

# **Mathematische Aspekte von fig2lat**

Dirk Krause

23. September 2017

# Inhaltsverzeichnis

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Gedrehte Ellipsen</b>   | <b>3</b>  |
| 1.1      | Abstand eines Punktes zu einer Geraden durch den Nullpunkt . . . . . | 3         |
| 1.2      | Bounding-Box für gedrehte Ellipsen . . . . .                         | 4         |
| <b>2</b> | <b>Pfeilspitzen</b>  | <b>6</b>  |
| 2.1      | Notwendigkeit der Längenkorrektur . . . . .                          | 6         |
| 2.2      | Längenkorrektur für Pfeilspitzen an Linien . . . . .                 | 8         |
| <b>3</b> | <b>X-Splines und Bezier-Splines</b>                                  | <b>9</b>  |
| 3.1      | Kurven . . . . .   | 9         |
| 3.2      | Kubische Bezier-Splines . . . . .                                    | 9         |
| 3.2.1    | Formel . . . . .   | 9         |
| 3.2.2    | Bounding-Box von Bezier-Splines . . . . .                            | 11        |
| 3.3      | X-Splines . . . . .  | 12        |
| 3.4      | Annäherung von X-Splines durch Bezier-Splines . . . . .              | 16        |
| 3.5      | Längenberechnung an Splines . . . . .                                | 17        |
| 3.6      | Längenkorrektur für Pfeilspitzen an Splines . . . . .                | 17        |
| <b>4</b> | <b>Kreisbögen</b>  | <b>18</b> |
| 4.1      | Berechnung von Mittelpunkt und Radius . . . . .                      | 18        |
| 4.2      | Längenkorrektur für Pfeilspitzen an Kreisbögen . . . . .             | 21        |
| 4.3      | Annäherung von Kreisbögen durch Bezier-Splines . . . . .             | 22        |

# 1 Gedrehte Ellipsen

## 1.1 Abstand eines Punktes zu einer Geraden durch den Nullpunkt

Für Berechnungen an gedrehten Ellipsen wird eine Formel für den Abstand  $l$  eines Punktes  $P(x_P; y_P)$  zu einer Geraden durch den Koordinatenursprung mit dem Winkel  $\beta$  benötigt.

Für die Gerade gilt:

$$\tilde{k}: \vec{r}(t) = \begin{pmatrix} t \cos \beta \\ t \sin \beta \end{pmatrix}$$

Für den Abstand zwischen  $P$  und einem durch  $t$  gegebenen Geradenpunkt gilt:

$$l(t) = \sqrt{(t \cos \beta - x_P)^2 + (t \sin \beta - y_P)^2}$$

Der Abstand des Punktes zur Geraden ist gleich der Länge des Lotes vom Punkt auf die Gerade. Da das Lot rechtwinklig zur Gerade steht, muss das Skalarprodukt von Lotvektor und Geradenvektor 0 ergeben.

$$\begin{aligned} 0 &= (\vec{r}(t_L) - \vec{r}_P) \cdot \vec{r}(t_L) \\ &= \begin{pmatrix} t_L \cos \beta - x_P \\ t_L \sin \beta - y_P \end{pmatrix} \cdot \begin{pmatrix} t_L \cos \beta \\ t_L \sin \beta \end{pmatrix} \\ &= t_L^2 \cos^2 \beta - x_P t_L \cos \beta + t_L^2 \sin^2 \beta - y_P t_L \sin \beta \\ &= t_L^2 - t_L (x_P \cos \beta + y_P \sin \beta) \\ &= t_L (t_L - (x_P \cos \beta + y_P \sin \beta)) \end{aligned}$$

Die Lösung

$$t_L = x_P \cos \beta + y_P \sin \beta$$

ergibt beim Einsetzen die Länge des Lotes:

$$l = \sqrt{(y_P - (y_P \sin \beta + x_P \cos \beta) \sin \beta)^2 + (x_P - (y_P \sin \beta + x_P \cos \beta) \cos \beta)^2}$$

Die Scheinlösung  $t_L = 0$  ergibt zwar ebenfalls ein Skalarprodukt mit dem Wert 0 aber nicht wegen eines rechten Winkels zwischen den Vektoren sondern weil ein Vektor die Länge 0 hat.

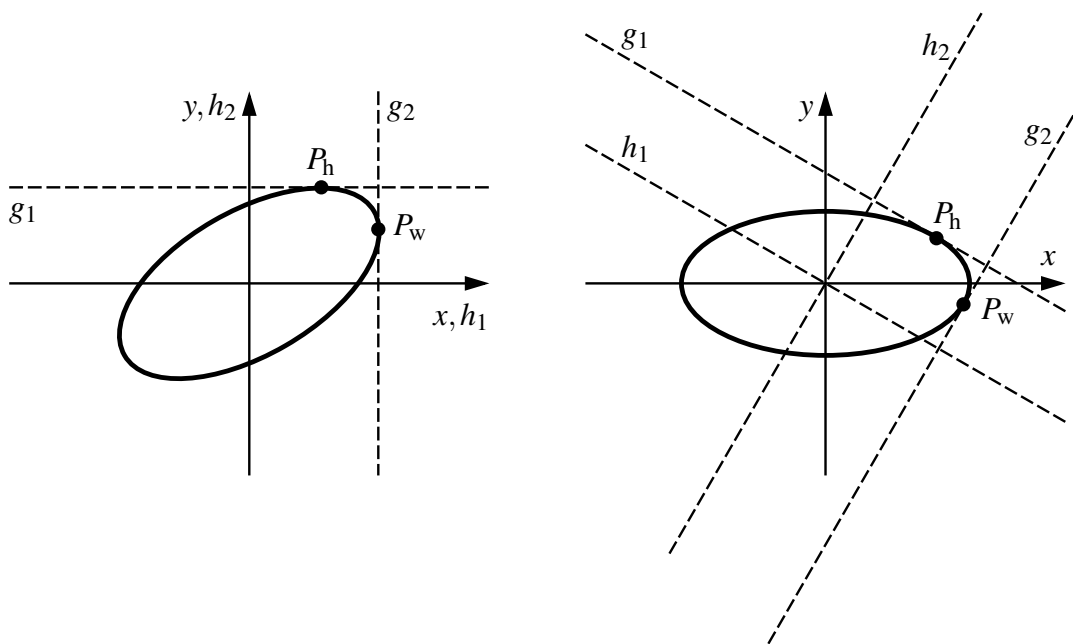


Abbildung 1: Berechnungen an der gedrehten Ellipse

## 1.2 Bounding-Box für gedrehte Ellipsen

Um die Bounding-Box einer um  $\alpha$  gedrehten Ellipse zu bestimmen, wird die Höhe  $h$  des höchsten Punktes und die Weite  $w$  des am weitesten rechts befindlichen Punktes benötigt.

