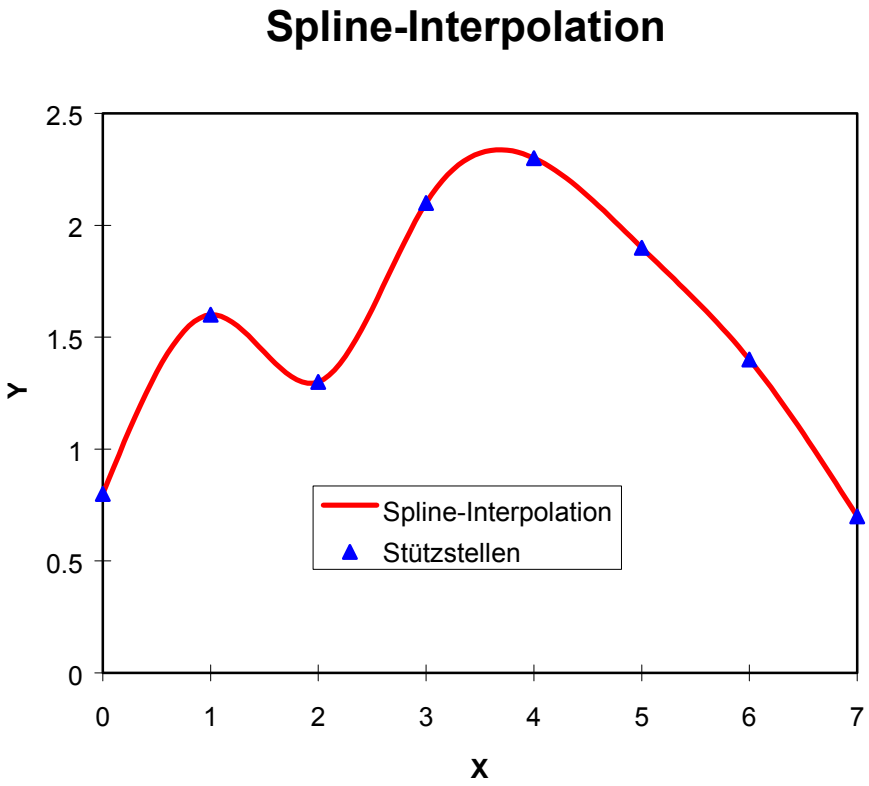


Theorie. Häufig benötigen wir als Ingenieure Kurven, die *exakt* durch einen Satz vorgegebener Punkte verlaufen. Die Aufgabe, eine solche Funktion zu bestimmen, heißt *Interpolationsaufgabe*. Sind insgesamt n+1 verschiedene Stützstellen vorgegeben, so läßt sich die Interpolationsaufgabe beispielsweise dadurch lösen, daß man ein Polynom der Ordnung n bestimmt, das exakt durch die vorgegebenen Punkte verläuft. Es existiert genau ein solches Polynom. Solche global definierten, "einheitlichen" Polynome höherer Ordnung tendieren allerdings zu *Oszillationen*, ganz besonders dann, wenn - was in der Praxis häufig der Fall ist - die Stützstellen äquidistant sind, und wenn man viele Stützstellen berücksichtigen will. Der Grad des Interpolationspolynoms ist dann starr an die Anzahl von Stützstellen gebunden.

Um solche Probleme zu vermeiden, wurde die Methode der *Spline-Interpolation* entwickelt. Mit dieser Methode wird durch vorgegebene Stützstellen eine Kurve gelegt, die möglichst geringe Krümmung besitzt und möglichst hohen Stetigkeitsanforderungen genügt. Die Kurve wird jedoch aus einzelnen polynomialen Funktionen *stückweise* bzw. *abschnittsweise* zusammengesetzt.

Zur Herleitung wollen wir folgende Vorbetrachtung anstellen: Als Ingenieure wissen wir, daß ein Durchlaufträger, dem vorgegebene Auflagerverschiebungen aufgezwungen werden, sich so verformt, daß die gesamte Biegeenergie des Balkens minimal wird. "Oszillationen" treten nicht auf, weil diese mit hoher Biegeenergie verbunden wären. Die Biegelinie eines elastischen Durchlaufträgers stellt also gerade eine Kurve dar, die die gewünschten Interpolationseigenschaften für die gegebenen Auflagerverschiebungen aufweist. Untersucht man einen Biegeträger nach Theorie I. Ordnung, und wird der Träger ausschließlich durch aufgezwungene Auflagerverschiebungen belastet, so stellt sich als Biegelinie in jedem Feld des Trägers ein kubisches Polynom ein. Die einzelnen kubischen Stücke sind an den Nahtstellen so zusammengefügt, daß die Momente dort stetig sind. Das heißt, eine solche Biegelinie stellt eine stückweise kubische, an den Nahtstellen bis zur zweiten Ableitung stetig zusammengefügte Kurve dar.

