Interpolierende Splines – mathematische Grundlagen und praktische Anwendung

Max Sauerbrey

Facharbeit 2013

Zusammenfassung

Es kommt in vielen Bereichen der Technik vor, dass einen Datensatz von Punktpaaren geliefert wird und eine glatte Kurve durch diese Punkte gefragt ist. Zum Beispiel wenn man ein Objekt erfasst hat (anhand eines Punkterasters) und will nun einen glatten Verlauf erkennen. Hierzu kann man beispielsweise lineare Interpolation benutzen, was aber nicht sehr schön aussieht und vor allem nicht wirklich glatt ist. Andererseits kann man Polynome durch die Punkte legen. Bei vielen Punkten werden diese jedoch enorm groß und umständlich zu handhaben. Bei 1000 Punkten etwa, wäre unter schlechten Umständen ein Polynom 999.Grades erforderlich. Einen einfachen Ausweg bietet die Spline-Interpolation. Hier kann man schon mit Polynomen maximal 3.Grades eine ausreichend glatte Kurve erhalten und sogar noch weitere Anforderungen an den Graphen stellen. Splines sind nämlich abschnittsweise Polynome. Die Abschnitte ergeben sich aus den Intervallen der Stützstellen des Datensatzes. Nun kann und soll ein Spline n-ter Ordnung, also bestehend aus Polynomen mit Grad ≤n, auch (n-1)-mal stetig-differenzierbar sein. Um nun einen Spline 3.Ordnung für einen Datensatz von n Punktepaaren zu finden gilt es ein Gleichungssystem mit n Variablen (nämlich den zweiten Ableitungen in den Stützstellen) zu lösen. Sind diese gefunden lassen sich aus ihnen und den Datenpaaren die abschnittsweisen Polynome errechnen.

Inhalt

1. Einleitung

Oft sind bestimmte Punkte vorgegeben und es gilt eine Abbildung zu finden, die all diese beinhaltet. Eine solche Funktion „interpoliert“ diesen Datensatz. Es gibt mehrere Arten eine solche Funktion zu finden, zum Beispiel kann man einfach ein Polynom suchen und die Koeffizienten geschickt wählen.

2. Interpolation

Das Interpolieren bezeichnet das Legen einer glatten Kurve durch gegebene Punktepaare z.B. eines Funktionsgraphen.

Es wird also ein f gesucht, sodass $f(x_i) = y_i$ für bestimmte $(x_i, y_i) \in \mathbb{R}^2, i = 0, 1, \dots, n$

mit $x_i \neq x_k,\ i \neq k$ (2.0.1)

3. Interpolierende Polynome

Nun kann man ein Polynom $P \in \prod_n,\ P(x) = \sum_{k=0}^{n} a_k\, x^k$ suchen, welches Bedingung (2.0.1) erfüllt. Hierzu ist folgendes Gleichungssystem zu lösen

$$\underbrace{\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ & \vdots & & & \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix}}_{Vandermondematrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix} \quad (2.0.1)$$

3.1 Vandermondematrix

Satz 3.1.1:

Behauptung: Für jede $n \times n$-Vandermondematrix, $n \in \mathbb{N}\backslash\{0,1\}$, gilt: $\det V_n \neq 0$

Beweis:

Induktionsanfang (n=2): $\det \begin{bmatrix} 1 & x_0 \\ 1 & x_1 \end{bmatrix} = x_1 - x_0 \neq 0$

Induktionsvoraussetzung: $\det\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \neq 0,$ wegen

Vertauschbarkeit also auch

$$\det\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n+1} & x_{n+1}^2 & \cdots & x_{n+1}^n \end{bmatrix} \neq 0$$

Induktionsschritt:

Durch Subtraktion des x_0 –fachen der j-ten Spalte von der j+1-ten Spalte für j = n+1, n, …, 1 erhält man:

$$\det\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n+1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{n+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n+1} & x_{n+1}^2 & \cdots & x_{n+1}^{n+1} \end{bmatrix} =$$

$$\det\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & x_1 - x_0 & x_1^2 - x_0 x_1 & \cdots & x_1^{n+1} - x_0 x_1^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n+1} - x_0 & x_{n+1}^2 - x_0 x_{n+1} & \cdots & x_{n+1}^{n+1} - x_0 x_{n+1}^n \end{bmatrix}$$

Nun fallen bei Entwicklung nach der ersten Zeile alle Summanden weg, außer die 1,1-Streichungsmatrix:

$$= \det\begin{bmatrix} (x_1 - x_0) & (x_1 - x_0) \cdot x_1 & \cdots & (x_1 - x_0) * x_1^n \\ \vdots & \vdots & \vdots & \vdots \\ (x_{n+1} - x_0) & (x_{n+1} - x_0) \cdot x_{n+1} & \ldots & (x_{n+1} - x_0) * x_{n+1}^n \end{bmatrix}$$

$$= \underbrace{\prod_{k=1}^{n+1} (x_k - x_0)}_{\neq 0 \; nach \; Bedingung \; (2.0.1)} \cdot \underbrace{\det\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n+1} & x_{n+1}^2 & \cdots & x_{n+1}^n \end{bmatrix}}_{\neq 0 \; nach \; IV} \neq 0$$

Daraus folgt die Aussage für alle $n \in \mathbb{N}$. ■

Satz 3.1.2

Voraussetzung:

Sei $n \in \mathbb{N}$ und $n+1$ Punktepaare $(x_i, y_i) \in \mathbb{R}^2, i = 0, 1, \dots, n$ gegeben mit $x_i \neq x_k,\ i \neq k$

Behauptung:

Durch das Gleichungssystem $\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$ sind $a_i, i = 0, 1, \dots, n$ eindeutig bestimmt.

Beweis:

Nach Satz 3.1.1 ist $\det \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \neq 0$, also die Vandermondematrix invertierbar und es existiert eine Matrix $S \in GL_{n+1}$, sodass:

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix} \Leftrightarrow \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = S \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

■

Bemerkung 3.1.3

Das bedeutet es gibt genau ein Polynom $P \in \prod_n,\ P(x) = \sum_{k=0}^{n} a_k\, x^k$, welches Gleichung (2.0.1) erfüllt. Nun ist es interessant ein solches zu finden, was mithilfe der Lagrange-Interpolation durch eine zwar etwas aufwendige, aber dafür leicht verständliche Methode ermöglicht wird.