In der linken Darstellung ist  $g_1$  die obere horizontale Tangente an die gedrehte Ellipse,  $g_2$  die rechte vertikale Tangente. Die Höhe  $h$  ergibt sich als Abstand des Berührungspunktes  $P_h$  zur  $x$ -Achse, die Weite  $w$  ist der Abstand des Berührungspunktes  $P_w$  zur  $y$ -Achse.

Für die Berechnung wird die gesamte Anordnung im Koordinatenursprung um  $-\alpha$  zurückgedreht, die Ellipse ist wieder an den Achsen ausgerichtet. Die Geraden  $g_1$  und  $g_2$  sind jetzt nicht mehr parallel zu den Achsen, die Hilfsgeraden  $h_1$  und  $h_2$  werden aus der Drehung der Achsen gewonnen. Die Höhe  $h$  ist nun der Abstand des Punktes  $P_h$  zur Hilfsgerade  $h_1$  (ehemals  $x$ -Achse), die Weite  $w$  der Abstand des Punktes  $P_w$  zur Hilfsgerade  $h_2$  (ehemals  $y$ -Achse).

Die Ellipsenkurve kann beschrieben werden durch:

$$\tilde{k}: \vec{r}(t) = \begin{pmatrix} r_x \cos t \\ r_y \sin t \end{pmatrix} \quad 0 \leq t \leq 2\pi$$

Für die Ableitungen nach  $t$  ergibt sich

$$\dot{\vec{r}}(t) = \frac{d\vec{r}}{dt} = \begin{pmatrix} -r_x \sin t \\ r_y \cos t \end{pmatrix}$$

Im Berührungspunkt  $P_h$  gilt

$$\tan(\pi - \alpha) = -\frac{r_y \cos t_h}{r_x \sin t_h}$$

Damit ergibt sich  $t_h$  zu:

$$t_h = \arctan -\frac{r_y}{r_x \tan(\pi - \alpha)}$$

Analog erhält man für  $P_w$

$$\tan\left(\frac{\pi}{2} - \alpha\right) = -\frac{r_y \cos t_w}{r_x \sin t_w}$$

$$t_w = \arctan -\frac{r_y}{r_x \tan\left(\frac{\pi}{2} - \alpha\right)}$$

Mit  $t_h$  und  $t_w$  können die Punkte  $P_h$  und  $P_w$  berechnet werden. Dann ergibt sich  $h$  als Abstand von  $P_h$  zur Hilfsgeraden  $h_1$  mit  $\beta_1 = -\alpha$  und  $w$  als Abstand von  $P_w$  zur Hilfsgeraden  $h_2$  mit  $\beta_2 = \frac{\pi}{2} - \alpha$ .

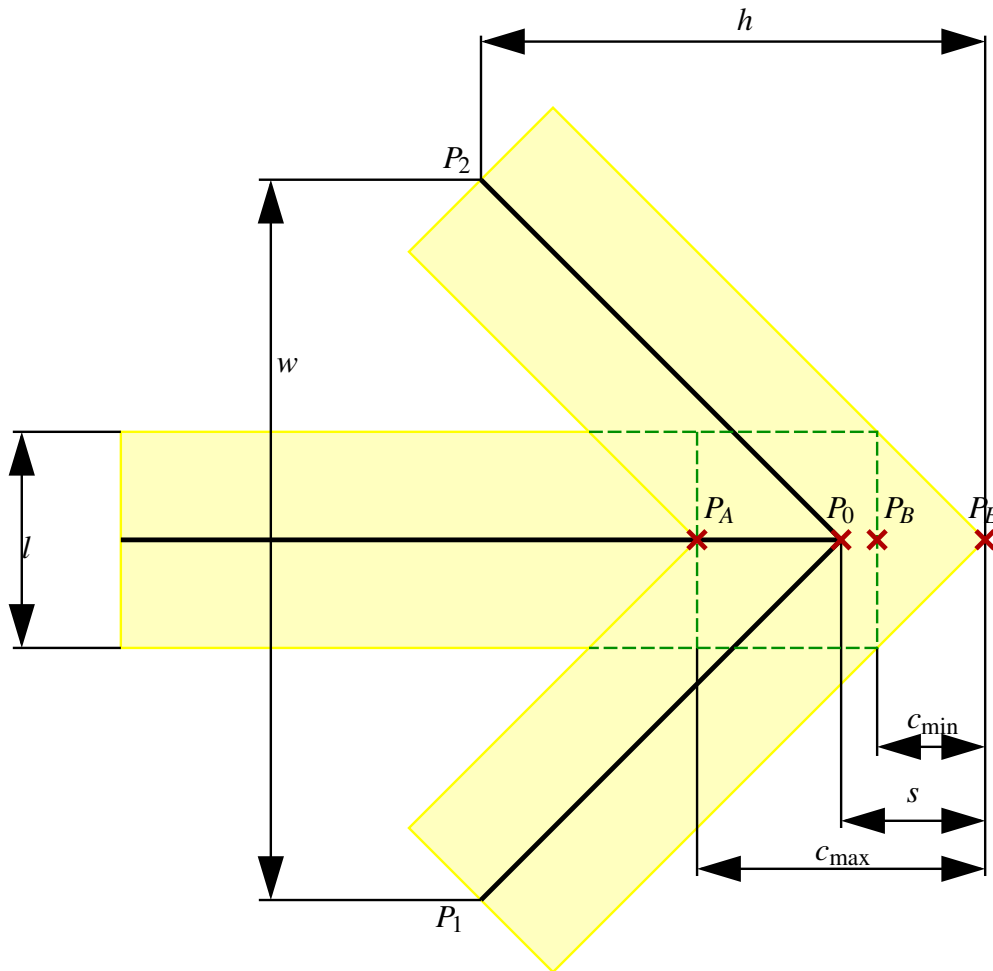


Abbildung 2: Längenkorrektur für Pfeilspitzen

## 2 Pfeilspitzen

### 2.1 Notwendigkeit der Längenkorrektur

Für Linien mit Pfeilspitzen gibt der in der Fig-Datei gegebene Endpunkt den Punkt an, in dem die Pfeilspitze endet. Die entsprechende Linie muss dann etwas kürzer gezeichnet werden.

