Masterarbeit

Extraktion von Geometriedaten aus CAD/CAM Systemen

Univeristät Passau / Siemens AG

Florian Lorenz

2. Mai 2014

Prof. Dr. Tomas Sauer
Prof. Dr. Brigitte Forster-Heinlein

Danksagung

Ich bedanke mich herzlich bei Herrn Dr. Carsten Hamm für die beispiellose Betreuung dieser Masterarbeit seitens der Siemens AG. Seine Vorschläge und Anregungen haben einen großen Teil zu meinem Verständnis über CAGD-Systeme beigetragen und einen wichtigen Beitrag für diese Arbeit geliefert. Darüber hinaus bedanke ich mich bei Herrn Prof. Dr. Tomas Sauer und Herrn Jörg Handeck für die Unterstützung im Bereich Mathematik und für Ihre Ideen, die großen Einfluss auf die Entwicklung dieser Abschlussarbeit hatten.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Die Prozesskette eines industriell gefertigten Werkstücks beginnt mit der Konstruktion einer Geometrie mithilfe eines CAD/CAM-Systems[1]) am Computer. Das CAM-System erzeugt daraus eine sogenannte Werkzeugbahn, die im Postprozessor in ein NC-Programm in Form von G-Code übersetzt wird. Dieser G-Code dient nun als Eingabe für verschiedene Steuerungen (z.B. CNC Sinumerik), die die Ansteuerung der Antriebe einer Werkzeugmaschine regeln. In diesem Gesamtprozess können an verschiedenen Stellen Probleme auftreten, die dazu führen, dass ungenauer G-Code an die Maschine übertragen wird und letztendlich das zu fertigende Bauteil nicht den gewünschten Qualitätsanforderungen entspricht. Das Fehlerspektrum reicht hier von ungenauen Konstruktionen im CAD-System bis hin zu numerischen Rundungsfehlern in diversen Steuerungen, die zur Bahnerzeugung benötigt werden.

Diese Masterarbeit ist Teil eines Gesamtprozesses zur Optimierung und Verbesserung von Fräsbahnen, die aus dem CAD-System NX entstehen. In Abbildung 1.1 ist der Zusammenhang dieser Masterarbeit mit dem Gesamtprojekt dargestellt.