Dies ist genau die Idee, die der *Spline*-Interpolation zugrunde liegt. Die Idee stammt aus dem Schiffsbau, wo man die Form der Beplankung von Schiffen mit Hilfe eines biegsamen Lineals (engl. *spline*) zu ermitteln pflegte.

Ein kubisches Polynom wird durch 4 Koeffizienten beschrieben. Je "Feld" des Durchlaufträgers bzw. je Interpolationsintervall haben wir also 4 Unbekannte zu bestimmen, um die *Spline*-Interpolation 3. Grades zu berechnen, insgesamt bei n+1 Stützstellen also 4n Unbekannte. Zur Bestimmung dieser Unbekannten stehen uns *2n* Interpolationsbedingungen (je eine am Anfang und Ende jedes Feldes) und 2(n-1) Stetigkeitbedingungen (Stetigkeit der 1. und 2. Ableitung an allen Innenpunkten) zur Verfügung. Es fehlen somit noch 2 Bedingungen. Es liegt nahe, aus der Analogie zum Durchlauf-Biegeträger zu schließen, daß wir die fehlenden Bedingungen dadurch ergänzen können, daß wir an beiden Enden des Durchlaufträgers verschwindende Momente bzw. Krümmung bzw. zweite Ableitungen fordern. Splines, die mit dieser Regel konstruiert werden, heißen *natürliche Splines 3. Ordnung*.

Gegeben seien n+1 paarweise verschiedene, nach aufsteigendem x sortierte Stützstellen

(x_i, y_i), i=0, 1, 2, ..., n.

Sei $s_i(x)$ das Interpolationspolynom im i-ten Abschnitt mit

$$s_i(x) = a_i\left(x - x_i\right)^3 + b_i\left(x - x_i\right)^2 + c_i\left(x - x_i\right) + d_i .$$

Die Länge dieses Abschnitts sei bezeichnet mit $h_i = x_{i+1} - x_i$. Dann können wir folgende Beziehungen im i-ten Abschnitt anschreiben:

$$s_i\left(x_i\right) = d_i = y_i$$
$$s_i\left(x_{i+1}\right) = a_i h_i^3 + b_h h_i^2 + c_i h_i + d_i = y_{i+1}$$
$$s_i'\left(x_i\right) = c_i$$
$$s_i'\left(x_{i+1}\right) = 3a_i h_i^2 + 2b_i h_i + c_i$$
$$s_i''\left(x_i\right) = 2b_i = \kappa_i$$
$$s_i''\left(x_{i+1}\right) = 6a_i h_i + 2b_i = \kappa_{i+1}$$

Drücken wir die unbekannten Koeffizienten a_i, b_i, c_i und d_i durch die (vorgegebenen) Funktionswerte y_i und y_{i+1} sowie durch die (unbekannten) Krümmumgen κ_i und κ_{i+1} aus, so erhalten wir aus der vorletzten Gleichung

$$b_i = \frac{\kappa_i}{2},$$

damit aus der letzten Gleichung

$$a_i = \frac{\kappa_{i+1} - \kappa_i}{6h_i},$$

und damit schließlich aus der zweiten vorstehenden Gleichung

$$c_i = \frac{y_{i+1} - y_i}{h_i} - \frac{h_i}{6}\left(2\kappa_i + \kappa_{i+1}\right)$$

Aus der ersten Gleichung erhalten wir direkt d_i. Schreiben wir nunmehr das kubische Polynom auch noch für den Abschnitt i-1 an, so finden wir (durch Anpassung der Indices):

$$s'_{i-1}(x_i) = 3\frac{\kappa_i - \kappa_{i-1}}{6h_{i-1}}h_{i-1} + 2\frac{\kappa_{i-1}}{2}h_{i-1} + \frac{y_i - y_{i-1}}{h_{i-1}} - \frac{h_{i-1}}{6}\left(2\kappa_{i-1} + \kappa_i\right)$$

$$= \frac{2\kappa_i + \kappa_{i-1}}{6}h_{i-1} + \frac{y_i - y_{i-1}}{h_{i-1}}$$

Nun können wir die Forderung nach Stetigkeit der 1. Ableitung an der Stelle x_i anschreiben als

$$\frac{2\kappa_i + \kappa_{i-1}}{6}h_{i-1} + \frac{y_i - y_{i-1}}{h_{i-1}} \stackrel{!}{=} -\frac{2\kappa_i + \kappa_{i+1}}{6}h_i + \frac{y_{i+1} - y_i}{h_i}$$

