Visualisierung von Fräsfehlern auf Basis von CAD-Daten

Dissertation

zur Erlangung des akademischen Grades
„Doktor der Naturwissenschaften“
(Dr. rer. nat.)

am Fachbereich Mathematik
der Justus-Liebig-Universität Gießen

vorgelegt von

Dipl.-Math. Dominik Wagenführ
geb. in Gotha

Erstgutachter:	Prof. Dr. Tomas Sauer
Zweitgutachter:	Prof. Dr. Johannes Wallner
Drittgutachter:	Prof. Dr. Wolfgang Papiernik

Erlangen, April 2008

Zuerst möchte ich mich bei Herrn Prof. Dr. Tomas Sauer für seine sehr gute Betreuung in den letzten Jahren bedanken. Er stand mir bei meinem Promotionsvorhaben immer zur Seite und hat durch zahlreiche Ideen sehr zu dieser Arbeit beigetragen.

Als Zweites möchte ich Herrn Prof. Dr. Wolfgang Papiernik meinen Dank aussprechen, der meine Arbeit bei Siemens betreut hat. Vor allem seine technischen Erklärungen haben zu einem besseren Verständnis beigetragen.

Inhaltsverzeichnis

Teil I
Einleitung

Bei der Erstellung von Werkstücken in der Industrie greift man häufig auf CAD-Programme (*Computer Aided Design*) zurück, mit deren Hilfe man die Geometrie der Flächen in mathematischer Form beschreiben kann (*Geometrie*). Mittels eines CAM-Systems (*Computer Aided Modelling*) wird eine Ausgabe generiert, wie das Werkstück hergestellt wird. So wird bei einem Fräsprozess über dreidimensionale Positionsangaben der Weg angegeben, den der Fräskopf zurücklegt (*Fräsbahn*). Zusätzlich erhält man zum Beispiel Angaben zur gewünschten Bewegungsgeschwindigkeit des Fräsers. Diese Daten können aus dem CAM-System heraus in einem speziellen Format gespeichert werden (*NC-Programm*). Das NC-Programm (NC steht für *Numerical Controlled*) wird danach an eine *Steuerung* gegeben, welche die Daten über die *Antriebe* für eine *Werkzeugmaschine* umwandelt. Diese fertigt zum Schluss das *Werkstück* (siehe Abbildung 1.1)

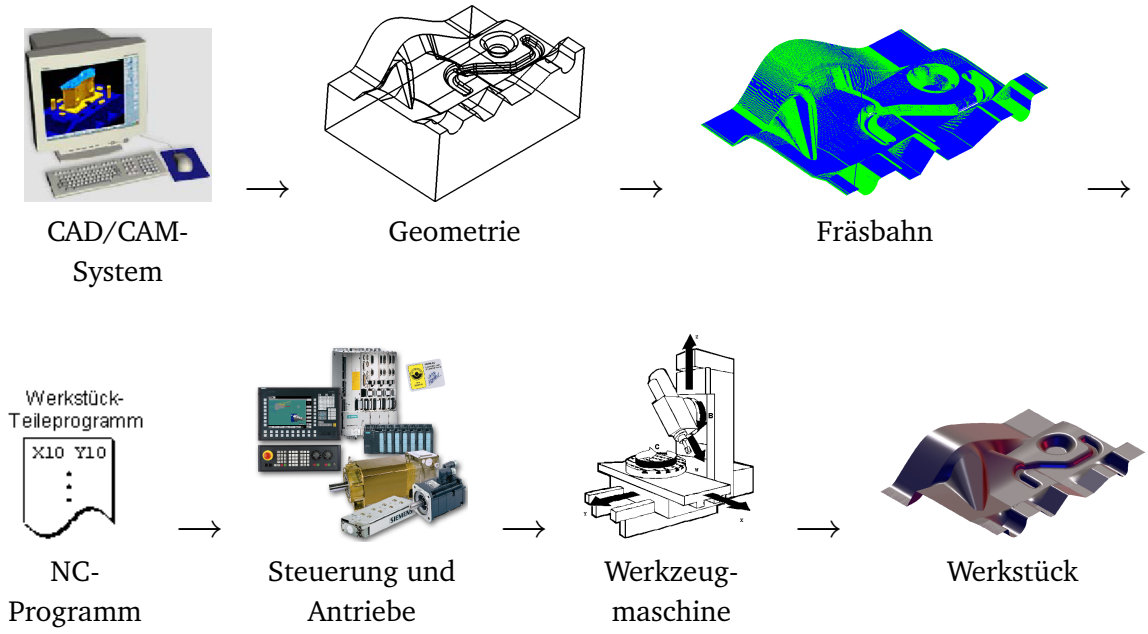


Abbildung 1.1: Prozesskette: Vom Werkstück-Design zur Fertigung (Abbildungen mit freundlicher Genehmigung der Siemens AG)

In dieser Prozesskette kann es an verschiedenen Stellen zu Fehlern kommen, die man meist erst am Ende der Fertigung sieht. So können die vom CAD-Designer eingegebenen Daten nicht exakt sein oder das CAM-System berechnet eine falsche Fräsbahn. Zusätzlich spielen bei der Fertigung auch physikalische Größen, wie zum Beispiel die Schwingung der Maschine oder Form und Größe des Fräsers eine starke Rolle. In dieser Arbeit soll der Frage nachgegangen werden, ob sich solche Fehler berechnen und an einem Computer darstellen lassen, ohne dass man das Werkstück fertigen muss.

Die Verarbeitung der CAM-Daten kann bei Siemens in einem speziellen Programm namens *VisuTool* simuliert werden. Hierzu werden die aus dem CAM-System extrahierten Fräsbahnen geladen und das VisuTool simuliert den Fräsprozess. Dabei wird aber nicht auf physikalische Eigenschaften einer echten Werkzeugmaschine eingegangen, so dass zum Beispiel in der Realität auftretende Schwingungen nicht mit in die Berechnung einfließen.

Als Ergebnis der Frässimulation erhält man eine dreidimensionale, triangulierte *Punktewolke*. Das VisuTool beherrscht dabei das *Dreiachsfräsen* und das *Fünfachsfräsen*, jeweils mit verschiedenen Fräsköpfen (zum Beispiel in Zylinder- oder Kugelform). Beim Dreiachsfräsen steht der Fräskopf senkrecht über der zu bearbeitenden Fläche. Er kann sich zwar in allen drei Dimensionen bewegen, aber nicht kippen. Beim Fünfachsfräsen ist darüber hinaus auch ein Kippen des Fräsers in zwei Richtungen möglich. Die Punktewolke wird generiert, indem beim Dreiachsfräsen ein zweidimensionales „Nagelbrett“ in der x/y-Ebene mit dem Fräskopf geschnitten wird. Beim Fünfachsfräsen wird der Fräskopf mit den drei „Nagelbrettern“ aus der x/y-, x/z- und y/z-Ebene geschnitten (siehe Hoschek und Lasser [22, S. 577 ff], Jerard und Drysdale [24], Schwanecke und Kobbelt [31]).

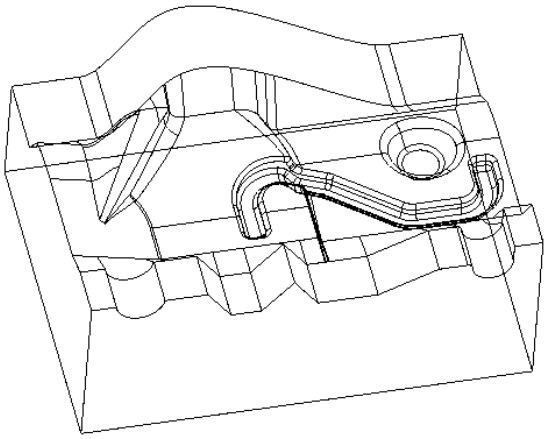
Für eine Fehlerdarstellung sollen diese simulierten Punktedaten mit den ursprünglichen CAD-Daten, die im *IGES-Format* vorliegen, verglichen werden. Es ist dabei nicht zwingend erforderlich, dass die Punktedaten vom VisuTool stammen. Auch direkt aus der Steuerung gespeicherte oder per Laserabtastung erhaltene Werte können zum Vergleich herangezogen werden.

Es gibt bereits viele allgemeine Verfahren für die Segmentierung von Punktewolken und deren Parametrierung (Dietz [8, S. 25 ff]), keine nutzt hierfür aber vorhandene Flächendaten aus. In dem hier vorgestellten Verfahren werden daher zuerst die Kurven und Flächen aus der CAD-Datei extrahiert und in eine Splinegestalt konvertiert. Danach wird jeder Punkt aus der Punktewolke genau einer Fläche zugeordnet. Hierbei greift man auf eine minimale Abstandseigenschaft zurück, um sie korrekt zu verbinden. Der Abstand von jedem Punkt zur zugehörigen Fläche kann zum Schluss ausgegeben oder farblich visualisiert werden.

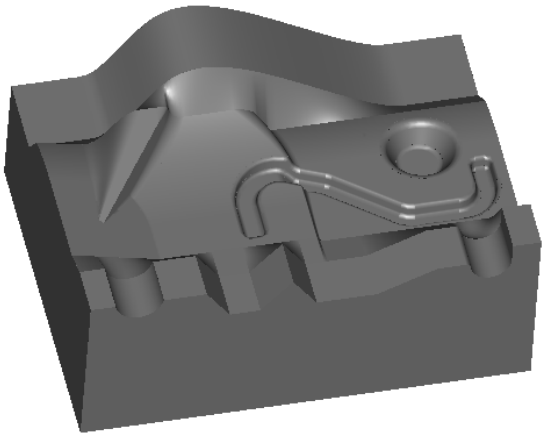
Als Anhaltspunkt für die Ausführung gibt es eine Beispieldatei, welche ein Testwerkstück von DaimlerChrysler beschreibt (Abbildung 1.2). Das Werkstück besitzt dabei zum einen verschiedene geometrischen Flächen wie Ebenen, Translationsflächen oder Rotationsflächen, bietet durch seine Struktur aber auch viele interessante Bereiche, die bei einem Fräsdurchgang zu Problemen führen können (siehe Kapitel IV).

Die Algorithmen und Methoden in dieser Arbeit wurden als Bibliotheken in der Programmiersprache C++ implementiert. Diese sollen bei Siemens später in das VisuTool eingebunden werden, so dass man direkt aus dem Programm heraus die Visualisierung der Fehler starten kann.

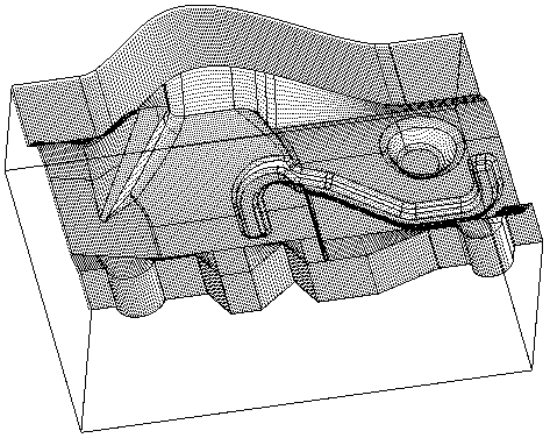
Die CAD-Daten liegen im IGES-Format als Splinekurven und -flächen vor oder können in diese konvertiert werden. Daher bilden die mathematischen Grundlagen der *B-Spline-Kurven* und *Tensor-Produkt-B-Spline-Flächen* in Kapitel II den Anfang dieser Arbeit. In Teil III werden die CAD-Daten und deren Extrahierung genau erklärt und es wird dargestellt, zu welchen



a. Wireframe



b. Solid



c. Punktewolke

Abbildung 1.2: DaimlerChrysler-Testwerkstück

Problemen es dabei kommen kann. Kapitel IV beschäftigt sich mit der eindeutigen Zuordnung der Punkte zu den Flächen, wofür ein mehrstufiges Verfahren durchlaufen wird. Nach der Zuordnung kann man die Punkte optional durch Tensor-Produkt-B-Spline-Flächen approximieren, wie dies in Kapitel V beschrieben ist. Im vorletzten Teil VI werden die einzelnen Fehler pro Punkt der Punktewolke berechnet und im VisuTool farblich dargestellt. Darüber hinaus werden die Algorithmen dieser Arbeit in Hinblick auf Geschwindigkeit und Genauigkeit mit der bereits existierenden Lösung *Net-Compare* verglichen. Zum Schluss werden die Methoden und Ergebnisse zusammengefasst.

Teil II
Splines

Der Begriff *Spline* ist eine Bezeichnung aus dem Schiffsbau. Der Spline ist eine lange Holzlatte, die an bestimmten Punkten fixiert ist und mit deren Hilfe man den Rumpf der Schiffe modellierte. Der Spline nahm dabei die Form mit der geringsten (Biege-)Energie an.

Auf den folgenden Seiten werden Bézier-Splines (auch Bézier-Kurven genannt), B-Spline-Kurven und B-Spline-Flächen samt ihrer Eigenschaften kurz vorgestellt. Bézier-Splines gehen zurück auf die (unabhängigen) Entwicklungen von Paul de Casteljau und Pierre Bézier Anfang der 60er Jahre des 20. Jahrhunderts (Farin [10, S. xvi]). B-Splines wurden schon früher Mitte der 40er Jahre von Isaac Jacob Schoenberg und Haskell Brooks Curry beschrieben, kamen aber auch erst Mitte der 60er Jahre „in Mode“, als sich andere Mathematiker, unter anderem Carl de Boor, Maurice Cox und Lois Mansfield, mit ihnen beschäftigten (Farin [10, S. 119]).

Für genauere Informationen bietet sich die Literatur von Farin [10], Hoschek und Lasser [22] und Piegl und Tiller [27] an.

1 Splinekurven

Splinekurven werden häufig als *parametrische Kurven* geschrieben – in diesem Beispiel als Raumkurve – mit

$$C(t) = (x(t), y(t), z(t))^T, \quad t \in [0,1],$$

wobei $x, y, z : [0,1] \to \mathbb{R}$ ist. Eine besondere Klasse von Splines sind *Bézier-Kurven*.

1.1 Bézier-Kurven

Definition 1.1. *Eine* Bézier-Kurve *vom Grad* $n \in \mathbb{N}$ *ist definiert als*

$$C(t) := \sum_{i=0}^{n} d_i B_{i,n}(t), \quad t \in [0,1], \tag{1.1}$$

mit den als Basis benutzten Bernstein-Polynomen

$$B_{i,n}(t) := \binom{n}{i} t^i (1-t)^{n-i}. \tag{1.2}$$

Die Punkte $d_i \in \mathbb{R}^l, l \in \{1,2,3\}$, *heißen* Kontrollpunkte.

Die Bernstein-Polynome haben einige wichtige Eigenschaften (Piegl und Tiller [27, S. 15 ff]):

1. Nicht-Negativität: $B_{i,n}(t) \geq 0$, $t \in [0,1]$.
2. Partition der Eins: $\sum_{i=0}^{n} B_{i,n}(t) = 1$, $t \in [0,1]$.

3. Rekursive Definition:

$$B_{i,n}(t) = \begin{cases} 0, & i < 0 \text{ oder } i > n, \\ 1, & i = 0, n = 0, \\ tB_{i-1,n-1}(t) + (1-t)B_{i,n-1}(t), & \text{sonst.} \end{cases} \tag{1.3}$$

Durch die Bernstein-Polynome haben auch Bézier-Kurven wichtige Eigenschaften (Farin [10, S. 60 ff]):

1. Affine Invarianz: Affine Transformationen wie Drehungen, Verschiebungen oder Skalierungen verändern die Form der Kurve nicht und müssen nur auf die Kontrollpunkte angewendet werden.

2. Konvexe-Hülle-Eigenschaft: Jeder Punkt der Kurve liegt in der konvexen Hülle des Kontrollpolygons.

Definition 1.2. *Die Kurve*

$$C(t) := \frac{\sum_{i=0}^{n} w_i d_i B_{i,n}(t)}{\sum_{i=0}^{n} w_i B_{i,n}(t)}, \quad t \in [0,1], \tag{1.4}$$

heißt rationale Bézier-Kurve *vom Grad n mit* Gewichten $w_i \in \mathbb{R}, i = 0, \ldots, n$.

(Rationale) Bézier-Kurven werden in dieser Arbeit bei der Graderhöhung von B-Spline-Kurven und bei der Konstruktion von B-Spline-Kurven aus geometrischen Kreisen benutzt.

1.2 B-Spline-Kurven

Eine B-Spline-Funktion der Ordnung $k := m + 1, m \in \mathbb{N}$, ist ein stückweises Polynom vom Grad m, das global stetig und lokal differenzierbar ist. Der Funktion liegen dabei eine endliche, geordnete Menge von Parameterwerten, so genannte *Knoten* $t_0 \leq \ldots \leq t_{k+n-1}, n \geq k$, zugrunde, welche auch als *Knotenvektor* $T := [t_0, \ldots, t_{k+n-1}]$ geschrieben werden können. Ist $t_i < t_{i+1} = \ldots = t_{i+l} < t_{i+l+1}$, so hat der Knoten t_{i+1} die *Vielfachheit* $l \leq k$.

Definition 1.3. *Sei ein Knotenvektor* $T := [t_0, \ldots, t_{k+n-1}]$ *gegeben. Dann bezeichnen* $N_{i,k}$ *die* normalisierten B-Spline-Funktionen *der Ordnung k und es ist für* $k = 1$*:*

$$N_{i,1}(t|T) := \begin{cases} 1, & t_i \leq t < t_{i+1}, \\ 0, & \textit{sonst}, \end{cases} \tag{1.5}$$

und für $k > 1$*:*

$$N_{i,k}(t|T) := \frac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t|T) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t|T) \tag{1.6}$$

mit $i = 0, \ldots, n-1$. *Für* $i < 0$ *und* $i \geq n$ *sei* $N_{i,k}(t|T) := 0$.

Konvention 1.4. Der Knotenvektor T in $N_{i,k}(t|T)$ wird im weiteren Verlauf nicht mehr explizit angeben, soweit T eindeutig aus dem Kontext erkennbar ist.

Zu den wichtigsten Eigenschaften der B-Spline-Funktionen zählen (Piegl und Tiller [27, S. 55 ff]):

1. Lokaler Träger: $N_{i,k}(t) = 0,\ t \notin [t_i, t_{i+k+1})$.
2. Nicht-Negativität: $N_{i,k}(t) \geq 0,\ t \in [t_0, t_{n+k-1}]$.
3. Partition der Eins: $\sum_{i=j-k+1}^{j} N_{i,k}(t) = 1,\ t \in [t_j, t_{j+1})$.

Lemma 1.5. *Es seien eine Ordnung $k \geq 1$ und ein Knotenvektor $T := [\underbrace{0,\ldots,0}_{k},\underbrace{1,\ldots,1}_{k}]$ gegeben. Dann gilt:*

$$N_{i,k}(t|T) = B_{i,k-1}(t), \quad t \in [0,1],\ i = 0,\ldots,k-1.$$

Die Aussage folgt sofort aus der rekursiven Definition der Bernstein-Polynome und der Basis-Splines (siehe Piegl und Tiller [27, S. 52]).

Definition 1.6. *Es seien n* Kontrollpunkte $d_i \in \mathbb{R}^l, l \in \{1,2,3\}$, Gewichte $w_i \in \mathbb{R}, i = 0,\ldots,n-1$, *und ein Knotenvektor $T := [t_0,\ldots,t_{n+k-1}]$ gegeben. Dann bezeichnet*

$$X(t) := \frac{\sum_{i=0}^{n-1} w_i d_i N_{i,k}(t)}{\sum_{i=0}^{n-1} w_i N_{i,k}(t)} \tag{1.7}$$

eine rationale B-Spline-Kurve *der Ordnung k. Die Punkte d_i werden auch* de-Boor-Punkte *genannt.*

In dieser Arbeit werden nicht nur rationale B-Spline-Kurven, auch *NURBS (Non-Uniform Rational B-Splines)* genannt, behandelt, sondern ebenso der Sonderfall der *„NUBS“*, deren Gewichte w_i alle 1 sind. Man spricht dann von einer *polynomialen B-Spline-Kurve*. Die Darstellung aus (1.7) vereinfacht sich zu

$$X(t) := \sum_{i=0}^{n-1} d_i N_{i,k}(t). \tag{1.8}$$

Die Bezeichnung „non-uniform“ bezieht sich auf die nicht-äquidistante Knotenfolge der inneren Knoten.

Konvention 1.7.

1. Die in dieser Arbeit behandelten Kurven sollen an den Randknoten t_0 und t_{n+k-1} die beiden äußeren Kontrollpunkte d_0 und d_{n-1} interpolieren:

$$X(t_0) = d_0, \quad X(t_{n+k-1}) = d_{n-1}.$$

Hierzu legt man eine k-fache Vielfachheit des ersten und des letzten Knotens fest (Hoschek und Lasser [22, S. 170], Sauer [28, S. 12]). Eine Knotenfolge hat hier also immer die Gestalt:

$$t_0 = \ldots = t_{k-1} < \ldots < t_n = \ldots = t_{n+k-1}.$$

2. In dieser Arbeit bezeichnen die allgemeineren Begriffe *Spline* und *Spline-Kurve* immer eine (rationale) B-Spline-Kurve, soweit nicht anders angegeben. Wird von einer B-Spline-Kurve X gesprochen, seien automatisch Kontrollpunkte, Gewichte, Ordnung und Knotenvektor wie in (1.7) gegeben.

3. Die Norm $\|\cdot\|$ bezeichnet immer die Euklidische Norm $\|\cdot\|_2$.

Korollar 1.8. *Es seien eine B-Spline-Kurve*

$$X(t) := \frac{\sum_{i=0}^{n-1} w_i d_i N_{i,k}(t)}{\sum_{i=0}^{n-1} w_i N_{i,k}(t)}$$

der Ordnung k und eine Bézierkurve

$$C(t) := \frac{\sum_{i=0}^{n-1} w_i d_i B_{i,n-1}(t)}{\sum_{i=0}^{n-1} w_i B_{i,n-1}(t)}$$

vom Grad $n-1$ mit Kontrollpunkten d_i, Gewichten w_i, $i = 0, \ldots, n-1$, und dem Knotenvektor $T := [\underbrace{0, \ldots, 0}_{k}, \underbrace{1, \ldots, 1}_{k}]$ gegeben. Dann gilt:

$$X(t) = C(t), \quad t \in [0,1].$$

Beweis: Da $k - 1 = n - 1$, folgt dies sofort aus Lemma 1.5. □

B-Spline-Kurven haben einige wichtige Eigenschaften (Piegl und Tiller [27, S. 82 ff]):

1. Endpunktinterpolation: Die Endpunkte d_0 und d_{n-1} liegen auf der Kurve (siehe oben).

2. Affine Invarianz: Affine Transformationen wie Drehungen, Verschiebungen oder Skalierungen verändern die Form der Kurve nicht und müssen nur auf die Kontrollpunkte angewendet werden.

3. Konvexe-Hülle-Eigenschaft: Jeder Punkt der Kurve liegt in der konvexen Hülle des Kontrollpolygons.

1.2.1 Auswertung

Für die Auswertung der B-Spline-Kurven steht der de-Boor-Algorithmus zur Verfügung. Dessen Vorteil ist, dass man die B-Spline-Funktionen nicht kennen muss, um die Kurve auszuwerten. Der Algorithmus ist dabei dank der Rekursionsformel für B-Splines leicht zu implementieren.

Erste Version – für rationale Splines

Die erste Version des de-Boor-Algorithmus funktioniert sowohl für rationale als auch polynomiale B-Spline-Kurven. Dennoch wird weiter unten der Algorithmus noch einmal explizit für polynomiale B-Spline-Kurven vorgestellt, da sich dadurch eine kleine, aber nicht unerhebliche Leistungssteigerung erzielen lässt.

Die B-Spline-Kurve X soll an einer Stelle t ausgewertet werden. Bedingung hierfür ist, dass $t_{k-1} \leq t \leq t_n$, denn nur auf diesem Intervall ist die Splinekurve definiert.

Algorithmus 1.9. (nach Farin [10, S. 241])

Eingabe: B-Spline-Kurve X, wobei $d_j^{(0)} := d_j, w_j^{(0)} := w_j, j = 0, \ldots, n-1$, Parameterwert t

Ausgabe: neue Kontrollpunkte $d_j^{(i)}, i = 1, \ldots, k-1, j = r-k+i+1, \ldots, r$, so dass $X(t) = d_r^{(k-1)}$

1. Suche einen Index r mit $t_r \leq t < t_{r+1}$. Sollte $t = t_{k-1}$ oder $t = t_n$ sein, kann man den ersten beziehungsweise letzten Kontrollpunkt als Auswertung von t nehmen, da die äußeren Punkte interpoliert werden.
2. Schleife i von 1 bis $k-1$:
   - Schleife j von r bis $r-k+i+1$:
     - Setze $c := \frac{t-t_j}{t_{j+k-i}-t_j}$.
     - Berechne $w_j^{(i)} := (1-c)w_{j-1}^{(i-1)} + cw_j^{(i-1)}$.
     - Neuer Kontrollpunkt

$$d_j^{(i)} := \frac{(1-c)w_{j-1}^{(i-1)}d_{j-1}^{(i-1)} + cw_j^{(i-1)}d_j^{(i-1)}}{w_j^{(i)}}. \tag{1.9}$$

Der Kontrollpunkt $d_r^{(k-1)}$ ist dann die Auswertung der Kurve an der Stelle t, das heißt $X(t) = d_r^{(k-1)}$.

Zweite Version – für polynomiale Splines

Obwohl dies nur ein Spezialfall der rationalen Version ist, ist eine Unterscheidung dennoch sinnvoll. Da in der ersten Version auch die Gewichte neu erstellt werden, verschwendet man Rechenzeit und Speicherplatz, wenn die Gewichte alle 1 sind, da sich diese in der Berechnung nicht ändern. Zusätzlich vereinfacht sich die Berechnung aus (1.9).

Algorithmus 1.10. (nach Hoschek und Lasser [22, S. 181 f])

Eingabe: polynomiale B-Spline-Kurve X, wobei $d_j^{(0)} := d_j, j = 0, \ldots, n-1$, Parameterwert t

Ausgabe: neue Kontrollpunkte $d_j^{(i)}, i = 1, \ldots, k-1, j = r-k+i+1, \ldots, r$, so dass $X(t) = d_r^{(k-1)}$

1. Suche einen Index r mit $t_r \leq t < t_{r+1}$. Wie oben sollte man den ersten beziehungsweise letzten Kontrollpunkt nehmen, falls t mit einem der beiden äußeren Knoten zusammenfällt.

2. Schleife i von 1 bis $k-1$:

   - Schleife j von r bis $r-k+i+1$:
     - Setze $c := \frac{t-t_j}{t_{j+k-i}-t_j}$.
     - Neuer Kontrollpunkt

$$d_j^{(i)} = (1-c)d_{j-1}^{(i-1)} + cd_j^{(i-1)}. \tag{1.10}$$

Es gilt dann $X(t) = d_r^{(k-1)}$ wie oben.

1.2.2 Ableitungen

Ableitungen der B-Spline-Kurven werden unter anderem für die Abstandsberechnung zu einem Punkt, bei der man auf die Newton-Methode zur Nullstellenfindung zurückgreift, benötigt. Für die Berechnung wird wieder der de-Boor-Algorithmus zu Hilfe genommen.

Ableitung der B-Spline-Funktionen

Zuerst müssen die B-Spline-Funktionen abgeleitet werden, da man die Ableitung der B-Spline-Kurven hierauf zurückführt.

Satz 1.11. *Ist ein Knotenvektor $T := [t_0, \ldots, t_{n+k-1}]$ und die B-Spline-Funktionen $N_{i,k}$ gegeben, so gilt:*

$$N'_{i,k}(t) = \frac{k-1}{t_{i+k-1}-t_i} N_{i,k-1}(t) - \frac{k-1}{t_{i+k}-t_{i+1}} N_{i+1,k-1}(t). \tag{1.11}$$

Der Beweis (siehe Piegl und Tiller [27, S. 59 ff]) wird per Induktion über der Ordnung k geführt und benutzt die rekursive Darstellung der B-Spline-Funktionen.

Konvention 1.12. In manchen *Fällen* können die Nenner in (1.11) Null sein. Dies fällt aber damit zusammen, dass auch die Basis-Funktion identisch verschwindet. In diesem Fall definiert man den Term als Null.

Ableitung der B-Spline-Kurven

Über die Ableitung der B-Spline-Funktionen kann man nun auch B-Spline-Kurven ableiten.

Satz 1.13. *Es sei eine polynomiale B-Spline-Kurve*

$$X(t) := \sum_{i=0}^{n-1} d_i N_{i,k}(t|T)$$

der Ordnung k mit Kontrollpunkten $d_0, \ldots, d_{n-1}$ und einem Knotenvektor $T := [t_0, \ldots, t_{n+k-1}]$ gegeben. Dann ist

$$X'(t) = \sum_{i=0}^{n-2} d_i' N_{i,k-1}(t|T') \tag{1.12}$$

eine B-Spline-Kurve der Ordnung $k-1$ mit Kontrollpunkten

$$d_i' := \frac{k-1}{t_{i+k} - t_{i+1}}(d_{i+1} - d_i), \quad i = 0, \ldots, n-2, \tag{1.13}$$

und einem Knotenvektor $T' := [t_1, \ldots, t_{n+k-2}]$.

Beweis: Es ist

$$\begin{aligned} X'(t) &= \sum_{i=0}^{n-1} d_i N_{i,k}'(t|T) = \sum_{i=0}^{n-1} d_i \left(\frac{k-1}{t_{i+k-1} - t_i} N_{i,k-1}(t|T) - \frac{k-1}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t|T) \right) \\ &= (k-1) \left(\sum_{i=-1}^{n-2} \frac{d_{i+1}}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t|T) - \sum_{i=0}^{n-1} \frac{d_i}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t|T) \right) \\ &= (k-1) \left(\frac{d_{i+1}}{t_{k-1} - t_0} N_{0,k-1}(t|T) + \sum_{i=0}^{n-2} \frac{d_{i+1} - d_i}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t|T) - \frac{d_i}{t_{n+k-1} - t_n} N_{n,k-1}(t|T) \right). \end{aligned}$$

Da der erste und letzte Term aufgrund von Konvention 1.12 Null sind, folgt

$$X'(t) = \sum_{i=0}^{n-2} d_i' N_{i+1,k-1}(t|T)$$

mit

$$d_i' = \frac{k-1}{t_{i+k} - t_{i+1}}(d_{i+1} - d_i), \quad i = 0, \ldots, n-2.$$

Wie man leicht nachvollziehen kann, ist $N_{i+1,k-1}(t|T) = N_{i,k-1}(t|T')$. Damit ist X' eine B-Spline-Kurve der Ordnung $k-1$. □

Für die Ableitung der rationalen B-Spline-Kurven kann Satz 1.13 nicht sofort übernommen werden. Eine rationale B-Spline-Kurve ist nach (1.7) definiert als

$$X(t) := \frac{\sum_{i=0}^{n-1} w_i d_i N_{i,k}(t)}{\sum_{i=0}^{n-1} w_i N_{i,k}(t)}.$$

Möchte man dies ableiten, setzt man

$$X(t) = \frac{P(t)}{Q(t)}$$

mit $P(t) = \sum_{i=0}^{n-1} w_i d_i N_{i,k}(t)$ und $Q(t) = \sum_{i=0}^{n-1} w_i N_{i,k}(t)$. $P(t)$ ist dabei eine vektorwertige und $Q(t)$ eine reellwertige B-Spline-Kurve. Jetzt kann man gemäß Quotienten-Regel ableiten:

$$X'(t) = \frac{Q(t)P'(t) - Q'(t)P(t)}{Q(t)^2}.$$

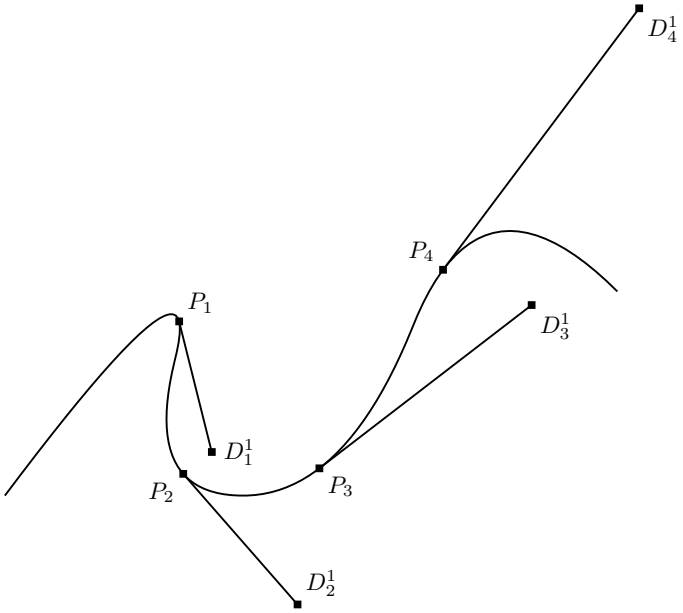


Abbildung 1.1: Kurve mit erster Ableitung in vier Punkten (Ableitungen verkleinert um Faktor 3)

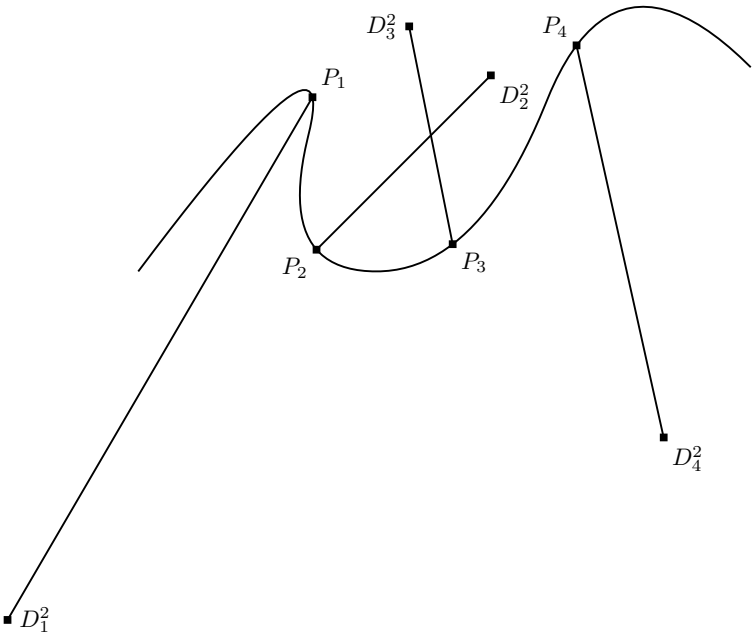


Abbildung 1.2: Kurve mit zweiter Ableitung in vier Punkten (Ableitungen verkleinert um Faktor 25)

Die Ableitung wird dabei nur punktweise am Parameterwert t betrachtet. Da P und Q polynomiale B-Spline-Kurven sind, hat man das Problem der rationalen Kurven auf ein polynomiales zurückgeführt.

Die zweite Ableitung rationaler Splines ist nichts anderes als die wiederholte Anwendung der Quotienten-Regel. Der Übersicht halber wird das Argument t in der folgenden Rechnung nicht aufgeführt. Es ist

$$\begin{aligned} X'' &= \left(\frac{QP' - Q'P}{Q^2}\right)' = \frac{Q^2(QP' - Q'P)' - (Q^2)'(QP' - Q'P)}{Q^4} \\ &= \frac{Q^2(QP'' + Q'P' - Q''P - Q'P') - 2QQ'(QP' - Q'P)}{Q^4} \\ &= \frac{Q(QP'' - Q''P) - 2Q'(QP' - Q'P)}{Q^3} \\ &= \frac{(2Q'^2 - QQ'')P - 2QQ'P' + Q^2P''}{Q^3}. \end{aligned}$$

Man benötigt für die Berechnung der zweiten Ableitung einer rationalen Kurve also erste und zweite Ableitungen einer polynomialen Kurve, wobei die zweite Ableitung als B-Spline-Kurve

$$X''(t) := \sum_{i=0}^{n-3} d_i'' N_{i,k-2}(t|T'') \tag{1.14}$$

der Ordnung $k-2$ mit Kontrollpunkten

$$d_i'' = \frac{k-2}{t_{i+k-1} - t_{i+1}}(d_{i+1}' - d_i'), \quad i = 0, \ldots, n-3,$$

und einem neuen Knotenvektor $T'' := [t_2, \ldots, t_{n+k-3}]$ geschrieben werden kann.

Implementierung

Für die reine Umsetzung, wie sie oben beschrieben ist, berechnet man die abgeleiteten Kontrollpunkte und Knotenvektoren und setzt diese in den de-Boor-Algorithmus ein. In vielen Fällen ist man aber sowohl an der Auswertung einer polynomialen B-Spline-Kurve, als auch an der ersten und zweiten Ableitung interessiert. Hier gibt der polynomiale de-Boor-Algorithmus 1.10 das Ergebnis gleich mit.

Satz 1.14. *Wurden die neuen Kontrollpunkte $d_j^{(i)}, i = 1, \ldots, k-1, j = r-k+i+1, \ldots, r$, mit $X(t) = d_r^{(k-1)}$ in Algorithmus 1.10 berechnet, gilt:*

$$d_j'^{(i)} = \frac{k-1}{t_{j+k-i} - t_{j+1}}\left(d_{j+1}^{(i)} - d_j^{(i)}\right), \quad j = 0, \ldots, n-2,\ i = 0, \ldots, k-2. \tag{1.15}$$

Beweis: Man bemerke zuerst, dass für den Knotenvektor $T' = [t_0', \ldots, t_{n'+k'-1}'] := [t_1, \ldots, t_{n+k-2}]$ mit $n' := n-1$ und $k' := k-1$ gilt:

$$t_{j'}' = t_j, \quad j' := j-1, \quad 0 \le j' \le n'+k'-1. \tag{1.16}$$

Sei nun $k = 2$. Dann ist laut (1.13):

$$d_j'^{(0)} = d_j' = \frac{1}{t_{j+2} - t_{j+1}}(d_{j+1} - d_j), \quad 0 \le j \le n-2,$$

was identisch zu (1.15) ist. Sei nun $k > 2$, dann führt (1.15) zu

$$d_j'^{(i)} = \frac{k-1}{t_{j+k-i} - t_{j+1}} \left(d_{j+1}^{(i)} - d_j^{(k-2)} \right).$$

Gleichung (1.10) eingesetzt für beide Kontrollpunkte und zusammengefasst ergibt

$$\begin{aligned} d_j'^{(i)} &= \frac{(k-1)(t - t_{j+1})}{(t_{j+k-i} - t_{j+1})(t_{j+1+k-i} - t_{j+1})} d_{j+1}^{(i-1)} + \\ &\left(\frac{(k-1)(t_{j+1+k-i} - t)}{(t_{j+k-i} - t_{j+1})(t_{j+1+k-i} - t_{j+1})} - \frac{(k-1)(t - t_j)}{(t_{j+k-i} - t_{j+1})(t_{j+k-i} - t_j)} \right) d_j^{(i-1)} - \\ &\frac{(k-1)(t_{j+k-i} - t)}{(t_{j+k-i} - t_{j+1})(t_{j+k-i} - t_j)} d_{j-1}^{(i-1)}. \end{aligned} \tag{1.17}$$

Die Auswertung der abgeleiteten Kontrollpunkte ist mit (1.10):

$$d_j'^{(i)} = \frac{t - t_j'}{t_{j+k'-i}' - t'j} d_j'^{(i-1)} + \frac{t_{j+k'-i}' - t_j'}{t_{j+k'-i}' - t'j} d_{j-1}'^{(i-1)}.$$

Wendet man (1.15) und (1.16) auf die beiden Kontrollpunkte auf der rechten Seite an und fasst zusammen, ergibt sich

$$\begin{aligned} d_j'^{(i)} &= \frac{(k-1)(t - t_{j+1})}{(t_{j+k-i} - t_{j+1})(t_{j+1+k-i} - t_{j+1})} d_{j+1}^{(i-1)} + \\ &\left(\frac{(k-1)(t - t_{j+1})}{(t_{j+k-i} - t_{j+1})(t_{j+1+k-i} - t_{j+1})} - \frac{(k-1)(t_{j+k_i} - t)}{(t_{j+k-i} - t_{j+1})(t_{j+k-i} - t_j)} \right) d_j^{(i-1)} - \\ &\frac{(k-1)(t_{j+k-i} - t)}{(t_{j+k-i} - t_{j+1})(t_{j+k-i} - t_j)} d_{j-1}^{(i-1)}. \end{aligned} \tag{1.18}$$

Durch Koeffizientenvergleich ist damit Gleichung (1.17) identisch zu (1.18) und die Behauptung (1.15) ist bewiesen. □

Korollar 1.15. *Wurden die neuen Kontrollpunkte $d_j^{(i)}, i = 1, \ldots, k-1, j = r-k+i+1, \ldots, r$, mit $X(t) = d_r^{(k-1)}$ in Algorithmus 1.10 berechnet, gilt:*

$$X'(t) = d_{r-1}'^{(k-2)} = c \left(d_r^{(k-2)} - d_{r-1}^{(k-2)} \right), \quad c := \frac{k-1}{t_{r+1} - t_r}. \tag{1.19}$$

Zusätzlich gilt:

$$\begin{aligned} X''(t) &= c_1 d_r^{(k-3)} + (c_2 - c_1) d_{r-1}^{(k-3)} + c_2 d_{r-2}^{(k-3)}, \\ c := \frac{k-1}{t_{r+1} - t_r}, &\quad c_1 := \frac{k-2}{t_{r+2} - t_r} c, \quad c_2 := \frac{k-2}{t_{r+1} - t_{r-1}} c. \end{aligned} \tag{1.20}$$

Beweis: Für (1.19) folgt dies sofort aus Satz 1.14. Der Beweis für die zweite Ableitung ist etwas aufwendiger, geht aber analog dazu. □

1.2.3 Numerische Integration

Für die Spline-Approximation (siehe Teil V) benötigt man das Integral über dem Produkt zweier Basis-Splines. Hier bedient man sich der numerischen Integration mittels Quadraturformeln. Das Integral einer einzelnen B-Spline-Funktion oder einer polynomialen B-Spline-Kurve kann man aber auch direkt berechnen (Dierckx [7, S. 9]).

Gauß-Legendre-Quadratur

Es sei eine Funktion f gegeben, zu der das Integral

$$I(f)_a^b := \int_a^b f(x)w(x)dx, \quad x \in [a,b], \tag{1.21}$$

mit einer nicht-negativen Gewichtsfunktion $w(x) \geq 0$ bestimmt werden soll.

Definition 1.16. *Eine* Quadraturformel *zu einer Funktion f mit Knoten c_i und Gewichten $w_i, i = 0,\ldots,n$, ist definiert durch*

$$Q_n(f) := \sum_{i=0}^{n} w_i f(c_i). \tag{1.22}$$

Die Formel hat einen Exaktheitsgrad *m für ein Integral I, falls gilt*

$$Q_n(p) = I(p), \quad p \in \Pi_m.$$

Es wird hier die Gauß-Quadratur benutzt, da diese den maximalen Exaktheitsgrad von $2n+1$ erreicht (Davis und Rabinowitz [4, S. 74]). Zur Vereinfachung werden hier speziell die Legendre-Polynome benutzt, bei denen die Gewichtsfunktion $w \equiv 1$ ist. Da das Integral auf das Intervall $[-1,1]$ eingeschränkt ist, muss man die Werte während der Berechnung noch umformen.

Bemerkung 1.17. Sind die Legendre-Knoten $\widehat{c}_i$ und -Gewichte $\widehat{w}_i, i = 0,\ldots,n$, gegeben, so lassen sich diese durch die Vorschriften

$$c_i = \frac{b-a}{2}\widehat{c}_i + \frac{a+b}{2},$$

$$w_i = \frac{b-a}{2}\widehat{w}_i,$$

auf das Intervall $[a,b] \subset \mathbb{R}$ transformieren.

Man kann so die Quadraturformel $Q_n(f)$ für zugehörige Splines erstellen. Tabelle 1.1 gibt die ersten vier Legendre-Knoten und -Gewichte wieder.

Bemerkung 1.18. Die Quadraturformel für die Polynome der Basis-Splines erreicht nur auf den einzelnen Intervallen den gewünschten Exaktheitsgrad. Daher muss man pro Knotenintervall $[t_i, t_{i+j}]$, mit j als kleinstem Index, bei dem $t_i \neq t_{i+j}$ gilt, die geforderte Gauß-Knotenanzahl auswerten.

$n = 1$	c_i	w_i
$i = 0$	0.0	2.0
$n = 2$	c_i	w_i
$i = 0$	-0.577350269189626	1.0
$i = 1$	0.577350269189626	1.0
$n = 3$	c_i	w_i
$i = 0$	-0.774596669241483	0.555555555555556
$i = 1$	0.0	0.888888888888889
$i = 2$	0.774596669241483	0.555555555555556
$n = 4$	c_i	w_i
$i = 0$	-0.861136311594053	0.347854845137454
$i = 1$	-0.339981043584856	0.652145154862546
$i = 2$	0.339981043584856	0.652145154862546
$i = 3$	0.861136311594053	0.347854845137454

Tabelle 1.1: Gauß-Legendre-Knoten und -Gewichte [1, S. 916]

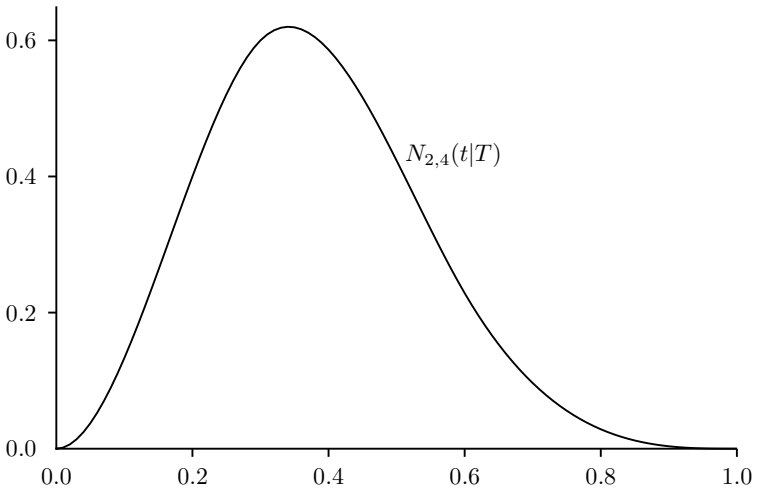


Abbildung 1.3: Basis-Spline $N_{2,4}$

Beispiel 1.19. Es soll das Integral von $N_{2,4}(t)$ über dem Knotenvektor $T = [0,0,0,0,0.3,0.6,1,1,1,1]$ berechnet werden (Abbildung 1.3). Das exakte Integral ist gegeben als

$$\int_T N_{2,4}(t)dt = \int_0^1 N_{2,4}(t)dt = \frac{t_6 - t_2}{4} = \frac{1}{4}.$$

Mittels Gauß-Legendre-Quadratur benötigt man pro Knotenintervall $I_1 := [0,0.3]$, $I_2 := [0.3,0.6]$ und $I_3 := [0.6,1]$ je zwei Knoten und Gewichte, da diese bis zu einem Grad von 3 (Ordnung 4) exakt integrieren. Es ist

$$\begin{aligned}\int_T N_{2,4}(t)dt &= \int_0^1 N_{2,4}(t)dt \approx \sum_{j=1}^{3} a_j \sum_{i=0}^{1} w_i N_{2,4}(a_j c_i + b_j) \\ &= 0.15\sum_{i=0}^{1} w_i N_{2,4}(0.15c_i + 0.15) + 0.15\sum_{i=0}^{1} w_i N_{2,4}(0.15c_i + 0.45) + \\ &\quad 0.2\sum_{i=0}^{1} w_i N_{2,4}(0.2c_i + 0.8)\end{aligned}$$

mit $a_j := \frac{I_{j,2}-I_{j,1}}{2}$, $b_j := \frac{I_{j,2}+I_{j,1}}{2}$ und $w_0 = w_1 = 1.0$, $-c_0 = c_1 = 0.57735026919$. Daraus folgt

$$\begin{aligned}\int_T N_{2,4}(t)dt &\approx 0.15(N_{2,4}(0.063) + N_{2,4}(0.237)) + 0.15(N_{2,4}(0.363) + N_{2,4}(0.537)) \\ &\quad +0.2(N_{2,4}(0.685) + N_{2,4}(0.915)) \\ &= 0.15(0.058 + 0.492) + 0.15(0.615 + 0.349) + 0.2(0.112 + 0.002) \\ &= 0.25,\end{aligned}$$

wie gewünscht.

1.2.4 Länge einer B-Spline-Kurve

In manchen Fällen benötigt man eine möglichst äquidistante Abtastung der B-Spline-Kurven. Hierzu wäre es am praktischsten, wenn die Kurven nach der Bogenlänge parametriert sind.

Definition 1.20. (aus Farin [10, S. 180]) *Es sei eine stetig differenzierbare Funktion $f : [a,b] \to \mathbb{R}^l, l \geq 2$, gegeben. Dann beschreibt die Funktion*

$$s_f(u) := \int_a^u \|f'\|\, du, \quad u \in [a,b], \tag{1.23}$$

die Bogenlänge von f an der Stelle u.

Es kann gezeigt werden, dass bei B-Spline-Kurven nur eine Gerade so parametriert werden kann, dass eine gleichmäßige Abtastung im Parameterbereich zu einer gleichmäßigen Abtastung im Bildbereich führt (Farin [10, S. 222 f]). Aus diesem Grund müsste man s_X für eine allgemeine B-Spline-Kurve X explizit berechnen, könnte dann aber über deren Umkehrfunktion s_X^{-1} eine äquidistante Abtastung im Bildbereich durchführen.

Im Allgemeinen lässt sich s_X nicht exakt bestimmen. Eine numerische Integration mittels Gauß-Quadratur (siehe Abschnitt 1.2.3) ist aber möglich (siehe Casciola und Morigi [3]). Floater und Rasmussen [15] benutzen ein punktbasierendes Verfahren, um eine allgemeine Funktion f als Polynom an gewissen Stellen zu interpolieren und dann eine Quadraturformel anzuwenden. In dieser Arbeit wird ein „Table Lookup Scheme“ benutzt. Dabei versucht man die Bogenlänge durch die Sehnenlänge anzunähern.

Definition 1.21. *Es sei eine stetige Funktion $f : [a,b] \to \mathbb{R}^l, l \geq 2$, und Parameterwerte $P := \{p_i \mid p_i := a + ih,\ h := \frac{b-a}{m-1}, i = 0,\dots,m-1\}$ gegeben. Dann bezeichnet*

$$S_f(p_i) := \begin{cases} 0, & i = 0, \\ \sum_{j=1}^{i} \|f(p_i) - f(p_{i-1})\|, & i = 1,\dots,m-1, \end{cases} \tag{1.24}$$

die Sehnenlänge von f an der Stelle p_i.

Für $m \to \infty$ ist $h \to 0$ und die Sehnenlänge konvergiert gegen die Bogenlänge (siehe Farin [10, S. 180]). Eine Idee ist es, die Sehnenlänge für hinreichend großes m zu berechnen und linear zwischen zwei abgetasteten Punkten zu interpolieren (siehe Walter und Fournier [37, S. 144 f]). In dieser Arbeit wird die Umkehrfunktion S_f^{-1} durch eine eindimensionale B-Spline-Kurve approximiert.

Es seien eine stetige Funktion $f : [a,b] \to \mathbb{R}^l, l \geq 2$, Parameterwerte

$$P := \left\{ p_i \mid p_i := a + ih,\ h := \frac{b-a}{m-1},\ i = 0,\dots,m-1 \right\}$$

und ausgewertete Punkte

$$V := \{v_i \mid v_i := f(p_i),\ i = 0,\dots,m-1\},$$

gegeben. Die eindimensionale B-Spline-Kurve S_f^* soll die Vektoren P mit Parameterwerten

$$U := \left\{ u_i \mid u_i := S_f(p_i) \frac{p_{m-1} - p_0}{S_f(p_{m-1}) - S_f(p_0)},\ i = 0,\dots,m-1 \right\}$$

approximieren. Es ist offensichtlich, dass S_f^* die Umkehrfunktion S_f^{-1} der Sehnenlänge approximiert. Es ist nach Teil V, Abschnitt 1

$$\max_{i=0}^{m-1} \|S_f^*(u_i) - p_i\| \leq \epsilon$$

und damit

$$\lim_{m\to\infty} \max_{u\in[a,b]} |S_f^*(u) - S_f^{-1}(u)| \leq \epsilon \tag{1.25}$$

beziehungsweise

$$\lim_{m\to\infty} \max_{u\in[a,b]} |S_f^*(u) - s_f^{-1}(u)| \leq \epsilon. \tag{1.26}$$

Korollar 1.22. *Es seien eine stetige Funktion $f : [a,b] \to \mathbb{R}^l, l \geq 2$, und S_f^* für hinreichend großes m gegeben. Dann gilt:*

$$|u_2 - u_1| = |u_3 - u_2| \Rightarrow \|f(S_f^*(u_2)) - f(S_f^*(u_1))\| \approx \|f(S_f^*(u_3)) - f(S_f^*(u_2))\|, \quad (1.27)$$

$$u_1, u_2, u_3 \in [a,b].$$

1.2.5 Knoteneinfügen

Das Knoteneinfügen hat man bereits bei der Auswertung einer B-Spline-Kurve in Abschnitt 1.2.1 gesehen. Dort wird wiederholt ein Knoten (der Parameterwert, an dem man die Kurve auswerten möchte) eingefügt, bis er die Vielfachheit $k-1$ hat und so der entstandene Kontrollpunkt auf der Kurve liegt. Das Knoteneinfügen ist also ein Schritt des de-Boor-Algorithmus.