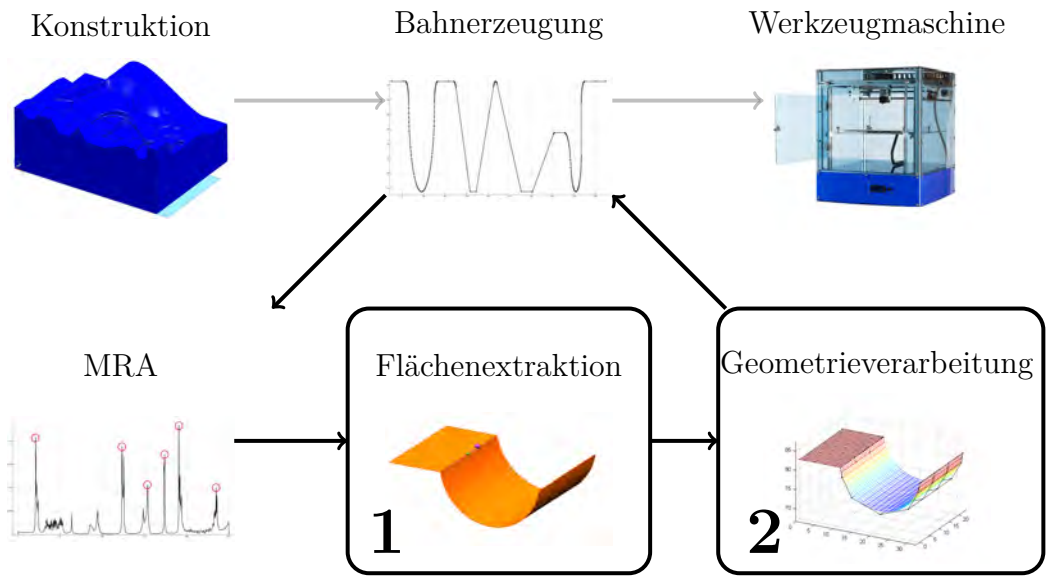


Abbildung 1.1: Prozess zur Optimierung von Fräsbahnen

[1]Computer Aided Design/Computer Aided Manufacturing

Nach der Konstruktion des Werkstückes in NX werden mithilfe eines Postprozessors Fräsbahnen, die in Form von G-Code vorliegen, erzeugt. An dieser Stelle können durch die Anwendung einer Multi-Resolution-Analysis (MRA, [4], [10]) für die Bearbeitung kritische Stellen im G-Code entdeckt werden. Zu diesem Zeitpunkt liegen jedoch keinerlei Informationen des CAD-Systems NX mehr vor. Der Hauptteil dieser Arbeit besteht darin, mithilfe der Schnittstelle NXOpen den Rückschritt in das CAD-System NX zu ermöglichen, um geometrische Informationen zu extrahieren (Teil 1) und diese für weitere Optimierungszwecke aufzuarbeiten (Teil 2).

In Abbildung 1.2 ist ein Problem mit den damit verbundenen Konsequenzen abgebildet. Die linke Grafik stellt das gewünschte Erzeugnis der Fräsmaschine dar. Die Fräsbahn (orange) in der Mitte wurde an der Knickstelle unzureichend abgetastet, was dazu führt, dass die Werkzeugmaschine die Ecke nicht wie gewünscht ausfährt und damit ein qualitativ mangelhaftes Ergebnis produziert. Da die MRA bereits im Stande ist einen Hinweis zu geben, dass es bei dieser Stelle zu Problemen kommen kann, ist der nächste Schritt, sich im CAD-System NX zu vergewissern, um welche Art von Feature es sich handelt und anschließend in weiterführenden Arbeiten die Fräsbahn entsprechend anzupassen und zu optimieren.

(a) Konstruktion (b) Fräsbahn (c) Ergebnis

Abbildung 1.2: Auswirkungen einer problematischen Fräsbahnberechnung

Dieser Arbeit liegt die Annahme zugrunde, dass es sich bei diesen potentiell problematischen Stellen um Flächenübergänge handelt, da diese Situation in der Praxis die häufigste Fehlerquelle darstellt.
Im Zuge dieser Master-Arbeit entstanden mehrere in C# geschriebenen Applikationen, die alle über die Schnittstelle NXOpen mit dem CAD/CAM-System NX kommunizieren. Diese *dll*-Dateien werden in NX gestartet und bieten ein graphisches User-Interface zur interaktiven Bedienung. Darüber hinaus hat der Benutzer die Möglichkeit, ausgewählte Flächen in MATLAB zu exportieren. Dies hat sich in der Praxis als sehr nützlich erwiesen, da die Flächen in MATLAB analytisch wesentlich detaillierter dargestellt und bearbeitet werden können als in NX.

2 NX/NXOpen - Informationsanalyse

Dieses Kapitel erarbeitet Schritt für Schritt die Extraktion der Geometrieeigenschaften von Oberflächen aus dem CAM-System NX. Da die hierfür verwendete Schnittstelle NXOpen ein sehr breites Spektrum an Funktionen bietet, um Geometrie-Informationen aus NX zu extrahieren und diese anschließend weiterzuverarbeiten, ist es unabdingbar, zu Beginn die fundamentalen Flächenbausteine, sowie die interne Datenverarbeitung von NX zu untersuchen und aufzulisten.

2.1 Das CAM-System NX und die Schnittstelle NXOpen

In dieser Masterarbeit wurde das CAM-System NX (Version 8.5) verwendet. Es besteht aus einer graphischen Benutzeroberfläche, die es dem Benutzer ermöglicht, Werkstücke zu konstruieren und zu manipulieren und dem darunterliegenden Kern, der notwendige Berechnungen (Datenverarbeitung, Erzeugung von Fräsbahnen, etc.) ausführt. Über das Interface *NXOpen* ist es möglich auf Funktionalitäten sowohl des Kerns als auch der graphischen Oberfläche zuzugreifen. Hierzu gehören zum Beispiel Funktionen zum Einfügen von dreidimensionalen Objekten, das Manipulieren bereits vorhandener Oberflächen und das Öffnen von Dialogen in der GUI, um eine Interaktion mit dem Benutzer zu ermöglichen. Die Programme, die dieses Interface nutzen, sind in C# implementierte *dll*-Dateien, die aus NX geladen werden können. In Abbildung 2.1 ist der schematische Zusammenhang der verschiedenen Komponenten dargestellt.

