Schnelle Kurvenapproximation durch adaptive Splines

Werner Vogt* Christoph Weidling**

Februar 1996

Eine aktuelle Aufgabe des wissenschaftlichen Rechnens (vgl. [4]) bildet die schnelle Approximation von Kurven zur computergraphischen Auswertung von Rechnungen. Besonders gut lassen sich bekanntlich Funktionen durch kubische Splines annähern. In diesem Beitrag sollen effiziente Verfahren zu deren Konstruktion vorgestellt und die Güte der erzielten Näherungen untersucht werden. Insbesondere wird dabei auf Möglichkeiten adaptiver Interpolation eingegangen, wozu geeignete adaptive Parametrisierungen untersucht werden. Das Problem des nachträglichen Einfügens und Entfernens von Knoten wird mittels einer lokalen Anpassung der kubischen Splines gelöst. Zusammen mit einer effizienten Strategie der Funktionswertberechnungen wird der vorgestellte Zugang in dem Programmsystem SPLINES implementiert.

*Werner Vogt, Technische Universität Ilmenau, Institut für Mathematik, PF 0565, D-98684 Ilmenau, email vogt@mathematik.tu.ilmenau.de

**Christoph Weidling, Technische Universität Ilmenau, Fakultät für Informatik und Automatisierung, PF 0565, D-98684 Ilmenau

1 Einleitung

Wir betrachten eine reelle Funktion $f : \mathbb{R} \to \mathbb{R}$, die als hinreichend glatt vorausgesetzt wird. Ferner seien von dieser Funktion $n+1$ Knoten und deren Funktionswerte gegeben, die wir als Vektoren $\boldsymbol{x} = (x_0, x_1, \ldots, x_n)^T$ und $\boldsymbol{y} = (y_0, y_1, \ldots, y_n)^T$ mit $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{n+1}$ notieren. Folgende Bedingungen seien erfüllt:

$$\begin{aligned} y_i &= f(x_i) \quad \text{für } i = 0(1)n \\ x_i &< x_{i+1} \quad \text{für } i = 0(1)n-1 \\ x_0 &= a\,, \quad x_n = b\,. \end{aligned} \tag{1}$$

Dann sei $S : \mathbb{R} \to \mathbb{R}$ eine Interpolationsfunktion, die durch die Knotenvektoren $\boldsymbol{x}$ und $\boldsymbol{y}$ definiert ist. Insbesondere muß S die Interpolationsforderung

$$S(x_i) = y_i \quad \text{für } i = 0(1)n. \tag{2}$$

erfüllen. Jede kubische Splinefunktion S über dem Intervall $[a, b]$, welche eine Funktion f interpoliert, besitzt bekanntlich folgende Eigenschaften (vgl.[2], [6]):

Bed. 1.1. Es gibt Knotenvektoren $\boldsymbol{x}$ und $\boldsymbol{y}$ aus $\mathbb{R}^{n+1}$, die zusammen mit f und S den Beziehungen (1) und (2) genügen.

Bed. 1.2. Im Intervall $[x_i, x_{i+1}]$ hat S die Form

$$S(x) = S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3\,, \quad i = 0(1)n-1\,. \tag{3}$$

Die Koeffizienten sollen als *Koeffizientenvektoren* $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}$ notiert werden.

Bed. 1.3. Für $x < a$ bzw. $x > b$ ist S die Tangente an den Graphen von f in den jeweiligen Randpunkten.

Weiterhin können wir nun mit den Koeffizientenvektoren folgende Eigenschaften festhalten:

Bed. 1.4. Aus (1) folgt sofort, daß $a_i = f(x_i) = y_i$ für $i = 0(1)n$ gilt.

Bed. 1.5. Ist gefordert, daß S stetig ist, so folgt $S_i(x_i) = a_i = S_{i-1}(x_i)$ für $i = 1(1)n$.

Bed. 1.6. Ist gefordert, daß S C^1-stetig ist, so folgt $S_i'(x_i) = b_i = S_{i-1}'(x_i)$ für $i = 1(1)n-1$.

Bed. 1.7. Ist gefordert, daß S C^2-stetig ist, so folgt $S_i''(x_i) = 2c_i = S_{i-1}''(x_i)$ für $i = 1(1)n-1$.

Bed. 1.8. Schließlich kommen zwei Randbedingungen hinzu, nämlich daß entweder (i) an beiden Rändern die erste, zweite oder dritte Ableitung gegeben ist, daß (ii) die dritte Ableitung in den Punkten (x_1, y_1) und (x_{n-1}, y_{n-1}) stetig ist (*not-a-knot*-Randbedingung) oder daß (iii) S periodisch ist, also $S(x_0) = S(x_n)$, $S'(x_0) = S'(x_n)$ und $S''(x_0) = S''(x_n)$ gilt.

Mit Hilfe dieser Bedingungen erhält man in jedem dieser Fälle ein lineares Gleichungssystem mit eindeutiger Lösung (vgl. [2]),womit für eine gegebene Zerlegung und gegebene Randbedingungen die Splinefunktion S eindeutig bestimmt ist.

Zur Kurvenapproximation werden jedoch Algorithmen gesucht, die eine gegebene Funktion so interpolieren, daß eine vorgegebene Fehlerschranke für $x \in [a, b]$ nicht überschritten wird. Wie man unschwer an konkreten Aufgabenstellungen sieht, liefern äquidistante Parametrisierungen dafür meist unbefriedigende Resultate. Also verwendet man entweder eine in der Literatur (vgl. [1], [3]) angegebene Parametrisierung oder aber eine Parametrisierung, die sich der gegebenen Funktion anpaßt, eine "adaptive" Parametrisierung. In Abschnitt 2 werden hierfür geeignete Algorithmen vorgestellt und begründet.

Hat man schließlich eine derartige Splinefunktion S ermittelt, so erhebt sich die zweite Frage: Wie ändert sich die Interpolierende, wenn man eine weitere Stützstelle einfügt? Diese Frage wird leider numerisch ungünstig beantwortet, da die Splines keine lokale Kontrolle besitzen. So müssen bei der geringsten Änderung eines der Knoten oder beim Einfügen bzw. Entfernen von Knoten alle Koeffizientenvektoren komplett neu berechnet werden. Einen gangbaren Ausweg bietet die in Abschnitt 3 vorgestellte "Methode der vierten Potenz" an, die der nachträglichen lokalen Anpassung kubischer Splines dient.

Schließlich wird die Effizienz der Kurvenapproximation entscheidend durch die häufige Berechnung der Funktionswerte der Interpolierenden S bestimmt. Die Werte einer Splinefunktion kann man sehr schnell für sukzessive Argumente x_i berechnen. Will man jedoch Funktionswerte an willkürlich gewählten Argumentstellen bestimmen, so muß man stets den entsprechenden lokalen Spline suchen. Bei einer großen Anzahl von Stützstellen ist eine einfache lineare Suche nicht mehr akzeptabel, da man pro zu berechnenden Funktionswert einen Suchaufwand von $O(n)$ hat. In Abschnitt 4 werden effizientere Strategien vorgestellt und in dem menügesteuerten Programm SPLINES implementiert.

2 Adaptive Spline-Interpolation

2.1 Fehlerdarstellungen

Um eine adaptive Interpolierende zu gewinnen, muß deren Fehler leicht und schnell bestimmbar sein. Wir definieren deshalb, bezogen auf das endliche Intervall $[a, b]$, folgende Größen:

Definition 1. Der *quadratische Flächenfehler* F laute:

$$F(a,b) = \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} (f(x) - S(x))^2 dx \; . \tag{4}$$

Der *relative quadratische Flächenfehler* F_{r} laute:

$$F_{\mathrm{r}}(a,b) = \frac{\sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} (f(x) - S(x))^2 dx}{b-a} \; . \tag{5}$$

Insbesondere erhält man für ein Teilintervall $[x_i, x_{i+1}]$ mit $0 \leq i \leq n-1$ diesen Fehler zu

$$F_{\mathrm{r}}(x_i, x_{i+1}) = \frac{\int_{x_i}^{x_{i+1}} (f(x) - S(x))^2 dx}{b-a} \; .$$

Diese Größe besitzt zudem die vorteilhafte Eigenschaft, daß für eine Funktion f und ihre Interpolierende S in zwei angrenzenden Intervallen $[a_1, a_2] \subset [a, b]$ und $[a_2, a_3] \subset [a, b]$ gilt:

$$F_{\mathrm{r}}(a_1, a_3) = \frac{(a_2 - a_1) F_{\mathrm{r}}(a_1, a_2) + (a_3 - a_2) F_{\mathrm{r}}(a_2, a_3)}{a_3 - a_1} \; .$$

Man kann also den relativen quadratischen Flächenfehler über ein Intervall als gewichtetes arithmetisches Mittel von relativen quadratischen Flächenfehlern berechnen, wobei die Gewichte das Verhältnis der Teilintervallängen zu der Länge des gesamten Intervalles sind. Ebenso kann man die Teilintervalle verfeinern, so daß man für ein Intervall $[a, b]$ den relativen quadratischen Fehler als gewichtetes arithmetisches Mittel von relativen quadratischen Fehlern in k beliebigen angrenzenden Intervallen berechnen kann. Im Falle äquidistanter Stützstellen ist offenbar der relative quadratische Flächenfehler über ein Intervall das arithmetisches Mittel der relativen quadratischen Flächenfehler der Teilintervalle. Damit läßt sich folgender Satz formulieren:

***Satz* 1.** Es seien in $[a, b]$ mit $a, b \in \mathbb{R}$ eine Funktion $f : \mathbb{R} \to \mathbb{R}$ sowie ihre Interpolierende $S : \mathbb{R} \to \mathbb{R}$ gegeben. Ferner seien durch $\boldsymbol{x}$ und $\boldsymbol{y}$ Stützstellen der Funktion gegeben, die die Bedingungen (1) und (2) erfüllen. Zudem sei bekannt, daß

$$F_{\mathrm{r}}(x_i, x_{i+1}) < \varepsilon \quad \text{für} \quad i = 0(1)n-1 \tag{6}$$

ist. Dann gilt $F_{\mathrm{r}}(a, b) < \varepsilon$ für das gesamte Intervall $[a, b]$.