3.2 Lagrange-Interpolation

Definition 3.2.1: Lagrange-Basis

Seien $n+1$ Stellen $x_i \in \mathbb{R},\ i = 0,1, \dots, n,$ mit $x_i \neq x_k,\ i \neq k$ gegeben, dann ist die dazugehörige Lagrange-Basis: $L_k(x) \coloneqq \prod_{k \neq j=0}^{n} \frac{(x - x_j)}{(x_k - x_j)}, k = 0,1, \dots, n$

Bemerkung 3.2.2

Der Name Lagrange-Basis rührt daher, dass jede Lagrange-Basis $(L_0, L_1, \dots, L_n)$ tatsächlich eine Basis vom Vektorraum $\prod_n$ ist.

Beweis:

Offensichtlich sind die Vektoren der Lagrange-Basis auch Vektoren aus $\prod_n$, da ihr Grad jeweils n ist

Sei weiter eine Lagrange-Basis $(L_0, L_1, \dots, L_n)$ mit den zugehörigen Stellen $(x_0, x_1, \dots, x_n)$, sowie ein darzustellendes Polynom $P \in \prod_n$ gegeben. Nun gilt:

$$L_k(x_i) = \frac{(x_i - x_1) \cdots (x_i - x_i) \cdots (x_i - x_n)}{\prod_{k \neq j=1}^{n} (x_k - x_j)} = 0, \qquad wenn\ i \neq k \quad (3.2.3)$$

$$L_k(x_k) = \prod_{k \neq j=1}^{n} \frac{(x_k - x_j)}{(x_k - x_j)} = \prod_{k \neq j=1}^{n} 1 = 1 \quad (3.2.4)$$

Sei nun $L(x) \coloneqq \sum_{i=0}^{n} P(x_i) \cdot L_i(x)$ und $y_k := P(x_k)$, sodass für $k = 0,1, \dots, n$ gilt:

$$L(x_k) = \sum_{i=0}^{n} P(x_i) \cdot L_i(x_k) = P(x_i) \cdot 0 + \cdots + P(x_k) \cdot 1 + \cdots + P(x_n) \cdot 0 = P(x_k)$$

$$= y_k$$

$$für\ k = 0,1, \dots, n$$

Da es nach Bemerkung (3.1.3) genau ein Polynom mit Grad $\leq n$ gibt, welches die Bedingung (2.0.1) erfüllt, folgt daraus, dass $P = L$, womit jedes Polynom $P \in \prod_n$ eine Darstellung bezüglich $(L_0, L_1, \dots, L_n)$ besitzt. Weil aber die Lagrange-Basis aus $n + 1$ Vektoren besteht und $\prod_n$ (n+1)-dimensional ist, ist die Lagrange-Basis sogar eine Basis von $\prod_n$. ■

Konstruktion 3.2.5: Lagrange-Interpolation

Input: $(x_i, y_i) \in \mathbb{R}^2$, $i = 0, 1, \dots, n$ gegeben mit $x_i \neq x_k,\ i \neq k$

Output: $L(x) := \sum_{i=0}^{n} y_i \cdot L_i(x)$

Es gilt: $L(x_k) = \sum_{i=0}^{n} y_i \cdot L_i(x_k) y_1 \cdot 0 + \cdots + y_k \cdot 1 + \cdots + y_n \cdot 0 = y_k, für\ k = 0,1, \dots, n$ nach Gleichung (3.2.3) und (3.2.4) ■

4. Spline-Interpolation

Definition 4.0.1:

Es sei $l \in \mathbb{N}$ und $\Delta = \{x_0, x_1, \dots, x_n\}$ eine Zerlegung des Intervalls $[a, b]$, dann ist der Raum aller Splinefunktionen der l-ten Ordnung auf der Zerlegung Δ heißt

$$S_{\Delta,l} = \{s \in C^{l-1}[a,b] : s|_{[x_{k-1},x_k]} = P|_{[x_{k-1},x_k]}, P \in \prod_l, k = 1,2,\dots,n\}$$

Also ist eine Splinefunktion eine möglichst glatte Zusammensetzung von Polynomen. Die Spline-Interpolation bezeichnet das Interpolieren von Datenpaaren durch eine Splinefunktion. Sei $s \in S_{\Delta,l}$ so nennt man die Abschnittsweisen Polynome $s_k \coloneqq s|_{[x_k,x_{k+1}]},\ k = 0,1,..,(n-1)$

4.1 lineare Splines

Konstruktion 4.1.1: lineare Spline-Interpolation

Input: $(x_i, y_i) \in \mathbb{R}^2, i = 0, 1, \dots, n$ gegeben mit $x_i \neq x_k,\ i \neq k$

Da durch zwei Punkte eine Gerade schon prädestiniert ist hat man hier keinen Spielraum für zusätzliche Anforderungen. Diese abschnittsweisen Geraden sind offensichtlich

$$s_0(x) = y_0 + \frac{y_1 - y_0}{x_1 - x_0} \cdot (x - x_0)$$

$$\vdots$$

$$s_{n-1}(x) = y_{n-1} + \frac{y_n - y_{n-1}}{x_n - x_{n-1}} \cdot (x - x_{n-1})$$

Output: $s: [x_0, x_n] \to \mathbb{R}$

4.2 kubische Splines

Sehr verbreitet in der Anwendung sind kubische Splines, also Spline-Funktionen der Ordnung 3. Hier hat man bei der Interpolation einmal die Anforderungen, dass jeder Abschnitt von zwei Punkten umgeben ist, welche für ihn vorgesehen sind. Dann noch, dass die erste und zweite Ableitung bei den Verbindungsstellen übereinstimmen. Insgesamt macht das $2n + (n-1) + (n-1) = 4n - 2$ Gleichungen die die Zuordnungen mit insgesamt $4n$

Parametern erfüllen müssen (wenn man von n Teilintervallen ausgeht). Also dürften, ohne genauer darauf einzugehen, noch konkretere Vorschriften für kubische Splines gemacht werden. Je nach Belieben oder Datenlage sind folgende Anforderungen üblich:

- Natürliche Randbedingung: $s''(a) = s''(b) = 0$
- Vollständige Randbedingung: $s'(a) = f'_0,\ s'(b) = f'_n$ für vorgegebene f'_0, f'_n
- Periodische Randbedingung: $s'(a) = s'(b),\ s''(a) = s''(b)$ (4.2.1)