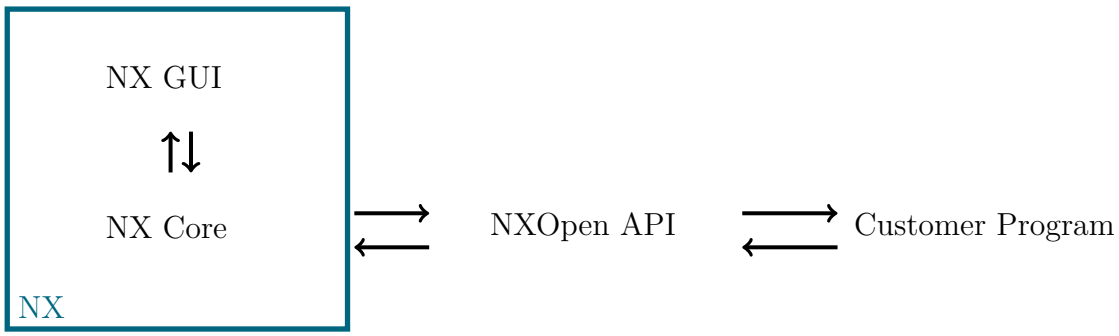


Abbildung 2.1: Einordnung der Schnittstelle NXOpen

Der Benutzer startet aus der NX GUI das C#-Programm, das wiederum mithilfe von NXOpen auf den Kern und die graphische Oberfläche zugreifen kann. Damit verfügt der Anwender über ein technisches Hilfsmittel, das Informationen aus NX extrahieren, weiterverarbeiten und visualisieren kann.

2.2 NX-Oberflächentypen

Die erste interessante Frage, die sich bei der Analyse des Werkstücks stellt ist, welche Beschaffenheit die zu untersuchenden Oberflächen aufweisen. Erste Aufschlüsse geben hier die in NX intern verwendeten 10 unterschiedlichen Oberflächentypen:

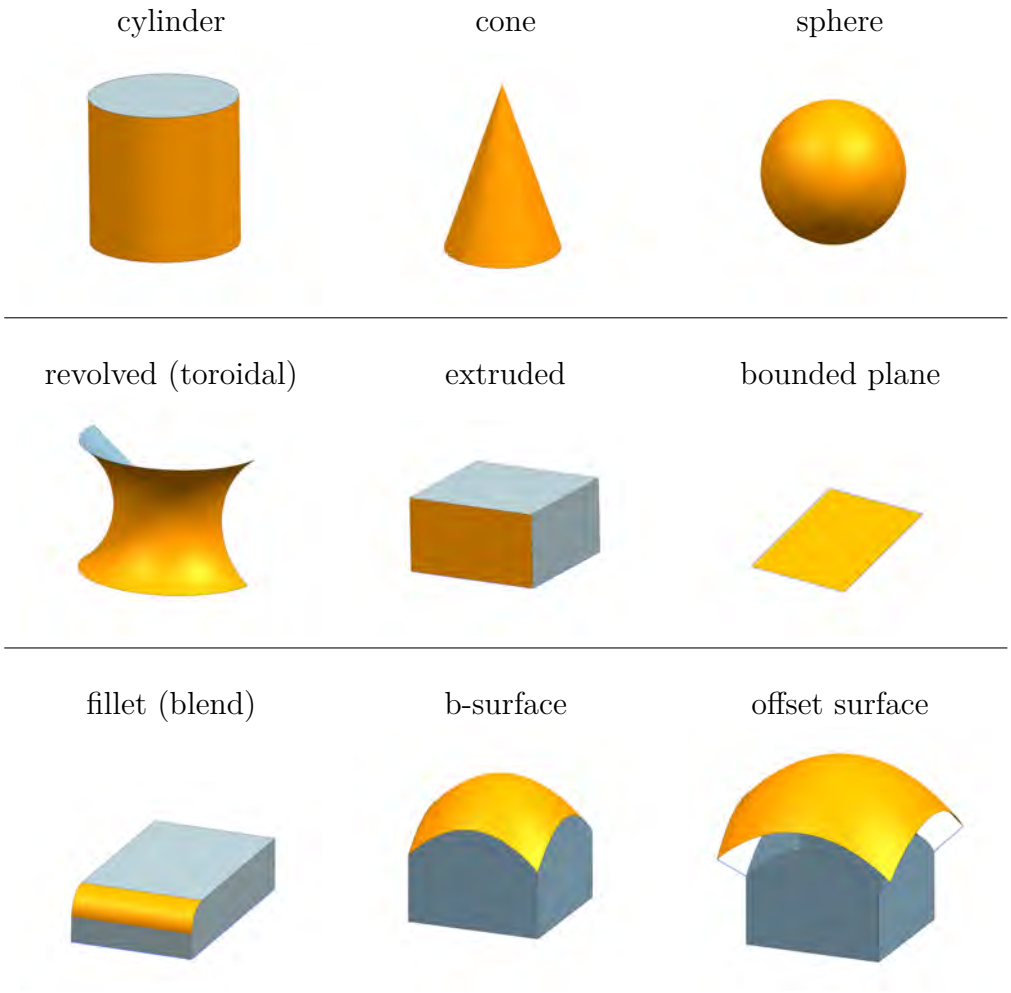


Abbildung 2.2: Oberflächentypen in NX

Der zehnte Flächentyp *foreign surface* taucht lediglich bei speziellen Imports aus anderen Formaten auf und wird im Rahmen dieser Arbeit nicht weiter untersucht.

Der Zugriff auf diese Informationen funktioniert über den Aufruf von

UF_ MODL_ ask_ face_ data

```
int UF_MODL_ask_face_data
(
tag_t    face       Input: ID der Oberflaeche
int*     type       Output: Oberflaechentyp
double   point[]    Output: Punktinformation der Flaeche
double   dir[]      Output: Richtungsinformation der Flaeche
double   box[]      Output: Umschliessende Box (NICHT minimal)
double*  radius     Output: Hauptradius
double*  rad_data   Output: Nebenradius
int*     norm_dir   Output: Richtung der Normalen
)
```

face (*tag_ t*):

Eindeutiger Identifier der Oberfläche (NX intern abgespeichert).

type (*int*):

Gibt an, um welchen Oberflächentyp es sich handelt.

point (*double[3]*):

Je nach Oberflächentyp beschreibt dieser Rückgabewert eine Position auf der Ebene (bounded plane), die Position im Zentrum der Fläche (sphere) oder eine Position auf der (Dreh-)Achse (revolved, cylinder, cone).

dir (*double[3]*):

Beschreibt die Richtung der Drehachse, bzw. die Richtung der Normalen.

box (*double[6]*):

Dieses Array besteht aus zwei gegenüberliegenden Punkten, die einen Quader definieren, der die gesamte Fläche umschließt. Diese Box ist **nicht** minimal und dient somit lediglich als Indikator der Flächenausdehnung.

radius (*double*):

Außenradius der Fläche (cone, torus).

rad_ data (*double*):

Innenradius der Fläche (cone, torus).

norm_ dir (*int*):

Gibt die Normalenrichtung der Fläche an (+1 falls die Richtung mit der des Kreuzprodukts der Ableitungen in $u-$ und $v-$Richtung übereinstimmt, -1 sonst).

Diese Informationen liefern bereits einen ersten groben Überblick, mit welcher Art von Fläche man arbeitet. Wie das nächste Beispiel zeigt, reichen sie aber nicht aus, um Flächen zu unterscheiden, analytisch zu beschreiben oder gar Berechnungen auf ihnen auszuführen.

Beispiel 1. Gegeben seien zwei ebene Flächen wie in Abbildung 2.3 dargestellt. Die erste Platte ist quadratisch konstruiert, während bei der zweiten Platte das rechte obere Eck entfernt wurde. Verwendet man nun die Funktion UF_ MODL_ ask_ face_ data zur Extraktion der Geometrieinformationen dieser zwei Flächen erhält man (bis auf Translation) identische Werte.