Beweis: Nach Definition von $F_{\mathrm{r}}(a, b)$ ist

$$F_{\mathrm{r}}(a, b) = \sum_{i=0}^{n-1} \lambda_i F_{\mathrm{r}}(x_i, x_{i+1}) \; , \quad \lambda_i = \frac{x_i - x_{i+1}}{b-a} \; , \quad i = 0(1)n-1$$

mit $\sum_{i=0}^{n-1} \lambda_i = 1$. Das bedeutet aber, daß $\lambda_i \leq 1$ für $i = 0(1)n-1$ ist. Damit folgt wegen $F_{\mathrm{r}}(x_i, x_{i+1}) < \varepsilon$

$$F_{\mathrm{r}}(a, b) \; < \; \sum_{i=0}^{n-1} \lambda_i \varepsilon \; < \; \varepsilon \; . \qquad \square$$

Damit können wir eine Aussage über den quadratischen Flächenfehler einer Interpolation treffen. Fordern wir nämlich von einer Interpolation, daß der quadratische Fehler F_{r} kleiner als ein Wert ε sein soll, so genügt es, von jedem Teilintervall zu fordern, daß der relative quadratische Fehler kleiner als $\varepsilon/(b-a)$ ist. Wie wir später sehen, kann man, sofern die Interpolierende aus Polynomstücken dritten Grades zusammengesetzt ist, nun folgenden rekursiven Schritt gehen: Sollten einige Intervalle obige Bedingung nicht erfüllen, so unterteile man diese Intervalle durch Einfügen neuer Knoten und untersuche sie nach einer erneuten Berechnung der Interpolierenden. Allerdings muß man

dabei die gesamte Interpolierende neu berechnen, denn eine Änderung in einem Intervall wirkt sich bekanntlich auch auf andere Intervalle aus.
Diese Größe besitzt noch einen weiteren Vorteil: Wenn wir die absoluten quadratischen Flächenfehler über mehrere Teilintervalle ausrechnen, können diese erheblich voneinander abweichen, weil etwa die Längen der Teilintervalle stark differieren. Daher gibt es große Ungenauigkeiten in der maschineninternen Darstellung dieser Zahlen. Beim Operieren (zum Beispiel dem Aufsummieren) mit diesen Fehlern gehen somit viele Stellen verloren. Die Division durch die Intervallängen hebt erstens Unterschiede in den Intervallängen wieder auf, so daß F_r auch als Maß angesehen werden kann, mit dem man Aussagen über den Verlauf des relativen Fehlers geben kann. Zweitens gehen beim Operieren mit dieser Größe eben die Informationen nicht verloren, die in oben genannten Stellen stecken. Um die Güte einer Interpolation messen zu können, geben wir folgende

Definition 2. Die Qualität $g(\varepsilon)$ bezeichne die kleinste Zahl k der notwendigen Stützstellen, für die es eine Zerlegung ζ mit k Knoten gibt, so daß $F_\mathrm{r}(a,b) < \varepsilon$ gilt.

Es liegt auf der Hand, daß eine Funktion um so besser interpoliert wird, je kleiner $g(\varepsilon)$ ausfällt. Ein Algorithmus, der eine Zerlegung mit möglichst wenigen Knoten finden soll, ist um so besser, je näher das Resultat, das er liefert, bei $g(\varepsilon)$ liegt. Um allerdings eine gute *gleichmäßige* Approximation im Sinne der Betragsmaximumnorm zu erhalten, ist anstelle des quadratischen Flächenfehlers $F_\mathrm{r}(a,b)$ der absolute Fehler zu betrachten.

Definition 3. Der maximale absolute Fehler r_max ist

$$r_\mathrm{max}(a,b) = \max_{a\le x\le b} |S(x) - f(x)| \; . \tag{7}$$

Analog können wir eine Qualität $g_\mathrm{abs}(\varepsilon)$ einführen, die sich auf den maximalen absoluten Fehler $r_\mathrm{max}(a,b)$ bezieht.

2.2 Parametrisierungen

In der Literatur werden vielfach Abschätzungen für den maximalen absoluten Fehler der Spline-Interpolation angegeben, zum Beispiel findet man in [5]

$$r_\mathrm{max}(a,b) \;\le\; \frac{1}{384}h^4 \max_{a\le x\le b} |f^{(4)}(x)| \;+\; \frac{2}{405}h^5 \max_{a\le x\le b} |f^{(5)}(x)| + O(h^6) \; , \tag{8}$$

wobei h jeweils den maximalen Abstand benachbarter Punkte auf der x-Achse bezeichnet. Oftmals ist eine vierte oder höhere Ableitung allerdings sehr schwer zu bestimmen. Immerhin erhält man die qualitative Aussage, daß sich die Genauigkeit der Interpolation wie $O(h^4)$ verhält. Darauf aufbauend, suchen wir Algorithmen, die eine gegebene Funktion so interpolieren, daß eine vorgegebene Fehlerschranke nicht überschritten wird.
Das Ergebnis dieser Algorithmen werden stets Parametrisierungen sowie die entsprechenden Koeffizientenvektoren sein. Wie man an der konkreten Aufgabenstellung sieht, liefern äquidistante Parametrisierungen oft nur unbefriedigende Resultate. Also verwendet man entweder eine in der Literatur (vgl. [1], [3]) angegebene Parametrisierung oder

aber eine Parametrisierung, die sich der gegebenen Funktion anpaßt, eine “adaptive“ Parametrisierung. Eine einfache Realisierung bietet sich an, indem man die Knoten so wählt, daß die Kurvenlänge in gleiche Teile aufgeteilt wird. Hat man die gesamte Kurvenlänge der Funktion f ermittelt, so erhält man die Kurvenlänge eines Kurvenstückes zwischen zwei benachbarten Stützstellen zu $\frac{s}{n}$, denn n ist die Anzahl der Intervalle der Splinefunktion.

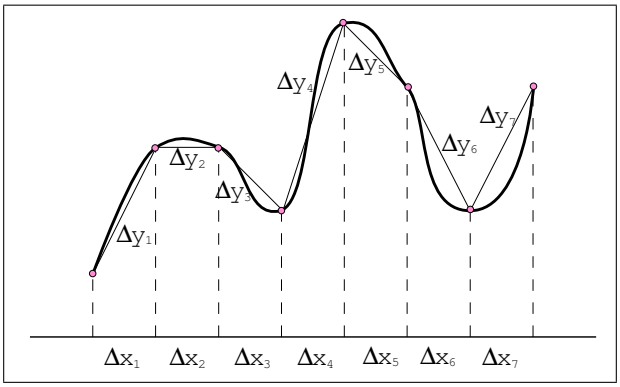


Abbildung 1: Diskretisierung der Bogenlänge

Das ist allerdings nur möglich, wenn die Kurvenlänge s explizit vorliegt. Dann teile man das zu interpolierende Intervall für $\nu = 0(1)k$ in k Teilintervalle $[X_\nu, X_{\nu+1}]$, wobei Δd_ν die angenäherte Kurvenlänge des Teilintervalles mit der Nummer ν und Δx_ν die Länge der Projektion auf die x-Achse sein soll, und ermittle die Länge des Streckenzuges, der die durch diese Aufteilung gegebenen Knoten verbindet (siehe Abb. 1). Außerdem muß noch $X_0 = x_0$ und $X_k = x_n$ gelten.
Sei nun $s(X)$ eine Funktion, die angibt, wie groß die auf diese Weise angenäherte Kurvenlänge im Intervall $[X_0, X]$ ist. Dazu berechne man zunächst sukzessive

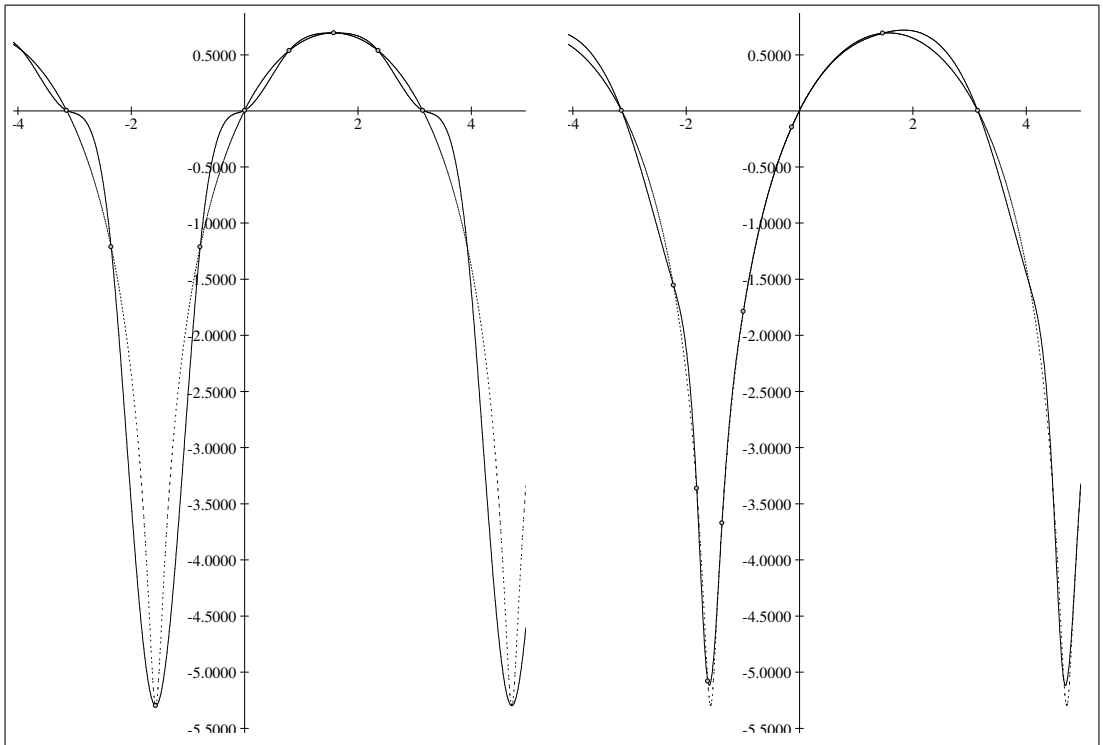
$$s_\nu = s(X_\nu) = \begin{cases} 0 & \text{falls } \nu = 0 \\ s_{\nu-1} + \Delta d_\nu & \text{falls } \nu > 0\,. \end{cases} \tag{9}$$