Aber zuerst zum Lösen der generellen Spline-Anforderungen. Seien hierzu die abschnittsweisen Funktionen von der Gestalt $s_k(x) = a_k + b_k(x - x_k) + c_k(x - x_k)^2 + d_k(x - x_k)^3, k = 0,1,..,(n-1)$ (4.2.2)

Lemma 4.2.3:

Voraussetzung:

Sei $\Delta = \{x_0, x_1, \dots, x_n\}$ eine Zerlegung zu Punktepaaren $(x_i, y_i) \in \mathbb{R}^2, i = 0, 1, \dots, n$ und $M_0, M_1, \dots, M_n \in \mathbb{R}$ erfüllen folgende Gleichungen (für $k = 1,2, \dots, (n-1)$):

$$h_{k-1}M_{k-1} + 2(h_{k-1} + h_k)M_k + h_kM_{k+1} = 6\frac{y_{k+1} - y_k}{h_k} - 6\frac{y_k - y_{k-1}}{h_{k-1}} \quad (4.2.4)$$

$$mit\ h_k \coloneqq x_{k+1} - x_k\ (auch\ für\ k = 0)$$

Behauptung:

Setzt man in Darstellung (4.2.2) $a_k = y_k,\ b_k = \frac{y_{k+1}-y_k}{h_k} - (M_{k+1} + 2M_k)\frac{h_k}{6},\ c_k = \frac{M_k}{2}, d_k = \frac{M_{k+1}-M_k}{6h_k}$ für $k = 0,1, \dots, (n-1)$ so erhält man einen interpolierenden kubischen Spline $s \in S_{\Delta,3}$.

Beweis:

Es gilt für $k = 0,1, \dots, (n-1)$:

a) $s_k(x_k) = a_k = y_k$

b) $s_k(x_{k+1}) = a_k + b_k h_k + c_k {h_k}^2 + d_k {h_k}^3 = y_k + \left(\frac{y_{k+1}-y_k}{h_k} - (M_{k+1} + 2M_k)\frac{h_k}{6}\right) h_k + \frac{M_k}{2}{h_k}^2 + \frac{M_{k+1}-M_k}{6h_k}{h_k}^3 = y_{k+1} + \frac{{h_k}^2}{6}(M_{k+1} - M_k - (M_{k+1} + 2M_k) + 3M_k) = y_k$

c) ${s'}_k(x_{k+1}) = b_k + 2c_k h_k + 3d_k {h_k}^2 = \frac{y_{k+1}-y_k}{h_k} - (M_{k+1} + 2M_k)\frac{h_k}{6} + 2\frac{M_k}{2}h_k + 3\frac{M_{k+1}-M_k}{6h_k}{h_k}^2 = \frac{y_{k+1}-y_k}{h_k} - (M_{k+1} + 2M_k)\frac{h_k}{6} + M_k h_k + \frac{M_{k+1}-M_k}{2}h_k = \frac{y_{k+1}-y_k}{h_k} + (2M_{k+1} + M_k)\frac{h_k}{6}$

d) ${s'}_{k+1}(x_{k+1}) = b_{k+1} = \frac{y_{k+2}-y_{k+1}}{h_{k+1}} - (M_{k+2} + 2M_{k+1})\frac{h_{k+1}}{6}$

e) ${s''}_{k+1}(x_{k+1}) = 2c_{k+1} = M_{k+1}$

f) ${s''}_k(x_{k+1}) = 2c_k + 6d_k h_k = M_k + M_{k+1} - M_k = M_{k+1}$

Nun folgt aus a) und b) bereits, dass s stetig ist und interpoliert. Weiter gilt durch verschieben des Index k um 1 nach links und anpassen der Indizes in Gleichung (4.2.4) sowie c) und d):

$$h_k M_k + 2(h_k + h_{k+1})M_{k+1} + h_{k+1}M_{k+2} = 6\frac{y_{k+2} - y_{k+1}}{h_{k+1}} - 6\frac{y_{k+1} - y_k}{h_k}$$

$$\Longrightarrow h_k M_k + 2h_k M_{k+1} + 6\frac{y_{k+1} - y_k}{h_k} = 6\frac{y_{k+2} - y_{k+1}}{h_{k+1}} - h_{k+1}M_{k+2} - 2h_{k+1}M_{k+1}$$

$$\Longrightarrow (M_k + 2M_{k+1})h_k + 6\frac{y_{k+1} - y_k}{h_k} = 6\frac{y_{k+2} - y_{k+1}}{h_{k+1}} - h_{k+1}(M_{k+2} + 2M_{k+1})$$

$$\Longrightarrow {s'}_k(x_{k+1}) = {s'}_{k+1}(x_{k+1})$$

Weiter folgt aus e) und f), dass s zweimal stetig differenzierbar ist. Damit ist $s \in S_{\Delta,3}$ ■

Bemerkung 4.2.5:

In 4.2.3 e) hat man gesehen, dass die „Momente“ $M_0, M_1, \dots, M_n$ mit der zweiten Ableitung der entsprechenden Stützstelle übereinstimmen.

Nun lässt sich das in Lemma 4.2.3 erwähnte Gleichungssystem auch in Matrizenform darstellen ($g_k := 6\frac{y_{k+1}-y_k}{h_k} - 6\frac{y_k - y_{k-1}}{h_{k-1}}$, $k = 1, \dots, n-1$):

$$\begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & & \\ & & \ddots & & \\ & & h_{n-2} & 2(h_{n-1}+h_{n-2}) & h_{n-1} \\ 0 & \cdots & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ M_{n-1} \\ M_n \end{bmatrix} = \begin{bmatrix} 0 \\ g_1 \\ \vdots \\ g_{n-1} \\ 0 \end{bmatrix}$$

Offensichtlich hat diese Gleichung wegen der zwei Nullzeilen eine mindestens zweidimensionale Lösungsmenge für $M_0, M_1, \dots, M_n$. Nun kann man je nach Problemstellung zusätzlich eine der drei Randbedingungen (4.2.1) fordern. Diese ergänzen das Gleichungssystem je um zwei Zeilen:

a) Natürliche Randbedingung: $s''(a) = s''(b) = 0$

Da $M_0 = s''(a)$ und $M_n = s''(b)$ folgt:

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & & \\ & & \ddots & & \\ & & h_{n-2} & 2(h_{n-1}+h_{n-2}) & h_{n-1} \\ 0 & \cdots & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ M_{n-1} \\ M_n \end{bmatrix} = \begin{bmatrix} 0 \\ g_1 \\ \vdots \\ g_{n-1} \\ 0 \end{bmatrix}$$

b) Vollständige Randbedingung: $s'(a) = f'_0,\ s'(b) = f'_n$ für vorgegebene f'_0, f'_n

Da $f'_0 = s'(a) = s'_0(a) = s'_0(x_0) = b_0 + 2c_0(x_0 - x_0) + 3d_0(x_0 - x_0)^2 = b_0$ mit $b_0 = \frac{y_1 - y_0}{h_0} - (M_1 + 2M_0)\frac{h_0}{6}$ nach Lemma (4.2.3) so folgt durch Gleichsetzen: $(M_1 + 2M_0)\frac{h_0}{6} = \frac{y_1 - y_0}{h_0} - f'_0 \Leftrightarrow 2h_0M_0 + h_0M_1 = 6\frac{y_1 - y_0}{h_0} - 6f'_0 =: g_0$

Weiter ist $f'_n = s'(b) = s'_{n-1}(b) = s'_{n-1}(x_n) = b_{n-1} + 2c_{n-1}(x_n - x_{n-1}) + 3d_{n-1}(x_n - x_{n-1})^2 = b_{n-1} + 2c_{n-1}h_{n-1} + 3d_{n-1}h_{n-1}^2 = \frac{y_n - y_{n-1}}{h_{n-1}} - (M_n + 2M_{n-1})\frac{h_{n-1}}{6} + 2\frac{M_{n-1}}{2}h_{n-1} + 3\frac{M_n - M_{n-1}}{6h_{n-1}}h_{n-1}^2$

$\Leftrightarrow f'_n - \frac{y_n - y_{n-1}}{h_{n-1}} = -\frac{h_{n-1}}{6}M_n - \frac{h_{n-1}}{3}M_{n-1} + h_{n-1}M_{n-1} + \frac{h_{n-1}}{2}M_n - \frac{h_{n-1}}{2}M_{n-1} \quad \Leftrightarrow f'_n - \frac{y_n - y_{n-1}}{h_{n-1}} = \frac{h_{n-1}}{3}M_n + \frac{h_{n-1}}{6}M_{n-1} \Leftrightarrow 6f'_n - 6\frac{y_n - y_{n-1}}{h_{n-1}} = 2h_{n-1}M_n + h_{n-1}M_{n-1}$ Ergänzt man das Gleichungssystem um diese beiden Gleichungen, so ergibt sich folgendes mit $g_n := 6f'_n - 6\frac{y_n - y_{n-1}}{h_{n-1}}$:

$$\begin{bmatrix} 2h_0 & h_0 & 0 & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & & \\ & & \ddots & & \\ & & h_{n-2} & 2(h_{n-1}+h_{n-2}) & h_{n-1} \\ 0 & \cdots & 0 & h_{n-1} & 2h_{n-1} \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ M_{n-1} \\ M_n \end{bmatrix} = \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{n-1} \\ g_n \end{bmatrix}$$

c) Periodische Randbedingung: $s'(a) = s'(b),\ s''(a) = s''(b)$

Nun gilt $\frac{y_1-y_0}{h_0} - (M_1 + 2M_0)\frac{h_0}{6} = b_0 = b_0 + 2c_0(x_0 - x_0) + 3d_0(x_0 - x_0)^2 = s_0'(x_0) = s_0'(a) = s'(a) = s'(b) = \frac{h_{n-1}}{3}M_n + \frac{h_{n-1}}{6}M_{n-1} + \frac{y_n-y_{n-1}}{h_{n-1}}$ nach b). Dies ist äquivalent zu: $g_0 := 6\frac{y_1-y_0}{h_0} - 6\frac{y_n-y_{n-1}}{h_{n-1}} = 2h_0M_0 + h_0M_1 + h_{n-1}M_{n-1} + 2h_{n-1}M_n$.

Für die zweite Bedingung gilt: $M_0 = s''(a) = s''(b) = M_n$, was man in die erste einsetzen kann. Nun resultiert folgendes Gleichungssystem:

$$g_0 = 2(h_0+h_{n-1})M_0 + h_0M_1 + h_{n-1}M_{n-1}$$
$$g_n := g_0 = 2(h_0+h_{n-1})M_n + h_0M_1 + h_{n-1}M_{n-1}$$

Hieraus folgt direkt, da $h_0 + h_{n-1} \neq 0$, dass $s''(0) = M_0 = M_n = s''(b)$ und damit auch die erste Bedingung:

$$\begin{bmatrix} 2(h_0+h_{n-1}) & h_0 & 0 & \cdots & 0 & h_{n-1} & 0 \\ h_0 & 2(h_0+h_1) & h_0 & & & & 0 \\ & & & \ddots & & & \\ 0 & & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & & h_0 & 0 & \cdots & 0 & h_{n-1} & 2(h_0+h_{n-1}) \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ M_{n-1} \\ M_n \end{bmatrix} = \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{n-1} \\ g_n \end{bmatrix}$$

Nun muss noch gezeigt werden, dass diese Gleichungssysteme jeweils nach den Momenten aufgelöst werden können. Hierzu verwendet man die eigenschaft, dass alle Matrizen, die die Gleichungssysteme beschreiben invertierbar sind.

Lemma 4.2.6

Vorraussetzung: Eine Matrix $A \in \mathbb{R}^{n\times n}$ eine strikt dominante Matrix, also $|a_{i,i}| > \sum_{i\neq j=1}^{n}|a_{i,j}|\ f\ddot{u}r\ i = 1,..,n$

Behauptung: A ist invertierbar

Beweis:

Hierzu sei angenommen es gäbe einen Vektor$x \in \mathbb{R}^n \backslash \{0\}$, der die Gleichung $Ax = 0$ erfüllt. Diesen normiert man bezüglich der Maximumsnorm:

$y \coloneqq \frac{x}{\|x\|_\infty}$, sodass es mindestens einen Eintrag mit $|y_i| = 1$ gibt. Für alle anderen gilt $|y_j| \leq 1, j \neq i$. Nun gilt für den i-ten Eintrag in dem Produkt $Ay =: v$:

$$|v_i| = \left|\sum_{j=1}^{n} a_{i,j} y_j\right| \geq \left||a_{i,i} y_i| - \sum_{i \neq j=1}^{n} |a_{i,j} y_j|\right| \geq \left||a_{i,i}| - \sum_{i \neq j=1}^{n} |a_{i,j}|\right| > 0$$

Dies steht im Widerspruch zu $v = Ay = A\frac{x}{\|x\|_\infty} = 0$. Also gilt $def(A) = 0 \Leftrightarrow rk(A) = n \Leftrightarrow A$ ist invertierbar ■

Bemerkung 4.2.7

Alle Matrizen aus Bemerkung 4.2.5 sind strikt dominant, da die Abstände h_k alle größer als null sind und sich deshalb nicht wegehaben. Nach Lemma 4.2.7 sind sie damit invertierbar, das heißt man kann die Gleichungen eindeutig nach den Momenten umformen. Das bedeutet wiederum für jeden Datensatz mit einer zusätzlichen Randbedingungen gibt es genau einen kubischen Spline der alle Anforderungen erfüllt.

