung verändert den Spline nur wenig (Abb. 32). Durch den Trägervektor hat man ein weiteres Mittel, um die Kurve zu beeinflussen: wird ein Knoten doppelt eingetragen, so wird die $(k-2)$ -te Ableitung unstetig. In der Abb. 33 wurde einmal der eingekreiste Knoten doppelt eingegeben.

Der interpolierender B-Spline

Wie schon beim Bézier-Spline, so läßt sich auch beim B-Spline ein Gleichungssystem für einen echten Spline formulieren [6]. Der Weg dorthin entspricht vollständig dem des Bézier-Splines – das Ergebnis auch. Der interpolierende B-Spline ist also ebenfalls mit dem parametrisch kubischen Spline identisch.

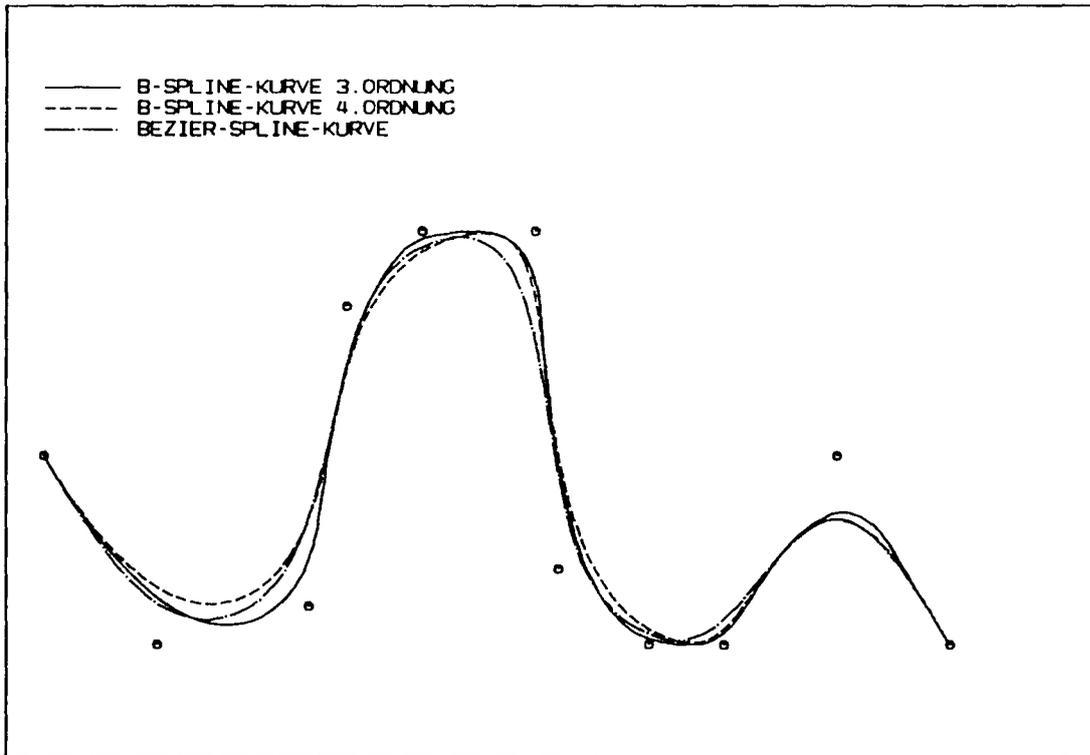


Abbildung 32: B-Spline-Kurve - Diagramm

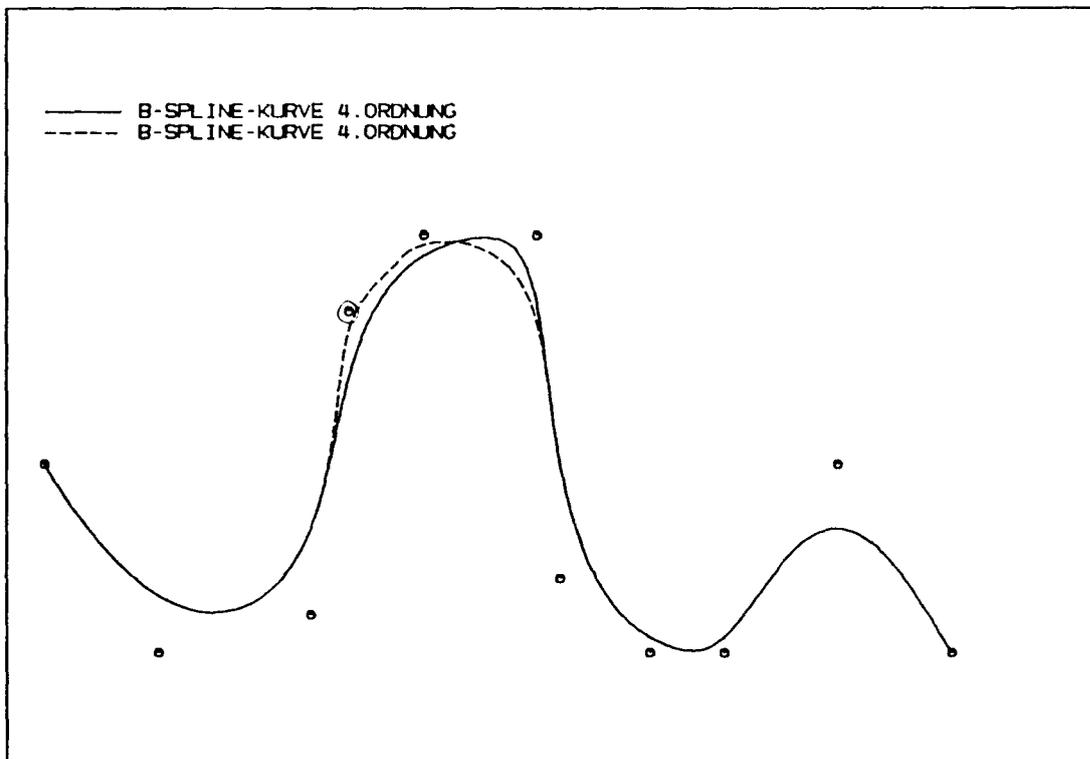


Abbildung 33: B-Spline-Kurve mit doppeltem Knoten

4 Periodische und zyklische Splines

4.1 Periodische Splines

Gelegentlich möchte man auch Punkthaufen straken, die aus irgendeiner Messung entstanden sind. Bei einer Schwingungsmessung wäre der Vorgang periodisch, d.h. $y(1)$ ist gleich $y(n)$ und auch die Randbedingung ist nun nicht mehr frei wählbar, sondern die Steigung des ersten und des letzten Punktes muß übereinstimmen

$$y_1 = y_n \quad (94)$$

$$y'_1 = y'_n \quad ; \quad y''_1 = y''_n \quad (95)$$

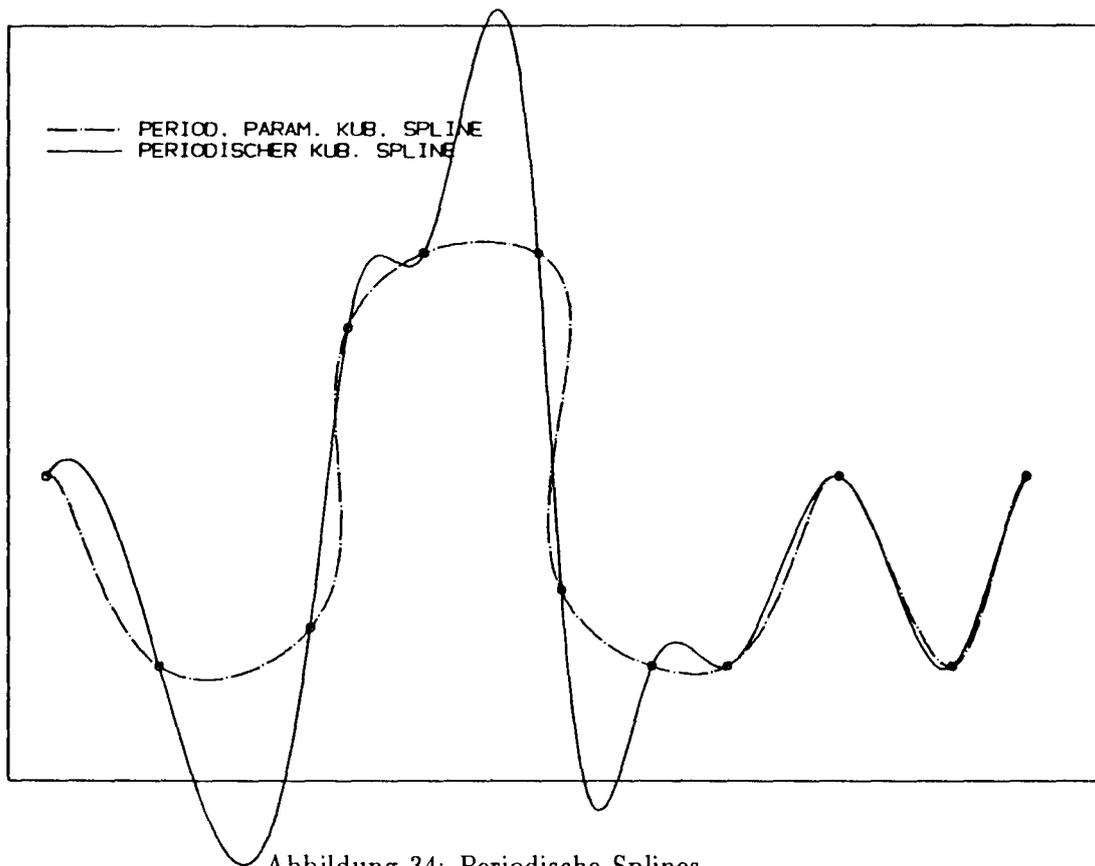


Abbildung 34: Periodische Splines

Grundsätzlich läßt sich jeder Spline-Algorithmus zu einem periodischen umbauen, man muß lediglich die Randbedingungen ändern. Das entstehende Gleichungssystem ist dann aber nicht mehr tridiagonal, da bei der Kopplung die Eckpositionen rechts-oben und links-unten besetzt werden. Die Abb. 34 zeigt zwei periodische Splines. Für die Güte der Splines gilt natürlich genau das Gleiche wie für die entsprechenden nichtperiodischen Spline-Algorithmen.

4.2 Zyklische Splines

Eine weitere Form sind zyklische Splines, d.h. die Kurven sind geschlossen. Für diese Form kommen nur Spline-Algorithmen in Frage, die auch nicht monotone Abszissen verarbeiten können. Notwendigerweise muß der erste und der letzte Punkt vollkommen identisch sein.

$$x_1 = x_n \quad ; \quad y_1 = y_n \quad (96)$$

$$y'_1 = y'_n \quad ; \quad y''_1 = y''_n \quad (97)$$

Abb. 35 zeigt zwei zyklische Spline-Algorithmen mit je zwei Kreisen. Der eine Spline wurde mit einer entsprechenden Version des parametrisch kubischen Splines erzeugt. Die Annäherung an die per Hand eingezeichneten Kreise ist trotz der geringen Punktzahl sehr gut. Der andere Spline ist der verbesserte Bézier-Spline. Hier sehen wir den Sonderfall, daß der verbesserte Bezier-Spline (mit seiner äquidistanten Parametrisierung) mit dem kubisch parametrischen Spline exakt übereinstimmt, da auch der kubisch parametrische Spline durch die Wahl der vorgegebenen Punkte äquidistant parametrisiert wurde.

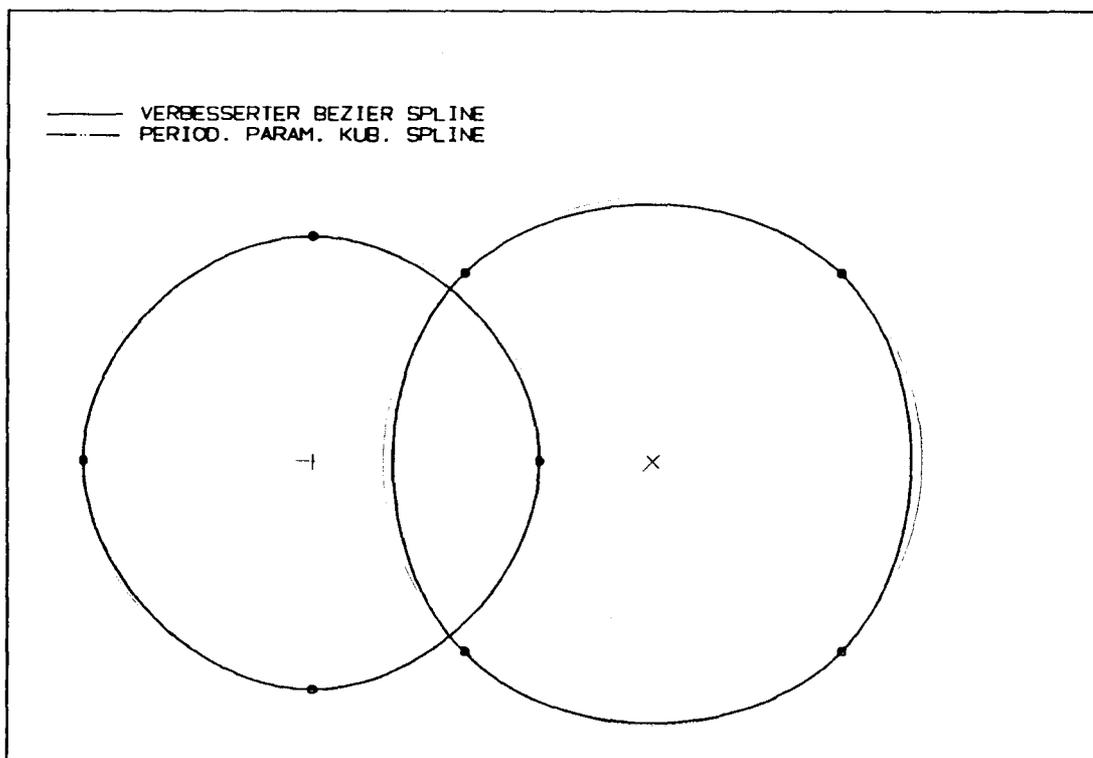


Abbildung 35: Zyklische Splines

5 Bewertung der Spline-Algorithmen

Die folgenden Tabellen geben einen Überblick über die Spline-Algorithmen.

Algorithmus PROGRAMM	Vorteile	Nachteile
Polynom POLY		<ul style="list-style-type: none"> - hoher Rechenaufwand - bei mehr als 5 Punkten kein Strak - keine Vertikale
kubischer Spline CUB1, CUB2 CUB3, CUBI	<ul style="list-style-type: none"> + geringer Rechenaufwand + i. allg. gutes Strak 	<ul style="list-style-type: none"> - Überschwingen bei großer Steigung - keine Vertikale
Hermiteischer Spline 3. Grades DEFY1	<ul style="list-style-type: none"> + sehr geringer Rechenaufwand + befriedigender Strak + 1. Ableitung kann vorgegeben werden 	<ul style="list-style-type: none"> - 2. Ableitung unstetig - keine Vertikale
Hermiteischer Spline 5. Grades QDEF	<ul style="list-style-type: none"> + 1. Ableitung kann vorgegeben werden 	<ul style="list-style-type: none"> - Schlangenlinie, kein Strak - keine Vertikale
exponentieller Spline SPLEXP	<ul style="list-style-type: none"> + guter Strak 	<ul style="list-style-type: none"> - keine Vertikalen - Parameter müssen per Hand eingegeben werden
rationaler Spline RAT + PARA	<ul style="list-style-type: none"> + sehr guter Strak 	<ul style="list-style-type: none"> - keine Vertikalen - langsam

Tabelle 1: Bewertung der Spline-Algorithmen – Funktionen : $y = f(x)$

Aus der Tabelle 1 und 2 kann man die Vorzüge und Nachteile der verschiedenen Spline-Algorithmen entnehmen. Tabelle 1 enthält die Funktionen, Tabelle 2 die Relationen. **Ich empfehle für einfache Straks den kubischen Spline CUB1**, vorausgesetzt, die Punkte liegen garantiert strakfreundlich: also keine großen Steigungsunterschiede angrenzender Intervalle, keine sehr großen Steigungen und möglichst äquidistante Abszissen.

Liegen die Knoten nicht strakfreundlich, aber der Spline muß eine echte Funktion $y = f(x)$ werden, so kann der rationale Spline RAT zusammen mit der automatischen Parametrisierung PARA die Überschwinger vermeiden und - auf Kosten der Rechenzeit - einen saubereren Strak liefern.

Algorithmus PROGRAMM	Vorteile	Nachteile
kubischer Spline mit mitgeführtem Koordi- natensystem CUBVAR	+ sehr guter Strak + geringer Rechenaufwand + keine aufsteigenden Abszissen notwendig	- entartet bei extremen Knotenkonfigurationen
parametrisch kub. Spline CUBPAR	+ sehr guter Strak + keine aufsteigenden Abszissen + gibt Kreise gut wieder	
Bézier-Spline-Kurve BEZIER	+ sehr guter Strak + sehr geringer Rechenaufwand + keine aufsteigenden Abszissen	- Kurve läuft neben Punkten
verbesserter Bézier-Spline VBEZ	+ sehr guter Strak + geringer Rechenaufwand + keine aufsteigenden Abszissen	
B-Spline-Kurve 3.Ordn. BS3	+ sehr guter Strak + sehr geringer Rechenaufwand + keine aufsteigenden Abszissen	- Kurve läuft neben Punkten
B-Spline-Kurve 4.Ordn. BS4	+ sehr guter Strak + sehr geringer Rechenaufwand + keine aufsteigenden Abszissen	- Kurve läuft neben Punkten
Bézier-Spline B-Spline	siehe parametrisch kubischer Spline siehe parametrisch kubischer Spline	

Tabelle 2: Bewertung der Spline-Algorithmen –
parametrisierte Funktionen : $x = f(t)$; $y = f(t)$

Ist die Anordnung der Punkte noch nicht bekannt, so empfehle ich den parametrisch kubischen Spline CUBPAR . Er ist unübertroffen betriebssicher und rechnet recht schnell. Auf jeden Fall ist eine Parametrisierung von Vorteil.