Nun ist zu klären, wo auf der x-Achse der Punkt X_* liegt, für den $s(X_*) = s_*$ gilt.
Offenbar liegt X_* in dem Intervall, für das $X_\nu \le X_* < X_{\nu+1}$ gilt. Für die angenäherte Kurvenlänge gelten bei der benutzten linearen Annäherung in diesem Intervall folgende Gleichungen:

$$\begin{aligned} X &= X_\nu + \lambda \Delta x_\nu \\ s &= s_\nu + \lambda \Delta d_\nu \\ s &= s(X) \end{aligned} \tag{10}$$

Setzt man $s = s_*$ und $X = X_*$, so erhält man

$$X_* = X_\nu + \frac{(s_* - s_\nu)\Delta x_\nu}{\Delta d_\nu} \quad \text{mit} \quad s_\nu \le s_* < s_{\nu+1}\,. \tag{11}$$



äquidistante Stützstellenverteilung

Stützstellenverteilung nach der Kurvenlänge

Dies ist ein Beispiel dafür, daß eine Stützstellenverteilung, die von der Kurvenlänge der Funktion abhängt, eine bessere Interpolation zu Folge haben kann als eine einfache äquidistante Verteilung

Abbildung 2: Auswirkungen der verschiedenen Parametrisierungen auf die Funktion $f(x) = \log(1.005 + \sin x)$

Um schließlich die gegebene Funktion f gleichmäßig bezüglich der Kurvenlänge zu unterteilen, plaziere man für $i = 1(1)n-1$ den Knoten mit der Nummer i an die Stelle

$$\left(X_\nu + \frac{\left(\frac{i}{n}s_k - s_\nu\right)\Delta x_\nu}{\Delta d_\nu}; f\left(X_\nu + \frac{\left(\frac{i}{n}s_k - s_\nu\right)\Delta x_\nu}{\Delta d_\nu} \right) \right) \quad , \quad s_\nu \leq \frac{i}{n}s_k < s_{\nu+1} \; . \tag{12}$$

Welche Parametrisierung man wählt, hängt immer von der konkreten Funktion ab. Bei Funktionen mit einer Spitze und einem ansonsten flachen Verlauf, wird die obengenannte Parametrisierung bessere Resultate liefern.
Bei einem großen Unterschied der Funktionswerte und vielen Wellen liefert wiederum die äquidistante Aufteilung bessere Resultate (siehe Abb. 2 und Abb. 3), wobei man hier eine Parametrisierung anstreben sollte, die die Knoten in Abhängigkeit vom Radius

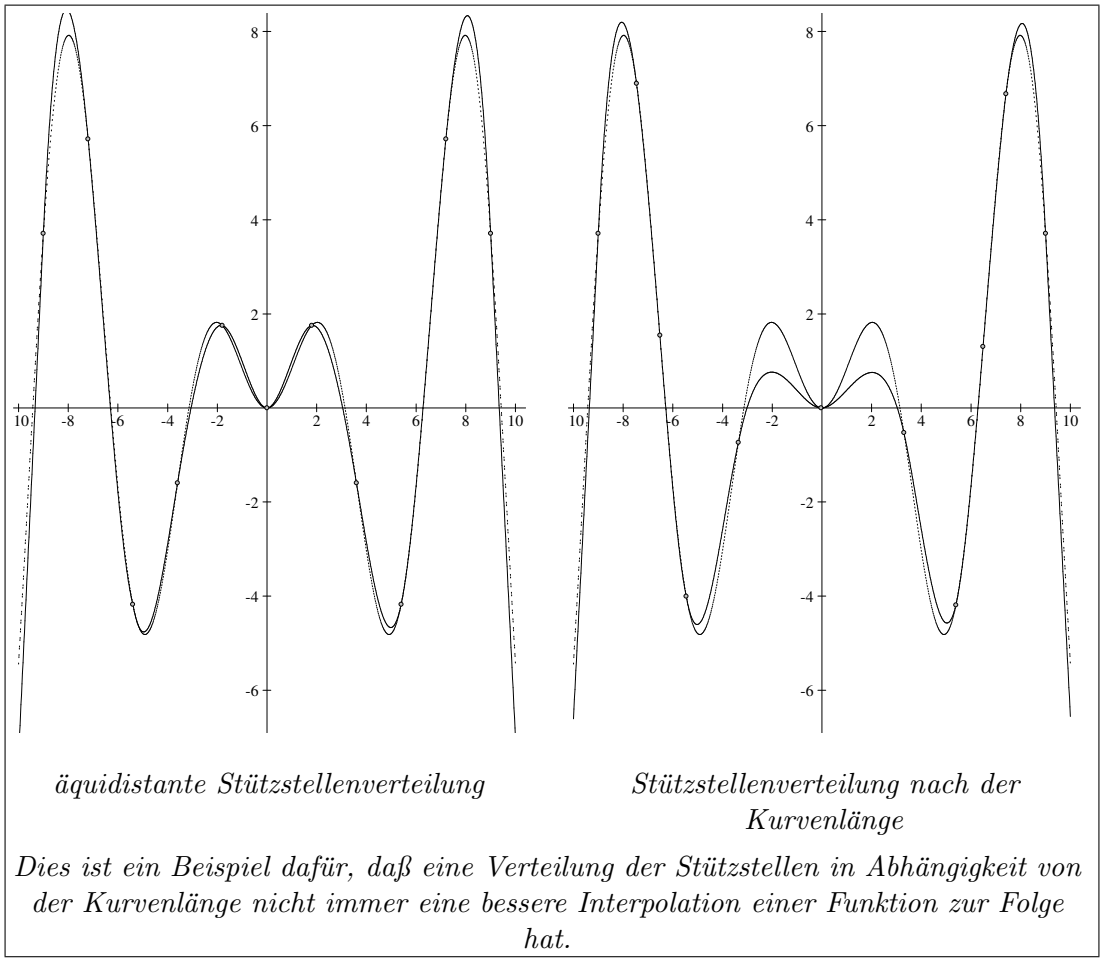


Abbildung 3: Auswirkungen der verschiedenen Parametrisierungen auf die Funktion $f(x) = x \sin x$

des Schmiegekreises der Funktion setzt (vgl. [2]).

2.3 Adaptive Interpolation

Stellen wir uns zuerst die Aufgabe, eine Funktion zu interpolieren, so daß $r_{\max} < \varepsilon$ ist. Wir wissen jedoch nicht von vornherein, wieviele Knoten wir dazu benötigen. Wir wollen aber versuchen, eine möglichst geringe Anzahl von Knoten zu benutzen, denn je mehr Knoten benötigt werden, um so mehr Speicherplatz benötigt man für die Koeffizienten und um so mehr Rechenzeit, um Werte der Kurve zu berechnen.
Wenn es für unsere Kurve eine sehr gute Parametrisierung gibt, wäre folgender Ansatz denkbar: Wir interpolieren die Funktion mit einer Anzahl von n Knoten. Ist das Ergebnis nicht befriedigend, so erhöhen wir die Anzahl der Knoten um einen Faktor und

setzen alle Knoten neu. Wegen $r_{\max} = O(h^4)$ wird sich der maximale Fehler auf etwa den reziproken Wert des Biquadrates dieses Faktors verringern. Allerdings wird man bei dieser Vorgehensweise sicher auch dort Knoten einfügen, wo es nicht erforderlich ist. Darum gehen wir folgendermaßen vor: Wir starten mit einer Anzahl n_{start} von Knoten und interpolieren die Kurve mit Hilfe dieser Knoten. Danach ermitteln wir die Menge Ξ aller Intervalle $I_j = [x_j, x_{j+1}]$ mit $j = 0(1)k-1$, für die $r_{\max}(x_j, x_{j+1}) \geq \varepsilon$ ist. Ist Ξ leer, so ist das Problem gelöst, ansonsten müssen wir diese (und vorerst nur diese) Intervalle erneut unterteilen. Da $r_{\max} = O(h^4)$ gilt, ist das Intervall I_j in $\lfloor \sqrt[4]{r_{\max}(x_j, x_{j+1})/\varepsilon} \rfloor$ Teilintervalle zu unterteilen. Um aber eine Fehlerschranke zu erhalten, die kleiner als ε ist, werden wir ein weiteres Teilintervall zur Unterteilung hinzunehmen. Da die Randknoten des Intervalls I_j schon gegeben sind, werden wir also jeweils $\lfloor \sqrt[4]{r_{\max}(x_j, x_{j+1})/\varepsilon} \rfloor$ neue Knoten einfügen. Danach berechnen wir die neuen Koeffizientenvektoren und wiederholen die letzten Schritte. Wegen des Fehlens der lokalen Kontrolle müssen wir alle Intervalle prüfen, denn durch das Einfügen von Knoten ändern sich sämtliche Komponenten der Koeffizientenvektoren. Somit können wir folgenden Algorithmus formulieren:

Algorithmus 1. *Gegeben: Funktion f mit $[a,b] \subseteq D(f)$, $\varepsilon > 0$*
Gesucht: Interpolierende S mit n Knoten, so daß $r_{\max}(a,b) < \varepsilon$, wobei $n \to g_{\text{abs}}(\varepsilon)$.
1. Wähle $2 \leq n \in \mathbb{N}$.
2. Setze die Knotenvektoren $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{n+1}$.
3. Berechne die Koeffizientenvektoren $\boldsymbol{a} \in \mathbb{R}^{n+1}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d} \in \mathbb{R}^n$.
4. Ermittle die Menge Ξ aller Intervalle I_j $(j \geq 0)$, für die $r_{\max} \geq \varepsilon$.
5. Setze $k := |\Xi|$. Wenn $k = 0$, dann Stop.
6. Füge in die Intervalle I_j für $j = 0(1)k-1$ jeweils $\lfloor \sqrt[4]{r_{\max}(x_j, x_{j+1})/\varepsilon} \rfloor$ neue Knoten ein.
7. Setze $n := n + \sum_{j=0}^{k-1} \lfloor \sqrt[4]{r_{\max}(x_j, x_{j+1})/\varepsilon} \rfloor$ und gehe zu 3.