Damit finden wir durch Umsortieren nach bekannten und unbekannten Größen:

$$\boxed{\kappa_{i-1}h_{i-1} + \kappa_i\, 2\left(h_i + h_{i-1}\right) + h_i\kappa_{i+1} = \frac{6}{h_i}\left(y_{i+1} - y_i\right) - \frac{6}{h_{i-1}}\left(y_i - y_{i-1}\right)}$$

Diese Gleichung können wir für alle Punkte i=1 bis i=n-1 anschreiben. Es entsteht somit ein Gleichungssystem, das nur noch die zweiten Ableitungen an den Innenpunkten $x_1, x_2, \ldots, x_n$ enthält (n-1 Gleichungen für n-1 unbekannte Krümmungen an den Innenpunkten). Für κ_0 und κ_n setzen wir, wie oben erläutert, den Wert 0 ein. Das Gleichungssystem ist tridiagonal. Darin spricht sich die Tatsache aus, daß Änderungen des Funktionswertes an einer einzelnen Stützstelle bei der Spline-Interpolation nur *lokale* Auswirkungen haben. Diese Eigenschaft der (ungeraden) Splines trägt dazu bei, Oszillationen des Interpolationspolynoms auch bei fehlerbehafteten Datenwerten zu vermeiden. Interpoliert man hingegen eine Reihe von Meßpunkten mit einem "durchgehenden" Polynom höherer Ordnung (also nicht mit *stückweisen* Polynomen niedriger Ordnung), so erhält man eine sehr oszillationsanfällige Interpolation, ganz besonders bei äquidistanten Stützstellen.

Für den häufigen Spezialfall äquidistanter Stützstellen (also $h_o = h_1 = \ldots = h_n = h$) hat das Spline-Gleichungssystem folgende spezielle Gestalt:

$$\begin{bmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & \\ & 1 & 4 & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 1 & 4 \end{bmatrix} \begin{bmatrix} \kappa_1 \\ \kappa_2 \\ \kappa_3 \\ \vdots \\ \kappa_{n-1} \end{bmatrix} = \begin{bmatrix} (y_0 - 2y_1 + y_2)\frac{6}{h^2} \\ (y_1 - 2y_2 + y_3)\frac{6}{h^2} \\ \\ \\ (y_{n-2} - 2y_{n-1} + y_n)\frac{6}{h^2} \end{bmatrix}$$

Das Gleichungssystem, das im Zuge der Spline-Interpolation zu lösen ist, ist positiv definit und *tridiagonal*, im Falle gleicher Intervalle sogar symmetrisch. Somit kann es einem Aufwand gelöst werden, der nur *linear* mit der Anzahl der Stützstellen wächst, während der Aufwand zum Lösen eines allgemeinen, vollbesetzten Gleichungssystems mit n^3 wächst und bei *polynomialer Interpolation* mit Polynomen der Ordnung n ein Aufwand der Ordnung n^2 erforderlich wird. Auch der Speicherplatzaufwand der Spline-Interpolation wächst nur *linear* mit n. Da desweiteren ja auch die Interpolationseigenschaften des Splines besonders günstig sind, ist Spline-Interpolation heute die bevorzugte Methode zur Konstruktion glatter Interpolationskurven.

Um das tridiagonale Gleichungssystem zu lösen, verfährt man wie folgt: Mit Hilfe der 1. Gleichung kann man κ_1, also die Krümmung an der ersten Binnenstützstelle des Intervalls,

durch κ_2 ausdrücken. Diesen Ausdruck setzt man dann in die zweite Gleichung ein. Mit dieser Gleichung, die dann nur noch κ_2 und κ_3 enthält, kann man somit κ_2 durch κ_3 ausdrücken. Führt man diesen Vorgang der "Vorwärtssubstitution" bis zur letzten Zeile fort, so kann man in der letzten Zeile κ_{n-1} direkt ausrechnen.