Abb. 2 zeigt eine von links kommende Linie, die in Punkt  $P_0$  endet. An diese Linie wird eine Pfeilspitze angefügt, die als Polylinie  $P_1 \dots P_0 \dots P_2$  gezeichnet wird. Zum Zeichnen der Pfeilspitze wird dieselbe Linienbreite  $l$  verwendet wie für die eigentliche Linie. Wie man sieht, endet die Pfeilspitze im Punkt  $P_E$ , der um die Länge  $s$  von  $P_0$  entfernt ist.

Bezeichnet man den Öffnungswinkel der Pfeilspitze (Winkel zwischen den zwei Schenkeln der Pfeilspitze) mit  $\alpha$ , so erhält man:

$$\frac{l}{2} = s \cdot \sin\left(\frac{\alpha}{2}\right)$$

Sind Endpunkt  $P_E$ , Linienbreite  $l$  und Öffnungswinkel  $\alpha$  vorgegeben, so muss der Punkt  $P_0$  für das Zeichnen der Pfeilspitze den Abstand

$$s = \frac{\frac{l}{2}}{\sin\left(\frac{\alpha}{2}\right)} = \frac{l}{w} \sqrt{h^2 + \frac{1}{4}w^2}$$

zu  $P_E$  haben. Die Punkte  $P_1$  und  $P_2$  können dann über die Pfeilspitzenlänge und -breite berechnet werden.

Die eigentliche Linie – die noch vor der Pfeilspitze gezeichnet werden sollte – muss zwischen den Punkten  $P_A$  und  $P_B$  bzw. auf einem dieser Punkte enden. Die Entfernung von  $P_A$  zu  $P_E$  beträgt  $2s$ , die Entfernung von  $P_B$  zu  $P_E$  ergibt sich als  $c_{\min}$  zu:

$$\frac{\frac{l}{2}}{c_{\min}} = \tan\left(\frac{\alpha}{2}\right) = \frac{w}{2h}$$

$$c_{\min} = \frac{\frac{l}{2}}{\tan\left(\frac{\alpha}{2}\right)} = \frac{l \cdot h}{w}$$

$$c_{\min} \leq s$$

$$c_{\max} = 2s$$

$$c = \frac{1}{2}(c_{\max} + c_{\min})$$

$$\Delta c = \frac{1}{2}(c_{\max} - c_{\min})$$

## 2.2 Längenkorrektur für Pfeilspitzen an Linien

Die Längenkorrektur um die Länge  $c$  an einer Linie von  $P_0(x_0, y_0)$  nach  $P_1(x_1, y_1)$  wird mit Hilfe folgender Formeln durchgeführt:

$$l = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2} \qquad q = \frac{l - c}{l}$$

für nichtnegative  $q$  gilt dann:

$$x'_1 = x_0 + q(x_1 - x_0)$$

$$y'_1 = y_0 + q(y_1 - y_0)$$



## 3 X-Splines und Bezier-Splines

### 3.1 Kurven

Kurven in  $\mathbb{R}^2$  können durch die Gleichung

$$\tilde{k}: \quad \vec{r}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} \quad t_s \leq t \leq t_e$$

beschrieben werden.

Im Bereich der Computergraphik wird dabei häufig  $t \in [0; 1]$  für die gesamte Kurve oder für ein Kurvensegment verwendet.

Ein Spline ist eine Beschreibung für eine Kurve, dabei wird eine kontinuierliche Kurve durch einen Satz diskreter Werte bestimmt.

### 3.2 Kubische Bezier-Splines

#### 3.2.1 Formel

Für kubische Bezier-Splines wird jedes Segment durch Startpunkt  $P_s(x_s, y_s)$  und Endpunkt  $P_e(x_e, y_e)$  sowie zwei Kontrollpunkte  $P_{cs}(x_{cs}, y_{cs})$  und  $P_{ce}(x_{ce}, y_{ce})$  beschrieben. Dabei bezeichnet  $P_{cs}$  den Kontrollpunkt „rechts“ vom Startpunkt.  $P_{ce}$  bezeichnet den Kontrollpunkt „links“ vom Endpunkt.

In der Literatur wird häufig auch die Schreibweise  $P_i$  und  $P_j$  für Start- und Endpunkt sowie  $P_{i+}$  bzw.  $P_{j-}$  oder auch  $P_i^+$  bzw.  $P_j^-$  für die Kontrollpunkte verwendet.

$$0 \leq t \leq 1$$

$$\vec{r}(t) = (1-t)^3 \cdot \vec{r}_s + 3 \cdot (1-t)^2 \cdot t \cdot \vec{r}_{cs} + 3 \cdot (1-t) \cdot t^2 \cdot \vec{r}_{ce} + t^3 \cdot \vec{r}_e$$

$$\vec{r}(t) = (1-t)^3 \cdot \vec{0P}_s + 3 \cdot (1-t)^2 \cdot t \cdot \vec{0P}_{cs} + 3 \cdot (1-t) \cdot t^2 \cdot \vec{0P}_{ce} + t^3 \cdot \vec{0P}_e$$

$$\vec{r}' = \frac{d}{dt} \vec{r}(t) = (-3 + 6t - 3t^2) \vec{r}_s + 3(1 - 4t + 3t^2) \vec{r}_{cs} + 3(2t - 3t^2) \vec{r}_{ce} + 3t^2 \vec{r}_e$$

An den Randpunkten  $t = 0$  und  $t = 1$  gilt:

$$\vec{r}'(0) = -3 \vec{r}_s + 3 \vec{r}_{cs}$$

$$\vec{r}'(1) = -3 \vec{r}_{ce} + 3 \vec{r}_e$$

Dies kann umgestellt werden, so dass bei bekannten Anstiegen in Start- und Endpunkt die Kontrollpunkte berechnet werden können:

$$\vec{r}_{cs} = \vec{r}_s + \frac{1}{3} \vec{r}'(0)$$

$$\vec{r}_{ce} = \vec{r}_e - \frac{1}{3} \vec{r}'(1)$$

Ein Nachteil der Bezier-Splines ist, dass es für unerfahrene Nutzer schwierig ist, abzuschätzen, wie sich eine Manipulation der Kontrollpunkte auf den Kurvenverlauf auswirkt.