Auf die Algorithmen zur Berechnung der Koeffizienten wollen wir nicht weiter eingehen, denn darüber findet man genügend Material in der Literatur, etwa in [1] oder [3]. Wesentlich mehr Beachtung dagegen findet der Algorithmus zum Auffinden des Maximums einer Funktion in einem gegebenen Intervall, in unserem Falle von $\phi(x) = |S(x) - f(x)|$. Denn wenn die Funktion f zum Beispiel in Form von Meßwerten vorliegt, ist es schwer, eine Ableitung zu bestimmen und darüber eine Extremwertsuche zu realisieren.
Wenden wir uns nun der Aufgabe zu, die Funktion derart zu interpolieren, daß $F_{\text{r}}(a,b) < \varepsilon$ wird. Zunächst können wir notieren, daß folgendes gilt:

$$\begin{aligned} F(a,b) &= \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} (f(x) - S(x))^2 dx \\ &\leq \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} r_{\max}^2(a,b) dx \;=\; (b-a) r_{\max}^2(a,b) \;. \end{aligned}$$

Um also die Forderung $\frac{F}{b-a} = F_{\text{r}}(a,b) < \varepsilon$ zu erfüllen, ist es hinreichend, die Forderung durch $r_{\max}(a,b) < \sqrt{\varepsilon}$ zu ersetzen und die durch Algorithmus 1 erhaltene Interpolie-

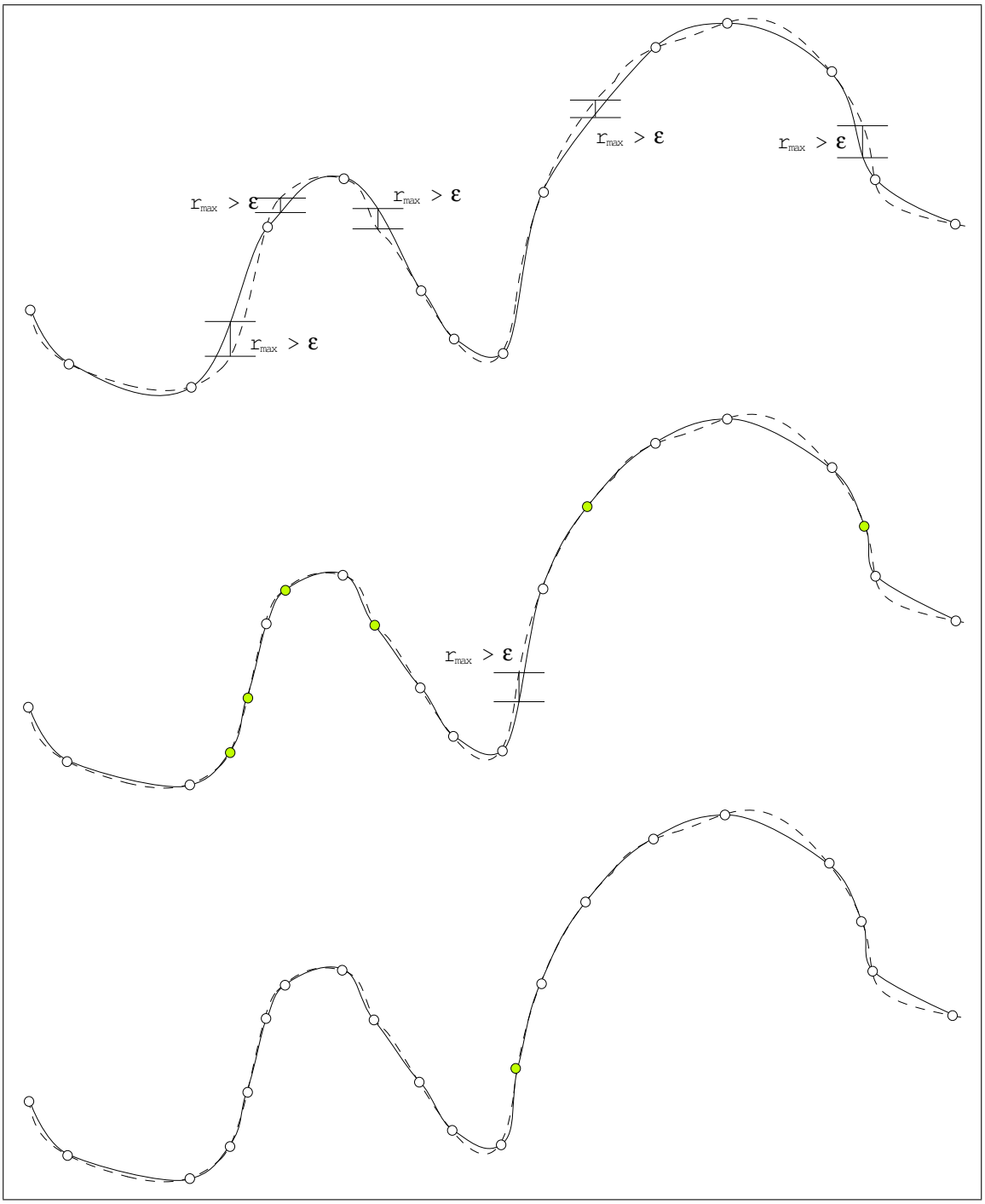


Abbildung 4: Vorgehensweise von Algorithmus 1

rende als Ergebnis dieser Forderung zu nehmen. Diese neue Forderung ist jedoch zu scharf formuliert, und es würden mehr Knoten als nötig gebraucht. Dennoch können wir Algorithmus 1 weitestgehend benutzen, lediglich das Einfügen neuer Knoten gestaltet sich nunmehr anders. Denn wegen $F_\mathrm{r} \leq r_{\max}^2$ mit $r_{\max} = O(h^4)$ ist nunmehr $F_\mathrm{r} = O(h^8)$, und wir müssen in die Teilintervalle (siehe Herleitung von Algorithmus 1) jeweils $\lfloor \sqrt[8]{F_\mathrm{r}(x_j, x_{j+1})/\varepsilon} \rfloor$ Knoten einfügen. Damit lautet unser Einfügealgorithmus nun

Algorithmus 2. *Gegeben: Funktion f mit $[a,b] \subseteq D(f)$, $\varepsilon > 0$*
Gesucht: Interpolierende S mit n Knoten, so daß $F_\mathrm{r}(a,b) < \varepsilon$, wobei $n \to g(\varepsilon)$.
1. Wähle $2 \leq n \in \mathbb{N}$.
2. Setze die Knotenvektoren $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{n+1}$.
3. Berechne die Koeffizientenvektoren $\boldsymbol{a} \in \mathbb{R}^{n+1}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d} \in \mathbb{R}^n$.
4. Ermittle die Menge Ξ aller Intervalle I_j $(j \geq 0)$, für die $F_\mathrm{r} \geq \varepsilon$.
5. Setze $k := |\Xi|$. Wenn $k = 0$, dann Stop.
6. Füge in die Intervalle I_j für $j = 0(1)k-1$ jeweils $\lfloor \sqrt[8]{F_\mathrm{r}(x_j, x_{j+1})/\varepsilon} \rfloor$ neue Knoten ein.
7. Setze $n := n + \sum_{j=0}^{k-1} \lfloor \sqrt[8]{F_\mathrm{r}(x_j, x_{j+1})/\varepsilon} \rfloor$ und gehe zu 3.

Schließlich stehen wir vor dem Problem, daß $F_\mathrm{r}(x_j, x_{j+1})$ berechnet werden muß. Das kann wiederum nur auf numerischem Wege geschehen, da das symbolische Integrieren noch weitaus komplizierter und unsicherer ist als das symbolische Differenzieren. Als einfaches Verfahren zur numerischen Integration können wir zum Beispiel die Simpson-Regel verwenden.
Wenden wir uns nochmals der Fehlerschätzung $F_\mathrm{r} < \varepsilon$ zu.Nach Definition dieser Größe ist es nicht nötig, daß *alle* Funktionswerte des Splines eine Fehlerschranke unterschreiten. Das bedeutet, daß Meßwertfehler, wie zum Beispiel solche Spitzen, wie sie Abb. 5 zeigt, nicht immer großen Einfluß auf die Interpolation haben müssen und die Kurve verunstalten. Wir wollen im folgenden betrachten, inwieweit man "Fast-Polstellen", z.B. Punkt $P(x_0, y_0)$ in Abb. 5, unterdrücken kann. Wir nehmen an, daß von der Interpolierenden gefordert wird, daß $F_\mathrm{r} < \varepsilon$ gilt. Außerdem setzen wir voraus, daß es Punkte $P(x_0, y_0), P_1(x_1, y_1), P_2(x_2, y_2)$ auf der Kurve der gegebenen Funktion f gibt, so daß $f''(x) \geq 0$ und $f'(x) \geq 0$ für $x_1 \leq x < x_0$, und $f''(x) \leq 0$ und $f'(x) \leq 0$ für $x_0 < x < x_2$ gilt. Dann soll die Interpolierende durch die Punkte P_1, P_2 gehen. Weiterhin wollen wir zunächst voraussetzen, daß die Interpolierende in (x_1, x_2) vollständig oberhalb der Geraden P_1P bzw. P_2P mit der Interpolierenden nur die Schnittpunkte P_1 bzw. P_2 gemeinsam hat. Zur Abkürzung werde die Fläche des Dreieckes P_1P_2P mit Δ bezeichnet. Unter diesen Voraussetzungen umhüllt das Dreieck P_1P_2P das Gebiet*, das durch die Funktion und die Interpolierende in $[x_1, x_2]$ abgegrenzt wird. Also ist $F(x_1, x_2) \leq \Delta$. Fordern wir

*Es ist in manchen Fällen sicher günstiger, die Voraussetzung folgendermaßen zu formulieren: Sind P_1 und P_2 zwei Knotenpunkte, so gebe es einen Punkt P, so daß das Dreieck P_1P_2P das Gebiet, das im Intervall $[x_1, x_2]$ durch die Funktion f und ihre Interpolierende S eingeschlossen wird, vollständig enthält.