Die übrigen Unbekannten erhält man dann durch "Rückwärtssubstitution". Unter Beachtung der speziellen Struktur des tridiagonalen Gleichungssystems, das bei Anwendung der SPLINE-Interpolation entsteht, erhalten wir zusammengefaßt folgenden Algorithmus zur Bestimmung der unbekannten Krümmungen κ_i :

Lies *n* und x_i und y_i für alle *i* von 0 bis *n* ein
Wiederhole für *i*=0 bis *n*-1:

$$h_i = x_{i+1} - x_i$$

$$e_i = \frac{6}{h_i}(y_{i+1} - y_i)$$

Vorwärtseinsetzen:

$$u_1 = 2(h_0 + h_1)$$

$$r_1 = e_1 - e_0$$

Wiederhole für *i*=2 bis *n*-1:

$$u_i = 2(h_i + h_{i-1}) - \frac{h_{i-1}^2}{u_{i-1}}$$

$$r_i = (e_i - e_{i-1}) - \frac{r_{i-1} h_{i-1}}{u_{i-1}}$$

Rückwärtseinsetzen:

$$\kappa_n = 0$$

Wiederhole für *i*=*n*-1 bis 1 in Schritten von -1:

$$\kappa_i = \frac{r_i - h_i \kappa_{i+1}}{u_i}$$

$$\kappa_0 = 0$$

Die erste Schleife im vorstehenden Algorithmus ermittelt die Längen der Teilintervalle und die Komponenten e_i der rechten Seite. Die rechte Seite der *i*-ten Gleichung lautet mit dieser Abkürzung $e_i - e_{i-1}$. Die zweite Schleife baut das tridiagonale Gleichungssystem (*n*-1 unbekannte Krümmungen) auf und führt gleichzeitig das Vorwärtseinsetzen durch. Da das Gleichungssystem tridiagonal und symmetrisch ist und wir seine sehr spezielle Struktur kennen, reichen einige Vektoren der Dimension *n*-1, um es zu speichern. Außerdem benötigen wir einen Vektor für die rechte Seite. Der Vektor u_i enthält die Diagonale nach der Vorwärtselimination, der Vektor r_i die umgeformte rechte Seite (mit den Beiträgen, die bei Elimination von κ_{i-1} entstehen). Zur Speicherung der Ergebnisse - der Krümmungen an den *n*+1 Stützstellen - benötigen wir außerdem den Vektor κ.

Eine Frage, die wir bisher nicht behandelt haben, ist die *Auswertung* der Spline-Interpolation an einer beliebigen Stelle $x_0 \le x \le x_n$. Wir waren davon ausgegangen, daß $x_0 \le x_1 \le ... \le x_n$. Um den Spline auswerten zu können, müssen wir zunächst feststellen, im wievielten Teilintervall *i* sich der Punkt *x* befindet. Die einfachste Lösung dieser Aufgabe besteht natürlich darin, das Feld der Stützstellen der Reihe nach abzusuchen, bis man ein Intervall findet, in dem $x_i < x \le x_{i+1}$ gilt. Der Nachteil dieser Methode ist aber, daß man im Mittel die

Hälfte der Elemente des Feldes durchsuchen muß, um das gesuchte Intervall aufzufinden. Der Suchaufwand wächst also proportional zur Größe *n* des zu durchsuchenden Feldes.

Daher werden wir das Problem des Auffindens des richtigen Intervalls mit einer Variante des schon von der Nullstellensuche bekannten *Bisektionsalgorithmus* (*binäre Suche*) lösen. Dazu ist es erforderlich, daß die n+1 Stützstellen sortiert nach aufsteigendem x in einem *Feld* gespeichert sind, auf dessen Inhalte wir per *Index direkt* zugreifen können. Der Algorithmus arbeitet völlig analog zur Bisektion bei der Nullstellensuche:

Wir untersuchen zunächst das mittlere Element dieses Feldes. Befindet sich das gegebene x links von dem Mittelelement, so brauchen wir nur noch die linken Teilintervalle zu testen. Ist x größer als das Element aus der Mitte des Feldes, so suchen wir in der rechten Feldhälfte weiter. Wir wiederholen die Suche mit derselben Strategie solange, bis wir das gesuchte Intervall gefunden haben. Mit k Vergleichen können wir auf diese Weise ein Feld mit 2^k Einträgen durchsuchen, oder, anders ausgedrückt, in einem Feld mit n Einträgen sind $\log_2 n$ Vergleichsoperationen notwendig, um das gesuchte Intervall zu identifizieren. Für große n ist das ein wesentlich geringerer Aufwand als der, der beim sequentiellen Suchen entsteht.