Der kubische Spline mit mitgeführten Koordinaten CUBVAR erfüllt die Bedingung der absoluten Betriebssicherheit nicht (vgl. Abb. 36).

Die B-Spline-Kurve ist der Bézier-Kurve überlegen, da sie näher an den Kontrollpunkten liegt und sie ist in der Anwendung flexibler.

Gegen die B-Spline-Kurve spricht, daß sie die vorgegebenen Punkte nicht einhält. Gewiß kann man sich an diese Eigenart gewöhnen, aber es ist eben

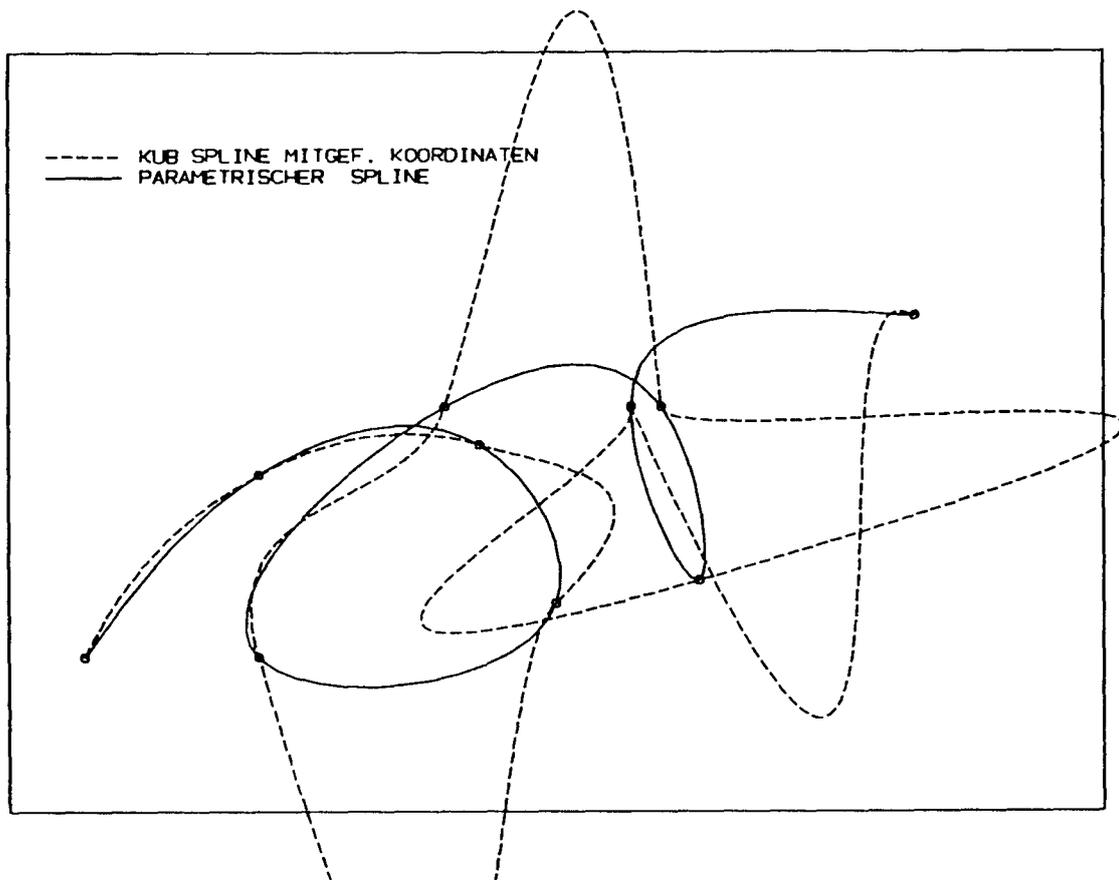


Abbildung 36: kubischer Spline mit mitgef. Koordinatensystem

gewöhnungsbedürftig. Dies wird auch nicht durch ihre Rechengeschwindigkeit aufgewogen.

Der verbesserte Bézier-Spline erfüllt die Knoten, ist dafür aber auch wieder langsamer. Dennoch wäre der verbesserte Bézier-Spline meine 2. Wahl.

Ein von verschiedenen Anwendern vorgetragenes Argument für die B-Spline-Kurve und gegen CUBPAR ist, daß eine Änderung oder das Einfügen eines Punktes nur die unmittelbare Umgebung dieses neuen Punktes verändert. Auch wird oft betont, daß ein doppelter Punkt beim Bézier-Spline einen Knick erzeugt. Beide Eigenschaften sind ebenfalls bei dieser Implementierung des parametrisch kubischen Spline vorhanden.

Ein veränderter Punkt hat praktisch immer nur Auswirkungen auf die zwei nächsten Nachbarintervall; man vergleiche hierzu Abb. 5: Die erheblich unterschiedlichen Randbedingungen führen nach zwei Intervallen auf die gleiche Kurve.

Wird zweimal derselbe Punkt eingegeben, so fügt CUBPAR ein sehr großes Stück in den Kurvenparameter t_i ein, so daß die vormals identischen Punkte über t sehr weit auseinander liegen und somit sehr weich verbunden werden. Nachdem das Gleichungssystem für die Kopplung der einzelnen Intervalle gelöst worden ist, wird dieser Zusatz im Kurvenparameter t wieder entfernt und der Strak hat seinen Knick.

Als Letztes muß unbedingt noch die "Schönheit" eines Straks angesprochen werden. Im Allgemeinen wird ein Strak als schön bezeichnet, wenn die Änderung der Krümmung klein ist $f'''(t) \rightarrow 0$. Hierbei muß man beachten, daß diese Eigenschaft nur zum Tragen kommt, wenn nicht zu viele Punkte vorgegeben werden und somit der Strak zwischen den Punkten Platz zum Leben hat. Will man nur eine existierende Kurven aufmessen, führt jeder Algorithmus, selbst ein Polygonzug, nach hinreichend vielen Aufmaßpunkten, zum gleichen Ergebnis.

Welcher Spline nun der Schönste ist, muß einjeder für sich entscheiden; die Abb. 37 und die Vorschiff-Spantrisse in den Abb. 38 bis 41 mögen ein weiterer Anhalt sein.