### 3.2.2 Bounding-Box von Bezier-Splines

Um die Bounding-Box eines Bezier-Spline-Segmentes zu bestimmen, müssen neben den Endpunkten möglicherweise auch Extremstellen der  $x$ - und  $y$ -Werte berücksichtigt werden.

Für die Berechnung der  $x$ -Koordinate in Abhängigkeit von  $t$  gilt:

$$\begin{aligned}x(t) &= (1-t)^3 \cdot x_s + 3 \cdot (1-t)^2 \cdot t \cdot x_{cs} + 3 \cdot (1-t) \cdot t^2 \cdot x_{ce} + t^3 \cdot x_e \\ &= a \cdot t^3 + b \cdot t^2 + c \cdot t + d \\ a &= x_e - 3x_{ce} + 3x_{cs} - x_s \\ b &= 3x_{ce} - 6x_{cs} + 3x_s \\ c &= 3x_{cs} - 3x_s \\ d &= x_s\end{aligned}$$

Für Extremstellen  $t_e$  gilt:

$$0 = \left. \frac{dx}{dt} \right|_{t=t_e}$$

Für echte Polynome dritten Grades mit  $a \neq 0$  ergeben sich die Lösungen für die Extremstellen:

$$\begin{aligned}t_{e1,2} &= \frac{-b \pm \sqrt{b^2 - 3ac}}{3a} \\ &= \frac{x_s - 2x_{cs} + x_{ce} \pm \sqrt{(x_e - x_{ce})x_s - x_{cs}x_e + x_{cs}^2 - x_{ce}x_{cs} + x_{ce}^2}}{x_s - 3x_{cs} + 3x_{ce} - x_e}\end{aligned}$$

Liegt mit  $a = 0$  ein Polynom zweiten Grades vor, gibt es nur eine Extremstelle. Dies kann z. B. in symmetrischen Anordnungen vorkommen.

$$\begin{aligned}t_e &= -\frac{c}{2b} \\ &= \frac{x_s - x_{cs}}{2x_s - 4x_{cs} + 2x_{ce}}\end{aligned}$$

Ist auch  $b = 0$ , liegt eine Geradengleichung vor, in diesem Fall genügt die Betrachtung der Endpunkte des Spline-Segmentes.

Liegen Extremstellen im Intervall  $0 < t_e < 1$ , müssen die entsprechenden  $x$ -Werte in der Bounding-Box-Berechnung berücksichtigt werden.

Die hier vorgenommenen Betrachtungen gelten in analoger Weise für die  $y$ -Koordinaten.

### 3.3 X-Splines

Cross-splines (X-Splines) wurden 1995 durch Schlick und Blanc vorgestellt. X-Splines werden hier nicht im Detail erklärt, hierzu sei auf die entsprechenden SIGGRAPH-Veröffentlichungen verwiesen, siehe [CB95]. Die Darstellung an dieser Stelle beschränkt sich auf die Verwendung von X-Splines in FIG-Dateien und die Behandlung im Programm fig2lat.<sup>1</sup>

Eine Funktion  $f(u)$  mit dem Parameter  $p$  wird als Gewichtsfunktion für die Approximation verwendet:

$$f(u, p) = \begin{cases} u^3 (10 - p + (2p - 15)u + (6 - p)u^2) & \text{für } u \in [0; 1] \\ 0 & \text{ansonsten} \end{cases}$$

$$= \begin{cases} (6 - p)u^5 + (2p - 15)u^4 + (10 - p)u^3 & \text{für } u \in [0; 1] \\ 0 & \text{ansonsten} \end{cases}$$

Zwei Teilfunktionen mit den Parametern  $p$  und  $q$  werden verwendet, um die Gewichtsfunktion  $e(u)$  für die Interpolation zu bilden:

$$e(u, p, q) = \begin{cases} qu + 2qu^2 + (10 - 12q - p)u^3 \\ \quad + (2p + 14q - 15)u^4 + (6 - 5q - p)u^5 & \text{für } u \in [0; 1] \\ qu + 2qu^2 - 2qu^4 - qu^5 & \text{für } u \in [-1; 0) \\ 0 & \text{ansonsten} \end{cases}$$

Die Ableitung der Funktionen ergibt:

$$\frac{\partial f}{\partial u} = \begin{cases} 5(6 - p)u^4 + 4(2p - 15)u^3 + 3(10 - p)u^2 & \text{für } u \in [0; 1] \\ 0 & \text{ansonsten} \end{cases}$$

$$\frac{\partial e}{\partial u} = \begin{cases} 5(6 - 5q - p)u^4 + 4(2p + 14q - 15)u^3 \\ \quad + 3(10 - 12q - p)u^2 + 4qu + q & \text{für } u \in [0; 1] \\ q + 4qu - 8qu^3 - 5qu^4 & \text{für } u \in [-1; 0) \\ 0 & \text{ansonsten} \end{cases}$$

<sup>1</sup>In [CB95] lautet Gleichung (20) für die Interpolationsfunktion:  $h(u) = qu + 2qu^2 - 2qu^4 + qu^5$ . Dies ist meiner Meinung nach fehlerhaft und muss stattdessen  $h(u) = qu + 2qu^2 - 2qu^4 - qu^5$  lauten, wie es auch in XFig und fig2dev implementiert ist. Die Version mit + würde auch die als Gleichungssystem (19) aufgestellten Forderungen nicht erfüllen, insbesondere die letzte nicht.

Bei der Verwendung von X-Splines in FIG-Dateien wird davon ausgegangen, dass äquidistante Kontrollpunkte vorliegen, fig2lat setzt hierfür  $\Delta t = 1$ .

Ein X-Spline wird durch  $n$  Kontrollpunkte mit Indizes im Bereich von 0 bis  $n - 1$  beschrieben, dabei ist jedem Kontrollpunkt  $P_k$  am Zeitpunkt  $t = T_k$  ein zusätzlicher Parameter  $s_k$  zugeordnet. Es gilt  $-1 \leq s_k \leq 1$ .