Satz 1.23. (aus Piegl und Tiller [27, S. 142 ff]) *Es seien eine B-Spline-Kurve X wie in (1.7), ein Knoten $t^* \in (t_0, t_{n+k-1})$ mit $t_r \leq t^* < t_{r+1}$ und der durch Einfügen entstandene Knotenvektor $T^* := [t_0^* = t_0, \ldots, t_r^* = t_r, t_{r+1}^* = t^*, t_{r+2}^* = t_{r+1}, \ldots, t_{n+k}^* = t_{n+k-1}]$ gegeben. Dann existiert eine B-Spline-Kurve $X^*(t)$ mit*

$$X(t) = X^*(t) := \frac{\sum_{i=0}^{n} w_i^* d_i^* N_{i,k}(t|T^*)}{\sum_{i=0}^{n} w_i^* N_{i,k}(t|T^*)}, \quad t \in [t_0, t_{n+k-1}], \quad (1.28)$$

$$\begin{aligned} w_i^* &:= \alpha_i w_i + (1-\alpha) w_{i-1}, \\ d_i^* &:= \frac{\alpha_i w_i d_i + (1-\alpha) w_{i-1} d_{i-1}}{w_i^*}, \end{aligned} \quad (1.29)$$

$$\alpha_i := \begin{cases} 1, & 0 \leq i \leq r-k+1, \\ \frac{t^* - t_i}{t_{i+k-1} - t_i}, & r-k+2 \leq i \leq r, \\ 0, & r+1 \leq n. \end{cases}$$

Man muss beim Knoteneinfügen also nur $k-1$ neue Kontrollpunkte berechnen.

Algorithmus 1.24. (Knoteneinfügen, nach Piegl und Tiller [27, S. 141 ff])

Eingabe: B-Spline-Kurve X, neuer Knoten t^* mit $t_{k-1} < t^* < t_n$
Ausgabe: B-Spline-Kurve X^* mit eingefügtem Knoten und $X^*(t) = X(t)$

1. Suche einen Index r mit $t_r \leq t^* < t_{r+1}$.
2. Füge t^* an der Stelle $r+1$ in T ein.
3. Schleife i von 0 bis $r-k+1$:
   - Setze $w_i^* := w_i$ und $d_i^* := d_i$.

4. Schleife i von $r-k+2$ bis r:
   - Setze $\alpha_i := \frac{t^*-t_i}{t_{i+k}-t_i}$.
   - Berechne
   $$w_i^* := \alpha_i w_i + (1-\alpha_i)w_{i-1}, \quad d_i^* := \frac{\alpha_i w_i d_i + (1-\alpha_i)w_{i-1}d_{i-1}}{w_i^*}.$$
5. Schleife i von $r+1$ bis n:
   - Setze $w_i^* := w_{i-1}$ und $d_i^* := d_{i-1}$.
6. Erstelle aus $d_i^*, w_i^*, i = 0,\ldots,n$, eine neue B-Spline-Kurve X^*.

Beispiel 1.25. Es sei der Knotenvektor $T := [0,0,0,0,2,2,2,2]$ gegeben mit Kontrollpunkten d_0, d_1, d_2, d_3. Es soll ein Knoten $t^* := 1.0$ eingefügt werden. Damit ist $k = 4$ und $r = 3$. Es müssen folgende Gleichungen berechnet werden:

$$\begin{aligned} d_1^* &= \alpha_1 d_1 + (1-\alpha_1)d_0 \\ d_2^* &= \alpha_2 d_2 + (1-\alpha_2)d_1 \\ d_3^* &= \alpha_3 d_3 + (1-\alpha_3)d_2 \end{aligned}$$

In diesem Fall ist $\alpha_1 = \alpha_2 = \alpha_3 = 0.5$ und die neuen Kontrollpunkte sind $(d_0^*, d_1^*, d_2^*, d_3^*, d_4^*) := (d_0, 0.5(d_0+d_1), 0.5(d_1+d_2), 0.5(d_2+d_3), d_3)$.

Obige Regel (1.29) lässt sich für polynomiale B-Spline-Kurven auch als Matrixmultiplikation $d^* = A(t^*)d$ mit

$$A(t^*) = \begin{pmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ & & 1-\alpha_{r-k+2} & \alpha_{r-k+2} & & & & \\ & & & \ddots & \ddots & & & \\ & & & & 1-\alpha_r & \alpha_r & & \\ & & & & & 1 & & \\ & & & & & & \ddots & \\ & & & & & & & 1 \end{pmatrix} \in \mathbb{R}^{n+1\times n} \tag{1.30}$$

und $d := (d_0,\ldots,d_{n-1})^T \in \mathbb{R}^{n\times 3}, d^* := (d_0^*,\ldots,d_n^*)^T \in \mathbb{R}^{n+1\times 3}$ schreiben (Sauer [28, S. 62]). Dadurch reicht es, beim Einfügen mehrerer Knoten $t_0^*,\ldots,t_s^*$ die Kontrollpunkte d mit der Matrix

$$A(t_0^*,\ldots,t_s^*) := A(t_s^*)\cdots A(t_0^*)$$

zu multiplizieren. Es ist dann:

$$d^* := A(t_0^*,\ldots,t_s^*)d.$$

Da die Matrizen $A(t_s^*), \ldots, A(t_0^*)$ bandiert und dünn besetzt sind, lassen sich diese schneller berechnen als das wiederholte Anwenden von Algorithmus 1.24.

Für rationale B-Spline-Kurven muss man ein Zweischrittverfahren benutzen, bei dem im ersten Schritt $w^* = A(t^*)w$ mit der Matrix $A(t^*)$ aus (1.30) berechnet wird und danach $d^* = A'(t^*)d$ mit

$$A'(t^*) = \begin{pmatrix} 1 & & & & & & & & \\ & \ddots & & & & & & & \\ & & 1 & & & & & & \\ & & \frac{w_{r-k+1}(1-\alpha_{r-k+2})}{w^*_{r-k+2}} & \frac{w_{r-k+2}\alpha_{r-k+2}}{w^*_{r-k+2}} & & & & & \\ & & & \ddots & \ddots & & & & \\ & & & & \frac{w_{r-1}(1-\alpha_r)}{w^*_r} & \frac{w_r\alpha_r}{w^*_r} & & & \\ & & & & & 1 & & & \\ & & & & & & \ddots & & \\ & & & & & & & 1 & \end{pmatrix}. \tag{1.31}$$

Da die Matrix $A'(t^*)$ jetzt von den Gewichten abhängt, ist ein wiederholtes Knoteneinfügen mit einer Gesamtmatrix zwar möglich, die einzelnen Ergebnisse $w^* = A(t^*)w$ müssen aber zwischengespeichert werden.

Beispiel 1.26. Es seien die Daten wie in Beispiel 1.25 gegeben. Dann ist

$$A(t^*) = \begin{pmatrix} 1 & & & & \\ 1-\alpha_1 & \alpha_1 & & & \\ & 1-\alpha_2 & \alpha_2 & & \\ & & 1-\alpha_3 & \alpha_3 & \\ & & & 1 & \end{pmatrix}$$

und die neuen Kontrollpunkte $d_0^*, d_1^*, d_2^*, d_3^*, d_4^*$ ergeben sich aus der Matrixmultiplikation $A(t^*)d = d^*$.

1.2.6 Knotenentfernen

Eingefügte Knoten können auch wieder entfernt werden. Dies ist zum Beispiel bei der Graderhöhung eines Splines wichtig (siehe Abschnitt 1.2.7). Der Nachteil des Algorithmus ist, dass man nicht von vornherein weiß, ob ein Knoten entfernt werden kann. Man nimmt dies daher an und versucht die Gleichungen zum Knoteneinfügen rückwärts aufzulösen. Erhält man am Ende den gleichen Kontrollpunkt wie den bereits vorhandenen, kann der Knoten entfernt werden.

Definition 1.27. *Es sei eine B-Spline-Kurve X wie in (1.7) gegeben. Es soll der Knoten $t_j \in T, t_{j-1} \leq t_j < t_j+1, k \leq j \leq n$, mit Vielfachheit s entfernt werden. (Äußere Knoten werden also*

*nicht entfernt.) Der Knoten t_j heißt r-*mal entfernbar, *$r \leq s$, falls es eine B-Spline-Kurve*

$$X^*(t) := \frac{\sum_{i=0}^{n-1-r} w_i^* d_i^* N_{i,k}(t|T^*)}{\sum_{i=0}^{n-1-r} w_i^* N_{i,k}(t|T^*)}$$

mit Knotenvektor $T^* := T \backslash \{t_{j-r+1}, \ldots, t_j\}$ und $X^*(t) = X(t), t \in [t_0, t_{n+k-1}]$, gibt.

Lemma 1.28. *Sei eine polynomiale B-Spline-Kurve X wie in (1.8) gegeben. Es soll ein Knoten $t_j \in T, t_{j-1} \leq t_j < t_{j+1}, k \leq j \leq n$, entfernt werden. Dann existieren genau dann eine Matrix*

$$A(t_j) = \begin{pmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ & & 1-\alpha_{j-k+1} & \alpha_{j-k+1} & & & & \\ & & & \ddots & \ddots & & & \\ & & & & 1-\alpha_{j-1} & \alpha_{j-1} & & \\ & & & & & 1 & & \\ & & & & & & \ddots & \\ & & & & & & & 1 \end{pmatrix} \tag{1.32}$$

und Kontrollpunkte $d^* := (d_0^*, \ldots, d_{n-2}^*)^T$ mit $A(t_j)d^* = d, d := (d_0, \ldots, d_{n-1})^T$, wenn t_j (mindestens) 1-mal entfernbar ist.

Beweis: Sei t_j 1-mal entfernbar. Dann existiert nach Definition 1.27 eine B-Spline-Kurve X^* mit Kontrollpunkten $d_0^*, \ldots, d_{n-2}^*$ und Knotenvektor $T^* := [t_0^* = t_0, \ldots, t_{j-1}^* = t_{j-1}, t_j^* = t_{j+1}, \ldots, t_{n+k-2}^* = t_{n+k-1}]$, wobei $T := [t_0, \ldots, t_j, \ldots, t_{n+k-1}]$ der Knotenvektor von X ist, mit $X^*(t) = X(t), t \in [t_0, t_{n+k-1}]$. Fügt man den Knoten t_j wieder in T^* an der Stelle $j-1$ ein, da $t_{j-1}^* \leq t_j < t_j^*$, ergibt sich Matrix $A(t_j)$ wie in (1.32). Damit ist die Rückrichtung bewiesen. Existieren umgekehrt eine B-Spline-Kurve X^* mit Kontrollpunkten $d_0^*, \ldots, d_{n-2}^*$ und der Matrix aus (1.32) mit $A(t_j)d^* = d$, dann gilt nach Satz 1.23 $X^*(t) = X(t), t \in [t_0, \ldots, t_{n+k-1}]$. Per Definition 1.27 heißt das, dass t_j 1-mal entfernbar ist. □

Lemma 1.28 bedeutet, dass man die Matrix $A(t_j)$ zu einem Knoten $t_j \in T, t_{j-1} \leq t_j < t_{j+1}, k \leq j \leq n$, erstellt. Existieren Kontrollpunkte d^* aus $A(t_j)d^* = d$, kann der Knoten entfernt werden. Für rationale B-Spline-Kurven mit der Matrix aus (1.31) geht die Argumentation ähnlich.

Beispiel 1.29. Man betrachte obiges Beispiel 1.26, nachdem der Knoten $t^* := 1.0$ eingefügt wurde. Es sei also $T := [0,0,0,0,1,2,2,2,2]$ gegeben mit Kontrollpunkten d_0, d_1, d_2, d_3, d_4 und der Knoten $t_j = 1.0, j = 4$, soll entfernt werden. Die Ordnung ist wieder $k = 4$. Man entfernt t_4 aus T und erhält $T^* := [0,0,0,0,2,2,2,2]$. Die Matrix $A(t_4)$ aus diesem neuen Knotenvektor T^* ist somit

$$A(t_j) = \begin{pmatrix} 1 & & & \\ 1-\alpha_1 & \alpha_1 & & \\ & 1-\alpha_2 & \alpha_2 & \\ & & 1-\alpha_3 & \alpha_3 \\ & & & 1 \end{pmatrix}.$$

Erstellt man ein Gleichungssystem $A(t_j)d^* = d$ mit unbekannten Kontrollpunkten $d^* := (d_0^*, d_1^*, d_2^*, d_3^*)$, ergibt sich sofort $d_0^* = d_0$ und $d_3^* = d_4$. Dies kann man in die zweite und vorletzte Zeile einsetzen und erhält daraus

$$d_1^* = \frac{d_1 - (1-\alpha_1)d_0}{\alpha_1}, \quad d_2^* = \frac{d_3 - \alpha_3 d_4}{1-\alpha_3}.$$

Für die mittlere Zeile ergibt sich somit:

$$\begin{aligned} d_2 &= \alpha_2 d_2^* + (1-\alpha_2)d_1^* \\ &= \alpha_2 \frac{d_3 - \alpha_3 d_4}{1-\alpha_3} + (1-\alpha_2)\frac{d_1 - (1-\alpha_1)d_0}{\alpha_1} \\ &= d_3 - 0.5d_4 + d_1 - 0.5d_0 \end{aligned} \tag{1.33}$$

Ist Gleichung (1.33) erfüllt, kann der Knoten t_4 entfernt werden. (Dies ist in diesem Beispiel natürlich gegeben, wie man leicht überprüfen kann, indem man die Ergebnisse aus Beispiel 1.25 einfügt.)

Bemerkung 1.30. Der Algorithmus arbeitet aufgrund der rationalen B-Spline-Kurven nur indirekt mit der Matrixdarstellung aus (1.30) beziehungsweise (1.31), indem die Gleichungen einzeln aufgelöst werden. Es ergeben sich dabei folgende Betrachtungen:

1. Ist die Ordnung k gerade, hat man eine ungerade Anzahl von Gleichungen, wobei die mittlere zur Überprüfung dient. Ist k ungerade, hat man eine gerade Anzahl von Gleichungen und die beiden mittleren müssen berechnet und verglichen werden.

2. Es kann vorkommen, dass der Koeffizient $c_i = 0$ oder $c_i = 1$ ist. In diesem Fall hält man bei der jeweiligen Gleichung an und ersetzt sukzessive die restlichen Gleichungen von der anderen Seite her, so dass man am Ende nur noch eine oder zwei Gleichungen zur Überprüfung übrig hat.

Algorithmus 1.31. (Knotenentfernen, nach Piegl und Tiller [27, S. 179 ff])

Eingabe: B-Spline-Kurve X, zu entfernender Knoten $t^* \in T$ mit $t_{k-1} < t^* < t_n$, Genauigkeit ϵ

Ausgabe: neue B-Spline-Kurve $X^*(t)$ ohne t^*

1. Suche einen Index r mit $t_{r+1} \leq t^* < t_{r+2}$. Dies ist die Position, an der der Knoten vorher hätte eingefügt werden müssen.

2. Setze Zähler $i := r + 2 - k$ und $j := r$.

3. Reserviere Platz für k neue Kontrollpunkte $\widehat{d}_i$ und Gewichte $\widehat{w}_i$.

4. Setze $\widehat{d}_0 := d_{i-1}$, $\widehat{d}_{k-1} := d_{j+1}$, $\widehat{w}_0 := w_{i-1}$, $\widehat{w}_{k-1} := w_{j+1}$.

5. Solange $i + 1 < j$:

   - Setze $c_i := \frac{t^* - t_i}{t_{i+k} - t_i}$ und $c_j := \frac{t^* - t_j}{t_{j+k} - t_j}$.

- Ist $c_i = 0$, setze $i := i - 1$,
  sonst:
  - Setze $l := i - (r + 2 - k) + 1$.
  - Berechne $\widehat{w}_l := \frac{w_i - (1-c_i)\widehat{w}_{l-1}}{c_i}$.
  - Neuer Kontrollpunkt $\widehat{d}_l := \frac{w_i d_i - (1-c_i)\widehat{w}_{l-1}\widehat{d}_{l-1}}{c_i \widehat{w}_l}$.
- Ist $c_j = 1$, setze $j := j + 1$,
  sonst:
  - Setze $l := j - (r + 2 - k)$.
  - Berechne $\widehat{w}_l := \frac{w_j - c_j \widehat{w}_{l+1}}{1-c_j}$.
  - Neuer Kontrollpunkt $\widehat{d}_l := \frac{w_j d_j - c_j \widehat{w}_{l+1}\widehat{d}_{l+1}}{(1-c_j)\widehat{w}_l}$.
- Setze $i := i + 1, j := j - 1$.

6. Ist $i = j$:

   - Setze $l := i - (r + 2 - k) + 1$.
   - Setze $c_i := \frac{t^* - t_i}{t_{i+k} - t_i}$.
   - Berechne $\widehat{w} := (1 - c_i)\widehat{w}_{i-1} + c_i \widehat{w}_i$.
   - Berechne $\widehat{d} := \frac{(1-c_i)\widehat{w}_{i-1}\widehat{d}_{i-1} + c_i \widehat{w}_i \widehat{d}_i}{\widehat{w}}$.
   - Ist $|w_i - \widehat{w}| < \epsilon$ und $\|d_i - \widehat{d}\| < \epsilon$, dann kann t^* entfernt werden.

   sonst ($i + 1 = j$):

   - Setze $c_i := \frac{t^* - t_i}{t_{i+k} - t_i}$ und $c_j := \frac{t^* - t_j}{t_{j+k} - t_j}$
   - Ist $c_i \neq 0$ und $c_j \neq 1$:
     - Setze $l := i - (r + 2 - k) + 1$.
     - Berechne $\widehat{w}_l := \frac{w_i - (1-c_i)\widehat{w}_{l-1}}{c_i}$.
     - Neuer Kontrollpunkt $\widehat{d}_l := \frac{w_i d_i - (1-c_i)\widehat{w}_{l-1}\widehat{d}_{l-1}}{c_i \widehat{w}_l}$.
     - Setze $l := j - (r + 2 - k)$.
     - Berechne $\widehat{w} := \frac{w_j - c_j \widehat{w}_{l+1}}{1-c_j}$.
     - Berechne $\widehat{d} := \frac{w_j d_j - c_j \widehat{w}_{l+1}\widehat{d}_{l+1}}{(1-c_j)\widehat{w}_l}$.
     - Ist $|\widehat{w} - \widehat{w}_l| < \epsilon$ und $\|\widehat{d} - \widehat{d}_l\| < \epsilon$, dann kann t^* entfernt werden.
   - Ist sonst $c_j = 1$:
     - Setze $l := i - (r + 2 - k) + 1$.
     - Berechne $\widehat{w}_l := \frac{w_i - (1-c_i)\widehat{w}_{l-1}}{c_i}$.
     - Neuer Kontrollpunkt $\widehat{d}_l := \frac{w_i d_i - (1-c_i)\widehat{w}_{l-1}\widehat{d}_{l-1}}{c_i \widehat{w}_l}$.
     - Und t^* kann entfernt werden.
   - Ist sonst $c_i = 0$:

- Setze $l := j - (r + 2 - k)$.
- Berechne $\widehat{w}_l := \frac{w_j - c_j \widehat{w}_{l+1}}{1 - c_j}$.
- Neuer Kontrollpunkt $\widehat{d}_l := \frac{w_j d_j - c_j \widehat{w}_{l+1} \widehat{d}_{l+1}}{(1-c_j)\widehat{w}_l}$.
- Und t^* kann entfernt werden.

7. Wenn t^* entfernt werden kann:
   - Lösche d_i und w_i.
   - Setze $d_{r+2-k+j-1} := \widehat{d}_j$ und $w_{r+2-k+j-1} := \widehat{w}_j$ für $j = 1, \ldots, k-2$.
   - Lösche den Knoten $t_{r+1} = t^*$.

1.2.7 Graderhöhung

Definition 1.32. *Sei eine B-Spline-Kurve X wie in (1.7) mit Ordnung k und Knotenvektor*

$$T := [\underbrace{a, \ldots, a}_{k}, \underbrace{u_1, \ldots, u_1}_{v_1}, \ldots, \underbrace{u_s, \ldots, u_s}_{v_s}, \underbrace{b \ldots, b}_{k}] \tag{1.34}$$

gegeben. (Die inneren Knoten u_j haben also Vielfachheit $v_j, j = 1, \ldots, s$.) Die Graderhöhung *bezeichnet die Erstellung einer B-Spline-Kurve*

$$X^*(t) := \frac{\sum_{i=0}^{n+s} w_i^* d_i^* N_{i,k+1}(t|T^*)}{\sum_{i=0}^{n+s} w_i^* N_{i,k+1}(t|T^*)} \tag{1.35}$$

der Ordnung $k+1$ auf dem Knotenvektor

$$T^* := [\underbrace{a, \ldots, a}_{k+1}, \underbrace{u_1, \ldots, u_1}_{v_1+1}, \ldots, \underbrace{u_s, \ldots, u_s}_{v_s+1}, \underbrace{b \ldots, b}_{k+1}]$$

mit $X(t) = X^*(t)$, $t \in [a, b]$.

Es gibt verschiedene Ansätze zur Berechnung der $w_i^*, d_i^*, i = 0, \ldots, n+s$. Hier wird der Algorithmus von Piegl und Tiller [27, S. 201 ff] benutzt, bei dem die Graderhöhung von B-Spline-Kurven auf die Graderhöhung von Bézier-Splines zurückgeführt wird. Es werden zuerst solange Knoten eingefügt, bis man segmentweise Bézier-Splines erhält, diese werden graderhöht und danach werden die Knoten wieder entfernt.

Lemma 1.33. (aus Piegl und Tiller [27, S. 203 f]) *Sei eine Bézier-Kurve C wie in (1.4) gegeben. Dann existiert eine graderhöhte Kurve*

$$C^*(t) := \frac{\sum_{i=0}^{n+1} w_i^* d_i^* B_{i,n+1}(t)}{\sum_{i=0}^{n+1} w_i^* B_{i,n+1}(t)}, \quad t \in [0,1],$$

mit

$$w_i^* := (1 - \alpha_i) w_i + \alpha_i w_{i-1}, \quad d_i^* := \frac{(1-\alpha_i) w_i d_i + \alpha_i w_{i-1} d_{i-1}}{w_i^*}, \quad \alpha_i := \frac{i}{n+1}, \tag{1.36}$$

für $i = 0, \ldots, n+1$, und $C^*(t) = C(t)$, $t \in [0,1]$.

Algorithmus 1.34. (Graderhöhung, nach Piegl und Tiller [27, S. 188 ff])
Eingabe: B-Spline-Kurve X der Ordnung k mit Knotenvektor T wie in (1.34)
Ausgabe: B-Spline-Kurve $X^*(t)$ der Ordnung $k+1$ mit $X(t) = X^*(t)$

1. Füge jeden Knoten $u_j, j = 1, \ldots, s$, gemäß Algorithmus 1.24 $(k-1-v_j)$-mal in $\widehat{T} := T$ ein, so dass jeder innere Knoten die Vielfachheit $k-1$ hat. Bezeichne $\widehat{d}_i$ die neuen Kontrollpunkte und $\widehat{w}_i, i = 0, \ldots, \widehat{n}-1$, die neuen Gewichte.

2. Schleife j von 1 bis s:

   - Suche Position r von u_j in $\widehat{T}$ mit $\widehat{t}_r \leq u_j < \widehat{t}_{r+1}$.
   - Setze $r := r+1-k$ als Startpunkt für die Iteration.
   - Schleife i von 0 bis k:
     - Setze $\alpha_i := \frac{i}{k}$.
     - Berechne

$$\begin{aligned} w^*_{i+r+j-1} &:= (1-\alpha_i)\widehat{w}_{i+r} + \alpha_i \widehat{w}_{i+r-1}, \\ d^*_{i+r+j-1} &:= \frac{(1-\alpha_i)\widehat{w}_{i+r}\widehat{d}_{i+r} + \alpha_i \widehat{w}_{i+r-1}\widehat{d}_{i+r-1}}{w^*_{i+r+j-1}}. \end{aligned}$$

3. Erstelle aus Knotenvektor

$$T^* := [\underbrace{a, \ldots, a}_{k+1}, \underbrace{u_1, \ldots, u_1}_{k}, \ldots, \underbrace{u_s, \ldots, u_s}_{k}, \underbrace{b \ldots, b}_{k+1}],$$

   Kontrollpunkten d^*_i und Gewichten $w^*_i, i = 0, \ldots, \widehat{n}+s+1$ eine neue B-Spline-Kurve $X^*(t)$ der Ordnung $k+1$.

4. Entferne jeden Knoten $u_j, j = 1, \ldots, s$, gemäß Algorithmus 1.31 $(k-1-v_j)$-mal aus T^*, so dass jeder innere Knoten die Vielfachheit v_j+1 hat.

1.2.8 Kurventrimmung

Ist eine B-Spline-Kurve X gegeben, so kann man die Kurve auf das Intervall $[a_1, a_2] \subset [t_0, \ldots, t_{n+k-1}]$ einschränken. Diesen Vorgang nennt man *Trimmung der Kurve*.

Lemma 1.35. *Es seien eine B-Spline-Kurve X wie in (1.7) und ein Intervall $[a_1, a_2] \subset [t_0, \ldots, t_{n+k-1}], a_1 < a_2$, gegeben. Dann existiert eine B-Spline-Kurve X^* mit*

$$X|_{[a_1,a_2]} = X^*.$$

Beweis: Ist $a_i = t_j \in T$ für ein $j = 0, \ldots, n+k-1$, setze $m_i := k-1-v_i, i = 1, 2$, wobei v_i die Vielfachheit von t_j ist. Ansonsten setze $m_i := k-1$. Ist $a_1 = t_0$ beziehungsweise $a_2 = t_{n+k-1}$,

fügt man diesen Knoten nicht mehr ein, da bereits die maximale Vielfachheit erreicht ist. Fügt man a_i jeweils m_i-mal in X ein, entsteht nach Satz 1.23 eine neue B-Spline-Kurve $\widehat{X}$ mit $\widehat{X}(t) = X(t), t \in [a_1, a_2]$. Sei der Knotenvektor von $\widehat{X}$ als

$$\widehat{T} := \left[\widehat{t}_0, \ldots, \widehat{t}_{j_1} = a_1, \ldots, \widehat{t}_{j_1+k-2} = a_1, \ldots, \widehat{t}_{j_2} = a_2, \ldots, \widehat{t}_{j_2+k-2} = a_2, \ldots, \widehat{t}_{m_1+m_2+n+k-1}\right]$$

gegeben. Da beide eingefügte Knoten Vielfachheit $k-1$ haben, ist $\widehat{X}(a_1) = \widehat{d}_{j_1+1}$ und $\widehat{X}(a_2) = \widehat{d}_{j_2+1}$, wobei $\widehat{d}_i, i = 0, \ldots, m_1 + m_2 + n - 1$, die Kontrollpunkte von $\widehat{X}$ sind. Nun erstellt man aus den Kontrollpunkten $\widehat{d}_{j_1+1}, \ldots, \widehat{d}_{j_2+1}$, Gewichten $\widehat{w}_{j_1+1}, \ldots, \widehat{w}_{j_2+1}$ und Knotenvektor $T^* := [a_1, \widehat{t}_{j_1} = a_1, \ldots, \widehat{t}_{j_1+k-2} = a_1, \ldots, \widehat{t}_{j_2} = a_2, \ldots, \widehat{t}_{j_2+k-2} = a_2, a_2]$ eine neue B-Spline-Kurve $X^*(t)$ der Ordnung k. Da X^* auf dem Intervall $[a_1, a_2]$ die gleichen Knoten, Kontrollpunkte und Gewichte wie $\widehat{X}$ hat, folgt

$$X|_{[a_1,a_2]} = \widehat{X}|_{[a_1,a_2]} = X^*.$$

□

Algorithmus 1.36. (Kurventrimmung)

Eingabe: B-Spline-Kurve X wie in (1.7), Intervall $[a_1, a_2] \subset [t_0, \ldots, t_{n+k-1}]$
Ausgabe: getrimmte B-Spline-Kurve X^*

1. Ist $a_i = t_j \in T$, setze $m_i := k - 1 - v_i, i = 1, 2$, wobei v_i die Vielfachheit von t_j ist. Ansonsten setze $m_i := k - 1$.

2. Füge a_i gemäß Algorithmus 1.24 $m - i$-mal in X ein und erstelle so eine neue B-Spline-Kurve $\widehat{X}$.

3. Entferne alle Knoten, Kontrollpunkte und Gewichte gemäß Beweis von Satz 1.35.

1.3 Zusammengesetzte Kurven

Zusammengesetzte Kurven sind eine geordnete Liste von B-Spline-Kurven, bei denen der Endpunkt der einen Kurve identisch mit dem Startpunkt der nächsten ist. In den meisten Fällen sind zusammengesetzte Kurven geschlossen, das heißt der Endpunkt der letzten Kurve ist identisch mit dem Startpunkt der ersten Kurve.

Definition 1.37. *Es sei eine B-Spline-Kurve X wie in (1.7) gegeben. Die Kurve heißt* geschlossen, *falls $X(t_0) = X(t_{n+k+1})$. Da in dieser Arbeit nur Kurven mit der Endpunktinterpolationseigenschaft behandelt werden, bedeutet das, dass $d_0 = d_{n-1}$. Der Punkt d_0 heißt* Schließpunkt.

Bemerkung 1.38. Es ist anzumerken, dass geschlossen nicht periodisch impliziert.

Definition 1.39. *Es seien B-Spline-Kurven $C^{(j)}, j = 0, \ldots, m-1$, mit jeweils $n^{(j)}$ Kontrollpunkten $d_i^{(j)}$ gegeben. (Die Gewichte sind aufgrund der Endpunktinterpolation nicht wichtig.) Die geordnete Menge*

$$\mathfrak{C} := \{C^{(j)} \mid j = 0, \ldots, m-1\} \tag{1.37}$$

heißt zusammengesetzte B-Spline-Kurve*, falls*

$$d^{(j-1)}_{n^{(j-1)}-1} = d^{(j)}_0, \quad j = 1, \ldots, m-1.$$

Ist zusätzlich $d^{(0)}_0 = d^{(m-1)}_{n^{(m-1)}-1}$*, heißt die zusammengesetzte B-Spline-Kurve* geschlossen*.*

Bemerkung 1.40. Ist $m = 1$ sind die Bedeutungen von *geschlossen* in beiden Definitionen identisch.

1.3.1 Composition

In vielen Fällen ist es wichtig, dass man nur mit einer B-Spline-Kurve rechnet, anstatt mit zusammengesetzten Kurvenstücken. Hierzu dient die Composition, die aus mehreren B-Spline-Kurven eine große B-Spline-Kurve erstellt.

Satz 1.41. *Es seien B-Spline-Kurven* $C^{(j)}, j = 0,1$*, mit Ordnungen* $k^{(j)}$*, Kontrollpunkten* $d^{(j)}_i$*, Gewichten* $w^{(j)}_i, i = 0, \ldots, n^{(j)} - 1$*, und Knotenvektoren* $T^{(j)} := \left[t^{(j)}_0, \ldots, t^{(j)}_{n^{(j)}+k^{(j)}-1}\right]$ *gegeben. Es sei* $C^{(0)}(t^{(0)}_{n^{(0)}+k^{(0)}-1}) = C^{(1)}(t^{(1)}_0)$*. Dann existiert eine B-Spline-Kurve* X *wie in (1.7), so dass*

$$X|_{\left[t^{(j)}_0, t^{(j)}_{n^{(j)}+k^{(j)}-1}\right]} = C^{(j)}, \quad j = 0,1. \tag{1.38}$$

Beweis: Der Satz wird bewiesen, indem die B-Spline-Kurve X konstruiert wird. Sei

$$\widehat{k} := \max\left(k^{(0)}, k^{(1)}\right).$$

Erstelle neue B-Spline-Kurven $\widehat{C}^{(0)}, \widehat{C}^{(1)}$ mit Ordnungen $\widehat{k}$, die gemäß Algorithmus 1.34 aus der Graderhöhung von $C^{(0)}$ beziehungsweise $C^{(1)}$ entstanden sind. Die Kontrollpunkte und Gewichte von $\widehat{C}^{(j)}, j = 0,1$, werden mit $\widehat{d}^{(j)}_i$ beziehungsweise $\widehat{w}^{(j)}_i, i = 0, \ldots, \widehat{n}^{(j)} - 1$, bezeichnet. Die neuen Knotenvektoren seien $\widehat{T}^{(j)} := \left[\widehat{t}^{(j)}_0, \ldots, \widehat{t}^{(j)}_{\widehat{n}^{(j)}+\widehat{k}-1}\right]$, wobei

$$\widehat{t}^{(j)}_0 = t^{(j)}_0, \quad \widehat{t}^{(j)}_{\widehat{n}^{(j)}+\widehat{k}-1} = t^{(j)}_{n^{(j)}+k^{(j)}-1}$$

für $j = 0,1$. Erstelle dann neue Kontrollpunkte

$$d_0 := \widehat{d}^{(0)}_0, \ldots, d_{\widehat{n}^{(0)}-2} := \widehat{d}^{(0)}_{\widehat{n}^{(0)}-2}, \quad d_{\widehat{n}^{(0)}-1} := \widehat{d}^{(1)}_0, \ldots, d_{\widehat{n}^{(0)}+\widehat{n}^{(1)}-2} := \widehat{d}^{(1)}_{\widehat{n}^{(1)}-1}$$

und analog dazu die Gewichte $w_i, i = 0, \ldots, n-1$, $n := \widehat{n}^{(0)} + \widehat{n}^{(1)} - 1$. Setze einen neuen Knotenvektor

$$T := [t_0, \ldots, t_{n+k-1}] := \left[\widehat{t}^{(0)}_0, \ldots, \widehat{t}^{(0)}_{\widehat{n}^{(0)}-1}, \widehat{t}^{(0)}_{\widehat{n}^{(0)}} - \widehat{t}^{(1)}_0 + \widehat{t}^{(1)}_1, \ldots, \widehat{t}^{(0)}_{\widehat{n}^{(0)}} - \widehat{t}^{(1)}_0 + \widehat{t}^{(1)}_{\widehat{n}^{(1)}+\widehat{k}-1}\right]$$

mit $k := \widehat{k}$ und erstelle mit diesen Daten eine B-Spline-Kurve

$$X(t) := \frac{\sum_{i=0}^{n-1} w_i d_i N_{i,k}(t|T)}{\sum_{i=0}^{n-1} w_i N_{i,k}(t|T)}.$$

Es ist noch (1.38) zu zeigen. Man stellt dafür zuerst fest, dass

$$t_{\widehat{n}^{(0)}} = \ldots = t_{\widehat{n}^{(0)}+k-2} = \widehat{t}^{(0)}_{\widehat{n}^{(0)}} = \ldots = \widehat{t}^{(0)}_{\widehat{n}^{(0)}+k-1},$$

der Knoten $t_{\widehat{n}^{(0)}}$ hat also Vielfachheit $k-1$. Daraus folgt

$$X(t_{\widehat{n}^{(0)}}) = X(t_{\widehat{n}^{(0)}+k-2}) = d_{\widehat{n}^{(0)}+k-2-k+1} = d_{\widehat{n}^{(0)}-1} = \widehat{d}^{(1)}_0 = \widehat{d}^{(0)}_{\widehat{n}^{(0)}-1}.$$

Gleiches gilt für die Gewichte. Damit hat dann die B-Spline-Kurve X auf dem Intervall $\left[t^{(0)}_0, t^{(0)}_{n^{(0)}+k-1}\right]$ nach Konstruktion identische Kontrollpunkte, Gewichte, Knoten und Ordnung wie $\widehat{C}^{(0)}$. Also gilt mit Definition 1.32:

$$X|_{\left[t^{(j)}_0, t^{(j)}_{n^{(j)}+k^{(j)}-1}\right]} = \widehat{C}^{(0)} = C^{(0)}.$$

Für $C^{(1)}$ geht der Beweis analog, die Knoten in T sind nur um $\widehat{t}^{(0)}_{\widehat{n}^{(0)}} - \widehat{t}^{(1)}_0$ verschoben. □

Korollar 1.42. (Composition) *Es sei eine zusammengesetzte (geschlossene) B-Spline-Kurve*

$$\mathfrak{C} := \{C^{(j)} \mid j = 0, \ldots, m-1\}$$

gegeben, wobei die B-Spline-Kurven $C^{(j)}$ Ordnungen $k^{(j)}$, Kontrollpunkte $d^{(j)}_i$, Gewichte $w^{(j)}_i, i = 0, \ldots, n^{(j)}-1$, und Knotenvektoren $T^{(j)} := \left[t^{(j)}_0, \ldots, t^{(j)}_{n^{(j)}+k^{(j)}-1}\right]$ haben. Dann existiert eine B-Spline-Kurve X mit

$$X|_{\left[t^{(j)}_0, t^{(j)}_{n^{(j)}+k^{(j)}-1}\right]} = C^{(j)}, \quad j = 0, \ldots, m-1. \tag{1.39}$$

Beweis: Der Beweis folgt aus der wiederholten Anwendung von Satz 1.41. □

Bemerkung 1.43. Die erstellte B-Spline-Kurve X aus Korollar 1.42 hat an den Übergangsstellen eine Knotenvielfachheit von $k-1$. Das bedeutet, an diesen Knoten ist die Kurve mindestens C^0-stetig (Piegl und Tiller, [27, S. 88]). Aus Gründen der numerischen Stabilität ist es sinnvoll zu überprüfen, ob man ein Anzahl $l \leq k-1$ der Knoten entfernen kann, um weniger Kontrollpunkte und Knoten zu erhalten. Zusätzlich lassen sich dann bessere Aussagen über die globale Differenzierbarkeit von X treffen.

Es folgt nun der zu obiger Folgerung 1.42 gehörende Algorithmus.

Algorithmus 1.44. (Composition)

Eingabe: zusammengesetzte Kurve $\mathfrak{C}$ wie in Folgerung 1.42
Ausgabe: B-Spline-Kurve X wie in Folgerung 1.42

1. Bestimme die maximale Ordnung $k := \max_{i=0}^{m-1} k^{(i)}$.
2. Erhöhe den Grad jeder Kurve C_i auf $k+1$. (Die Bezeichnungen der Kontrollpunkte, Gewichte und Knoten bleiben erhalten.)

3. Bestimme die Anzahl $n := 1 + \sum_{j=0}^{m-1} \left(n^{(j)} - 1 \right)$ der Kontrollpunkte d_i und Gewichte $w_i, i = 0, \ldots, n-1$.

4. Schleife j von 0 bis $m-1$:

   - Schleife s von 0 bis $n^{(j)} - 2$:
     - Setze $d_{p+s} := d_s^{(j)}$ mit $p := \sum_{r=0}^{j-1} \left(n^{(r)} - 1 \right)$ und analog dazu die Gewichte w_{p+s}.

5. Setze $d_p := d^{(m-1)}_{n^{(m-1)}-1}$ mit $p := \sum_{r=0}^{m-1} \left(n^{(r)} - 1 \right)$ und analog dazu das Gewicht w_p.

6. Setze $i := 0$

7. Setze $t_i := t_0^{(0)}$.

8. Schleife j von 0 bis $m-1$:

   - Schleife s von 1 bis $n^{(j)} - 1$:
     - Setze $t_i := t_s^{(j)}$ und erhöhe $i := i + 1$.

9. Schleife s von 1 bis k:

   - Setze $t_i := t^{(m-1)}_{n^{(m-1)}+k-1}$ und erhöhe $i := i + 1$.

10. Erstelle Splinekurve X der Ordnung k aus Kontrollpunkten d_i, Gewichten $w_i, i = 0, \ldots, n-1$, und Knoten $t_j, j = 0, \ldots, n+k-1$.

1.4 Abstandsberechnung

Für die spätere Zuordnung von Punkten (siehe Teil IV) muss der minimale Abstand eines Punktes zu einer Kurve bestimmt werden. Für die Lösung des Minimierungsproblems berechnet man die erste Ableitung der Kurve und erhält dann mittels eines Newton-Bisektion-Hybrids die Nullstellen als Parameterwerte. Über diese kann der Funktionswert und somit auch der Abstand berechnet werden. Durch die Hybrid-Methode erhält man den Vorteil der quadratische Konvergenz der Newton-Methode, ohne auf die sichere Konvergenz der Bisektionsmethode verzichten zu müssen.

1.4.1 Minimierungsproblem

Ist die B-Spline-Kurve durch die Funktion $X := X(t) \in \mathbb{R}^3, t \in U := [a, b] \subset \mathbb{R}$, und ein Punkt $P \in \mathbb{R}^3$ gegeben, so ist die Suche nach dem Punkt $X(t^*)$ ein Minimierungsproblem. Das heißt

$$t^* := \underset{t \in U}{\operatorname{argmin}} \| X(t) - P \|_2^2. \tag{1.40}$$

Das Minimierungsproblem $\min \|X-P\|_2^2$ kann auch geschrieben werden als Minimierung der Funktion $F(t)$ mit

$$\begin{aligned} F &:= \|X-P\|_2^2 = X^TX - 2X^TP + P^TP = \sum_{i=1}^{3} X_i^2 - 2P_iX_i + P_i^2 \\ &= \sum_{i=1}^{3} X_i(X_i-2P_i) + P_i^2 = (X-P)^T(X-P). \end{aligned} \tag{1.41}$$

Das Argument t wird dabei der Übersicht halber weggelassen. Um dieses Problem zu lösen, muss man die (reellen) Nullstellen von F' berechnen. Die erste Ableitung von F ist gegeben durch

$$F' = \sum_{i=1}^{3} \left(X_i^2 - 2P_iX_i + P_i^2\right)' = 2\sum_{i=1}^{3}(X_i-P_i)X_i' = 2(X-P)^TX'. \tag{1.42}$$

Möchte man zur Nullstellenberechnung die Newton-Methode benutzen, benötigt man zusätzlich noch die zweite Ableitung:

$$\begin{aligned} F'' &= \left(2\sum_{i=1}^{3}(X_i-P_i)X_i'\right)' = 2\sum_{i=1}^{3}\left((X_i-P_i)X_i'\right)' = 2\left(\sum_{i=1}^{3}(X_i-P_i)'X_i' + (X_i-P_i)X_i''\right) \\ &= 2\left(\sum_{i=1}^{3} X_i'^2 + (X_i-P_i)X_i''\right) = 2X'^TX' + 2(X-P)^TX''. \end{aligned} \tag{1.43}$$

Auswahl der Startwerte

Da die Newton-Methode nur lokal konvergiert, wenn man einen guten Startwert hat, ist nicht gesichert, dass das Verfahren korrekt konvergiert. Man benutzt deshalb die Bisektionsmethode als Sicherheit, für die man ein geeignetes Startintervall benötigt.

Definition 1.45. *Es seien eine B-Spline-Kurve X wie in (1.7) und geordnete Parameterwerte*

$$U := \{u_j \in [t_0, t_{n+k-1}] \mid u_i < u_j, i < j, i, j = 0, \ldots, m-1\}$$

gegeben. Bezeichne $j := \operatorname{argmin}_{i=0}^{m-1}\|X(u_i)-P\|_2^2$ den Index des kleinsten Abstands. Dann heißt

$$[v_1, v_2] := \begin{cases} [u_0, u_1], & j = 0, \\ [u_{j-1}, u_{j+1}], & 1 < j < m-1, \\ [u_{m-2}, u_{m-1}], & j = m-1, \end{cases} \tag{1.44}$$

das Konfidenzintervall *für die Bisektionsmethode. Der Wert u_j ist der* Startwert *für die Newton-Methode.*

So kann man einen Newton-Bisektion-Hybrid erstellen.

1.4.2 Newton-Bisektion-Hybrid

Auch wenn die Bisektionsmethode sehr bekannt ist, wird der Algorithmus kurz vorgestellt.

Algorithmus 1.46. (Bisektionsmethode)

Eingabe: Minimierungsfunktion F und Ableitung F', Konfidenzintervall $[b_l,\ b_r]$ mit $F'(b_l)F'(b_r) < 0$, Genauigkeit ϵ

Ausgabe: Parameterwert t mit $|F'(t)| \leq \epsilon$

1. Wiederhole:
   - Setze $t := \frac{b_l+b_r}{2}$.
   - Ist $F'(b_l) \cdot F'(t) > 0$, setze $b_l := t$, sonst setze $b_r := t$.
2. solange $|F'(t)| > \epsilon$.

Auch die Newton-Methode sollte bekannt sein. Daher wird der Algorithmus gleich zum Newton-Bisektion-Hybrid umgeformt.

Algorithmus 1.47. (Newton-Bisektion-Hybrid)

Eingabe: Minimierungsfunktion F und Ableitungen F', F'', Konfidenzintervall $[b_l,\ b_r]$ mit $F'(b_l)F'(b_r) < 0$, Genauigkeit ϵ

Ausgabe: Parameterwert t mit $|F'(t)| \leq \epsilon$

1. Solange $|F'(t)| > \epsilon$:
   - Ist $F''(t) = 0$, berechne für b_l, b_r und t einen Schritt des Bisektionsverfahrens. Sonst:
     - Setze $t := t - \frac{F'(t)}{F''(t)}$.
     - Ist $t < b_l$ oder $t > b_r$, berechne für b_l, b_r und t einen Schritt des Bisektionsverfahrens.

Bemerkung 1.48. Es ist mit der diskreten Abtastung zur Startwertbestimmung in Definition 1.45 nicht gesichert, dass der Newton-Bisektion-Hybrid wirklich gegen das Minimum konvergiert. Die Abtastung an den m Punkten muss daher fein genug sein und es wird davon ausgegangen, dass der Punkt P „nah“ an der Kurve liegt.

Geschlossene Splinekurven

Für geschlossene Kurven (siehe Abschnitt 1.3) legt man das Konfidenzintervall (1.44) wie folgt fest:

$$[v_1, v_2] := \begin{cases} [u_{m-1}, u_1], & j = 0, \\ [u_{j-1}, u_{j+1}], & 1 < j < m-1, \\ [u_{m-2}, u_0], & j = m-1. \end{cases}$$

Damit ist $[v_1, v_2]$ kein Intervall mehr im klassischen Sinne, da $v_2 < v_1$ sein kann. In dem Fall wird der Algorithmus für den Newton-Bisketion-Hybrid (Algorithmus 1.47) entsprechend angepasst.

Problematisch ist die Berechnung der Abstände von Punkten, die in der Nähe des Schließpunktes liegen, da keine eindeutige Zuordnung mehr garantiert werden kann.

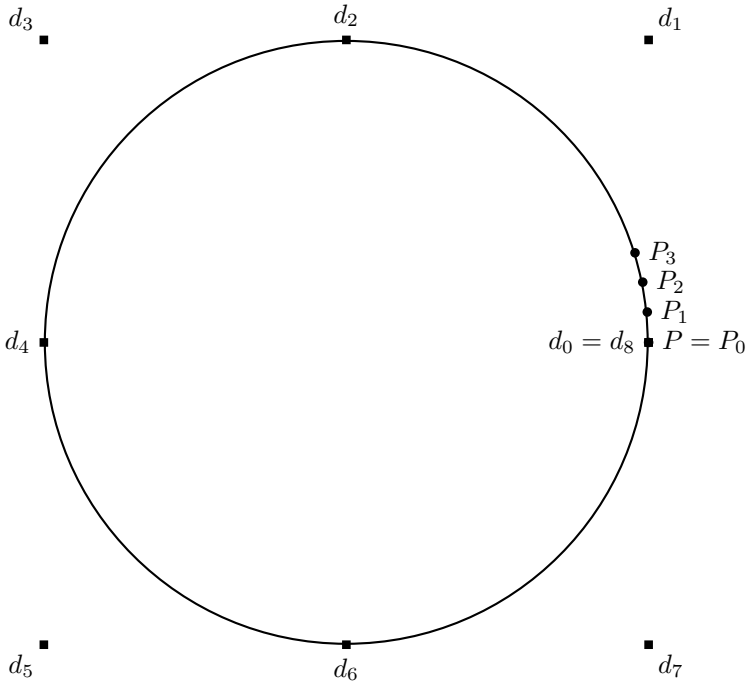


Abbildung 1.4: Parameterberechnung am Kreis

Beispiel 1.49. Sei ein Kreis K in geometrischer Darstellung auf $[0, 2\pi]$ und ein Punkt $P = K(0) = K(2\pi)$ gegeben (Abbildung 1.4). Möchte man den Parameterwert t zum zu P nächstgelegenen Punkt auf K bestimmen, so erhält man entweder $t = 0$ oder $t = 2\pi$, da $K(0) = K(2\pi) = P$. Für eine reine Parametrierung des einzelnen Punktes P ist dies unerheblich. Kommen aber weitere Punkte P_1, P_2, P_3 ins Spiel, die später gegebenenfalls durch eine Kurve approximiert werden sollen, ist es wichtig, dass $t = 0$ gewählt wird. Andernfalls sähe die Zuordnung wie folgt aus:

$$t = 2\pi \leftarrow P_0, t = 0.1 \leftarrow P_1, t = 0.2 \leftarrow P_2, t = 0.3 \leftarrow P_3$$

Eine Approximation der Punkte $P_0, \ldots, P_3$ über die gefundenen Parameterwerte gäbe kein gutes Ergebnis mehr, da die entstehende Approximationskurve X einen „Bogen“ machen würde (Abbildung 1.5).

Aus diesem Grund sind geschlossene Kurven (ebenso wie geschlossene Flächen in Teil III, Abschnitt 2.3) eine Besonderheit und berechnete Parameterwerte müssen in einer externen Routine gegebenenfalls korrigiert werden.

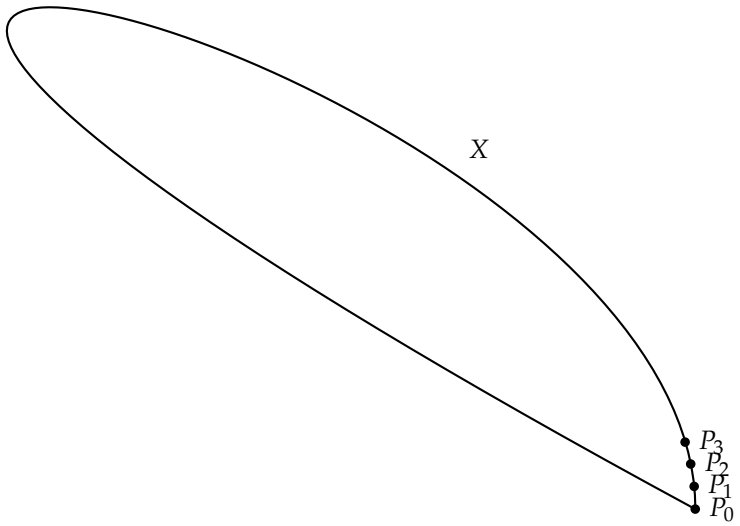


Abbildung 1.5: Approximierter Kreisbogen

2 Splineflächen

Splineflächen werden oft beim Computer-Design von Freiformflächen benutzt. Sie können auf verschiedene Arten dargestellt werden, was vor allem von der Struktur des Definitionsbereiches abhängt (Floater und Hormann [14]). Die am weitesten verbreitetste Art ist die Definition auf einem rechteckigen Parametergebiet $\{(u,v) \mid u \in [a_1,b_1] \subset \mathbb{R}, v \in [a_2,b_2] \subset \mathbb{R}\}$. Diese wird auch teilweise in dem vorliegendem IGES-Dateiformat verwendet, weshalb sich die Benutzung in dieser Form hier anbietet.

Definition 2.1. *Die Fläche*

$$X(u,v) := \sum_{i=0}^{n_u-1} \sum_{j=0}^{n_v-1} d_{i,j} N_{i,k_u}(u|T_u) N_{j,k_v}(v|T_v) \tag{2.1}$$

heißt polynomiale Tensor-Produkt-B-Spline-Fläche *der Ordnungen k_u, k_v über dem rechteckigen* Parametergebiet

$$D := [u_0, u_{n_u+k_u-1}] \times [v_0, v_{n_v+k_v-1}] \tag{2.2}$$

mit einem Kontrollnetz $d_{i,j} \in \mathbb{R}^3, i = 0,\ldots,n_u-1, j = 0,\ldots,n_v-1$, *und Knotenvektoren* $T_u := [u_0,\ldots,u_{n_u+k_u-1}], T_v := [v_0,\ldots,v_{n_v+k_v-1}]$. *Das Kontrollnetz wird auch* de-Boor-Netz *genannt.*

Die Darstellung der Fläche kann man sich vorstellen, indem man eine B-Spline-Kurve

$$X(u) = \sum_{i=0}^{n_u-1} d_i N_{i,k_u}(u|T_u)$$

entlang einer zweiten Kurve durch den Raum bewegt. Dabei ist es erlaubt, dass sich X verändert beziehungsweise dessen Kontrollpunkte variieren.