5.Anwendung

5.1 Interpolation einer Sinuskurve

Hierzu sei die Zuordnung $f: [0,10] \longrightarrow [0,1], x \mapsto \sin(x)$ und die Punkte $(k, f(k)), k = 0,1, \ldots, 10$ gegeben.

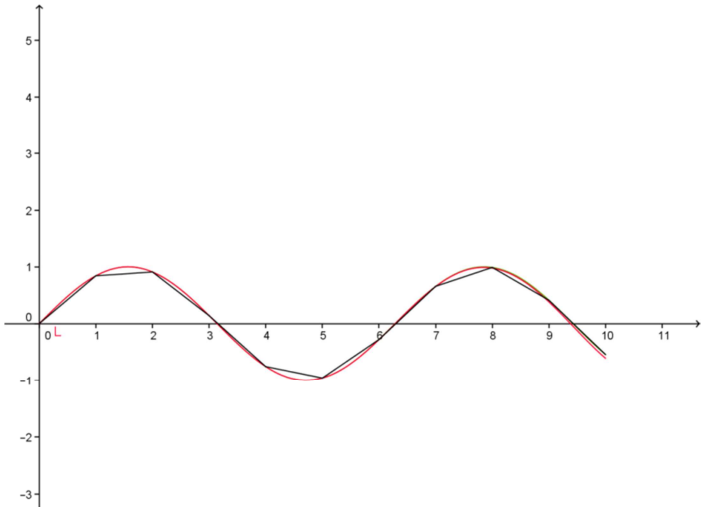
5.1.1 Interpolation durch lineare Splines: Nach Konstruktion 4.1.1 ist die interpolierende lineare Splinefunktion

$$s: [0,10] \longrightarrow \mathbb{R}, x \mapsto \begin{cases} \sin(0) + (\sin(1) - \sin(0)) \cdot (x - 0), & x \in [0,1] \\ \vdots & \\ \sin(9) + (\sin(10) - \sin(9)) \cdot (x - 9), & x \in [9,10] \end{cases}$$

5.1.2 Interpolation durch Lagrange-Polynom: Nach Konstruktion 3.2.5 ist $L(x) \coloneqq \sum_{i=0}^{10} y_i \cdot L_i(x)$ mit $L_i(x) = \prod_{i \neq j=1}^{10} \frac{(x - x_j)}{(x_i - x_j)}$, also $L(x) \coloneqq \sum_{i=0}^{10} \sin(i) \cdot \prod_{i \neq j=1}^{10} \frac{(x-j)}{(i-j)} \approx 1{,}73694 \cdot 10^{-7}\, x\text{^}10 - 8{,}21415 \cdot 10^{-6}\, x\text{^}9 + 1{,}52031 \cdot 10^{-4} x\text{^}8 - 1{,}34485 \cdot 10^{-3}\, x\text{^}7 + 5{,}41909 \cdot 10^{-3}\, x\text{^}6 - 8{,}34218 \cdot 10^{-3}\, x\text{^}5 + 3{,}31972 \cdot 10^{-2}\, x\text{^}4 - 0{,}207566\, x\text{^}3 + 2{,}80316 \cdot 10^{-2}\, x\text{^}2 + 0{,}991961\, x$

Die vielen Nachkommastellen sind zwar unschön, jedoch notwendig, da zum Beispiel die Potenz $x\text{^}10$ auch im Intervall $[0,10]$ schon sehr groß wird.

Vergleicht man f, s und L graphisch in Geogebra, so erkennt man, dass die Polynominterpolation hier klar besser ist (Graphen einzeln und Nahansicht im Anhang):



5.1.3 Fehlermessung

Nun gilt es noch rechnerisch nachzuweisen, dass die Lagrange-Interpolation hier besser funktioniert. Dazu kann man mit der 2-Metrik für stetige Graphen den Abstand innerhlab eines Intervalls $[a, b]$ zwischen der jeweiligen Näherungsfunktion und der Sinuskurve messen:

$$d_2(f, g) = \sqrt{\int_a^b (f(x) - g(x))^2 dx}$$

Hier bei entsteht aber ein Produkt eines Polynoms $P(x) = \sum_{k=0}^{n} a_k x^k$ und einer trigonometrischen Funktion. Mithilfe partieller Integration kann man dies aber lösen:

$$\int \mathrm{t}(x)\, P(x) dx = \int \mathrm{t}(x) \sum_{k=0}^{n} a_k x^k \, dx = \mathrm{T}(x) \sum_{k=0}^{n} a_k x^k - \int \mathrm{T}(x) \sum_{k=0}^{n-1} (k+1) \cdot a_{k+1} x^k$$

Mit einer trigonometrischen Funktion $t = sin, cos, -sin$ oder $-cos$ und ihrer Stammfunktion T. Durch mehrfache Anwendung folgt:

$$\int \mathrm{t}_m(x) \sum_{k=0}^{n} a_k x^k \, dx = \sum_{j=0}^{n} t_{m-j-1}(x)(-1)^j \sum_{k=j}^{n} \frac{k!}{(k-j)!} \cdot a_k x^{k-j} + c \qquad (5.1.4)$$

$$mit\ t_0 = sin, t_1 = cos, t_2 = -sin, t_3 = -\cos, sonst\ t_k = t_{(k\ mod\ 4)}$$