Implementierung. Zunächst sind die n+1 Stützstellen und Funktionswerte einzulesen; dann sind die (n-1) unbekannten Krümmungen an den Innenpunkten zu bestimmen; schließlich ist das Spline-Interpolationspolynom an einer beliebigen Stelle x auszuwerten.

Die Stützstellen x, die Funktionswerte y, aber auch die gesuchten Krümmungen κ lassen sich zu (n+1)-dimensionalen Vektoren zusammenfassen. Weitere Vektoren benötigen wir zur Speicherung der Tridiagonalmatrix und deren rechter Seite.

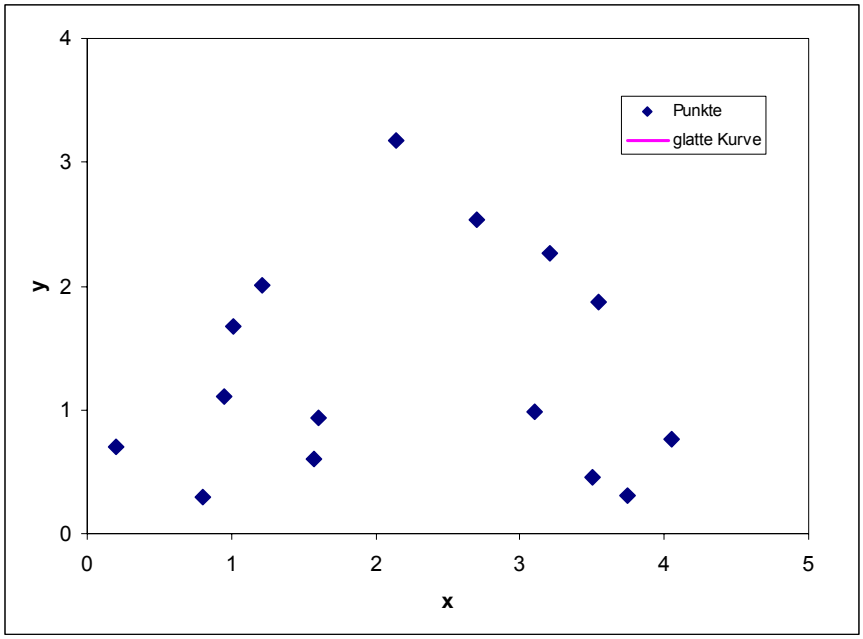
Spline-Interpolation ist das Mittel der Wahl, wo auch immer im praktischen Ingenieurwesen in Wertetabellen zu interpolieren ist. Das Verfahren zeichnet sich durch Robustheit (keine Oszillationen, lokale Auswirkung von Änderungen an der Wertetabelle) ebenso aus wie durch Effizienz (linear mit der Anzahl der Stützstellen wachsender Aufwand an Speicherplatz und Rechenzeit zur Bestimmung der Koeffizienten). Mit durchgehenden Interpolationspolynomen höherer Ordnung sind solche Ergebnisse nicht erzielbar!

Anwendungsbeispiel glatte Kurve durch gegebene Punkte. Wir wollen die Spline-Interpolation nun benützen, um eine kleine Aufgabe aus der rechnergestützten Geometrie zu lösen. Gegeben seien n+1 Punkte in der Ebene, und zwar durch die zugehörigen Wertepaare der Koordinaten $\begin{pmatrix} x_i \\ y_i \end{pmatrix}$, i=0,1,2,...,n. Es soll nun eine glatte Kurve konstruiert werden, die durch alle diese Punkte verläuft. Z.B. könnte eine Straße gesucht sein, deren Trasse durch die gegebenen Punkte als Zwangspunkte verlaufen soll:

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
x_i	0.2	0.8	1.57	1.6	0.95	1.01	1.21	2.14	2.7	3.21	3.54	3.1	3.5	3.74	4.05
y_i	0.7	0.3	0.6	0.93	1.11	1.67	2.01	3.17	2.54	2.26	1.87	0.98	0.45	0.31	0.76

Um dieses Problem zu lösen, betrachten wir die gesuchte Kurve als parametrisch definierte Kurve, d.h. wir setzen $\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}$. Der Anfangspunkt $\begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$ unserer Kurve soll t=0

entsprechen, und für $t = t_i > 0$ sollen sich die vorgegebenen Zwangspunkte ergeben, also $\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{pmatrix} x(t_i) \\ y(t_i) \end{pmatrix}$.