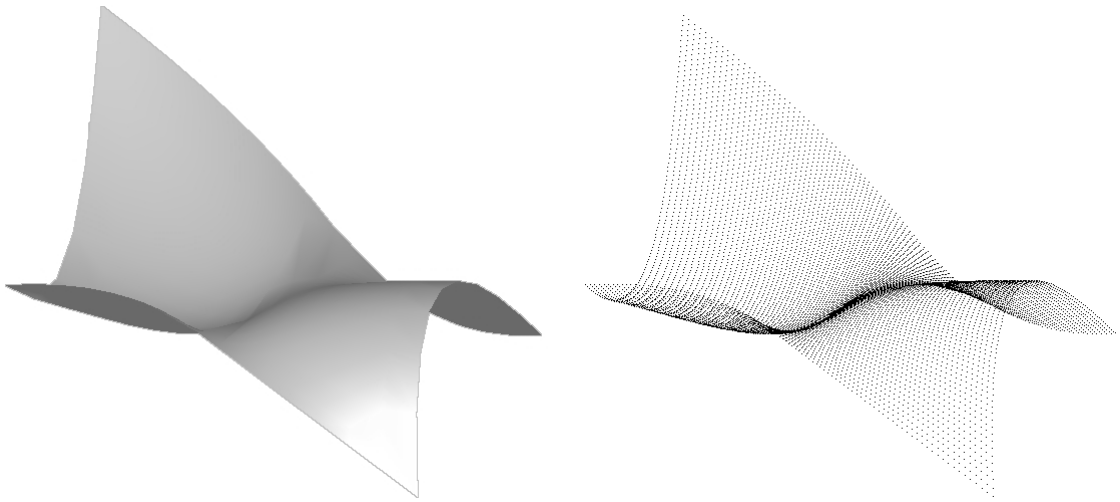
Auf die gleiche Art ist es nun möglich, die allgemeineren rationalen B-Spline-Flächen zu beschreiben.

Definition 2.2. *Eine* rationale Tensor-Produkt-B-Spline-Fläche *ist definiert durch*

$$X(u,v) := \frac{\sum_{i=0}^{n_u-1}\sum_{j=0}^{n_v-1} w_{i,j} d_{i,j} N_{i,k_u}(u|T_u) N_{j,k_v}(v|T_v)}{\sum_{i=0}^{n_u-1}\sum_{j=0}^{n_v-1} w_{i,j} N_{i,k_u}(u|T_u) N_{j,k_v}(v|T_v)}, \tag{2.3}$$

mit Ordnungen, Kontrollpunkten und Knotenvektoren wie in Definition 2.1 und zusätzlichen Gewichten $w_{i,j} \in \mathbb{R}, i = 0,\ldots,n_u-1, j = 0,\ldots,n_v-1$.

B-Spline-Flächen (auch *Patches* genannt) haben die gleichen Eigenschaften wie B-Spline-Kurven, so dass sie beispielsweise affin-invariant sind. Ebenfalls liegt jeder Punkt der Fläche in der konvexen Hülle des Kontrollnetzes.



a. solid b. gepunktet

Abbildung 2.1: Beispielpatch

Konvention 2.3.

1. Die in dieser Arbeit behandelten B-Spline-Flächen sollen an den Randknoten u_0 und $u_{n_u+k_u-1}$ Vielfachheit k_u und an v_0 und $v_{n_v+k_v-1}$ Vielfachheit k_v besitzen. Dadurch werden die vier Eckpunkte interpoliert.

2. Die Knotenvektoren T_u und T_v in $N_{i,k_u}(u|T_u)$ beziehungsweise $N_{j,k_v}(v|T_v)$ werden im weiteren Verlauf nicht mehr explizit angeben, soweit T_u und T_v eindeutig aus dem Kontext erkennbar sind.

3. In dieser Arbeit bezeichnen die allgemeineren Begriffe *Spline-Fläche* und *B-Spline-Fläche* immer eine (rationale) Tensor-Produkt-B-Spline-Fläche, soweit nicht anders angegeben. Wird von einer B-Spline-Fläche X gesprochen, seien automatisch Kontrollpunkte, Gewichte, Ordnungen, Knotenvektoren und Parametergebiet wie in (2.3) gegeben.

2.1 Auswertung

Die Auswertung eines Punktes $X(u,v)$ wird auf die Auswertung der Kurven zurückgeführt. So wird erst entlang der einen Richtung die Kurve in u ausgewertet, woraus sich neue Kontrollpunkte ergeben, in denen dann v ausgewertet wird.

Lemma 2.4. *Ist eine B-Spline-Fläche X wie in (2.3) gegeben, definiere B-Spline-Kurven*

$$\begin{aligned} R_j(u) &:= \frac{\sum_{i=0}^{n_u-1} w_i d_i N_{i,k_u}(u|T_u)}{\sum_{i=0}^{n_u-1} w_i N_{i,k_u}(u|T_u)}, \\ W_j(u) &:= \sum_{i=0}^{n_u-1} w_i N_{i,k_u}(u|T_u), \qquad j = 0,\ldots,n_v-1, \end{aligned}$$

und erstelle eine B-Spline-Kurve

$$S(v) := \frac{\sum_{j=0}^{n_v-1} W_j(u) R_j(u) N_{j,k_v}(v|T_v)}{\sum_{j=0}^{n_v-1} W_j(u) N_{j,k_v}(v|T_v)}.$$

Es gilt dann: $X(u,v) = S(v)$.

Beweis: Sei X eine polynomiale B-Spline-Fläche gemäß (2.1). Diese kann man direkt über die Basis-Funktionen auswerten. In Matrixschreibweise bedeutet das:

$$X(u,v) = \begin{pmatrix} N_{0,k_u}(u|T_u) & \ldots & N_{n_u-1,k_u}(u|T_u) \end{pmatrix} \begin{pmatrix} d_{0,0} & \ldots & d_{0,n_v-1} \\ \vdots & & \vdots \\ d_{n_u-1,0} & \ldots & d_{n_u-1,n_v-1} \end{pmatrix} \begin{pmatrix} N_{0,k_v}(v|T_v) \\ \vdots \\ N_{n_v-1,k_v}(v|T_v) \end{pmatrix}.$$

Setzt man nun

$$R(u) := \begin{pmatrix} N_{0,k_u}(u|T_u) & \ldots & N_{n_u-1,k_u}(u|T_u) \end{pmatrix} \begin{pmatrix} d_{0,0} & \ldots & d_{0,n_v-1} \\ \vdots & & \vdots \\ d_{n_u-1,0} & \ldots & d_{n_u-1,n_v-1} \end{pmatrix}$$

und

$$S(v) := R(u) \begin{pmatrix} N_{0,k_v}(v|T_v) \\ \vdots \\ N_{n_v-1,k_v}(v|T_v) \end{pmatrix},$$

so ist

$$X(u,v) = S(v).$$

Analog geht der Beweis für rationale B-Spline-Flächen gemäß (2.3). □

Dies führt zu folgendem Algorithmus.

Algorithmus 2.5.

Eingabe:	B-Spline-Fläche X wie in (2.3)
Ausgabe:	B-Spline-Kurve S mit $X(u,v) = S(v)$

1. Schleife j von 0 bis $n_v - 1$:
   - Erstelle B-Spline-Kurve R_j der Ordnung k_u mit Kontrollpunkten $d_{i,j}$, Gewichten $w_{i,j}, i = 0, \ldots, n_u - 1$, und Knotenvektor T_u.
   - Erstelle eindimensionale polynomiale B-Spline-Kurve W der Ordnung k_u mit Kontrollpunkten $w_{i,j}, i = 0, \ldots, n_u - 1$, und Knotenvektor T_u.
   - Setze $d'_j := R(u)$ und $w'_j := W(u)$.
2. Erstelle B-Spline-Kurve S der Ordnung k_v mit Kontrollpunkten d'_j, Gewichten $w'_j, j = 0, \ldots, n_v - 1$, und dem Knotenvektor T_v.

2.2 Ableitungen

Da man bei B-Spline-Flächen im Gegensatz zu Kurven ein Parametergebiet in zwei Veränderlichen hat, muss man die partiellen Ableitungen berechnen. Dies wird wieder auf die Auswertung der Ableitungen von Kurven zurückgeführt.

2.2.1 Gradient

Lemma 2.6. *Ist eine B-Spline-Fläche X wie in (2.3) gegeben, definiere B-Spline-Kurven*

$$\begin{aligned} R_j(u) &:= \frac{\sum_{i=0}^{n_u-1} w_i d_i N_{i,k_u}(u|T_u)}{\sum_{i=0}^{n_u-1} w_i N_{i,k_u}(u|T_u)}, \\ W_j(u) &:= \sum_{i=0}^{n_u-1} w_i N_{i,k_u}(u|T_u), \qquad j = 0, \ldots, n_v - 1, \end{aligned}$$

und erstelle eine B-Spline-Kurve

$$S(v) := \frac{\sum_{j=0}^{n_v-1} W_j(u) R_j(u) N_{j,k_v}(v|T_v)}{\sum_{j=0}^{n_v-1} W_j(u) N_{j,k_v}(v|T_v)}.$$

Es gilt dann $\frac{\partial X(u,v)}{\partial v} = S'(v)$.

Dies führt zu folgendem Algorithmus.

Algorithmus 2.7.

Eingabe: B-Spline-Fläche X wie in (2.3)
Ausgabe: partiellen Ableitungen $\frac{\partial X(u,v)}{\partial u}$ und $\frac{\partial X(u,v)}{\partial v}$

1. Schleife j von 0 bis $n_v - 1$:
   - Erstelle B-Spline-Kurve R_j der Ordnung k_u mit Kontrollpunkten $d_{i,j}$, Gewichten $w_{i,j}, i = 0, \ldots, n_u - 1$, und dem Knotenvektor T_u.
   - Erstelle eindimensionale polynomiale B-Spline-Kurve W der Ordnung k_u mit Kontrollpunkten $w_{i,j}, i = 0, \ldots, n_u - 1$, und Knotenvektor T_u.

- Setze $d'_j := R(u)$ und $w'_j := W(u)$.

2. Erstelle B-Spline-Kurve S der Ordnung k_v mit Kontrollpunkten d'_j, Gewichten $w'_j, j = 0, \ldots, n_v - 1$, und dem Knotenvektor T_v.
3. Es ist $\frac{\partial X(u,v)}{\partial v} = S'(v)$.
4. Vertausche die Rollen von i und j und gehe Schritt 1 bis 3 erneut durch. So erhält man $\frac{\partial X(u,v)}{\partial u} = S'(u)$.

Es ist dann

$$\nabla X(u,v) = \begin{pmatrix} \frac{\partial X(u,v)}{\partial u} \\ \frac{\partial X(u,v)}{\partial v} \end{pmatrix}.$$

Bemerkung 2.8. Die Einträge von ∇X und auch von der folgenden Hessematrix sind vektorwertig.

2.2.2 Hesse-Matrix

Die zweite Ableitung in zwei Veränderlichen ist eine 2x2-Matrix (mit Vektoreinträgen), bestehend aus der zweifachen Ableitung in jeweils eine Richtung und der gemischten Ableitung.

Lemma 2.9. *Ist eine B-Spline-Fläche X wie in (2.3) gegeben, definiere B-Spline-Kurven*

$$\begin{aligned} R_j(u) &:= \frac{\sum_{i=0}^{n_u-1} w_i d_i N_{i,k_u}(u|T_u)}{\sum_{i=0}^{n_u-1} w_i N_{i,k_u}(u|T_u)}, \\ W_j(u) &:= \sum_{i=0}^{n_u-1} w_i N_{i,k_u}(u|T_u), \qquad j = 0, \ldots, n_v - 1, \end{aligned}$$

und erstelle eine B-Spline-Kurve

$$S(v) := \frac{\sum_{j=0}^{n_v-1} W_j(u) R_j(u) N_{j,k_v}(v|T_v)}{\sum_{j=0}^{n_v-1} W_j(u) N_{j,k_v}(v|T_v)}.$$

Es gilt dann $\frac{\partial^2 X(u,v)}{\partial v^2} = S''(v)$. Setzt man

$$S(v) := \frac{\sum_{j=0}^{n_v-1} W'_j(u) R'_j(u) N_{j,k_v}(v|T_v)}{\sum_{j=0}^{n_v-1} W'_j(u) N_{j,k_v}(v|T_v)},$$

gilt $\frac{\partial^2 X(u,v)}{\partial u \partial v} = \frac{\partial^2 X(u,v)}{\partial v \partial u} = S'(v)$.

Dies führt zu folgendem Algorithmus.

Algorithmus 2.10.

Eingabe: B-Spline-Fläche X wie in (2.3)
Ausgabe: Hessematrix H_X

1. Schleife j von 0 bis $n_v - 1$:
   - Erstelle B-Spline-Kurve R_j der Ordnung k_u mit Kontrollpunkten $d_{i,j}$, Gewichten $w_{i,j}, i = 0, \ldots, n_u - 1$, und dem Knotenvektor T_u.
   - Erstelle eindimensionale polynomiale B-Spline-Kurve W der Ordnung k_u mit Kontrollpunkten $w_{i,j}, i = 0, \ldots, n_u - 1$, und Knotenvektor T_u.
   - Setze $d'_j := R(u)$ und $w'_j := W(u)$.
2. Erstelle B-Spline-Kurve S der Ordnung k_v mit Kontrollpunkten d'_j, Gewichten $w'_j, j = 0, \ldots, n_v - 1$, und dem Knotenvektor T_v.
3. Es ist $\frac{\partial^2 X(u,v)}{\partial v^2} = S''(v)$.
4. Vertausche die Rollen von i und j und gehe Schritt 1 bis 3 erneut durch. So erhält man $\frac{\partial^2 X(u,v)}{\partial u^2} = S''(u)$.
5. Für die gemischte Ableitung setze in Schritt 1 $d'_j := R'(u)$ und $w'_j := W'(u)$. Man erhält $\frac{\partial^2 X(u,v)}{\partial v \partial u} = \frac{\partial^2 X(u,v)}{\partial u \partial v} = S'(v)$.

Es ist dann

$$H_X(u,v) = \begin{pmatrix} \frac{\partial^2 X(u,v)}{\partial u^2} & \frac{\partial^2 X(u,v)}{\partial u \partial v} \\ \frac{\partial^2 X(u,v)}{\partial v \partial u} & \frac{\partial^2 X(u,v)}{\partial v^2} \end{pmatrix}.$$

Bemerkung 2.11. Bei der Ableitung rationaler Splineflächen, benötigt man wie bei den B-Spline-Kurven die zugehörige polynomiale Auswertung. Es ist daher sinnvoll, den abgeänderten de-Boor-Algorithmus (siehe Abschnitt 1.2.2) zu benutzen.

2.3 Normalenebene

Die Ausgleichsebene einer B-Spline-Fläche wird als Least-Squares-Ebene berechnet. Algorithmisch einfacher lässt sich eine so genannte Normalenebene berechnen, die hier auch genutzt wird.

Definition 2.12. *Es sei eine B-Spline-Fläche X wie in (2.3) gegeben, die an allen Stellen $(u,v) \in D$ differenzierbar ist. Bezeichne*

$$\mathfrak{N}(u,v) := \frac{\partial X(u,v)}{\partial u} \times \frac{\partial X(u,v)}{\partial v} \tag{2.4}$$

die Normale von X an der Stelle $(u,v) \in D$. *Seien $m > 0$ Parameterwerte $P := \{p_l \in D \mid l = 0, \ldots, m-1\}$ gegeben. Dann wird durch*

$$\mathfrak{n} := \frac{1}{m} \sum_{l=0}^{m-1} \mathfrak{N}(p_l), \quad \mathfrak{o} := \frac{1}{m} \sum_{l=0}^{m-1} X(p_l) \tag{2.5}$$

die Normale $\mathfrak{n}$ *und der* Stützvektor $\mathfrak{o}$ *und dadurch die* Normalenebene $\mathfrak{A}(X,P)$ von X bezüglich P *definiert.*

Dabei sollte man darauf achten, dass die Fläche möglichst gleichmäßig abgetastet wird, um einen Überhang an Punkten an einer Stelle zu vermeiden. Im Gegensatz zu Kurven kann man eine Längenfunktion nicht mehr so einfach und schnell approximieren. Es empfiehlt sich daher, als grobe Näherung die Länge der Kontrollpolygone zu benutzen.

Definition 2.13. *Es sei eine B-Spline-Fläche X wie in (2.3) gegeben. Dann bezeichne*

$$L_i^{(u)} := \sum_{j=1}^{n_v-1} \|d_{i,j} - d_{i,j-1}\|, \quad i = 0, \ldots, n_u - 1, \tag{2.6}$$

die Länge der Kontrollpolygone *in Richtung u beziehungsweise*

$$L_j^{(v)} := \sum_{i=1}^{n_u-1} \|d_{i,j} - d_{i-1,j}\|, \quad j = 0, \ldots, n_v - 1, \tag{2.7}$$

in Richtung v.

Algorithmus 2.14. (Normalenebene)
Eingabe: B-Spline-Fläche X wie in (2.3), Blockgrößen $b_u, b_v > 0$ für die Abtastung
Ausgabe: Normalenebene $\mathfrak{A}(X, P)$

1. Bestimme die Anzahl

$$m_u := \frac{\max_{i=0}^{n_u-1} L_i^{(u)}}{b_u} \text{ und } m_v := \frac{\max_{j=0}^{n_v-1} L_j^{(v)}}{b_v}$$

   der auszuwertenden Punkte in jeweils eine Richtung.

2. Berechne Punktegitter

$$P := \left\{ \left(u_0 + \frac{i}{m_u - 1} \left(u_{n_u+k_u-1} - u_0 \right), v_0 + \frac{j}{m_v - 1} \left(v_{n_v+k_v-1} - v_0 \right) \right) \in D \mid \right.$$
$$\left. i = 0, \ldots, m_u - 1, j = 0, \ldots, m_v - 1 \right\}.$$

3. Berechne Normale $\mathfrak{n}$ und Schwerpunkt $\mathfrak{o}$ gemäß (2.5).

4. Erstelle Normalenebene $\mathfrak{A}(X, P)$ mit Normalenvektor $\mathfrak{n}$ und Stützvektor $\mathfrak{o}$.

2.4 Knoteneinfügen

Der Algorithmus ist identisch zu denen bei B-Spline-Kurven (siehe Abschnitt 1.2.5). Man muss darauf achten, dass das Einfügen eines Knotens in Richtung u eine komplette Reihe neuer Kontrollpunkte in Richtung v erzeugt. Dadurch kann die Anzahl der Kontrollpunkte sehr schnell steigen.

2.5 Getrimmte Flächen

Es gibt Freiformflächen wie zum Beispiel Kreisscheiben oder Dreiecke, die sich durch Tensor-Produkt-Spline-Flächen nur schlecht oder gar nicht darstellen lassen. Der Grund ist die Abbildung des rechteckigen Parametergebietes auf eine andersförmige Fläche. Daher benutzt man in solchen Fällen meistens *Trimmkurven*.

Trimmkurven sind im Parameterbereich definierte, geschlossene Kurven, die angeben, welche eingeschlossenen Gebiete zu einer Fläche gehören sollen und welche nicht. Nur die jeweiligen Parameterwerte, die innerhalb der Trimmkurven liegen, werden im Bildbereich dargestellt (siehe auch Teil IV, Abschnitt 2).

Definition 2.15. *Es seien eine B-Spline-Fläche X wie in (2.3) und eine geschlossene B-Spline-Kurve $\mathfrak{B}(t) \in D, t \in [t_0, t_{n+k-1}]$, wie in (1.7) gegeben. Dann bezeichne $\mathfrak{M}(\mathfrak{B}) \subseteq D$ das von $\mathfrak{B}$* begrenzte Parametergebiet *(siehe Jordanscher Kurvensatz in Brouwer [2], Maehara [25], Veblen [36]). Verläuft $\mathfrak{B}$ gegen den Uhrzeigersinn, heißt $\mathfrak{B}$* äußere Trimmkurve *(im Parameterbereich), verläuft $\mathfrak{B}$ im Uhrzeigersinn* innere Trimmkurve *(im Parameterbereich) von X.*

Bemerkung 2.16. Die Trimmkurven können auch als zusammengesetzte B-Spline-Kurve (siehe Abschnitt 1.3) gegeben sein. Zur Verdeutlichung werden daher für Trimmkurven Frakturbuchstaben benutzt. Der Einfachheit halber wird aber angenommen, dass es sich nur um eine einzelne Kurve handelt, soweit nicht anders angegeben.

Definition 2.17. *Es seien eine B-Spline-Fläche X und eine Trimmkurve $\mathfrak{B}$ im Parameterbereich gemäß Definition 2.15 gegeben. Dann heißt*

$$\mathfrak{C}(t) := (X \circ \mathfrak{B})(t) := X(\mathfrak{B}(t)), \quad t \in [t_0, t_{n+k-1}], \tag{2.8}$$

die zugehörige Trimmkurve im Bildbereich *und durch*

$$\mathfrak{M}(\mathfrak{C}) := \{X(u,v) \mid (u,v) \in \mathfrak{M}(\mathfrak{B})\}$$

sei der sichtbare Bereich von X *definiert.*

Definition 2.18. *Es sei ein Gebiet $G \subset \mathbb{R}^2$ gegeben. Definiere*

$$B_{\min}(G) = \begin{pmatrix} b_{\min}^{(x)} \\ b_{\min}^{(y)} \end{pmatrix} := \begin{pmatrix} \min_{(g_1,g_2)\in G}(g_1) \\ \min_{(g_1,g_2)\in G}(g_2) \end{pmatrix}, \quad B_{\max}(G) = \begin{pmatrix} b_{\max}^{(x)} \\ b_{\max}^{(y)} \end{pmatrix} := \begin{pmatrix} \max_{(g_1,g_2)\in G}(g_1) \\ \max_{(g_1,g_2)\in G}(g_2) \end{pmatrix}. \tag{2.9}$$

Dann ist die Bounding Box *B als minimales, achsenparalleles Rechteck definiert, das G komplett umschließt:*

$$B(G) := \left[b_{\min}^{(x)}, b_{\max}^{(x)}\right] \times \left[b_{\min}^{(y)}, b_{\max}^{(y)}\right]. \tag{2.10}$$

Man schreibt kurz:

$$B(G) = [B_{\min}(G), B_{\max}(G)]. \tag{2.11}$$

Im Dreidimensionalen ist die Bounding Box als minimaler, achsenparalleler Quader definiert.

Ist die Bounding Box $B(\mathfrak{M}(\mathfrak{B}))$ der äußeren Trimmkurve $\mathfrak{B}$ kleiner als der Parameterbereich D der Tensor-Produkt-Spline-Fläche,

$$B(\mathfrak{M}(\mathfrak{B})) \subsetneq D,$$

lohnt es sich, die Fläche auf die Bounding Box einzuschränken, da die Punkte außerhalb des sichtbaren Bereichs nicht angezeigt werden:

$$\nexists (u,v) \in D \backslash \mathfrak{M}(\mathfrak{B}) \text{ mit } X(u,v) \in \mathfrak{M}(\mathfrak{C}).$$

Dies reduziert zum einen die Anzahl der Kontrollpunkte, zum anderen hilft es bei der Approximation der Gebiete, in denen keine Punkte der Punktewolke liegen.

Folgender Algorithmus trimmt eine B-Spline-Fläche auf die Größe der Bounding Box der äußeren Trimmkurve. Die Trimmung wird dabei genauso wie bei den B-Spline-Kurven durchgeführt (siehe Abschnitt 1.2.8). Man fügt die gewünschten Knoten so oft ein, bis sie Vielfachheit $k_u - 1$ beziehungsweise $k_v - 1$ besitzen und verwirft alle Knoten, Kontrollpunkte und Gewichte außerhalb.

Algorithmus 2.19. (Flächen-Trimmung)
Eingabe: B-Spline-Fläche X wie in (2.3), Trimmkurve $\mathfrak{B}$ im Parameterbereich
Ausgabe: getrimmte B-Spline-Fläche $X^*(u,v)$

1. Betrachte die Bounding Box $B(\mathfrak{M}(\mathfrak{B})) := [a_1, b_1] \times [a_2, b_2]$.

2. Füge Knoten a_1 und b_1 so oft in X ein, dass diese jeweils Vielfachheit $k_u - 1$ haben.

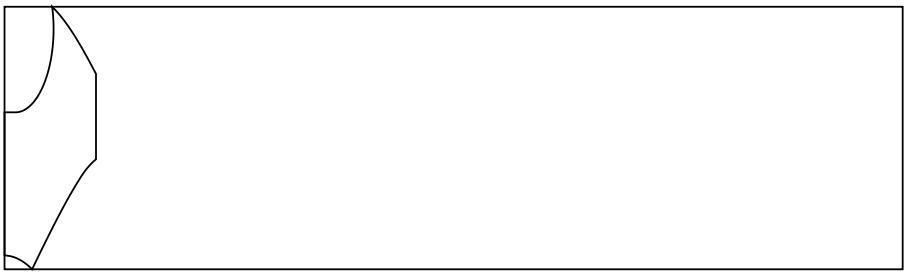
3. Füge Knoten a_2 und b_2 so oft in X ein, dass diese jeweils Vielfachheit $k_v - 1$ haben.

4. Entferne die Knoten und Kontrollpunkte, die außerhalb des Parametergebietes $[a_1, b_1] \times [a_2, b_2]$ liegen.

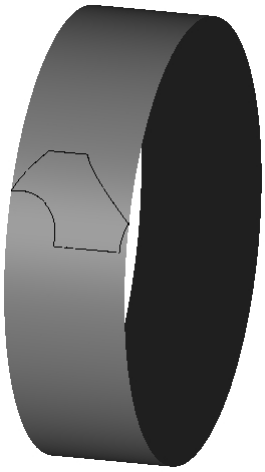
Beispiel 2.20. In Abbildung 2.2 sieht man eine Rotationsfläche und deren äußere Trimmkurve im Bildbereich und Parameterbereich. Trimmt man diese Fläche nun, erhält man das Ergebnis in Abbildung 2.3.

Definition 2.21. *Es sei eine B-Spline-Fläche X wie in (2.3) gegeben. Dann wird durch*

$$\mathfrak{R}(X) := \{R_0, R_1, R_2, R_3\} \tag{2.12}$$

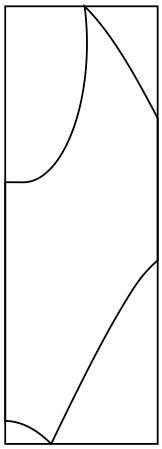


a. Parameterbereich

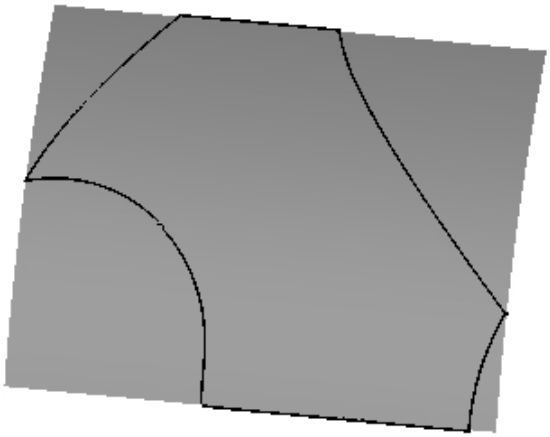


b. Bildbereich

Abbildung 2.2: Rotationsfläche und Trimmkurve



a. Parameterbereich



b. Bildbereich

Abbildung 2.3: Getrimmte Fläche

mit

$$
\begin{aligned}
R_0(u) &:= \frac{\sum_{i=n_u-1}^{0} w_{i,0} d_{i,0} N_{i,k_u}(u|T_u)}{\sum_{i=n_u-1}^{0} w_{i,0} N_{i,k_u}(u|T_u)} \\
R_1(v) &:= \frac{\sum_{j=0}^{n_v-1} w_{0,j} d_{0,j} N_{j,k_v}(v|T_v)}{\sum_{j=0}^{n_v-1} w_{0,j} N_{j,k_v}(v|T_v)} \\
R_2(u) &:= \frac{\sum_{i=0}^{n_u-1} w_{i,n_v-1} d_{i,n_v-1} N_{i,k_u}(u|T_u)}{\sum_{i=0}^{n_u-1} w_{i,n_v-1} N_{i,k_u}(u|T_u)} \\
R_3(v) &:= \frac{\sum_{j=n_v-1}^{0} w_{n_u-1,j} d_{n_u-1,j} N_{j,k_v}(v|T_v)}{\sum_{j=n_v-1}^{0} w_{n_u-1,j} N_{j,k_v}(v|T_v)}
\end{aligned}
$$

die Randkurve von X *beschrieben.*

Bemerkung 2.22. Nach Definition 2.21 ist die Randkurve eine zusammengesetzte und geschlossene B-Spline-Kurve.

2.6 Abstandsberechnung

Für die spätere Zuordnung von Punkten (siehe Teil IV) muss man neben dem minimalen Abstand eines Punktes zu einer B-Spline-Kurve auch den Abstand zu einer B-Spline-Fläche bestimmen. Die Newton-Methode lässt sich dafür leicht auf das Zweidimensionale übertragen.

Bemerkung 2.23. Bei der Berechnung des Punktes mit dem kleinsten Abstand wird nicht nur der nächste Punkt auf der B-Spline-Fläche berechnet, sondern auch der zugehörige Parameterwert (u^*, v^*), den man später für die Approximation der Punkte benutzt (siehe Teil V).

2.6.1 Minimierungsproblem

Ist die Tensor-Produkt-B-Spline-Fläche X wie in (2.3) und ein Punkt $P \in \mathbb{R}^3$ gegeben, so ist die Suche nach dem Punkt $X(u^*, v^*)$, der zu P den geringsten Abstand hat, ein Minimierungsproblem. Dabei sind (u^*, v^*) definiert als

$$
(u^*, v^*) := \operatorname*{argmin}_{(u,v)\in D} \|X(u,v) - P\|_2^2. \tag{2.13}
$$

Dies kann auf die gleiche Art wie bei den B-Spline-Kurven umgeschrieben werden (siehe Abschnitt 1.4) und man erhält eine minimierende Funktion (Der Übersicht halber werden die Parameterwerte u und v nicht mit angegeben.)

$$
F := \|X - P\|_2^2 = X^T X - 2X^T P + P^T P = \sum_{i=1}^{3} X_i^2 - 2P_i X_i + P_i^2 = (X-P)^T (X-P)
$$

Um dieses Problem zu lösen, muss man die (reellen) Nullstellen von F' berechnen, wobei zu beachten ist, dass F nun eine Funktion in zwei Unbekannten ist. Demzufolge ist

$$\nabla F = 2\sum_{i=1}^{3}(X_i - P_i)\begin{pmatrix}\frac{\partial X_i}{\partial u}\\ \frac{\partial X_i}{\partial v}\end{pmatrix} \in \mathbb{R}^2. \tag{2.14}$$

Für die Newton-Methode benötigt man zusätzlich die zweite Ableitung, welche durch die Hesse-Matrix von F gegeben ist. Es ist

$$H_F = 2\sum_{i=1}^{3}\begin{pmatrix}\left(\frac{\partial X_i}{\partial u}\right)^2 + (X_i - P_i)\frac{\partial^2 X_i}{\partial u^2} & \left(\frac{\partial X_i}{\partial u}\cdot\frac{\partial X_i}{\partial v}\right) + (X_i - P_i)\frac{\partial^2 X_i}{\partial u \partial v}\\ \left(\frac{\partial X_i}{\partial v}\cdot\frac{\partial X_i}{\partial u}\right) + (X_i - P_i)\frac{\partial^2 X_i}{\partial v \partial u} & \left(\frac{\partial X_i}{\partial v}\right)^2 + (X_i - P_i)\frac{\partial^2 X_i}{\partial v^2}\end{pmatrix} \in \mathbb{R}^{2\times 2}. \tag{2.15}$$

Natürlich benötigt man als Startwert für die Newton-Methode einen Wert, der Nahe am globalen Minimum liegt, damit die Methode gegen das Minimum konvergiert.

2.6.2 Newton-Methode im $\mathbb{R}^2$

Im Mehrdimensionalen unterscheidet sich die Newton-Methode nur soweit, dass man nicht durch die Ableitung teilt, da diese als Matrix gegeben ist, sondern mit der inversen Matrix multipliziert. Es ergibt sich als Iterationsschritt

$$x_{i+1} := x_i - H_F^{-1}\nabla F. \tag{2.16}$$

Natürlich berechnet man im Höherdimensionalen nicht die Inverse der Matrix, sondern löst ein lineares Gleichungssystem. In diesem Fall ist die Inverse der 2x2-Matrix aber leicht aufzustellen und kann explizit berechnet werden, denn ist

$$M := \begin{pmatrix}a & b\\ c & d\end{pmatrix} \in \mathbb{R}^{2\times 2}$$

gegeben, dann kann die inverse Matrix als

$$M^{-1} := \frac{1}{ad - bc}\begin{pmatrix}d & -b\\ -c & a\end{pmatrix} \in \mathbb{R}^{2\times 2} \tag{2.17}$$

geschrieben werden.

Auswahl der Startwerte

Wie auch schon bei B-Spline-Kurven wird mit einem Verfahren gearbeitet, bei dem man zuerst einige Punkte auf der Fläche berechnet und dann den Parameter (u, v) des Punktes Q mit kleinsten Abstand zu P als Startwert für die Newton-Iteration benutzt.

Definition 2.24. *Es seien eine B-Spline-Fläche X wie in (2.3) und Parameterwerte*

$$U := \{p_l \in D \,|\, l = 0, \ldots, m-1\}$$

gegeben. Dann bezeichne

$$(u,v) := \underset{0\leq l\leq m-1}{argmin} \|X(p_l) - P\|_2^2 \tag{2.18}$$

den Startwert *für die Newton-Methode.*

Parameterkorrektur

Es kann vorkommen, dass der Parameterwert $x_{i+1} \in \mathbb{R}^2$ nach einem Schritt der Newton-Methode aus (2.16) außerhalb des Parametergebietes D liegt:

1. Bei einer schlechten Wahl des Startwertes beziehungsweise zu groben Abtastung der Spline-Fläche,
2. bei einer nicht eindeutigen Lösung der quadratischen Funktion F' und
3. falls der anzunähernde Punkt P sehr dicht am Rand der Fläche oder außerhalb der Fläche liegt.

Da sich die Bisektionsmethode nicht so einfach auf das Mehrdimensionale übertragen lässt, kann man in so einem Fall mit dem vorherigen Startwert ein *Gradientenverfahren* benutzen (Nocedal und Wright [26]). Da Gradientenverfahren im Allgemeinen wesentlich langsamer sind als die Newton-Methode, wird in dieser Arbeit ein anderer Ansatz verfolgt.

Definition 2.25. *Es sei eine B-Spline-Fläche X wie in (2.3) gegeben. Dann ist ∂D der* Rand von X im Parameterbereich *und*

$$\{X(u,v) \mid (u,v) \in \partial D\}$$

der Rand von X im Bildbereich.

Algorithmus 2.26. (Parameterkorrektur)

Eingabe: B-Spline-Fläche X wie in (2.3), Parameterwert x_i vor Newton-Iteration (2.16)

Ausgabe: Parameterwert x_{i+1} nach korrigierter Newton-Iteration

1. Berechne $x_{i+1} := x_i - H_F^{-1}\nabla F$.
2. Ist $x_{i+1} \notin D$:
   - Suche $\alpha \in [0,1)$ mit
     $$x_i - \alpha H_F^{-1}\nabla F \in \partial D.$$
   - Setze $x_{i+1} := x_i - \alpha H_F^{-1}\nabla F$.

Bemerkung 2.27. Man sollte die Notation nicht mit einer Schrittweitensteuerung α wie bei den Line-Search-Methoden verwechseln (siehe Nocedal und Wright [26, S. 48]). Es wird zwar die Schrittweite der Newton-Methode korrigiert, aber keine Minimierung der Funktion $G(\alpha) := x_i - \alpha H_F^{-1}\nabla F$ vorgenommen.

Lemma 2.28. *Der Faktor α aus Algorithmus 2.26 existiert immer.*

Beweis: Angenommen nicht. Da $x_i \in D$, ist

$$x_i - \alpha H_F^{-1} \nabla F \in D, \quad \alpha \in [0,1).$$

Daraus folgt $x_i - H_F^{-1} \nabla F \in D$ oder $x_i - H_F^{-1} \nabla F \in \partial D$. Dies ist ein Widerspruch zur Voraussetzung. □

Man iteriert nun erneut mit diesem neuen Startwert x_{i+1}. Wird der Parameterbereich D erneut verlassen, bricht die Newton-Methode ab und der Punkt wird der äußeren Trimmkurve $\mathfrak{C}(X)$ zugeordnet (siehe Abschnitt 1.4). Ansonsten wird wie gewohnt weiter iteriert.

Teil III
IGES-Daten

Initial Graphics Exchange Specification (IGES) ist ein standardisiertes Format, um Daten aus einem *Computer Aided Design* (*CAD*) System zu speichern. Ein großer Vorteil von IGES ist, dass die Daten in reinem ASCII-Format vorliegen, so dass man sie mit jedem Editor betrachten kann, zusätzlich aber auch ein leichtes Auslesen der CAD-Daten gewährleistet ist.

Im vorherigen Kapitel wurden B-Spline-Kurven und Tensor-Produkt-Spline-Flächen vorgestellt, in welche die Elemente der IGES-Datei in diesem Kapitel konvertiert werden sollen. Die Information zum Dateiaufbau und die Entity-Nummern können dem IGES-Handbuch [23] entnommen werden.

1 Konvertierung

In der IGES-Datei stehen verschiedene Kurven- und Flächentypen, zum Beispiel Kreisbögen, Geraden, Ebenen, Translationsflächen, Regelflächen und viele mehr. Einige von ihnen werden in diesem Abschnitt dargestellt und die Konvertierung in B-Spline-Kurven beziehungsweise Tensor-Produkt-Flächen beschrieben.

1.1 Entity 100 – Kreisbogen

Definition 1.1. *Es seien die Werte* $z, x_m, y_m, x_s, y_s, x_e, y_e \in \mathbb{R}$ *gegeben. Definiere* Mittelpunkt $M := (x_m, y_m)$, Startpunkt $S := (x_s, y_s)$, Endpunkt $E := (x_e, y_e)$, Radius

$$r := \|S - M\| = \|E - M\| \geq 0$$

und die Drehwinkel

$$\alpha_s := \arccos\left(\frac{x_s - x_m}{r}\right), \quad \alpha_e := \arccos\left(\frac{x_e - x_m}{r}\right) \in [0, 2\pi).$$

Ist $y_s - y_m < 0$*, setze* $\alpha_s := 2\pi - \alpha_s$*. Ist* $y_e - y_m < 0$*, setze* $\alpha_e := 2\pi - \alpha_e$*. Ist* $\alpha_e = \alpha_s$*, setze* $\alpha_e := 2\pi + \alpha_e$*. Dadurch ist ein zur* x/y*-Ebene paralleler* Kreisbogen

$$K(t) := \begin{pmatrix} x_m + r\cos t \\ y_m + r\sin t \\ z \end{pmatrix}, \quad t \in [\alpha_s, \alpha_e] \tag{1.1}$$

definiert.

Die Angabe des Kreisbogens ist damit eine Besonderheit, da die Koordinatendaten nur lokal in einer zur x/y-Ebene parallelen Ebene angegeben und für die spätere Darstellung erst mit einer Rotationsmatrix gedreht und dann um einen Translationsvektor verschoben werden.

Bei der Extrahierung erfordern Kreisbögen mehr Aufwand, da diese zum einen bisher nur durch drei Punkte definiert sind und zum anderen weil ein polynomialer Spline einen Kreisbogen nicht exakt darstellen kann (Farin [10, S. 239]). Es muss daher auf rationale Splines zurückgegriffen werden.

Idee

Jeder offene Kreisbogen lässt sich als Bézier-Kurve

$$X(u) := \frac{d_0 w_0 B_{0,2} + d_1 w_1 B_{1,2} + d_2 w_2 B_{2,2}}{w_0 B_{0,2} + w_1 B_{1,2} + w_2 B_{2,2}}, \tag{1.2}$$

mit den Bernstein-Polynomen

$$B_{0,2} = (1-u)^2, \quad B_{1,2} = 2u(1-u), \quad B_{2,2} = u^2 \tag{1.3}$$

darstellen (siehe Farin [11]). Für einen Kreis mit Startpunkt d_0, Endpunkt d_2 und Winkel $\alpha < 2\pi$ kann man aufgrund der Symmetrieeigenschaft des Kreises den mittleren Kontrollpunkt d_1 und die Gewichte direkt angeben. Es ist dann

$$d_1 := \frac{l}{\|d_0 + d_2\|}(d_0 + d_2), \quad w_0 := 1, \quad w_1 := \cos\left(\frac{\alpha}{2}\right), \quad w_2 := 1 \tag{1.4}$$

mit $l = \frac{r}{w_1}$ und Kreisradius r.

Wichtig ist, dass man nur jeden *offenen Kreisbogen* (Winkel $< 2\pi$) als rationale C^2-stetige Bézier-Kurve darstellen kann. Aus diesem Grund unterteilt man einen ganzen Kreis in mehrere Segmente (genauer Bézier-Kurven) und fügt diese aneinander. Diese Splinedarstellung ist dann immerhin noch C^1-stetig (Farin [10, S. 222]).

Umsetzung

Bei der Erstellung eines Kreissplines werden die drei definierenden Punkte (Start-, End- und Mittelpunkt) übergeben. Die Richtung der Rotationsachse gibt an, ob die Drehung gegen oder mit dem Uhrzeigersinn verläuft (Abbildung 1.1).

Algorithmus 1.2. (Winkelberechnung)
Eingabe: Startpunkt A, Endpunkt E, Mittelpunkt C, Rotationsachse N
Ausgabe: Winkel α

1. Berechne Radius

$$r := \|\widetilde{A}\| = \|\widetilde{E}\|, \quad \widetilde{A} := A - C, \quad \widetilde{E} := E - C.$$

2. Berechne Winkel α zwischen $\widetilde{A}$ und $\widetilde{E}$:

$$\alpha := \cos\left(\frac{(\widetilde{A}, \widetilde{E})}{\|\widetilde{A}\| \cdot \|\widetilde{E}\|}\right) = \cos\left(\frac{(\widetilde{A}, \widetilde{E})}{r^2}\right).$$

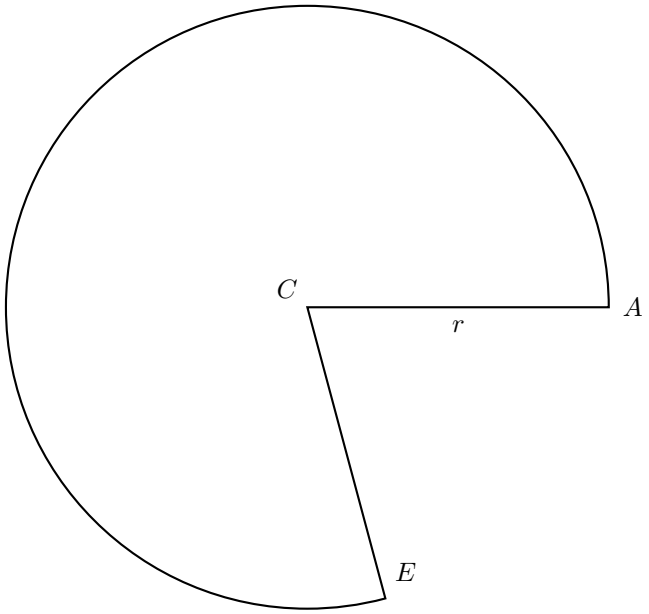


Abbildung 1.1: Kreisbogen

3. Ist $\|\widetilde{A} \times \widetilde{E} + N\| \leq \epsilon, \epsilon \geq 0$, setze $\alpha := 2\pi - \alpha$.

Nach der Winkelberechnung kann man den Kreis in Segmente mit einem Winkel $\xi < 2\pi$ unterteilen und die Kontrollpunkte bestimmen. Der Grund für die Unterteilung mit einem festen Wert ξ liegt in der leichteren Umformung von geometrischer Parametrierung in Spline-Parametrierung (siehe Abschnitt 1.3.1). Die Berechnung wird mittels einer rekursiv definierten Funktion durchgeführt, welche Startpunkt V, Endpunkt W, sowie den Winkel α des Kreisbogens übergeben bekommt.

Algorithmus 1.3. (Kreisbogenberechnung)
Eingabe: Startpunkt V, Endpunkt W, Winkel $\alpha \leq 2\pi$
Ausgabe: B-Spline-Kurve X, die den Kreis beschreibt

1. Ist $\alpha > \xi$:
   - Drehe V um ξ und erhalte damit einen Zwischenpunkt Z.
   - Berechne zwei B-Spline-Kurven von V nach Z mit Winkel ξ und von Z nach W mit Winkel $\alpha - \xi$. Diese werden dann gemäß Algorithmus 1.44 aus Teil II zusammengefügt.
2. Ansonsten ist $\alpha \leq \xi$:
   - Der Kreisbogen kann als Bézier-Kurve

$$C(t) := \frac{\sum_{i=0}^{2} w_i d_i B_{i,2}(t)}{\sum_{i=0}^{2} w_i B_{i,2}(t)}, \quad t \in [0,1],$$

mit

$$d_0 := V, \quad d_1 := \frac{l}{\|V+W\|}(V+W), \quad d_2 := W,$$

$$w_0 := 1, \quad w_1 := \cos\left(\frac{\alpha}{2}\right), \quad w_2 := 1$$

und $l := \frac{r}{w_1}$ dargestellt werden (Abbildung 1.2). Die genaue Berechnung findet man in Anhang A.

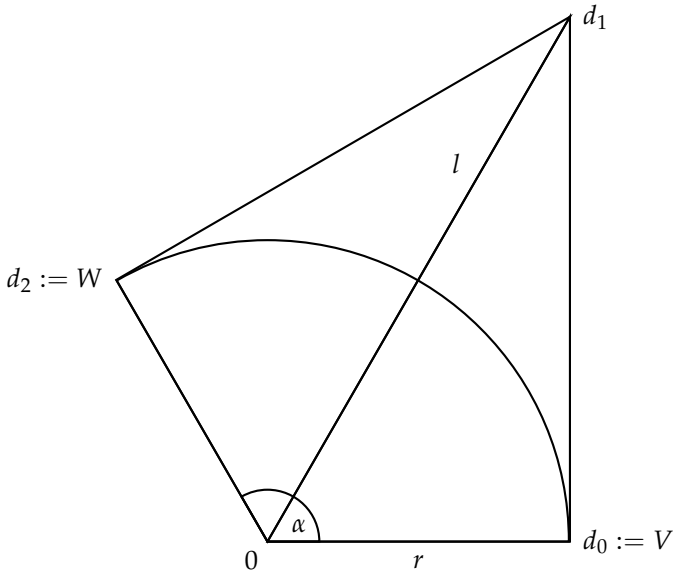


Abbildung 1.2: „Externer" Kontrollpunkt

Bemerkung 1.4. Der Winkel $\xi < 2\pi$ kann prinzipiell frei gewählt werden. Der Kontrollpunkt d_1 liegt immer auf der Winkelhalbierenden von d_0 und d_2 (ohne Berücksichtigung der Orientierung) und ist der Schnittpunkte der Tangenten am ersten und letzten Knoten. Für $\xi < \pi$ ist $0 < w_1 < 1$, für $\xi = \pi$ ist $w_1 = 0$ und man muss einen „unendlichen" Kontrollpunkt einfügen (Farin [10, S. 223]) und für $\pi < \xi < 2\pi$ ist $-1 < w_1 < 0$. Da in dieser Arbeit $\xi = \frac{2}{3}\pi$ oder $\xi = \frac{1}{2}\pi$, ist $0 < w_1 < 1$ (siehe Abschnitt 1.3.1).

1.2 Entity 118 – Regelfläche

Regelflächen entstehen, indem man zwei Splinekurven im Raum in jedem Punkt durch eine Gerade verbindet (Piegl und Tiller [27, S. 337 ff]).

Definition 1.5. (aus IGES [23, S. 104]) *Es seien zwei B-Spline-Kurven C_1 und C_2 mit gleicher Parametrierung (entweder nach Bogenlänge oder nach Parameterwerten) auf dem Gebiet $[a,b]$ beziehungsweise $[c,d]$ gegeben. Dann ist durch*

$$X(u,v) := (1-v)C_1(s) + vC_2(t), \quad s \in [a,b],\ t \in [c,d],\ u,v \in [0,1], \tag{1.5}$$

mit

$$\begin{aligned} s &= a + u(b-a) \\ t &= c + u(d-c) \end{aligned}$$

eine Regelfläche *definiert (Abbildung 1.3). Ist $s = t$ für alle $u \in [0,1]$, dann sind C_1 und C_2 nach gleichen Parameterwerten parametriert.*

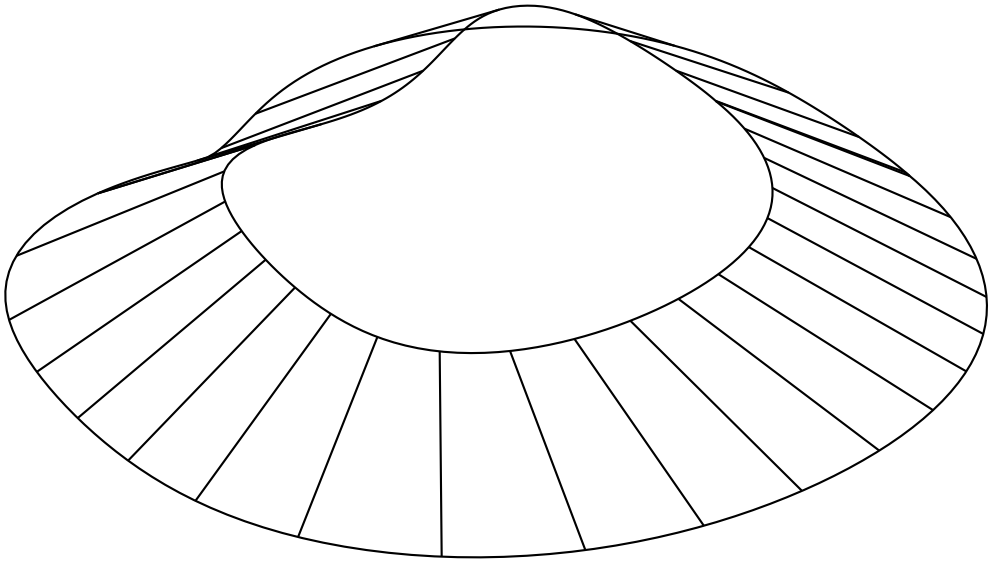


Abbildung 1.3: Entity 118 – Regelfläche

Konvention 1.6. Es sollen hier nur Regelflächen betrachtet werden, deren Kurven nach gleichen Parameterwerten parametriert sind.

Bei einer Regelfläche extrahiert man zuerst die beiden Splinekurven. Um ein identisches Kontrollpunktegitter zu erhalten, müssen die beiden Kurven den gleichen Knotenvektor erhalten. Danach verbindet man die Kurven linear.

Algorithmus 1.7. (Regelflächenkonvertierung)

Eingabe: B-Spline-Kurve C_1 der Ordnung k_1 mit Kontrollpunkten d'_i, Gewichten w'_i, $i = 0,\ldots,n_1-1$, und Knotenvektor $T_1 := [u_i^{\prime(r_i)}, i = 0,\ldots,m_1-1]$, B-Spline-Kurve C_2 der Ordnung k_2 mit Kontrollpunkten d^*_j, Gewichten w^*_j, $j = 0,\ldots,n_2-1$, und Knotenvektor $T_2 := [v_j^{\prime(s_j)}, j = 0,\ldots,m_2-1]$, r_i und s_j geben die Vielfachheit eines Knotens an

Ausgabe: B-Spline-Fläche X, die die Regelfläche beschreibt

1. Parametriere beide Knotenvektoren identisch. Setze dazu

$$v'_j := v'_0 + (v'_j - v'_0)\frac{u'_{m_1-1} - u'_0}{v'_{m_2-1} - v'_0}, \quad j = 0,\ldots,m_2-1.$$

(T_1 wird als Referenzknotenvektor benutzt.)

2. Setze gemäß Algorithmus 1.34 aus Teil II die Ordnung von C_1 und C_2 auf $k_u := \max(k_1, k_2)$. Dadurch ändern sich gegebenenfalls die Vielfachheiten r_i und s_j.

3. Erstelle einen neuen Knotenvektor

$$\begin{aligned} T_u \quad := \quad & [u_0^{(q_0)}, \ldots, u_{m-1}^{(q_{m-1})} \mid u_i \in T_1 \cup T_2,\ q_i := \max(r^*, s^*), \\ & r^* \geq 0 \text{ Vielfachheit von } u_i \text{ in } T_1, s^* \geq 0 \text{ Vielfachheit von } u_i \text{ in } T_2] \end{aligned}$$

4. Füge gemäß Algorithmus 1.24 aus Teil II in T_1 beziehungsweise in T_2 so viele Knoten ein, dass $T_1 = T_2 = T_u$. Es ändern sich dadurch gegebenenfalls die Anzahl n_1, n_2 der Kontrollpunkte und Gewichte.