Punkte mit  $-1 \leq s_k < 0$  sind dabei Interpolationspunkte, Punkte mit  $0 \leq s_k \leq 1$  sind Approximationspunkte.

Zu jedem Kontrollpunkt  $P_k$  gehört eine linksseitige Gewichtsfunktion (blending function)  $F_k^-$  und eine rechtsseitige Gewichtsfunktion  $F_k^+$ . Diese Gewichtsfunktionen geben an, wie stark der Kontrollpunkt sich auf den Verlauf in den jeweils angrenzenden Segmenten auswirkt. Die Gewichtsfunktionen erstrecken sich jeweils über bis zu 2 Segmente, ein Kontrollpunkt kann somit also insgesamt 4 Segmente beeinflussen.

Ein Kurvensegment wird jeweils von den zwei Kontrollpunkten an den Segmentenden und von deren Nachbarn beeinflusst.

Der Wert  $s_k$  eines jeden Kontrollpunktes  $P_k$  legt die Art und die Parameter der rechts- und linksseitigen Gewichtsfunktionen der *benachbarten* Punkte fest (*nicht* aber des Punktes  $P_k$  selbst).

Für  $0 \leq s_k \leq 1$  gilt:

|   |                                   |
|---|-----------------------------------|
| $T_{k-1}^+ = T_k + s_k$   | Ende des Wirkungsbereiches        |
| $p_{k-1}^+ = 2(T_{k-1}^+ - T_{k-1})^2$  | Parameter in der Gewichtsfunktion |
| $F_{k-1}^+(t) = f\left(\frac{t - T_{k-1}^+}{T_{k-1} - T_{k-1}^+}, p_{k-1}^+\right)$ | Gewichtsfunktion linker Nachbar   |
| $T_{k+1}^- = T_k - s_k$   |                                   |
| $p_{k+1}^- = 2(T_{k+1}^- - T_{k+1})^2$  |                                   |
| $F_{k+1}^-(t) = f\left(\frac{t - T_{k+1}^-}{T_{k+1} - T_{k+1}^-}, p_{k+1}^-\right)$ |                                   |

Für  $-1 \leq s_k < 0$  gilt<sup>2</sup>:

$$\begin{aligned}
p_{k-1}^+ &= 2 \\
q_{k-1}^+ &= -\frac{s_k}{2} \\
F_{k-1}^+(t) &= e(T_k - t, p_{k-1}^+, q_{k-1}^+) \\
p_{k+1}^- &= 2 \\
q_{k+1}^- &= -\frac{s_k}{2} \\
F_{k+1}^-(t) &= e(t - T_k, p_{k+1}^-, q_{k+1}^-)
\end{aligned}$$

Nachdem für alle Punkte  $P_k$  die Funktionen  $F_k^-$  und  $F_k^+$  festgelegt wurden, können für jedes Kurvensegment mit den Randpunkten  $P_i$  und  $P_{i+1}$  Kurvenpunkte berechnet werden mit

$$\begin{aligned}
z_x(t) &= x_{i-1}F_{i-1}^+(t) + x_iF_i^+(t) + x_{i+1}F_{i+1}^-(t) + x_{i+2}F_{i+2}^-(t) \\
z_y(t) &= y_{i-1}F_{i-1}^+(t) + y_iF_i^+(t) + y_{i+1}F_{i+1}^-(t) + y_{i+2}F_{i+2}^-(t) \\
n(t) &= F_{i-1}^+(t) + F_i^+(t) + F_{i+1}^-(t) + F_{i+2}^-(t) \\
z'_x(t) &= \frac{dz_x}{dt} = x_{i-1} \frac{\partial F_{i-1}^+(t)}{\partial t} + x_i \frac{\partial F_i^+(t)}{\partial t} + x_{i+1} \frac{\partial F_{i+1}^-(t)}{\partial t} + x_{i+2} \frac{\partial F_{i+2}^-(t)}{\partial t} \\
z'_y(t) &= \frac{dz_y}{dt} = y_{i-1} \frac{\partial F_{i-1}^+(t)}{\partial t} + y_i \frac{\partial F_i^+(t)}{\partial t} + y_{i+1} \frac{\partial F_{i+1}^-(t)}{\partial t} + y_{i+2} \frac{\partial F_{i+2}^-(t)}{\partial t} \\
n'(t) &= \frac{\partial F_{i-1}^+(t)}{\partial t} + \frac{\partial F_i^+(t)}{\partial t} + \frac{\partial F_{i+1}^-(t)}{\partial t} + \frac{\partial F_{i+2}^-(t)}{\partial t} \\
x(t) &= \frac{z_x(t)}{n(t)} \\
y(t) &= \frac{z_y(t)}{n(t)} \\
x'(t) &= \frac{dx}{dt} = \frac{n(t) \cdot z'_x(t) - z_x(t) \cdot n'(t)}{n^2(t)} \\
y'(t) &= \frac{dy}{dt} = \frac{n(t) \cdot z'_y(t) - z_y(t) \cdot n'(t)}{n^2(t)}
\end{aligned}$$

<sup>2</sup>In fig2lat wird  $q = -s_k/2$  verwendet, wie in [CB95] beschrieben. Dort wird  $0 \leq q \leq \frac{1}{2}$  gefordert, um unerwünschte Oszillationen zu vermeiden. In XFig und fig2dev wird jedoch  $q = -s_k$  verwendet, wodurch auch  $q$ -Werte mit  $q > \frac{1}{2}$  möglich werden. Das Ergebnis ist ein übermäßig starkes Ausschlagen bei interpolierten Splines und z. T. Oszillationen, beispielsweise bei einer Anordnung entsprechend Abbildung 15 in [CB95] zwischen den Punkten  $P_2$  und  $P_3$ .

Für das erste X-Spline-Segment entfällt der erste Summand, da der Anfangspunkt keinen „linken“ Nachbarpunkt hat. Analog entfällt für das letzte X-Spline-Segment der jeweils letzte Summand.

Für die Vereinfachung des Berechnungsprogrammes wird jedes X-Spline-Segment so in eine Anordnung mit vier Punkten  $A$ ,  $B$ ,  $C$  und  $D$  gelegt, dass der Kontrollpunkt für den Anfang des Segments auf  $B$  liegt und der Kontrollpunkt für den Endpunkt des Segmentes auf  $C$ .  $A$  entspricht dann dem „linken“ benachbartem Kontrollpunkt,  $D$  dem „rechten“.