Für die Integration von $sin^2(x)$ schreibt man den Term in analytischer Form: $\int sin^2(x)dx = \int (\frac{e^{ix}-e^{-ix}}{2i})^2\, dx = \int \frac{e^{i2x}-2+e^{-i2x}}{-4}\, dx = \int \frac{1-\cos(2x)}{2}\, dx = \frac{1}{2}x - \frac{1}{4}\sin(2x) + c$ (5.1.5)

$$d_2(f,s) = \sqrt{\int_0^{10} \big(f(x)-s(x)\big)^2 dx} = \sqrt{\sum_{k=0}^{9} \int_k^{k+1} \big(f(x)-s_k(x)\big)^2 dx}$$

$$\sqrt{\sum_{k=0}^{9} \int_k^{k+1} \big(sin(x)-(a_{1_k}x + a_{0_k})\big)^2 dx}$$

$$mit\ a_{1_k} \coloneqq \sin(k+1)-\sin(k)\ \ und\ a_{0_k} \coloneqq -\mathrm{k}\cdot\sin(k+1)+(1+k)\sin(k)$$

$$= \sqrt{\sum_{k=0}^{9} \int_k^{k+1} sin^2(x) - 2sin(x)(a_{1_k}x + a_{0_k}) + (a_{1_k}x + a_{0_k})^2 dx}$$

$$\sqrt{\sum_{k=0}^{9} \left[\frac{1}{2}x - \frac{1}{4}\sin(2x) - 2(-\cos(x)\left(a_{1_k}x + a_{0_k}\right) + a_{1_k}\sin(x)) + \frac{{a_{1_k}}^2}{3}x^3 + a_{0_k}a_{1_k}x^2 + {a_{0_k}}^2 x\right]}$$

Iteriert man nun über $\mathrm{k} = 1,..9$ (Python-Quellcode im Anhang) so erhält man näherungsweise folgendes Ergebnis:

$$d_2(f,s) \approx \sqrt{0{,}0377} \approx 0{,}1943$$

Für den Abstand zum Lagrange-Polynom gilt:

$$d_2(f,s) = \sqrt{\int_0^{10} \big(f(x)-L(x)\big)^2 dx} = \sqrt{\int_0^{10} \left(\sin(\mathrm{x}) - \sum_{k=0}^{10} a_k x^k\right)^2 dx}$$

$$= \sqrt{\int_0^{10} \sin^2(\mathrm{x}) - 2\sin(\mathrm{x})\sum_{k=0}^{10} a_k x^k + \left(\sum_{k=0}^{10} a_k x^k\right)^2 dx}$$

Dies könnte man nun auch mithilfe von Gleichung (5.1.4) und (5.1.5) ausrechnen, jedoch würde dies nun zu umständlich werden. Stattdessen erfolgt die Berechnung mithilfe des Moduls „scipy“ von Python (Quellcode siehe Anhang). Dies liefert das Ergebnis

$$d_2(f,s) \approx 0{,}05401$$

Hiermit wurde nachgewiesen, dass in diesem Fall die Polynominterpolation besser funktioniert.

D(f,L)=0,053999

5.2 Modellierung eines Autos

In einem Szenario der Industriespionage ist folgendes vorstellbar: ein Spion hat folgendes Foto eines Erlkönigs gemacht:

Quelle: Autobild

Nun will die konkurrierende Firma ein Profil des Umrisses machen. Hierzu kann man die kubische Spline-Interpolation nutzen: Mithilfe von GIMP werden charakteristische Punkte des oberen Umrisses entnommen:

Zusätzlich wird die Anforderung der natürlichen Randbedingung gestellt. Es wird die Lösung des Gleichungssystems aus Bemerkung 4.2.7 durchgeführt:

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
10 & 42 & 11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 11 & 36 & 7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 7 & 84 & 35 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 35 & 146 & 38 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 38 & 162 & 43 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 43 & 158 & 36 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 36 & 164 & 46 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 46 & 146 & 27 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 27 & 110 & 28 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 28 & 96 & 20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 20 & 100 & 30 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 30 & 102 & 21 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 21 & 84 & 21 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 21 & 84 & 21 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 21 & 90 & 24 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 24 & 114 & 33 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 33 & 138 & 36 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 36 & 134 & 31 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 31 & 114 & 26 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 26 & 104 & 26 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 26 & 98 & 23 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 23 & 90 & 22 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 22 & 90 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix} M_0 \\ M_1 \\ M_2 \\ M_3 \\ M_4 \\ M_5 \\ M_6 \\ M_7 \\ M_8 \\ M_9 \\ M_{10} \\ M_{11} \\ M_{12} \\ M_{13} \\ M_{14} \\ M_{15} \\ M_{16} \\ M_{17} \\ M_{18} \\ M_{19} \\ M_{20} \\ M_{21} \\ M_{22} \\ M_{23} \\ M_{24} \end{bmatrix}
=
\begin{bmatrix} 0 \\ 0{,}76363 \\ 0{,}77922 \\ -10{,}97143 \\ 0{,}93383 \\ 1{,}12729 \\ -0{,}73256 \\ -0{,}45652 \\ -1{,}26570 \\ -0{,}20635 \\ -1{,}67143 \\ -0{,}5 \\ -0{,}82857 \\ -0{,}85714 \\ 1{,}71429 \\ 2{,}82143 \\ -0{,}43182 \\ -0{,}31818 \\ -0{,}08065 \\ -0{,}80397 \\ 0{,}46154 \\ -0{,}64214 \\ -0{,}88932 \\ -1{,}19763 \\ 0 \end{bmatrix}$$

Nun ergibt sich durch den Pythonquellcode (siehe Anhang) folgendes Ergebnis:

$$\begin{bmatrix} M_0 \\ M_1 \\ M_2 \\ M_3 \\ M_4 \\ M_5 \\ M_6 \\ M_7 \\ M_8 \\ M_9 \\ M_{10} \\ M_{11} \\ M_{12} \\ M_{13} \\ M_{14} \\ M_{15} \\ M_{16} \\ M_{17} \\ M_{18} \\ M_{19} \\ M_{20} \\ M_{21} \\ M_{22} \\ M_{23} \\ M_{24} \end{bmatrix}
\approx
\begin{bmatrix} 0 \\ 0{,}00514 \\ 0{,}04981 \\ -0{,}15294 \\ 0{,}04362 \\ -0{,}00214 \\ -0{,}00427 \\ 0{,}00095 \\ -0{,}00997 \\ 0{,}00543 \\ -0{,}01910 \\ 0{,}00051 \\ -0{,}00564 \\ -0{,}01280 \\ 0{,}01604 \\ 0{,}03029 \\ -0{,}01007 \\ -0{,}00034 \\ 0{,}00168 \\ -0{,}00951 \\ 0{,}00874 \\ -0{,}00770 \\ -0{,}00495 \\ -0{,}01210 \\ 0 \end{bmatrix}$$

Nach Lemma 4.2.3 erhält man durch $a_k = y_k,\ b_k = \frac{y_{k+1}-y_k}{h_k} - (M_{k+1} + 2M_k)\frac{h_k}{6}$, $c_k = \frac{M_k}{2}, d_k = \frac{M_{k+1}-M_k}{6h_k}, k = 1, \dots, n-1$ die Splinefunktion (Berechnung und Ausgabe in Bild durch das Pythonprogramm im Anhang):

Ein angemessenes Resultat für diesen Aufwand.