jetzt $\Delta \leq \varepsilon(x_2 - x_1)$, so ist

$$F(x_1, x_2) \leq \Delta \leq \varepsilon(x_2 - x_1) \; . \tag{13}$$

Für die Dreiecksfläche erhalten wir

$$\Delta = \frac{h}{2}\sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2} \tag{14}$$

Somit erhalten wir unter Verwendung von (13) die Beziehung

$$h \cdot \sqrt{\left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 + 1} \leq 2\varepsilon \; , \tag{15}$$

und die Substitution

$$\tan\alpha = m = \frac{y_2 - y_1}{x_2 - x_1} \tag{16}$$

liefert

$$h\sqrt{m^2 + 1} \leq 2\varepsilon \; . \tag{17}$$

Bezeichnet y_0^* den Wert der Geraden P_1P_2 an der Stelle x_0, so erhält man für die Dreieckshöhe

$$h = (y_0 - y_0^*)\cos\alpha \; . \tag{18}$$

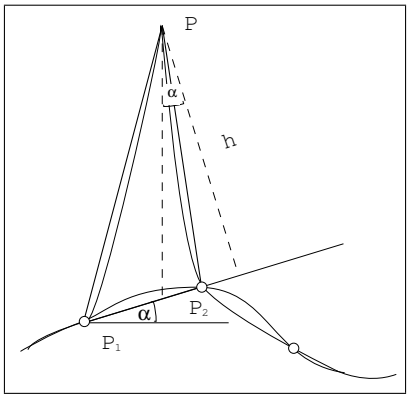


Abbildung 5: Skizze zur Herleitung von Satz 2

Wegen $\tan\alpha = m$ kann man $-\frac{\pi}{2} < \alpha < \frac{\pi}{2}$ wählen, womit $\cos\alpha > 0$ ist. Damit können wir $\cos\alpha = \frac{1}{\sqrt{1+\tan^2\alpha}}$ substituieren, weshalb wir durch Einsetzen in (17)

$$\frac{y_0 - y_0^*}{\sqrt{1 + m^2}}\sqrt{1 + m^2} = y_0 - y_0^* \leq 2\varepsilon \tag{19}$$

erhalten. Wenn also die Funktion und ihre Interpolierende in $[x_1, x_2]$ die oben genannten Voraussetzungen erfüllen und außerdem die Funktion an der Stelle x_0 nicht über den Wert $m(x_0 - x_1) + y_1 + 2\varepsilon$ hinauswächst, so ist

$$F_{\mathrm{r}}(x_1, x_2) < \varepsilon \; . \tag{20}$$

Wenn die Interpolierende nicht vollständig oberhalb der Geraden P_1P_2 liegt, so muß für die Dreiecksfläche

$$\Delta \leq \varepsilon(x_2 - x_1) - \left| \int_{x_1}^{x_2} (S(x) - g(x)dx \right| \tag{21}$$

gelten, wobei $g(x) = y_1 + m(x - x_1)$ ist. Das Integral kann man vereinfachen, denn wenn a_1, b_1, c_1, d_1 die Koeffizienten des kubischen Splines sind, erhält man für den Wert des Integrals:

$$I = \frac{1}{4}d_1(x_2 - x_1)^4 + \frac{1}{3}c_1(x_2 - x_1)^3 + \frac{1}{2}b_1(x_2 - x_1)^2 + a_1(x_2 - x_1) \; . \tag{22}$$

Damit bekommt man anstelle von (17)

$$h \cdot \sqrt{\left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 + 1} \leq 2\varepsilon - |I| \tag{23}$$

und schließlich

$$y_0 - y_0^* \leq 2\frac{|I|}{x_2 - x_1} \; . \tag{24}$$

Allerdings benötigt man in diesem Fall die Koeffizienten der Interpolierenden.

Bemerkung 1. Selbstverständlich kann man in dem Fall, daß die Interpolierende vollständig über der Geraden P_1P_2 liegt, anstelle von (13) auch die schärfere Forderung stellen, daß

$$\Delta \leq \varepsilon(x_2 - x_1) + |I| \tag{25}$$

gilt. Doch um diese Ungleichung verwenden zu können, braucht man ebenfalls die Koeffizienten der Interpolierenden.

Analog kann man für den Fall

$$f''(x) \leq 0 \quad \text{und} \quad f'(x) \leq 0 \qquad \text{für} \quad x_1 < x < x_0 \tag{26}$$

$$f''(x) \geq 0 \quad \text{und} \quad f'(x) \geq 0 \qquad \text{für} \quad x_0 < x < x_2 \tag{27}$$

errechnen, daß $F_{\mathrm{r}}(x_1, x_2) < \varepsilon$ gilt, wenn

- $y_0^* - y_0 \leq 2\varepsilon$, also falls die Interpolierende vollständig unterhalb der Geraden P_1P_2 liegt bzw.
- $y_0^* - y_0 \leq 2\left(\varepsilon - \frac{|I|}{x_2 - x_1}\right)$ im anderen Falle.

Zusammengefaßt ergibt sich daraus folgender

***Satz* 2.** Seien f eine zweimal stetig differenzierbare Funktion, $P(x_0, y_0)$ ein beliebiger Punkt, $P_1(x_1, y_1)$ und $P_2(x_2, y_2)$ zwei Punkte auf dem Graphen von f, m der Anstieg der Geraden P_1P_2 und S ein Polynom mit

$$S(x) = a_1 + b_1(x - x_1) + c_1(x - x_1)^2 + d_1(x - x_1)^3,$$

das den Bedingungen $S(x_1) = y_1$, $S(x_2) = y_2$ genügt und mit den Geraden P_1P und P_2P jeweils genau einen Schnittpunkt besitzt. Zudem seien folgende Bedingungen erfüllt:

- Fall A : $f''(x) \leq 0\,,\ f'(x) \leq 0$ für $x_1 < x < x_0\,,\quad f''(x) \geq 0\,,\ f'(x) \geq 0$ für $x_0 < x < x_2\,,$
  $S(x) \geq y_0 + m(x - x_0)$ oder
- Fall B : $f''(x) \geq 0\,,\ f'(x) \geq 0$ für $x_1 < x < x_0\,,\quad f''(x) \leq 0\,,\ f'(x) \leq 0$ für $x_0 < x < x_2\,,$
  $S(x) \leq y_0 + m(x - x_0)\,.$

Dann ist stets $F_{\mathrm{r}}(x_1, y_1) < \varepsilon$, sofern für $y_0^* = y_1 + x_0 m$ eine der folgenden Voraussetzungen zutrifft:

- $|y_0 - y_0^*| \leq 2\varepsilon$ für den Fall, daß f und S beide vollständig oberhalb (Fall A) bzw. unterhalb (Fall B) der Geraden P_1P_2 liegt,
- $|y_0 - y_0^*| \leq 2\varepsilon - |I|$ für den Fall, daß f und S nicht beide vollständig oberhalb (Fall A) bzw. unterhalb (Fall B) der Geraden P_1P_2 liegt, wobei

$$I = \frac{1}{4}d_1(x_2 - x_1)^4 + \frac{1}{3}c_1(x_2 - x_1)^3 + \frac{1}{2}b_1(x_2 - x_1)^2 + a_1(x_2 - x_1) \tag{28}$$

gilt.

3 Die Methode der vierten Potenz

3.1 Die einfache Methode

Sollen in eine interpolierende Funktion $S(x)$, bestehend aus kubischen Splines, neue Knoten eingefügt werden, so muß die gesamte Interpolierende neu berechnet werden. Das kostet bekanntermaßen viel Rechenzeit.

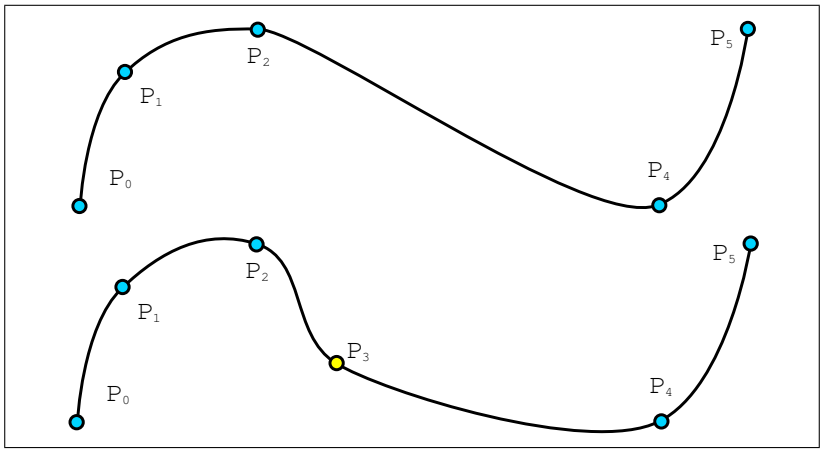


Abbildung 6: Einfügen eines neuen Knotens

Nehmen wir an, wir haben eine Interpolierende gegeben und wollen einen Punkt P_3 einfügen (siehe Abb. 6). Dies bedeutet, daß das Kurvenstück P_2P_4 durch zwei neue

Stücken P_2P_3 und P_3P_4 ersetzt werden soll, von denen die Koeffizienten nicht bekannt sind. Es werden somit 8 unbekannte Koeffizienten gesucht. Da $S(x)$ überall zweimal stetig differenzierbar sein soll, muß das auch an den Knoten P_2, P_3 und P_4 gelten. Wenn nun die Stücken der neuen Interpolierenden mit S_i und die der alten Interpolierenden mit S_i^* bezeichnet sind, so heißt das, daß am Knoten P_2 die Bedingungen

$$S_2(x_2) = S_2^*(x_2)\ , \quad S_2'(x_2) = S_2^{*\prime}(x_2)\ , \quad S_2''(x_2) = S_2^{*\prime\prime}(x_2), \tag{29}$$

am Knoten P_3 die Bedingungen

$$S_3(x_3) = y_3\ , \quad S_3(x_3) = S_2(x_3)\ , \quad S_3'(x_3) = S_2'(x_3)\ , \quad S_3''(x_3) = S_2''(x_3) \tag{30}$$

und am Knoten P_4 die Bedingungen

$$S_4(x_3) = S_3(x_3)\ , \quad S_4'(x_3) = S_3'(x_3)\ , \quad S_4''(x_3) = S_3''(x_3) \tag{31}$$

zu erfüllen sind. Von diesen 10 Bedingungen geht keine aus einer anderen hervor, womit ein überbestimmtes System entsteht. Um nicht Bedingungen streichen zu müssen, kann man einen weiteren Koeffizienten hinzunehmen, so daß Splinestücken der Form