### 3.4 Annäherung von X-Splines durch Bezier-Splines

Die für fig2lat vorgesehenen Ausgabeformate unterstützen Bezier-Splines. Es ist erforderlich, X-Splines näherungsweise durch Bezier-Splines nachzubilden.

Zunächst werden Kurvenpunkte berechnet, die den Kontrollpunkten entsprechen (d.h. Kurvenpunkte mit  $t = T_1, t = T_2 \dots$ ). Um die Kontrollpunkte für die Bezier-Splines zu errechnen, wird der Anstieg in den Kurvenpunkten benötigt. Da X-Splines Knicke in den Kurvenpunkten aufweisen können, die den Kontrollpunkten entsprechen, wird sowohl ein „linksseitiger“ als auch ein „rechtsseitiger“ Anstieg berechnet.

X-Splines sind Kurven fünften Grades – und weichen somit von Bezier-Splines dritten Grades ab – es ist deshalb sinnvoll, jedes X-Spline-Segment durch mehrere Bezier-Spline-Segmente darzustellen. Zu diesem Zweck werden Kurvenpunkte und Anstiege auch zwischen den im vorangegangenen Schritt berechneten Kurvenpunkten berechnet. Da hier keine Knicke auftreten, genügt jeweils eine Berechnung des Anstieges.

Erfolgt die Darstellung jedes X-Spline-Segmentes durch mehrere ( $m$ ) Bezier-Spline-Segmente, ist eine Korrektur der Anstiege erforderlich. Bezier-Splines beginnen und enden jeweils bei  $t = 0$  und  $t = 1$ . Die bisher errechneten Anstiege beziehen sich aber auf Bereiche mit  $t = t_s$  und  $t = t_s + \frac{1}{m}$ . Die Verwendung der berechneten Kurvenpunkte als Anfangs- und Endpunkte von Bezier-Splines entspricht also einer Streckung entlang der  $t$ -Achse um den Faktor  $m$ . Damit geht eine Verringerung des Anstieges einher, alle bisher berechneten Anstiege müssen also korrigiert werden, indem sie durch  $m$  dividiert werden.



### 3.5 Längenberechnung an Splines

Die Längenberechnung einer Kurve erfolgt durch das Integral

$$l = \int \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2} dt$$

Da ich weder für X-Splines noch für Bezier-Splines eine analytische Lösung des Problems gefunden habe, wird die numerische Näherung

$$l \approx \sum_{i=1}^n \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}$$

verwendet.

Das Spline oder das Spline-Segment wird in viele kleine Intervalle unterteilt, für jedes Intervall werden Anfangs- und Endpunkt berechnet, die daraus gewonnenen Intervalllängen werden aufaddiert.

In einer Iteration wird bei jedem Durchlauf die Intervallanzahl erhöht (verdoppelt). Die Iteration wird erfolgreich beendet, falls die Differenz zwischen dem Ergebnis des aktuellen Durchlaufes und dem Ergebnis des vorangegangenen Durchlaufes eine bestimmte Genauigkeitsschranke unterschreitet. Wurde nach einer bestimmten Anzahl Durchläufe die Iteration nicht erfolgreich beendet, so wird sie erfolglos abgebrochen.

### 3.6 Längenkorrektur für Pfeilspitzen an Splines

Die Längenkorrektur an Splines ist vergleichsweise schwierig, da hier die Formeln nicht analytisch so umgestellt werden können, dass man ein  $t$  für einen vorgegebenen Abstand zum Endpunkt erhält.

Um ein  $t$  zu bestimmen, für das die Länge vom zugehörigen Punkt bis zum Spline-Ende den vorgegebenen Wert  $s$  hat, wird das Intervallschachtelungsverfahren (Halbierungsverfahren) benutzt. Zunächst wird überprüft, ob  $s$  kleiner ist als die Gesamtlänge des Splines. Falls ja, werden als Startwerte für die Intervallgrenzen das  $t$  für den ersten Punkt (0) und den letzten Punkt ( $n - 1$ ) benutzt.

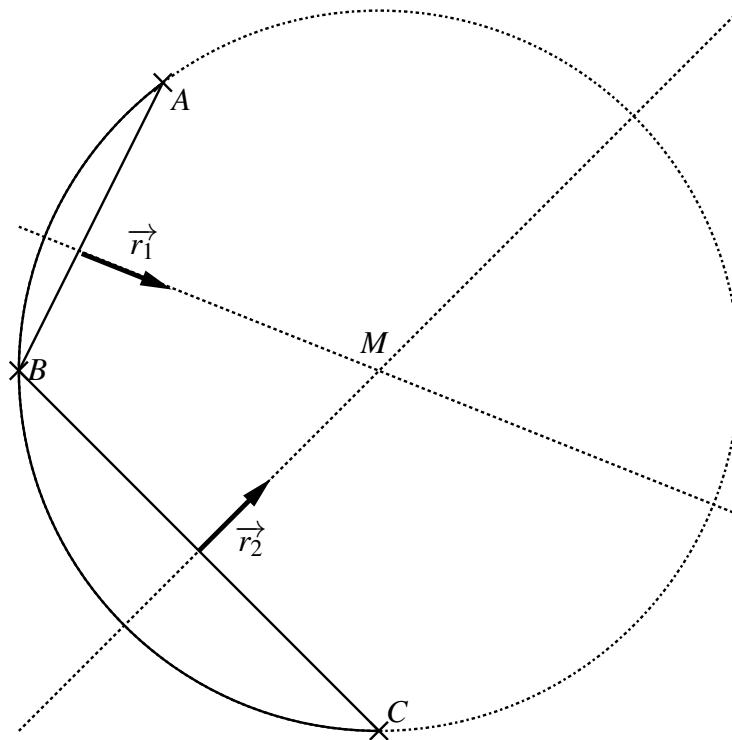


Abbildung 3: Berechnungen am Kreisbogen

## 4 Kreisbögen

### 4.1 Berechnung von Mittelpunkt und Radius

Kreisbögen sind in Fig-Dateien durch drei Punkte gegeben, die hier mit  $A(x_A, y_A)$ ,  $B(x_B, y_B)$  und  $C(x_C, y_C)$  bezeichnet werden. Mit  $\vec{r}_A$ ,  $\vec{r}_B$  und  $\vec{r}_C$  werden die Vektoren vom Koordinatenursprung zu den drei Punkten bezeichnet. A und C sind die Endpunkte des Bogens, B ist ein innerer Punkt.