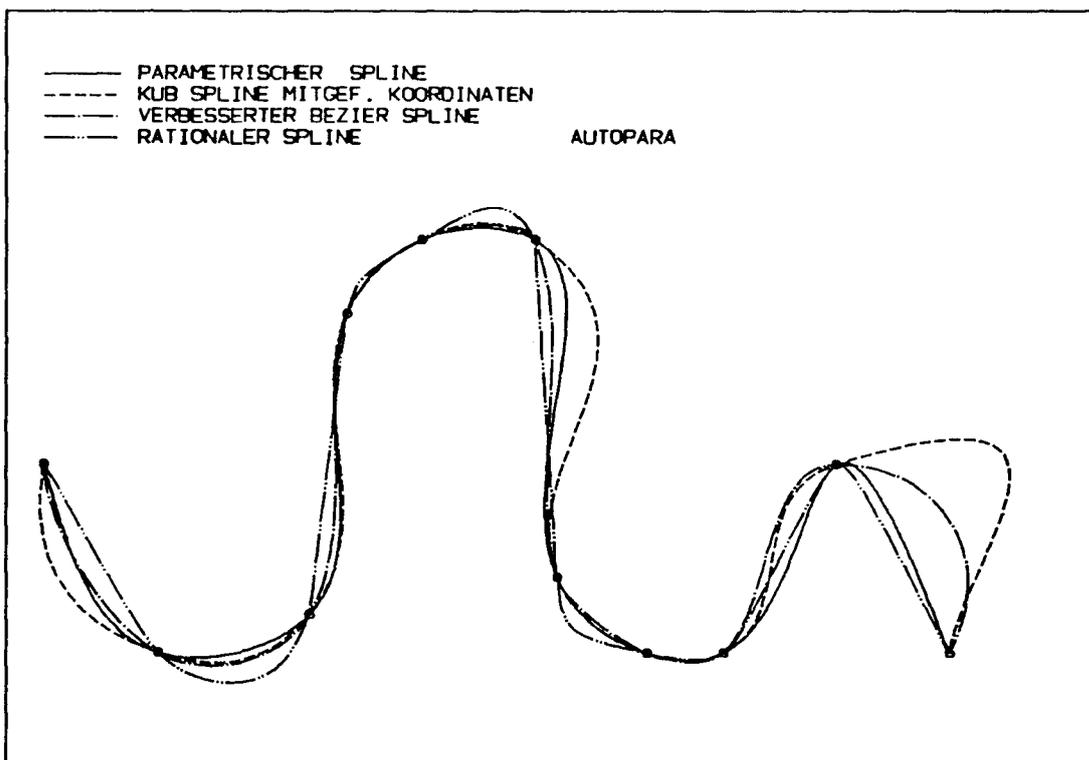


Abbildung 37: Verschiedene Spline-Algorithmen

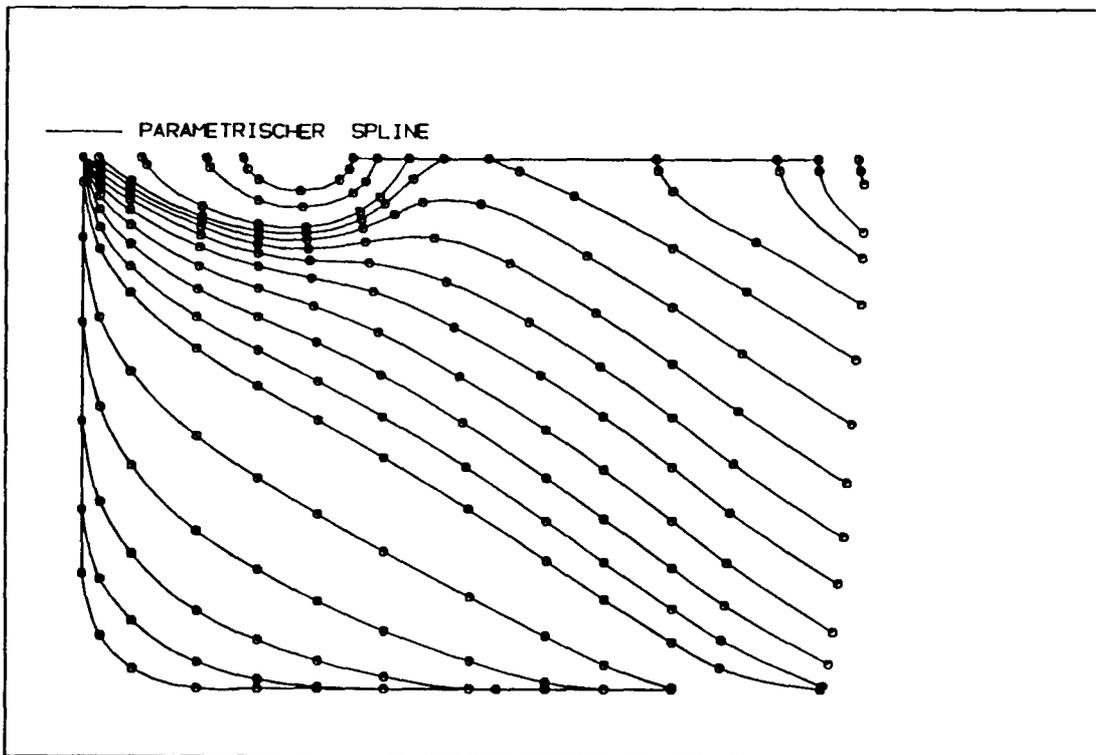


Abbildung 38: Parametrisch kubischer Spline

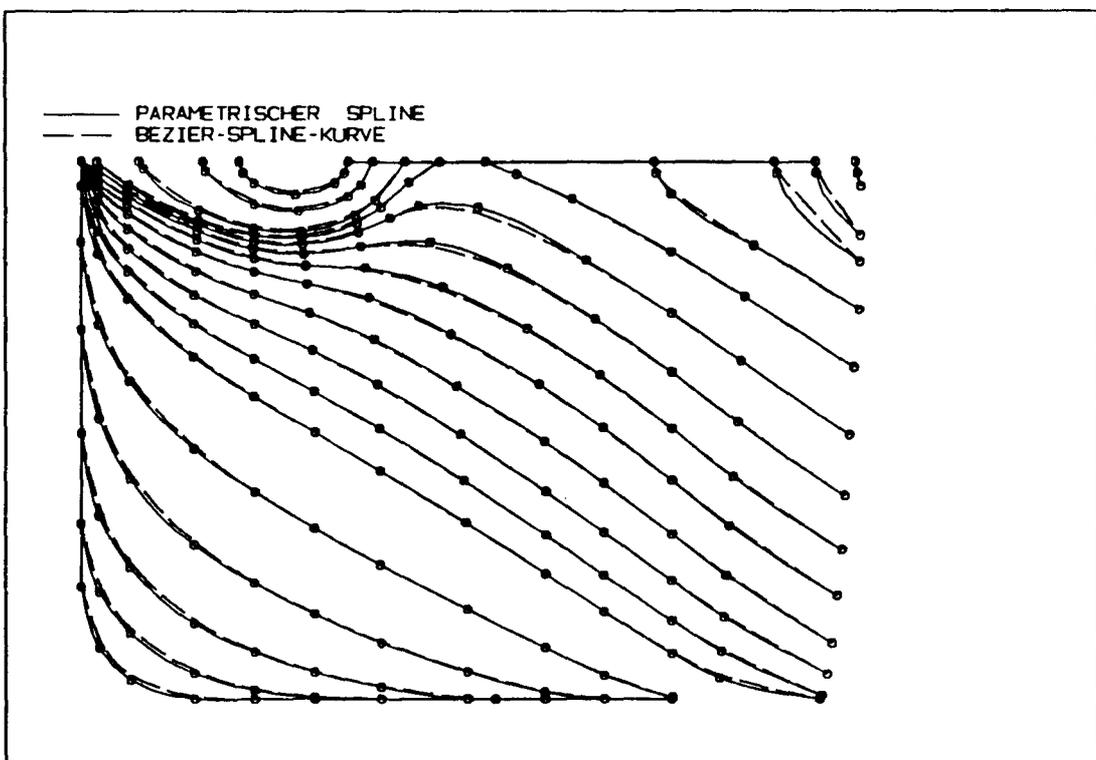


Abbildung 39: Parametrisch kubischer Spline – Bézier-Kurve

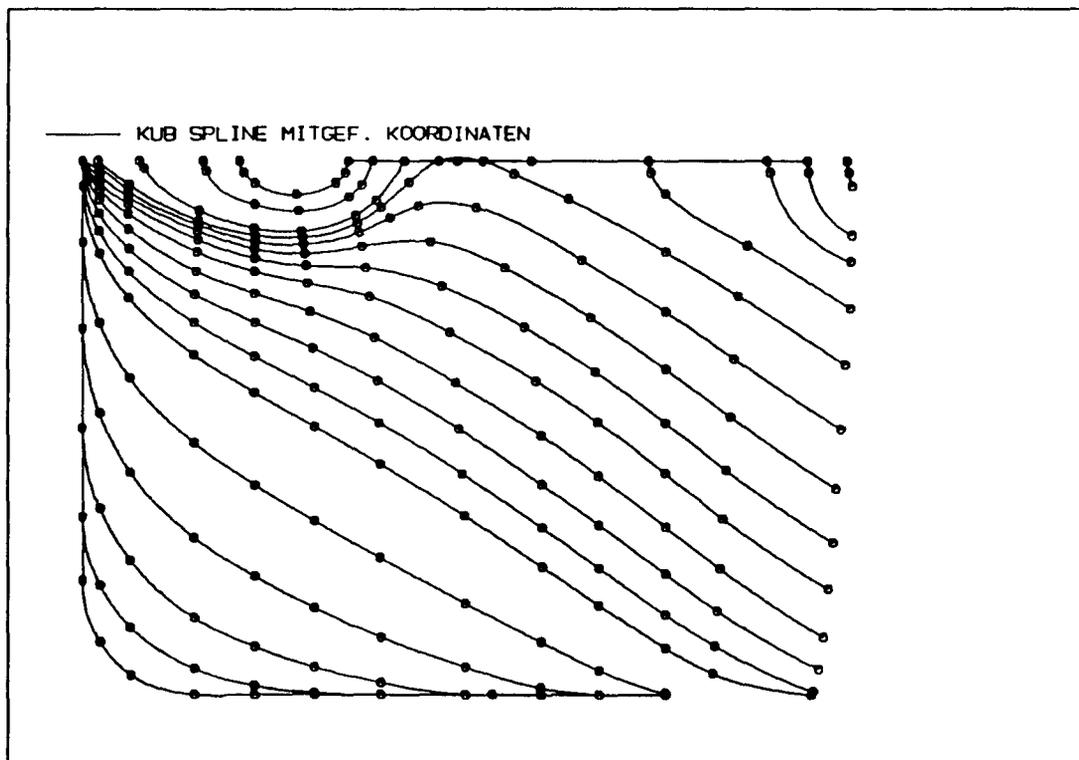


Abbildung 40: Kubischer Spline mit mitgeführtem Koordinatensystem

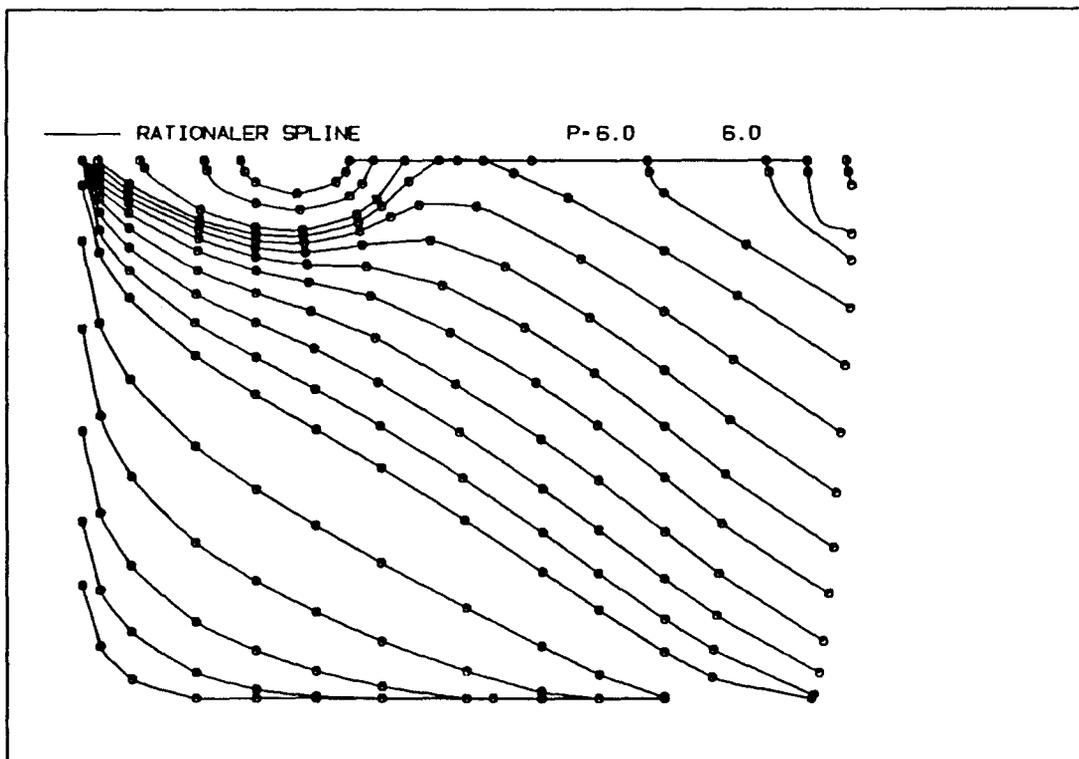


Abbildung 41: Rationaler Spline

5.1 Module zum parametrisch kubischen Spline

Da ein reines Strak-Programm allein die vielfältigen Wünsche der Anwender nicht befriedigt, hat sich seit dem Seminarvortrag im Oktober '86 eine Reihe weiterer Routinen im Umgang mit dem parametrisch kubischen Spline angesammelt (s.S. 82ff).

PROGRAM	Seite	Funktion des Moduls
CUBPAR	82	parametrisch kubischer Spline - die Grundfunktion muß einmal am Anfang eines jeden Straks aufgerufen werden. Erst danach können dann die folgenden Funktionen verwendet werden.
CUBPIT	83	Strakt mit fünf Iterationen über der Bogenlänge. Wird diese Funktion nicht aufgerufen, so läuft der Strak über der Polygonzuglänge.
CUB1	52	kubischer Spline - diese Subroutine wird von CUBPAR aufgerufen und tritt für den Benutzer nicht in Erscheinung.
PLCUBP	85	plottet den Spline mittels der Calcomp-Plotsoftware, die vom Benutzer separat zur Verfügung gestellt werden muß.
FCUBP	86	berechnet <i>eine</i> Interpolationsstelle im Strak $y_0 = f(x_0)$. Eingeeben wird dabei die Abszisse x_0 , ausgegeben die Ordinate y_0 und der Kurvenparameter t_0 an dieser Stelle.
TCUBP	87	berechnet <i>alle</i> Interpolationsstellen im Strak mit $x_0 = f(t_j)$ bzw. $y_0 = f(t_j)$; $j = 0, 1, \dots$. Ausgegeben wird t_j
DCUBP	88	berechnet für einen gegebenen Kurvenparameter t_0 die Funktionwerte $y_0 = f(t_0)$ und $x_0 = f(t_0)$, sowie die Ableitungen $dy(t_0)/dx$ und $dx(t_0)/dy$.
NULL3		eine Hilfsroutine
NULL		- " -
NODOUB		- " -
CUBITEST	92	Ein kleines Hauptprogramm zum Testen.

6 Programme

6.1 Programmaufruf am Institut für Schiffbau

Die hier vorgestellten Unterprogramme befinden sich in der frei zugänglichen VAX-Library \$DISK3:[130011.LIB]BUSCHLIB.OLB . Diese Library muß beim Linken mit aufgeführt werden [9].

Beispiel: Das Programm TEST.FOR greift auf ein Spline-Unterprogramm zurück. Zunächst wird TEST ganz normal compiliert

```
$ FORTRAN TEST
```

und dann zusammen mit der Library gelinkt

```
$ LINK/MAP TEST,$DISK3:[130011.LIB]BUSCHLIB.OLB/LIBRARY
```

Als Kontrolle kann man nun in TEST.MAP sehen, ob alle gewünschten Unterprogramme aus der richtigen Library gelinkt wurden.

6.2 Übersicht über die verwendeten Routinen

Es folgt eine Übersicht und Kurzbeschreibung der verwendeten Spline-Programme. Der Source-Code wurde direkt in diese Textdatei hineinkopiert und muß (was muß schon ?) daher funktionieren. (Auf einer VAX und auf einem Atari laufen die Routinen unter FORTRAN-77 , wenn die Name auf die mageren 6 Buchstaben gekürzt werden.)

Für die Plotaufrufe ist es notwendig, daß die beiden Calcomp-Routinen PLOTS und PLOT vom Benutzer zur Verfügung gestellt werden oder entsprechender Ersatz in meine Routine DASHAB (s.S. 70) eingebaut wird.

Die einzelnen Parameter werden zu Beginn jeden Programms kurz erläutert, diese Hinweise sind unbedingt gewissenhaft zu lesen.

Die Angabe $X(*)$ bedeutet dabei, daß X ein Feld mit einer Feldlänge gleich der Anzahl n der Punkte ist. Der Hinweis IN/OUT bedeutet, daß die Variable beim Eintritt in das Unterprogramm definiert sein muß, bzw. im Unterprogramm definiert wird.

Tabelle 3: Programmtexte der Splines

PROGRAMM (Parameter)	Seite	Bezeichnung
POLY (N,X,Y)	52	ein Polynom über alle Punkte
CUB1 (N,X,Y,Y2,F,G)	* 52	kubischer Spline RB: $y_1'' = 0$
CUB2 (N,X,Y,Y2,F,G)	*	kubischer Spline RB: $y_1'' = u \cdot y_2''$
CUB3 (N,X,Y,Y2,F,G)	*	kubischer Spline RB: $y_1' = 0$
PLCUB (N,X,Y,Y2)	53	plottet kubischen Spline
CUBI (N,X,Y,Y1,F,G)	54	kubischer Spline RB: $y_1' = c$
PLCUBI (N,X,Y,Y1)	55	plottet CUBI
CUBA (N1,X,DX,Z)	*	kub. Spline mehrere Ordinatenansätze
CUBB (N1,DX,Y,Y1,F,Z)	*	- " -
DEFY1 (N,X,Y,Y1)	56	definiert y' durch Parabel
PLY1 (N,X,Y,Y1)	56	plottet y' durch kurzen Strich
QDEF (N,X,Y,Y1,A,B,C,D)	*	Hermiteischer Spline 5.Grades
PLQDEF (N,X,Y,Y1,A,B,C,D)	57	plottet QDEF
GENSPL (N,X,Y,A,B,C,D,Y1,F,G)	*	verallgemeinerter kub. Spline
SPLEXP (N,X,Y,P,Y1,A,B,C,D)	*	exponentieller Spline
PLEXP (N,X,Y,P,Y1,A,B,C,D)	??	plottet SPLEXP
RAT (N,X,Y,P,Q,Y1,A,B,C,D)	* 58	rationaler Spline
PARA (X,Y,P,Q,N1)	60	Parameterberechnung für RAT
PLRAT (N,X,Y,P,Q,A,B,C,D)	59	plottet RAT
RATA (N1,P,X,DX,Z)	*	rat. Spline mehrere Ordinatenansätze
RATB (N1,P,DX,Y,Y1,F,Z)	*	- " -
RATC (N1,P,X,Y,Y1,A,B,C,D)	*	- " -
CUBV1 (N,X,Y,R,S,A,T,F,V)	61	liest Randbed. für CUBVAR
CUBVAR (N,X,Y,R,S,A,T,F,V)	61	kubischer Spline mitgeführtes Koord.
PLCUBV (N,X,Y,R,S,A,T)	63	plottet CUBVAR
CUBPAR (N,T,X,X2,Y,Y2,F,G)	82ff	parametrischer kubischer Spline
PLCUBP (N,T,X,X2,Y,Y2)	85	plottet CUBPAR
BEZIER (N,X,Y,B,T)	63	Bézier-Kurve
VBEZ (N,X,Y,B,S)	64	verbesserter Bézier-Spline
PLBEZ (N,X,Y,B,S)	66	plottet Bézier-Spline
BS3 (N,T,X,Y)	66	B-Spline 3.Ordnung
BS4 (N,T,X,Y)	68	B-Spline 4.Ordnung
PLBS3 (N,T,X,Y)	67	plottet B-Spline 3.Ordnung
PLBS4 (N,T,X,Y)	69	plottet B-Spline 4.Ordnung
CUBP (N,X,Y,Y2,F,G,H)	*	periodischer kubischer Spline
CYCLIC (N,T,X,X2,Y,Y2,F,G,H)	*	zyklischer parametr. kub. Spline
SEMI	71ff	Testprogramm

* Programm in H. Spaeth [1]

6.3 FORTRAN 77 - Source Codes

```

      SUBROUTINE POLY (N,X,Y)
C *****
C
C POLY LEGT EIN POLYNOM N-1 TEN GRADES DURCH DIE PUNKTE
C UND PLOTTET SIE
C MAX. 30 PUNKTE
C
C AXEL BUSCH
C
C UNTERPROGRAMME: GL
C                DASHAB
C
C N      - IN  - ANZAHL DER PUNKTE
C X(*)   - IN  - X-KOORDINATEN
C Y(*)   - IN  - Y-KOORDINATEN
C
C
C      REAL X(*),Y(*)
C      REAL B(30,30),A(30)
C      DO 1 K=1,N
C        B(K,1)=1.
C      DO 1 I=2,N
1     B(K,I)=X(K)**(I-1)
C      CALL GL (B,Y,A,N)
C
C
C      CALL DASHA(X(1),Y(1),3)
C      XX=X(1)
2     IF(XX.GE.(X(N)-.05)) GOTO 4
C      F=A(1)
C      DO 3 K=2,N
3     F=F+A(K)*XX**(K-1)
C      CALL DASHA (XX,F,2)
C      XX=XX+.05
C      GOTO 2
4     CALL DASHA (X(N),Y(N),2)
C      RETURN
C      END

      SUBROUTINE CUB1(N,X,Y,Y2,F,G)
C *****
C
C H.SPAETH, 'SPLINE-ALGORITHMEN ZUR KONSTRUKTION GLATTER KURVEN
C UND FLAECHEEN', OLDENBURG-VERAG 1978
C
C KUBISCHER SPLINE, UNBEKANNTE Y'' , RANDWERTE Y''(1),Y''(N)
C
C H.SPAETH / A.BUSCH
C
C N      - IN  - ANZAHL DER PUNKTE

```

```

C X(*) - IN - X-KOORDINATEN DER PUNKTE
C Y(*) - IN - Y-KOORDINATEN DER PUNKTE
C Y2(1) - IN - 2. ABLEITUNG AM PUNKT 1
C Y2(N) - IN - 2. ABLEITUNG AM PUNKT N
C Y2(*) - OUT - 2. ABLEITUNG AN DEN PUNKTEN
C F(*) - - ARBEITSFELD
C G(*) - - ARBEITSFELD
C
C

```

```

      REAL X(*),Y(*),Y2(*),F(*),G(*)
      N1=N-1
      G(1)=0.
      F(1)=0.
      DO 2 K=1,N1
      J2=K+1
      H2=X(J2)-X(K)
      R2=(Y(J2)-Y(K))/H2
      IF (K.EQ.1) GOTO 1
      Z=1./(2.*(H1+H2)-H1*G(J1))
      G(K)=Z*H2
      H=6.*(R2-R1)
      IF (K.EQ.2) H=H-H1*Y2(1)
      IF (K.EQ.N1) H=H-H2*Y2(N)
      F(K)=Z*(H-H1*F(J1))
1      J1=K
      H1=H2
      R1=R2
2      CONTINUE
      Y2(N1)=F(N1)
      IF (N1.LE.2) RETURN
      N2=N-2
      DO 3 J1=2,N2
      K=N-J1
      Y2(K)=F(K)-G(K)*Y2(K+1)
3      CONTINUE
      RETURN
      END

```

```

      SUBROUTINE PLCUB(N,X,Y,Y2)
C *****
C
C PLOTTET KUBISCHE SPLINES
C
C AXEL BUSCH
C
C UNTERPROGRAMME : DASHAB
C
C N - IN - ANZAHL DER PUNKTE
C X(*) - IN - X-KOORDINATEN DER PUNKTE
C Y(*) - IN - Y-KOORDINATEN DER PUNKTE
C Y2(*) - IN - 2. ABLEITUNG AN DEN PUNKTEN
C
C

```

```

REAL X(2),Y(2),Y2(2)
CALL DASHA (X(1),Y(1),3)
N1=N-1
DO 2 K=1,N1
DX=X(K+1)-X(K)
DY=Y(K+1)-Y(K)
      A=(Y2(K+1)-Y2(K))/6./DX
      B=Y2(K)/2.
      C=DY/DX-DX/6.*(Y2(K+1)+2.*Y2(K))
      D=Y(K)
XX=0.
1   IF (XX.GT.(X(K+1)-X(K))) GOTO 2
    F=A*XX*XX*XX+B*XX*XX+C*XX+D
    CALL DASHA ((X(K)+XX),F,2)
    XX=XX+ 0.05
    GOTO 1
2   CONTINUE
    CALL DASHA (X(N),Y(N),2)
    RETURN
END

```

```

SUBROUTINE CUBI (N,X,Y,Y1,F,G)
C *****
C
C H.SPAETH, 'SPLINE-ALGORITHMEN ZUR KONSTRUKTION GLATTER KURVEN
C UND FLAECHEEN', OLDENBURG-VERAG 1978
C
C KUBISCHER SPLINE , UNBEKANNTE Y' , RANDBED. Y'(1),Y'(N)
C
C H.SPAETH / A.BUSCH
C
C N - IN - ANZAHL DER PUNKTE
C X(*) - IN - X-KOORDINATEN DER PUNKTE
C Y(*) - IN - Y-KOORDINATEN DER PUNKTE
C Y1(1) - IN - 1. ABLEITUNG AM PUNKT 1
C Y1(N) - IN - 1. ABLEITUNG AM PUNKT N
C Y1(*) - OUT - 1. ABLEITUNG AN DEN PUNKTEN
C F(*) - - ARBEITSFELD
C G(*) - - ARBEITSFELD
C
C
C
REAL X(*),Y(*),Y1(*),F(*),G(*)
N1=N-1
G(1)=0.
F(1)=0.
DO 2 K=1,N1
J2=K+1
H2=1./(X(J2)-X(K))
R2=3.*H2*H2*(Y(J2)-Y(K))
IF (K.EQ.1) GOTO 1
Z=1./(2.*(H1+H2)-H1*G(J1))
G(K)=Z*H2
H=R1+R2

```

```

      IF (K.EQ.2) H=H-H1*Y1(1)
      IF (K.EQ.N1) H=H-H2*Y1(N)
      F(K)=Z*(H-H1*F(J1))
1     J1=K
      H1=H2
      R1=R2
2     CONTINUE
      Y1(N1)=F(N1)
      IF (N1.LE.2) RETURN
      N2=N-2
      DO 3 J1=2,N2
      K=N-J1
      Y1(K)=F(K)-G(K)*Y1(K+1)
3     CONTINUE
      RETURN
      END