5. Es ist nun $n := n_1 = n_2$. Setze

$$d_{i,0} := d_i', \quad d_{i,1} := d_i^*, \quad w_{i,0} := w_i', \quad w_{i,1} := w_i^*, \quad i = 0, \ldots, n-1.$$

6. Erstelle eine Tensor-Produkt-Spline-Fläche

$$X(u,v) = \frac{\sum_{i=0}^{n-1} \sum_{j=0}^{1} w_{i,j} d_{i,j} N_{i,k_u}(u|T_u) N_{j,2}(v|T_v)}{\sum_{i=0}^{n-1} \sum_{j=0}^{1} w_{i,j} N_{i,k_u}(u|T_u) N_{j,2}(v|T_v)} \tag{1.6}$$

mit $T_v := [0,0,1,1]$ beziehungsweise $T_v := [0,0,l_v,l_v]$, wobei

$$l_v := \max_{i=0}^{n-1} \|d_{i,1} - d_{i,0}\|.$$

Beispiel 1.8. Ein Beispiel soll zeigen, was in den ersten vier Schritten geschieht. Es sei $T_1 := [0,0,0,2,3,4,4,4]$ und $T_2 := [0,0,0,0,0.5,1,1,1,2,2,2,2]$.

1. T_1 soll als Referenz dienen, folglich muss man T_2 umparametrieren zu $T_2 = [0,0,0,0,1,2,2,2,4,4,4,4]$.

2. Die Ordnung der Splines ist 3 beziehungsweise 4. Setzt man beide auf 4, bleibt T_2 gleich und T_1 wird zu $T_1 = [0,0,0,0,2,2,3,3,4,4,4,4]$.

3. Der neue Knotenvektor T ist dann $T = [0,0,0,0,1,2,2,2,3,3,4,4,4,4]$.

4. In T_1 müssen die Knoten $[1,2]$ eingefügt werden und in T_2 die Knoten $[3,3]$.

Man erhält am Ende zwei Splines der Ordnung 4 mit Knotenvektoren $T_1 = T_2 = [0,0,0,0,1,2,2,2,3,3,4,4,4,4]$ und je 10 Kontrollpunkten.

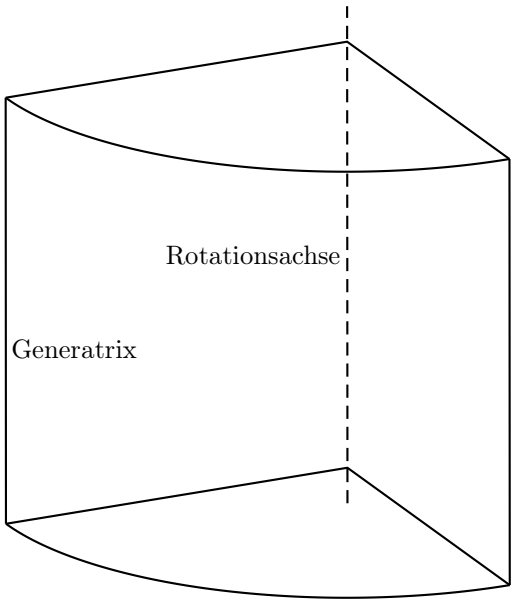


Abbildung 1.4: Entity 120 – Rotationsfläche

1.3 Entity 120 – Rotationsfläche

Rotationsflächen sind durch eine Rotationsachse (eine Gerade, Entity-Nummer 100), eine Kurve (*Generatrix*) sowie Start- und Endwinkel der Drehung gegeben (Abbildung 1.4). Die Generatrix wird dabei gegen den Uhrzeigersinn vom Start- zum Endwinkel um die Rotationsachse gedreht (Piegl und Tiller [27, S. 340 ff]). Aus diesen Daten kann man die Rotationsfläche konstruieren.

Algorithmus 1.9. (Rotationsflächenkonvertierung)

Eingabe: Generatrix C der Ordnung k_u mit Kontrollpunkten d'_i, Gewichten $w'_i, i = 0, \ldots, n_u - 1$, und Knotenvektor T_u, Rotationsachse $v_a = (x, y, z)^T$, Startwinkel α, Endwinkel γ

Ausgabe: B-Spline-Fläche X, die die Rotationsfläche beschreibt

1. Bestimme aus dem Startwinkel α und der Rotationsachse v_a eine Rotationsmatrix

$$R_{v_a,\alpha} := \begin{pmatrix} tx^2 + c & txy + sz & txz - sy \\ txy - sz & ty^2 + c & tyz + sx \\ txz + sy & tyz - sx & tz^2 + c \end{pmatrix} \tag{1.7}$$

   mit $c := \cos\alpha, s := \sin\alpha$ und $t := 1 - \cos\alpha$ (siehe Anhang B).

2. Setze $d'_i := R_{v_a,\alpha} d_i$, $i = 0, \ldots, n_u - 1$. Dadurch wird die Kurve C auf die korrekte Startposition gedreht.

3. Bestimmte den Drehwinkel $\beta := \gamma - \alpha$ als Differenz zwischen End- und Startwinkel und erstelle eine neue Rotationsmatrix $R_{v_a,\beta}$ wie in (1.7).

4. Schleife i von 0 bis $n_u - 1$:

   - Setze $v_s := d'_i$.
   - Berechne Projektionspunkt v_c von v_s auf der Rotationsachse.
   - Berechne den Endpunkt der Drehung $v_e := v_c + R_{v_a,\beta}(v_s - v_c)$. (Die Verschiebung ist dabei wichtig, da die Rotation um die Achse geschieht.)
   - Berechne aus v_s, v_e, v_c und v_a gemäß Algorithmus 1.3 den Kreisspline

     $$S(v) = \frac{\sum_{j=0}^{n_v-1} w_j^* d_j^* N_{j,k_v}(v|T_v)}{\sum_{j=0}^{n_v-1} w_j^* N_{j,k_v}(v|T_v)}$$

     der Ordnung k_v mit Knotenvektor T_v.
   - Setze

     $$d_{i,j} := d_j^*, \quad w_{i,j} := w'_i w_j^*, \quad j = 0, \ldots, n_v - 1.$$

5. Erstelle Tensor-Produkt-Spline-Fläche

$$X(u,v) = \frac{\sum_{i=0}^{n_u-1} \sum_{j=0}^{n_v-1} w_{i,j} d_{i,j} N_{i,k_u}(u|T_u) N_{j,k_v}(v|T_v)}{\sum_{i=0}^{n_u-1} \sum_{j=0}^{n_v-1} N_{i,k_u}(u|T_u) N_{j,k_v}(v|T_v)}. \tag{1.8}$$

Bemerkung 1.10. Man sollte beachten, dass k_v, n_v und T_v in jedem Schleifendurchlauf identisch sind, da diese nur vom Winkel β abhängen und nicht von Start-, End- oder Mittelpunkt des Kreisbogens. (Ausnahmen sind entartete Kreisbögen, bei denen $v_s = v_e = v_c$ gilt.)

1.3.1 Parametrierung

Extrahiert man die Rotationsfläche auf diese Art, stößt man bei den Trimmkurven im Parameterbereich auf das Problem der Parametrierung. Die Kurven in der IGES-Datei sind immer bezüglich der geometrischen Darstellung einer Rotationsfläche über den Sinus und Kosinus definiert. Die Durchlaufgeschwindigkeit ist aber verschieden von der der Splinedarstellung und damit nicht mehr kompatibel (Farin [10, S. 222 f]). Man muss zuerst alle gegebenen Parameterwerte der IGES-Datei zu einer Rotationsfläche auf den Knotenvektor transferieren.

Anhand eines (zweidimensionalen) Kreisbogens soll die Problematik verdeutlicht werden. Hierzu sei der Einheitsviertelkreis auf folgende drei Arten definiert:

$$K(u) := \left(\cos\left(\frac{u\pi}{2}\right), \sin\left(\frac{u\pi}{2}\right)\right), \quad u \in [0,1], \tag{1.9}$$

$$X(u) := \left(\frac{(1-u)^2 + \sqrt{2}u(1-u)}{(1-u)^2 + \sqrt{2}u(1-u) + u^2}, \frac{\sqrt{2}u(1-u) + u^2}{(1-u)^2 + \sqrt{2}u(1-u) + u^2}\right), \quad u \in [0,1], \tag{1.10}$$

$$P(u) := \left(\frac{1-u^2}{1+u^2}, \frac{2u}{1+u^2}\right), \quad u \in [0,1]. \tag{1.11}$$

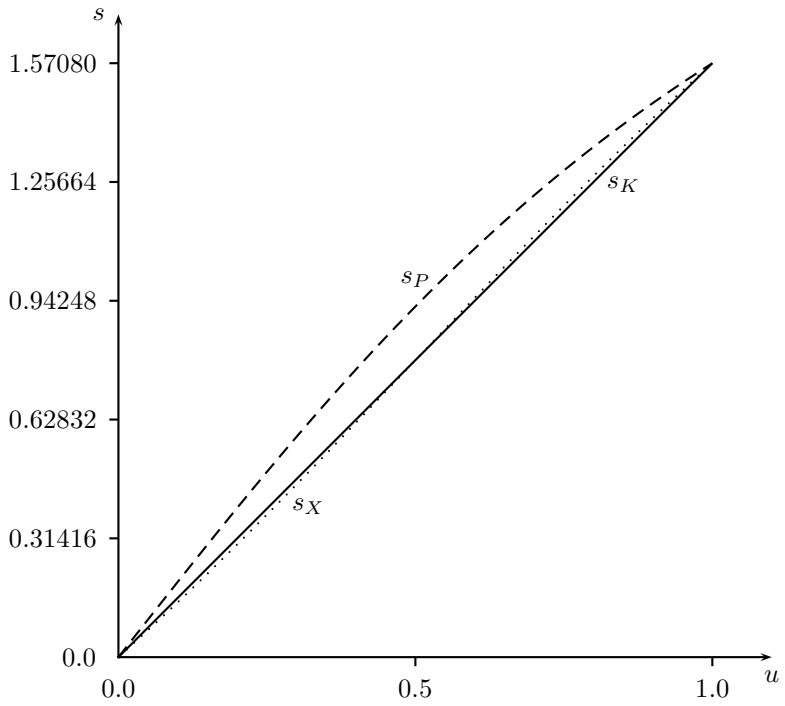


Abbildung 1.5: Verschiedene Bogenlängenfunktionen

(1.9) beschreibt die geometrische Darstellung, (1.10) die Darstellung als Bézier-Kurve und (1.11) die häufig genutzt Darstellung als rationales Polynom. (Man erhält P auch aus der Darstellung von X, wenn man die Gewichte $w_0 = 1, w_1 = 1, w_2 = 2$ setzt.)

Abbildung 1.5 zeigt die drei Bogenlängenfunktionen

$$s_K(u) := \frac{u\pi}{2}, \quad u \in [0,1], \tag{1.12}$$

$$s_X(u) := 2\arctan\left(\frac{2u-1}{1+\sqrt{2}}\right) + \frac{\pi}{4}, \quad u \in [0,1], \tag{1.13}$$

$$s_P(u) := 2\arctan(u), \quad u \in [0,1]. \tag{1.14}$$

Die Funktion s_K ist linear, da die Durchlaufgeschwindigkeit bei der geometrischen Darstellung (1.9) konstant ist. Die Bogenlängenfunktion s_X ist ähnlich zu s_K, aber aufgrund der symmetrischen Gewichte (1.10) nur an den Stellen $u = 0, u = 0.5$ und $u = 1$ identisch. Vor allem P hat eine vollkommen andere Durchlaufgeschwindigkeit. Die B-Spline-Darstellung (1.10) ist bei der Darstellung eines Kreisbogen also zu bevorzugen, da sie gegenüber Darstellung (1.11) den geometrischen Kreisbogen (1.9) im Hinblick auf die Bogenlänge besser beschreibt.

Man benötigt nun eine Funktion, die jeden Parameterwert der geometrischen Darstellung auf den der Splinedarstellung transformiert, was der Umkehrfunktion s_X^{-1} entspricht. Der große Vorteil dieser Methode ist, dass man bei der Abstandsberechnung von Punkten (siehe Teil II, Abschnitt 1.4) leicht den Parameterwert der geometrischen Darstellungen berechnen kann und dann diesen Wert nur in den Parameterbereich des Splines transferieren muss. In

Teil II, Abschnitt 1.2.4 wurde gezeigt, wie man die s_X^{-1} über die Sehnenlänge approximieren kann. Im Falle von Kreisbögen lässt sich die Umkehrfunktion aber zumindest für die beiden Winkel $\frac{\pi}{2}$ und $\frac{2\pi}{3}$ für ξ geschlossen berechnen. (Es gibt gegebenenfalls noch weitere.) Aus diesem Grund werden Rotationsflächen in Richtung der Rotation immer über ein Vielfaches von ξ beschrieben, da ansonsten die Umrechnung nicht mehr möglich ist.

Bemerkung 1.11. Ein Winkel von $\frac{2\pi}{3}$ hat den Vorteil, dass die Rotationsfläche aus maximal drei Segmenten besteht, wogegen es bei $\frac{\pi}{2}$ maximal vier sind. Dagegen passiert es bei $\frac{2\pi}{3}$ häufiger, dass ein unnötiger „Überhang“ entsteht. Dies kommt vor, wenn die Trimmkurve im Parameterbereich zum Beispiel bis $\alpha = \frac{241\pi}{180}$ definiert ist. Mit $\xi = \frac{2\pi}{3}$ muss die Fläche bis 2π definiert werden, da $2\xi < \alpha \leq 3\xi$, und man hat $2\pi - \alpha$ an „Überhang“. Mit $\xi = \frac{\pi}{2}$ wäre die Fläche nur bis $\frac{3\pi}{2}$ definiert und der „Überhang“ mit $\frac{3\pi}{2} - \alpha$ um $\frac{\pi}{2}$ geringer.

Die Bogenlängenfunktion (1.23) aus Teil II lässt sich schreiben als

$$s_X(u) = \int \|X'(u)\|\, du = \begin{cases} 2\arctan\left(\frac{1}{\sqrt{2}+1}(2u-1)\right) + \frac{\pi}{4}, & \xi = \frac{\pi}{2}, \\ 2\arctan\left(\frac{\sqrt{3}}{3}(2u-1)\right) + \frac{\pi}{3}, & \xi = \frac{2\pi}{3}. \end{cases} \tag{1.15}$$

Die Umkehrabbildung (nach u aufgelöst) ist dann

$$s_X^{-1}(s) = \begin{cases} \frac{\sqrt{2}+1}{2}\tan\left(\frac{4s-\pi}{8}\right) + \frac{1}{2}, & \xi = \frac{\pi}{2}, \\ \frac{\sqrt{3}}{2}\tan\left(\frac{3s-\pi}{6}\right) + \frac{1}{2}, & \xi = \frac{2\pi}{3}, \end{cases} \tag{1.16}$$

mit $s \in [0, \xi]$, $s_X^{-1} \in [0,1]$.

Die Umkehrfunktion s_X^{-1} ist damit nur auf dem Intervall $[0,\xi]$ definiert. Wie in Abschnitt 1.1 beschrieben, bestehen Kreisbögen größer ξ aus der Zusammensetzung von mehreren Bézier-Kurven. Für die Parameterwerte der gesamten Kurve in $[0,2\pi]$ beziehungsweise aus ganz $\mathbb{R}^+$ muss man eine extra Funktion definieren.

Definition 1.12. *Es seien ein Winkel $\alpha \in [0,2\pi]$ und die Funktion s_X^{-1} aus (1.16) auf dem Intervall $[0,\xi]$ gegeben. Dann definiere*

$$\widehat{s}_X(\alpha) := s_X^{-1}(\beta) + i, \quad \beta = \alpha - \lfloor\alpha\rfloor_\xi \in [0,\xi), \quad i = \frac{\lfloor\alpha\rfloor_\xi}{\xi} \in \mathbb{N},$$

$$\lfloor\alpha\rfloor_\xi := \max\{\, i\xi \mid i\xi \leq \alpha, i \in \mathbb{N}\}. \tag{1.17}$$

Korollar 1.13. *Es seien ein Kreisbogen K mit Winkel $0 < \gamma \leq 2\pi$ als B-Spline-Kurve X gemäß (1.4) und ein Winkel $0 \leq \alpha \leq \gamma$ gegeben. Dann gilt:*

$$K(\alpha) = X(\widehat{s}_X(\alpha)), \quad \alpha \in [0,\gamma]. \tag{1.18}$$

Beweis: Dies folgt sofort aus der Konstruktion der B-Spline-Kurve X und der Anwendung der Funktion s_X^{-1} aus (1.16) auf die einzelnen Bézier-Segmente der Kurve. □

Ein Problem ist noch, dass diese Umrechnung nur für Kreisbögen mit einer Bogenlänge von $i\xi, i \in \mathbb{Z}$, funktioniert. Kreisbögen müssen daher bei der Umwandlung in B-Spline-Kurven immer auf ein Vielfaches von ξ erweitert werden. Damit sich der Kreisbogen beziehungsweise die Rotationsfläche bei dieser Erweiterung nicht selbst überlappt, wird sie so umparametriert, dass der Startwinkel α immer bei 0 liegt. Diese Rotation wird im Parameterbereich durch eine einfache Verschiebung in entgegensetzte Richtung (also $-\alpha$) erreicht.

1.4 Entity 122 – Translationsfläche

Eine Translationsfläche entsteht, wenn man ein Geradenstück (Generatrix) entlang einer Kurve (Directrix) verschiebt (Piegl und Tiller [27, S. 334 ff]). Die Generatrix fängt dabei immer im Ursprung eines lokalen Koordinatensystems an und endet an einem lokalen Endpunkt. Danach wird das lokale System so in das globale verschoben, dass der lokale Ursprung mit dem Startpunkt der Directrix-Kurve übereinstimmt (Abbildung 1.6).

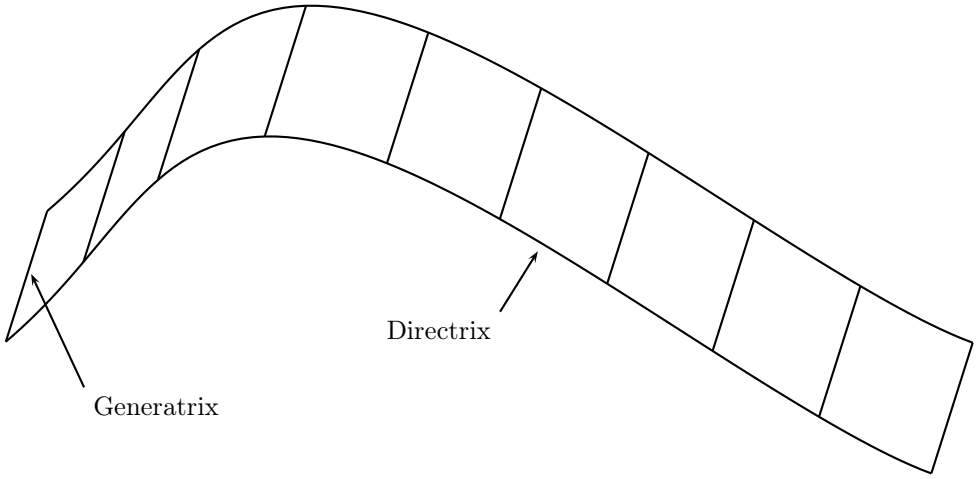


Abbildung 1.6: Entity 122 – Translationsfläche

Definition 1.14. (aus IGES [23, S. 110]) *Es seien eine B-Spline-Kurve C (*Directrix*) auf dem Parametergebiet $[t_0, t_{n+k-1}]$ und ein Vektor V (*Generatrix*) gegeben. Dann ist durch*

$$X(u,v) := C(t) + v(V - C(t_0)), \quad t = t_0 + u(t_{n+k-1} - t_0),\ u,v \in [0,1], \tag{1.19}$$

eine Translationsfläche *definiert.*

Translationsflächen sind einfach zu handhaben, da es sich nur um einen Sonderfall der B-Spline-Flächen handelt.

Algorithmus 1.15. (Translationsflächenkonvertierung)

Eingabe: B-Spline-Kurve C (*Directrix*) der Ordnung k_u mit Kontrollpunkten d'_i, Gewichten $w'_i, i = 0, \ldots, n_u - 1$, und Knotenvektor $T_u := [u_0, \ldots, u_{n_u+k_u-1}]$, Vektor V (*Generatrix*)

Ausgabe: B-Spline-Fläche X, die die Translationsfläche beschreibt

1. Setze
$$d_{i,j} := d'_i + j(V - d'_0), \quad w_{i,j} := w'_i, \quad i = 0, \ldots, n_u - 1,\ j = 0, 1.$$
2. Erstelle B-Spline-Fläche
$$X(u,v) = \frac{\sum_{i=0}^{n_u-1} \sum_{j=0}^{1} w_{i,j} d_{i,j} N_{i,k_u}(u|T_u) N_{j,2}(v|T_v)}{\sum_{i=0}^{n_u-1} \sum_{j=0}^{1} N_{i,k_u}(u|T_u) N_{j,2}(v|T_v)} \tag{1.20}$$
mit $T_v := [0,0,1,1]$ beziehungsweise $T_v := [0,0,\|V\|,\|V\|]$

2 Trimmkurven

Um später entscheiden zu können, ob ein Punkt innerhalb der getrimmten Fläche liegt, muss man den Parameterbereich der Trimmkurve bestimmen. In der IGES-Datei sind diese Kurven nicht immer gegeben, so dass sie selbst berechnet werden müssen, oder sie haben kleinere Fehler wie zum Beispiel Überschneidungen und Lücken, die erst ausgebessert werden müssen.

Die Idee ist folgende: Ist die Trimmkurve im Parameterbereich nicht gegeben, tastet man sie im Bildbereich fein ab, berechnet dann zu diesen Punkten mittels Abstandsberechnung (siehe Teil IV, Abschnitt 4) die zugehörigen Parameterwerte und approximiert aus diesen Werten eine zweidimensionale Splinekurve. Je nach Art der Fläche wählt man hierbei einen leicht anderen Ansatz.

Konvention 2.1.

1. Bei der Angabe der Trimmkurven handelt es sich immer um zusammengesetzte Kurven $\mathfrak{C} := \{C^{(j)} \mid j = 0, \ldots, m-1\}$ im Bildbereich. Der Einfachheit halber wird in den Algorithmen aber nur eine Kurve $\mathfrak{C}$ betrachtet.
2. Wird von einer Trimmkurve $\mathfrak{C}$ gesprochen, so seien automatisch Kontrollpunkte d'_j, Gewichte $w'_j, j = 0, \ldots, n'-1$, ein Knotenvektor $T' := \left[t'_0, \ldots, t'_{n'+k'-1}\right]$ und eine Ordnung k' gegeben.
3. Die Trimmkurve $\mathfrak{C}$ muss geschlossen sein.
4. Eine B-Splinefläche X ist gemäß (2.3) aus Teil II definiert, falls nicht anders angegeben.
5. Die Trimmkurve $\mathfrak{C}$ muss auf der B-Spline-Fläche X liegen. Das heißt:
$$\mathfrak{C}(t) \in X, \quad t \in \left[t'_0, t'_{n'+k'-1}\right].$$

2.1 Allgemeine Berechnung

Definition 2.2. *Es seien B-Spline-Kurven C_1 und C_2 gegeben, wobei $[a,b]$ das Parametergebiet von C_1 ist. Dann heißt C_1* in *C_2* enthalten*, falls*
$$C_1(t) \in C_2, \quad t \in [a,b]. \tag{2.21}$$
Man schreibt dann kurz $C_1 \subseteq C_2$.

Lemma 2.3. *Es sei eine injektive B-Spline-Fläche X auf dem Parametergebiet D gegeben. Zusätzlich seien zwei B-Spline-Kurven B_j auf dem Parameterbereich $[a_j, b_j]$ mit $B_j \in D, j = 1,2$, definiert. Sei $C_j(t) := X(B_j(t)), t \in [a_j, b_j], j = 1,2$. Dann gilt:*

$$B_1 \subseteq B_2 \Leftrightarrow C_1 \subseteq C_2. \tag{2.22}$$

Beweis: Sei $B_1 \subseteq B_2$. Angenommen, $C_1(t) \notin C_2$. Es ist nach Definition 2.2 $B_1(t) \in B_2, t \in [a_1, b_1]$. Folglich existiert ein $u \in [a_2, b_2]$ mit $B_1(t) = B_2(u)$. Es ist dann

$$C_1(t) = X(B_1(t)) = X(B_2(u)) = C_2(u).$$

Dies ist ein Widerspruch zu $C_1(t) \notin C_2$. Sei umgekehrt $C_1 \subseteq C_2$. Es ist $C_1(t) \in C_2, t \in [a_1, b_1]$. Dann existiert ein u mit $C_1(t) = C_2(u)$ und es ist

$$X(B_1(t)) = C_1(t) = C_2(u) = X(B_2(u)).$$

Da X nach Voraussetzung injektiv ist, gilt $B_1(t) = B_2(u)$ und damit Behauptung (2.22). □

Bemerkung 2.4. Die Voraussetzung einer injektiven B-Spline-Fläche ist nur in besonderen Fällen nicht erfüllt:

1. Bei geschlossenen B-Spline-Flächen. Hier ist die erste und letzte Kontrollpunktreihe identisch (siehe Abschnitt 2.3).
2. Bei entarteteten B-Spline-Flächen, bei denen eine Randkurve auf einen Punkt zusammenfällt (siehe Abschnitt 2.4).
3. Bei sich selbst schneidenden Flächen, die aber bei der CAD-Konstruktion nicht vorkommen können. Zu Selbstschnitten zählen auch einzelne Berührpunkte.

Diese Fälle werden getrennt behandelt.

Der Algorithmus für die allgemeine Berechnung der Trimmkurven ergibt sich nun direkt aus der Abstandsberechnung beziehungsweise Parametrierung aus Teil II, Abschnitt 2.6.

Algorithmus 2.5. (allgemeine Trimmkurven-Berechnung)

Eingabe: B-Spline-Fläche X mit Randkurve $\mathfrak{R} = \{R_i \mid i = 0, \ldots, 3\}$ gemäß (II.2.12), Trimmkurve $\mathfrak{C}$ im Bildbereich (Abbildung 2.1a)

Ausgabe: Trimmkurve $\mathfrak{B}$ im Parameterbereich D

1. Ist $\mathfrak{C} \subseteq R_i$ für ein $i \in \{0, \ldots, 3\}$, gilt dies auch für die Trimmkurven im Parameterbereich nach Lemma 2.3:
   - Bestimme $c_0, c_1 \in D$ mit

$$R_i(c_0) = d'_0, \quad R_i(c_1) = d'_{n-1}.$$

- Für $l = 0, 1$ setze

$$d_l^* := \begin{cases} (u_{k_u+n_u-1} - c_l, v_0), & i = 0, \\ (u_0, c_l), & i = 1, \\ (c_l, v_{k_v+n_v-1}), & i = 2, \\ (u_{k_u+n_u-1}, v_{k_v+n_v-1} - c_l), & i = 3. \end{cases}$$

- Erstelle B-Spline-Kurve

$$\mathfrak{B}(t) := \sum_{l=0}^{1} d_l^* N_{l,2}(t|T)$$

  mit $T := [0, 0, 1, 1]$.

sonst ist $\mathfrak{C} \not\subset R_i, i = 0, \ldots, 3$:

- Taste die Kurve $\mathfrak{C}$ möglichst gleichmäßig an Punkten $P_0, \ldots, P_{s-1}$ ab.
- Bestimme mittels der Abstandsberechnung Parameterwerte q_l mit $X(q_l) = P_l, l = 0, \ldots, s-1$.
- Approximiere Parameterwerte $q_l, l = 0, \ldots, s-1$, durch eine polynomiale B-Spline-Kurve $\mathfrak{B}$ (Abbildung 2.1b).

2.2 Ebenen

Da Ebenen in der IGES-Datei nur als Normalenvektor angegeben sind, werden die Tensor-Produkt-Spline-Flächen eigenständig erstellt. Dadurch lassen sich die Parameterwerte der Trimmkurve leicht durch folgenden Algorithmus bestimmen, ohne die Kurve abtasten zu müssen.

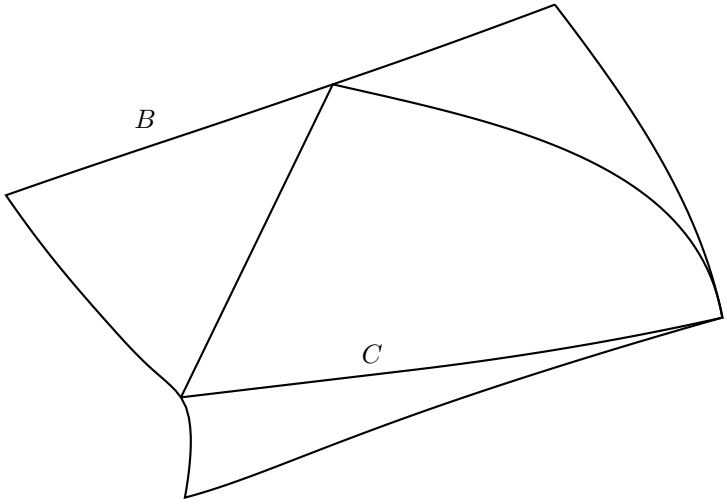
Algorithmus 2.6. (Trimmkurven-Berechnung für Ebenen)

Eingabe: ebene B-Spline-Fläche X mit Kontrollpunkten $d_{i,j} = \left(d_{i,j}^{(x)}, d_{i,j}^{(y)}, d_{i,j}^{(z)}\right), i, j = 0, 1$, Trimmkurve $\mathfrak{C}$ im Bildbereich

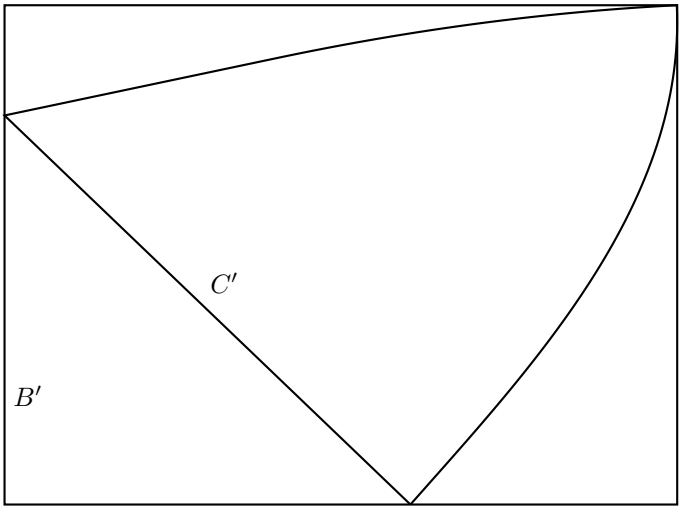
Ausgabe: Trimmkurve $\mathfrak{B}$ im Parameterbereich D

1. Ist Normalenvektor $\mathfrak{N}(u_0, v_0) = (0, 0, 1)$ oder $\mathfrak{N}(u_0, v_0) = (0, 0, -1)$, gehe zu Punkt 4.
2. Berechne Rotationsmatrix R, so dass X parallel zur x/y-Ebene gedreht werden kann (siehe Anhang B).
3. Drehe $\mathfrak{C}$ mittels Rotationsmatrix R (Anwendung auf die Kontrollpunkte) und setze $d_{i,j} := Rd_{i,j}$, $i, j = 0, 1$.
4. Berechne Seitenverhältnisse

$$l^{(x)} := \frac{\|d_{1,0} - d_{0,0}\|}{|u_{k_u+n_u-1} - u_0|}, \quad l^{(y)} := \frac{\|d_{0,1} - d_{0,0}\|}{|v_{k_v+n_v-1} - v_0|}.$$



a. Bildbereich



b. Parameterbereich

Abbildung 2.1: Flächenrandkurve und Trimmkurve

5. Sei $d'_j := \left(d'^{(x)}_j, d'^{(y)}_j, d'^{(z)}_j\right), j = 0, \ldots, n' - 1.$

6. Setze

$$\widetilde{d}_j := \left(u_0 + l^{(x)}\left(d'^{(x)}_j - d^{(x)}_{0,0}\right), v_0 + l^{(y)}\left(d'^{(y)}_j - d^{(y)}_{0,0}\right)\right), \quad j = 0, \ldots, n' - 1.$$

   Dies ist eine einfache lineare Transformation. (Die z-Komponente ist Null und kann ignoriert werden.)

7. Erstelle B-Spline-Kurve

$$\mathfrak{B}(t) := \frac{\sum_{j=0}^{n'-1} w'_j \widetilde{d}_j N_{j,k'}(t|T')}{\sum_{j=0}^{n'-1} w'_j N_{j,k'}(t|T')}.$$

2.3 Geschlossene Flächen

Geschlossene Flächen sind Flächen, deren erste und letzte Kontrollpunktreihe oder -spalte in einer oder zwei Richtungen übereinstimmen. Dies ist zum Beispiel bei Rotationsflächen mit einem Winkel von 2π der Fall.

Definition 2.7. *Es sei eine B-Spline-Fläche X gegeben. Die Fläche heißt* geschlossen in Richtung *u, falls*

$$X(u_0, v) = X(u_{n_u+k_u+1}, v), \quad v \in [v_0, \ldots, v_{n_v+k_v-1}]. \tag{2.23}$$

Da in dieser Arbeit nur B-Spline-Flächen mit der Endpunktinterpolationseigenschaft behandelt werden, bedeutet das, dass

$$d_{0,j} = d_{n_u-1,j}, \quad j = 0, \ldots, n_v - 1.$$

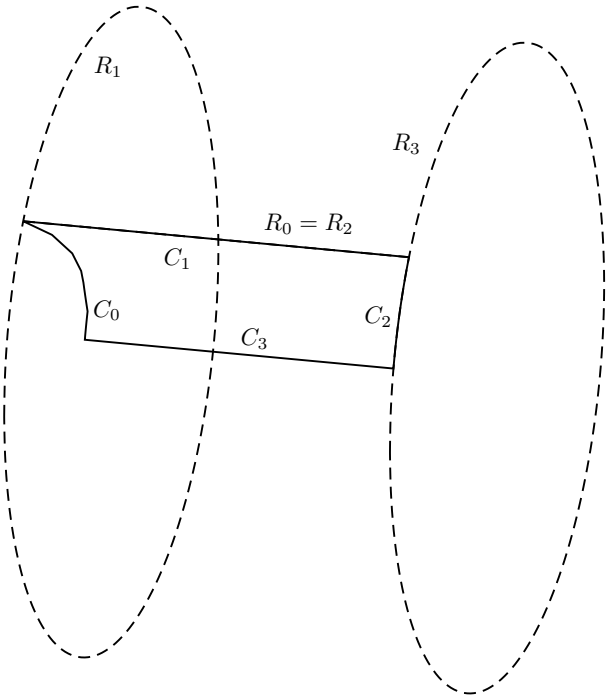
Diese Kontrollpunktreihe heißt Naht. *Analog dazu ist* geschlossen in Richtung *v definiert.*

Geschlossene Flächen erfordern einige Überlegungen, damit vor allem die Abstandsberechnung korrekt funktioniert. Bei der Umsetzung stößt man auf folgende Probleme:

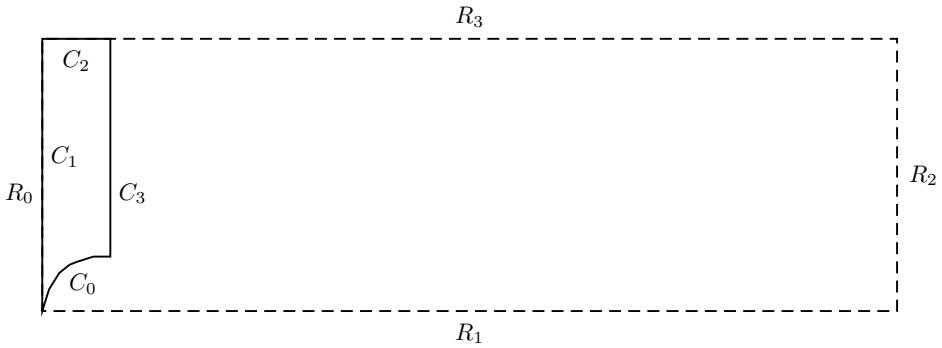
1. Trimmkurven, die auf der Naht der Fläche liegen, haben zwei Randseiten im Parameterbereich zur Auswahl. In Abbildung 2.2 ist Kurve C_1 identisch zu R_0 oder R_2. Bei mehreren zusammengesetzten Trimmkurven auf dieser Naht kann man sogar zwischen beiden Randseiten hin- und herspringen.

2. Trimmkurven, die über diese Naht verlaufen, sind im Bildbereich korrekt, im Parameterbereich verlassen sie aber das Intervall und treten auf der anderen Seite wieder ein.

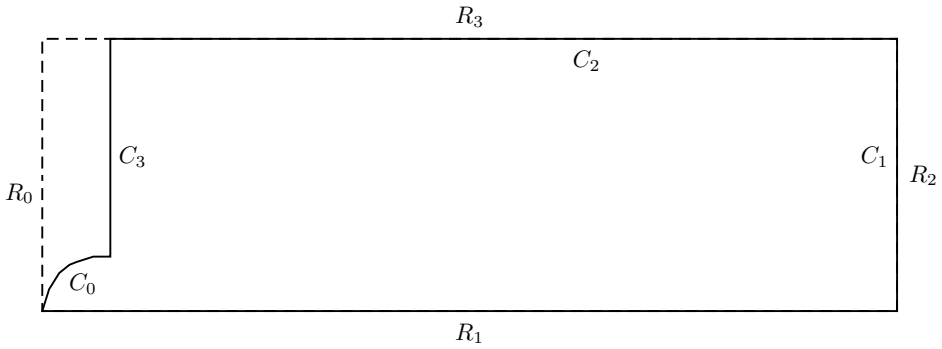
Um diese Probleme zu beheben, wird ein leicht anderer Ansatz als im allgemeinen Fall benutzt.



a. Rotationsfläche mit Trimmkurve



b. Korrekte Trimmkurve im Parameterbereich



c. Falsche Trimmkurve im Parameterbereich

Abbildung 2.2: Rotationsfläche und falsche Trimmkurve

2.3.1 Erweiterung geschlossener Flächen

Definition 2.8. *Es sei eine B-Spline-Fläche X gegeben, die in Richtung u geschlossen ist. Dann heißt*

$$X^*(u,v) := \frac{\sum_{i=0}^{3(n_u-1)} \sum_{j=0}^{n_v-1} w^*_{i,j} d^*_{i,j} N_{i,k_u}(u|T^*_u) N_{j,k_v}(v|T_v)}{\sum_{i=0}^{3(n_u-1)} \sum_{j=0}^{n_v-1} w^*_{i,j} N_{i,k_u}(u|T^*_u) N_{j,k_v}(v|T_v)} \tag{2.24}$$

zweiseitige stetige Fortsetzung von X in Richtung u, *wobei*

$$T^*_u := [u_0 - l_u, \ldots, u_{n_u-1} - l_u, u_1, \ldots, u_{n_u+k_u-2}, u_{k_u} + l_u, \ldots, u_{n_u+k_u-1} + l_u]$$

mit $l_u := u_{n_u+k_u-1} - u_0$ *und*

$$d^*_{3(n_u-1),j} = d_{0,j}, \quad d^*_{2(n_u-1)+i,j} = d^*_{(n_u-1)+i,j} = d^*_{i,j} = d_{i,j},$$
$$w^*_{3(n_u-1),j} = w_{0,j}, \quad w^*_{2(n_u-1)+i,j} = w^*_{(n_u-1)+i,j} = w^*_{i,j} = w_{i,j}$$

für $i = 0, \ldots, n_u - 2, j = 0, \ldots, n_v - 1$. *Analog definiert man* zweiseitige stetige Fortsetzung von X in Richtung v.

X^* ist somit nichts anderes als eine aus X dreimal zusammengesetzte Fläche.

Beispiel 2.9. Anhand eines zweidimensionalen Beispiels soll Definition 2.8 verdeutlicht werden. Es sei eine B-Spline-Kurve

$$K(t) := \frac{\sum_{i=0}^{n-1} w_i d_i N_{i,k}(t)}{\sum_{i=0}^{n-1} w_i N_{i,k}(t|T)}$$

mit $n := 9, k := 3, T := \left[0,0,0,\frac{2\pi}{3},\frac{2\pi}{3},\frac{4\pi}{3},\frac{4\pi}{3},2\pi,2\pi,2\pi\right]$ gegeben, die einen Kreis beschreibt. Man kann die Kurve K an beiden Seiten stetig (und sogar stetig-differenzierbar) als

$$K^*(t) := \frac{\sum_{i=0}^{3(n-1)} w^*_i d^*_i N_{i,k}(t|T^*)}{\sum_{i=0}^{3(n-1)} w^*_i N_{i,k}(t|T^*)}$$

fortsetzen, wobei

$$T^* := \left[-2\pi,-2\pi,-2\pi,-\frac{4\pi}{3},-\frac{4\pi}{3},-\frac{2\pi}{3},-\frac{2\pi}{3},0,0,\frac{2\pi}{3},\frac{2\pi}{3},\frac{4\pi}{3},\frac{4\pi}{3},2\pi,2\pi,\frac{8\pi}{3},\frac{8\pi}{3},\frac{10\pi}{3},\frac{10\pi}{3},4\pi,4\pi,4\pi\right],$$

$$d^*_{24} := d_0, \quad d^*_{16+i} = d^*_{8+i} = d^*_i := d_i,$$
$$w^*_{24} := w_0, \quad w^*_{16+i} = w^*_{8+i} = w^*_i := w_i, \quad i = 0, \ldots, 7.$$

Die B-Spline-Kurve K^* durchläuft den Kreis somit dreimal bei einem Durchlauf von $[0, 4\pi]$ (Abbildung 2.3).

Bemerkung 2.10. Es ist wichtig, dass man die geschlossene B-Spline-Fläche X nach beiden Seiten erweitert. Hat man Punkte $P_0, \ldots, P_{s-1}$ und liegt P_0 auf der Naht von X, kann man ohne nähere Betrachtung von $P_1, \ldots, P_{s-1}$ nicht sagen, nach welcher Seite sich die abgetasteten Punkte fortsetzen. Somit ist es nicht klar, ob man $p_0 := u_0$ oder $p_0 := u_{n_u+k_u-1}$ für $X(p_0) = P_0$ setzen soll. Eine Erweiterung nach beiden Seiten behebt dieses Problem. Wichtig hierfür ist allein, dass $p_0 \in [u_0, u_{n_u+k_u-1}]$.

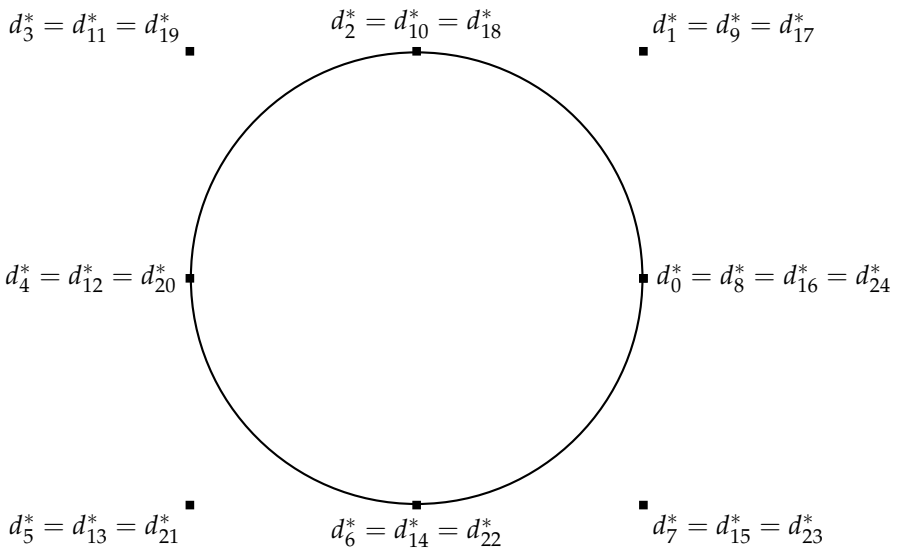


Abbildung 2.3: Erweiterter Kreis

2.3.2 Parameterkorrektur

Tastet man die Kurve an vielen Punkten ab und berechnet die Parameterwerte, kann es vorkommen, dass einzelne Punkte, die auf der Naht liegen, zwischen den Parametergrenzwerten hin und her springen (siehe Teil II, Abschnitt 1.4 auf Seite 32). Daraus resultiert Problem 1 oben.

Die Idee ist zu überprüfen, wie weit die Parameterwerte zu zwei aufeinanderfolgenden Punkten auseinanderliegen und den zweiten Wert gegebenenfalls zu korrigieren. Voraussetzung dafür ist, dass die abgetasteten Punkte dicht beieinander liegen.

Algorithmus 2.11.

Eingabe: erweiterte, geschlossene B-Spline-Fläche X^* wie in (2.24), Parameterwerte $P_i \in D^* := A \times B, i = 0, \ldots, s-1$, mit $A := [u_0 - l_u, u_{n_u+k_u-1} + l_u]$, $B := [v_0, v_{n_v+k_v-1}]$

Ausgabe: zusammenhängende Parameterwerte P_i, $i = 0, \ldots, s-1$

1. Setze $A_1 := [u_0 - l_u, u_0]$, $A_2 := [u_0, u_0 + l_u]$, $A_3 := [u_0 + l_u, u_{n_u+k_u-1} + l_u]$.

2. Ist $P_0 := (P_0^{(x)}, P_0^{(y)}) \notin A_2 \times B$, setze

$$P_0 := \begin{cases} \left(P_0^{(x)} + l_u, P_0^{(y)}\right), & P_0 \in A_1 \times B, \\ \left(P_0^{(x)} - l_u, P_0^{(y)}\right), & P_0 \in A_3 \times B. \end{cases}$$

   Damit ist $P_0 \in A_2 \times B$.

3. Schleife i von 1 bis $s-1$:

   - Ist $|P_i^{(x)} - P_{i-1}^{(x)}| \geq \frac{l_u}{2}$, suche $j \in \{-3, \ldots, 3\}$, so dass $|P_i^{(x)} + jl_u - P_{i-1}^{(x)}|$ minimal ist und $P_i^{(x)} + jl_u \in A$.

- Setze $P_i := \left(P_i^{(x)} + jl_u, P_i^{(y)}\right)$.

Abschließend kann nun der Algorithmus zur Trimmkurven-Berechnung angegeben werden. Hierfür sei dieses Mal $\mathfrak{C} := \{\mathfrak{C}_j \mid j = 0, \ldots, s-1\}$ eine zusammengesetzte Trimmkurve.

Algorithmus 2.12. (Trimmkurven-Berechnung für geschlossene Flächen)
Eingabe: geschlossene B-Spline-Fläche X, Trimmkurve $\mathfrak{C}$ im Bildbereich
Ausgabe: Trimmkurve $\mathfrak{B}$ im Parameterbereich D

1. Ersetze X durch die Erweiterung X^* gemäß (2.24) mit neuem Parameterbereich D^*.

2. Schleife j von 0 bis $s-1$:

   - Taste die einzelnen Kurvenstücke $\mathfrak{C}_j$ im Bildbereich an möglichst äquidistanten Punkten ab:

   $$V_j := \left\{v_i \in \mathfrak{C}_j \mid i = 0, \ldots, r_j - 1,\ \|v_l - v_{l-1}\| = \text{const},\ l = 1, \ldots, r_j - 1\right\}.$$

3. Setze $V := \bigcup_{j=0}^{s-1} V_j$.

4. Berechne Parameterwerte

   $$P := \left\{ p_m := \underset{p \in D^*}{\operatorname{argmin}} \|X(p) - V_m\| \mid m = 0, \ldots, \left(\sum_{j=0}^{s-1} r_j\right) - 1 \right\}$$

   mit Hilfe der Abstandsberechnung zur Fläche X.

5. Wende die Parameterkorrektur aus Algorithmus 2.11 auf P an.

6. Schleife j von 0 bis $s-1$:

   - Setze

   $$P_j := \left\{ p_m \mid m = \sum_{i=0}^{j-1} r_i, \ldots, \left(\sum_{i=0}^{j-1} r_i\right) + r_j - 1 \right\}.$$

   - Approximiere die Parameterwerte P_j als B-Spline-Kurve $\mathfrak{B}_j$.

Bemerkung 2.13. Nach Algorithmus 2.12 ist die Bounding Box von $\mathfrak{B}$ in die geschlossene Richtung immer kleiner gleich l_u aus Definition 2.8. Man sollte X daher auf $B(\mathfrak{B})$ beschränken, da man sonst ein Parametergebiet hat, von dem mindestens zwei Drittel nicht benutzt werden. Dies würde vor allem spätere Berechnungen erheblich verlangsamen. Bei Rotationsflächen trimmt man hierbei nur auf das nächste Vielfache des Trennwinkels ξ (siehe Abschnitt 1.3.1).

2.4 Entartete Flächen

Entartete B-Spline-Flächen entstehen, wenn eine Seite der Fläche auf einen Punkt reduziert wird. Auf diese Weise kann man auch Dreiecksflächen ohne Trimmkurven in Tensor-Produkt-Gestalt darstellen.

Definition 2.14. *Es seien eine B-Spline-Fläche X und ihre Randkurve $\mathfrak{R} := \{R_0, \ldots, R_3\}$ gegeben. Ist $R_i(t) = c \in \mathbb{R}^3$ für (mindestens) ein $i \in \{0,1,2,3\}$ und alle $t \in [u_0, u_{n_u+k_u-1}]$ (für $i = 0,2$) beziehungsweise $t \in [v_0, v_{n_v+k_v-1}]$ (für $i = 1,3$), so heißt die Fläche* entartet an der Seite *i und c heißt* Entartungspunkt. *Das bedeutet:*

$$\begin{aligned}
i = 0 &\Rightarrow X(s, v_0) = X(t, v_0) = c, \quad s,t \in [u_0, u_{n_u+k_u-1}],\\
i = 1 &\Rightarrow X(u_0, s) = X(u_0, t) = c, \quad s,t \in [v_0, v_{n_v+k_v-1}],\\
i = 2 &\Rightarrow X(s, v_{n_v+k_v-1}) = X(t, v_{n_v+k_v-1}) = c, \quad s,t \in [u_0, u_{n_u+k_u-1}],\\
i = 3 &\Rightarrow X(u_{n_u+k_u-1}, s) = X(u_{n_u+k_u-1}, t) = c, \quad s,t \in [v_0, v_{n_v+k_v-1}].
\end{aligned}$$

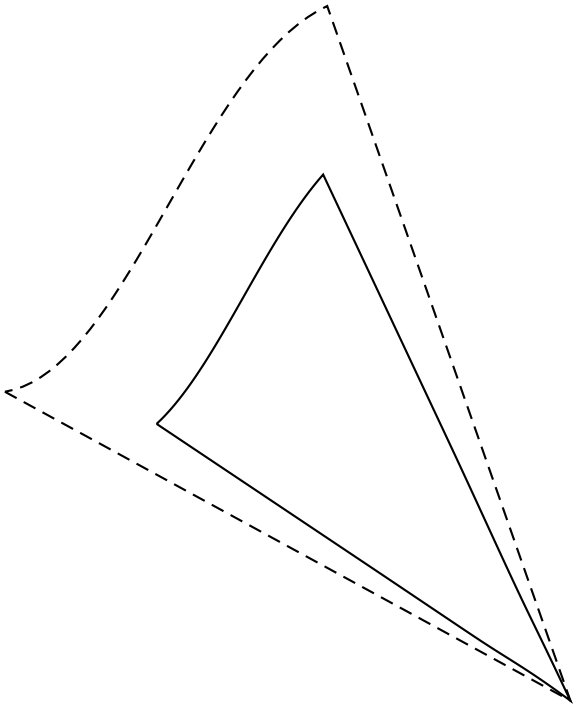
Konvention 2.15.

1. Es können auch Entartungen an mehreren Seiten entstehen. Es wird hier aber angenommen, dass es maximal zwei Entartungen an den gegenüberliegenden Seiten $i, j \in \{0,1,2,3\}$ gibt. Das heißt: $i = 0, j = 2$ oder $i = 1, j = 3$.
2. Ohne Einschränkung sei im Folgenden eine entartete Fläche immer in Richtung u entartet.

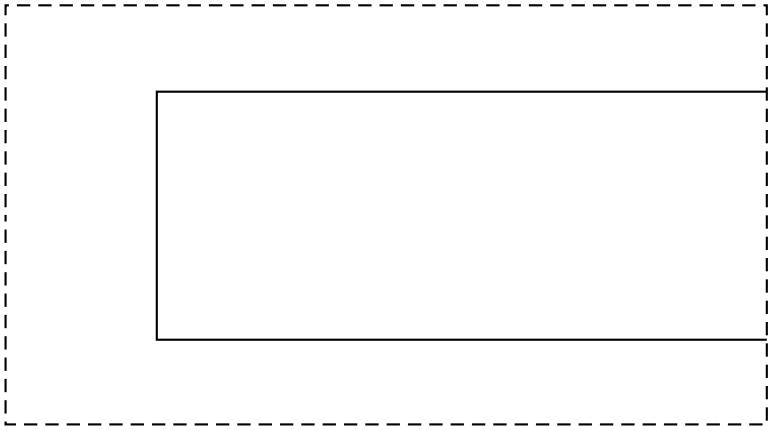
Entartete Flächen haben mehrere Nachteile:

1. Ist in der IGES-Datei die Trimmkurve im Parameterbereich gegeben, hat man pro Seite im Bildbereich genau eine Kurve. Das bedeutet für die „fehlende“ Seite, die entartet ist, fehlt die Kurve im Parameterbereich (Abbildung 2.4).
2. Es kann keine Normale in einer Entartung berechnet werden, da die abzuleitende B-Spline-Kurve auf einen Punkt zusammenfällt und die Ableitung dort verschwindet (vgl. Teil II, Abschnitt 2.2).
3. Parameterwerte für Punkte in der Nähe einer Entartung, die mittels der Abstandsberechnung bestimmt werden sollen, fangen dort an zu „wackeln“. Dies hat den Grund, dass an der entarteten Seite alle Parameterwerte (u, v), $v = v_0$ oder $v = v_{n_v+k_v-1}$, auf den gleichen Punkt im Bildbereich abgebildet werden (siehe Definition 2.14).