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 + e_i(x - x_i)^4 \tag{32}$$

notiert werden. Durch die Hinzunahme einer vierten Potenz können wir nun diese Bedingungen in folgende Gleichungen schreiben: Am Knoten P_2 erhält man

$$\begin{aligned} a_2 &= y_2^* \\ b_2 &= b_2^* \\ 2c_2 &= 2c_2^*\ , \end{aligned} \tag{33}$$

am Knoten P_3

$$\begin{aligned} a_3 &= y_3 \\ a_3 &= a_2 + b_2(x_3 - x_2) + c_2(x_3 - x_2)^2 + d_2(x_3 - x_2)^3 + e_2(x_3 - x_2)^4 \\ b_3 &= b_2 + 2c_2(x_3 - x_2) + 3d_2(x_3 - x_2)^2 + 4e_2(x_3 - x_2)^3 \\ 2c_3 &= 2c_2 + 6d_2(x_3 - x_2) + 12e_2(x_3 - x_2)^2 \end{aligned} \tag{34}$$

und am Knoten P_4

$$\begin{aligned} a_4^* &= a_3 + b_3(x_4^* - x_3) + c_3(x_4^* - x_3)^2 + d_3(x_4^* - x_3)^3 + e_3(x_4^* - x_3)^4 \\ b_4^* &= b_3 + 2c_3(x_4^* - x_3) + 3d_3(x_4^* - x_3) + 4e_3(x_4^* - x_3)^3 \\ 2c_4^* &= 2c_3 + 6d_3(x_4^* - x_3) + 12e_3(x_4^* - x_3)^2\ . \end{aligned} \tag{35}$$

Da man die Werte für a_2, b_2, c_2 und a_3 direkt erhält, muß man nur für die 6 Koeffizienten d_2, e_2, b_3, c_3, d_3 und e_3 ein Gleichungssystem

$$A \cdot \begin{pmatrix} d_2 \\ e_2 \\ b_3 \\ c_3 \\ d_3 \\ e_3 \end{pmatrix} = \begin{pmatrix} a_3 - a_2 - b_2(x_3 - x_2) - c_2(x_3 - x_2)^2 \\ -b_2 - 2c_2(x_3 - x_2) \\ -2c_2 \\ a_4 - a_3 \\ b_4 \\ 2c_4 \end{pmatrix} \tag{36}$$

mit der Koeffizientenmatrix

$$A = \begin{pmatrix} (x3 - x_2^*)^3 & (x_3 - x_2^*)^4 & 0 & 0 & 0 & 0 \\ 3(x_3 - x_2^*)^2 & 4(x_3 - x_2^*)^2 & -1 & 0 & 0 & 0 \\ 6(x_3 - x_2^*) & 12(x_3 - x_2^*) & 0 & -1 & 0 & 0 \\ 0 & 0 & x_4 - x_3 & (x_4^* - x_3)^2 & (x_4^* - x_3)^3 & (x_4^* - x_3)^4 \\ 0 & 0 & 1 & 2(x_4^* - x_3) & 3(x_4^* - x_3)^2 & 4(x_4^* - x_3)^3 \\ 0 & 0 & 0 & -1 & 6(x_4^* - x_3) & 12(x_4^* - x_3)^2 \end{pmatrix} \tag{37}$$

lösen. Sollen nun k neue Knoten P_i für $i = 1(1)k$ zwischen zwei Knoten P_m und P_{m+1} eingefügt werden, so kann man die neuen Koeffizienten auf folgendem Wege berechnen: Man bestimmt zunächst die ersten $k-1$ neuen Koeffizienten aus der Forderung, daß S zweimal stetig differenzierbar sein soll. Zunächst setze man $P_m = P_0$. Damit erhält man

$$\begin{aligned} e_0 &= 0 \\ d_0 &= \frac{y_1 - y_0}{(x_1 - x_0)^3} - \frac{b_0}{(x_1 - x_0)^2} - \frac{c_0}{x_1 - x_0} - e_0(x_1 - x_0) \; . \end{aligned} \tag{38}$$

Weiterhin erhält man durch sukzessives Einsetzen für $i = 1(1)k - 2$

$$\begin{aligned} a_{i+1} &= y_{i+1} \\ b_{i+1} &= b_i + 2c_i(x_{i+1} - x_i) + 3d_i(x_{i+1} - x_i)^2 + 4e_i(x_{i+1} - x_i)^3 \\ 2c_{i+1} &= 2c_i + 6d_i(x_{i+1} - x_i) + 12e_i(xi + 1 - x_i)^2 \\ e_{i+1} &= 0 \\ d_{i+1} &= \frac{y_{i+2} - y_{i+1}}{(x_{i+2} - x_{i+1})^3} - \frac{b_{i+1}}{(x_{i+2} - xi + 1)^2} - \frac{c_{i+1}}{x_{i+2} - x_{i+1}} - e_{i+1}(x_{i+2} - x_{i+1}) \; . \end{aligned} \tag{39}$$

Den letzten Knoten, an den die beiden noch fehlenden Intervalle angrenzen, fügt man – wie oben beschrieben – mit Hilfe der vierten Potenz ein.

3.2 Verbesserte Methoden der vierten Potenz

Die einfache Methode der vierten Potenz liefert häufig unbefriedigende Resultate, weil beim Einfügen eines Knotens nur die Ableitungen der linken Nachbarknoten berücksichtigt werden. Daher wird nach geeigneten Alternativen gesucht.
So kann man zum Beispiel die Ableitung in einem Knoten schätzen, indem man den Anstieg der Geraden berechnet, die durch den linken und rechten Nachbarknoten verläuft und als Ableitung für diesen Punkt vorschreibt (siehe Abb. 7). Für den Abschnitt P_2P_3 der Kurve sehen die Gleichungen dann folgendermaßen aus:

$$\begin{aligned} m &= \frac{y_4 - y_2}{x_4 - x_2} \\ a_3 &= a_2 + b_2(x_3 - x_2) + c_2(x_3 - x_2)^2 + d_2(x_3 - x_2)^3 + e_2(x_3 - x_2)^4 \\ m &= b_2 + 2c_2(x_3 - x_2) + 3d_2(x_3 - x_2)^2 + 4e_2(x_3 - x_2)^3 \; . \end{aligned} \tag{40}$$

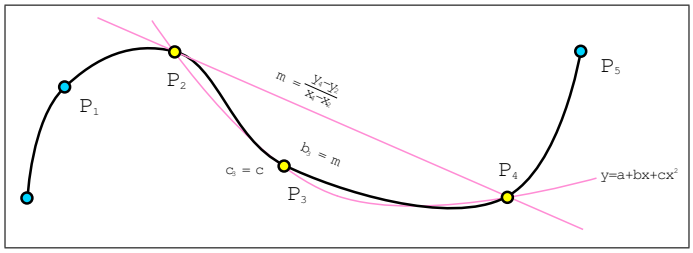


Abbildung 7: Schätzen der ersten und zweiten Ableitung

Wenn man annimmt, daß a_2, b_2 und c_2 bekannt sind, erhält man durch Umstellung die Koeffizienten

$$e_2 = -3\frac{a_3 - a_2}{(x_3 - x_2)^4} + \frac{c_2}{(x_3 - x_2)^2} + \frac{2b_2 + m}{(x_3 - x_2)^3} \tag{41}$$

$$d_2 = \frac{a_3 - a_2}{(x_3 - x_2)^3} - \frac{b_2}{(x_3 - x_2)^2} - \frac{c_2}{x_3 - x_2} + e_2(x_3 - x_2)\ . \tag{42}$$

Als zweite Möglichkeit bietet sich an, die zweite Ableitung zu schätzen, indem man durch die Punkte P_2, P_3 und P_4 eine quadratische Parabel legt und deren zweite Ableitung als Schätzwert annimmt. Die Koeffizienten der Parabel durch diese drei Punkte $y = a + bx + cx^2$ erhält man durch Lösung des Gleichungssystems mit VANDERMONDEscher Matrix

$$\begin{pmatrix} 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \\ 1 & x_4 & x_4^2 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} y_2 \\ y_3 \\ y_4 \end{pmatrix}\ . \tag{43}$$

Die geschätzte zweite Ableitung beträgt also $2c$. Die Koeffizienten des Splines berechnen sich jetzt durch analoges Aufstellen der Gleichungen wie folgt:

$$a_2 + b_2(x_3 - x_2) + c_2(x_3 - x_2)^2 + d_2(x_3 - x_2)^3 + e_2(x_3 - x_2)^4 = a_3 \tag{44}$$

$$2c_2 + 6d_2(x_3 - x_2) + 12e_2(x_3 - x2_)^2 = 2c\ . \tag{45}$$

Wenn man nun wiederum davon ausgeht, daß a_2, b_2 und c_2 bekannt sind, erhält man durch Umstellen die gesuchten Koeffizienten

$$e_2 = -\frac{a_3 - a_2}{(x_3 - x_2)^4} + \frac{1}{3}\frac{c + 2c_2}{(x_3 - x_2)^2} + \frac{b_2}{(x_3 - x_2)^3} \tag{46}$$

$$d_2 = \frac{a_3 - a_2}{(x_3 - x_2)^3} - \frac{b_2}{(x_3 - x_2)^2} - \frac{c_2}{x3 - x2} + e_2(x_3 - x_2)\ . \tag{47}$$