Zum Zeichnen der Kreise und zur Berechnung der Bounding-Box werden der Mittelpunkt  $M$  und der Radius  $r$  benötigt.

Der Mittelpunkt  $M(x_M, y_M)$  ergibt sich als Schnittpunkt der Mittelsenkrechten auf den Strecken  $\overline{BA}$  und  $\overline{BC}$

$$g_1: \quad \vec{r} = \vec{r}_B + \frac{1}{2}\vec{BA} + t\vec{r}_1 \quad t \in \mathbb{R}$$

$$g_2: \quad \vec{r} = \vec{r}_B + \frac{1}{2}\vec{BC} + s\vec{r}_2 \quad s \in \mathbb{R}$$

Da die Mittelsenkrechten senkrecht auf den jeweiligen Strecken stehen, muss gelten:

$$\begin{aligned} 0 &= \vec{r}_1 \cdot \vec{AB} & 0 &= \vec{r}_2 \cdot \vec{CB} \\ 0 &= \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \cdot \begin{pmatrix} x_B - x_A \\ y_B - y_A \end{pmatrix} & 0 &= \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \cdot \begin{pmatrix} x_B - x_C \\ y_B - y_C \end{pmatrix} \end{aligned}$$

Dies wird erfüllt durch:

$$\begin{aligned} x_1 &= y_B - y_A & x_2 &= y_B - y_C \\ y_1 &= x_A - x_B & y_2 &= x_C - x_B \end{aligned}$$

Somit ergibt sich für die Geradengleichungen:

$$\begin{aligned} g_1: \quad \vec{r} &= \frac{1}{2} \begin{pmatrix} x_A + x_B \\ y_A + y_B \end{pmatrix} + t \begin{pmatrix} y_B - y_A \\ x_A - x_B \end{pmatrix} \\ g_2: \quad \vec{r} &= \frac{1}{2} \begin{pmatrix} x_C + x_B \\ y_C + y_B \end{pmatrix} + s \begin{pmatrix} y_B - y_C \\ x_C - x_B \end{pmatrix} \end{aligned}$$

Im Schnittpunkt  $M$  beider Geraden gilt:

$$\begin{aligned} x_B + \frac{1}{2}(x_A - x_B) + t_M(y_B - y_A) &= x_B + \frac{1}{2}(x_C - x_B) + s_M(y_B - y_C) \\ y_B + \frac{1}{2}(y_A - y_B) + t_M(x_B - x_A) &= y_B + \frac{1}{2}(y_C - y_B) + s_M(x_B - x_C) \\ \frac{1}{2}(x_A + x_B) + t_M(y_B - y_A) &= \frac{1}{2}(x_C + x_B) + s_M(y_B - y_C) \\ \frac{1}{2}(y_A + y_B) + t_M(x_B - x_A) &= \frac{1}{2}(y_C + y_B) + s_M(x_B - x_C) \\ t_M(y_B - y_A) + s_M(y_C - y_B) &= \frac{1}{2}(x_C - x_A) \\ t_M(x_A - x_B) + s_M(x_B - x_C) &= \frac{1}{2}(y_C - y_A) \end{aligned}$$

Dieses Gleichungssystem hat entweder

- genau eine Lösung

$$t_M = \frac{\frac{1}{2} \left( (x_B - x_C)(x_C - x_A) - (y_C - y_B)(y_C - y_A) \right)}{(y_B - y_A)(x_B - x_C) - (y_C - y_B)(x_A - x_B)}$$

- keine Lösung (die Punkte liegen auf einer Geraden) oder
- unendlich viele Lösungen (zwei oder drei Punkte sind identisch).

In den beiden letzten Fällen hat der Nenner den Wert 0. Ist der Nenner positiv, wird der Kreisbogen im mathematisch positiven Umlaufsinn gezeichnet. Ist der Nenner negativ, wird im mathematisch negativen Umlaufsinn gezeichnet.

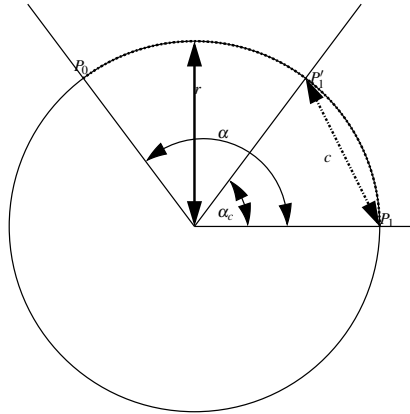


Abbildung 4: Korrektur am Kreisbogen

## 4.2 Längenkorrektur für Pfeilspitzen an Kreisbögen

Ist ein Kreisbogen von  $P_0$  nach  $P_1$  durch den Radius  $r$  und Winkel  $\alpha$  gegeben und soll so verkürzt werden, dass  $c$  die Entfernung zwischen dem neuen Endpunkt  $P'_1$  und dem alten Endpunkt  $P_1$  ist, so kann die Winkelverkürzung  $\alpha_c$  folgendermaßen berechnet werden:

$$\begin{aligned}
 c &= \sqrt{(x'_1 - x_1)^2 + (y'_1 - y_1)^2} \\
 &= \sqrt{(r \cos \alpha_c - r)^2 + (0 - r \sin \alpha_c)^2} \\
 &= \sqrt{r^2 - 2r^2 \cos \alpha_c + r^2 \cos^2 \alpha_c + r^2 \sin^2 \alpha_c} \\
 &= r \sqrt{1 - 2 \cos \alpha_c + \cos^2 \alpha_c + \sin^2 \alpha_c} \\
 &= r \sqrt{2 - 2 \cos \alpha_c} \\
 \frac{c}{r} &= \sqrt{2 - 2 \cos \alpha_c} \\
 2 - 2 \cos \alpha_c &= \left(\frac{c}{r}\right)^2 \\
 2 \cos \alpha_c &= 2 - \left(\frac{c}{r}\right)^2 \\
 \cos \alpha_c &= 1 - \frac{c^2}{2r^2} \\
 \alpha_c &= \arccos\left(1 - \frac{c^2}{2r^2}\right)
 \end{aligned}$$

### 4.3 Annäherung von Kreisbögen durch Bezier-Splines

Die nachfolgende Untersuchung wurde inspiriert durch die deutsche Wikipedia-Seite zum Thema „Bezierkurven“. Dort ist ein Verfahren angegeben, wie Kontrollpunkte für die näherungsweise Darstellung eines Viertelkreises durch ein Bezier-Spline-Segment gewonnen werden können. Dieses Verfahren wird hier verallgemeinert, um Kreisbögen mit beliebigem Öffnungswinkel darstellen zu können. In fig2lat wird die Formel für Kreisbogensegmente mit  $|\alpha| \leq \frac{\pi}{2}$  verwendet, größere Kreisbögen werden dann durch mehrere Bezier-Segmente realisiert.