Das erste Problem kann man lösen, indem man die Start- und Endpunkte der einzelnen Trimmkurven im Parameterbereich betrachtet und offene Seiten durch einen linearen Spline verbindet. Das zweite Problem lässt sich lösen, indem man die Normalen in der Umgebung der Entartung betrachtet und eine Grenzwertberechnung durchführt. Das dritte Problem ist etwas schwieriger zu handhaben.



a. Bildbereich



b. Parameterbereich (Trimmkurve rechts nicht definiert)

Abbildung 2.4: Fläche (gestrichelt) und Trimmkurve (durchgehend)

2.4.1 Ein Maßstab für die Entartung

Die Idee ist es, einen Maßstab für die Entartung zu liefern, das heißt die Fläche in Bereiche einzuteilen und Punkte in der Nähe einer Entartung zu ignorieren. Zu diesen ignorierten Punkten sucht man dann aus den zulässigen gegebenen eine stetige Fortsetzung.

Definition 2.16. *Es sei eine entartete B-Spline-Fläche X gegeben. Dann heißt*

$$\psi := (\psi_i)_{i=0}^{n_u-1}, \quad \psi_i := \frac{\sum_{j=1}^{n_v-1} \|d_{i,j} - d_{i,j-1}\|}{n_v - 1}, \tag{2.25}$$

Entartungsgrad der Kontrollpunkte. *Der Einfachheit halber setzt man*

$$\psi := \frac{\psi}{\|\psi\|_\infty},$$

so dass $\psi \in [0,1]^{n_u}$. Der Vektor

$$\lambda := (\lambda_i)_{i=0}^{n_u-1}, \quad \lambda_i := \begin{cases} 0, & i = 0, \\ \lambda_{i-1} + \frac{\sum_{j=0}^{n_v-1} \|d_{i,j} - d_{i-1,j}\|}{n_v}, & i = 1, \dots, n_u - 1, \end{cases} \tag{2.26}$$

heißt Entartungsparameter. *Man setzt*

$$\lambda_i := u_0 + \frac{\lambda_i}{\lambda_{n_u-1}}(u_{k_u+n_u-1} - u_0), \quad i = 0, \dots, s-1,$$

so dass $\lambda \in [u_0, u_{k_u+n_u-1}]^{n_u}$.

Der Entartungsgrad ist an der Entartung immer Null, an der „breitesten“ Stelle immer Eins. Über den Entartungsparameter λ kann man zu jedem Parameterwert (u,v) durch lineare Interpolation bestimmen, ob er in der Nähe einer Entartung liegt (siehe nächster Abschnitt).

Beispiel 2.17. Es sei die entartete Fläche in Abbildung 2.5a gegeben. Man berechnet dazu den Entartungsgrad ψ und den Entartungsparameter λ der einzelnen Kontrollpunktreihen. Dies ergibt eine Aufteilung wie in Abbildung 2.5b. Die genauen Werte können Tabelle 2.1 entnommen werden.

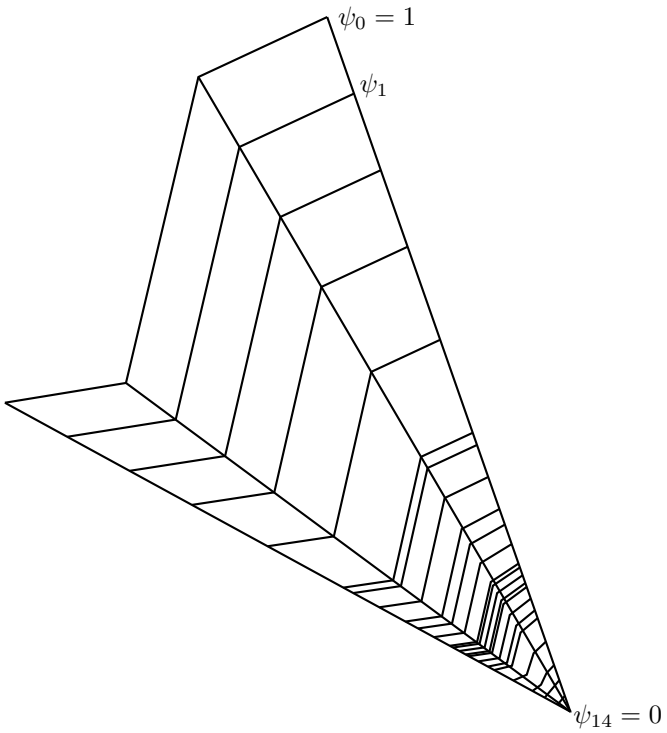
2.4.2 Parameterkorrektur

Nach der Berechnung des Entartungsgrades kann für jeden Parameterwert (u,v) der Entartungswert bestimmt werden und alle Werte, die eine gewisse Entartungsgrenze unterschreiten, werden durch ein approximierendes Polynom angenähert. Auf diese Weise verhindert man eine Oszillation der Parameterwerte in der Nähe einer Entartung.

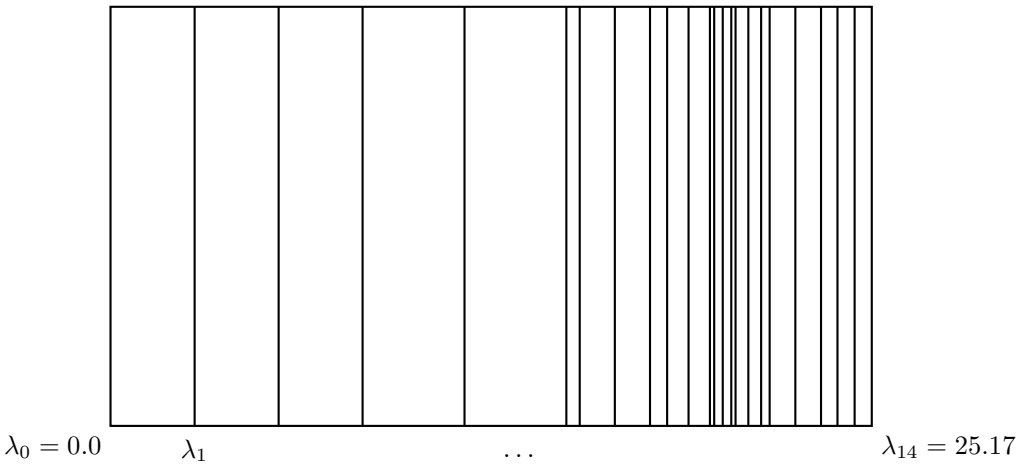
Definition 2.18. *Es seien eine entartete B-Spline-Fläche X, Entartungsgrad ψ (2.25), Entartungsparameter λ (2.26) und ein Parameterwert $(u,v) \in D$ gegeben. Dann bezeichnet*

$$\widehat{\psi}(u,v) := \psi_r + \frac{u - \lambda_r}{\lambda_{r+1} - \lambda_r}(\psi_{r+1} - \psi_r), \quad \lambda_r \le u \le \lambda_{r+1}, \tag{2.27}$$

den Entartungswert von (u,v).



a. Knotenreihen im Bildbereich



b. Einteilung im Parameterbereich

Abbildung 2.5: Entarteter Beispielpatch

Reihe i	ψ_i	λ_i
0	1.0	0.0
1	0.890	2.782
2	0.780	5.558
3	0.669	8.334
4	0.535	11.703
5	0.402	15.071
6	0.337	16.673
7	0.265	18.402
8	0.190	20.241
9	0.137	21.510
10	0.087	22.641
11	0.054	23.490
12	0.035	24.034
13	0.017	24.601
14	0.0	25.168

Tabelle 2.1: Entarteter Beispielpatch, Maß ψ und Aufteilung λ

Damit kann man einen Algorithmus zur Parameterkorrektur angeben.

Algorithmus 2.19.

Eingabe: entartete B-Spline-Fläche X, Punkte $\{p_l \in \mathbb{R}^3 \,|\, l = 0,\ldots,s-1\}$ und Parameterwerte $Q := \{q_l \in D \,|\, l = 0,\ldots,s-1\}$, Entartungsgrenze ϵ_ψ

Ausgabe: korrigierte Parameterwerte Q

1. Bestimme chordale Parametrierung (siehe Teil V, Abschnitt 1.1)

$$V := \left\{ v_l \,|\, v_0 := 0, v_l := v_{l-1} + \frac{\|p_l - p_{l-1}\|}{d},\ l = 1,\ldots,s-1,\ d := \sum_{i=1}^{s-1} \|p_i - p_{i-1}\| \right\}. \tag{2.28}$$

2. Setze geordnete Mengen

$$P := \left\{ q_l \in Q \,|\, \widehat{\psi}(q_l) > \epsilon_\psi \right\}, \quad U := \left\{ v_l \in V \,|\, \widehat{\psi}(q_l) > \epsilon_\psi \right\}.$$

3. Approximiere Punkte P mit Parametern U durch ein Polynom H auf dem Intervall $[v_0, v_{s-1}]$ (zum Beispiel durch Least-Squares-Approximation [26, S. 245 ff]).

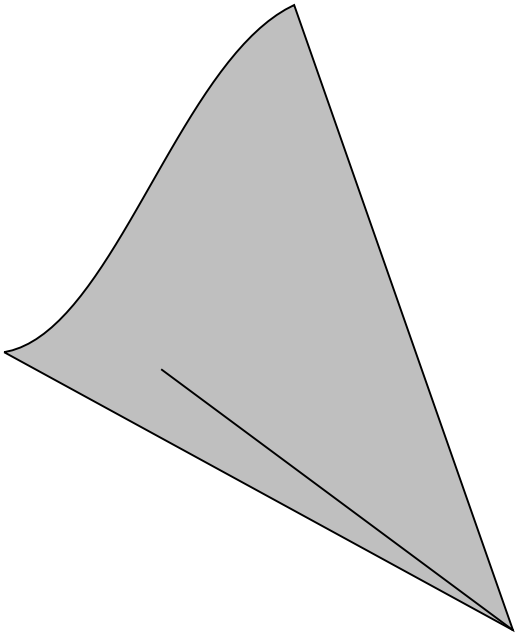
4. Setze

$$q_l := \left(1 - \widehat{\psi}(p_l)\right) H(v_l) + \widehat{\psi}(p_l) p_l, \quad l = 0,\ldots,s-1. \tag{2.29}$$

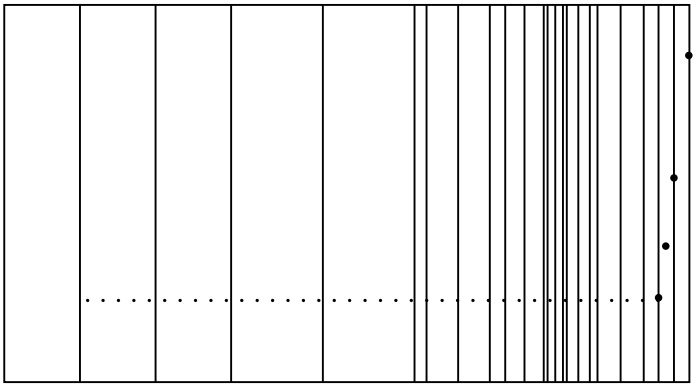
Gleichung (2.29) sorgt für eine Konvexkombination aus dem Entartungsgrad $\widehat{\psi}(q_l)$, so dass in der Nähe einer Entartung der über das approximierte Polynom H berechnete Wert und weit entfernt von einer Entartung der ursprüngliche Parameterwert q_l stärker gewichtet wird.

Nach der Parameterkorrektur fährt man mit der Approximation der Parameterwerte wie im allgemeinen Fall fort (siehe Abschnitt 2.1).

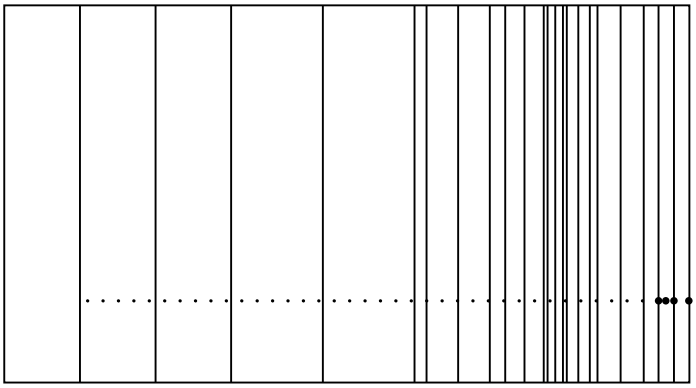
Beispiel 2.20. Es seien die Fläche und Aufteilung aus Beispiel 2.17 gegeben. Es soll nun zu einer Trimmkurve auf der Fläche die zugehörige Kurve im Parameterbereich berechnet werden (Abbildung 2.6a). Nach der ersten Berechnung erhält man die Parameterwerte wie in Abbildung 2.6b. Am rechten Rand in der Nähe der Entartung ist der Parameterwert in u Schwankungen unterworfen. Die letzten drei Punkte werden daher ignoriert, da $\widehat{\psi}(q_l) \leq \epsilon_\psi$ und nur mit den restlichen eine stetige Fortsetzung approximiert (Abbildung 2.6c). Diese Parameterwerte können dann als Trimmkurve im Parameterbereich approximiert werden (siehe Teil V).



a. Fläche mit Kurve



b. Parameterwerte nach Annäherung



c. Korrigierte Parameterwerte

Abbildung 2.6: Entarteter Beispielpatch

Teil IV
Zuordnung der Punkte

1 Einleitung

Wenn man die CAD-Daten wie im letzten Kapitel beschrieben als B-Spline-Kurven und Tensor-Produkt-Spline-Flächen extrahiert hat, kann man die Punkte der Frässimulation zuordnen.

Obwohl die Newton-Methode zur Parameterbestimmung und Punktzuordnung eine quadratische Konvergenz hat, ist es dennoch ein teures Verfahren, da viele Parameterwerte der Fläche ausgewertet werden müssen. Aus diesem Grund ist ein mehrstufiges Zuordnungsverfahren sinnvoll, welches in jeder Stufe mehr Punkte aussortiert.

Der erste Gedanke ist eine einfache Parallelprojektion aller Punkte und Trimmkurven in die x/y-Ebene. Dies funktioniert aber nur bei Spline-Flächen, die fast parallel zur x/y-Ebene liegen, gut. Daher werden die Punkte auf die Normalenebene einer B-Spline-Fläche parallelprojiziert, was im Aufwand kaum höher ist, da es sich nur um eine Koordinatentransformation (Rotation) handelt.

Problematisch sind Flächen, deren Normalenebene nicht die Form der Fläche widerspiegelt. Dies ist bei gedrehten Flächen oder bei Rotationsflächen der Fall. Die Idee ist, zu jeder B-Spline-Fläche einen Grad der Verdrehung zu bestimmen und sie in mehrere Teilflächen aufzusplitten. Auf die Normalenebenen dieser kleineren B-Spline-Flächen wird dann korrekt projiziert.

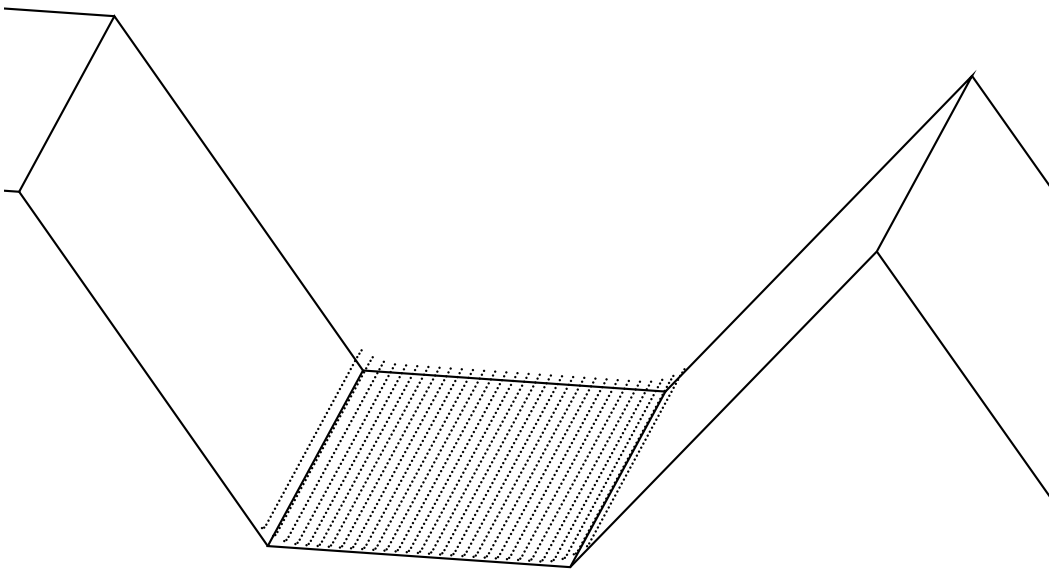


Abbildung 1.1: „Tal“-Patch

Projiziert man parallel auf die Normalenebene einer Fläche, passiert es, dass man einige Punkte am Rand doppelt zählt. Hierzu betrachtet man die B-Spline-Fläche in Abbildung 1.1

und die potentielle Punktezuordnung. Eine Projektion in die Normalenebene, die in diesem Fall mit der Fläche selbst übereinstimmt, geht recht leicht. Um die falsche Zuordnung zu sehen, betrachtet man die B-Spline-Fläche mit den beiden angrenzenden Patch-„Hügeln“ von der Seite (Abbildung 1.2). Die senkrechten durchgezogenen Linien grenzen die Punktmenge ein, die bei einer Parallelprojektion in die Ebene der Fläche zugeordnet werden würden. Die gestrichelte Begrenzung ist die richtige Zuordnung über den Abstand eines Punktes zur Fläche. Zusätzlich kann man die gestrichelte Linie als Abgrenzung zwischen den benachbarten Flächen ansehen, da eine Parallelprojektion der Punkte auf die Nachbarflächen diese erneut zuordnen würde. Die Idee ist es, mittels einer Abstandsberechnung zu überprüfen, zu welcher Fläche ein Punkt den kleinsten Abstand hat. Dieser wird der Punkt dann zugeordnet.

Abbildung 1.2: Punkteinschließung

Die Punktezuordnung verläuft somit in mehreren Stufen:

1. Zuordnungsstufe – Punkte werden nur über die Bounding Box der Fläche zugeordnet.

2. Zuordnungsstufe – Gedrehte Flächen werden aufgeteilt und die Punkte werden zusammen mit den Trimmkurven in die Normalenebenen parallelprojiziert.

3. Zuordnungsstufe – Zu jedem Punkt wird der Parameterwert durch Orthogonalprojektion (Newton-Verfahren) berechnet.

Die Verfahren sind dabei nach Aufwand aufsteigend sortiert, so dass man bei der dritten Zuordnungsstufe nur noch sehr wenige Punkte hat, die man überprüfen muss.

Konvention 1.1.

1. Die Punktewolke der Frässimulation sei ab sofort durch

$$\mathfrak{P} := \{P_j \,|\, j = 0, \ldots, s-1\} \tag{1.1}$$

   definiert.

2. Ist eine B-Spline-Fläche X gegeben, dann bezeichne

$$\mathfrak{D} := \{d_{i,j} \,|\, i = 0, \ldots, n_u - 1,\ j = 0, \ldots, n_v - 1\} \tag{1.2}$$

   die Menge der Kontrollpunkte von X.

2 Zuordnung durch Bounding Box

Da ein Kugelfräser beispielsweise keine (inneren) Ecken nachbilden kann, kommt es vor, dass die Punkte, die zu einer Fläche gehören, an dieser Stelle ziemlich weit entfernt liegen. Je nach Winkel, der in dieser Ecke gebildet wird, kann dieser Wert variieren, aber niemals einen Fräskopfradius überschreiten (Abbildung 2.1). Aus diesem Grund erweitert man die Bounding Box in alle Richtungen um diesen Wert, um alle Punkte, die in Frage kommen, zu betrachten.

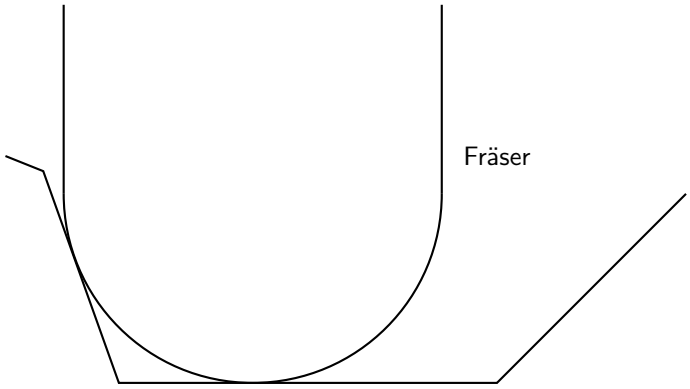


Abbildung 2.1: Ein Kugelfräser kommt nicht in Ecken

Definition 2.1. *Es sei ein Gebiet $G \subset \mathbb{R}^2$, die Bounding Box $B(G)$ gemäß (II.2.10) und ein Radius $r > 0$ gegeben. Dann bezeichnet*

$$B_r(G) := \left[B_{\min}(G) - (r,r)^T, B_{\max}(G) + (r,r)^T\right] \tag{2.3}$$

die um r erweiterte Bounding Box. *Analog ist dies für $G \subset \mathbb{R}^3$ definiert.*

Alle Punkte der Punktewolke, die innerhalb von $B_r(\mathfrak{D})$ mit r als Fräskopfradius liegen, können zu der B-Spline-Fläche gehören. Es ist nicht mehr möglich, dass es noch Punkte gibt, die zu der Fläche gehören, die man hiermit nicht eingegrenzt hat.

3 Zuordnung durch Normalenebenen

In diesem Abschnitt werden die Punkte feiner zugeordnet, indem man sie mitsamt der Trimmkurven auf die Normalenebene der Fläche projiziert und überprüft, welche Punkte innerhalb oder außerhalb des eingeschlossenen Gebietes liegen. Aus Gründen der Komplexität werden die in die Normalenebene projizierten Kurven durch 2-D-Linearsätze dargestellt.

3.1 Projektion auf die Normalenebene

Wie die Normalenebene einer B-Spline-Fläche berechnet wird, kann man in Teil II, Abschnitt 2.3 nachlesen. Für eine einfache Projektion der Punkte auf diese Ebene dreht man das gesamte System so, dass die Normalenebene parallel zur x/y-Ebene liegt. Das bedeutet, der

Normalenvektor $n = (x, y, z)^T$ der Normalenebene muss in Richtung $(0, 0, 1)^T$ gedreht werden. Die Rotationsmatrix

$$R := \begin{pmatrix} y^2 + \frac{z - y^2 z}{l} & -xy + \frac{xyz}{l} & 1 - \frac{z+dx}{l} \\ -xy + \frac{xyz}{l} & x^2 + \frac{z - x^2 z}{l} & 1 - \frac{z+dy}{l} \\ 1 - \frac{z-dx}{l} & 1 - \frac{z-dy}{l} & 1 \end{pmatrix}, \quad l := \sqrt{x^2 + y^2 + z^2}, \quad d := \sqrt{x^2 + y^2}, \tag{3.4}$$

erfüllt genau diesen Zweck. Die genaue Berechnung von R kann in Anhang B nachgelesen werden.

3.2 2-D-Clipping

Unter *Clipping* versteht man in der Informatik das Zurechtschneiden eines Gebietes auf einen bestimmten (sichtbaren) Bereich. In dieser Arbeit handelt es sich im Speziellen um zweidimensionale *polygonale Regionen*, bei denen entschieden werden soll, ob ein Punkt P innerhalb der Region liegt oder nicht.

Definition 3.1. *Es sei ein planares, geschlossenes Polygon*

$$G := (g_i \in \mathbb{R}^2 \mid g_{m-1} = g_0,\ i = 0, \ldots, m-1) \tag{3.5}$$

ohne Selbstschnitte oder Berührpunkte außer an den Ecken gegeben. Mit Definition 2.15 aus Teil II, angewandt auf das Polygon P, sei die polygonale Region von P *als*

$$L := \mathfrak{M}(G) \tag{3.6}$$

definiert.

Polygonale Regionen sind also die von einem geschlossenen Polygon eingeschlossenen Gebiete. Die Richtung der Polygonsegmente gibt an, ob das Gebiet „innerhalb“ des Polygons betrachtet wird oder „außerhalb“.

Konvention 3.2.

1. Wenn von einem Polygonen G gesprochen wird, ist immer ein geschlossenes, planares Polygon nach Definition 3.1 gemeint.

2. Eine Laufrichtung des Polygons G gegen den Uhrzeigersinn bedeutet eine einschließende Region L, mit dem Uhrzeigersinn eine ausschließende. Der Einfachheit halber wird L als einschließende Region angenommen.

3.2.1 Geradenargument

Der erste Algorithmus für die In/Out-Unterscheidung ist ziemlich leicht und wurde unter anderem von Finley [12], Haines [20] und Snoeyink [33, S. 767] beschrieben.

Definition 3.3. *Es seien ein Polygon G, die zugehörige polygonale Region L, ein Punkt $P \in \mathbb{R}^2$ und eine planare Gerade h mit $h(0) = P$ gegeben. Es bezeichne S die Menge der Schnittpunkte von h und G:*

$$S(h,G) = \{s_0, \ldots, s_{l-1}\} := \{s \in h \,|\, s \in G\}. \tag{3.7}$$

Zusätzlich sei

$$F(h,G) := \{f_i \,|\, h(f_i) = s_i,\ i = 0, \ldots, l-1\} \tag{3.8}$$

die Menge der Schnittpunktparameter von h und G. Dabei soll gelten:

$$f_i < f_j, \quad i < j, \quad i,j = 0, \ldots, l-1.$$

Ist nun

$$F_- := \{f_i \in F(h,G) \,|\, f_i < 0\}, \quad F_+ := \{f_i \in F(h,G) \,|\, f_i > 0\}, \tag{3.9}$$

so gilt:

$$P \in L \Leftrightarrow \#F_- \text{ ungerade }, \#F_+ \text{ ungerade },$$

$$P \notin L \Leftrightarrow \#F_- \text{ gerade }, \#F_+ \text{ gerade }.$$

Ist $0 \in F(h,G)$, so ist $h(0) = P \in G$ und man muss definieren, ob dieser Punkt zu einer Region L gehört oder nicht. Die Idee dahinter ist, dass, wenn man von außen in ein geschlossenes Gebiet eindringt, man dieses auch wieder verlassen muss. Das heißt, man hat immer die gleiche Anzahl von Eintritts- und Austrittspunkten.

Definition 3.4. *Es seien ein Polygon G, die zugehörige polygonale Region L, ein Punkt $P \in \mathbb{R}^2$, eine planare Gerade h, sowie $S(h,G)$ gemäß (3.7) gegeben. Existiert für ein $s \in S(h,G)$ ein $j \in \{0, \ldots, m-1\}$, so dass $s = g_j$, heißt dies* Eckpunktberührung in s.

Problematisch sind solche Eckpunktberührungen. Hierfür wird zusätzlich eine Überhalb/ Unterhalb-Unterscheidung vorgenommen. Dabei wird ein Schnittpunkt $s = g_j = h(f_i)$, $f_i \in F(h,G)$, nur dann gezählt, wenn sich ein Eckpunkt des Polygonsegments (g_j, g_{j+1}) „unterhalb“ der Geraden h befindet und der andere überhalb oder auf der Geraden liegt (siehe Abbildung 3.1, aus Finley [12]). Mit anderen Worten, s wird gezählt, falls

$$(g_i \in H_- \ \wedge\ g_{i+1} \in H_+ \cup h) \vee (g_{i+1} \in H_- \ \wedge\ g_i \in H_+ \cup h),$$

wobei H_- und H_+ die beiden disjunkten Halbebenen bezeichnen, in die h die Ebene aufteilt. In allen anderen Fällen wird kein Schnittpunkt gezählt.

Beispiel 3.5. In Abbildung 3.1 wird eine zur x-Achse parallele Gerade durch den Punkt D gezogen. Seite b erzeugt dabei klar einen Schnittpunkt, die Seiten a und f definitiv nicht. Seite e erzeugt einen Schnittpunkt, da ein Eckpunkt unterhalb der Geraden liegt und der andere Eckpunkt auf der Geraden. Seite d erzeugt keinen Schnittpunkt, da beide Eckpunkte auf der Geraden liegen. Und Seite c erzeugt ebenfalls keinen Schnittpunkt, da sich kein Eckpunkt unterhalb der Geraden befindet. In diesem Fall hat man je einen Schnittpunkt rechts und links und der Punkt D liegt im Inneren der polygonalen Region.

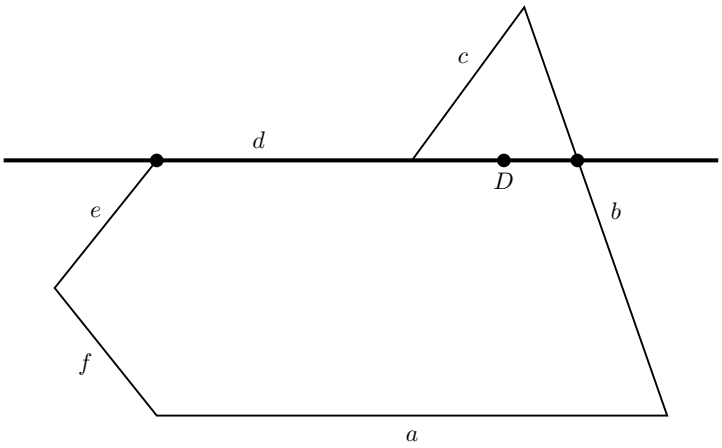


Abbildung 3.1: Einfaches Clipping

Bemerkung 3.6. In der Regel wählt man die Gerade h nicht zufällig, sondern man benutzt eine Gerade parallel zur x- oder y-Achse, da sich die Schnittpunkte so leichter berechnen lassen.

3.2.2 Binary Space Partition

Binary Space Partition bezeichnet die Unterteilung eines Raumes in Hyperebenen. Im zweidimensionalen Fall wird eine Ebene durch Geraden unterteilt. Mit der ersten Geraden teilt man die Ebene zum Beispiel in rechts und links. Danach kann man die linke und die rechte Hälfte in oben und unten aufteilen und so weiter. Als Darstellung hierfür benutzt man Binärbäume (Bäume mit maximal zwei Ästen), die *Binary Space Partition Trees* (kurz: BSP-Trees) genannt werden. Eingeführt wurden sie von H. Fuchs, Z. Kedem und B. Naylor [16]. Dieser Abschnitt soll keine Einführung in BSP-Trees geben. Nähere Informationen zu dem Thema können bei de Berg, de Kreveld und Overmars [5, S. 251 ff] gefunden werden.

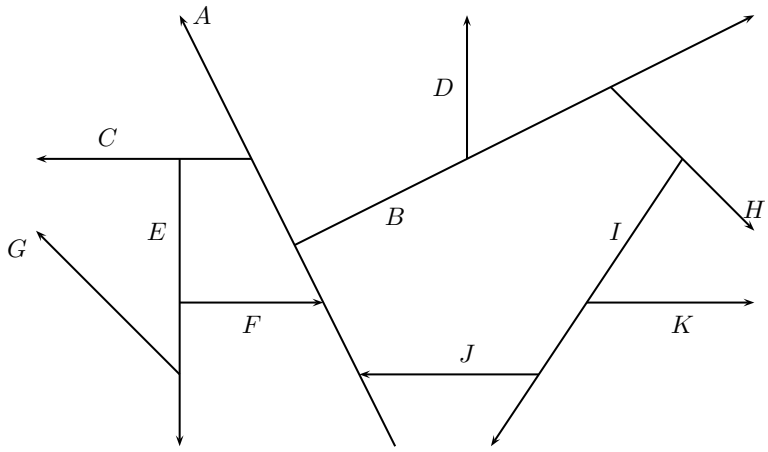


Abbildung 3.2: Beispiel-Aufteilung der Ebene

Beispiel 3.7. Es sei die Ebene wie in Abbildung 3.2 aufgeteilt. Dabei hat man zuerst an der Geraden A unterteilt, so dass man zwei Halbebenen erhält, die dann wieder an den Halbgeraden B und C aufgeteilt wurden. Daraus entsteht der BSP-Tree, den man in Abbildung 3.3 sieht.

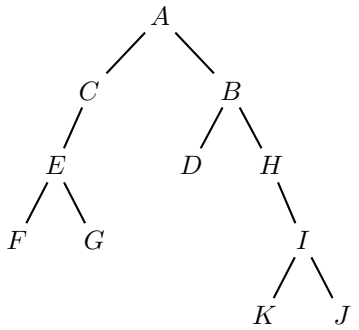


Abbildung 3.3: Der zu obiger Aufteilung gehörige BSP-Tree

Bei polygonalen Regionen L wählt man die Unterteilungsgeraden nicht willkürlich, sondern benutzt die einzelnen Polygonsegmente $(g_i, g_{i+1}), i = 0, \ldots, m-2$. Alle übrigen Segmente werden dann entsprechend links und rechts aufgeteilt und an diesen die Halbebenen weiter geteilt.

Sei ein Polygon G wie in (3.5) gegeben. Die Ebene soll an der Geraden h, die durch das Polygonsegment (g_i, g_{i+1}) beschrieben ist, in disjunkte Halbebenen H_- und H_+ geteilt werden. Bei der Aufteilung der restlichen Polygonsegmente (g_j, g_{j+1}), $j = 0, \ldots, m-2$, $j \neq i$, gibt es vier Fälle:

1. Das Polygonsegment (g_j, g_{j+1}) liegt in der linken Halbebene H_-:

   $$g_j \in H_- \cup h \wedge g_{j+1} \in H_- \cup h.$$

   Dann fügt man es in den linken Baum ein.

2. Das Polygonsegment (g_j, g_{j+1}) liegt in der rechten Halbebene H_+:

   $$g_j \in H_+ \cup h \wedge g_{j+1} \in H_+ \cup h.$$

   Dann fügt man es in den rechten Baum ein.

3. Das Polygonsegment (g_j, g_{j+1}) liegt auf der Geraden h:

   $$g_j \in h \wedge g_{j+1} \in h.$$

   Dann kann man es verwerfen, da bereits eine Aufteilung an dieser Geraden durch das Polygonsegment (g_i, g_{i+1}) vorgenommen wurde.

4. Das Polygonsegment (g_j, g_{j+1}) schneidet die Gerade h:

   $$(g_j \in H_- \wedge g_{j+1} \in H_+) \vee (g_{j+1} \in H_- \wedge g_j \in H_+).$$

Dann berechnet man den Schnittpunkt

$$s := (g_i, g_{i+1}) \cap (g_j, g_{j+1}),$$

erstellt Polygonsegmente $(g_j, s), (s, g_{j+1})$ und fügt diese in den jeweiligen linken beziehungsweise rechten Baum ein.

Definition 3.8. *Es sei ein Polygon G gegeben. Dann bezeichne L_i den linken Teilbaum und R_i den rechten Teilbaum, wenn man die Ebene am Polygonsegment (g_i, g_{i+1}) aufteilt. Zusätzlich sei*

$$S_i := \{(g_j, g_{j+1}) \in G \backslash \{(g_i, g_{i+1})\} \mid (g_i, g_{i+1}) \cap (g_j, g_{j+1}) \neq \emptyset,\ j = 0, \ldots, m-2\},$$

$i = 0, \ldots, m-1$. *Dann definiert man eine* Bewertungsfunktion

$$\rho_i := |\#R_i - \#L_i| + \#S_i, \quad i = 0, \ldots, m-1. \tag{3.10}$$

Die Art der Ebenenunterteilung ist essentiell für die Größe des Baumes und für die Geschwindigkeit einer späteren Überprüfung. Es gibt hierfür verschiedene Ansätze. Anhand von drei Beispielen sollen die verschiedenen Ergebnisse aufgezeigt werden. Die einzelnen Polygonsegmente sind dabei nach ihrem Starteckpunkt benannt. Das Segment, das aus den Punkte g_i und g_{i+1} gebildet wird, erhält somit die Nummerierung i, $i = 0, \ldots, 11$. In manchen Fällen müssen einzelne Polygonsegmente aufgrund von Schnitten mit anderen Geraden (siehe Punkt 4 bei der Fallunterscheidung oben) in weitere Subsegmente a, b und/oder c aufgesplittet werden. Der Pfeil an jedem Segment gibt in den Abbildungen immer die Richtung an, in der innen liegt (rechter Ast des Baumes).

1. Abbildung 3.4 – Die Aufteilung des Baumes geschieht in der Mitte. Das heißt, wenn man m Polygonsegmente hat, nimmt man das $\lceil m/2 \rceil$-te Polygonsegment als Gerade zur Unterteilung.

2. Abbildung 3.5 – Bei dieser Methode teilt man die Ebene immer am ersten Polygonsegment auf. Es sollte nicht verwundern, dass das Polygonsegment 4 nicht im Baum auftaucht. Da dieses zusammen mit Segment 0 auf einer Geraden liegt, muss es nicht weiter betrachtet werden (siehe Punkt 3 in der Fallunterscheidung oben).

3. Abbildung 3.6 – Diese Methode ist die aufwendigste, liefert aber normalerweise die besten Ergebnisse in Hinblick auf die Geschwindigkeit beim Baumdurchlauf. Man versucht hierbei die Größe/Höhe des Baumes möglichst gering zu halten. Das bedeutet man teilt die Polygonsegmente immer an Segment Nummer

$$\operatorname*{argmin}_{i=0,\ldots,m-1} \rho_i$$

auf, wobei ρ_i wie in (3.10) definiert ist.

Eine Implementierung hat gezeigt, dass es zwischen allen drei Methoden nur geringe Unterschiede in der Laufzeit gibt. Dies hat vor allem den Grund, dass sehr viele solcher Binärbäume erstellt werden müssen und unter Umständen nur einmalig für eine Punktmenge benutzt werden. Bei Methode 3 fällt der Vorteil des schnelleren Baumdurchlaufs dadurch nicht mehr ins Gewicht.

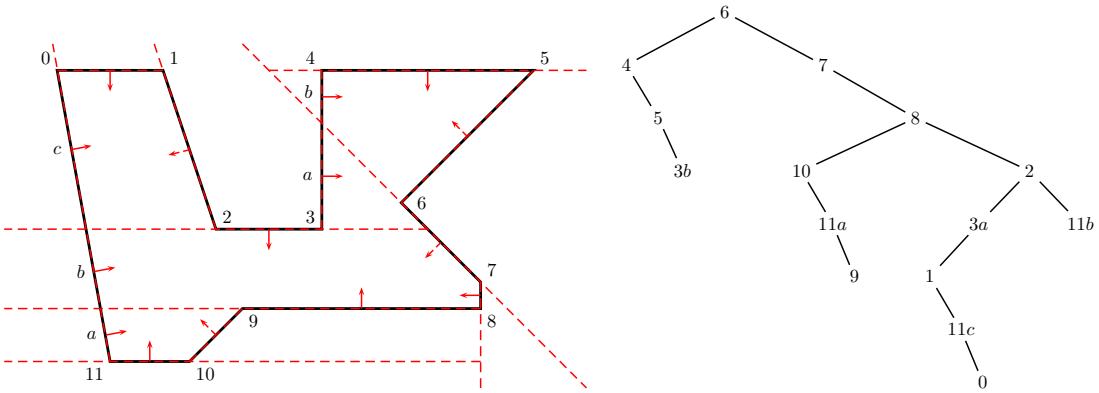


Abbildung 3.4: BSP-Aufteilung in der Mitte

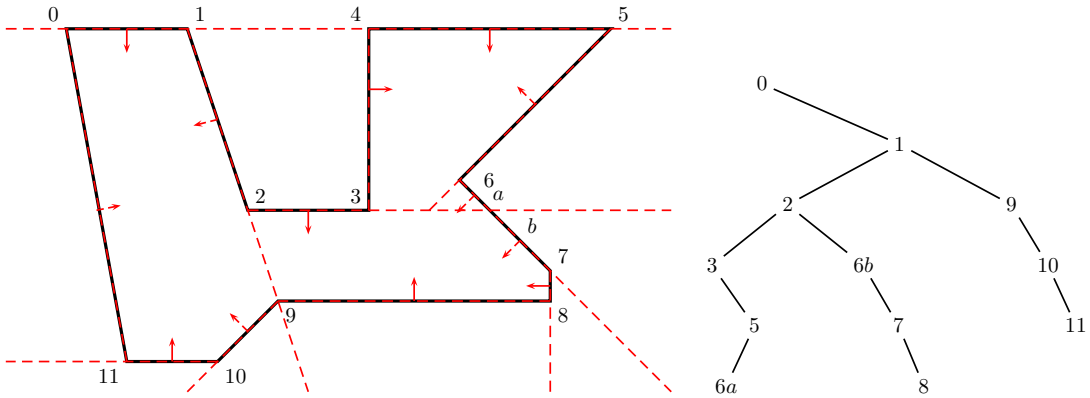


Abbildung 3.5: BSP-Aufteilung am Anfang

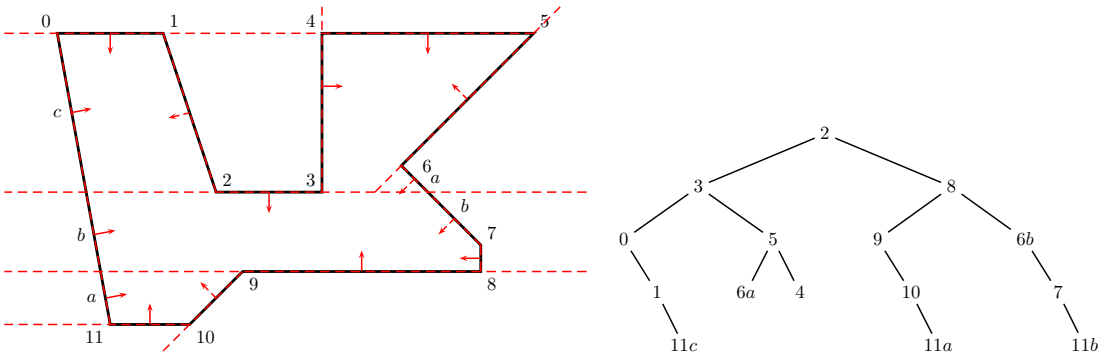


Abbildung 3.6: BSP-Aufteilung nach Balance

Beispiel 3.9. Abbildung 3.7 zeigt, wie anhand der letzten Aufteilung nach Balance für zwei Punkte entschieden wird, ob sie innerhalb oder außerhalb der polygonalen Region liegen.

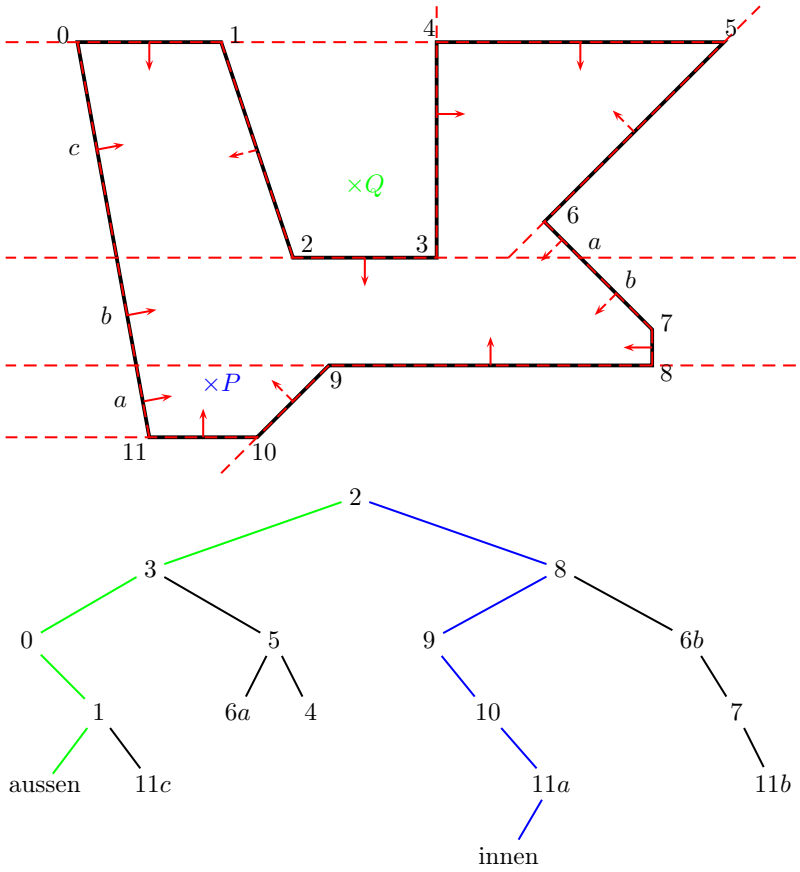


Abbildung 3.7: Beispiel für einen Baumdurchlauf

3.3 Gedrehte B-Spline-Flächen

Die Projektion und Zuordnung der Punkte auf die Normalenebene schlägt bei B-Spline-Flächen fehl, die zu stark gedreht sind, da die Normalenebene die Struktur der Fläche nicht gut wiedergibt. Um dies zu umgehen, erstellt man aus jeder Fläche einen Patch-Quadtree.

Definition 3.10. *Ein* Patch-Quadtree *ist eine (ungetrimmte) B-Spline-Fläche X mit Definitionsbereich $D := [u_0, u_{n_u+k_u-1}] \times [v_0, v_{n_v+k_v-1}]$, die zusätzlich vier Verweise auf die Flächen*

$$X_1 := X|_{[u_0,u^*]\times[v_0,\ldots,v^*]}, \quad X_2 := X|_{[u^*,u_{n_u+k_u-1}]\times[v_0,\ldots,v^*]},$$

$$X_3 := X|_{[u_0,u^*]\times[v^*,v_{n_v+k_v-1}]}, \quad X_4 := X|_{[u^*,u_{n_u+k_u-1}]\times[v^*,v_{n_v+k_v-1}]},$$

$u^* := \frac{u_0+u_{n_u+k_u-1}}{2}$, $v^* := \frac{v_0+v_{n_v+k_v-1}}{2}$, haben kann. Dabei sind $X_1, \ldots, X_4$ wiederum Patch-Quadtrees.

Ob ein Patch-Quadtree Verweise auf die vier Teilflächen besitzt, ist abhängig vom Verdrehungsgrad der B-Spline-Fläche.

Definition 3.11. *Es seien eine B-Spline-Fläche X, Parameterwerte $P := \{p_i \in D \mid l = 0, \ldots, l-1\}$, die Normalen $\mathfrak{N}(p_i), i = 0, \ldots, l-1$ und die daraus entstandene Normalenebene $\mathfrak{A}(X,P)$ gegeben. Sei $\mathfrak{n}$ der Normalenvektor von $\mathfrak{A}(X,P)$. Dann heißt*

$$\nu(X,P) := \max_{i=0}^{l-1} \angle(\mathfrak{n}, \mathfrak{N}(p_i)) \tag{3.11}$$

der Verdrehungsgrad von X bezüglich P.

Als Gütemaß für eine Verdrehung wird also die B-Spline-Fläche möglichst gleichmäßig an einem festen Gitter im Parameterbereich abgetastet. An diesen Punkten wird die Normale berechnet und mit einem Referenzvektor verglichen. Ist der Winkel größer als ein bestimmter Wert, wird die Fläche als gedreht markiert und in beiden Knotenvektoren ein neuer Knoten der jeweiligen Ordnung eingefügt, so dass die Fläche in vier Teilflächen unterteilt wird.

Algorithmus 3.12. (Flächen-Aufteilung)

Eingabe: B-Spline-Fläche X

Ausgabe: Patch-Quadtree X mit Verweisen auf die neuen B-Spline-Flächen $X_1, \ldots, X_4$

1. Füge Knoten
$$u^* := \frac{u_0 + u_{n_u+k_u-1}}{2}, \quad v^* := \frac{v_0 + v_{n_v+k_v-1}}{2}$$
in T_u beziehungsweise T_v ein, so dass diese Vielfachheiten $k_u - 1$ beziehungsweise $k_v - 1$ haben. (Dadurch entstehen gegebenenfalls neue Kontrollpunkte und Gewichte.)

2. Suche r_u mit $u_{r_u} \le u^* < u_{r_u+1}$ und r_v mit $v_{r_u} \le v^* < v_{r_u+1}$.

3. Definiere
$$X_1(u,v) := \frac{\sum_{i=0}^{r_u-k_u+1} \sum_{j=0}^{r_v-k_v+1} w_{i,j} d_{i,j} N_{i,k_u}(u|T_{u_1}) N_{j,k_v}(v|T_{v_1})}{\sum_{i=0}^{r_u-k_u+1} \sum_{j=0}^{r_v-k_v+1} w_{i,j} N_{i,k_u}(u|T_{u_1}) N_{j,k_v}(v|T_{v_1})},$$
$$X_2(u,v) := \frac{\sum_{i=r_u-k_u+1}^{n_u-1} \sum_{j=0}^{r_v-k_v+1} w_{i,j} d_{i,j} N_{i,k_u}(u|T_{u_2}) N_{j,k_v}(v|T_{v_1})}{\sum_{i=r_u-k_u+1}^{n_u-1} \sum_{j=0}^{r_v-k_v+1} w_{i,j} N_{i,k_u}(u|T_{u_2}) N_{j,k_v}(v|T_{v_1})},$$
$$X_3(u,v) := \frac{\sum_{i=0}^{r_u-k_u+1} \sum_{j=r_v-k_v+1}^{n_v-1} w_{i,j} d_{i,j} N_{i,k_u}(u|T_{u_1}) N_{j,k_v}(v|T_{v_2})}{\sum_{i=0}^{r_u-k_u+1} \sum_{j=r_v-k_v+1}^{n_v-1} w_{i,j} N_{i,k_u}(u|T_{u_1}) N_{j,k_v}(v|T_{v_2})},$$
$$X_4(u,v) := \frac{\sum_{i=r_u-k_u+1}^{n_u-1} \sum_{j=r_v-k_v+1}^{n_v-1} w_{i,j} d_{i,j} N_{i,k_u}(u|T_{u_2}) N_{j,k_v}(v|T_{v_2})}{\sum_{i=r_u-k_u+1}^{n_u-1} \sum_{j=r_v-k_v+1}^{n_v-1} w_{i,j} N_{i,k_u}(u|T_{u_2}) N_{j,k_v}(v|T_{v_2})}$$
mit
$$T_{u_1} := [u_0, \ldots, u^*], \quad T_{u_2} := [u^*, \ldots, u_{n_u+k_u-1}],$$
$$T_{v_1} := [v_0, \ldots, v^*], \quad T_{v_2} := [v^*, \ldots, v_{n_v+k_v-1}].$$

Es ist:

$$X|_{[u_0,u^*]\times[v_0,\ldots,v^*]} = X_1, \quad X|_{[u^*,u_{n_u+k_u-1}]\times[v_0,\ldots,v^*]} = X_2,$$

$$X|_{[u_0,u^*]\times[v^*,v_{n_v+k_v-1}]} = X_3, \quad X|_{[u^*,u_{n_u+k_u-1}]\times[v^*,v_{n_v+k_v-1}]} = X_4.$$

Das Verfahren zur Unterteilung wendet man auf jeden Patch-Quadtree erneut an, bis keine Teilfläche mehr gedreht ist.

Algorithmus 3.13.

Eingabe: B-Spline-Fläche X, Parameterwerte P, maximaler Verdrehungsgrad ϵ
Ausgabe: Patch-Quadtree X, bei dem keine Teilfläche mehr gedreht ist

1. Ist $v(X,P) \geq \epsilon$:

   - Erstelle $X^* := (X_1,\ldots,X_4)$ gemäß Algorithmus 3.12.
   - Setze

$$P_1 := \{p \in P \mid p \in [u_0,u^*] \times [v_0,\ldots,v^*]\},$$

$$P_2 := \{p \in P \mid p \in [u^*,u_{n_u+k_u-1}] \times [v_0,\ldots,v^*]\},$$

$$P_3 := \{p \in P \mid p \in [u_0,u^*] \times [v^*,v_{n_v+k_v-1}]\},$$

$$P_4 := \{p \in P \mid p \in [u^*,u_{n_u+k_u-1}] \times [v^*,v_{n_v+k_v-1}]\}.$$

   - Wende diesen Algorithmus rekursiv auf $(X_i,P_i,\epsilon), i = 1,\ldots,4$, an.

Beispiel 3.14. Abbildung 3.8 zeigt eine gedrehte B-Spline-Fläche und in Abbildung 3.9 ist die zugehörige Unterteilung zu sehen.