Punkte, die zu einer glatten Kurve verbunden werden sollen

Damit haben wir das Problem umgewandelt in die Aufgabe, die beiden unbekannten *Funktionen* $x(t)$ und $y(t)$ anhand der gegebenen Stützstellen t_i, x_i sowie t_i, y_i zu ermitteln. Irgendeine sinnvolle Parametrisierung t können wir noch frei wählen. Zunächst muß der Parameter nur die Bedingung erfüllen, daß $t_{i+1} > t_i$, weil sonst $x(t)$ und $y(t)$ keine Funktionen sind.

Die einfachste Wahl wäre, $t_{i+1} = i$ zu nehmen! Nun ist unser Problem vollständig definiert, und wir können $x(t)$ und $y(t)$ durch Spline-Interpolation ermitteln. Wir haben also die beiden folgenden Interpolationsaufgaben zu lösen:

1) Bestimme eine Spline-Interpolation für $x(t)$ aus

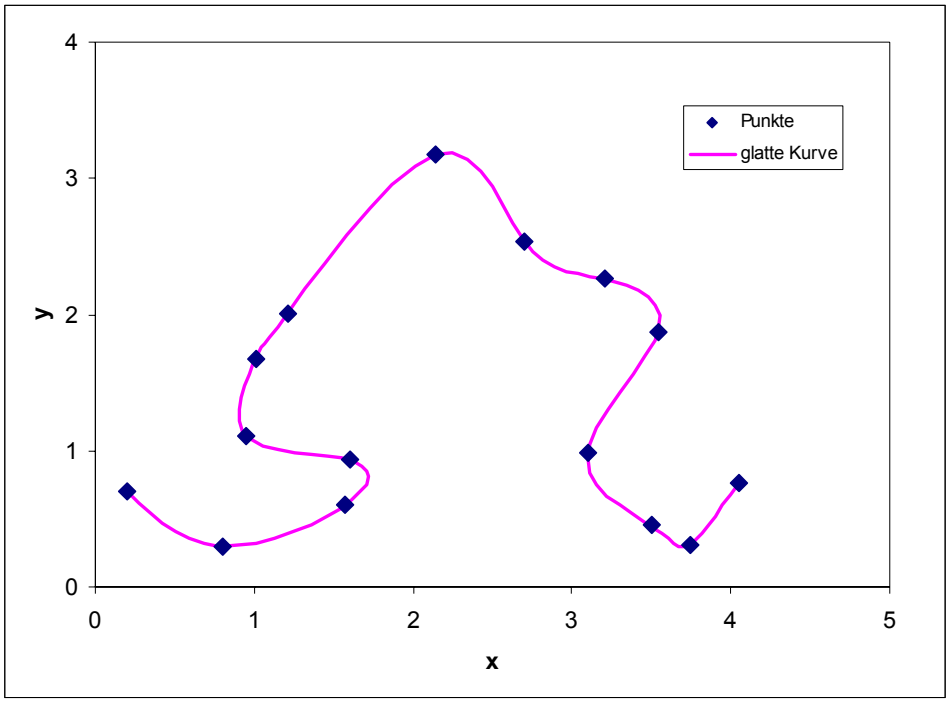
t_i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
x_i	0.2	0.8	1.57	1.6	0.95	1.01	1.21	2.14	2.7	3.21	3.54	3.1	3.5	3.74	4.05

2) Bestimme eine Spline-Interpolation für $y(t)$ aus:

t_i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
y_i	0.7	0.3	0.6	0.93	1.11	1.67	2.01	3.17	2.54	2.26	1.87	0.98	0.45	0.31	0.76

Sobald wir diese beiden Spline-Interpolationen berechnet haben, können wir mit Hilfe der ersten für beliebiges $0 \le t \le t_n$ die zugehörige Koordinate $x(t)$ und mit Hilfe der zweiten die Koordinate $y(t)$ bestimmen.

Das Ergebnis sieht aus wie folgt:



Kurve, die mit dem Parameter $t_i = i$ entsteht.