5.3 weitere Anwendungen

Kubische Splines können überall angewandt werden, wo es um die Glättung von Datenpaaren geht. Wenn zum Beispiel ein Objekt mit einzelnen Datenpaaren erfasst wird, so kann man durch diese mit kubischer Splineinterpolation eine Kurve legen. Dies beschränkt sich nicht nur auf zweidimensionale Gebilde auch in der dritten Dimension können durch ähnliche Verfahren auch zusammengesetzte Abbildungen ein Objekt modellieren.

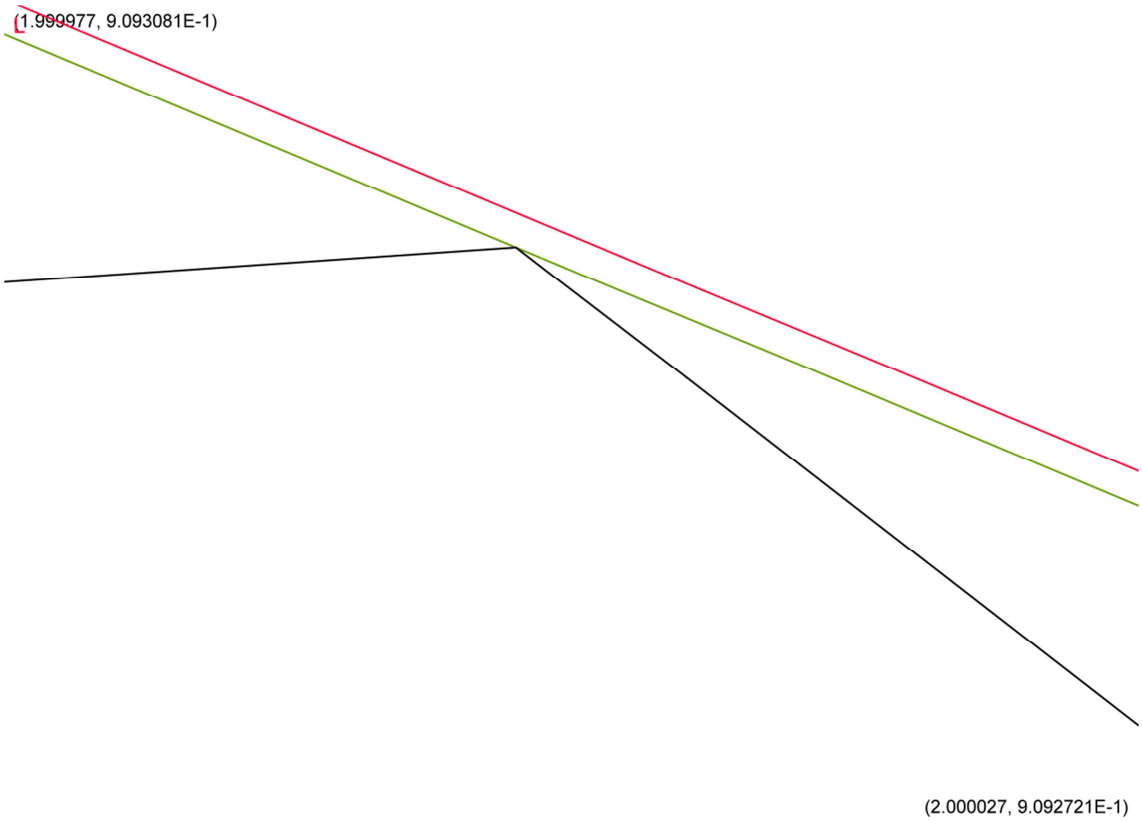
Als praktisches Beispiel kann man etwa das digitale Abspeichern von Kunstobjekten zur Erhaltung dieser nennen.

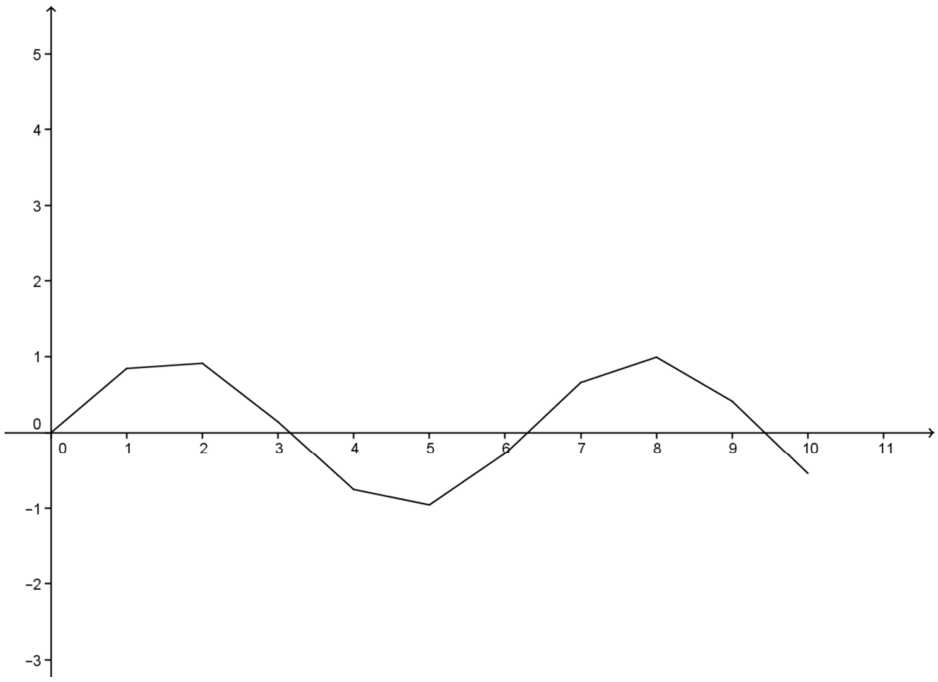
Das praktische an kubischen Splines ist, dass sie eine sehr geringe Krümmung haben, was versichert, dass sie den „optimalen“ Weg durch die Datenpaare finden und nicht dazwischen hin-und-her schwanken. Deshalb ist die Glättung ein guter Anwendungsbereich.