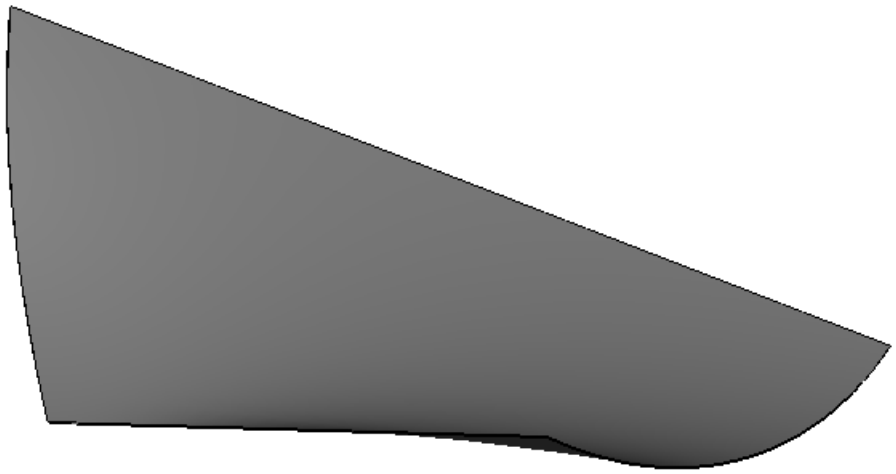


Abbildung 3.8: Gedrehte Fläche

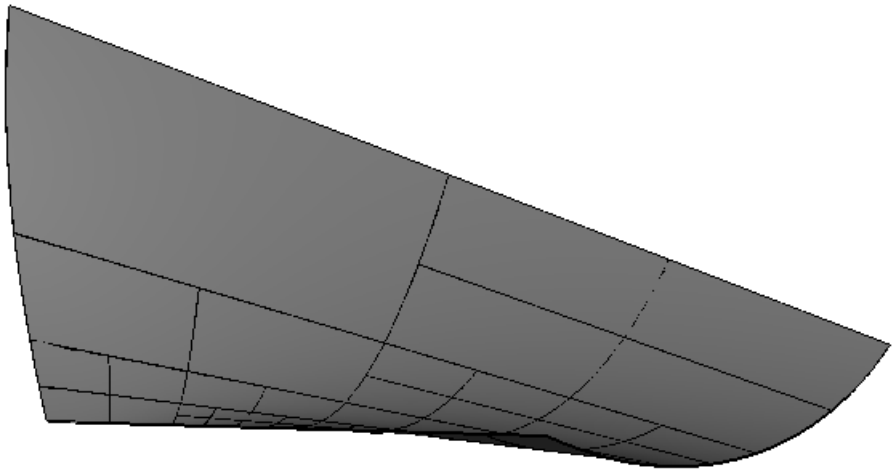


Abbildung 3.9: Unterteilte Fläche

Einschränkungen

Diese Unterteilung ist für ungetrimmte Flächen kein Problem. Bei getrimmten Flächen ist es aber nicht einfach möglich, die Trimmkurven im Parameterbereich identisch zu den Patches aufzuteilen. Abbildung 3.10 zeigt die Trimmkurven im Parameterbereich einer gedrehten Fläche. Würde man diese nach obigem Algorithmus aufteilen, entstände eine Unterteilung wie in Abbildung 3.11. Die einzelnen Subpatches bestehen also nur noch aus einem Teil der Trimmkurven. Der schraffierte Teil würde sogar komplett wegfallen. Da sich diese neuen Trimmkurven nur sehr schwer berechnen lassen, wird für eine Projektion grundsätzlich die gesamte Randkurve $\mathfrak{R}$ einer B-Spline-Fläche benutzt, wenn die Fläche gedreht ist. Ansonsten werden die tatsächlichen Trimmkurven $\mathfrak{C}$ projiziert.

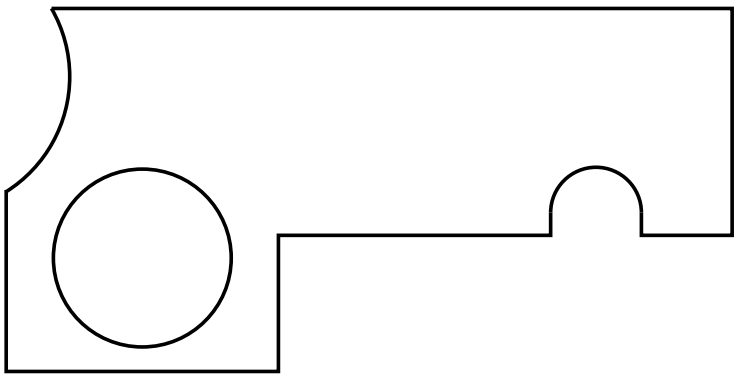


Abbildung 3.10: Trimmkurven einer gedrehten Fläche im Parameterbereich

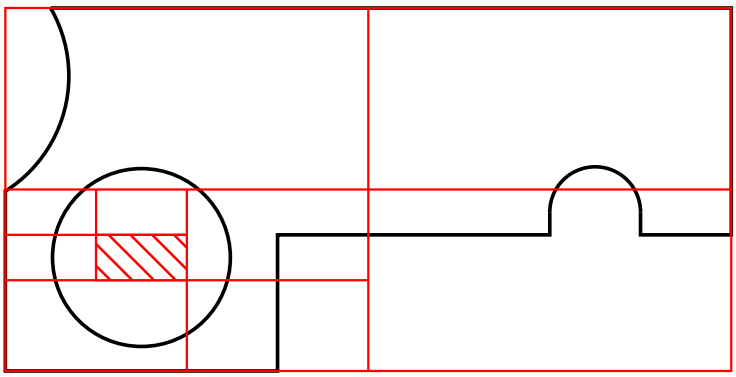


Abbildung 3.11: Aufteilung des Parameterbereichs

3.4 Näherung für gefilterte Punkte

Die Vorselektion über eine Parallelprojektion der Punkte auf die Normalenebene $\mathfrak{N}$ hat das Problem, dass gegebenenfalls zu viele Punkte entfernt werden. In Abbildung 3.12 sieht man als zweidimensionales Beispiel ein Kurve C mit Endpunkten c_0 und c_1 und deren Normalengerade E. Die Normalengerade ist dabei wie die Normalenebene als Mittel der Normalen in den Punkten von C definiert. Die Normalen von C stehen in einem maximalen Winkel von $\alpha = \frac{\pi}{4}$ zueinander. Das heißt, die Kurve ist nicht gedreht und wird nicht unterteilt. Die Punkte der Menge P gehören alle zu C, da sie bei einer Orthogonalprojektion (über die Newton-Methode) innerhalb der Kurve lägen. Bei der Vorselektion über Parallelprojektion werden die Endpunkte von C auf die Gerade E projiziert, ebenso wie die Punkte aus P. Nur die parallelprojizierten Punkte aus P, die zwischen c_0' und c_1' liegen, werden überhaupt für die Newton-Methode und Abstandsberechnung genutzt. Der Punkt p würde aussortiert werden, obwohl er (nach Orthogonalprojektion) zur Kurve gehört.

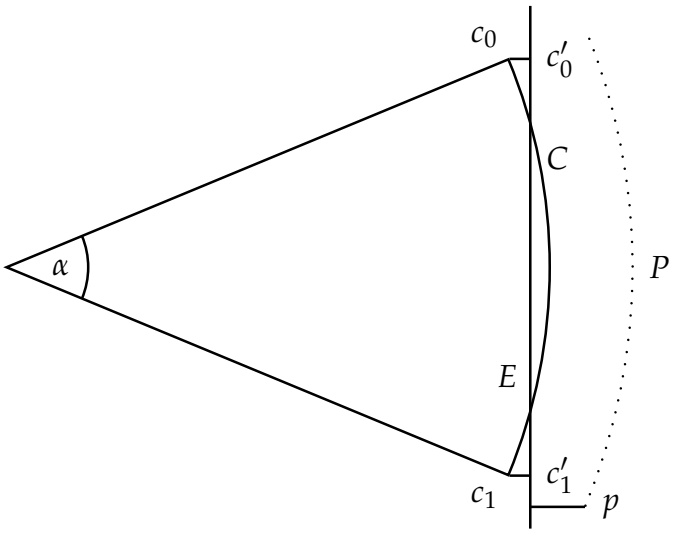


Abbildung 3.12: Projektion auf Normalengerade

Dies ist problematisch, wenn es eine andere Fläche gibt, in dessen Normalenebenen-Parallelprojektion diese Punkte liegen und zugeordnet werden. In Abbildung 3.13 sieht man dieses Phänomen, bei dem der rot eingefärbte Punkt näher an der hinteren Fläche liegt, aber der unteren Fläche zugeordnet wird (erkennbar an den Zuordnungslinien).

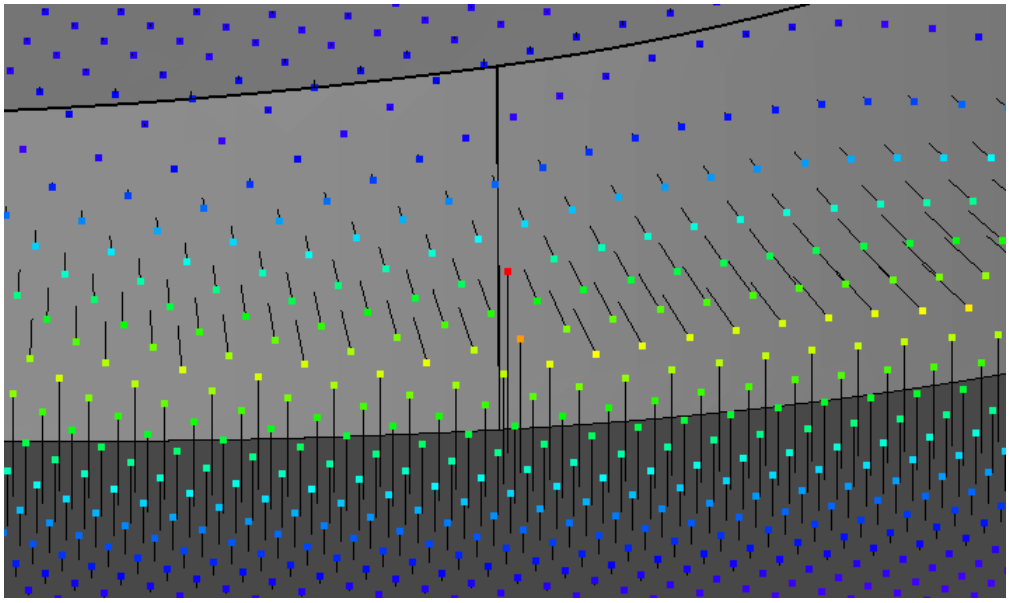


Abbildung 3.13: Falsche Zuordnung der Punkte

Es gibt verschiedene Lösungsansätze:

1. Man kann die Clipping-Region auf der Normalenebene erweitern, so dass weniger Punkte aussortiert werden. Es gibt aber keinerlei Maß, wie weit die Region erweitert werden müsste. Je weiter die Punktewolke zur zugehörigen Fläche entfernt liegt und je gedrehter eine Fläche ist, desto eher fällt ein eigentlich zugehöriger Punkt heraus.

2. Man kann anstelle einer Parallelprojektion eine Orthogonalprojektion benutzen. Hierfür verschiebt man die Normalenebene in jeden einzelnen Punkt der Punktewolke, erstellt die Flächennormalen in der Randkurve beziehungsweise in den Trimmkurven, bestimmt die Schnittpunkte der Normalen mit der Normalenebene und berechnet hierüber, ob der jeweilige Punkt innerhalb der Region liegt oder nicht.

3. Alle aussortierten Punkte liegen auf beziehungsweise in der Nähe einer Trimmkurve einer Fläche. Es reicht, wenn man diese Punkte den Trimmkurven zuordnet.

Zur Veranschaulichung soll der Würfel in Abbildung 3.14 gefräst werden. Der Würfel hat eine Kantenlänge von 1 mm, die untere Plattform ist 3 mm lang und breit. Zum Fräsen wurde mit Absicht ein im Verhältnis großer Kugelfräser mit einem Durchmesser von 1 mm benutzt. Der Bahnabstand beträgt 0.0333 mm (im hinteren Teil nur 0.1 mm, was die recht großen Frässrillen erklärt). Das Ergebnis der Frässimulation sieht man in Abbildung 3.15. Man erkennt,

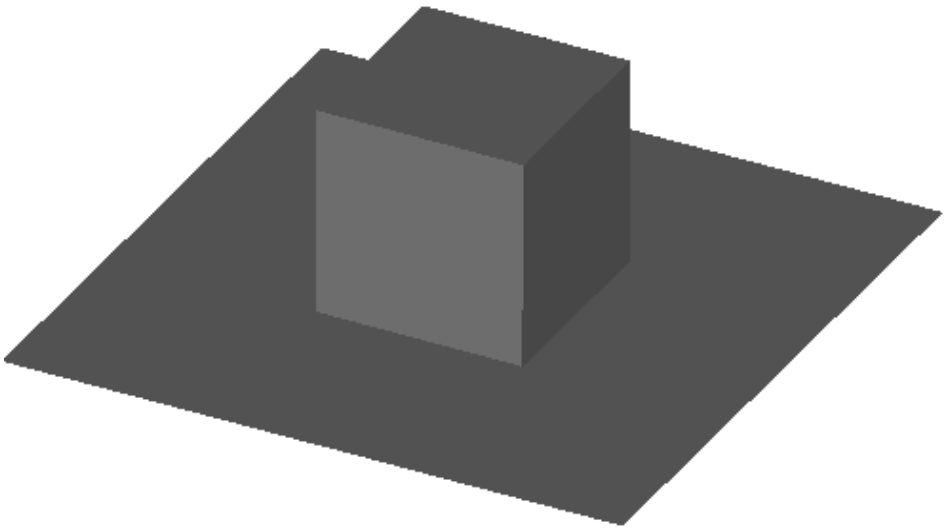


Abbildung 3.14: Würfel

dass der Kugelfräser nicht in die Ecken kommt, an denen Würfel und Grundfläche verbunden sind. Gleiches gilt auch für die vier Eckkanten des Würfels, die das eigentliche Problem sind. Schaut man sich die Punktezuordnung an, ergibt sich ein Bild wie in Abbildung 3.16. Alles was rot ist, liegt mehr als $0.1\,mm$ von der zugeordneten Fläche entfernt. Wie man sieht, entsteht an den vier Kanten ein rotes Gebiet, was dort nicht hingehört. Dies liegt daran, dass bei der zweiten Zuordnungsstufe die dort liegenden Punkte bei beiden Würfelseitenflächen herausfallen und somit der Bodenfläche zugeordnet werden. Dieses Phänomen kann auch mit der Orthogonalprojektion nicht behoben werden, die in diesem Beispiel bereits benutzt wurde.

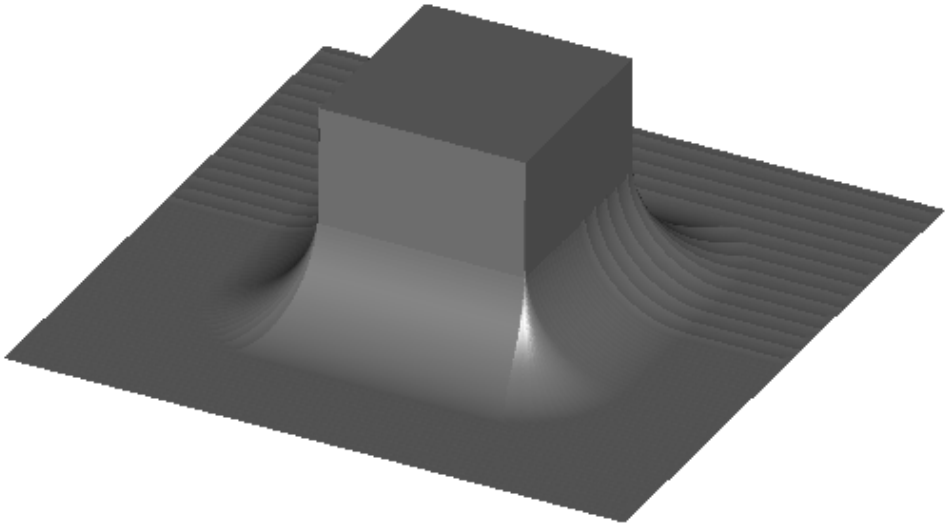


Abbildung 3.15: Gefräster Würfel

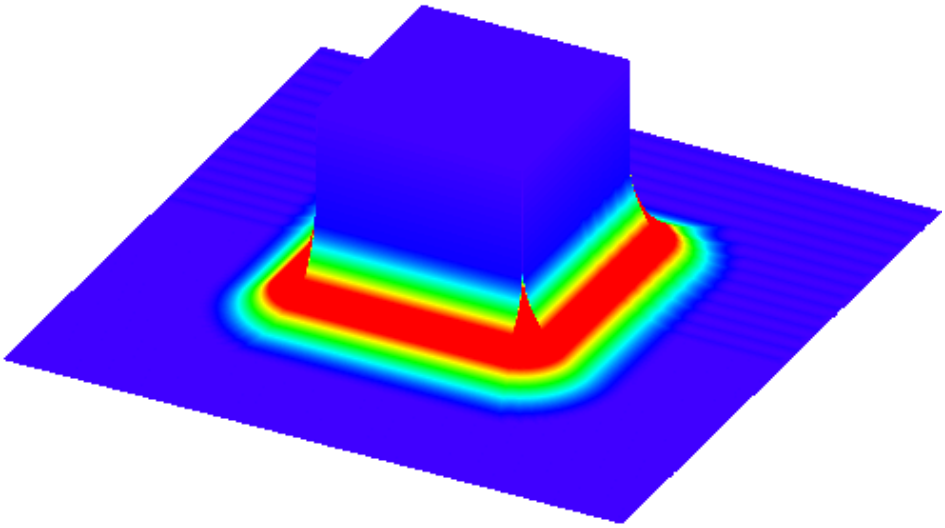


Abbildung 3.16: Falsche Punktezuordnung und Fräsfehler

Methode 1 oben wird bereits genutzt, um kleine Rundungsfehler auszuschließen. Methode 2 wurde implementiert, ist aber wesentlich langsamer als Methode 3 und kann vor allem nicht das eigentliche Problem beheben. Der Ansatz funktioniert nämlich nur korrekt, wenn die benachbarten Flächen nicht nur stetig, sondern auch differenzierbar ineinander übergehen, das heißt, wenn die Normalen in der Berührkurve identisch sind und keine Lücken entstehen lassen. Aus diesem Grund wird Methode 3 benutzt. Diese verlangsamt zwar den gesamten Algorithmus etwas, es gibt aber keine falsche Zuordnungen mehr (Abbildung 3.17).

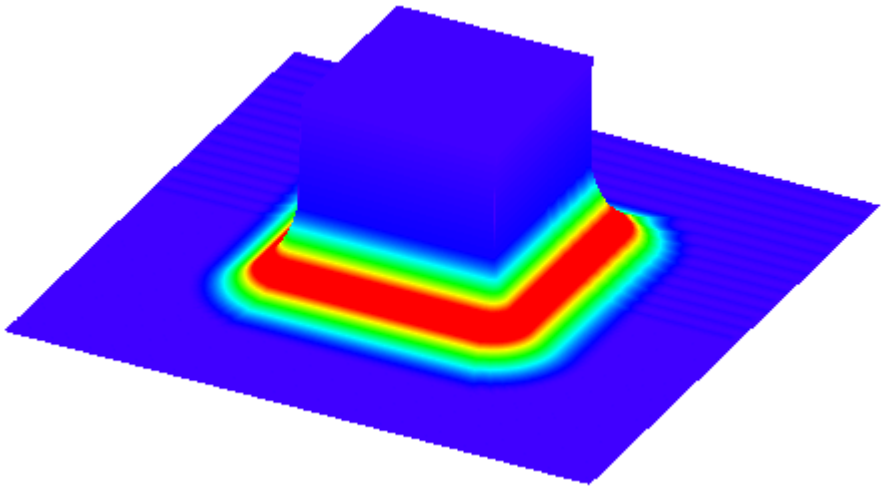


Abbildung 3.17: Korrekte Punktezuordnung und Fräsfehler

Algorithmus 3.15. (Zuordnung zu den Trimmkurven als Näherung)

Eingabe: B-Spline-Fläche X mit Trimmkurve $\mathfrak{C}$ im Bildbereich, Zuordnung Z_1 der Punktewolke $\mathfrak{P}$ über Bounding Box, Zuordnung Z_2 über Parallelprojektion auf Normalenebene $\mathfrak{N}$

Ausgabe: Zuordnung für alle Punkte in $Z := Z_1 \backslash Z_2$

1. Bestimme $Z := Z_1 \backslash Z_2$. Man sucht also alle Punkte aus der groben Zuordnung Z_1, die bei der feineren Zuordnung Z_2 herausgefallen sind.

2. Für jeden Punkt $p \in Z$:

   - Berechne
   $$q := \min_{\widehat{q} \in \mathfrak{C}} \|p - \widehat{q}\|$$
   über die Newton-Methode.
   - Bestimme $d := \|p - q\|$ und ordne die B-Spline-Fläche X zu, falls der Abstand d kleiner ist als der einer gegebenenfalls vorher zugeordneten Fläche.

Bemerkung 3.16. Die auf diese Art zugeordneten Punkte haben keinen Parameterwert und sind daher auch nicht für eine spätere Approximation (siehe Teil V, Abschnitt 2) zu verwenden.

4 Zuordnung durch Orthogonalprojektion

Die letzte Stufe der Punktzuordnung berechnet den Parameterwert (u, v) zu einem Punkt P mit Hilfe der Newton-Methode (siehe Teil II, Abschnitt 2.6).

Algorithmus 4.1. (Parameterbestimmung)

Eingabe: B-Spline-Fläche X, Punktemenge P

Ausgabe: zugeordnete Punkte P

1. Für jeden Punkt $p \in P$:

   - Bestimme Parameterwert $(u, v) \in D$ über zweidimensionale Newton-Methode, so dass $\|p - X(u, v)\|$ minimal ist. Dabei greift man auf Algorithmus 3.15 zurück, falls (u, v) außerhalb des begrenzten Parameterbereichs $\mathfrak{M}(\mathfrak{B})$ liegen würde.
   - Setze $q := X(u, v)$ und bestimme den Abstand $d := \|p - q\|$.
   - Ist d kleiner als der bisher zu p zugeordnete Abstand, ordne p der Fläche X zusammen mit (u, v) und q zu.
   - Ist (u, v) nur eine Näherung, markiere p entsprechend.

Diese Punktezuordnung muss man für alle umgewandelten CAD-Flächen der IGES-Datei vornehmen, um am Ende jeden Punkt zugeordnet zu haben.

Definition 4.2. *Es sei ein Punkt $p \in \mathfrak{P}$ gegeben. Dann heißt*

$$\vartheta(p) = \Big(\vartheta_X(p), \vartheta_q(p), \vartheta_d(p), \vartheta_{(u,v)}(p)\Big) := (X, q, d, (u, v)) \tag{4.12}$$

die Zuordnung von *p, wobei X die zugeordnete B-Spline-Fläche ist, $q \in X$ der zugeordnete Punkt mit $\|p - q\|$ minimal, $d := \|p - q\|$ der Abstand der beiden Punkte und $(u, v) \in \mathfrak{M}(\mathfrak{B})$, $X(u, v) = q$, ein optional zugeordneter Parameterwert. Zusätzlich bezeichne*

$$\Theta(\mathfrak{P}) := \{\vartheta(p) \mid p \in \mathfrak{P}\} \tag{4.13}$$

die (geordnete) Zuordnung aller Punkte der Punktewolke $\mathfrak{P}$.

Teil V
Approximation

In manchen Fällen möchte man die zu einer Fläche zugeordnete Punktmenge durch eine Tensor-Produkt-Spline-Fläche approximieren, um sie mit weniger Daten darzustellen, da die approximierende Fläche weniger Kontrollpunkte hat. Dazu wurden im letzten Kapitel die Punkte der Punktewolke den einzelnen B-Spline-Flächen zugeordnet. Bei einer Orthogonalprojektion erhielt man neben dem nächstgelegenen Punkt auf der Fläche auch noch den zugehörigen Parameterwert, den man für die Approximation benötigt.

Bevor eine Punktewolke aber durch B-Spline-Flächen approximiert wird, betrachtet man zuerst die Approximation mit B-Spline-Kurven.

1 Kurvenapproximation

Kurvenapproximation wird vor allem bei der Berechnung der Trimmkurven zu einer Fläche benötigt (siehe Teil III, Abschnitt 2). Oft hat man diese Informationen nicht in der IGES-Datei gegeben, sondern tastet die Trimmkurve an bestimmten, möglichst äquidistanten Punkten ab, projiziert diese auf die Fläche, so dass man die (nicht mehr zwingend äquidistanten) (u,v)-Parameter erhält und versucht diese Punktmenge dann durch eine zweidimensionale Kurve zu approximieren. Dabei gibt es zwei Ansätze: Die Least-Squares-Approximation mit Smoothing Splines und die Quasi-Interpolation mit dem Schoenberg-Operator.

Konvention 1.1. In diesem Abschnitt wird nur die Kurvenapproximation im $\mathbb{R}^2$ behandelt. Dies lässt sich aber genauso auf den $\mathbb{R}^3$ anwenden.

1.1 Parametrierung

Die Parametrierung $U := \{u_j \,|\, j = 0,\ldots,s-1\} \subset \mathbb{R}$ der zu approximierenden Punkte $P := \{p_j \,|\, j = 0,\ldots,s-1\} \subset \mathbb{R}^2$ hat direkten Einfluss auf das Endergebnis. Es gibt verschiedene Ansätze für dieses Problem (Piegl und Tiller [27, S. 364 f]):

- äquidistant: $u_0 := 0, u_{s-1} := 1, u_j := \frac{j}{s-1}, j = 1,\ldots,s-2$.
- chordal: Setze $d := \sum_{j=1}^{s-1} \|p_j - p_{j-1}\|$ und danach $u_0 := 0, u_{s-1} := 1, u_j := u_{j-1} + \frac{\|p_j - p_{j-1}\|}{d}, j = 1,\ldots,s-2$.
- zentripetal: Setze $d := \sum_{j=1}^{s-1} \sqrt{\|p_j - p_{j-1}\|}$ und dann $u_0 := 0, u_{s-1} := 1, u_j := u_{j-1} + \frac{\sqrt{\|p_j - p_{j-1}\|}}{d}, j = 1,\ldots,s-2$.

In vielen Fällen – wie auch hier – wird die chordale Parametrierung benutzt, da sie eine gute Repräsentation der Punkteverteilung liefert (siehe auch Teil II, Abschnitt 1.2.4). Dabei

werden die Parameterwerte aber nicht auf dem Intervall $[0,1]$, sondern auf $[0,d]$ bestimmt. Jeder Parameterwert $u_j, j = 0,\ldots,s-1$, wird dazu mit d multipliziert.

1.2 Approximation

Hat man die Parametrierung U zu den zu approximierenden Punkten P berechnet, sucht man eine polynomiale B-Spline-Kurve

$$X(t) = \sum_{i=0}^{n-1} d_i N_{i,k}(t)$$

der Ordnung k mit einem Knotenvektor T, für die gilt:

$$\sum_{j=0}^{s-1} \|X(u_j) - p_j\|_2^2 \text{ minimal.} \tag{1.1}$$

Bei iterativen Verfahren begnügt man sich damit, dass eine gewisse Fehlergrenze ϵ unterschritten wird:

$$\sum_{j=0}^{s-1} \|X(u_j) - p_j\|_2^2 \leq \epsilon. \tag{1.2}$$

Konvention 1.2. In dieser Arbeit betrachtet man nicht die Summe der Fehlerquadrate, sondern fordert, dass

$$\max_{j=0}^{s-1} \|X(u_j) - p_j\|_2^2 \leq \epsilon. \tag{1.3}$$

Dies hat den Grund, da einzelne, sehr schlechte Fehler in (1.2) zu wenig ins Gewicht fallen.

Bei der Approximationsiteration gibt es zwei verschiedene Verfahren (Piegl und Tiller [27, S. 405 ff]):

1. Beginne mit wenigen Kontrollpunkten und Knoten und approximiere die Punktdaten P. Ist (1.3) nicht erfüllt, füge mehr Knoten und Kontrollpunkte hinzu.

2. Fange mit „vielen“ Kontrollpunkten und Knoten an, so dass die Approximation Bedingung (1.3) erfüllt. Reduziere Kontrollpunkte und Knoten, bis (1.3) nicht mehr erfüllt ist und kehre zur letzten Kurve zurück.

In dieser Arbeit wird der erste Ansatz verfolgt, weil nicht so einfach entschieden werden kann, welche Knoten man beim zweiten Ansatz wählen müsste, damit (1.3) erfüllt ist. Daher wird für beide Approximationsmethoden (Least-Squares und Quasi-Interpolation) ein Initialknotenvektor

$$T := [u_0, u_0, u_0, u_0, u_{s-1}, u_{s-1}, u_{s-1}, u_{s-1}]$$

erstellt und dann werden bei der Iteration an den Stellen neue Knoten eingefügt, an denen der Fehler zu groß ist.

Es hat sich gezeigt, dass eine kubische Approximation (Ordnung $k = 4$) sehr gute Ergebnisse liefert. Eine quadratische Approximation ist oft zu „starr“ und ab einem Grad von 4 fängt die approximierende B-Spline-Kurve leichter zu schwingen an. Man beachte, dass es sich nicht um den „natürlichen“ kubischen Spline handelt, bei dem die zweiten Ableitungen im ersten und letzten Knoten Null sind (siehe [10, S. 157 ff]).

Da der Knotenvektor T und die Ordnung k vorgegeben sind, sucht man somit die Kontrollpunkte $d_0, \ldots, d_{n-1}$.

1.2.1 Least-Squares mit Smoothing Splines

Da man bei der Approximation der Punktemenge P meistens wesentlich mehr Punkte als Kontrollpunkte gegeben hat, ist das eigentlich Problem

$$AD = P \tag{1.4}$$

mit der *Kollokationsmatrix*

$$A := \begin{pmatrix} N_0(u_0) & N_1(u_0) & \cdots & N_{n-1}(u_0) \\ \vdots & \vdots & \ddots & \vdots \\ N_0(u_{s-1}) & N_1(u_{s-1}) & \cdots & N_{n-1}(u_{s-1}) \end{pmatrix} \in \mathbb{R}^{s \times n}, \tag{1.5}$$

$D \in \mathbb{R}^{n \times 2}, P \in \mathbb{R}^{s \times 2}$, überbestimmt und besitzt keine exakte Lösung mehr. In diese Fall ist man nur an der besten Approximation interessiert ist, wofür man die *Least-Squares-Methode* (Methode der kleinsten Fehlerquadrate) benutzt und die Normalengleichung

$$A^T AD = A^T P, \tag{1.6}$$

löst, wobei $A^T A$ eine symmetrische $n \times n$-Matrix ist (Dietz [8, S. 8], Piegl und Tiller [27, S. 410 ff]). (Allgemeine Informationen zu Least-Squares-Problemen findet man in Stoer und Bulirsch [35, S. 249 ff] oder Nocedal und Wright [26, S. 245 ff].)

In manchen Fällen kommt es zu Singularitäten, wenn die Punktewolke nur auf einem kleinen Gebiet relativ zur Parametrierung gegeben ist. In diesem Fall sind die Schoenberg-Whitney-Bedingungen verletzt, die besagen, dass es zur Gleichung (1.4) genau dann eine eindeutige Lösung gibt, wenn unter den Parameterwerten $u_i, i = 0, \ldots, s-1$, mindestens m Parameterwerte u_j existieren mit $N_j(u_j) > 0, j = 0, \ldots, m-1$ (Dietz [8, S. 8 f]). Es passiert dann, dass die Matrix A aus (1.5) Nullspalten enthält, da die Basis-Splines auf diesem Gebiet verschwinden. Dadurch ist die Matrix $A^T A$ trotz Überbestimmtheit des Systems singulär und das Gleichungssystem (1.6) kann nicht gelöst werden.

Dieses Problem kann mit Smoothing-Splines umgangen werden, deren eigentliche Aufgabe es ist, ein Schwingen zu unterbinden, um möglichst glatte (im geometrischen Sinn) B-Spline-Kurven zu erzeugen. Hierfür wird zu der Matrix $A^T A$ ein Korrekturterm λB addiert, wobei B *Glättematrix* heißt und λ *Glättefaktor*.

Für die Berechnung der Glättematrix benutzt man normalerweise das Integral über der Norm der zweiten Ableitung zum Quadrat

$$\int_T \|X''(t)\|_2^2 dt. \tag{1.7}$$

Für die zweite Ableitung einer B-Spline-Kurve gilt

$$X''(t) = \sum_{i=0}^{n-1} d_i N''_{i,k}(t) = D^T F \tag{1.8}$$

mit $D = (d_i)_{i=0}^{n-1} \in \mathbb{R}^{n\times 2}$ und $F = (N''_{i,k}(t))_{i=0}^{n-1} \in \mathbb{R}^n$. Zur besseren Übersicht wird die Ordnung k und das Argument t nicht weiter mit aufgeführt. Es folgt

$$\|X''\|_2^2 = \|D^T F\|_2^2 = (D^T F)^T D^T F = F^T D D^T F = D^T F F^T D. \tag{1.9}$$

Hierüber kann man das Integral bestimmen und es ist

$$\int_T \|X''\|_2^2 dt = \int_T D^T F F^T D dt = D^T \int_T F F^T dt D = D^T B D \tag{1.10}$$

mit

$$B := \int_T F F^T dt \in \mathbb{R}^{n\times n}.$$

Es ist

$$B = (B_{i,j})_{i,j=0}^{i,j=n-1}, \quad B_{i,j} := \int_T N''_{i,k}(t) N''_{j,k}(t) dt. \tag{1.11}$$

Die Einträge der symmetrischen Matrix B werden über Gauß-Legendre-Quadratur berechnet (siehe Teil II, Abschnitt 1.2.3). Das ursprüngliche Least-Squares-Problem formt sich um zu

$$(A^T A + \lambda B) D = A^T P. \tag{1.12}$$

Der Glättefaktor λ gibt an, wie stark die Glättematrix gewichtet wird. Ist der Wert recht groß, erhält man glatte B-Spline-Kurven, die sich aber nur schlecht an eine vorgegebene Punktemenge P annähern. Ist λ sehr klein, wird P gut approximiert, der approximierende Spline neigt aber zum Schwingen.

Wichtig ist es, λ geschickt zu wählen. Die beste Lösung ist es, diesen Wert während des Approximationsalgorithmus variabel zu halten und gegebenenfalls anzupassen. Wie bei der Spline-Approximation ergeben sich zwei Methoden:

1. Man fängt mit einem „großen“ λ an. Passt sich die approximierende B-Spline-Kurve nur schlecht den Punktedaten P an, wird λ verkleinert.

2. Man fängt mit einem kleinen λ oder $\lambda = 0$ an. Ist (1.3) erfüllt, vergrößert man den Wert und überprüft, ob dies immer noch gilt.

Die zweite Methode hat den Nachteil, dass singuläre Probleme mit $\lambda = 0$ immer noch singulär sind und man die Vorteile der Smoothing-Splines nicht ausnutzt. Zusätzlich kann es so passieren, dass die approximierende B-Spline-Kurve immer noch schwingt. Aus diesen Gründen wird mit einem „großen“ λ (hier: $\lambda = 10^{-10}$) begonnen, so dass die Kurve X sehr starr ist. Nur wenn sich durch das Einfügen neuer Kontrollpunkte keine Verbesserung des Fehlers erzielen lässt, wird λ verkleinert.

Zusätzlich gibt es verschiedene Ansätze, wie man λ verkleinert. Dabei kann man die Iterationsvorschrift

$$\lambda \leftarrow \frac{\lambda}{\rho}, \quad \rho > 1, \tag{1.13}$$

benutzen. Alternativ ist auch

$$\lambda \leftarrow \lambda(\epsilon_2 - \epsilon_1), \quad 0 < \epsilon_2 - \epsilon_1 < \epsilon < 1, \tag{1.14}$$

eine gute Wahl, wobei

$$\epsilon_1 := \max_{i=0}^{s-1} \|X(u_i) - p_i\|_2^2$$

und ϵ_2 der maximale Fehler der vorhergehenden Iteration ist. Man wählt λ also abhängig von der Differenz zweier Iterationsfehler. In dieser Arbeit wird eine Mischung beider Ansätze benutzt.

Algorithmus 1.3. (Least-Squares-Approximation mit Smoothing-Splines)

Eingabe: Punkte $P := \{p_0, \ldots, p_{s-1}\}$ und zugehörige Parameterwerte $U := \{u_0, \ldots, u_{s-1}\}$, maximaler Fehler ϵ

Ausgabe: B-Spline-Kurve X, so dass (1.3) erfüllt ist

1. Setze $k := 4, n := 4$ und $T = [t_0, \ldots, t_{n+k-1}] := [u_0, u_0, u_0, u_0, u_{s-1}, u_{s-1}, u_{s-1}, u_{s-1}]$.
2. Setze Fehler $\epsilon_1 := 0, \epsilon_2 := 0$, $\lambda := 10^{-10}$, $\rho := 10$ und Flag $smooth = false$, was angibt, ob λ verfeinert wurde oder nicht.
3. Schleife:
   - Erstelle Least-Squares-Matrix $A^T A$ mit A aus (1.5) und Glättematrix B aus (1.11).
   - Löse $(A^T A + \lambda B)D = A^T P$ und erstelle eine B-Spline-Kurve:

     $$X(t) := \sum_{i=0}^{n-1} d_i N_{i,k}(t).$$

   - Setze $\epsilon_2 := \epsilon_1$ und berechne

     $$\epsilon_1 := \max_{j=0}^{s-1} \|X(u_j) - p_j\|_2^2.$$

   - Ist $\epsilon_1 \leq \epsilon$, dann Ende.
   - Ist $smooth = false$ und $\epsilon_2 - \epsilon_1 < \epsilon$:
     - Setze $smooth := true$ und

       $$\lambda := \begin{cases} \frac{\lambda}{\rho}, & \epsilon_2 - \epsilon_1 \leq 0 \\ \lambda(\epsilon_2 - \epsilon_1), & \text{sonst.} \end{cases}$$

     ansonsten:

- Ist $\|X(u_j) - p_j\|_2^2 > \epsilon, j = 0, \ldots, s-1$, füge in T im Knotenintervall $[t_r, t_{r+1}]$ mit $u_j \in [t_r, t_{r+1})$ einen neuen Knoten $\frac{t_r + t_{r+1}}{2}$ ein.
- Setze $n := \#T - k$.

4. bis $\epsilon_1 \leq \epsilon$.

Bemerkung 1.4. Folgende Dinge sollten bemerkt werden:

1. Ist $u_i = t_r$, sollte man im Intervall $[t_{r-1}, t_r]$ ebenfalls einen neuen Knoten einfügen. Eine zweite Lösung ist es, vielfache Knoten zuzulassen.

2. Es kann passieren, dass $n > s$ wird. In diesem Fall muss man auf die Quasi-Interpolation (siehe unten) zurückgreifen, da die Matrix A aus (1.5) unterbestimmt ist.

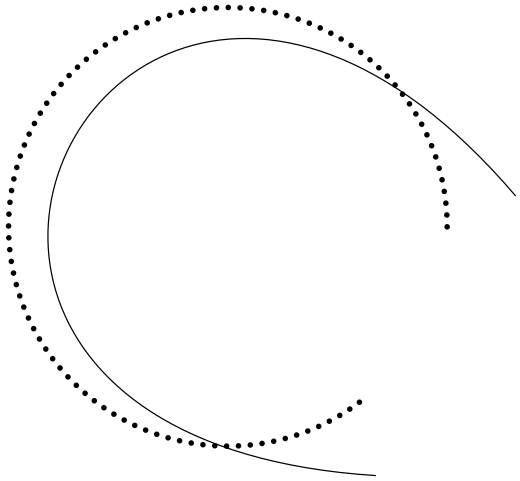
3. Bei geschickter Implementierung muss man, nachdem $smooth = true$ gesetzt wurde, die Matrizen $A^T A$ und B nicht erneut berechnen, da sich diese nicht verändert haben.

4. Wurde im Intervall $[t_{r-1}, t_r]$ für $u_j, j = 0, \ldots, s-1$, ein Knoten eingefügt, fügt man für $u_l, j < l < s-1$ keinen mehr ein.

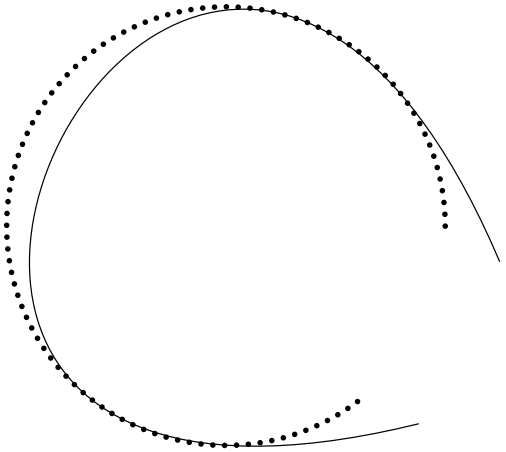
Beispiel 1.5. Es sollen 100 Punkte, die einen Kreisbogen beschreiben, mit Startwert $\lambda = 0.01$ approximiert werden. Als Approximationsfehler sei $\epsilon = 0.001$ gegeben. Abbildung 1.1 zeigt Ausschnitte aus dem Verlauf, der in Tabelle 1.1 komplett festgehalten ist. Ab Schritt 5 sieht man, dass das Hinzufügen von Knoten und Kontrollpunkten kaum eine Verbesserung der Approximation bringt beziehungsweise diese sogar verschlechtert. Es ist hier ganz allein der Glättefaktor λ, der das Ergebnis beeinflusst und in den Schritten zuvor für ein zu glattes Ergebnis sorgt.

Schritt	Fehler ϵ_1	Glättefaktor λ	Anzahl Kontrollpunkte
1	1.71364	0.01	4
2	1.48584	0.001	4
3	0.332663	0.001	5
4	0.426918	0.001	7
5	0.0780577	10^{-4}	7
6	0.121135	10^{-4}	11
7	0.0260059	10^{-5}	11
8	0.0318482	10^{-5}	19
9	0.0070059	10^{-6}	19
10	0.00734731	10^{-6}	23
11	0.00149861	10^{-7}	23
12	0.00134211	10^{-7}	25
13	0.00008988	$1.565 \cdot 10^{-11}$	25

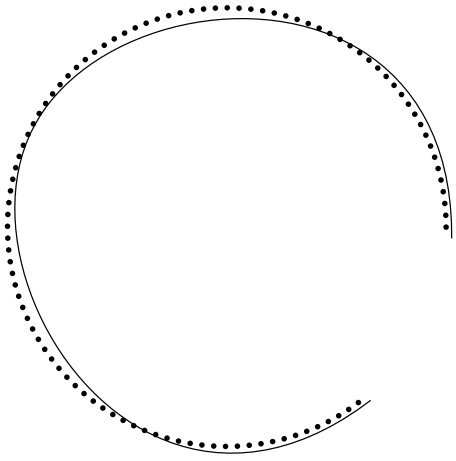
Tabelle 1.1: Approximationsverlauf per Least-Squares



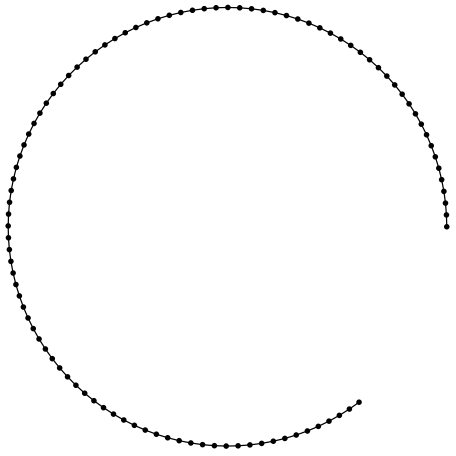
a. Schritt 1: 4 Kontrollpunkte, $\lambda = 0.01$



b. Schritt 2: 4 Kontrollpunkte, $\lambda = 0.001$



c. Schritt 3: 5 Kontrollpunkte, $\lambda = 0.001$



d. Schritt 13: 25 Kontrollpunkte, $\lambda = 1.565 \cdot 10^{-11}$

Abbildung 1.1: Kurvenapproximation per Least-Squares

Matrixstruktur

Die obige Matrix $A^T A$ hat aufgrund der Erstellung von A in (1.5) eine symmetrische Bandstruktur, wobei das Band unabhängig von der Anzahl der Kontrollpunkte eine Gesamtbreite von $2k-1$ hat (Dietz [8, S. 9 f]). Man sollte bei der Berechnung und Verarbeitung der Matrizen zwingend ein Algebra-Paket benutzen, welches solche Bandstrukturen beherrscht. Für diese Arbeit wurde auf das Open-Source-Paket *LAPACK++* von Christian Stimming [34] zurückgegriffen, welches die Klasse `LaSymmBandMatDouble` für symmetrische, bandierte, reelle Matrizen bereitstellt. LAPACK++ ist ein in C++ geschriebener Wrapper für die in Fortran vorliegenden, bekannten und vor allem schnellen Algebra-Routinen aus BLAS und LAPACK. Auch Golub und Van Loan [17, S. 19 ff] geben Hinweise, wie man solche Matrizen effizient speichern und verarbeiten kann.

1.2.2 Quasi-Interpolation mit Schoenberg-Operator

Ein zweiter Ansatz ist die Approximation mittels eines Quasi-Interpolanten, wozu auch der Schoenberg-Operator gehört.

Definition 1.6. *Sei $f : [t_0, t_{n+k-1}] \to \mathbb{R}$ eine stetige Funktion und T ein Knotenvektor mit $t_0 = \ldots = t_{k-1} < \ldots < t_n = \ldots = t_{n+k-1}$, dann heißt*

$$C_f := \sum_{i=0}^{n-1} f(c_i) N_{i,k}(\cdot), \quad c_i := \frac{t_{i+1} + \ldots + t_{i+k-1}}{k-1}, \tag{1.15}$$

der Schoenbergsche Splineoperator von *f und c_i sind die Greville-Abzissen (de Boor [6]).*

In dieser Arbeit hat man es nicht mit einer zu approximierenden Funktion f zu tun, sondern mit einer Punktmenge $P := \{p_j \,|\, j = 0, \ldots, s-1\}$ und deren Parameterwerte $U = \{u_j \,|\, j = 0, \ldots, s-1\}$. Für die Auswertung verbindet man die Punkte linear, so dass die Funktion f den zugehörigen Polygonzug beschreibt. Man definiert dazu

$$f(c_i) := p_{r_i} + \left(\frac{c_i - u_{r_i}}{u_{r_i+1} - u_{r_i}} \right) (p_{r_i+1} - p_{r_i}) \in \mathbb{R}^2, \tag{1.16}$$

wobei r_i durch $u_{r_i} \leq c_i \leq u_{r_i+1}$ gegeben ist.

Algorithmus 1.7. (Approximation mittels Schoenberg-Operator)

Eingabe: Punkte $P := \{p_0, \ldots, p_{s-1}\}$ und zugehörige Parameterwerte $U := \{u_0, \ldots, u_{s-1}\}$, maximaler Fehler ϵ

Ausgabe: B-Spline-Kurve X, so dass (1.3) erfüllt ist

1. Setze $k := 4, n := 4$ und $T = [t_0, \ldots, t_{n+k-1}] := [u_0, u_0, u_0, u_0, u_{s-1}, u_{s-1}, u_{s-1}, u_{s-1}]$.
2. Setze Fehler $\epsilon_1 := 0$.
3. Schleife:

- Berechne Kontrollpunkte

$$d_i := f(c_i), \quad i = 0, \ldots, n-1,$$

  wie in (1.16).

- Berechne

$$\epsilon_1 := \max_{i=0}^{s-1} \|X(u_i) - p_i\|_2^2.$$

- Ist $\epsilon_1 > \epsilon$:
  - Ist $\|X(u_i) - p_i\|_2^2 > \epsilon, i = 0, \ldots, s-1$, füge in T im Knotenintervall $[t_r, t_{r+1}]$ mit $u_i \in [t_r, t_{r+1})$ einen neuen Knoten $\frac{t_r + t_{r+1}}{2}$ ein.
  - Setze $n := \#T - k$.

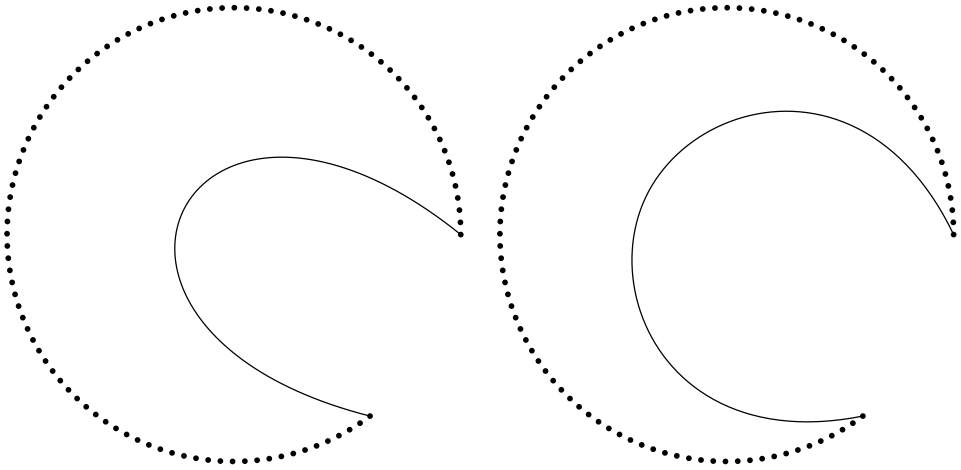
4. bis $\epsilon_1 \leq \epsilon$.

Beispiel 1.8. Es sollen die gleichen 100 Punkte wie in Beispiel 1.5 approximiert werden. Als Approximationsfehler ist $\epsilon = 0.001$ gegeben. Abbildung 1.2 zeigt Ausschnitte aus dem Verlauf, der in Tabelle 1.2 komplett festgehalten ist.

Schritt	Fehler ϵ_1	Anzahl Kontrollpunkte
1	3.76527	4
2	3.03258	5
3	1.32919	7
4	0.363662	11
5	0.0943805	19
6	0.0251005	35
7	0.00727211	67
8	0.0031628	131
9	0.00201312	229
10	0.00149624	324
11	0.000998468	360

Tabelle 1.2: Approximationsverlauf per Schoenberg-Operator

Anhand des Beispiels sieht man, wie sich die B-Spline-Kurve langsam dem Kreis annähert. Der Algorithmus läuft zwar sehr schnell und dauert nur einen Bruchteil des Least-Squares-Algorithmus, er produziert aber sehr viele Kontrollpunkte. Der Grund ist, dass die Kontrollpunkte $d_i = f(c_i)$ für die Spline-Kurve immer aus dem Polygonzug selbst heraus berechnet wurden. Aufgrund der Eigenschaft, dass eine B-Spline-Kurve immer nur in der konvexen Hüllen seines Kontrollpolygons liegt, dauert es sehr lange bis die Grenze ϵ unterschritten wird.



a. 1.Schritt: 4 Kontrollpunkte

b. 2.Schritt: 5 Kontrollpunkte

c. 3.Schritt: 7 Kontrollpunkte

d. 12.Schritt: 332 Kontrollpunkte

Abbildung 1.2: Kurvenapproximation per Schoenberg-Operator

1.2.3 Approximationsauswahl

Wann sollte man nun eine der beiden Methoden vorziehen? Es hat sich gezeigt, dass die Quasi-Interpolation vor allem bei relativ gerade verlaufenden Kurven ein besseres und schnelleres Ergebnis erzielt als die Least-Squares-Approximation.

Lemma 1.9. *Es seien Punkte* $P := \{p_i \in \mathbb{R}^2 \mid i = 0, \ldots, s-1\}$ *und zugehörige Parameterwerte* $U := \{u_i \mid i = 0, \ldots, s-1\}$ *gegeben. Dann ist die* Ausgleichsgerade *(auch* Regressionsgerade *genannt)*

$$q_1 + t(q_2 - q_1), \quad q_1, q_2 \in \mathbb{R}^2,\ t \in \mathbb{R}, \tag{1.17}$$

gegeben durch

$$q_1 := \frac{\left(\sum_{i=0}^{s-1} p_i\right)\left(\sum_{i=0}^{s-1} u_i^2\right) - \left(\sum_{i=0}^{s-1} u_i\right)\left(\sum_{i=0}^{s-1} u_i p_i\right)}{s\sum_{i=0}^{s-1} u_i - \left(\sum_{i=0}^{s-1} u_i\right)^2},$$

$$q_2 := \frac{\left(\sum_{i=0}^{s-1} p_i\right)\left(\sum_{i=0}^{s-1} u_i^2\right) - \left(\sum_{i=0}^{s-1} u_i\right)\left(\sum_{i=0}^{s-1} u_i p_i\right) + s\sum_{i=0}^{s-1} u_i p_i - \left(\sum_{i=0}^{s-1} u_i\right)\left(\sum_{i=0}^{s-1} p_i\right)}{s\sum_{i=0}^{s-1} u_i - \left(\sum_{i=0}^{s-1} u_i\right)^2}.$$

Der Beweis ergibt sich direkt aus der Auflösung der Normalengleichung

$$\begin{pmatrix} s & \sum_{i=0}^{s-1} u_i \\ \sum_{i=0}^{s-1} u_i & \sum_{i=0}^{s-1} u_i^2 \end{pmatrix} \begin{pmatrix} q_1 \\ q_2 - q_1 \end{pmatrix} = \begin{pmatrix} \sum_{i=0}^{s-1} p_i \\ \sum_{i=0}^{s-1} u_i p_i \end{pmatrix}.$$

Sind also Punkte $P := \{p_i \in \mathbb{R}^2 \mid i = 0, \ldots, s-1\}$ und ein Toleranzwinkel α gegeben, berechnet man die Ausgleichsgerade (1.17) und setzt dann

$$\sigma := \max_{i=0}^{s-2} \angle(p_{i+1} - p_i, q_2 - q_1). \tag{1.18}$$

Ist $\sigma > \alpha$, wird die Least-Squares-Approximation benutzt, ansonsten die Quasi-Interpolation.