```

```

      SUBROUTINE PLCUBI (N,X,Y,Y1)
C *****
C
C PLOTTET KUBISCHE SPLINES VON X , Y , Y'
C
C AXEL BUSCH
C
C UNTERPROGRAMME : DASHAB
C
C N      - IN  - ANZAHL DER PUNKTE
C X(*)   - IN  - X-KOORDINATEN DER PUNKTE
C Y(*)   - IN  - Y-KOORDINATEN DER PUNKTE
C Y1(*)  - IN  - 1. ABLEITUNG AN DEN PUNKTEN
C
C
      REAL X(*),Y(*),Y1(*)
      CALL DASHA(X(1),Y(1),3)
      N1=N-1
      DO 2 K=1,N1
      DX=X(K+1)-X(K)
      DY=Y(K+1)-Y(K)
      A=(-2.*DY/DX+Y1(K)+Y1(K+1))/DX/DX
      B=(3.*DY/DX-2.*Y1(K)-Y1(K+1))/DX
      C=Y1(K)
      D=Y(K)
      XX=0.
1     IF (XX.GT.(X(K+1)-X(K))) GOTO 2
      F=A*XX*XX*XX+B*XX*XX+C*XX+D
      CALL DASHA((X(K)+XX),F,2)
      XX=XX+ 0.05
      GOTO 1
2     CONTINUE
      CALL DASHA(X(N),Y(N),2)
      RETURN
      END

```

```

      SUBROUTINE DEFY1 (N,X,Y,Y1)
C *****
C
C DEFINIERT DIE ERSTE ABLEITUNG Y'(K) DURCH EINE PARABEL
C DURCH DIE PUNKTE K-1,K,K+1
C AXEL BUSCH
C
C N - IN - ANZAHL DER PUNKTE
C X(*) - IN - X-KOORDINATEN
C Y(*) - IN - Y-KOORDINATEN
C Y1(*)- OUT - 1. ABLEITUNG
C
C
      REAL X(*),Y(*),Y1(*)
      DY1=Y(2)-Y(1)
      DX1=X(2)-X(1)
      Y1(1)=DY1/DX1
      N1=N-1
      DO 1 K=2,N1
      DY2=Y(K+1)-Y(K)
      DX2=X(K+1)-X(K)
      Y1(K)=(DX2/DX1*DY1+DX1/DX2*DY2)/(DX1+DX2)
      DY1=DY2
      DX1=DX2
1 CONTINUE
      Y1(N)=DY2/DX2
      RETURN
      END

```

```

      SUBROUTINE PLY1 (N,X,Y,Y1)
C *****
C
C PLOTTET DIE ERSTE ABLEITUNG AN DEN PUNKTEN
C
C AXEL BUSCH
C
C N - IN - ANZAHL DER PUNKTE
C X(*) - IN - X-KOORDINATEN
C Y(*) - IN - Y-KOORDINATEN
C Y1(*)- IN - 1. ABLEITUNG
C
C
C
      REAL X(*),Y(*),Y1(*)
      DX=.06
      DO 1 I=1,N
      CALL PLOT ((X(I)-DX),(Y(I)-Y1(I)*DX),3)
      CALL PLOT ((X(I)+DX),(Y(I)+Y1(I)*DX),2)
1 CONTINUE
      RETURN
      END

```

```

      SUBROUTINE PLQDEF (N,X,Y,Y1,A,B,C,D)
C *****
C
C PLOTTET HERMITISCHE SPLINES 5.GRADES
C
C AXEL BUSCH
C
C UNTERPROGRAMME : DASHAB
C
C N      - IN  - ANZAHL DER PUNKTE
C X(*)   - IN  - X-KOORDINATEN DER PUNKTE
C Y(*)   - IN  - Y-KOORDINATEN DER PUNKTE
C Y1(*)  - IN  - 1. ABLEITUNG AN DEN PUNKTEN
C A(*)   - IN  - KOEFFIZIENTEN AUS QDEF
C B(*)   - IN  - KOEFFIZIENTEN AUS QDEF
C C(*)   - IN  - KOEFFIZIENTEN AUS QDEF
C D(*)   - IN  - KOEFFIZIENTEN AUS QDEF
C
      REAL X(*),Y(*),Y1(*),A(*),B(*),C(*),D(*)
      CALL DASHA (X(1),Y(1),3)
      N1=N-1
      DO 2 K=1,N1
      XX=0.
1       IF (XX.GE.(X(K+1)-X(K))) GOTO 2
      F=A(K)*XX**5+B(K)*XX**4+C(K)*XX**3+D(K)*XX*XX+Y1(K)*XX+Y(K)
      CALL DASHA ((X(K)+XX),F,2)
      XX=XX+ 0.05
      GOTO 1
2       CONTINUE
      CALL DASHA (X(N),Y(N),2)
      RETURN
      END

      SUBROUTINE PLSPLEXP (N,X,Y,P,A,B,C,D)
C *****
C
C PLOTTET EXPONENTIELLE SPLINES
C
C AXEL BUSCH
C
C UNTERPROGRAMME : DASHAB
C
C
C
C N      - IN  - ANZAHL DER PUNKTE
C X(*)   - IN  - X-KOORDINATEN DER PUNKTE
C Y(*)   - IN  - Y-KOORDINATEN DER PUNKTE
C P(*)   - IN  - KURVENPARAMETER AUS SPLEXP
C A(*)   - IN  - KOEFFIZIENTEN
C B(*)   - IN  - KOEFFIZIENTEN
C C(*)   - IN  - KOEFFIZIENTEN

```

```

C D(*) - IN - KOEFFIZIENTEN
C
REAL X(*),Y(*),P(*),A(*),B(*),C(*),D(*)
PHI(F,G)=(SINH(F*G)-G*SINH(F))/(SINH(F)-F)
N1=N-1
CALL DASHA(X(1),Y(1),3)
DO 2 K=1,N1
T=0.
DX=X(K+1)-X(K)
DT=.05/DX
1 IF (T.GE.1.) GOTO 2
F=A(K)*(1.-T)+B(K)*T+C(K)*PHI(P(K),(1-T))+D(K)*PHI(P(K),T)
CALL DASHA ((X(K)+T*DX),F,2)
T=T+DT
GOTO 1
2 CONTINUE
CALL DASHA(X(N),Y(N),2)
RETURN
END

```

```

SUBROUTINE RAT (N,X,Y,P,Q,Y1,A,B,C,D)

```

```

C *****
C
C H.SPAETH, 'SPLINE-ALGORITHMEN ZUR KONSTRUKTION GLATTER KURVEN
C UND FLAECHEEN', OLDENBURG-VERAG 1978
C
C SPEZIELLE RATIONALE SPLINE-FUNKTION MIT PARAMETERN P UND Q
C
C H. SPAETH / A. BUSCH
C
C N - IN - ANZAHL DER PUNKTE
C X(*) - IN - X-KOORDINATEN DER PUNKTE
C Y(*) - IN - Y-KOORDINATEN DER PUNKTE
C P(*) - IN - PARAMETER P, Q = 0 :KUB. SPLINE
C Q(*) - IN - PARAMETER P, Q >1000 :POLYGONZUG
C Y1(1) - IN - 1. ABLEITUNG AM PUNKT 1
C Y1(N) - IN - 1. ABLEITUNG AM PUNKT N
C Y1(*) - OUT - 1. ABLEITUNG AN DEN PUNKTEN
C A(*) - OUT - KOEFFIZIENTEN
C B(*) - OUT - KOEFFIZIENTEN
C C(*) - OUT - KOEFFIZIENTEN
C D(*) - OUT - KOEFFIZIENTEN
C

```

```

REAL X(*),Y(*),Y1(*),P(*),Q(*),A(*),B(*),C(*),D(*)
N1=N-1
C(1)=0.
D(1)=0.
DO 2 K=1,N1
J2=K+1
PP=P(K)
QQ=Q(K)
PP2=PP*(PP+3.)+3.
QQ2=QQ*(QQ+3.)+3.

```

```

P22=PP+2.
Q22=QQ+2.
A(K)=X(J2)-X(K)
H=1./A(K)
B(K)=1./(P22*Q22-1.)
H2=H*B(K)
R2=H*H2*(Y(J2)-Y(K))
IF (K.EQ.1) GOTO 1
HQ=H1*QQ1
HP=H2*PP2
Z=1./(HQ*(P21-C(J1))+HP*Q22)
C(K)=Z*HP
H=R1*QQ1*(1.+P21)+R2*PP2*(1.+Q22)
IF (K.EQ.2) H=H-HQ*Y1(1)
IF (K.EQ.N1)H=H-HP*Y1(N)
D(K)=Z*(H-HQ*D(J1))
1   J1=K
P21=P22
QQ1=QQ2
H1=H2
R1=R2
2   CONTINUE
Y1(N1)=D(N1)
IF (N1.LE.2) GOTO 4
N2=N1-1
DO 3 J1=2,N2
K=N-J1
Y1(K)=D(K)-C(K)*Y1(K+1)
3   CONTINUE
4   DO 5 K=1,N1
J2=K+1
H=B(K)*(Y(J2)-Y(K))
Z=B(K)*A(K)
P2=P(K)+2.
Q2=Q(K)+2.
C(K)= (1.+Q2)*H-Z*(Y1(J2)+Q2*Y1(K))
D(K)=- (1.+P2)*H+Z*(P2*Y1(J2)+Y1(K))
A(K)=Y(K)-C(K)
B(K)=Y(J2)-D(K)
5   CONTINUE
RETURN
END

```

```

SUBROUTINE PLRAT (N,X,Y,P,Q,A,B,C,D)
C *****
C
C PLOTTET RATIONALE SPLINES
C
C AXEL BUSCH
C
C UNTERPROGRAMME : DASHAB
C
C

```

```

C N      - IN  - ANZAHL DER PUNKTE
C X(*)   - IN  - X-KOORDINATEN DER PUNKTE
C Y(*)   - IN  - Y-KOORDINATEN DER PUNKTE
C P(*)   - IN  - KURVENPARAMETER AUS RAT
C Q(*)   - IN  - KURVENPARAMETER AUS RAT
C A(*)   - IN  - KOEFFIZIENTEN AUS RAT
C B(*)   - IN  - KOEFFIZIENTEN AUS RAT
C C(*)   - IN  - KOEFFIZIENTEN AUS RAT
C D(*)   - IN  - KOEFFIZIENTEN AUS RAT
C
C
      REAL X(*),Y(*),P(*),Q(*),A(*),B(*),C(*),D(*)
      PHI(F,G)=G*G*G/(F*(1.-G)+1.)
      N1=N-1
      CALL DASHA(X(1),Y(1),3)
      DO 2 K=1,N1
      T=0.
      DX=X(K+1)-X(K)
      DT=.05/DX
1      IF (T.GE.1.) GOTO 2
      F=A(K)*(1.-T)+B(K)*T+C(K)*PHI(P(K),(1-T))+D(K)*PHI(Q(K),T)
      CALL DASHA ((X(K)+T*DX),F,2)
      T=T+DT
      GOTO 1
2      CONTINUE
      CALL DASHA(X(N),Y(N),2)
      RETURN
      END

      SUBROUTINE PARA (X,Y,P,Q,N1)
C*****
C
C PARA ERZEUGT AUTOMATISCH DIE PARAMETER FUER RATIONALE SPLINES
C
C AXEL BUSCH          10.86
C
C X(*) - IN  - X-KOORDINATEN DER PUNKTE
C Y(*) - IN  - Y-KOORDINATEN DER PUNKTE
C P(*) - OUT - PARAMETER P FUER RAT
C Q(*) - OUT - PARAMETER Q FUER RAT
C N1  - IN  - ANZAHL DER PUNKTE
C
C
      REAL X(*),Y(*),P(*),Q(*)
      DO 10 I=2,N1
      P(I)=ABS((Y(I+1)-Y(I))/(X(I+1)-X(I))-(Y(I)-Y(I-1))/
+
      (X(I)-X(I-1)))
      P(I)=P(I)*1.2
10     CONTINUE
C
      DO 20 I=2,N1
20     Q(I-1)=P(I)
      P(1)=P(2)

```

```

Q(N1)=Q(N1-1)
RETURN
END

```

```

      SUBROUTINE CUBV1(N,X,Y,R,S,A,T,F,V)
C *****
C
C   EINGABE DER RANDBEDINGUNGEN FUER CUBVAR
C
C   AXEL BUSCH
C
C   N      - IN  - ANZAHL DER PUNKTE
C   X(*)   - IN  - X-KOORDINATEN DER PUNKTE
C   Y(*)   - IN  - Y-KOORDINATEN DER PUNKTE
C   S(1)   - OUT - U1      Y''(1)=U1*Y''(2)
C   R(N-1) - OUT - U2      Y''(N)=U2*Y''(N-1)
C   R(*)   -     - KOEFFIZIENT AUS CUBVAR
C   S(*)   -     - KOEFFIZIENT AUS CUBVAR
C   A(*)   -     - WINKEL DER FELDER AUS CUBVAR
C   T(*)   -     - 1./LAENGE DER FELDER AUS CUBVAR
C   F(*)   -     - ARBEITSFELD
C   V(*)   -     - ARBEITSFELD
C
      REAL X(*),Y(*),R(*),S(*),A(*),T(*),F(*),V(*)
      N1=N-1
C   PRINT*, 'RANDBEDINGUNGEN FUER DEN ANFANG EINGEBEN (1=TANGENTE)'
C   IL=IPICKSET(0.,5)
      IL=0
      IF(IL.EQ.1) THEN
        PRINT*, 'TANGENTE AM ANFANG =?'
        R(1)=PICK(0.)-ATAN2((Y(2)-Y(1)),(X(2)-X(1)))
      ELSE
        PRINT*, 'KRUEMMUNGSVERHAELTNIS AM ANFANG = ?'
        S(1)=PICK(0.)
      ENDIF
C   PRINT*, 'RANDBEDINGUNGEN FUER DAS ENDE EINGEBEN'
C   IR=IPICKSET(0.,5)
      IR=0
      IF (IR.EQ.1) THEN
        PRINT*, 'TANGENTE AM ENDE =?'
        S(N1)=PICK(0.)-ATAN2((Y(N)-Y(N1)),(X(N)-X(N1)))
      ELSE
        PRINT*, 'KRUEMMUNGSVERHAELTNIS AM ENDE'
        R(N1)=PICK(0.)
      ENDIF
C   CALL CUBVAR (N,X,Y,R,S,A,T,F,V,IL,IR)
      CALL CUBVAR (N,X,Y,R,S,A,T,F,V)
      RETURN
      END

      SUBROUTINE CUBVAR (N,X,Y,R,S,A,T,F,V)

```

```

C *****
C
C KUBISCHE SPLINES MIT MITGEFUEHRTEM KOORDINATENSYSTEM
C
C (c) AXEL BUSCH      10.86 / 6.90
C
C N      - IN  - ANZAHL DER PUNKTE
C X(*)   - IN  - X-KOORDINATEN DER PUNKTE
C Y(*)   - IN  - Y-KOORDINATEN DER PUNKTE
C S(1)   - IN  - Y''(1)= S(1) *Y''(2) ; 0. <= S(1) <= 1.
C R(N-1) - IN  - Y''(N)= R(N-1) *Y''(N-1) ; 0. <= R(N-1) <= 1.
C R(*)   - OUT - KOEFFIZIENT AUS CUBVAR
C S(*)   - OUT - KOEFFIZIENT AUS CUBVAR
C A(*)   - OUT - WINKEL DER FELDER AUS CUBVAR
C T(*)   - OUT - 1./LAENGE DER FELDER AUS CUBVAR
C F(*)   -     - ARBEITSFELD
C V(*)   -     - ARBEITSFELD
C
      REAL X(*),Y(*),R(*),S(*),A(*),T(*),F(*),V(*)
      U1=S(1)
      U2=R(N1)
      N1=N-1
      Q=0.
      H2=0.
      F(1)=0.
      V(1)=0.
      DO 1 K=1,N1
          DX=X(K+1)-X(K)
          DY=Y(K+1)-Y(K)
          A(K)=ATAN2(DY,DX)
          T(K)=1./SQRT(DX*DX+DY*DY)
1      CONTINUE
      DX=T(N1)
      A(N)=0.
      DO 2 K=1,N1
          H1=T(K)*(A(K)-A(K+1))
          Z=2.*(Q+T(K))
          IF (K.EQ.1) Z=T(1)*(2.+U1)/(1.+2.*U1)
          IF (K.EQ.N1) Z=2.*Q+T(N1)*(2.-(1.+2.*U2)/(2.+U2))
          Z=1./(Z-Q*V(K))
          IF (K.EQ.N1) T(N1)=0.
          V(K+1)=Z*T(K)
          IF (K.EQ.N1) H1=0.          ! 25.6.90
          F(K+1)=Z*((H1+H2)-Q*F(K))
          H2=2.*H1
          Q=T(K)
2      CONTINUE
      T(N1)=DX
      R(N1)=F(N)
      S(N1)=-R(N1)*(1.+2.*U2)/(2.+U2)
      DO 3 K=N1,2,-1
          R(K-1)=F(K)-V(K)*R(K)
          S(K-1)=R(K)+A(K)-A(K-1)
3      CONTINUE
C

```

```

C KORREKUR FUER STETIGE 1. ABLEITUNG
C
      DO 4 K=1,N1
      R(K)=TAN(R(K))
4     S(K)=TAN(S(K))
      RETURN
      END