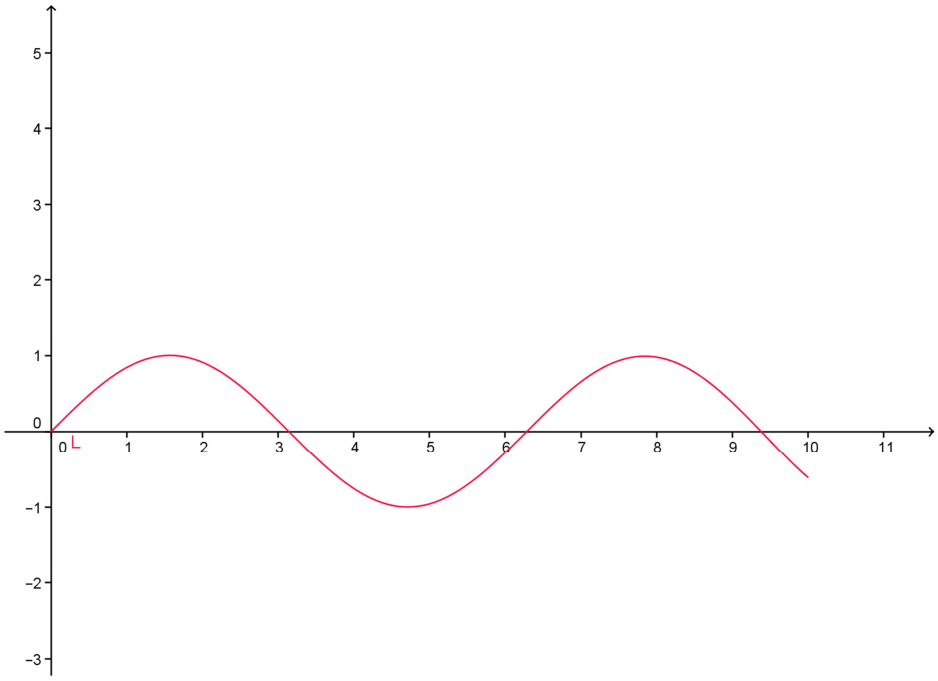
Jedoch hat die Splineinterpolation in dieser Form auch Grenzen. Denn oft sind auch Punkte gegeben, die nicht durch eine Funktion dargestellt werden können, weil einem x mehrere y zugeordnet werden können.

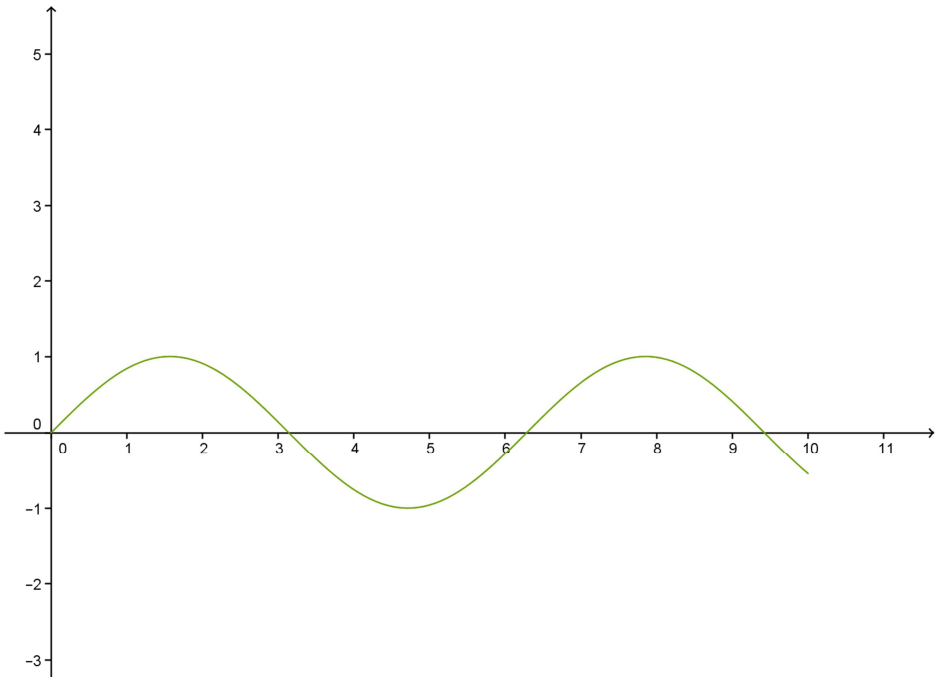
Anhang:

```
import math as m

def sin(x):
    return(m.sin(x))

def cos(x):
    return(m.cos(x))

def klammern (k,x):
    p=sin(k+1)-sin(k)
    q=-k*sin(k+1)+(1+k)*sin(k)
    a = 0.5*x-0.25*sin(2*x)+2*cos(x)*(p*x+q)-2*p*sin(x)+p**2/3*x**3+p*q*x**2+q**2*x
    return(a)

a=0
for k in range(0,10):
    a=a+klammern(k,k+1)-klammern(k,k)
    print(a)

print (m.sqrt(a))
```




5
4
3
2
1
0
−1
−2
−3
0 1 2 3 4 5 6 7 8 9 10 11


5
4
3
2
1
0
−1
−2
−3
0 1 2 3 4 5 6 7 8 9 10 11



Python-Datei: siehe Zusatzmaterial

Literatur:

Günter Bärwolff: Numerik 1, aktueller Stand (27. Januar 2012)

Erklärung der eigenständigen Anfertigung der Facharbeit:

Ich, Max Sauerbrey, versichere, dass ich die vorliegende Facharbeit selbständig verfasst und keine außer den angegebenen Hilfsmitteln verwendet habe. Alle Textstellen, die dem Wortlaut nach anderen Texten entsprechen, wurden entsprechend gekennzeichnet. Dies gilt ebenso für Bilder und Zeichnungen. Mir ist bewusst, dass wahrheitswidrige Angaben als Täuschungsversuch behandelt werden.

_____ _____