4 Die Berechnung der Splines

Funktionswerte einer ermittelten Splinefunktion kann man sehr schnell sukzessive berechnen. Will man jedoch Funktionswerte an einigen willkürlichen Argumentstellen bestimmen, so muß man erst den jeweiligen lokalen Spline suchen. Bei einer großen Anzahl

n von Stützstellen ist eine einfache lineare Suche nicht mehr akzeptabel, da man pro zu berechnenden Funktionswert einen Suchaufwand von durchschnittlich $\frac{n}{2}$ hat.
Seien die Stützstellen der Splines in einem Stützstellenvektor $\boldsymbol{x} = (x_0, x_1, \ldots, x_n)^T$ gespeichert, und die Interpolationsgrenzen $a = x_0$ und $b = x_n$ seien darin enthalten. Sei nun der Index k gesucht, so daß für ein gegebenes $a \leq x \leq b$ gilt

$$k = \begin{cases} n-1 & \text{falls } \; x = b\,, \\ k' \text{ mit } \; x_{k'} \leq x < x_{k'+1} & \text{sonst .} \end{cases} \tag{48}$$

Sind die Stützstellen äquidistant, so kann man den Index k leicht mittels

$$k = \left\lfloor n \frac{x-a}{b-a} \right\rfloor \quad \text{für} \quad a \leq x < b \tag{49}$$

berechnen. Sind die Stützstellen dagegen nicht äquidistant angeordnet, so muß man den Index k über ein Suchverfahren ermitteln. Wie oben erwähnt, hat eine einfache lineare Suche einen Aufwand von $\frac{n}{2}$ Operationen. Diesen Aufwand kann man durch eine binäre Suche nach dem Prinzip einer *Intervallschachtelung* erheblich unterbieten, der Aufwand hierbei beträgt im Durchschnitt $\log_2 n$. Jedoch kann man auch dieses Verfahren weiter verbessern. Dazu definieren wir eine Funktion $\text{Index}_m : \mathbb{R}^{n+1} \times \mathbb{N}_0 \to \mathbb{N}_0$, die zu festem $m \in \mathbb{N}$ einen Index liefert. Mit obigem $\boldsymbol{x} \in \mathbb{R}^{n+1}$ und $p \in \mathbb{N}_0$ erhalten wir

$$\text{Index}_m(\boldsymbol{x}, p) \;=\; \begin{cases} n & \text{falls } \; p = n \;\; , \\ k^0 \;\; \text{mit } \; x_{k^0} \leq \left(1 - \frac{p}{m}\right) a + \frac{p}{m} b < x_{k^0+1} & \text{sonst .} \end{cases} \tag{50}$$

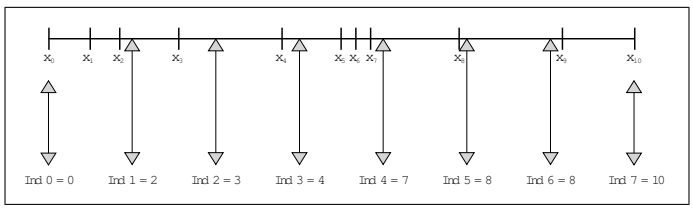


Abbildung 8: Zuordnung der Indizes

Damit liegt der Punkt x in $[x_{k_l}, x_{k_r}]$, wenn

$$\begin{aligned} k_l &= \text{Index}_m(\boldsymbol{x}, k^*) \quad \text{und} \\ k_r &= \text{Index}_m(\boldsymbol{x}, k^* + 1)\,, \quad \text{wobei} \quad k^* = \left\lfloor n \frac{x-a}{b-a} \right\rfloor \text{ ist.} \end{aligned} \tag{51}$$

Die Funktion Index_m wird man als Vektor aus $\mathbb{R}^m$ abspeichern, so daß man schnell auf die Funktionswerte zugreifen kann. Im Durchschnitt besitzen die auf diese Weise eingeschränkten Suchintervalle nur noch die rund $\frac{1}{m}$-fache Anzahl von Indizes, so daß sich der Suchaufwand auf rund $\frac{n}{2m}$ bzw. $\frac{\log_2 n}{m}$ Operationen bei der linearen Suche bzw. Intervallsuche reduziert.

Wie man leicht sieht, wird das verbleibende Intervall kleiner, wenn m vergrößert wird. Wenn sogar

$$m \geq \frac{b-a}{h_{\min}} \qquad \text{mit} \quad h_{\min} = \min_{i=0(1)n-1} (x_{i+1} - x_i) \tag{52}$$

gilt, so liefert Index_m den *Index des lokalen Splines*. Dieser letzte Ansatz lohnt sich immer, wenn die Stützstellen einigermaßen gleichmäßig verteilt sind, also wenn

$$\frac{h_{\max}}{h_{\min}} < 10 \qquad \text{mit} \quad h_{\max} = \max_{i=0(1)n-1} (x_{i+1} - x_i) \tag{53}$$

erfüllt ist.

5 Zwei Beispiele

In diesem Abschnitt geht es darum, die Wirkungsweise der vorgestellten Algorithmen anhand von zwei Beispielen zu demonstrieren. Vorher aber noch einige Bemerkungen zu parametrischen Splines.

Interessanter als einfache Kurven der Art

$$y = f(x) \tag{54}$$

sind parametrische Kurven der Art

$$\boldsymbol{x} = \boldsymbol{f}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}. \tag{55}$$

Für diesen Typ von Kurven benötigt man auch eine parametrische Interpolierende der Form

$$\boldsymbol{S}(t) = \begin{pmatrix} S_x(t) \\ S_y(t) \end{pmatrix}, \tag{56}$$

sogenannte parametrische Splines. Die Komponenten dieser Interpolierenden können im Prinzip völlig unabhängig voneinander sein. Es erweist sich jedoch als günstig, wenn beide Komponenten $S_x(t)$ und $S_y(t)$ den gleichen Stützstellenvektor $\boldsymbol{t} = (t_0, t_1, \ldots, t_n)^T$ besitzen. Weiterhin erweist es sich als günstig, den maximalen Fehler neu festzulegen:

$$\begin{aligned} r_{\max}(a,b) &= \max_{a \leq t \leq b} |\boldsymbol{S}(t) - \boldsymbol{f}(t)| \\ &= \max_{a \leq t \leq b} \left| \begin{pmatrix} S_x(t) - x(t) \\ S_y(t) - y(t) \end{pmatrix} \right| \\ &= \max_{a \leq t \leq b} \sqrt{(S_x(t) - x(t))^2 + (S_y(t) - y(t))^2} \end{aligned} \tag{57}$$

Dieser Ansatz kann natürlich auch auf Raumkurven und Kurven höherer Dimension verallgemeinert werden.

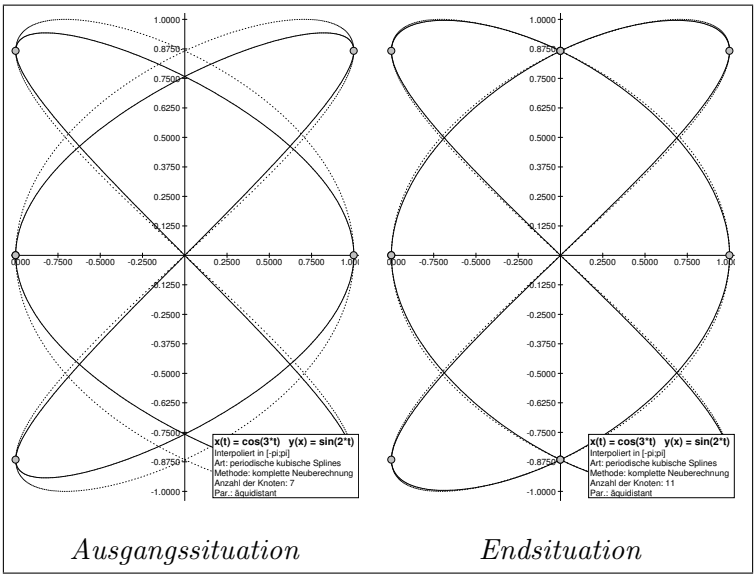


Abbildung 9: Grafische Darstellung der Entwicklung der Interpolation im Beispiel 1

5.1 Ein Beispiel aus der Physik

Zunächst ein Beispiel mit einer parametrischen Funktion, nämlich

$$\boldsymbol{x} = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} \cos 3t \\ \sin 2t \end{pmatrix}. \tag{58}$$

Diese Funktion beschreibt eine LISSAJOUS-Figur, wie sie in der Physik zum vergleichenden Messen von Frequenzen benutzt wird. Da diese Funktion periodisch ist, wird als Randbedingung *periodisch* und als Interpolationsintervall $[a, b] = [-\pi, \pi]$ gewählt. Gestartet wird mit 7 Knoten.

i	t_i	$r_{\max}(t_i, t_{i+1})$
0	−3.142	0.10885719
1	−2.094	0.02423982
2	−1.047	0.10885719
3	0.000	0.10885719
4	1.047	0.02423982
5	2.094	0.10885719
6	3.142	

Tabelle 5.1.1: Ausgangssituation

i	t_i	$r_{\max}(t_i, t_{i+1})$
0	−3.142	0.02009688
1	−2.618	0.02040131
2	−2.094	0.02012864
3	−1.047	0.02040131
4	−0.524	0.02009688
5	0.000	0.02009688
6	0.524	0.02040131
7	1.047	0.02012864
8	2.094	0.02040131
9	2.618	0.02009688
10	3.142	

Tabelle 5.1.2: Endsituation

Tabellierte Fehlerentwicklung der Funktion $\boldsymbol{x} = \begin{pmatrix} \cos 3t \\ \sin 2t \end{pmatrix}$

Ausgehend von dieser Situation, soll die Interpolation verbessert werden, so daß für den

absoluten Fehler gilt:

$$r_{\max} < 0.05, \tag{59}$$

was offenbar nicht erfüllt ist. Betrachtet man sich Tabelle 5.1.1, so sieht man, daß in vier Intervalle Knoten eingefügt werden müssen, und zwar jeweils

$$k = \left\lfloor \sqrt[4]{\frac{0.10885719}{0.05}} \right\rfloor = 1 \tag{60}$$

Knoten. Damit eregeben sich die in Tabelle 5.1.2 aufgelisteten Fehler. Die Abbildung 9 zeigt einen Ausdruck des oben erwähnten Programmes.