```

```

      SUBROUTINE PLCUBV (N,X,Y,R,S,A,T)
C *****
C
C PLOTTET KUBISCHE SPLINES MIT MITGEFUEHRETM KOORDINATENSYSTEM
C
C AXEL BUSCH
C
C UNTERPROGRAMME : DASHAB
C
C N      - IN  - ANZAHL DER PUNKTE
C X(*)   - IN  - X-KOORDINATEN DER PUNKTE
C Y(*)   - IN  - Y-KOORDINATEN DER PUNKTE
C R(*)   - IN  - KOEFFIZIENT AUS CUBVAR
C S(*)   - IN  - KOEFFIZIENT AUS CUBVAR
C A(*)   - IN  - WINKEL DER FELDER AUS CUBVAR
C T(*)   - IN  - 1./LAENGE DER FELDER AUS CUBVAR
C
      REAL X(*),Y(*),R(*),S(*),A(*),T(*)
      N1=N-1
      CALL DASHA(X(1),Y(1),3)
      DO 2 K=1,N1
      DT= .05 * T(K)
      T(K)=1./T(K)
      TT=0.
      CA=COS(A(K))
      SA=SIN(A(K))
1     IF (TT.GE.(1.-DT)) GOTO 2
      E=T(K)*(TT*(1.-TT)*(R(K)*(1-TT)-S(K)*TT))
      XP=X(K)+TT*T(K)*CA-E*SA
      YP=Y(K)+TT*T(K)*SA+E*CA
      CALL DASHA (XP,YP,2)
      TT=TT+DT
      GOTO 1
2     CONTINUE
      CALL DASHA(X(N),Y(N),2)
      RETURN
      END

```

```

      SUBROUTINE BEZIER (N,X,Y,B,T)
C *****
C

```

```

C   BERECHNET DIE BEZIER-VEKTOREN B
C
C   (c) AXEL BUSCH
C
C   N       - IN  - ANZAHL DER PUNKTE
C   X(*)    - IN  - X-KOORDINATEN DER PUNKTE
C   Y(*)    - IN  - Y-KOORDINATEN DER PUNKTE
C   B(*)    - IN  - BEZIER-PUNKTE   ! COMPLEX !
C   T(*)    - OUT - ARBEITSFELD     ! COMPLEX !
C
      REAL X(*),Y(*)
C   MINDEST-DIMENSION !! B(0:3*N1) ,T(0:N1)
      COMPLEX B(0:*),T(0:*)
      N1=N-1
      N2=N-2
      DO 10 K=1,N
10    T(K-1)=CMLPX (X(K),Y(K))
C
      B(2)=(T(0)+2*T(1))/3.
      DO 20 K=1,N2
      B(3*K-2)=(2.*T(K-1)+T(K))/3.
      B(3*K)  =(T(K-1)+4.*T(K)+T(K+1))/6.
      B(3*K+2)=(T(K)+2.*T(K+1))/3.
20    CONTINUE
      B(3*N1-2)=(2.*T(N2)+T(N1))/3.
C
      B(0)=T(0)
      B(3*N1)=T(N1)
      RETURN
      END

```

```

      SUBROUTINE VBEZ (N,X,Y,B,S)
C   *****
C
C   VERBESSERTER BEZIER SPLINE
C   - DIE PUNKTE WERDEN ERFUELLT
C
C   (c) AXEL BUSCH           10.86
C
C   N       - IN  - ANZAHL DER PUNKTE
C   X(*)    - IN  - X-KOORDINATEN DER PUNKTE
C   Y(*)    - IN  - Y-KOORDINATEN DER PUNKTE
C   B(*)    - OUT - BEZIER-PUNKTE   COMPLEX !
C   S(*)    -     - ARBEITSFELD     COMPLEX !
C
      REAL X(*),Y(*)
      COMPLEX B(*),S(*),V1,V2,V3
      N1=N-1
      N2=N-2
      DO 10 I=1,N
10    B(I*3-2)=CMLPX(X(I),Y(I))
C

```

```

DO 20 I=2,N2
F=(X(I+1)-X(I))*(Y(I)-Y(I+2))-(X(I)-X(I+2))*(Y(I+1)-Y(I))
G=((X(I-1)-X(I+1))*(Y(I)-Y(I+2))-(X(I)-X(I+2))*(Y(I-1)
*   -Y(I+1)))
  IF (ABS(G).LT.1E-8) THEN
    G=0.1
    GOTO 17
  ENDIF
F=F/G
C
  IF (ABS(X(I)-X(I+2)).LT.1E-3) THEN
    G=((Y(I+1)-Y(I))+F*(Y(I+1)-Y(I-1)))/(Y(I)-Y(I+2))
    GOTO 17
  ENDIF
  G=((X(I+1)-X(I))+F*(X(I+1)-X(I-1)))/(X(I)-X(I+2))
17 IF(F.GE.0..OR.G.GE.0.) THEN
  V1=B(3*I+1)-B(3*I-5)
  V2=B(3*I+1)-B(3*I-2)
  V3=B(3*I-2)-B(3*I+4)
  F1=SQRT((V2*CONJG(V2))/(V1*CONJG(V1)))/3.
  F2=SQRT((V2*CONJG(V2))/(V3*CONJG(V3)))/3.
  B(3*I-1)=B(3*I-2)+F1*V1
  B(3*I) =B(3*I+1)+F2*V3
  GOTO 20
  ENDIF
S(I)=B(3*I-2)+F*(B(3*I-5)-B(3*I+1))
18 B(3*I-1)=(B(3*I-2)+S(I))/2.
  B(3*I) =(B(3*I+1)+S(I))/2.
C
20 CONTINUE
C
C
V1=(B(4)-B(1))*(B(4)-B(1))/(B(7)-B(1))
F=REAL(B(4)-B(1))*AIMAG(B(7)-B(1))-AIMAG(B(4)-B(1))*
*   REAL(B(7)-B(1))
G=REAL(V1)*AIMAG(B(7)-B(1))-AIMAG(V1)*REAL(B(7)-B(1))
IF (ABS(G).LT.1E-8) THEN
  S(1)=(B(1)+B(4))/2.
  GOTO 30
ENDIF
F= ABS(F/G)
S(1)=B(1)+F*V1
C
30 I=N
V1=(B(I*3-5)-B(I*3-2))*(B(I*3-5)-B(I*3-2))/(B(I*3-8)-B(I*3-2))
V2=B(I*3-2) -B(I*3-8)
F=REAL(B(I*3-5)-B(I*3-2))*AIMAG(V2)-AIMAG(B(I*3-5)-
*   B(I*3-2))*REAL(V2)
G=REAL(V1)*AIMAG(V2)-AIMAG(V1)*REAL(V2)
IF (ABS(G).LT.1E-8) THEN
  S(N1)=(B(3*I-2)+B(3*I-5))/2.
  GOTO 40
ENDIF
F=F/G
S(N1)=B(I*3-2)+F*V1

```

66

```
C
40  B(2)=(B(1)+S(1))/2.
      B(3)=(B(4)+S(1))/2.
      B(3*N1-1)=(B(3*N1-2)+S(N1))/2.
      B(3*N1)  =(B(3*N1+1)+S(N1))/2.
      RETURN
      END
```

SUBROUTINE PLBEZ (N,X,Y,B)

```
C *****
C
C PLOTTET BEZIER-SPLINES
C
C AXEL BUSCH
C
C UNTERPROGRAMME : DASHAB
C
C N      - IN  - ANZAHL DER PUNKTE
C X(*)   - IN  - X-KOORDINATEN DER PUNKTE
C Y(*)   - IN  - Y-KOORDINATEN DER PUNKTE
C B(*)   - IN  - BEZIER-PUNKTE AUS ROUTINE BEZIER ! COMPLEX !
C
C
      REAL X(*),Y(*)
      COMPLEX B(0:*),P
      N2=N-2
      CALL DASHA (X(1),Y(1),3)
C
      DO 20 K=0,N2
      K3=K*3
      V=0.
      DX=X(K+2)-X(K+1)
      DY=Y(K+2)-Y(K+1)
      DV=.1
      IF(DX.NE.0. .OR. DY.NE.0.) DV=.1/SQRT(DX*DX+DY*DY)
C
10     V=V+DV
      V1=1.-V
      IF(V.GT.1) GOTO 20
      P=B(K3)*V1*V1*V1+3.*B(K3+1)*V1*V1*V+3.*B(K3+2)*V1*V*V+
*   B(K3+3)*V*V*V
      PX=REAL(P)
      PY=AIMAG(P)
      CALL DASHA(PX,PY,2)
      GOTO 10
20    CONTINUE
C
      CALL DASHA(X(N),Y(N),2)
      RETURN
      END
```

```

      SUBROUTINE BS3(N,T,X,Y)
C *****
C
C (C) AXEL BUSCH      7.90
C
C ERZEUGT EINEN TRAEGERVEKTOR FUER B-SPLINE-KURVE 3. ORDNUNG
C
C N      - IN      - ANZAHL DER PUNKTE
C T(**+6) - OUT    - ARBEITSFELD MIT TRAEGERVEKTOR DIMENSION (N+4)!
C X(**+1) - IN/OUT - X-KOORDINATEN DIMENSION (N+1)
C Y(**+1) - IN/OUT - Y-KOORDINATEN DIMENSION (N+1)
C
      REAL T(0:*),X(0:*),Y(0:*)
      T(0)=0
      T(1)=0
      T(2)=0
      N1=N-1
      X(N)=X(N1)
      Y(N)=Y(N1)

      DO 10 I=1,N1
      DX=X(I)-X(I-1)
      DY=Y(I)-Y(I-1)
      T(I+2)=T(I+1)
      IF( DX.NE..0 .OR. DY.NE..0) T(I+2)=T(I+2)+SQRT(DX*DX+DY*DY)
10     CONTINUE

      T(N+2)=T(N+1)
      T(N+3)=T(N+1)
      RETURN
      END

```

```

      SUBROUTINE PLBS3(N,T,X,Y)
C *****
C
C (C) AXEL BUSCH      7.90
C
C PLOTET B-SPLINE-KURVE 3. ORDNUNG NACH ALGORITHMUS V. DE BOOR
C N      - IN      - ANZAHL DER PUNKTE
C T(**+6) - IN      - ARBEITSFELD MIT TRAEGERVEKTOR DIMENSION (N+6)!
C X(*)    - IN      - X-KOORDINATEN
C Y(*)    - IN      - Y-KOORDINATEN
C
      REAL T(0:*),X(0:*),Y(0:*)
      CALL DASHA(X(0),Y(0),3)
      DO 100 I=2, N
      TT=0.
20     IF(TT.GE.(T(I+1)-T(I))) GOTO 90

      A = (TT +T(I)-T(I-1))/(T(I+1)-T(I-1))
      XXO =X(I-2)*(1-A) + X(I-1)*A
      YYO =Y(I-2)*(1-A) + Y(I-1)*A

```

```

      A=TT/ (T(I+2)-T(I))
      XX1 =X(I-1)*(1-A) + X(I)*A
      YY1 =Y(I-1)*(1-A) + Y(I)*A

      A=TT/ (T(I+1)-T(I))
      XXX = XX0*(1-A) + XX1*A
      YYY = YYO*(1-A) + YY1*A

      CALL DASHA( XXX, YYY, 2)
      TT=TT +.1
      GOTO 20
90    CALL DASHA(XXX,YYY,3)

100   CONTINUE
      CALL DASHA( X(N-1),Y(N-1),2)
      RETURN
      END

      SUBROUTINE BS4(N,T,X,Y)
C *****
C
C (C) AXEL BUSCH      10.7.90
C
C ERZEUGT EINEN TRAEGERVEKTOR FUER B-SPLINE-KURVE 4. ORDNUNG
C
C N      - IN      - ANZAHL DER PUNKTE
C T(**+6) - OUT    - ARBEITSFELD MIT TRAEGERVEKTOR DIMENSION (N+6)!
C X(**+1) - IN/OUT - X-KOORDINATEN DIMENSION (N+1)
C Y(**+1) - IN/OUT - Y-KOORDINATEN DIMENSION (N+1)
C
      REAL T(0:*),X(0:*),Y(0:*)
      T(0)=0
      T(1)=0
      T(2)=0
      T(3)=0
      N1=N-1
      X(N)=X(N1)
      X(N+1)=X(N1)
      Y(N)=Y(N1)
      Y(N+1)=Y(N1)

      DO 10 I=1,N1
      DX=X(I)-X(I-1)
      DY=Y(I)-Y(I-1)
      T(I+3)=T(I+2)
      IF( DX.NE..0 .OR. DY.NE..0) T(I+3)=T(I+3)+SQRT(DX*DX+DY*DY)
10    CONTINUE

      T(N+3)=T(N+2)
      T(N+4)=T(N+2)
      T(N+5)=T(N+2)
      RETURN
      END

```

```

      SUBROUTINE PLBS4(N,T,X,Y)
C *****
C
C (C) AXEL BUSCH      7.90
C
C PLOTTET B-SPLINE-KURVE 4. ORDNUNG NACH ALGORITHMUS V. DE BOOR
C
C N      - IN      - ANZAHL DER PUNKTE
C T(**+6) - IN      - ARBEITSFELD MIT TRAEGERVEKTOR  DIMENSION (N+6)!
C X(*)    - IN      - X-KOORDINATEN
C Y(*)    - IN      - Y-KOORDINATEN
C
      REAL T(0:*),X(0:*),Y(0:*)
      A(J,I)= (T0-T(I))/(T(I+4-J)-T(I))

      CALL DASHA(X(0),Y(0),3)
      DO 100 I=3, N+1
      TO= T(I)
20      IF(T0.GE.T(I+1)) GOTO 90

      X2= (1-A(1,I-2))*X(I-3) + A(1,I-2)*X(I-2)
      X1= (1-A(1,I-1))*X(I-2) + A(1,I-1)*X(I-1)
      X0= (1-A(1,I) ) *X(I-1) + A(1,I)*X(I)

      XX1=(1-A(2,I-1))*X2 + A(2,I-1)*X1
      XX0=(1-A(2,I) ) *X1 + A(2,I) *X0

      XXX=(1-A(3,I) ) *XX1 + A(3,I)*XX0

      Y2= (1-A(1,I-2))*Y(I-3) + A(1,I-2)*Y(I-2)
      Y1= (1-A(1,I-1))*Y(I-2) + A(1,I-1)*Y(I-1)
      Y0= (1-A(1,I) ) *Y(I-1) + A(1,I)*Y(I)

      YY1=(1-A(2,I-1))*Y2 + A(2,I-1)*Y1
      YY0=(1-A(2,I) ) *Y1 + A(2,I) *Y0

      YYY=(1-A(3,I) ) *YY1 + A(3,I)*YY0

      CALL DASHA( XXX, YYY, 2)
      TO=TO +.1
      GOTO 20
90      CALL DASHA(XXX,YYY,3)

100     CONTINUE

      CALL DASHA( X(N-1),Y(N-1),2)
      RETURN
      END

```

```

SUBROUTINE DASHAB (X,Y,NPAR,ARRAY,NARDUM)
C
C (C) A. BUSCH 7.86 <12.88>
C
C REVISION 21.6.90 ENGERE TEILUNG IN ALLEN STD-STRICHEN
C
C DASHAB ERZEUGT STRICH-PUNKTIERTE/GESTRICHELTE LINIEN
C IM ERSTEN TEIL DES PROGRAMMS WIRD NUR DER STRICHTYP GESETZT,
C NACH DEM ENTRY DASHA(X,Y,NPAR) DIE LINIE ERZEUGT.
C FOLGENDE LINIENTYPEN SIND ALS VOREINSTELLUNG AUFRUFBAR:
C NARDUM= -1 -----
C          -2 -----
C          -3 .....
C          -4 .....
C          -5 .....
C          LE-6 .....
C
C X,Y - KOORDINATEN DES NAECHSTEN ANZUFAHRENDEN PUNKTES
C NPAR - WIE BEI CALCOMP-PLOT =2 STIFT UNTEN
C                               3 STIFT OBEN
C                               999 PLOT SCHLIESSEN
C ARRAY(NARDUM) - STRICHLAENGEN BIS ZUM ALTERNIEREN DES
C STIFTMODES
C NARDUM - S.OBEN ,
C          ODER (BEI EIGENEN STRICHEN) DIE ANZAHL IN ARRAY
C
C UNTERPROGRAMME : PLOT (X,Y,IMODE)
C ENTRY          : DASHA(X,Y,IMODE)
C - - - - -
C
C REAL ARRAY(*),A(0:16),L(4,8)
C DIMENSION IL(6)
C SAVE
C DATA (L(1,I),I=1,2)/10.,10./ ! VOLLINIE
C DATA (L(2,I),I=1,2)/.4,.5/ ! LANG GESTRICHELT
C DATA (L(3,I),I=1,2)/.15,.2/ ! GEPUNKTET
C          STRICH-PUNKT-PKT-PKT
C DATA (L(4,I),I=1,8)/.4,.44,.46,.5,.52,.56,.58,.62/
C DATA (IL(I),I=1,6) /2,2,2,4,6,8/
C VOREINSTELLUNG AUF DURCHGEZOGENE LINIE
C DATA A(1)/10.,A(2)/10.,IE/2/,XA/0./,YA/0./,T/0./,I/1/
C ITEST(I)= I-INT(FLOAT(I)/2.+0.01)*2
C NAR=NARDUM
C T=0.
C CALL PLOT (X,Y,NPAR)
C XA=X
C YA=Y
C IF (NAR.LT.0) THEN
C   IF(NAR.LT.-6) NAR= -6
C   NAR=-NAR
C   IE=IL(NAR)
C   IF(NAR.GT.4) NAR=4