Bemerkung 1.10. Die Krümmung $\kappa(X)$ der B-Spline-Kurve X beziehungsweise die zweite Differenz der Punktmenge P ist als Kriterium nicht geeignet. Als Beispiel sei $\epsilon_\kappa > 0$ als Kriterium zur Unterscheidung der beiden Approximationsmethoden gegeben. Es seien die Kreise

$$K_1(t) := \frac{1}{2\epsilon_\kappa}(\cos t, \sin t), \quad K_2(t) := \frac{2}{\epsilon_\kappa}(\cos t, \sin t). \quad t \in [0, 2\pi],$$

definiert. Die Krümmung eines Kreises ist immer konstant $\frac{1}{r}$, r Radius, und hier

$$\kappa(K_1) = 2\epsilon_\kappa, \quad \kappa(K_2) = \frac{1}{2}\epsilon_\kappa.$$

Demzufolge würde man bei einer Abtastung von K_1 die Quasi-Interpolation benutzen und bei K_2 die Least-Squares-Approximation, obwohl beide Funktionen bis auf Skalierung die gleiche Kurve beschreiben.

2 Flächenapproximation

Wie in der Einleitung dieses Kapitels beschrieben, ist man in manchen Fällen an einer Approximation der Punkte aus der Frässimulation interessiert. Man kann über den Approximanten, der weniger Kontrollpunkte hat als die Punktewolke, die zugeordneten Punkte als Struktur leichter darstellen. Zusätzlich ist es möglich, die Original-Spline-Fläche mit der approximierten Fläche zu vergleichen und so den Fehler zwischen beiden Flächen nicht nur an den gegebenen, diskreten Parameterwerten auszugeben. Man sollte sich aber bewusst sein, dass durch die Approximation ein neuer Fehler auftritt, der auch bei einem Vergleich mit ins Gewicht fällt. Zusätzlich dauert die Approximation von vielen Punkten unter Umständen einige Zeit (abhängig von der Leistung des ausführenden Rechners).

2.1 Parametrierung

Bei der Berechnung des Abstandes eines Punktes zu einer Fläche erhält man neben dem Abstand und dem nächstgelegenen Punkt auch den Parameterwert (u,v), solange der Wert per Orthogonalprojektion berechnet werden konnte (siehe Teil IV, Abschnitt 4). Diese Parameterwerte nimmt man für die Parametrierung. Punkte, die nur auf die Trimmkurven projiziert wurden, kann man nicht für eine Approximation verwenden (siehe Teil IV, Abschnitt 3.4).

Bemerkung 2.1. Man könnte bei auf die Trimmkurven projizierten Punkten auch über die Newton-Methode einen Parameterwert des nächstgelegenen Punktes auf der Fläche berechnen. Dies ist aber nicht sinnvoll, wie ein Beispiel zeigen soll. Es seien dazu eine lineare B-Spline-Kurve S mit einem Knotenvektor $T := [0,0,1,1]$ und Punkte $P := \{p_0,\dots,p_{84}\}$ wie in Abbildung 2.1 gegeben. Dabei liegen die ersten 75 Punkte direkt über S, Punkte p_{75} bis p_{84} rechts davon. Eine Orthogonalprojektion wäre in diesem Fall identisch zu einer Parallelprojektion und für die zugeordneten Parameterwerte $u_i, i = 0,\dots,84$, gilt:

$$u_i := \begin{cases} \frac{i}{74}, & i = 0,\dots,74, \\ 1, & i = 75,\dots,84. \end{cases}$$

Damit ist $\|S(u_i) - p_i\|$ minimal für alle $i = 0,\dots,84$. Man erhält als Gleichungssystem für die Approximation

$$\begin{pmatrix} N_{0,2}(u_0) & N_{1,2}(u_0) \\ N_{0,2}(u_1) & N_{1,2}(u_1) \\ \vdots & \vdots \\ N_{0,2}(u_{73}) & N_{1,2}(u_{73}) \\ N_{0,2}(u_{74}) & N_{1,2}(u_{74}) \\ N_{0,2}(u_{75}) & N_{1,2}(u_{75}) \\ \vdots & \vdots \\ N_{0,2}(u_{84}) & N_{1,2}(u_{84}) \end{pmatrix} D = \begin{pmatrix} 1-u_0 & u_0 \\ 1-u_1 & u_1 \\ \vdots & \vdots \\ 1-u_{73} & u_{73} \\ 1-u_{74} & u_{74} \\ 1-u_{75} & u_{75} \\ \vdots & \vdots \\ 1-u_{84} & u_{84} \end{pmatrix} D = \begin{pmatrix} 1 & 0 \\ \frac{73}{74} & \frac{1}{74} \\ \vdots & \vdots \\ \frac{1}{74} & \frac{73}{74} \\ 0 & 1 \\ 0 & 1 \\ \vdots & \vdots \\ 0 & 1 \end{pmatrix} D = \begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_{73} \\ p_{74} \\ p_{75} \\ \vdots \\ p_{84} \end{pmatrix},$$

wobei D die Matrix der gesuchten Kontrollpunkte ist. Das Gleichungssystem ist aufgrund der linear abhängigen Zeilen am Ende aber nicht mehr lösbar. Daher berechnet man zu

auf Trimmkurven projizierten Punkten keinen Parameterwert und benutzt diese Punkte auch nicht für die Approximation einer Fläche.

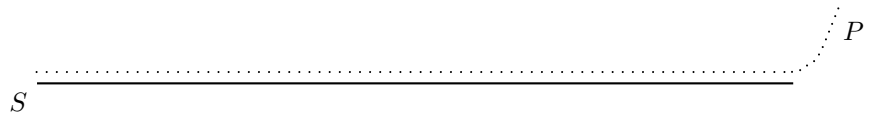


Abbildung 2.1: Angenäherte Punkte

2.2 Approximation

Es seien eine (geordnete) Punktmenge $P = \{p_j = (x_j, y_j, z_j)^T \mid j = 0, \ldots, s-1\}$ und zugehörige Parameterwerte $\{(a_j, b_j) \in \mathbb{R}^2 \mid j = 0, \ldots, s-1\}$ gegeben. Diese Werte sollen durch eine polynomiale Tensor-Produkt-Spline-Fläche

$$X(u,v) = \sum_{i=0}^{n_u-1} \sum_{j=0}^{n_v-1} d_{i,j} N_{i,k_u}(u) N_{j,k_v}(v)$$

mit Ordnungen k_u, k_v und Knotenvektoren T_u, T_v angenähert werden. Wie bei B-Spline-Kurven (siehe Abschnitt 1) soll gelten:

$$\max_{j=0}^{s-1} \|X(a_j, b_j) - p_j\|_2^2 \leq \epsilon. \tag{2.19}$$

Ebenfalls wie bei den Kurven kann man mit wenigen Kontrollpunkten anfangen und neue einfügen oder mit vielen Kontrollpunkten und welche entfernen. Es wird sich auch hier wieder für die erste Methode entschieden.

2.2.1 Auswahl der Startwerte

Es ist wichtig, gute Startwerte für die Ordnung und die Knotenvektoren zu finden, denn danach richtet sich der weitere Verlauf der Approximation. Die erste Idee ist es, alle Daten (Knoten, Ordnung, Anzahl der Kontrollpunkte) der B-Spline-Fläche zu übernehmen, der die Punkte zugeordnet wurden. Dies führt aber beispielsweise bei Ebenen, die in beide Richtungen Ordnung 2 haben, zu einer rein bilinearen Approximation, die dann nicht auf auftretende Fehler in der Punktewolke eingehen kann. Zusätzlich fangen bei zu hohem Grad die Splines leichter zu schwingen an. Aus diesem Grund werden die Ordnungen auf $k_u := 4, k_v := 4$ wie bei den B-Spline-Kurven (siehe Abschnitt 1) festgesetzt und auch während des Approximationsprozesses nicht verändert.

Die Bestimmung der Initial-Knotenvektoren ist aufgrund der vorherigen Trimmung (siehe Kapitel II, Abschnitt 2.5) leicht, denn so können T_u und T_v direkt von der Originalfläche übernommen werden.

Da die Knotenvektoren T_u, T_v und die Ordnungen k_u, k_v vorgegeben sind, sucht man somit die de-Boor-Punkte $d_{i,j}\, i = 0, \ldots, n_u - 1, j = 0, \ldots, n_v - 1$. Nach Floater [13, S. 9] lässt sich dies als lineares Gleichungssystem $AD = P$ mit der Kollokationsmatrix

$$A = \begin{pmatrix} N_{0,k_u}(a_0)N_{0,k_v}(b_0) & N_{1,k_u}(a_0)N_{0,k_v}(b_0) & \cdots & N_{n_u-1,k_u}(a_0)N_{n_v-1,k_v}(b_0) \\ \vdots & \vdots & \ddots & \vdots \\ N_{0,k_u}(a_{s-1})N_{0,k_v}(b_{s-1}) & N_{1,k_u}(a_{s-1})N_{0,k_v}(b_{s-1}) & \cdots & N_{n_u-1,k_u}(a_{s-1})N_{n_v-1,k_v}(b_{s-1}) \end{pmatrix}, \tag{2.20}$$

$A \in K^{s \times n_u n_v}$, schreiben.

2.2.2 Least-Squares-Approximation

Da man wie bei der Approximation von B-Spline-Kurven (siehe Abschnitt 1.2.1) in der Regel wesentlich mehr Punkte als Kontrollpunkte hat (s ist viel größer als $n_u n_v$), ist dieses System überbestimmt und man ist nur an der besten Approximation interessiert. Hierzu löst man die Normalengleichung

$$A^T AD = A^T P, \tag{2.21}$$

wobei $A^T A \in \mathbb{R}^{n_u n_v \times n_u n_v}$.

Wie auch bei der Approximation von Kurven kommt es in manchen Fällen zu Singularitäten, wenn die Punktewolke nur auf einem kleinen Gebiet relativ zur Parametrierung gegeben ist. Dann passiert es, dass die Matrix A Nullspalten enthält, da die Basis-Splines in diesem Gebiet konstant Null sind. Dadurch ist die Matrix $A^T A$ trotz Überbestimmtheit des Systems singulär und die Gleichung kann nicht gelöst werden. Um dennoch ein Ergebnis zu erhalten, löst man anstelle (2.21) das leicht veränderte Problem

$$(A^T A + \lambda I)D = A^T P, \quad 0 < \lambda < 1. \tag{2.22}$$

Dadurch wird die Matrix regulär und man erhält eine Lösung.

Der Nachteil dieser Methode ist, dass die Kontrollpunkte an den Stellen, wo keine Punkte aus der Punktewolke in der Nähe liegen, auf 0 gesetzt werden beziehungsweise dass die Fläche am Rand stark zu schwingen beginnt (Abbildung 2.2). Aus diesem Grund werden Smoothing-Splines benutzt, welche die Fläche an solchen undefinierten Stellen geometrisch glatt fortsetzt (Abbildung 2.3).

2.2.3 Smoothing-Splines

Smoothing-Splines für Flächen sind ähnlich definiert wie bei Kurven (siehe Abschnitt 1.2.1). Für die Berechnung der Glättematrix benutzt man normalerweise das Energiefunktional

$$E_1(f) := \frac{1}{2} \int_{T_u} \int_{T_v} \|\Delta f(u,v)\|^2 \, dv \, du \tag{2.23}$$

mit der Anwendung des *Laplace-Operators* auf die Funktion f,

$$\Delta f(u,v) = \frac{\partial^2}{\partial u^2} f(u,v) + \frac{\partial^2}{\partial v^2} f(u,v), \tag{2.24}$$

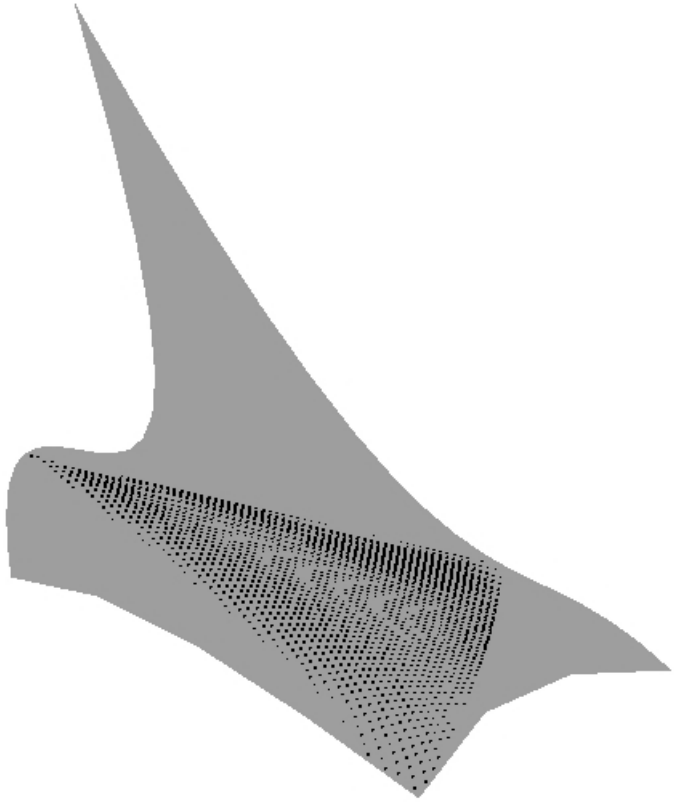


Abbildung 2.2: Approximierte Punktewolke (quasi-singuläres Problem)

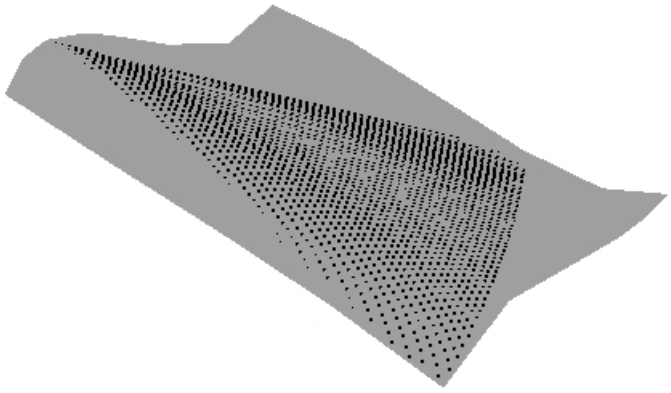


Abbildung 2.3: Approximierte Punktewolke (Smoothing-Spline)

oder das *Simple-Thin-Plate-Energiefunktional* (Dietz [8, S. 43], Floater [13, S. 8])

$$E_2(f) := \int_{T_u}\int_{T_v} \|H_f\|_F^2 \, dv\, du \tag{2.25}$$

mit Hessematrix H_f und *Frobenius-Norm*

$$\|H_f\|_F^2 = \left(\frac{\partial^2}{\partial u^2} f(u,v)\right)^2 + 2\left(\frac{\partial^2}{\partial u \partial v} f(u,v)\right)^2 + \left(\frac{\partial^2}{\partial v^2} f(u,v)\right)^2. \tag{2.26}$$

In dieser Arbeit wird wegen der gemischten Ableitungen das Funktional (2.25) benutzt.

Setzt man eine B-Spline-Fläche X in (2.25) ein, erhält man

$$\begin{aligned} B := E_2(X) &= \int_{T_u}\int_{T_v} \left(\frac{\partial^2}{\partial u^2} X(u,v)\right)^2 + 2\left(\frac{\partial^2}{\partial u \partial v} X(u,v)\right)^2 + \left(\frac{\partial^2}{\partial v^2} X(u,v)\right)^2 dv\, du, \\ &= \int_{T_u}\int_{T_v} \left(\frac{\partial^2}{\partial u^2} X(u,v)\right)^2 dv\, du + 2\int_{T_u}\int_{T_v} \left(\frac{\partial^2}{\partial u \partial v} X(u,v)\right)^2 dv\, du + \\ &\quad \int_{T_u}\int_{T_v} \left(\frac{\partial^2}{\partial v^2} X(u,v)\right)^2 dv\, du. \end{aligned} \tag{2.27}$$

Dies kann man auch schreiben als

$$B = \int_{T_u}\int_{T_v} F_{u,u} F_{u,u}^T \, dv\, du + 2\int_{T_u}\int_{T_v} F_{u,v} F_{u,v}^T \, dv\, du + \int_{T_u}\int_{T_v} F_{v,v} F_{v,v}^T \, dv\, du,$$

wobei

$$F_{u,u} = \begin{pmatrix} N''_{0,k_u}(u) N_{0,k_v}(v) \\ N''_{0,k_u}(u) N_{1,k_v}(v) \\ \vdots \\ N''_{n_u-1,k_u}(u) N_{n_v-1,k_v}(v) \end{pmatrix}, \quad F_{u,v} = \begin{pmatrix} N'_{0,k_u}(u) N'_{0,k_v}(v) \\ N'_{0,k_u}(u) N'_{1,k_v}(v) \\ \vdots \\ N'_{n_u-1,k_u}(u) N'_{n_v-1,k_v}(v) \end{pmatrix},$$

$$F_{v,v} = \begin{pmatrix} N_{0,k_u}(u) N''_{0,k_v}(v) \\ N_{0,k_u}(u) N''_{1,k_v}(v) \\ \vdots \\ N_{n_u-1,k_u}(u) N''_{n_v-1,k_v}(v) \end{pmatrix}, \quad F_{u,u}, F_{u,v}, F_{v,v} \in \mathbb{R}^{n_u n_v}.$$

Das Integral kann man in die Matrizen hineinziehen und mit dem Satz von Fubini aufteilen. Man muss damit nur die sechs Matrizen

$$\left(\int_{T_u} N_i'' N_k'' \, du\right)_{i=0,k=0}^{n_u-1,n_u-1}, \quad \left(\int_{T_u} N_i' N_k' \, du\right)_{i=0,k=0}^{n_u-1,n_u-1}, \quad \left(\int_{T_u} N_i N_k \, du\right)_{i=0,k=0}^{n_u-1,n_u-1},$$

$$\left(\int_{T_v} N_j N_l \, dv\right)_{j=0,l=0}^{n_v-1,n_v-1}, \quad \left(\int_{T_v} N_j' N_l' \, dv\right)_{j=0,l=0}^{n_v-1,n_v-1}, \quad \left(\int_{T_v} N_j'' N_l'' \, dv\right)_{j=0,l=0}^{n_v-1,n_v-1}$$

berechnen und die Einträge korrekt verbinden.

Damit kann man das eigentliche Least-Squares-Problem 2.21 umformen zu

$$(A^T A + \lambda B) D = A^T P, \tag{2.28}$$

wobei λ wieder der Glättefaktor ist (siehe Kurvenapproximation in Abschnitt 1.2.1).

2.2.4 Approximationsverfeinerung

Die Approximation der Punktewolke im ersten Schritt ist selten optimal. Daher werden solange Kontrollpunkte beziehungsweise Knoten eingefügt, bis (2.19) erfüllt ist. Der Glättewert λ wird dabei identisch zur Kurvenapproximation bestimmt, nur dass man mit einem wesentlich größeren Startwert $\lambda = 0.1$ anfängt. Dies hat den Grund, dass die Punkteverteilung im dreidimensionalen Raum sehr ungleichmäßig sein kann und die approximierende B-Spline-Fläche leichter anfängt zu schwingen.

Der Algorithmus ist fast identisch zu Algorithmus 1.3.

Algorithmus 2.2. (Least-Squares-Approximation mit Smoothing-Splines)

Eingabe: Punkte $P = \{p_j \in \mathbb{R}^3 \,|\, j = 0, \ldots, s-1\}$ und zugehörige Parameterwerte $\{(a_j, b_j) \in \mathbb{R}^2 \,|\, j = 0, \ldots, s-1\}$, maximaler Fehler ϵ

Ausgabe: B-Spline-Fläche X, so dass (2.19) erfüllt ist

1. Setze $k_u := 4, k_v := 4, n_u := 4, n_v :=$ und sei

$$a_1^* := \min_{j=0}^{s-1} a_j, \quad a_2^* := \max_{j=0}^{s-1} a_j, \quad b_1^* := \min_{j=0}^{s-1} b_j, \quad b_2^* := \max_{j=0}^{s-1} b_j.$$

2. Setze $T_u = [u_0, \ldots, u_{n_u+k_u-1}] := [a_1^*, a_1^*, a_1^*, a_1^*, a_2^*, a_2^*, a_2^*, a_2^*]$, $T_v = [v_0, \ldots, v_{n_v+k_v-1}] := [b_1^*, b_1^*, b_1^*, b_1^*, b_2^*, b_2^*, b_2^*, b_2^*]$.

3. Setze Fehler $\epsilon_1 := 0, \epsilon_2 := 0$, $\lambda := 10^{-1}$, $\rho := 10$ und Flag $smooth = false$, was angibt, ob λ verfeinert wurde oder nicht.

4. Schleife:

   - Erstelle Least-Squares-Matrix $A^T A$ mit A aus (2.20) und Glättematrix B (2.27).
   - Löse $(A^T A + \lambda B)D = A^T P$ und erstelle eine B-Spline-Fläche:

$$X(u, v) := \sum_{i=0}^{n_u-1} \sum_{j=0}^{n_v-1} d_{i,j} N_{i,k_u}(u) N_{j,k_v}(v).$$

   - Setze $\epsilon_2 := \epsilon_1$ und berechne

$$\epsilon_1 := \max_{j=0}^{s-1} \|X(a_j, b_j) - p_j\|_2^2.$$

   - Ist $\epsilon_1 \leq \epsilon$, dann Ende.
   - Ist $smooth = false$ und $\epsilon_2 - \epsilon_1 < \epsilon$:
     - Setze $smooth := true$ und

$$\lambda := \begin{cases} \frac{\lambda}{\rho}, & \epsilon_2 - \epsilon_1 \leq 0 \\ \lambda(\epsilon_2 - \epsilon_1), & \text{sonst.} \end{cases}$$

ansonsten:

- Ist $\|X(a_j, b_j) - p_j\|_2^2 > \epsilon, j = 0, \ldots, s-1$, füge in T_u im Knotenintervall $[u_{r_u}, u_{r_u+1}]$ mit $a_j \in [u_{r_u}, u_{r_u+1})$ einen neuen Knoten $\frac{u_{r_u}+u_{r_u+1}}{2}$ und in T_v im Knotenintervall $[v_{r_v}, v_{r_v+1}]$ mit $b_j \in [v_{r_v}, v_{r_v+1})$ einen neuen Knoten $\frac{v_{r_v}+v_{r_v+1}}{2}$ ein (siehe auch Piegl und Tiller [27, S. 427]).
- Setze $n_u := \#T_u - k_u$ und $n_v := \#T_v - k_v$

5. bis $\epsilon_1 \leq \epsilon$.

Bemerkung 2.3. Es gelten die gleichen Bemerkungen wie für B-Spline-Kurven in Bemerkung 1.4. Man sollte zusätzlich beachten, dass das Einfügen eines Knotens nicht einen Kontrollpunkt, sondern eine ganze Reihe zusätzlicher Kontrollpunkte erzeugt. Auf diese Art kann das Problem sehr schnell sehr groß werden.

Beispiel 2.4. Als Beispiel betrachtet man die Punktewolke in Abbildung 2.4. Die Fläche ist leicht gebogen, vor allem am oberen linken Rand gibt es eine starke Veränderung. Es ist zu erwarten, dass hier viele Kontrollpunkte eingefügt werden müssen, um ein adäquates Ergebnis zu erreichen. Abbildung 2.5 zeigt die Kontrollpunktereihen in den einzelnen Schritten in der Draufsicht. Tabelle 2.1 zeigt den gesamtem Fortlauf mitsamt der Größe des Kontrollnetzes und des maximalen Fehlers.

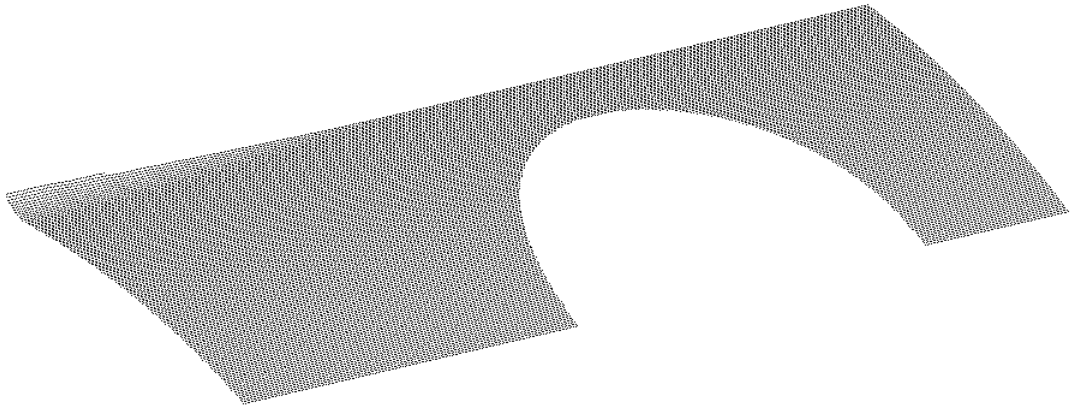
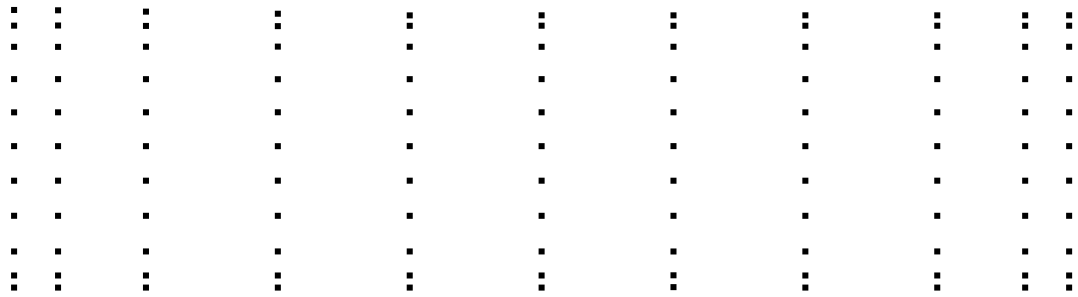


Abbildung 2.4: Punktewolke, dc.igs, Patch 1

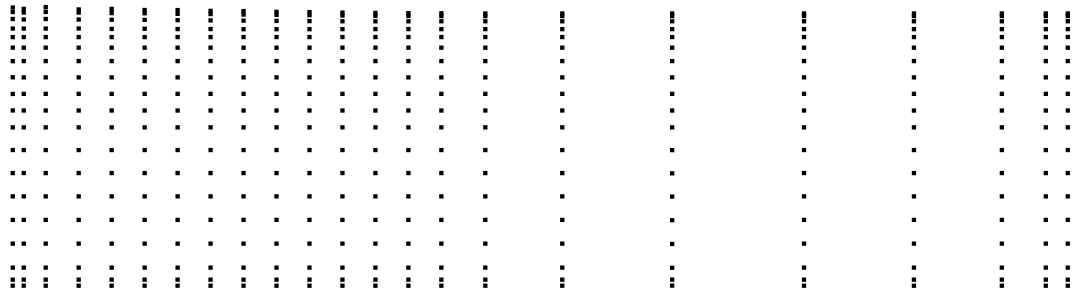
Bemerkung 2.5. Ein großes Problem ist, dass sehr viele Kontrollpunkte berechnet werden müssen, um die gewünschte Grenze ϵ unterschreiten zu können. Dies wirkt sich einmal auf die Dauer der Berechnung aus, die in obigem Beispiel bei der letzten 1682×1682-Matrix immerhin bei 36 Sekunden lag[1], und zum anderen auf den Speicherverbrauch, der im letzten Schritt bei über 500 MB liegt. Dies ist darauf zurückzuführen, dass die Matrix $A^T A$ zwar dünn besetzt ist (siehe nächster Abschnitt), aber die Matrix A zuerst berechnet werden muss. In dem Beispiel handelte es sich um circa 40.000 Punkte bei einer Punktauflösung von $200\,\mu m$.

[1] Auf einem Laptop mit 1.73 GHz Rechenleistung und 1 GB Arbeitsspeicher.

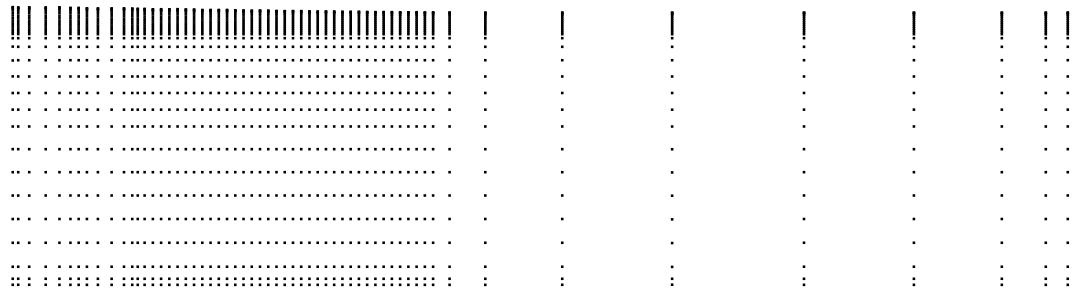
a. Schritt 1: 4×4 Kontrollpunkte



b. Schritt 5: 11×11 Kontrollpunkte



c. Schritt 8: 23×19 Kontrollpunkte



d. Schritt 11: 58×29 Kontrollpunkte

Abbildung 2.5: Approximationsverfeinerung bei Patch-Approximation

Schritt	max. Fehler e_i	Glättefaktor λ	Anzahl Kontrollpunkte
1	0.615977	0.1	4×4
2	0.615943	0.01	4×4
3	0.477840	0.01	5×5
4	0.221741	0.01	7×7
5	0.076608	0.01	11×11
6	0.072152	0.01	16×17
7	0.069577	$4.45 \cdot 10^{-5}$	16×17
8	0.052958	$4.45 \cdot 10^{-5}$	23×19
9	0.024120	$4.45 \cdot 10^{-5}$	36×22
10	0.010927	$4.45 \cdot 10^{-5}$	57×28
11	0.009158	$4.45 \cdot 10^{-5}$	58×29

Tabelle 2.1: Approximationsverfeinerung bei Patch-Approximation

Es kann hier helfen, wenn man anstelle der globalen Glättematrix eine lokale Gewichtsfunktion benutzt, die abhängig vom Parameterwert den Glättefaktor λ steuert (Dietz [8, S. 82 ff]).

2.2.5 Matrixstruktur

Wie auch bei der B-Spline-Approximation hat die Matrix $A^T A$ eine spezielle symmetrische Bandstruktur, die man ausnutzen sollte (Dietz [8, S. 14 ff], Floater [13, S. 10 f]). In diesem Fall besteht die Matrix aus einem großen Band, dessen Weite sich aus dem Produkt von k_1 und k_2 ergibt, und darin enthalten sind wiederum viele kleine symmetrische Bandmatrizen mit einer Bandweite von k_1 oder k_2, je nachdem welche Richtung man bei der Speicherung bevorzugt.

In Tabelle 2.2 ist dargestellt, welche Matrixstruktur wie viel Speicherplatz beanspruchen müsste. Hierbei soll die Tensor-Produkt-Spline-Fläche $n_u \times n_v$ Kontrollpunkte und eine Ordnung von k_u beziehungsweise k_v besitzen.

Matrix	Anzahl Elemente
voll besetzt	$n_u^2 n_v^2$
symmetrisch	$\frac{(n_u n_v)(n_u n_v+1)}{2}$
bandiert	$(n_u(2(k_u-1)+1)-(n_u-k_u)(n_u-k_u+1))n_v^2$
symmetrisch bandiert	$\frac{n_u(n_u+1)-(n_u-k_1)(n_u-k_u+1)}{2} n_v^2$
symmetrisch doppelt bandiert	$\frac{n_u(n_u+1)-(n_u-k_u)(n_u-k_u+1)}{2} \frac{n_v(n_v+1)-(n_v-k_v)(n_v-k_v+1)}{2}$

Tabelle 2.2: Zu speichernde Elemente in einer Matrix

Teil VI
Visualisierung des Fräsfehlers

Nachdem die Punkte aus der Punktewolke den einzelnen Flächen zugeordnet sind und diese gegebenenfalls noch durch Tensor-Produkt-Spline-Flächen approximiert wurden, kann man den Fräsfehler in den einzelnen Punkten berechnen und farblich darstellen.

1 Fehlerdarstellung

Die Visualisierung des Fehlers geschieht über den Abstand eines Punktes aus der Punktewolke zum zugeordneten Punkt auf der B-Spline-Fläche. Da bei jeder Punktezuordnung auch der zugehörige Punkt auf der Fläche gespeichert wurde, ist ein Vergleich sehr leicht.

Es seien eine Punktewolke $\mathfrak{P} := \{p_j \,|\, j = 0,\ldots,s-1\}$ und die Zuordnung $\Theta(\mathfrak{P}) := \{\vartheta(p_j) \,|\, j = 0,\ldots,s-1\}$ gegeben. Dann ist der Fehler für $p \in \mathfrak{P}$ durch $\vartheta_d(p)$ bestimmt. Die Differenzvektoren $p - \vartheta_q(p)$ können in einer Datei abgespeichert und im VisuTool geladen werden. Dort kann der Fehler dann mit beliebigen Toleranzgrenzen für Minimum und Maximum eingefärbt werden (Abbildung 1.1). Blau bedeutet in der Abbildung einen Fehler von 0 und rot stellt einen Fehler über 10 μm dar.

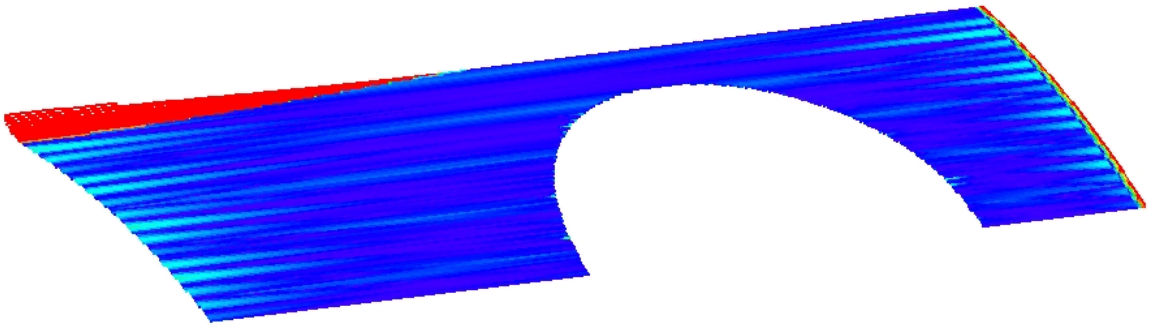


Abbildung 1.1: Fehlerdarstellung: Punkte – Originalpatch

Hat man den zu einer Fläche X zugeordneten Punktebereich

$$\mathfrak{P}_X := \{p \in \mathfrak{P} \,|\, \vartheta_X(p) = X\}$$

noch als Tensor-Produkt-Spline-Fläche X^* approximiert (siehe Teil V), hat man zwei weitere Vergleichsmöglichkeiten.

1.1 Punktewolke und approximierende Fläche

Es seien eine B-Spline-Fläche X, die zugeordneten Punkte $\mathfrak{P}_X$, die Zuordnung $\Theta(\mathfrak{P}_X)$ und der zugehörige Approximant X^* gegeben. Dann stellt

$$X^*(\vartheta_{(u,v)}(p)) - p, \quad p \in \mathfrak{P}_X, \tag{1.1}$$

den Fehler zwischen der Punktewolke und der approximierenden Fläche dar (Abbildung 1.2). Das Optimum wäre, wenn die gesamte Fläche komplett blau eingefärbt wäre, dann hätte man eine Interpolationseigenschaft an allen Punkten. In der Regel erhält man diese Eigenschaft aber nicht.

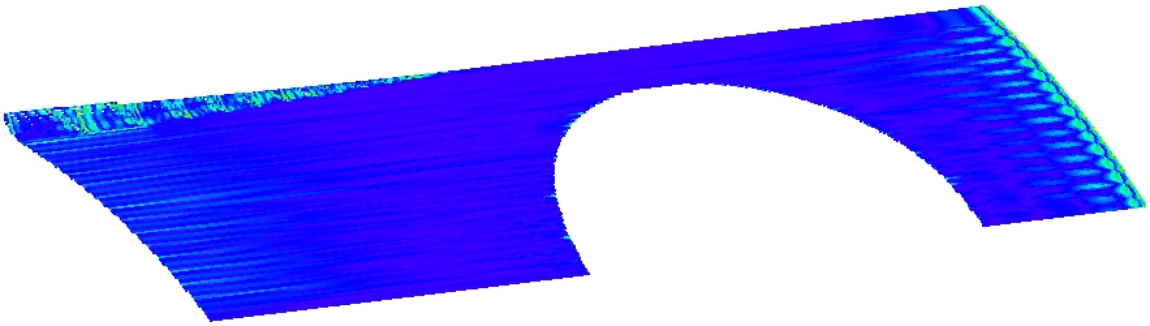


Abbildung 1.2: Fehlerdarstellung: Punkte – Approximierende Fläche

1.2 Originalfläche und approximierende Fläche

Es seien eine B-Spline-Fläche X mit Parametergebiet D, die zugeordneten Punkte $\mathfrak{P}_X$, die Zuordnung $\Theta(\mathfrak{P}_X)$ und der zugehörige Approximant X^* mit Parametergebiet D^* gegeben. Da $D^* \subseteq D$ gewählt wurde, kann man den Fehler leicht bestimmen.

Definition 1.1. *Es seien polynomiale B-Spline-Flächen*

$$X(u,v) := \sum_{i=0}^{n_u-1} \sum_{j=0}^{n_v-1} d_{i,j} N_{i,k_u}(u|T_u) N_{i,k_v}(v|T_v),$$

$$X^*(u,v) := \sum_{i=0}^{n_u-1} \sum_{j=0}^{n_v-1} d^*_{i,j} N_{i,k_u}(u|T_u) N_{i,k_v}(v|T_v)$$

mit $T_u := [u_0, \ldots, u_{n_u+k_u-1}]$, $T_u := [v_0, \ldots, v_{n_v+k_v-1}]$ *gegeben. Definiere*

$$(X - X^*)(u,v) := \sum_{i=0}^{n_u-1} \sum_{j=0}^{n_v-1} \left(d_{i,j} - d^*_{i,j} \right) N_{i,k_u}(u|T_u) N_{i,k_v}(v|T_v). \tag{1.2}$$

Satz 1.2. *Es seien polynomiale B-Spline-Flächen X und X^* wie in Definition 1.1 gegeben. Dann gilt:*

$$(X - X^*)(u,v) = X(u,v) - X^*(u,v).$$

Dies folgt aus der Konvexen-Hüllen-Eigenschaft und gilt auch für polynomiale B-Spline-Kurven. Man kann den Fehler zwischen zwei polynomialen B-Spline-Flächen somit darstellen, indem man beide Flächen auf den gleichen Grad erhöht und die Knotenvektoren durch Knoteneinfügen gleichsetzt.

2 Vergleich mit Net-Compare

Ein bereits existierendes Verfahren für die Visualisierung von Fräsfehlern ist *Net-Compare*. Dieses ordnet eine Punktewolke einer triangulierten Basisfläche zu. Es werden dabei zuerst die Dreiecke in einem Binary-Space-Partition-Tree (siehe Teil IV, Abschnitt 3.2.2) einsortiert und danach verglichen, zu welchem Dreieck ein Punkt aus der Punktewolke den kleinsten Abstand hat und diesem Dreieck zugeordnet.

Net-Compare hat aber zwei große Nachteile:

1. Die Daten der Vergleichsfläche, das heißt der Flächen aus der IGES-Datei, müssen als trianguliertes Gitter gegeben sein, um einen Vergleich zu starten.

2. Ein Triangulierung bedeutet immer eine diskrete Abtastung einer Fläche und damit einen zusätzlichen Fehler beim Vergleich.

2.1 Triangulierung der IGES-Daten

Die konvertierten Tensor-Produkt-Spline-Flächen aus der IGES-Datei müssen vor einem Vergleich mit Net-Compare zuerst trianguliert werden. Dies kann auf zwei Arten geschehen. Man tastet entweder die Fläche im Dreidimensionalen ab und trianguliert die entstehenden Punkte oder man tastet nur den Parameterbereich ab, trianguliert diese 2-D-Daten und erstellt hierüber die Dreiecke. Da die erste Methode komplizierter und auch langsamer ist, wurde die zweite gewählt. Eine Einführung in die Triangulierung und deren Verfahren findet man in Hjelle und Daehnen [21].

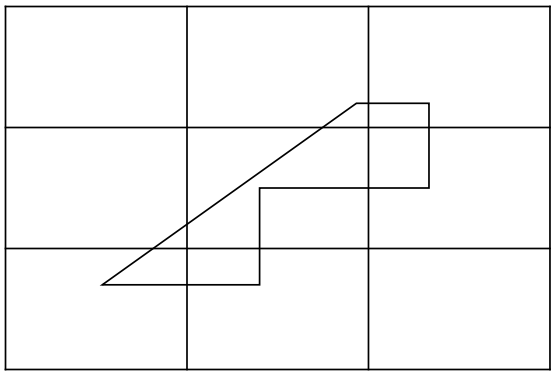


Abbildung 2.1: Planar Straight Line Graph

Die reine Triangulierung von ungetrimmten B-Spline-Flächen ist sehr leicht, da man nur ein Gitter aus dem rechteckigen Parameterbereich erstellen muss. Im getrimmten Fall hat man aber zusätzlich noch die Trimmkurve als *Planar Straight Line Graph (PSLG)* gegeben (Abbildung 2.1). Dies ist ein ebener, gerichteter und geschlossener Polygonzug. Für diesen Fall wird die *Constrained Delaunay-Triangulation* benutzt. Die Delaunay-Eigenschaft stellt sicher, dass die erzeugte Triangulierung eine gute Qualität (keine zu kleinen Winkel bei den Dreiecken)

hat. *Constrained* bedeutet, dass jede Verbindungslinie aus dem PSLG auch wieder als eine Seite eines Dreieckes in der Triangulierung auftaucht.

Beispiel 2.1. Als Beispiel soll der PSLG aus Abbildung 2.1 in ein vorhandenes Gitter eingebettet werden. Die Idee wäre es, den Graph an jedem Rechteck zu clippen (Abbildung 2.2a) und dann die vorhandenen Punkte (Abbildung 2.2b) zu triangulieren. Dabei entsteht aber unter Umständen eine Triangulierung, die nicht alle Verbindungslinien enthält (Abbildung 2.2c). Um diese einzufügen, hat man zwei Möglichkeiten. Entweder man fügt die Linie ein, indem man die Schnittpunkte mit den Dreiecksseiten sucht und so neue Dreiecke erstellt (Abbildung 2.2d) oder man entfernt alle Dreicke, durch die die Linie läuft und trianguliert vorhandene Gebiete rechts und links dieser Linie neu (Abbildung 2.2e).

Da diese Verfahren sehr komplex sind, wurde auf eine externe Lösung zurückgegriffen. Es kommt das Open-Source-Programm *Triangle* von J. R. Shewchuk [30] zum Einsatz. Shewchuk [29] gibt in seiner Beschreibung zu dem Programm einen guten Einblick in vorhandene Triangulierungsalgorithmen und erklärt, wie Triangle funktioniert. In dieser Arbeit wird eine vorhandene Triangulierung (auf Basis des ungetrimmten Parameterbereichs) durch einen PSLG erweitert.

Um ein gutes Ergebnis zu erhalten, sollten die Dreiecke in etwa alle die gleiche Größe haben. In den seltensten Fällen spiegelt das Seitenverhältnis des Parameterbereichs einer Fläche deren Gestalt im Bildbereich wieder. Zum Beispiel kann eine Ebene auf dem Gebiet $[0,1] \times [0,1]$ definiert sein, im Bildbereich aber die Eckpunkte $(0,0,0), (1000,0,0), (0,1,0)$ und $(1000,1,0)$ besitzen. Würde man nun ein gleichmäßiges Gitter im Parameterbereich erstellen und eine Triangulierung hierauf berechnen, hätten die späteren 3-D-Dreiecke sehr spitze Winkel und würden bei der Darstellung zu Artefakten führen (Abbildung 2.3).

Aus diesem Grund werden alle Parametergebiete der B-Spline-Flächen so umparametriert, dass ihr Seitenverhältnis in etwa dem der dreidimensionalen Gestalt entspricht.

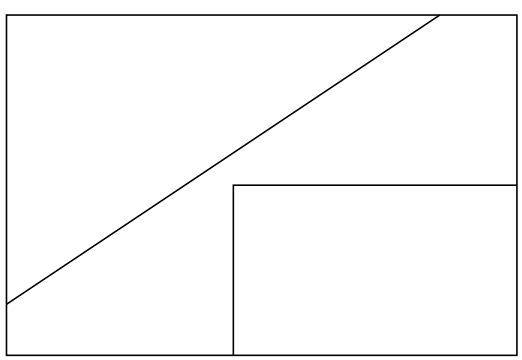
Algorithmus 2.2. (Reparametrierung)

Eingabe: B-Spline-Fläche X mit Parametergebiet D und Knotenvektoren T_u, T_v

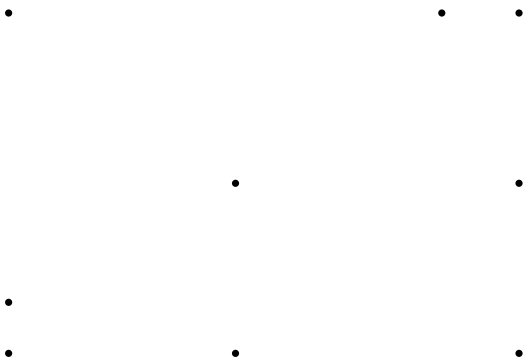
Ausgabe: B-Spline-Fläche X mit neuem Parametergebiet D^* und Knotenvektoren T_u^*, T_v^*

1. Berechne
$$L_{\max}^{(u)} := \max_{i=0}^{n_u-1} L_i^{(u)}, \quad L_{\max}^{(v)} := \max_{j=0}^{n_v-1} L_j^{(v)}$$
(siehe Definition 2.13 in Teil II, Abschnitt 2.3).

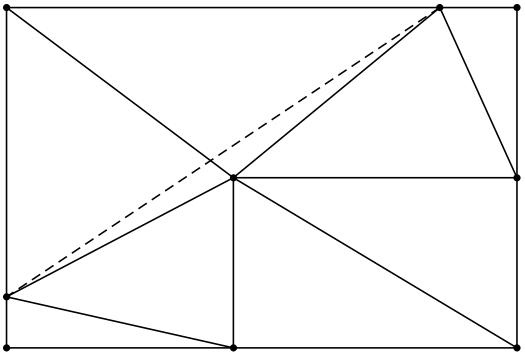
2. Setze
$$T_u^* := \left[u_0 + L_{\max}^{(u)} \frac{u_i - u_0}{u_{n_u+k_u-1} - u_0} \mid i = 0, \ldots, n_u + k_u - 1 \right],$$
$$T_v^* := \left[v_0 + L_{\max}^{(v)} \frac{v_j - v_0}{v_{n_v+k_v-1} - v_0} \mid j = 0, \ldots, n_v + k_v - 1 \right],$$
$$D^* := \left[u_0, u_0 + L_{\max}^{(u)} \right] \times \left[v_0, v_0 + L_{\max}^{(v)} \right].$$



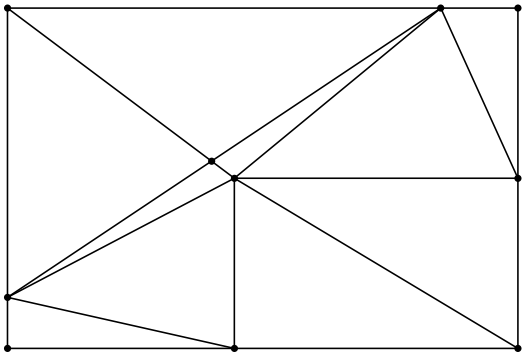
a. Clipping-Bereich



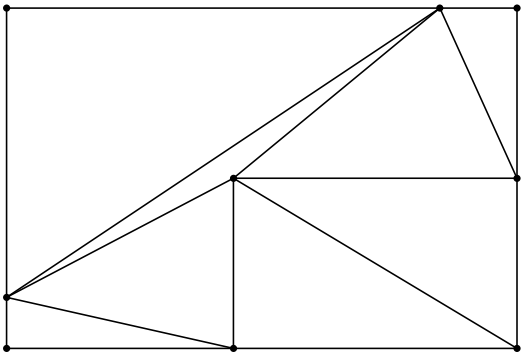
b. Zu triangulierende Punkte



c. Trianguliertes Gebiet ohne Verbindungslinie (gestrichelt)

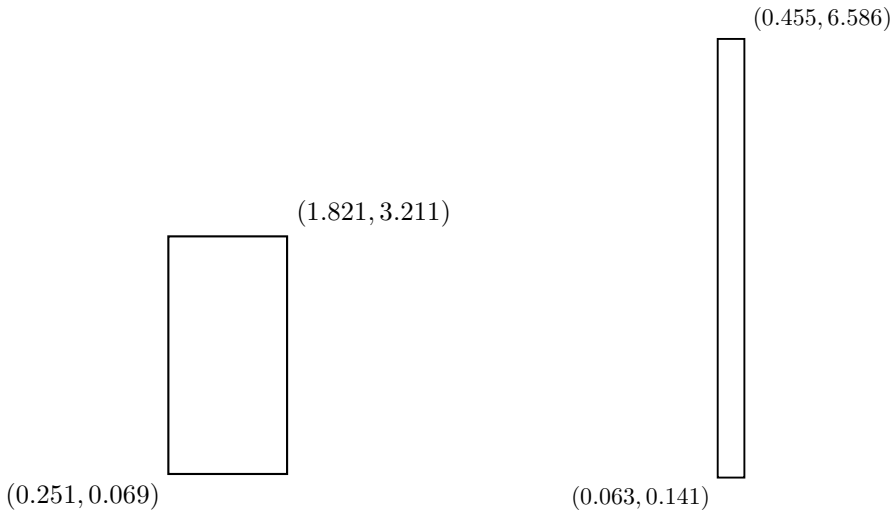


d. Neue Triangulierung (Methode 1)



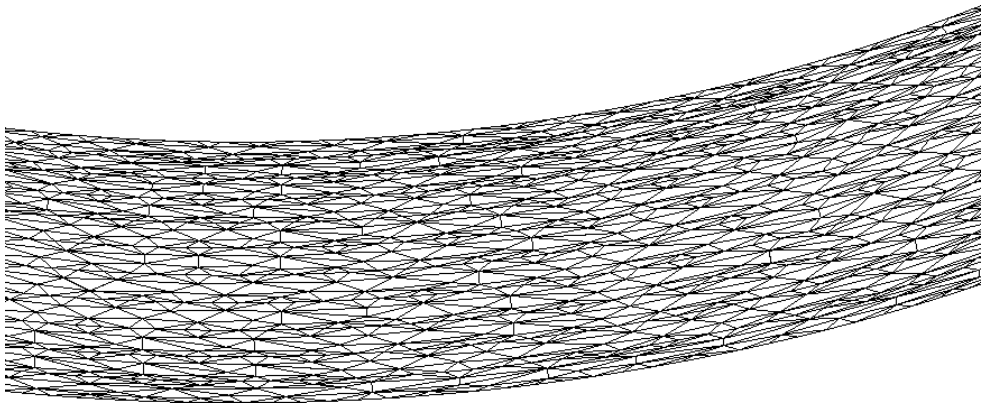
e. Neue Triangulierung (Methode 2)

Abbildung 2.2: Triangulierung eines eingebetteten PSLG (mittleres Rechteck)

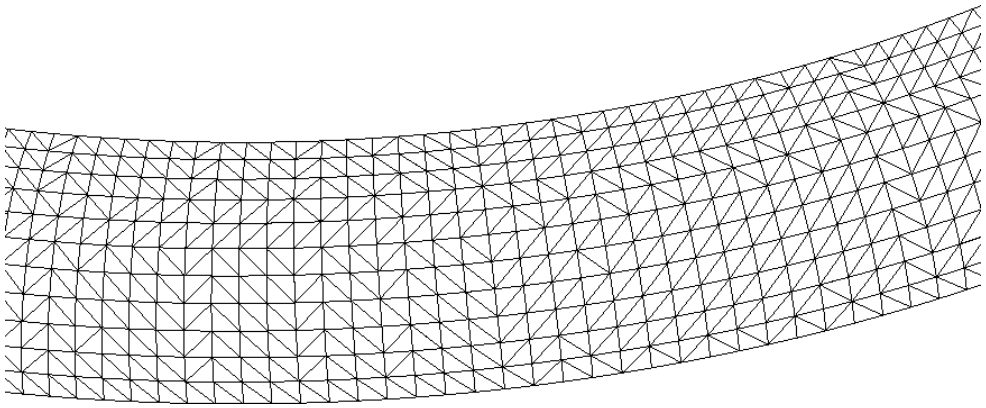


a. Parameterbereich (ohne Reparametrierung)

b. Parameterbereich (mit Reparametrierung)



c. Triangulierung im Bildbereich (ohne Reparametrierung)



d. Triangulierung im Bildbereich (mit Reparametrierung)

Abbildung 2.3: Beispieltriangulierung

Bemerkung 2.3. Wichtig ist, dass man bei der Reparametrierung auch die Kontrollpunkte der Trimmkurven $\mathfrak{B}$ von X im Parameterbereich ändert.