5.2 Eine Spirale

Das folgende Beispiel zeigt eine weitere parametrische Funktion, nämlich

$$\boldsymbol{x} = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} \sqrt{t}\sin t \\ \sqrt{t}\cos t \end{pmatrix}. \tag{61}$$

Interpoliert wurde im Intervall $[a, b] = [0.5, 9]$, mit 6 Knoten beginnend. Wie im vorangegangenen Beispiel sei gefordert

$$r_{\max} < 0.05. \tag{62}$$

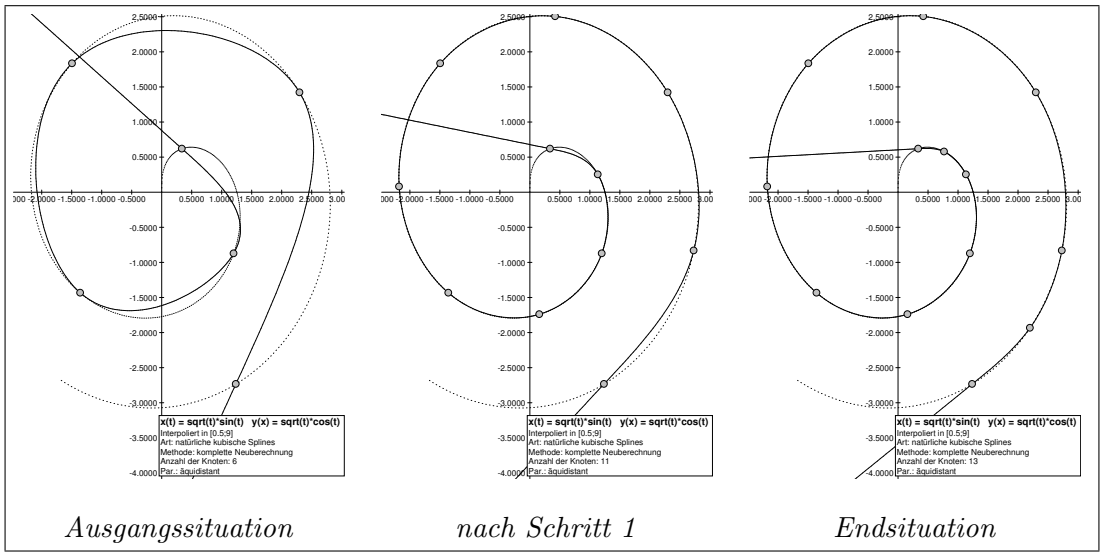


Abbildung 10: Grafische Darstellung der Entwicklung der Interpolation im Beispiel 2

Da es sich bei dem Graphen der Funktion um keine geschlossene Kurve handelt, kann man zwischen den Randbedingungen *not-a-knot* oder *natürlich* wählen – hier wurde die letztere Randbedingung gewählt.
Um die geforderte Genauigkeit zu erreichen, benötigt der Algorithmus zwei Durchläufe. Die Entwicklung der Fehler wird in den Tabellen 5.2.1 bis 5.2.3 dargestellt. Figur 10 zeigt grafisch, wie der Algorithmus vorgeht. Diese Bilder wurden ebenfalls mit dem oben erwähnten Programm erzeugt.

i	t_i	$r_{\max}(t_i, t_{i+1})$
0	0.5	0.36495018
1	2.2	0.16907287
2	3.9	0.10171848
3	6.6	0.25010092
4	7.3	0.60546226
5	9.0	

Tabelle 5.2.1: Ausgangssituation

i	t_i	$r_{\max}(t_i, t_{i+1})$
0	0.50	0.07068537
1	1.35	0.01823096
2	2.20	0.00321642
3	3.05	0.00508662
4	3.90	0.00383264
5	4.75	0.00387275
6	5.60	0.00649634
7	6.45	0.00595518
8	7.30	0.03060556
9	8.15	0.11787894
10	9.00	

Tabelle 5.2.2: nach Schritt 1

i	t_i	$r_{\max}(t_i, t_{i+1})$
0	0.500	0.01750917
1	0.925	0.00526455
2	1.350	0.00365312
3	2.200	0.00459925
4	3.050	0.00365298
5	3.900	0.00399586
6	4.750	0.00439196
7	5.600	0.00372726
8	6.450	0.00607276
9	7.300	0.01404742
10	8.150	0.00725973
11	8.575	0.02753450
12	9.000	

Tabelle 5.2.3: Endsituation

Tabellierte Fehlerentwicklung der Funktion $\boldsymbol{x} = \begin{pmatrix} \sqrt{t}\sin t \\ \sqrt{t}\cos t \end{pmatrix}$

5.3 Bemerkungen zu den Beispielen

Wie man an diesen beiden Beispielen sieht, sind die am Ende erhaltene erhaltenen Stützstellen nicht äquidistant angeordnet. Eine einfache Interpolation mit der Anzahl der Ende erhaltenen Knoten führt zu einem schlechteren Resultat (siehe Abb. 11). Das liegt daran, daß der Algorithmus genau dort Knoten einfügt, wo Abweichungen von der geforderten Fehlertoleranz auftreten.
Als letztes ist das Ergebnis einer Interpolation mit der Methode der vierten Potenz gezeigt (Abb. 12). Auffallend am Graphen der Interpolierenden ist, daß die erzeugte Kurve nicht mehr glatt ist. Der Grund hierfür liegt darin, daß kubische Splines die Eigenschaft besitzen, eine Kurve derart zu interpolieren, daß die Gesamtkrümmung minimal ist. Das heißt, die natürliche kubische Spline-Funktion $S(t)$ minimiert das Integral

$$\int_a^b (g''(t))^2 dt \quad \forall g \in C^2[a,b] \text{ mit } g(t_i) = f(t_i)\ ,\ 0 \le i \le n \tag{63}$$

Diese Aussage findet man als Satz von Holladay. Wegen dieser Minimalkrümmung sieht die Interpolierende glatt aus. Nimmt man aber Polynomstücken höheren Grades, so erreicht man nicht mehr die minimale Krümmung. Je mehr Potenzen man benutzt, um so größer wird die Gesamtkrümmung der Interpolierenden, was man besonders beim Lagrangeschen Interpolationspolynom sieht.
Aus diesem Grund sollte man die Methoden der vierten Potenz nur lokal anwenden, also wenn die Interpolierende bis auf einige wenige Intervalle die geforderte Toleranz unterschreitet.
Die hier vorgestellten Verfahren wurden als Programmsystem SPLINES in der Programmiersprache Turbo-PASCAL mit dem Compiler von Borland erstellt und implementiert. Um diese Verfahren einem breiteren Nutzerkreis zugänglich zu machen, wurden sie in einer *dynamic link library* (DLL) zusammengefaßt und können unter MS Windows mit jedem Compiler, der Aufrufe von Funktionen aus DLLs unterstützt, benutzt werden.

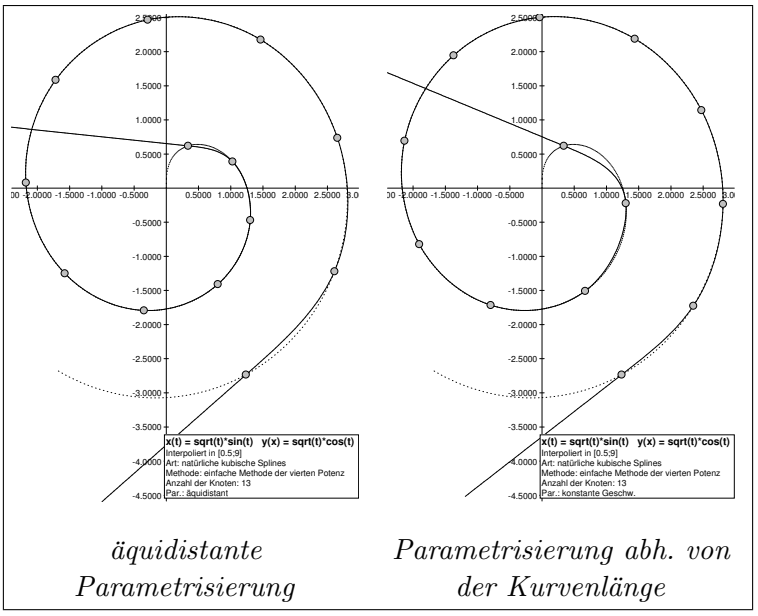


Abbildung 11: Die Kurve aus Beispiel 2 wurde mit der Anzahl der Knoten interpoliert, die man am Ende des Interpolationsalgorithmus' erhält.

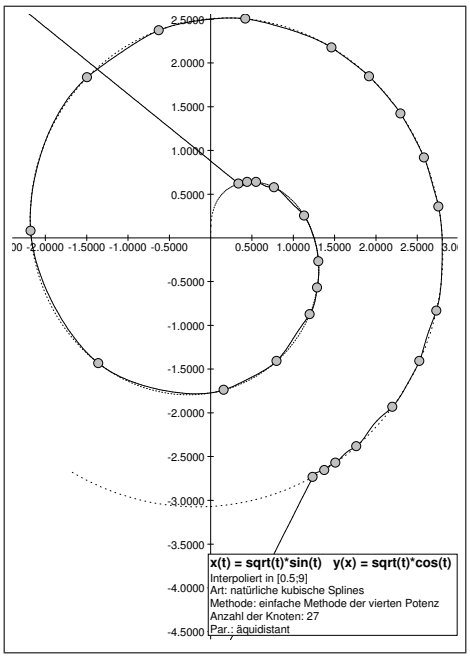


Abbildung 12: Ergebnis der Methode der vierten Potenz

Literatur

[1] Deuflhard, P.; Hohmann, A.: Numerische Mathematik I . 2. Aufl., Walter de Gruyter, Berlin - New York 1993.

[2] Engeln-Müllges, G.; Reutter, F.: Numerik-Algorithmen mit ANSI-C-Programmen. 1. Aufl., Wissenschaftsverlag Mannheim 1993.

[3] Farin, G.: Kurven und Flächen im Computer Aided Geometric Design. 2. Aufl., Vieweg-Verlag Braunschweig 1994.

[4] Golub, G.H.; Ortega, J.M.: Wissenschaftliches Rechnen und Differentialgleichungen. Heldermann-Verlag Berlin 1995.

[5] Maess, G.: Vorlesungen über Numerische Mathematik, Band 2. Akademie-Verlag Berlin 1988.

[6] Späth, H.: Numerik - Eine Einführung für Mathematiker und Informatiker. Vieweg-Verlag Braunschweig 1994.