```

```

      DO 1 I=1,IE
1      A(I)=L(NAR,I)
      I=1
      RETURN
ENDIF
IF (NAR.EQ.0) NAR=1
A(0)=0.
DO 5 I=1,NAR
5      A(I)=ARRAY(I)+A(I-1)
      IE=NAR
      I=1
      RETURN
C
      ENTRY DASHA(X,Y,NPAR)
      IF (IABS(NPAR).NE.2) THEN
          CALL PLOT(X,Y,NPAR)
          XA=X
          YA=Y
          I=1
          T=0.
          RETURN
      ENDIF
      DX=X-XA
      DY=Y-YA
      DT=SQRT(DX*DX+DY*DY)
      IF (DT.LT.0.0000001) RETURN
      SX=DX/DT
      SY=DY/DT
      N=0
C
10     IF((T+DT-A(IE)*N).LT.A(I)) THEN
          CALL PLOT (X,Y,(3-ITEST(I)))
          T=T+DT-A(IE)*N
          XA=X
          YA=Y
          RETURN
      ENDIF
      XP=XA+(A(I)+A(IE)*N-T)*SX
      YP=YA+(A(I)+A(IE)*N-T)*SY
      CALL PLOT (XP,YP,(3-ITEST(I)))
      I=I+1
      IF(I.LE.IE) GOTO 10
      I=1
      N=N+1
      GOTO 10
      END

```

C LETZTE AENDERUNG 22.6.90 16.00H

C*****

```

      PROGRAM SEMI
      COMMON/DIM/IDIM
      REAL X(30),X2(30),Y(30),Y1(30),Y2(30),F(30),G(30),T(30)

```

```

REAL A(30),B(30),C(30),D(30)
REAL P(30),Q(30)
COMPLEX CB(120),CT(30)
REAL PU(3,3),K1(3)
CHARACTER YN*1
IDIM=30

C
OPEN (7,FILE='SEMI.DAT',STATUS='OLD')

CALL PLOTS (0,0,13)
CALL NEWPEN(3)
1 CONTINUE
C CALL AXIS(0.,0.,'X',-1,26., 0.,0.,1.)
C CALL AXIS(0.,0.,'Y', 1,16.,90.,0.,1.)

CALL PLOT(1.3,1.3,-3)
CALL PLOT (14.5,0.,2)
CALL PLOT (14.5,10.,2)
CALL PLOT (0.,10.,2)
CALL PLOT (0.,0.,2)

C
C STARTWERT FUER DIE UEBERSCHRIFT
UE=8.

C
5 WRITE (6,1010)
1010 FORMAT (////
*, ' WELCHER SPLINE-TYP SOLL BENUTZT WERDEN ?' /
*, ' =====' /
*, ' 0 - POLY - EIN POLYNOM N-1 TEN GRADES' /
*, ' 1 - CUB1 - KUBISCHER SPLINE (RB = GERADER AUSLAUF)' /
*, ' 2 - CUB2 - KUBISCHER SPLINE (RB = KREIS)' /
*, ' 3 - CUB3 - KUBISCHER SPLINE (RB = TANGENTE )' /
*, ' 4 - CUBI - KUBISCHER SPLINE Y1 (RB = TANGENTE)' /
*, ' 5 - CUBPAR - PARAMETR. KUB. SPLINE (RB=GERADER AUSLAUF)' /
*, ' 6 - QDEF - HERMITISCHER SPLINE 5.GRADES' /
*, ' 7 - DEFY1 - HERMITISCHER SPLINE 3.GRADES' /
*, ' 9 - CUBVAR - MITGEFUEHRTER KUB. SPLINE' /
*, ' 10 - BEZIER - BEZIER-SPLINES' /
*, ' 11 - SPLEXP - EXPONNTIELLER SPLINE ( RB = Y1 )' /
*, ' 12 - PASPL1 - RATIONALER SPLINE (RB = Y1 )' /
*, ' 15 - CUBP - PERIODISCHER KUB. SPLINE (RB =TANGENTE)' /
*, ' 16 - CICLIC - ZYKLISCHER PARAMETR. KUB. SPLINE' /
C *, ' 17 - UEBER - UEBERTRAGUNGSMATRIX KUB. SPLINE' /
*, ' 18 - VBEZ - VERBESSERTER BEZIER-SPLINE' /
*, ' 19 - NEUER RAHMEN '
*, ' 20 - ENDE' /)
CALL SETUNIT(5)
NR=IPICK(5)
IF (NR.LT.0.OR.NR.GT.20) GOTO5
IF (NR.EQ.19) THEN
CALL PLOT (30.,0.,-3)
GOTO 1
ENDIF
IF (NR.EQ.20) GOTO 990
8 WRITE (6,1020)

```

```

1020 FORMAT (///
*, ' LINIENTYP ?' /
*, ' =====' /
*, ' 1 = -----' /
*, ' 2 = - - - - -' /
*, ' 3 = .....' /
*, ' 4 = _ _ _ _ _' /
*, ' 5 = _ _ _ _ _' /)
READ*,LINE
IF (LINE.LT.1.OR.LINE.GT.6) GOTO 8
C
C LINIENTYP SETZEN
C
CALL DASHAB (0.,0.,3,DUM,-LINE)
C
C UEBERSCHRIFT
C
UE=UE+.35
CALL BESCHR (NR,UE)
C
C PARAMETER FUER RATIONALE SPLINES EINLESEN
C
IF (NR.GT.10.AND.NR.LT.15) THEN
PRINT*, 'PARAMETER FUER ALLE FELDER ANGEBEN '
55 PRINT*, 'P = '
P1=PICK(5.)
IF (NR.EQ.12)THEN
PRINT*, 'Q = '
Q1=PICK(5.)
ENDIF
DO 56 I=1,30
P(I)=P1
56 Q(I)=Q1
ENDIF
C
C EINLESEN DER PUNKTE
C
J=0
C
C WAHL DER KURVENSCHAR
C
9 REWIND(7)
WRITE (6,1040)
1040 FORMAT (///
*, ' KURVENSCHAR :' /
*, ' =====' /
*, ' 1 - DIAGRAMM' /
*, ' 2 - HEBELARM' /
*, ' 3 - WASSERLINIE' /
*, ' 4 - SPANTRISS' //
*, ' 6 - SPANTRISS GEDREHT' /
*, ' 7 - SPANTRISS OHNE SENKRECHTEN' /
*, ' 8 - SCHWEINESCHWANZ' /
*, ' 9 - SPANTEN MIT WENIGER PUNKTEN' //
*, ' 10 - GROSSES DIAGRAMM' //

```

```

*, ' 15 - PERIODISCHES DIAGRAMM'/
*, ' 16 - KREIS'//
*, ' 20 - ANDERER SPLINE-TYP'/
*, ' 21 - ENDE'/)
  READ*, IKURV
  IF (IKURV.LT.1.OR.IKURV.GT.21.OR.IKURV.EQ.5.OR.
* (IKURV.GE.11.AND.IKURV.LE.14)) GOTO 9
  IF (IKURV-20) 12,5,990
12  CONTINUE
  CALL SETUNIT(7)

13  I=IPICK(998.)
  c   print*, ' spline:', i
  IF(I.EQ.998) GOTO 9
  IF(I.EQ.IKURV) GOTO 10
14  XY=PICK(9999.9)
  IF (XY-9999.) 14,14,13

  C
10  I=1
20  X(I)=PICKSET(9999.9 ,7)
  IF (X(I).GT.999.) GOTO 30
  Y(I)=PICK(0.)
  c   print*, 'Punkt:', x(I), y(I)
  I=I+1
  GOTO 20

  C
30  CONTINUE
  IF (X(I).GT.9999.) GOTO 9
  J=J+1
  PRINT*, 'SPLINE ', J

  C
  N=I-1
  N1=N-1

  C
  C KORREKTUR FUER DEN STARTPUNKT
  C
  CALL DREH (X,Y,N,IKURV,NR)

  C
  C RANBED.
  C
  DO 35 I=1, IDIM
  X2(I)=0.
  Y1(I)=0.
35  Y2(I)=0.

  C
  IF (NR.EQ.5.OR.NR.EQ.9.OR.NR.EQ.10.OR.NR.EQ.16.OR.NR.EQ.18)
* GOTO 39
  IF (IKURV.EQ.4) THEN
    PRINT*, ' GEDREHTEN SPANTRISS WAEHLEN ! '
    GOTO 9
  ENDIF
  Y1(1)=(Y(2)-Y(1))/(X(2)-X(1))
  Y1(N)=(Y(N)-Y(N1))/(X(N)-X(N1))

  C
  C PUNKTE MARKIEREN

```

```

C
39  DO 40 I=1,N
    CALL SYMBOL (X(I),Y(I),.1,1,0.,-1)
40  CONTINUE
C
C  AUSWAHL  DES  ALGORHYTHMUSES
C  *****
C
    IF (NR.EQ.0) CALL POLYNOM (N,X,Y)
C
    IF (NR.EQ.1) CALL CUB1 (N,X,Y,Y2,F,G)
    IF (NR.EQ.2) CALL CUB2 (N,X,Y,Y2,F,G)
    IF (NR.EQ.3) CALL CUB3 (N,X,Y,Y2,F,G)
    IF (NR.GE.1.AND.NR.LE.3) CALL PLCUB (N,X,Y,Y2)
C
    IF (NR.EQ.4) THEN
    CALL CUBI (N,X,Y,Y1,F,G)
C    CALL CUBA (N1,X,X2,T)
C    CALL CUBB (N1,X2,Y,Y1,F,T)
    CALL PLCUBI (N,X,Y,Y1)
    ENDIF
C
    IF (NR.EQ.5) CALL CUBPAR (N,T,X,X2,Y,Y2,F,G)
    IF (NR.EQ.5) CALL PLCUBP (N,T,X,X2,Y,Y2)
C
    IF (NR.EQ.6) THEN
    CALL DEFY1 (N,X,Y,Y1)
    CALL QDEF (N,X,Y,Y1,A,B,C,D)
    CALL PLQDEF (N,X,Y,Y1,A,B,C,D)
    ENDIF
C
    IF (NR.EQ.7) THEN
    CALL DEFY1 (N,X,Y,Y1)
    CALL PLY1(N,X,Y,Y1)
    CALL PLCUBI (N,X,Y,Y1)
    ENDIF
C
    IF (NR.EQ.8) THEN
    PRINT*, 'PARAMTER = ?'
    READ*, PAR
    CALL HALB(N,X,Y,Y1,PAR)
    CALL PLCUBI (N,X,Y,Y1)
    ENDIF
C
    IF (NR.EQ.9) THEN
    CALL CUBV1 (N,X,Y,A,B,C,D,Y1,Y2)
C    A(N1) = 1.
    B(1) = 1.
    CALL CUBVAR (N,X,Y,A,B,C,D,Y1,Y2,IL,IR)
    CALL PLCUBV (N,X,Y,A,B,C,D)
    ENDIF
C
    IF (NR.EQ.10) THEN
    CALL BEZIER (N,X,Y,CB,CT)
    CALL PLBEZIER (N,X,Y,CB)

```

```

ENDIF
C
IF (NR.GE.11.AND.NR.LT.15) THEN
PRINT*, 'SONDERWUENSCHEN FÜR DIE PARAMETER (Y/N)'
READ(*,1050) YN
1050 FORMAT (A1)
IF (YN.EQ.'Y') THEN
C
PRINT*, 'AUTOMATISCHE PARAMETER (Y/N)'
READ(*, '(1A1)') YN
U=UE
IF (YN.EQ.'Y') THEN
CALL PARA (X,Y,P,Q,N1)
CALL SYMBOL (7.5,U,.175,'AUTOPARA',0.,8)
U=U-.25
DO 48 I=1,N1
C
CALL SYMBOL (7.5,U,.1,'P=',0.,2)
C
CALL NUMBER (7.9,U,.1,P(I),0.,1)
C
CALL NUMBER (9.6,U,.1,Q(I),0.,1)
48 U=U-.25
GOTO 61
ENDIF
DO 50 I=1,N1
PRINT*, 'FELD :', I
READ*, P(I)
CALL SYMBOL (7.5,U,.1,'P=',0.,2)
CALL NUMBER (7.9,U,.1,P(I),0.,1)
IF(NR.EQ.12) READ*, Q(I)
IF(NR.EQ.12) CALL NUMBER (9.6,U,.1,Q(I),0.,1)
U=U-.25
50 CONTINUE
CALL SYMBOL (11.5,UE,.175,8,0.,-2)
ELSE
CALL SYMBOL (7.5,UE,.175,'P=',0.,2)
CALL NUMBER (7.9,UE,.175,P1,0.,1)
IF(NR.EQ.12) CALL NUMBER (9.6,UE,.175,Q1,0.,1)
ENDIF
C
61 IF (NR.EQ.11) THEN
CALL SPLEXP (N,X,Y,P,Y1,A,B,C,D)
CALL PLSPLEXP (N,X,Y,P,A,B,C,D)
ENDIF
IF (NR.EQ.12) THEN
CALL RAT (N,X,Y,P,Q,Y1,A,B,C,D)
C
CALL RATA (N1,P,X,DX,T)
C
CALL RATB (N1,P,DX,Y,Y1,A,T)
C
CALL RATC (N1,P,X,Y,Y1,A,B,C,D)
CALL PLRAT (N,X,Y,P,P,A,B,C,D)
ENDIF
ENDIF
C
IF (NR.EQ.15) THEN
CALL CUBP (N,X,Y,Y2,F,G,T)
CALL PLCUB (N,X,Y,Y2)
ENDIF

```

```

C
  IF (NR.EQ.16) THEN
    CALL CYCLIC (N,T,X,X2,Y,Y2,F,G,Q)
    CALL PLCUBPAR (N,T,X,X2,Y,Y2)
  ENDIF

C
  IF (NR.EQ.17) THEN
    IDIM=3
    CALL UEBERSPL (N,X,Y,PU)
    CALL DRUCK (PU,3,3,'P =')
    CALL RAND2 (N,X,Y,PU,K1)
    CALL DRUCK (K1,3,1,'K1')
    CALL PLUEBERA (N,X,Y,K1,A,B,C)
    CALL DRUCK (A,4,1,'A')
    CALL DRUCK (B,4,1,'B')
    CALL DRUCK (C,4,1,'C')
    CALL PLUEBERB (N,X,Y,A,B,C)
  ENDIF

C
  IF (NR.EQ.18) THEN
    CALL VBEZ(N,X,Y,CB,CT)
    CALL PLBEZIER (N,X,Y,CB)
  ENDIF

C
  GOTO 10

C
C
990 CALL PLOT (0.,0.,999)
    CLOSE (7,STATUS='KEEP')
    STOP
    END

      SUBROUTINE DREH (X,Y,N,IKURV,NR)
C *****
C
C  VERSCHIEBT DEN NULLPUNKT DER KURVENSCHAREN
C  DREHT KURVENSCHAR WENN IKURV>5< ,ISPLINE=4
C  GEHOET ZUM PROGRAM SEMI
C
C  AXEL BUSCH
C
    REAL X(*),Y(*),XSTART(20),YSTART(20)
    DATA (XSTART(I),I=1,11)/0.,0.,0.,0.,0.,-4.,0.0,0.,0.,0.,.0/
    DATA (YSTART(I),I=1,11)/0.,0.,0.,0.,0.,5.,0.0,0.,0.,0.,.0/
    DO 10 I=1,N
      IF (IKURV.EQ.6) THEN
        XR=X(I)
        X(I)=XR*COS(-.5)-Y(I)*SIN(-.5)
        Y(I)=XR*SIN(-.5)+Y(I)*COS(-.5)
      ENDIF
    10 X(I)=(X(I)+XSTART(IKURV))
    RETURN

```

END

```

      SUBROUTINE BESCHR (IKURV,UE)
C *****
C
C BESCHRIFTET PLOTTS VOM PROGRAM SEMI
C
C AXEL BUSCH      (6.90)
C
      CALL DASHA (.5,UE,3)
      CALL DASHA (1.5,UE,2)
      UE=UE-.07
      IF(IKURV.LT.1.OR.IKURV.GT.18) GOTO 100
      GOTO (81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98
* ),IKURV
81  CALL SYMBOL(1.75,UE,.175,'KUB. SPLINE NAT.RB      ',0.,27)
      RETURN
82  CALL SYMBOL(1.75,UE,.175,'KUB. SPLINE KREIS-RB    ',0.,27)
      RETURN
83  CALL SYMBOL(1.75,UE,.175,'KUB. SPLINE TANGENTEN-RB ',0.,27)
      RETURN
84  CALL SYMBOL(1.75,UE,.175,'KUB. SPLINE TANG-RB Y1   ',0.,27)
      RETURN
85  CALL SYMBOL(1.75,UE,.175,'PARAMETRISCHER SPLINE   ',0.,27)
      RETURN
86  CALL SYMBOL(1.75,UE,.175,'HERMITISCHER SPLINE 5.GRADES',0.,28)
      RETURN
87  CALL SYMBOL(1.75,UE,.175,'HERMITISCHER SPLINE 3.GRADES',0.,28)
      RETURN
88  CALL SYMBOL(1.75,UE,.175,'HALBPUNKT SPLINE        ',0.,27)
      RETURN
89  CALL SYMBOL(1.75,UE,.175,'KUB SPLINE MITGEF. KOORDINATEN',0.,30)
      RETURN
90  CALL SYMBOL(1.75,UE,.175,'BEZIER SPLINE            ',0.,27)
      RETURN
91  CALL SYMBOL(1.75,UE,.175,'EXPONENTIELLER SPLINE    ',0.,27)
      RETURN
92  CALL SYMBOL(1.75,UE,.175,'RATIONALER SPLINE        ',0.,27)
      RETURN
95  CALL SYMBOL(1.75,UE,.175,'PERIODISCHER KUB. SPLINE ',0.,27)
      RETURN
96  CALL SYMBOL(1.75,UE,.175,'PERIOD. PARAM. KUB. SPLINE ',0.,27)
      RETURN
97  CALL SYMBOL(1.75,UE,.175,'UEBERTRAGUNGSMATRIX KUBISCH',0.,28)
      RETURN
98  CALL SYMBOL(1.75,UE,.175,'VERBESSERTER BEZIER SPLINE ',0.,27)
      RETURN
93  CONTINUE
94  CONTINUE
100 CALL SYMBOL(1.75,UE,.175,'POLYNOM                  ',0.,27)
      RETURN
      END