2.2 Fehler bei diskreter Abtastung

Das genaueste Ergebnis mit Net-Compare würde man erreichen, wenn die Triangulierung identisch zu den Splineflächen wäre, was bedeutet, dass man unendlich (oder zumindest sehr) kleine Dreiecke nutzen müsste. Dies würde aber zum einen den Speicherverbrauch in die Höhe schnellen lassen, zum anderen aber auch Net-Compare noch mehr verlangsamen, da der Algorithmus aufgrund der BSP-Tree-Berechnung immens von der Anzahl der Dreiecke abhängt. Aus diesem Grund muss man immer einen kleinen Fehler in Kauf nehmen.

Zur Verdeutlichung, wo der Fehler herkommt, betrachtet man einen Kreis

$$K(t) := r(\cos t, \sin t)^T, \quad t \in [0, 2\pi].$$

Trianguliert man K (im Zweidimensionalen entspricht das einem Linearsatz), so dass zwei Punkte des Linearsatzes maximal l auseinander liegen, so ist der maximale Abstand zwischen Kreis und Liniensegment (genau in der Mitte):

$$p = r - h, \quad h := 0.5\sqrt{4r^2 - l^2}.$$

Dieser Fehler verfälscht das Ergebnis bei der Abstandsberechnung (Abbildung 2.4).

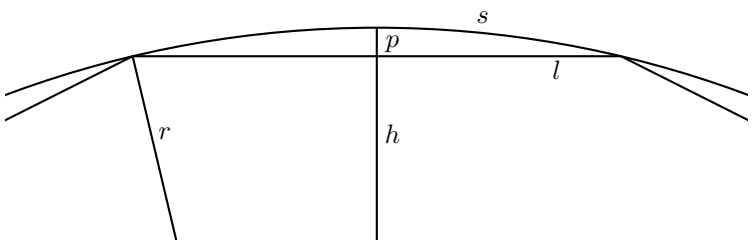
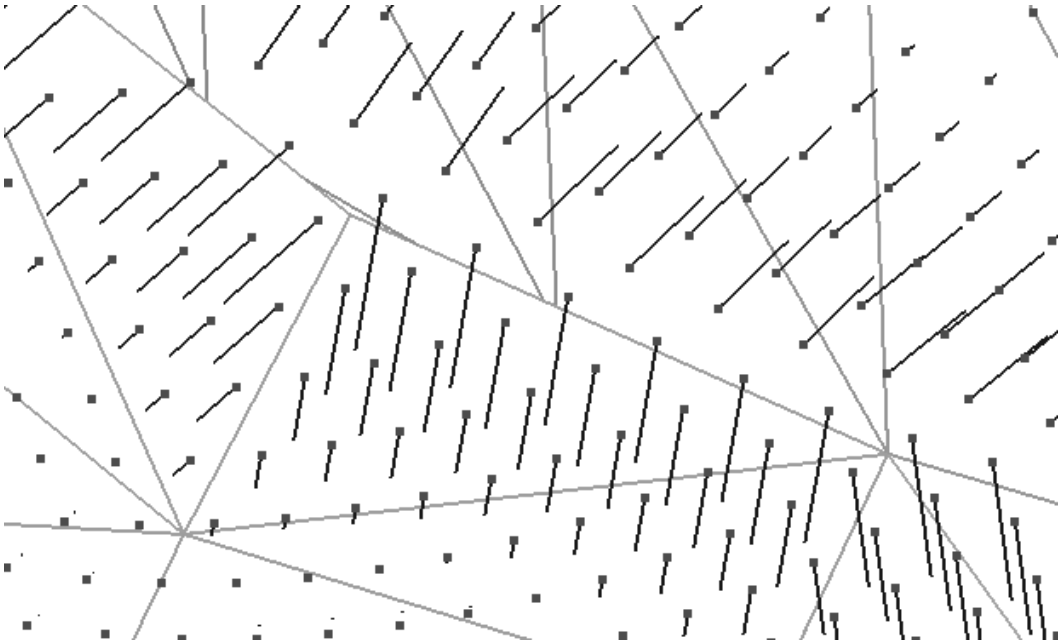


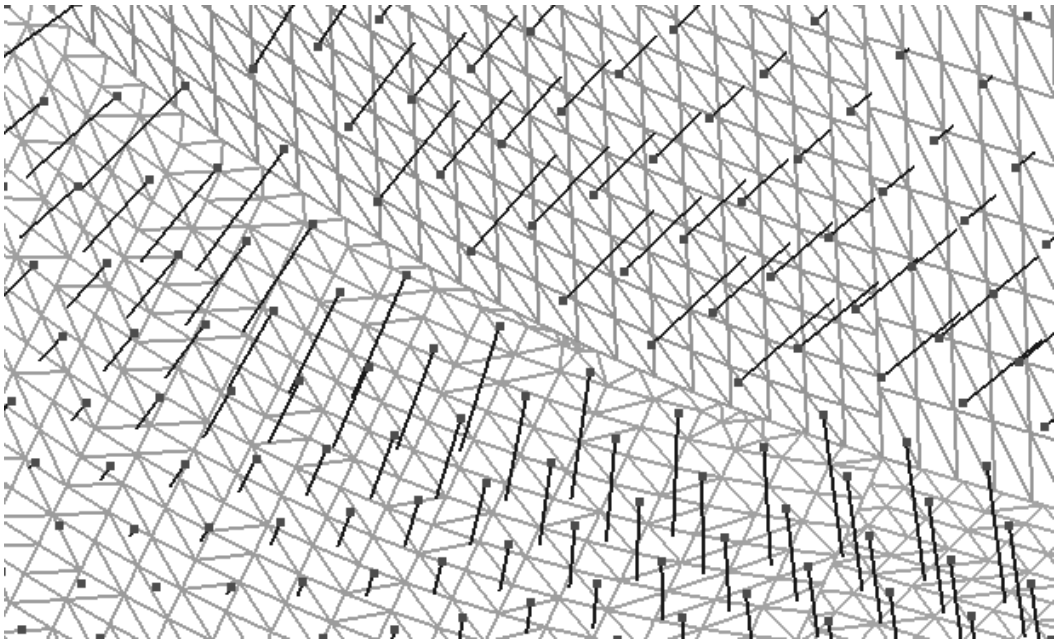
Abbildung 2.4: Linearsatz und Fehler eines Kreises

Um diesen Fehler zu minimieren, beziehungsweise innerhalb einer gewissen Toleranz ϵ halten zu können, muss $p = \epsilon$ gelten, woraus sich der maximale Abstand $l = 2\sqrt{2rp - p}$ und als Segmentumfang $s = 2\arcsin\frac{l}{2r}$ ergibt. Man muss daher $\lceil \frac{2\pi}{s} \rceil$ Segmente auf dem Kreis berechnen, um die gewünschte Genauigkeit von ϵ garantieren zu können.

Die Schwierigkeit bei der Triangulierung ist die Abschätzung, wie genau ein Patch trianguliert werden muss, damit er die eingestellte Fehlergrenze nicht überschreitet. Bei Ebenen und Rotationsflächen ist dies noch einfach, bei allgemeinen Splineflächen aber nicht mehr möglich. Man müsste hier zuerst die maximale Krümmung eines Patches berechnen, um eine obere Grenze zu erhalten, was mit viel Aufwand verbunden wäre. Es ist sinnvoll, die Größe der Segmente beziehungsweise das Raster für eine Triangulierung von vornherein klein genug zu wählen, so dass der Diskretisierungsfehler kaum ins Gewicht fällt. Es kann hierfür aber kein allgemein gültiger Wert angegeben werden.



a) Triangulierung bei 2.5 *mm* Blockgröße



b) Triangulierung bei 0.2 *mm* Blockgröße

Abbildung 2.5: Net-Compare (verschiedene Triangulierungen und Punktezuordnungen)

In Abbildung 2.5 ist zur Veranschaulichung die Triangulierung mit einer Blockgröße von $2.5\,mm$ und mit $0.2\,mm$ zusammen mit der Zuordnung der Punkte bei einer $200\,\mu m$-Fräsung dargestellt. Die Verbindungslinien stellen die Zuordnung eines Punktes zu einer Dreiecksfläche dar. Die Auswirkungen dieser Zuordnung sind in Abbildung 2.6 zu sehen, bei der die berechneten Fehler eingefärbt wurden. Dunkelblau bedeutet einen Fehler von 0 und Rot einen Fehler größer gleich $400\,\mu m$. Wie man sieht, entstehen bei der groben Triangulierung Artefakte und auch falsche Einfärbungen, da die IGES-Flächen nicht genau genug durch Dreiecke dargestellt werden konnten.

2.3 Geschwindigkeitsvergleich

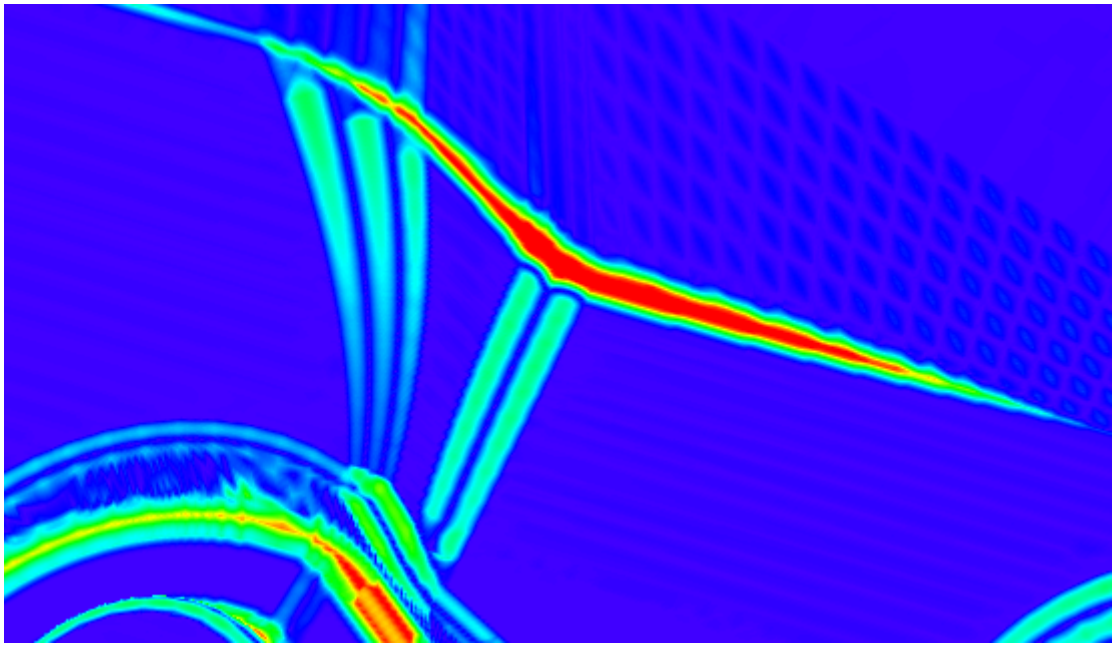
Je mehr Dreiecke die Triangulierung der Vergleichsfläche besitzt, desto langsamer wird Net-Compare. Je weniger Dreiecke es sind, desto ungenauer wird das Ergebnis. Das ist wie oben gesagt der große Nachteil von Net-Compare. Um dies zu verdeutlichen soll das Verfahren mit den Algorithmen dieser Arbeit (*Milling Error Visualization (MEV)* genannt) in einem Beispiel verglichen werden[2]. Hierfür wird das Testwerkstück von DaimlerChrysler benutzt, das mit einer Blockgröße von $0.2\,mm$ trianguliert wurde (siehe Abbildung 2.5b). Zusätzlich findet auch noch ein grober Vergleich mit einer Blockgröße von $1\,mm$ statt. Das Werkstück wird dann mit verschiedenen Punktedichten gefräst und die Zeit beider Algorithmen für die Zuordnung verglichen. Die Dauer für die Extrahierung der IGES-Daten und die Triangulierung wird dabei in beiden Verfahren nicht mit einberechnet.

Punktedichte Gitterabstand	Punkteanzahl	Net-Compare ($0.2\,mm$)	Net-Compare ($1\,mm$)	MEV
$1000\,\mu m$	30.000	2:42 min	0:07 min	0:14 min
$500\,\mu m$	120.392	5:04 min	0:15 min	0:24 min
$200\,\mu m$	752.450	23:49 min	1:12 min	1:19 min
$100\,\mu m$	3.009.800	79:30 min	4:19 min	4:22 min

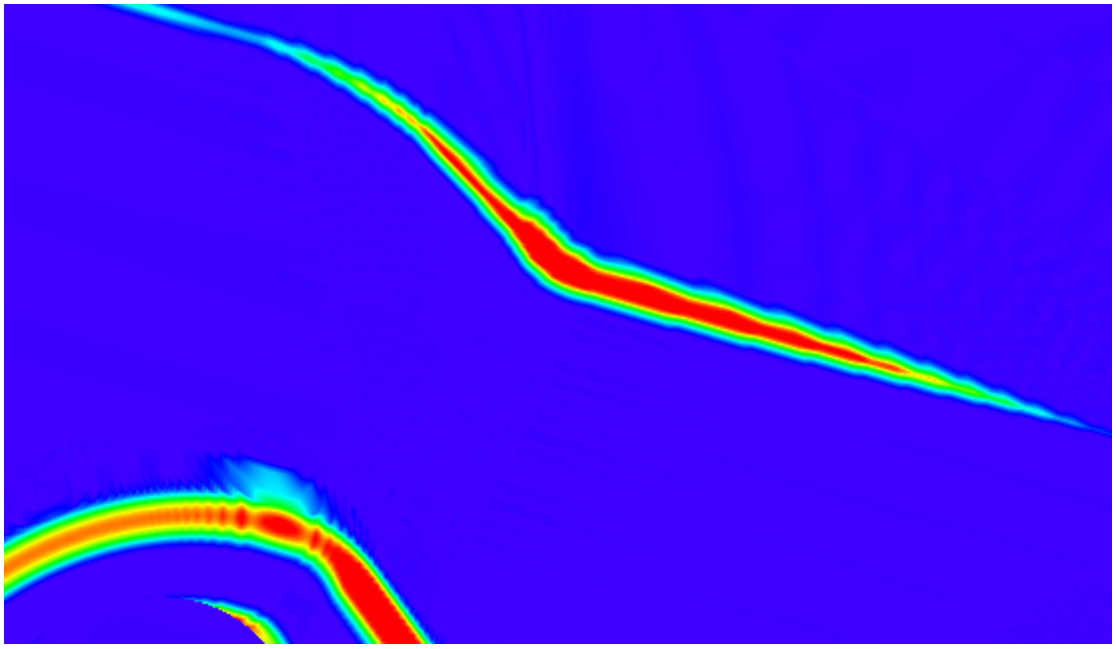
Tabelle 2.1: Vergleich der Geschwindigkeit von Net-Compare und MEV

Wie man in Tabelle 2.1 sieht, ist Net-Compare bei einer Blockgröße von $1\,mm$ in etwa so schnell wie die Ergebnisse aus dieser Arbeit. Der Nachteil ist, dass diese Blockgröße so ungenau ist, dass bei der Darstellung zusätzliche Artefakte entstehen, die das Bild verfälschen (Abbildung 2.7). Bei einer Blockgröße von $0.2\,mm$ ist die Darstellung dagegen korrekt (Abbildung 2.8), benötigt aber um einen Faktor 20 länger für die Verarbeitung. Die Ergebnisse von MEV sind hierzu fast identisch (Abbildung 2.9).

[2]Der Vergleich fand dabei auf einem Laptop mit 1.73 GHz Rechenleistung und 1 GB Arbeitsspeicher statt.



a) Fehlereinfärbung bei 2.5 *mm* Blockgröße



b) Fehlereinfärbung bei 0.2 *mm* Blockgröße

Abbildung 2.6: Net-Compare (Fehlereinfärbung)

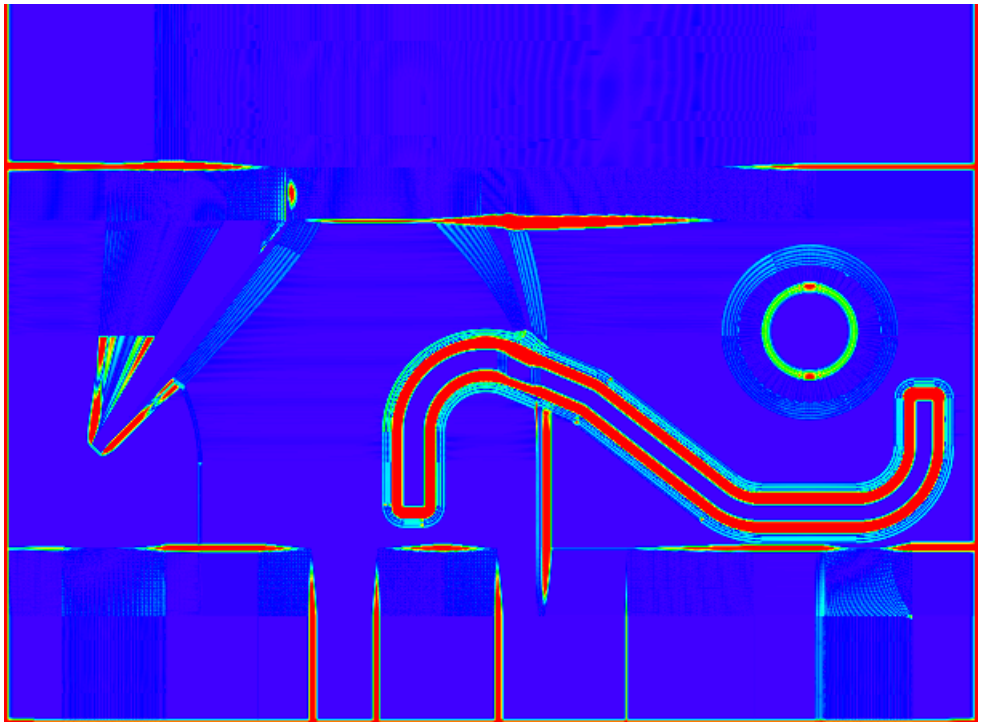


Abbildung 2.7: Fehlerdarstellung mit Net-Compare (Blockgröße $1\,mm$*), max. Fehler* $0.1mm$

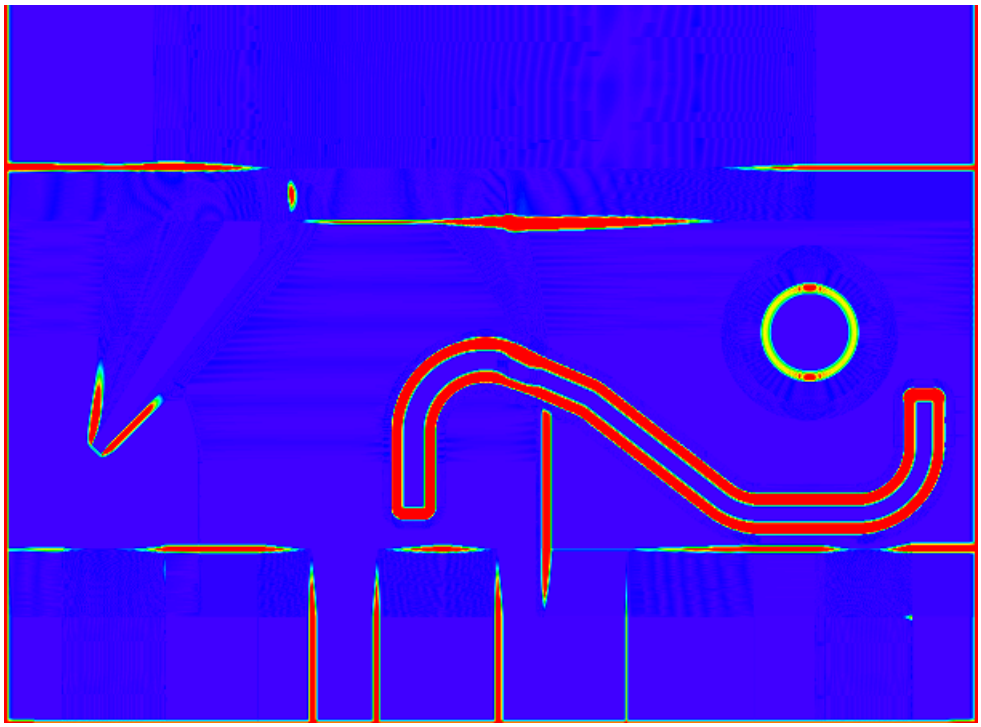


Abbildung 2.8: Fehlerdarstellung mit Net-Compare (Blockgröße $0.2\,mm$*), max. Fehler* $0.1mm$

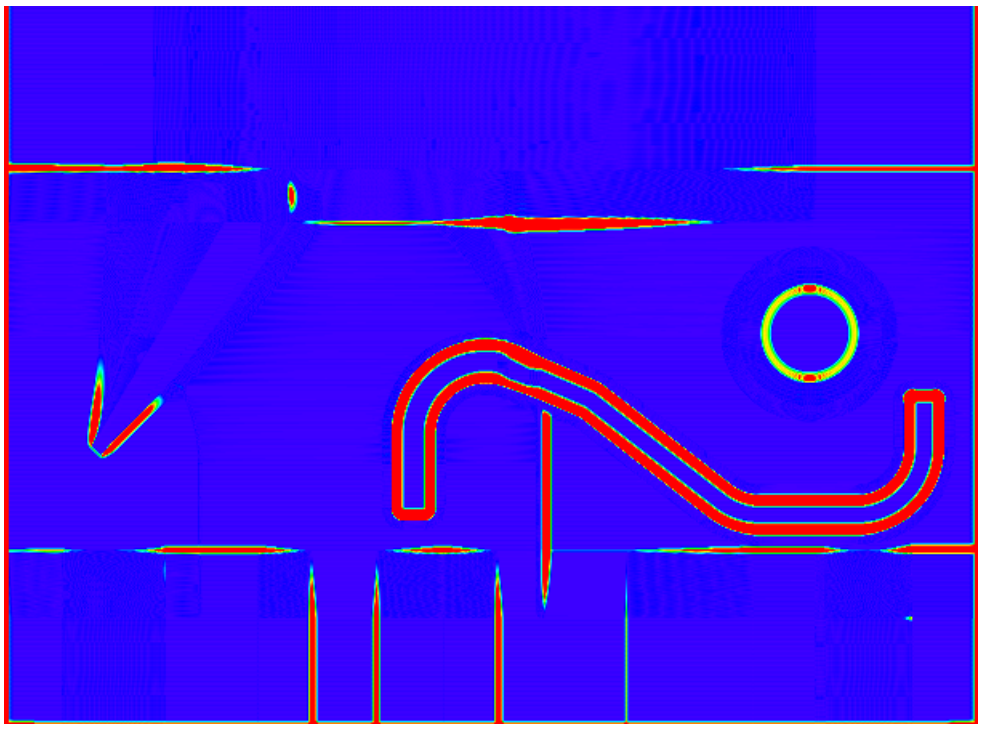


Abbildung 2.9: Fehlerdarstellung mit MEV, max. Fehler 0.1 *mm*

Teil VII
Zusammenfassung

Auf den vorhergehenden Seiten wurden Methoden und Algorithmen vorgestellt, um CAD-Flächen zu extrahieren, in B-Spline-Kurven und B-Spline-Flächen zu konvertieren, diesen dann Punkte aus einem simulierten Fräsprozess zuzuordnen und den dabei berechneten Fehler zwischen Original-Fläche und simuliertem Werkstück darzustellen.

Dabei wurde in Kapitel II zuerst eine kurze Einführung in die Darstellung und Verarbeitung von B-Spline-Kurven und -Flächen gegeben. Vor allem wichtige Verfahren wie die Auswertung und Ableitung, das Knoteneinfügen und die Graderhöhung wurden mathematisch und algorithmisch angegeben. Die Abstandsberechnung über die Newton-Methode bildete die Grundlage für die spätere Orthogonalprojektion der Punkte aus der Punktewolke auf eine Fläche. Zusätzlich wurden so genannte Trimmkurven für B-Spline-Flächen eingeführt, die im computergestützten Design sehr häufig zur Gestaltung benutzt werden.

Teil III beschäftigte sich mit der Beschreibung und Konvertierung einiger, in der IGES-Datei vorliegender, geometrischen Objekte wie Kreisbögen, Regelflächen, Rotationsflächen oder Translationsflächen. Es wurden Verfahren angegeben, wie man diese in B-Spline-Darstellung konvertieren kann. Wichtig für die spätere Zuordnung war die Berechnung der Trimmkurven im Parameterbereich, da diese in der IGES-Datei nicht immer gegeben sind.

Nach dem Einlesen der Punktewolke konnten deren Punkte in Teil IV in einem mehrstufigen Verfahren eindeutig den einzelnen Flächen zugeordnet werden, so dass man die jeweils nächstgelegenen Punkte auf den Flächen samt Abstand erhielt. Das Zuordnungsverfahren war dabei nach Berechnungsaufwand geordnet, so dass zuerst eine einfache Vorselektion der Punkte auf Basis der Bounding Box einer Fläche statt fand. Danach konnte man diese Punkte auf die Normalenebenen der B-Spline-Flächen parallelprojizieren, wobei vor allem bei gedrehten Flächen eine Unterteilung notwendig war, um korrekte Ergebnisse erzielen zu können. Die dritte Zuordnungsstufe benutzte dann ein korrigierendes Newton-Verfahren, um zu jedem Punkt aus der vorselektierten Punktewolke den nächstgelegenen Punkt auf der Fläche und den zugehörigen Parameterwert zu berechnen.

Kapitel V behandelte die Approximation der zugeordneten Punkte durch B-Spline-Flächen, um die mitunter sehr große Datenmenge reduzieren zu können. Hierfür wurde eine Least-Squares-Approximation mit Glätterterm vorgestellt. Viel wichtiger noch war die Approximation von B-Spline-Kurven, die bei der Berechnung der Trimmkurven im Parameterbereich benutzt wurde. Für die Approximation wurden ebenfalls ein Least-Squares-Approximation mit Smoothing-Splines benutzt und zusätzlich die Quasi-Interpolation mit dem Schoenberg-Operator. Beide Approximationsverfahren hatten Vor- und Nachteile, so dass man je nach Verteilung der vorliegenden zu approximierenden 2-D-Punkte entscheiden musste, welchen Algorithmus man benutzt.

Der letzte Teil VI zeigte die Visualisierung des Fräsfehlers im VisuTool, indem der Fehler farblich dargestellt wurde. Neben dem Vergleich der diskreten Punkte war auch ein Vergleich

zwischen der Originalfläche und eines Approximanten möglich. Abschnitt 2 zeigte die Vorteile der Ergebnisse dieser Arbeit, indem die Algorithmen mit der bereits existierenden Lösung Net-Compare verglichen wurden. Es wurde an einem realen Beispiel gezeigt, dass das entwickelte Verfahren bei gleicher Genauigkeit um einen Faktor 20 schneller ist. Die Angabe hängt dabei aber natürlich von der Struktur der gelesenen Daten ab und kann nicht verallgemeinert werden.

Es ergeben sich aber noch weitere Fragen, die im Anschluss dieser Arbeit erörtert werden können. So sind vor allem entartete Tensor-Produkt-Spline-Flächen, bei denen eine Randkurve auf einen Punkt zusammenfällt, numerisch schwer zu handhaben. Die Newton-Methode kann in der Nähe der Entartungen zu falschen Ergebnissen führen, da die Ableitung der Fläche dort fast Null ist. Eine Parameterbestimmung war in diesem Fall nicht möglich und es musste sich mit einer Näherung des Punktes zu den Randkurven begnügt werden. Man sollte diese Flächen in Zukunft näher untersuchen und Methoden entwickeln, die eine genauere Parameterbestimmung ermöglichen.

Teil VIII
Anhang

A Kontrollpunkte und Kreise

Es wird in diesem Abschnitt gezeigt, wie man den mittleren Kontrollpunkt einer zu einem Kreissegment gehörigen rationalen B-Spline-Kurve

$$X := \frac{\sum_{i=0}^{n-1} w_i d_i N_{i,k}(t|T)}{\sum_{i=0}^{n-1} w_i N_{i,k}(t|T)}$$

bestimmt.

Konvention A.1.

1. Es wird davon ausgegangen, dass der Winkel α zwischen Startpunkt V und Endpunkt W des Kreisbogens mit Mittelpunkt 0 kleiner oder gleich dem Trennwinkel ξ ist gemäß Teil III, Abschnitt 1.1.
2. Es sei der Radius $r := \|V\| = \|W\|$ und der Winkel $\beta := \frac{\alpha}{2}$ definiert.

Der Kreisbogen wird durch drei Kontrollpunkte d_0, d_1, d_2 bestimmt, wobei der erste und dritte Kontrollpunkt durch den Start- und Endpunkt gegeben ist:

$$d_0 := V, \quad d_2 := W.$$

Der mittlere Kontrollpunkt ist gesucht beziehungsweise dessen Länge l, da die Gestalt von d_1 durch d_0 und d_2 gegeben ist. Es gilt:

$$d_1 := \frac{l}{\|d_0 + d_2\|}(d_0 + d_2). \tag{A.1}$$

Konvention A.2.

1. Die Knotenfolge ist $T := [0,0,0,1,1,1]$, damit eine B-Spline-Kurve der Ordnung $k := 3$ erzeugt wird, dessen äußere Kontrollpunkte interpoliert werden.
2. Aufgrund der Interpolation erhalten die Gewichte zu den äußeren Kontrollpunkten den Wert 1.0, der innere Punkt die Gewichtung $\cos\beta$ (Farin [10, S. 222]). Es ist also

$$w_0 := 1.0, \quad w_1 := \cos\beta, \quad w_2 := 1.0.$$

Damit kann man den rationalen deBoor-Algorithmus auf X mit $t := 0.5$ anwenden. Bestimmt man die Position r von t in der Knotenfolge, erhält man $t_r \leq t < t_{r+1}$ mit $r = 2$ als Ausgangsposition. Der deBoor-Algorithmus besteht aus zwei Schleifen, wobei die äußere mit i von 1 bis $k-1$ läuft und die innere rückwärts mit j von r bis $r-k+i+1$. In diesem Fall ist also $i = 1,2$ und $j = 2,\ldots,i$. Der Algorithmus läuft wie folgt:

- $i = 1, j = 2$:
  - $c := \frac{t-t_2}{t_4-t_2} = \frac{1}{2}$
  - $w_2' := (1-c)w_1 + cw_2 = \frac{1}{2}(\cos\beta + 1)$
  - $d_2' := \frac{(1-c)w_1d_1 + cw_2d_2}{w_2'} = \frac{1}{\cos\beta+1}(\cos\beta \cdot d_1 + d_2)$
- $i = 1, j = 1$:
  - $c := \frac{1}{2}$
  - $w_1' := (1-c)w_0 + cw_1 = \frac{1}{2}(\cos\beta + 1)$
  - $d_1' := \frac{(1-c)w_0d_0 + cw_1d_1}{w_1'} = \frac{1}{\cos\beta+1}(d_0 + \cos\beta \cdot d_1)$
- $i = 2, j = 2$:
  - $c := \frac{1}{2}$
  - $w_2'' := (1-c)w_1' + cw_2' = \frac{1}{2}(\cos\beta + 1)$
  - $d_2'' := \frac{(1-c)w_1'd_1' + cw_2'd_2'}{w_2'} = \frac{1}{2}(d_1' + d_2') = \frac{1}{2\cos\beta+2}(d_0 + 2\cos\beta \cdot d_1 + d_2)$

Es ist also $X(t) = d_2''$. Der Wert $X(t)$ auf dem Kreisbogen ist aber aufgrund der Symmetrieeigenschaft des Kreissplines explizit bekannt. Es ist

$$X(t) = \frac{r}{\|V+W\|}(V+W) = \frac{r}{\|d_0+d_2\|}(d_0+d_2).$$

Dies führt zu folgender Gleichung:

$$\begin{aligned}
0 &= \frac{1}{2\cos\beta+2}(d_0 + 2\cos\beta \cdot d_1 + d_2) - \frac{r}{\|d_0+d_2\|}(d_0+d_2) \\
&= \frac{1}{2\cos\beta+2}d_0 + \frac{2\cos\beta}{2\cos\beta+2}d_1 + \frac{1}{2\cos\beta+2}d_2 - \frac{r}{\|d_0+d_2\|}(d_0+d_2) \\
&= \frac{1}{2\cos\beta+2}d_0 + \frac{2l\cos\beta}{(2\cos\beta+2)\|d_0+d_2\|}(d_0+d_2) + \frac{1}{2\cos\beta+2}d_2 - \frac{r}{\|d_0+d_2\|}(d_0+d_2) \\
&= \frac{2l\cos\beta}{(2\cos\beta+2)\|d_0+d_2\|}(d_0+d_2) + \frac{1}{2\cos\beta+2}(d_0+d_2) - \frac{r}{\|d_0+d_2\|}(d_0+d_2) \\
&= \left(\frac{2l\cos\beta}{(2\cos\beta+2)\|d_0+d_2\|} + \frac{1}{2\cos\beta+2} - \frac{r}{\|d_0+d_2\|}\right)(d_0+d_2) \\
&= \frac{2l\cos\beta}{(2\cos\beta+2)\|d_0+d_2\|} + \frac{1}{2\cos\beta+2} - \frac{r}{\|d_0+d_2\|}
\end{aligned}$$

Dies stellt man nach l um. Es folgt:

$$\frac{2l\cos\beta}{(2\cos\beta+2)\|d_0+d_2\|} = \frac{r}{\|d_0+d_2\|} - \frac{1}{2\cos\beta+2}$$

$\Rightarrow$

$$2l\cos\beta = r(2\cos\beta+2) - \|d_0+d_2\|$$

$\Rightarrow$

$$\begin{aligned} l &= \frac{r(2\cos\beta+2)}{2\cos\beta} - \frac{\|d_0+d_2\|}{2\cos\beta} \\ &= \frac{r}{\cos\beta} + \left(r - \frac{\|d_0+d_2\|}{2\cos\beta}\right) \end{aligned} \tag{A.2}$$

Das Verhältnis der beiden eingeklammerten Terme in (A.2) wird in Abbildung A.1 verdeutlicht. Es ist offensichtlich

$$2q = \|d_0+d_2\|. \tag{A.3}$$

Außerdem gilt

$$\cos\beta = \frac{q}{r}. \tag{A.4}$$

Löst man (A.4) nach q auf und setzt dies in (A.3) ein, erhält man

$$2r\cos\beta = \|d_0+d_2\|. \tag{A.5}$$

Das stellt man noch nach r um und sieht sofort den Zusammenhang in (A.2):

$$r = \frac{\|d_0+d_2\|}{2\cos\beta} \quad \text{beziehungsweise} \quad r - \frac{\|d_0+d_2\|}{2\cos\beta} = 0.$$

Dadurch hat man also l mit

$$l = \frac{r}{\cos\beta}$$

bestimmen können, woraus man sofort den gesuchten Kontrollpunkt

$$d_1 = \frac{l}{\|d_0+d_2\|}(d_0+d_2) = \frac{l}{\|V+W\|}(V+W)$$

berechnen kann.

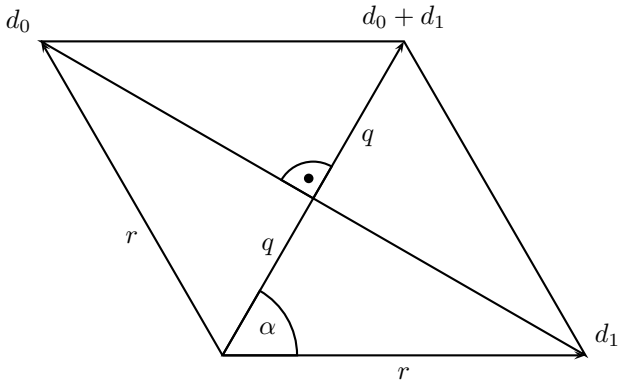


Abbildung A.1: Seitenverhältnis

B Koordinatentransformation

Um ein Koordinatensystem in ein anderes zu überführen, muss man die beiden Normalenvektoren zweier korrespondierender Ebenen richtig transformieren beziehungsweise drehen. Hier geht es speziell darum, eine beliebige Ebene mit Normalenvektor $n = (x, y, z)^T \neq 0$ so zu drehen, dass der gedrehte Normalenvektor n' in Richtung $e_z := (0, 0, 1)^T$ zeigt.

Methode 1

Ein Ansatz hierzu wäre es, mittels zweier Drehmatrizen den Normalenvektor n erst in eine Ebene, beispielsweise in die x/z-Ebene, und dann auf die y-Achse zu drehen. Dies geht aber auch einfacher, wenn man um die Achse dreht, in die der zu n und e_z senkrechte Vektor zeigt. Abbildung B.1 verdeutlicht, wie die Drehung von statten geht.

Man definiert

$$\widehat{n} := \frac{n}{l}, \quad a := \frac{\widehat{n} \times e_z}{\|\widehat{n} \times e_z\|} = \begin{pmatrix} \frac{y}{d} \\ \frac{-x}{d} \\ 0 \end{pmatrix}, \quad \alpha := \angle(\widehat{n}, e_z) = \arccos\left(\frac{z}{l}\right), \tag{B.6}$$

$$d := \sqrt{x^2 + y^2}, \quad l := \sqrt{x^2 + y^2 + z^2}.$$

Ist $a = 0$, wären n und e_z linear abhängig und die Ebene ist bereit korrekt gedreht. Es ist hierbei unerheblich, ob $n = e_z$ oder $n = -e_z$, da man die Ebene nur dreht, um auf sie parallel zu projizieren.

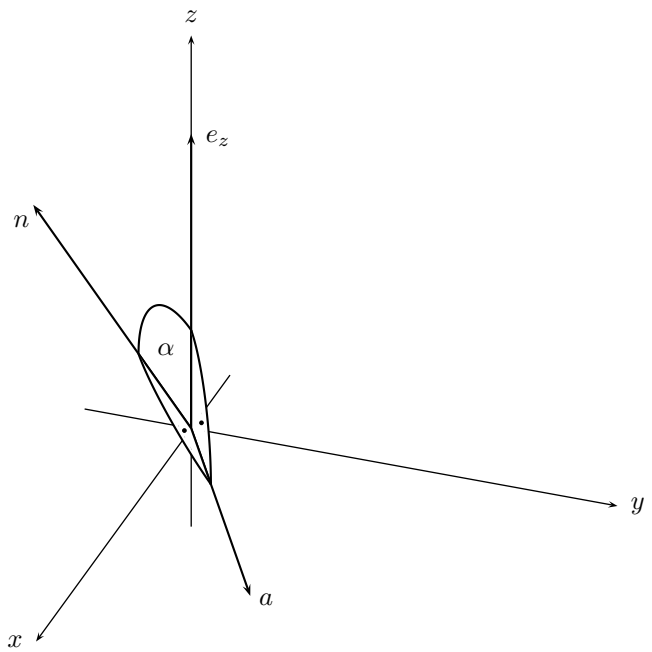


Abbildung B.1: Drehung von n

Definition B.1. *Es sei eine* Rotationsachse $a := (x, y, z)^T$, $\|a\| = 1$, *und ein Winkel* α *gegeben. Dann definiert man die* Rotationsmatrix

$$R_{a,\alpha} := \begin{pmatrix} (1-\cos\alpha)x^2 + \cos\alpha & (1-\cos\alpha)xy - \sin\alpha \cdot z & (1-\cos\alpha)xz + \sin\alpha \cdot y \\ (1-\cos\alpha)xy + \sin\alpha \cdot z & (1-\cos\alpha)y^2 + \cos\alpha & (1-\cos\alpha)yz - \sin\alpha \cdot x \\ (1-\cos\alpha)xz - \sin\alpha \cdot y & (1-\cos\alpha)yz + \sin\alpha \cdot x & (1-\cos\alpha)z^2 + \cos\alpha \end{pmatrix}. \tag{B.7}$$

Lemma B.2. *Es seien* $n = (x, y, z)^T \neq 0$ *und* $e_z := (0, 0, 1)^T$ *gegeben. Dann gilt*

$$(R_{a,\alpha} n, e_z) = 0,$$

wobei a *und* α *wie in (B.6) definiert sind.*

Beweis: Setzt man a und α in (B.7) ein, erhält man

$$R_{a,\alpha} := \begin{pmatrix} \frac{(1-z/l)y^2}{d^2} + \frac{z}{l} & \frac{-(1-z/l)xy}{d^2} & \frac{-x}{l} \\ \frac{-(1-z/l)xy}{d^2} & \frac{(1-z/l)x^2}{d^2} + \frac{z}{l} & \frac{-y}{l} \\ \frac{x}{l} & \frac{y}{l} & \frac{z}{l} \end{pmatrix}.$$

Daraus folgt

$$R_{a,\alpha} n = \begin{pmatrix} \frac{(1-z/l)xy^2}{d^2} + \frac{xz}{l} - \frac{(1-z/l)xy^2}{d^2} + \frac{-xz}{l} \\ -\frac{(1-z/l)x^2y}{d^2} + \frac{(1-z/l)x^2y}{d^2} + \frac{zy}{l} - \frac{yz}{l} \\ \frac{x^2}{l} + \frac{y^2}{l} + \frac{z^2}{l} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ l \end{pmatrix}$$

und folglich ist $(R_{a,\alpha} n, e_z) = 0$. □

Methode 2

Als zweite Möglichkeit hilft die einfache Tatsache, dass man die Koordinatentransformation über die Einheitsvektoren des lokalen Systems vornehmen kann. Sind also lokale Einheitsvektoren $\widehat{e}_x, \widehat{e}_y, \widehat{e}_z$ gegeben, kann man über die Rotationsmatrix

$$R := \begin{pmatrix} \widehat{e}_x & \widehat{e}_y & \widehat{e}_z \end{pmatrix} \in \mathbb{R}^{3\times 3} \tag{B.8}$$

einen Vektor $v \in \mathbb{R}^3$ in das globale Koordinatensystem transformieren. Die lokalen Einheitsvektoren ergeben sich als

$$\widehat{e}_z := \frac{n}{\|n\|}, \quad \widehat{e}_x := \begin{pmatrix} \sin\alpha \\ \cos\alpha \\ 0 \end{pmatrix}, \quad \widehat{e}_y := \widehat{e}_z \times \widehat{e}_x, \quad \alpha := \arctan\left(\frac{2y}{x}\right) - \frac{\pi}{2}.$$

Ist $x = 0$, setzt man $\alpha := 0$.

C Notation

Liste der in dieser Arbeit benutzten Notation.

$(\cdot, \cdot)$	Skalarprodukt zweier Vektoren
λ	Entartungsparameter, aber auch Glättefaktor
$\mathfrak{A}$	Normalenebene einer B-Spline-Fläche
$\mathfrak{B}$	Trimmkurve im Parameterbereich
$\mathfrak{C}$	Trimmkurve im Bildbereich
$\mathfrak{D}$	Menge der Kontrollpunkte einer B-Spline-Fläche
$\mathfrak{M}$	von einer Kurve umschlossenes Gebiet
$\mathfrak{M}(\mathfrak{C})$	sichtbarer Bereich einer B-Spline-Fläche
$\mathfrak{n}$	Normale der Normalenebene $\mathfrak{A}$
$\mathfrak{N}(u, v)$	Flächennormale an der Stelle (u, v)
$\mathfrak{o}$	Stützvektor der Normalenebene $\mathfrak{A}$
$\mathfrak{P}$	Punktewolke einer Frässimulation
$\mathfrak{R}$	zusammengesetzte, geschlossene Randkurve einer B-Spline-Fläche
∇X	erste Ableitung einer B-Spline-Fläche (Gradient)
$\nu(X, P)$	Verdrehungsgrad einer B-Spline-Fläche
∂D	Rand einer B-Spline-Fläche im Parameterbereich
ψ	Entartungsgrad
ρ_i	Bewertungsfunktion für BSP-Trees
Θ	Zuordnung der Punktewolke
$\times$	Kreuzprodukt zweier Vektoren
ϑ	Zuordnung eines Punktes
$\widehat{\psi}$	Entartungswert eines Parameters
$\widehat{s}_X$	Umkehrfunktion der Bogenlänge für Kreissplines
B	Bounding Box
$B_{\max}$	Maximum der Bounding Box

$B_{\min}$	Minimum der Bounding Box
$B_{i,n}$	Bernstein-Polynom vom Grad n
D	Parametergebiet einer B-Spline-Fläche
d_i	Kontrollpunkte für B-Spline-Kurven
$d_{i,j}$	Kontrollpunkte einer B-Spline-Fläche
G	Polygon
H_X	zweite Ableitung einer B-Spline-Fläche (Hessematrix)
k	Ordnung einer B-Spline-Kurve (entspricht Grad+1)
k_u, k_v	Ordnungen einer B-Spline-Fläche
L	Polygonale Region
$L_i^{(u)}, L_j^{(v)}$	Länge der Kontrollpolygone einer B-Spline-Fläche in eine Richtung
n	Anzahl der Kontrollpunkte einer B-Spline-Kurve
n_u, n_v	Anzahl der Kontrollpunkte einer B-Spline-Fläche
$N_{i,k}$	B-Spline-Funktion der Ordnung k
$R_0, \ldots, R_3$	einzelne Randkurven einer B-Spline-Fläche
$R_{a,\alpha}$	Rotationsmatrix um Achse a mit Winkel α
S_f^*	Approximierte Umkehrabbildung der Sehnenlänge von f
S_f	Sehnenlänge von f
s_f	Bogenlänge von f
S_f^{-1}	Umkehrabbildung der Sehnenlänge von f
s_f^{-1}	Umkehrabbildung der Bogenlänge von f
T	Knotenvektor einer B-Spline-Kurve
t_i	Knoten einer B-Spline-Kurve
T_u, T_v	Knotenvektoren einer B-Spline-Fläche
u_i, v_i	Knoten einer B-Spline-Fläche
w_i	Gewichte einer B-Spline-Kurve
$w_{i,j}$	Gewichte einer B-Spline-Fläche
X	B-Spline-Kurve oder B-Spline-Fläche

Literatur

[1] Abramowitz, M., Stegun, I.A. (eds.): *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Table*, (Dover, 1972)

[2] Brouwer, L.E.J.: Beweis des Jordanschen Kurvensatzes, in: *Mathematische Annalen 71*, (1912), Seiten 169-175

[3] Casciola, G., Morigi, S. : Reparametrization of NURBS Curves, in: *International Journal of Shape Modeling Vol. 2, No. 2&3*, (1996), Seiten 103-116

[4] Davis, P. J., Rabinowitz, P.: *Methods of Numerical Integration*, (Academic Press, 1975)

[5] de Berg, M., de Kreveld, M., Overmars, M.: *Computational Geometry. Algorithms and Applications* (Second Edition), (Springer, 2000)

[6] de Boor, C.: The quasi-interpolant as a tool in elementary polynomial spline theory, in: *Approximation Theory*, (Academic Press, 1973), Seiten 269-276

[7] Dierckx, O.: *Curve and Surface Fitting with Splines*, (Oxford, 1993)

[8] Dietz, U.: *Geometrie-Rekonstruktion aus Meßpunktewolken mit glatten B-Spline-Flächen*, (Shaker, 1998)

[9] Evans, F., Skiena, S. S., Varshney, A.: Optimizing Triangle Strips for Fast Rendering, in: *IEEE Visualization '96*, Yagel, R., Nielson, G. M. (eds.), (1996), Seiten 319-326

[10] Farin, G. E.: *Curves and Surfaces for Computer Aided Geometric Design* (5. Auflage), (Academic Press, 2002)

[11] Farin, G.: Rational Quadratic Circles are Chord Length Parametrized, in: *Computer Aided Geometric Design 23(9)*, (2006), Seiten 722-724

[12] Finley, D. R.: *Point-In-Polygon Algorithm - Determining whether a Point Is Inside a Complex Polygon*, (1998), `http://alienryderflex.com/polygon/` (letzter Aufruf: 03.03.2008)

[13] Floater, M. S.: Meshless Parameterization and B-spline Surface Approximation, in: *The Mathematics of Surfaces IX*, Cipolla, R., Martin, R. (eds.), (Springer, 2000), Seiten 1-18

[14] Floater, M. S., Hormann, K.: Surface Parameterization: a Tutorial and Survey, in: *Advances in Multiresolution for Geometric Modelling*, Dodgson, N. A., Floater, M. S., Sabin M. A. (eds.), (Springer, 2005), Seiten 157-186

[15] Floater, M. S., Rasmussen, A. F., Point-based methods for estimating the length of a parametric curve, in: *Journal of Computational and Applied Mathematics Vol. 196, No. 2*, (2006), Seiten 512 - 522

[16] Fuchs, H., Kedem, Z., Naylor, B. : On Visible Surface Generation by A Priori Tree Structure, in: *ACM Computer Graphics* 14(3), (1980), Seiten 124-133

[17] Golub, G. H., Van Loan, C. F.: *Matrix Computations* (Third Edition), (Johns Hopkins University Press, 1996)

[18] Heuser, H.: *Lehrbuch der Analysis, Teil 1* (9. Auflage), (B. G. Teubner Stuttgart, 1990)

[19] Heuser, H.: *Lehrbuch der Analysis, Teil 2* (7. Auflage), (B. G. Teubner Stuttgart, 1991)

[20] Haines, E.: Point in Polygon Strategies, in: *Graphics Gems IV*, Heckbert, P. (ed.), (Academic Press, 1994), Seiten 24-46

[21] Hjelle, O., Daehlen, M.: *Triangulations and Applications*, (Springer, 2006)

[22] Hoschek, J., Lasser, D.: *Grundlagen der geometrischen Datenverarbeitung* (2. Auflage), (B. G. Teubner Stuttgart, 1992)

[23] *IGES - An American National Standard*, (Initial Graphics Exchange Specification 5.3), (IGES/PDES Organization, 1996), `http://www.uspro.org/documents/` (letzter Aufruf: 03.03.2008)

[24] Jerard, R. B., Drysdale, R. L.: Methods for Geometric Modeling, Simulation and Spatial Verification of NC Machining Programs, in: *Product Modelling for Computer-Aided Design and Manufacturing*, (North-Holland Publishing, 1991)

[25] Maehara, R.: The Jordan Curve Theorem Via the Brouwer Fixed Point Theorem, in: *The American Mathematical Monthly Vol. 91, No. 10*, (1984), Seiten 641-643

[26] Nocedal, J., Wright, S. J.: *Numerical Optimization* (Second Edition), (Springer, 2006)

[27] Piegl, L., Tiller, W.: *The NURBS Book* (2. Edition), (Springer, 1997)

[28] Sauer, T.: *Splinekurven und -flächen im CAGD* (Vorlesungsskript), (Justus-Liebig-Universität Gießen, 2007), `http://www.uni-giessen.de/tomas.sauer/` (letzter Aufruf: 03.03.2008)

[29] Shewchuk, J. R.: Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator, in: *Applied Computational Geometry: Towards Geometric Engineering*, Lin, M. C., Manocha, D. (eds.), (Springer, 1996), Seiten 203-222

[30] Shewchuk, J. R.: *Triangle – A Two-Dimensional Quality Mesh Generator and Delaunay Triangulator*, `http://www.cs.cmu.edu/~quake/triangle.html` (letzter Aufruf: 03.03.2008)

[31] Schwanecke, U., Kobbelt, L.: Approximate envelope reconstruction for moving solids, in: *Mathematical Methods for Curves and Surfaces* (Oslo 2000), Lyche, T., Schumaker, L. L. (eds.), (Vanderbuilt University Press, 2001), Seiten 455-466

[32] Seong, J.-K., Elber, G., Kim, M.-S.: Trimming local and global self-intersections in offset curves/surfaces using distance maps, in: *Computer-Aided Design, Vol. 39, Issue 3*, Goodman, J.E., O'Rourke, J. (eds.), (2006), Seiten 183-193

[33] Snoeyink, J.: Point Location, in: *Handbooks of Discrete and Computational Geometry* (Second Edition), Goodman, J.E., O'Rourke, J. (eds.), (Chapman & Hall / CRC, 2004) Seiten 767-786

[34] Stimming, C.: *LAPACK++*, `http://lapackpp.sourceforge.net` (letzter Aufruf: 03.03.2008)

[35] Stoer, J., Bulirsch, R.: *Numerische Mathematik 1* (10. Auflage), (Springer, 2007)

[36] Veblen, O.: Theory of plane curves in nonmetrical analysis situs, in: *Transactions of the American Mathematical Society 6*, (1905), Seiten 83-98

[37] Walter, M., Fournier, A.: Approximate Arc Length Parametrization, in: *Proc. Brazilian Symposium on Computer Graphics and Image Processing*, (1996), Seiten 143-150

Erklärung

Ich habe die vorgelegte Dissertation selbständig und ohne unerlaubte fremde Hilfe und nur mit den Hilfen angefertigt, die ich in der Dissertation angegeben habe.
Alle Textstellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen sind, und alle Angaben, die auf mündlichen Auskünften beruhen, sind als solche kenntlich gemacht.
Bei den von mir durchgeführten und in der Dissertation erwähnten Untersuchungen habe ich die Grundsätze guter wissenschaftlicher Praxis, wie sie in der „Satzung der JustusLiebig-Universität Gießen zur Sicherung guter wissenschaftlicher Praxis“ niedergelegt sind, eingehalten.

Erlangen, April 2008