Vielleicht sind wir mit dem Ergebnis schon zufrieden. Allerdings mag es den einen oder anderen doch stören, daß der Parameter t nun in den unterschiedlichen Abschnitten unserer Kurve unterschiedlich schnell läuft, da das Inkrement von einem Zwangspunkt zum nächsten immer 1 ist, die durchlaufene Kurvenlänge aber stark unterschiedlich sein kann. Daraus resultiert ein etwas "eckiges" Aussehen unserer Kurve.

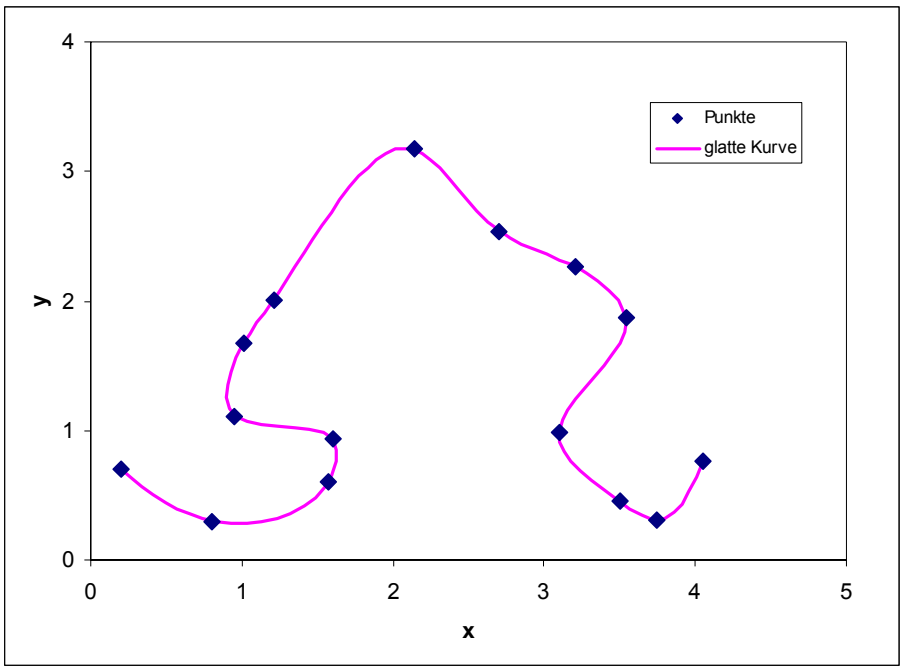
Das Problem läßt sich vermeiden, wenn wir die Spline-Interpolation auf Basis einer Parametrisierung durchführen, die näher an der "natürlichen Parametrisierung" der Kurve liegt, nämlich an der Bogenlänge. Natürlich kennen wir a priori die Bogenlänge noch nicht, die sich zum Schluß ergeben wird. Wir können die Bogenlänge aber ganz gut abschätzen durch die Länge eines Polygonzugs, der die Zwangspunkte verbindet.

Damit haben wir $t_i = \sum_{j=1}^{i} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}$.

In unserem Beispiel:

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
t_i	0.00	0.72	1.39	1.78	2.22	2.81	3.50	4.75	5.59	6.18	6.69	7.68	8.34	8.62	9.17
x_i	0.2	0.8	1.57	1.6	0.95	1.01	1.21	2.14	2.7	3.21	3.54	3.1	3.5	3.74	4.05
y_i	0.7	0.3	0.6	0.93	1.11	1.67	2.01	3.17	2.54	2.26	1.87	0.98	0.45	0.31	0.76

Die Kurvenlänge vom Start- zum Endpunkt beträgt in unserem Beispiel also mindestens 9.17, die Länge des zugehörigen Sehnenzuges. Erneute Interpolation mit Splines für $x(t)$ und $y(t)$ liefert nun folgendes Ergebnis:



Kurve, die mit dam Parameter "Länge des Sehnenpolygonzuges" entsteht.

Man sieht, daß diese Kurve noch etwas "glatter" aussieht. Man könnte nun durch Integration der tatsächlichen Kurvenlänge wieder neue, verbesserte Schätzungen für die t_i ermitteln und so schrittweise zu einer immer besseren Parametrisierung der Kurve gelangen.

Die hier dargestellte Technik zur Konstruktion glatter Kurven kann sehr gut zur Speicherung auch von recht komplizierten Linienzügen mit relativ wenigen zugehörigen Daten eingesetzt werden. Man verwendet sie u.a. in der Computergraphik.