```

SEMI.DAT , DIE DATEI MIT KNOTENKOORDINATEN DER TESTBILDER

1 KLEINES DIAGRAM

0.5 5.5 1. 5. 2. 6. 2.4 7. 3. 7.25 3.7 7. 4. 4.8
 4.5 4.5 4.7 4.7 999.9 9999.9

2 HEBELARM

8. 2.25 9. 2.7 10. 3.05 11. 3.2 12. 2.3 12.1 2.0
 999.9
 8. 2.25 9. 2.8 10. 3.35 11. 3.55 12. 3.3 12.5 2.75 12.75 2.
 999.9 9999.9

3 WASSERLINIEN

4.4 3.25 4.9 2.5 5.5 2. 6.5 1.4 8.5 1.05 11. 1. 13. 1.
 999.9
 0.5 3.25 0.55 3.1 1. 2.7 1.5 2.55 2. 2.5 2.5 2.5 3. 2.45
 4. 2.1 6.5 1. 8.5 .7 11. .6 13.5 .6
 999.9
 0.5 1.55 1. 1.49 1.5 1.43 2. 1.37 2.5 1.31 3. 1.245 3.5 1.184
 4. 1.123 4.5 1.062 5. 1.0 5.5 .938 6. .877 6.5 .818 7. .761
 7.5 .708 8. .658 8.5 .6141 9. .576 9.5 .5445 10. .521
 10.5 .506 11. .5 11.5 .5 12. .5 12.5 .5 13. .5 13.5 .5
 999.9 9999.9

4 SPANTRISS

9.5 4.6 10. 4.95 10.25 5.3 10.35 6.5 10.5 7.25 11. 8.
 11.5 8.75 11.65 9.
 999.9
 9.5 4.5 9.75 4.5 10.8 4.8 11.5 5.6 12.5 7.1 13.1 7.7 13.6 8.25
 999.9
 9.5 4.5 11. 4.5 12.5 4.5 12.95 4.5 13. 4.5 13.75 5.25
 13.75 5.3 13.75 5.75 13.75 7. 13.75 8.25
 999.9
 5.25 8. 5.25 7. 5.25 5.75 5.25 5.3 5.25 5.25 5.4 4.75 6. 4.5
 6.05 4.5 6.5 4.5 8.0 4.5 9.5 4.5
 999.9
 5.5 7.4 5.75 6. 6. 5.55 6.5 5.15 7. 4.88 8. 4.65 9.5 4.5
 999.9 9999.9

6 SPANTRISS ZUM DREHEN

9.5 4.6 10. 4.95 10.25 5.3 10.35 6.5 10.5 7.25 11. 8.
 11.5 8.75 11.65 9.
 999.9
 9.5 4.5 9.75 4.5 10.8 4.8 11.5 5.6 12.5 7.1 13.1 7.7 13.6 8.25
 999.9
 9.5 4.5 11. 4.5 12.5 4.5 12.95 4.5 13. 4.5 13.75 5.25
 13.75 5.3 13.75 5.75 13.75 7. 13.75 8.25
 999.9 9999.9

7 SPANTRISS OHNE VERTIKALEN

9.5 4.6 10. 4.95 10.25 5.3 10.35 6.5 10.5 7.25 11. 8.
 11.5 8.75 11.65 9.
 999.9

9.5 4.5 9.75 4.5 10.8 4.8 11.5 5.6 12.5 7.1 13.1 7.7 13.6 8.25
999.9

9.5 4.5 11. 4.5 12.5 4.5 12.95 4.5 13. 4.5 13.75 5.25
13.8 5.75 13.9 7. 14. 8.25

999.9

5. 8. 5.1 7. 5.2 5.75 * 5.25 5.25 5.4 4.75 6. 4.5

6.1 4.5 6.5 4.5 8.0 4.5 9.5 4.5

999.9

5.5 7.4 5.75 6. 6. 5.55 6.5 5.15 7. 4.88 8. 4.65 9.5 4.5

999.9 9999.9

8 SCHWEINESCHWANZ

1. 2. 3.3 4.4 6.2 4.8 7.2 2.7 3.3 2. 5.75 5.3 8.6 5.3

9.1 3. 8.2 5.3 12. 6.5

999.9 9999.9

9 SPANTRISS OHNE DOPPELPUNKTE

9.5 4.6 10. 4.95 10.25 5.3 10.35 6.5 10.5 7.25 11. 8.

11.5 8.75 11.65 9.

999.9

9.5 4.5 9.75 4.5 10.8 4.8 11.5 5.6 12.5 7.1 13.1 7.7 13.6 8.25

999.9

9.5 4.5 11. 4.5 12.5 4.5 * 13. 4.5 13.75 5.25 *

13.75 5.75 13.75 7. 13.75 8.25

999.9

5.25 8. 5.25 7. 5.25 5.75 * 5.25 5.25 5.4 4.75 6. 4.5

* 6.5 4.5 8.0 4.5 9.5 4.5

999.9

5.5 7.4 5.75 6. 6. 5.55 6.5 5.15 7. 4.88 8. 4.65 9.5 4.5

999.9 9999.9

10 GROSSES DIAGRAMM

.5 4. 2. 1.5 4. 2. 4.5 6. 5.5 7. 7. 7. 7.3 2.5 8.5 1.5

9.5 1.5 11. 4. 12.5 1.5

999.9 9999.9

15 PERIODISCHES GROSSES DIAGRAMM

.5 4. 2. 1.5 4. 2. 4.5 6. 5.5 7. 7. 7. 7.3 2.5 8.5 1.5

9.5 1.5 11. 4. 12.5 1.5 13.5 4.

999.9 9999.9

16 KREISE

1. 4. 4. 7. 7. 4. 4. 1. 1. 4.

999.9

6. 6.5 11. 6.5 11. 1.5 6. 1.5 6. 6.5

999.9 9999.9

17 PUNKTE FUER POLYNOM

1. 1. 3. 4. 4.5 5.2 6. 5. 8. 3.

999.9 9999.9

18

1. 1. 3. 4. 4.5 5.2 6. 5. 8. 3. 8.3 1.

999.9 9999.9

19

1. 1. 3. 4. 4.5 5.2 6. 5. 6.5 4.8 8. 3. 8.3 1.

999.9 9999.9

```

C CUBPAR - PARAMETRISCH KUBISCHER SPLINE
C
C (C) Axel Busch 11.88 / 26.1.90 / 21.2.90 / 17.7.90
C
C Unterprogrammpaket CUBPAR.F enthaelt alles, um mit
c CUBPAR einen Spline zu berechnen, ihn dann mit
c PLCUBP plotten zu lassen oder mit
c FCUBP Interpolationswerte an einzelnen Abzissen auszurechnen.
c Das vollstaendige Paket enthaelt folgende Unterprogramme:
c 1.) CUBPAR - kubischer, parametrischer Spline
c 1a.)CUBPIT - iterativer Strak ueber Bogenlaenge
c 2.) CUB1 - kubischer Spline
C 3.) PLCUBP - Plotroutine zu CUBPAR
C 7.) FCUBP - ein Interpolationswert aus CUBPAR  $y = f(x)$ 
C 8.) TCUBP - alle Interpolationswerte aus CUBPAR  $x = f(t)$ 
C 9.) DCUBP - Funktionswert und Ableitung
C 10.) NULL3 - alle Nullstellen eines kubischen Polynoms
C 11.) NULL - eine Nullstelle kubisches Polynoms
C
C ( DASHAB ) - wie PLOT aber mit strichpunktiierten Linien
C

```

```

SUBROUTINE CUBPAR (N,T,X,X2,Y,Y2,F,G)
C *****
C
C PARAMETRISCHER KUBISCHER SPLINE
C DOPPELTE PUNKTE WERDEN ALS KNICK INTERPRETIERT (1.90)
C
C AXEL BUSCH 10.86
C
C UNTERPROGRAMME : CUB1 (KUBISCHER SPLINE)
C
C N - IN - ANZAHL DER PUNKTE
C T(*) - OUT - POLYGONZUG LAENGE
C X(*) - IN - X-KOORDINATEN DER PUNKTE
C Y(*) - IN - Y-KOORDINATEN DER PUNKTE
C X2(1) - IN - 2. ABLEITUNG AM PUNKT 1 DX/DT
C X2(N) - IN - 2. ABLEITUNG AM PUNKT N DX/DT
C Y2(1) - IN - 2. ABLEITUNG AM PUNKT 1 DY/DT
C Y2(N) - IN - 2. ABLEITUNG AM PUNKT N DY/DT
C X2(*) - OUT - 2. ABLEITUNG AN DEN PUNKTEN
C Y2(*) - OUT - 2. ABLEITUNG AN DEN PUNKTEN
C F(*) - - ARBEITSFELD
C G(*) - - ARBEITSFELD
C
C PARAMETER :
C
C RFANG - FANGRADIUS**2 FUER IDENTISCHE PUNKTE
C
C PARAMETER (RFANG = 1.E-08)
C REAL T(*),X(*),X2(*),Y(*),Y2(*),F(*),G(*)
C
C IF (N.LT.2) RETURN

```

```

TKNICK=0.          ! T-ZUSCHLAG DURCH KNICKE
T(1)=0.
DO 1 K=2,N
  U=X(K)-X(K-1)
  V=Y(K)-Y(K-1)
  D=U*U+V*V
  IF ( ABS(D).GT. RFANG ) THEN
    D=SQRT(D)
  ELSE
    D=(T(K-1)-TKNICK)
    IF(D. LT. RFANG) THEN
C   SPLINE BEGINNT MIT KNICK
      D=10000.
    ELSE
C   D IST 10000 * MITTLERE LAENGE
      D=D *FLOAT(10000/(K-2))
    ENDIF
    TKNICK=TKNICK+D
  ENDIF
  T(K)=T(K-1)+D
1  CONTINUE
C
  CALL CUB1(N,T,X,X2,F,G)
  CALL CUB1(N,T,Y,Y2,F,G)
C
  IF(TKNICK.LT. 0.01) GOTO 4
  DO 3 K=2,N
    U=X(K)-X(K-1)
    V=Y(K)-Y(K-1)
    D=U*U+V*V
    IF ( ABS(D).GT. RFANG ) THEN
      D=SQRT(D)
      T(K)=T(K-1)+D
    ELSE
      T(K)=T(K-1)
    ENDIF
3  CONTINUE
4  RETURN
END

SUBROUTINE CUBPIT (N,T,X,X2,Y,Y2,F,G)
C *****
C
C   PARAMETRISCH KUBISCHER SPLINE UEBER BOGENLAENGE, 5 ITERATIONEN
C   ZUVOR MUSS CUBPAR EINMAL AUFGERUFEN WORDEN SEIN !
C   (LANGSAMER ALGORITHMUS)
C
C   AXEL BUSCH    17.7.90
C
C   UNTERPROGRAMME : CUB1
C
C   N      - IN      - ANZAHL DER PUNKTE

```

```

C T(*) - IN/OUT - POLYGONZUGLAENGE AUS CUBPAR
C X(*) - IN - X-KOORDINATEN DER PUNKTE
C Y(*) - IN - Y-KOORDINATEN DER PUNKTE
C X2(*) - IN/OUT - 2. ABLEITUNG AN DEN PUNKTEN DX/DT
C Y2(*) - IN/OUT - 2. ABLEITUNG AN DEN PUNKTEN DY/DT
C F - - ARBEITSFELD
C G - - "
C
PARAMETER (RFANG = 1.E-08)
REAL X(*),Y(*),X2(*),Y2(*),T(*), F(*),G(*)
FX(TT)=AX*TT*TT*TT+BX*TT*TT+CX*TT+DDX
FY(TT)=A*TT*TT*TT+B*TT*TT+C*TT+D

IF (N.LT.2) RETURN

C 5 ITERATIONEN
DO 3 ITER=1,5
TKNICK=0.
T(1)=0.

DT=T(2)-T(1)
N1=N-1
DO 2 K=1,N1
DX=X(K+1)-X(K)
DY=Y(K+1)-Y(K)
IF (DX*DX + DY*DY. GT.RFANG) THEN
RDT=1./DT
A=(Y2(K+1)-Y2(K))*RDT*.16666667
B=Y2(K)*.5
C=DY*RDT-DT*.16666667*(Y2(K+1)+2.*Y2(K))
D=Y(K)
AX=(X2(K+1)-X2(K))*RDT*.16666667
BX=X2(K)*.5
CX=DX*RDT-DT*.16666667*(X2(K+1)+2.*X2(K))
DDX=X(K)

TT=0.
TTNEU=0.
TSTEP=DT *.1
DO 1 I=1,10
DX=FX(TT+TSTEP) - FX(TT)
DY=FY(TT+TSTEP) - FY(TT)
TTNEU=TTNEU+ SQRT(DX*DX + DY*DY)
TT=TT+TSTEP
1 CONTINUE

ELSE
TTNEU=(T(K)-TKNICK)
IF(TTNEU. LT. RFANG) THEN
C SPLINE BEGINNT MIT KNICK
TTNEU=10000.
ELSE
C TTNEU IST 10000 * MITTLERE LAENGE
TTNEU=TTNEU *FLOAT(10000/(K-1))
ENDIF

```

```

        TKNICK=TKNICK+TTNEU
    ENDIF
    IF(K.LT.N1) DT=T(K+2)-T(K+1)
    T(K+1)=T(K)+TTNEU
2    CONTINUE

    CALL CUB1(N,T,X,X2,F,G)
    CALL CUB1(N,T,Y,Y2,F,G)

3    CONTINUE

C    PARAMETER KORRIGIEREN, KNICKE RAUSSCGNEIDEN
    TKNICK=0.
    T(1)=0.

    DT=T(2)-T(1)
    N1=N-1
    DO 12 K=1,N1
        DX=X(K+1)-X(K)
        DY=Y(K+1)-Y(K)
        IF (DX*DX + DY*DY. GT.RFANG) THEN
            RDT=1./DT
            A=(Y2(K+1)-Y2(K))*RDT*.16666667
            B=Y2(K)*.5
            C=DY*RDT-DT*.16666667*(Y2(K+1)+2.*Y2(K))
            D=Y(K)
            AX=(X2(K+1)-X2(K))*RDT*.16666667
            BX=X2(K)*.5
            CX=DX*RDT-DT*.16666667*(X2(K+1)+2.*X2(K))
            DDX=X(K)

            TT=0.
            TTNEU=0.
            TSTEP=DT *.1
            DO 11 I=1,10
                DX=FX(TT+TSTEP) - FX(TT)
                DY=FY(TT+TSTEP) - FY(TT)
                TTNEU=TTNEU+ SQRT(DX*DX + DY*DY)
                TT=TT+TSTEP
11        CONTINUE

            ELSE
                TTNEU=0.
            ENDIF
            IF(K.LT.N1) DT=T(K+2)-T(K+1)
            T(K+1)=T(K)+TTNEU
12    CONTINUE

    RETURN
    END

```

```

        SUBROUTINE PLCUBP (N,T,X,X2,Y,Y2)
C    *****

```

```

C
C PLOTTET PARAMETRISCHE KUBISCHE SPLINES
C
C AXEL BUSCH
C
C UNTERPROGRAMME : PLOT (ODER DASHAB)
C
C N      - IN  - ANZAHL DER PUNKTE
C T(*)   - IN  - POLYGONZUGLAENGE AUS CUBPAR
C X(*)   - IN  - X-KOORDINATEN DER PUNKTE
C Y(*)   - IN  - Y-KOORDINATEN DER PUNKTE
C X2(*)  - IN  - 2. ABLEITUNG AN DEN PUNKTEN DX/DT
C Y2(*)  - IN  - 2. ABLEITUNG AN DEN PUNKTEN DY/DT
C
C
C      REAL X(*),Y(*),X2(*),Y2(*),T(*)
C      CALL DASHA(X(1),Y(1),3)
C      CALL PLOT(X(1),Y(1),3)
C      N1=N-1
C      DO 2 K=1,N1
C          DT=T(K+1)-T(K)
C          IF (DT.LE. 0.) GOTO 2
C          DX=X(K+1)-X(K)
C          DY=Y(K+1)-Y(K)
C          A=(Y2(K+1)-Y2(K))/6./DT
C          B=Y2(K)/2.
C          C=DY/DT-DT/6.*(Y2(K+1)+2.*Y2(K))
C          D=Y(K)
C          AX=(X2(K+1)-X2(K))/6./DT
C          BX=X2(K)/2.
C          CX=DX/DT-DT/6.*(X2(K+1)+2.*X2(K))
C          DDX=X(K)
C
C          TT=0.
C          IF (TT.GT.DT) GOTO 2
C          FY=A*TT*TT*TT+B*TT*TT+C*TT+D
C          FX=AX*TT*TT*TT+BX*TT*TT+CX*TT+DDX
C      CALL DASHA(FX,FY,2)
C      CALL PLOT(FX,FY,2)
C      TT=TT+.3
C      GOTO 1
C
C 2      CONTINUE
C      CALL DASHA(X(N),Y(N),2)
C      CALL PLOT(X(N),Y(N),2)
C      CALL PLOT(X(N),Y(N),3)
C      RETURN
C      END

C      FUNCTION FCUBP (X0,T0, N,T,X,X2,Y,Y2)
C      *****
C      EIN FUNKTIONSWERT AUS PARAMETRISCHE KUBISCHE SPLINES
C      EINFACHE SUCHE , OHNE BAEUCHE - SONST DCUBP BENUTZEN
C      SOLL X = F(Y) GEFUNDEN WERDEN ...
C
C          X = FCUBP (Y0,T0,N,T,Y,Y2,X,X2)