Im Abschnitt 3.2 auf Seite 9 wurde gezeigt, wie aus Start- und Endpunkt einer parametrischen Kurve und den Ableitungen  $d\vec{r}/dt$  in Start- und Endpunkt die Kontrollpunkte für ein Bezier-Spline-Segment gewonnen werden können. Damit wird ein Bezier-Spline-Segment erzeugt, das den Verläufen  $x(t)$  und  $y(t)$  möglichst nahe kommt.

Hauptziel der Annäherung ist jedoch eine möglichst gute Annäherung an gewünschte  $x$ - $y$ -Verläufe. Deshalb wird hier untersucht, ob eine Variation der Kontrollpunkte zu besseren Ergebnissen führt.

Die mit

$$\begin{aligned}\vec{r}_{cs} &= \vec{r}_s + \frac{1}{3} \dot{\vec{r}}(0) \\ \vec{r}_{ce} &= \vec{r}_e - \frac{1}{3} \dot{\vec{r}}(1)\end{aligned}$$

gewonnenen Bezier-Kurven weisen zwar am Rand des Intervalles  $[0; 1]$  den geforderten Radius auf, im Intervall-Inneren verläuft die Kurve aber innerhalb des Kreises, d. h. der Radius wird unterschritten.

Um die Kurve zu ändern, werden die Kontrollpunkte etwas weiter von Start- und Endpunkt weggeschoben, sie bleiben aber auf den Tangenten in Start- und Endpunkt. Aufgrund der Symmetrie der Anordnung werden beide Kontrollpunkte in gleichem Abstand zum jeweiligen Endpunkt gehalten.

Für den Kreis gilt:

$$\tilde{k}_k : \quad \vec{r}_k(t) = \begin{pmatrix} r \cos \alpha t \\ r \sin \alpha t \end{pmatrix}$$

Die erste Ableitung ergibt

$$\dot{\vec{r}}_k(t) = \begin{pmatrix} -r\alpha \sin \alpha t \\ r\alpha \cos \alpha t \end{pmatrix}$$

Als Start- und Endpunkt erhalten wir

$$\vec{r}_s = \vec{r}_k(0) = \begin{pmatrix} r \\ 0 \end{pmatrix}$$

$$\vec{r}_e = \vec{r}_k(1) = \begin{pmatrix} r \cos \alpha \\ r \sin \alpha \end{pmatrix}$$

Als Ansatz für die Kontrollpunkte wählen wir

$$\vec{r}_{cs} = \vec{r}_s + \varkappa \dot{\vec{r}}_k(0)$$

$$\vec{r}_{ce} = \vec{r}_e - \varkappa \dot{\vec{r}}_k(1)$$

und erhalten damit die Bezier-Kurve

$$\tilde{k}_b: \quad \vec{r}_b(t) = (1-t)^3 \vec{r}_s + 3(1-t)^2 t \left( \vec{r}_s + \varkappa \dot{\vec{r}}_k(0) \right) \\ + 3(1-t)t^2 \left( \vec{r}_e - \varkappa \dot{\vec{r}}_k(1) \right) + t^3 \vec{r}_e$$

Die oben aufgeführten Gleichungen werden in die Forderung

$$r = \left| \vec{r}_b\left(\frac{1}{2}\right) \right|$$

eingesetzt und diese dann nach  $\varkappa$  umgestellt:

$$\varkappa = \frac{2\sqrt{8(1-\cos\alpha)} - 4\sin\alpha}{3\alpha(1-\cos\alpha)}$$

Dabei ergeben sich recht lange Gleichungen. Zum Einsetzen, Umstellen und Lösen von Gleichungen wurde deshalb die Mathematik-Software wxMaxima mit folgenden Befehlen benutzt:

```

1  /* x- und y-Koordinaten des Kreises. */
2  X: R * cos(alpha * t);
3  Y: R * sin(alpha * t);
4
5  /* Start- und Endpunkt des Kreisbogens. */
6  xs: at(X, t=0);
7  ys: at(Y, t=0);
8  xe: at(X, t=1);
9  ye: at(Y, t=1);
10
11 /* Ableitungen der Kreis-Koordinaten. */
12 dXdt: diff(X, t);
13 dYdt: diff(Y, t);
14
15 /* Formeln fuer Kontrollpunkte, kappa ist noch unbekannt. */
16 xcs: xs + kappa * at(dXdt, t=0);
17 ycs: ys + kappa * at(dYdt, t=0);
18 xce: xe - kappa * at(dXdt, t=1);
19 yce: ye - kappa * at(dYdt, t=1);
20
21 /* x- und y-Koordinaten der Bezierkurve. */
22 xb: (1-t)^3*xs + 3*(1-t)^2*t*xcs + 3*(1-t)*t^2*xce + t^3*xe;
23 yb: (1-t)^3*ys + 3*(1-t)^2*t*yys + 3*(1-t)*t^2*yce + t^3*ye;
24
25 /* Fuer t=0.5 soll Bezierkurve auch den Radius R haben.
26     Wir suchen das dafuer passende kappa.
27 */
28 res: solve( at(xb, t=1/2)^2 + at(yb, t=1/2)^2 = R^2, kappa);
29
30 /* Ergebnis vereinfachen. */
31 trigsimp(rhs(res[2]));

```



## Literatur

- [CB95] Christophe Schlick Carole Blanc. *X-Splines: A Spline Model Designed for the End-User*. SIGGRAPH proceedings 1995, 1995.