```

```

C
C AXEL BUSCH 11.88 / 11.89
C
C UNTERPROGRAMME: NULL (NULLSTELLENSUCHE QUBISCHER POLYNOME)
C
C FCUBP - OUT - F(X0)
C X0 - IN - X0-KOORDINATE DES PUNKTES
C TO - OUT - DAZUGEHÖRIGER KURVENPARAMETER
C TO = -1. FALLS KEIN SCHNITT GEFUNDEN WURDE
C N - IN - ANZAHL DER PUNKTE
C T(*) - IN - POLYGONZUGLÄNGE AUS CUBPAR
C X(*) - IN - X-KOORDINATEN DER PUNKTE
C Y(*) - IN - Y-KOORDINATEN DER PUNKTE
C X2(*) - IN - 2. ABLEITUNG AN DEN PUNKTEN DX/DT
C Y2(*) - IN - 2. ABLEITUNG AN DEN PUNKTEN DY/DT
C
C
C REAL X(*),Y(*),X2(*),Y2(*),T(*),T00(3)
C N1=N-1
C T0=-1.
C FCUBP =0.
C
C DO 2 K=1,N1
C IF ( (X0-X(K)) * (X0-X(K+1)) ) 1,1,2
1 DT=T(K+1)-T(K)
IF (DT.LE. 0.) GOTO 2
DX=X(K+1)-X(K)
DY=Y(K+1)-Y(K)
RDT=1./DT
A=(Y2(K+1)-Y2(K))*1.6666667*RDT
B=Y2(K)*.5
C=DY*RDT-DT*1.6666667*(Y2(K+1)+2.*Y2(K))
D=Y(K)
AX=(X2(K+1)-X2(K))*1.6666667 *RDT
BX=X2(K)*.5
CX=DX*RDT-DT*1.6666667 *(X2(K+1)+2.*X2(K))
DDX=X(K)
IANZ=0
CALL NULL(0.,DT,DT*.0001,T0,IANZ,AX,BX,CX,DDX-X0)
FCUBP=A*T0*T0*T0 +B*T0*T0 +C*T0 +D
T0=T0+T(K)
RETURN
2 CONTINUE
D PRINT*, 'FUNCTION FCUBP: KEIN SCHNITT GEFUNDEN !'
RETURN
END
SUBROUTINE TCUBP (X0,TO,IANZ, N,T,X,X2,Y,Y2)
C *****
C BESTIMMT ALLE NULLSTELLEN/FUNKTIONSWERTE EINES PARAMETRISCHE,
C KUBISCHE SPLINES

```

```

C FUER FUNKTIONSWERTE ... CALL TCUBP(YO,TO,IANZ, N,T,Y,Y2,X,X2)
C
C AXEL BUSCH 11.89
C
C XO - IN - X-FUNCTIONSWERT DER GESUCHTEN STELLE
C TO( ) - OUT - KURVENPARAMETER DER PUNKTE MIT F(TO) = XO
C MIT DCUBP LASSEN SICH WEITER INFORMATIONEN HOLEN
C IANZ -IN/OUT - IANZ = IANZ + ANZAHL DER EINTRAEGE IN TO
C
C N - IN - ANZAHL DER PUNKTE
C T(*) - IN - POLYGONZUGLAENGE AUS CUBPAR
C X(*) - IN - X-KOORDINATEN DER PUNKTE
C Y(*) - IN - Y-KOORDINATEN DER PUNKTE
C X2(*) - IN - 2. ABLEITUNG AN DEN PUNKTEN DX/DT
C Y2(*) - IN - 2. ABLEITUNG AN DEN PUNKTEN DY/DT
C
C UNTERPROGRAMME : NULL3, NULL, NODOUBLE
C
C
C INTEGER IANZ,N,N1,K
C REAL XO,TO(*),AX,BX,CX,DX,DT,RDT
C REAL T(*),X(*),Y(*),X2(*),Y2(*)
C
C N1=N-1
C IANZ=0
C DO 2 K=1,N1
C DT=T(K+1)-T(K)
C IF (DT.LE. 0.) GOTO 2
C RDT=1./DT
C AX=(X2(K+1)-X2(K))*1.6666667*RDT
C BX=X2(K)*.5
C CX=(X(K+1)-X(K))*RDT-DT*1.6666667*(X2(K+1)+2.*X2(K))
C DX=X(K) -XO
C CALL NULL3(0.,DT,DT*.0001,TO,T(K),IANZ,AX,BX,CX,DX)
C 2 CONTINUE
C CALL NODOUBLE ( TO, IANZ)
C RETURN
C END
C
C FUNCTION DCUBP (TO,DX,DY,XX,YY, N,T,X,X2,Y,Y2)
C *****
C ABLEITUNGEN UND FUNKTIONSWERTE EINES PARAMETRISCH, KUB. SPLINES
C
C AXEL BUSCH 11.89
C
C DCUBP - OUT - =0 ERROR, =-1 OK.
C
C TO - IN - KURVENPARAMETER DES PUNKTES (AUS FCUBP)
C DX - OUT - DX/DY BEI TO
C DY - OUT - DY/DX BEI TO
C XX - OUT - FUNKTIONSWERT BEI TO
C YY - OUT - FUNKTIONSWERT BEI TO
C

```

```

C N      - IN  - ANZAHL DER PUNKTE
C T(*)   - IN  - POLYGONZUGLAENGE AUS CUBPAR
C X(*)   - IN  - X-KOORDINATEN DER PUNKTE
C Y(*)   - IN  - Y-KOORDINATEN DER PUNKTE
C X2(*)  - IN  - 2. ABLEITUNG AN DEN PUNKTEN DX/DT
C Y2(*)  - IN  - 2. ABLEITUNG AN DEN PUNKTEN DY/DT
C
C
      INTEGER N,N1,K
      REAL TO,DX,DY,XX,YY, AX,BX,CX, AY,BY,CY
      REAL T(*),X(*),Y(*),X2(*),Y2(*)

      N1=N-1
      DCUBP = 0
      DO 2 K=1,N1
        IF ( (TO-T(K)) * (TO-T(K+1)) ) 1,1,2
1       DT=T(K+1)-T(K)
          RDT=1./DT
          AY=(Y2(K+1)-Y2(K))*1.16666667*RDT
          BY=Y2(K)*.5
          CY=(Y(K+1)-Y(K))*RDT-DT*.16666667*(Y2(K+1)+2.*Y2(K))
          DY=Y(K)
          AX=(X2(K+1)-X2(K))*1.16666667 *RDT
          BX=X2(K)*.5
          CX=(X(K+1)-X(K))*RDT-DT*.16666667 *(X2(K+1)+2.*X2(K))
          DX=X(K)

          TOO=TO-T(K)
          XX= AX*TOO*TOO*TOO + BX*TOO*TOO + CX*TOO + DX
          YY= AY*TOO*TOO*TOO + BY*TOO*TOO + CY*TOO + DY

C DX/DT UND DY/DT BEI TOO
          ABLX=3.*AX*TOO*TOO +2.*BX*TOO +CX
          ABLY=3.*AY*TOO*TOO +2.*BY*TOO +CY
C SICHERHEITSTESTS MAX. EXPONENT=34
          IF(ABS(ABLX).LT.1.E-16) ABLX=SIGN(1.E-16 ,ABLX)
          IF(ABS(ABLY).LT.1.E-16) ABLY=SIGN(1.E-16 ,ABLY)
          IF(ABS(ABLX).GT.1.E+16) ABLX=SIGN(1.E+16 ,ABLX)
          IF(ABS(ABLY).GT.1.E+16) ABLY=SIGN(1.E+16 ,ABLY)

          DX = ABLX / ABLY
          DY = ABLY / ABLX
          DCUBP = -1
          RETURN
2       CONTINUE
D       PRINT*,
D       * 'FUNCTION DCUBP: UNZULAESSIGER KURVENPARAMETER TO=',TO,'!'
          RETURN
          END

      SUBROUTINE NULL3 (XLOW,XHIGH,DX, X0,XOFFSET,IANZ, A,B,C,D)
C *****
C BESTIMMT NULLSTELLEN EINES GEGEBENEN, QUBISCHEN POLYNOMS

```

```

C
C A.BUSCH  11.89
C
C DX      - IN - TOLERANZ FUER X
C XO( )   - OUT - NULLSTELLEN + XOFFSET
C XOFFSET- IN - SUMMANT FUER XO( )
C IANZ    - OUT - ANZAHL DER NULLSTELLEN
C A,B,C,D- IN - KOEFFIZIENTEN DES POLYNOMS F=AXX+BXX+CX+D
C
C
C XGRENZ - GRENZEN DES INTERVALL'S
C          REAL A,B,C,D,XO(*)
C XAO SIND DIE BEREICHSGRENZEN ZWISCHEN DEN
C                                     NULLSTELLEN DER 1.ABLEITUNG
C          REAL XAO(4)
C ANZAHL DIESER BEREICHE, ANZAHL DER NULLSTELLEN
C          INTEGER IABSCHN, IANZ
C
C          REAL P,Q,WURZ
C
C          F(X)=A*X*X*X+B*X*X+C*X+D
C
C BESTIMMUNG DER NULLSTELLEN IN DER 1. ABLEITUNG
C F'(X)=3AXX+2BX+C = 0.    , XAO=-P/2+-SQRT(PP/4-Q)
C
C          IABSCHN=1
C
C          IF (A.NE.0.) THEN
C              P=2.*B/(3.*A)
C              Q=C/(3.*A)
C              WURZ=P*P*.25 -Q
C              IF (WURZ.EQ.0.) IABSCHN=2
C              IF (WURZ.GT.0.) IABSCHN=3
C
C          IF (IABSCHN.GT.1) X1= -P*.5-SQRT(WURZ)
C          IF (IABSCHN.GT.2) X2= -P*.5+SQRT(WURZ)
C
C          ELSE
C              IF(B.NE.0.) IABSCHN=2
C              IF (IABSCHN.EQ.2) X1=C/(2.*B)
C
C          ENDIF
C
C          IFLAG=0
C          XAO(1)= XLOW
C          XAO(2)=0.
C          XAO(3)=0.
C          XAO(4)=0.
C          IF (IABSCHN.GT. 1 .AND. X1.GT.XLOW .AND. X1.LT.XHIGH) THEN
C              XAO(2)=X1
C              IFLAG=1
C          ENDIF
C          IF (IABSCHN.GT. 2 .AND. X2.GT.XLOW .AND. X2.LT.XHIGH) THEN
C              IF (IFLAG.EQ.1) THEN
C                  XAO(3)=X2
C                  IFLAG=2
C              ELSE

```

```

        XAO(2)=X2
        IFLAG=1
        ENDIF
    ENDIF
    XAO(IFLAG+2)= XHIGH
C
    IABSCHN= IFLAG+1

    DO 10 I=1,IABSCHN
        IF (F(XAO(I)) * F(XAO(I+1)) .LE. 0.0 ) THEN
            CALL NULL (XAO(I), XAO(I+1), DX, XO, IANZ, A, B, C, D)
            XO( IANZ) = XO( IANZ) + XOFFSET
        ENDIF
10    CONTINUE

    RETURN
    END

        SUBROUTINE NULL (XX1,XX2,DX,XO, IANZ, A, B, C, D)
C *****
C BESTIMMT EINE NULLSTELLE KUBISCHER POLYNOME
C ZWISCHEN XX1<=XO<=XX2 . EINE KONTROLLE,
C OB UEBERHAUPT EIN SCHNITT VORLIEGT, ERFOLGT NICHT!
C
C A.BUSCH          9.88
C
C XX1      - IN - INTERVALL-ANFANG
C XX2      - IN - INTERVALL-ENDE
C DX       - IN - TOLERANZ
C XO( )    - OUT - NULLSTELLE
C IANZ     - IN - WIRD IMMER! UM 1 ERHOEHT XO(IANZ+1) KRIEGT DIE
C           AKTUELLE NULLSTELLE
C A ...D   - IN - POLYNOM-KOEFFIZIENTEN
C
    REAL XO(*)
    F(X)=A*X*X*X+B*X*X+C*X+D

    X1=XX1
    X2=XX2
    F1=F(X1)
    F2=F(X2)
    STEIG=1.
    IF(F2-F1.LT.0.)STEIG=-1.
    N= INT(ALOG((X2-X1)/DX) *1.442695) +2
    DO 10 I=1,N
    X= (X1+X2)*.5
    FX=F(X)
    IF(FX.EQ.0.) GOTO 20
    IF ((FX*STEIG).LE.0.) THEN
        X1=X
        F1=FX
    ELSE
        X2=X

```

```

        F2=FX
        ENDIF
10     CONTINUE
20     IANZ=IANZ+1
        XO(IANZ)=X
        END

```

```

        SUBROUTINE NODOUBLE(XO, IANZ)
C   BESEITIGT DOPPELTE NULLSTELLEN
C
C   AXEL BUSCH   11.89
C
C   XO(*) - IN - X-KOORDINATE
C   IANZ - IN - ANZAHL DER EINTAEGE IN XO
C
C   PARAMETER : RFANG - FANGRADIUS FUER GLEICHE PUNKTE
C
        PARAMETER (RFANG = 1.E-03)
        REAL XO(*)

        GOTO 40
20     IANZ=IANZ-1
        DO 30 J=I, IANZ
30     XO(J)=XO(J+1)
40     DO 50 I=2, IANZ
        IF ( ABS(XO(I-1)-XO(I)).LT.RFANG ) GOTO 20
50     CONTINUE
        RETURN
        END

```

```

PROGRAM CUBITEST
PARAMETER (NP=99)
REAL T(NP), X(NP), Y(NP), X2(NP), Y2(NP), F(NP), G(NP)
REAL TO(10)

PRINT*, ' READING CUBPOINT.DAT !'
OPEN (1, FILE='D:\TEST\DAT\CUBPOINT.DAT', STATUS='OLD')

DO 5 I=1, 99
  READ(1, *, END=6) X(I), Y(I)
C  END-KENNUNG X=Y > 9999.
  IF (X(I).EQ.Y(I).AND. X(I).GT.9999.) GOTO 6
5  CONTINUE

6  X2(1) =0.
   Y2(1) =0.
   N=I-1
   X2(N)=0.
   Y2(N)=0.

```

```

        CALL PLOTS(0.,0.,0)
        CALL PLOT(1.,1.,-3)

10      CALL CUBPAR (N,T,X,X2,Y,Y2,F,G)
20      DO 1 I=1,N
1       CALL SYMBOL(X(I),Y(I),.3, 2 ,0. ,-1)
        CALL PLCUBP(N,T,X,X2,Y,Y2)
C XO EINLESEN
        READ*, XO

C          CALL CLEARS                ! CLEAR SCREEN
C XO >= 100 -> END-KENNUNG
        IF(XO.GT.100.) GOTO 999
        YO= FCUBP(XO,T00,          N,T,X,X2,Y,Y2)
        PRINT*,'X:',XO,' YO:',YO,' TO',T00
        PRINT*
        IANZ=0
        CALL TCUBP(XO,T0,IANZ, N,T,X,X2,Y,Y2)
        DO 30 I=1,IANZ
        JJ=DCUBP(TO(I),DX,DY,XX,YY, N,T,X,X2,Y,Y2)
        PRINT*,'X:',XX,' YO:',YY,' TO',TO(I),' DX:',DX,' DY:',DY

30      CONTINUE
        GOTO 20

999     CALL PLOT(0.,0.,999)
        CLOSE(1,STATUS='KEEP')
        STOP
        END

```

Literatur

- [1] H. Späth . *Spline-Algorithmen zur Konstruktion glatter Kurven und Flächen*, Oldenburg Verlag (1978) ISBN 3-486-39473-8
- [2] G. Jordan-Engeln , F. Reutter : *Numerische Mathematik für Ingenieure*, Bibliographisches Institut, Zürich (1982) ISBN 3-411-01647-7
- [3] H. Nowacki, R. Gnatz : *Geometrisches Modellieren*, Springer-Verlag, Berlin (1982)
- [4] I. N. Bronstein, K. A. Semendjajew : *Taschenbuch der Mathematik*, Verlag Harry Deutsch, Frankfurt/Main (1981) ISBN 3-87144-492-3
- [5] J. Hoschek, D. Lasser : *Grundlagen der geometrischen Datenverarbeitung*, B. G. Teubner Stuttgart (1989) ISBN 3-519-02962-6
- [6] G. Hämmerlin, K.-H. Hoffmann : *Numerische Mathematik*, Springer-Verlag, Berlin (1989) ISBN 3-540-15306-3
- [7] D. G. Schweikert : *An Interpolation Curve Using a Spline in Tension* , Jour. Math. A. Physics 45 (1966) 312-317
- [8] C. de Boor : *On calculating with B-Splines* , Jour. Approx. Theory 6 (1972) 50-62
- [9] Inst. für Schiffbau Benutzerinformation Nr.:11