Abbildung 2.3: Zwei unterschiedliche *bounded planes* (links) mit dazugehörigem umschließenden Quader (rechts)

Die Flächennormale, der Mittelpunkt der Fläche sowie das umschließende Quader sind in beiden Fällen gleich. Damit weiß der Benutzer nicht, ob es sich bei der Fläche vom Typ *bounded plane* um einem Quadrat oder um ein (reguläres) Vieleck handelt. Dieses Prinzip kann auf alle Oberflächentypen (Zylinder, Kegel, usw.) erweitert werden.

Flächen, die zum Beispiel durch den Schnitt mit einer Kurve oder Ebene verändert wurden, nennt man auch **getrimmte Oberflächen**. Im nächsten Kapitel wird geklärt, was genau der Begriff *getrimmt* bedeutet, wie man diese Information aus NX extrahieren kann und wozu sie benötigt wird.

2.3 Getrimmte Oberflächen

Definition 2. Eine Oberfläche heißt **getrimmt**, wenn die Grundfläche einem der in Kapitel 2.2 vorgestellten Typen entspricht und sie zusätzlich durch Kurven auf dieser Fläche beschränkt ist.

In der Praxis stellen solche Oberflächen keine Seltenheit dar, wie das nachfolgende Beispiel zeigt.

Beispiel 3. Ein Anwender möchte ein Stoppschild nach dem Vorbild eines echten Straßenschildes konstruieren. Dazu erstellt er eine Fläche in Form eines Oktagons. Intern speichert NX diese Fläche wie im letzten Kapitel vorgestellt als *bounded plane* und trimmt diese. Dies bedeutet im Speziellen, dass NX einerseits die Rechtecksfläche und andererseits die daraufliegenden Trimmkurven speichert (siehe Abbildung 2.4 links). Auf

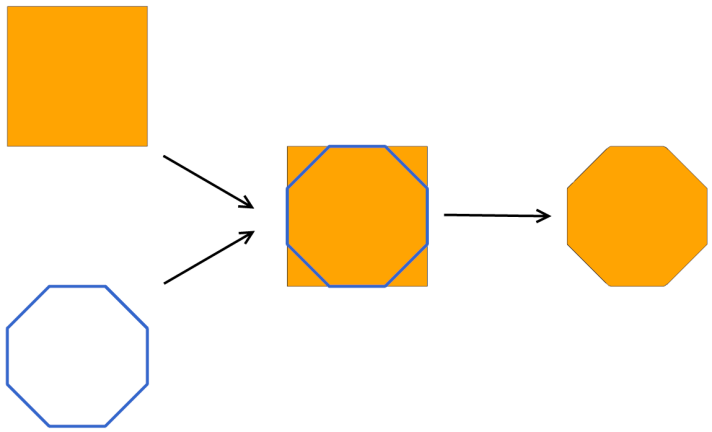


Abbildung 2.4: Interne Datenverarbeitung von getrimmten Oberflächen (links) bis zum optischen Endresultat (rechts)

der rechten Seite der Abbildung ist das optische Endresultat in der graphischen Oberfläche von NX zu sehen. Hieraus ist keineswegs ersichtlich, dass es sich intern um eine getrimmte Fläche handelt.

Um zu prüfen, ob eine Trimmung vorliegt, muss der Benutzer die Fläche markieren und im Menü unter ***Information*** → ***Object***[1] den Informationsdialog (siehe Abbildung 2.5) öffnen. Dort ist hinterlegt, ob NX die Fläche intern durch Trimmung erzeugt oder nicht. Unglücklicherweise bietet die Schnittstelle NXOpen keine zusätzliche Funktionalität diese Eigenschaft explizit abzufragen. Da es jedoch möglich ist, den Informationsdialog zu öffnen und in eine txt-Datei umzuleiten wurde folgender Workaround entwickelt:

```
1 ListingWindow lw = theSession.ListingWindow;
2 lw.SelectDevice(ListingWindow.DeviceType.File, currentPath);
3 theSession.Information.DisplayObjectsDetails(selectedObjects);
4 ...
5 Regex word = new Regex("Surface.*Type.*Trimmed");
6 ...
```

Zeile 2: Leitet die Ausgabe des Informationsdialoges in eine txt-Datei um.
Zeile 3: Ruft den Informationsdialog mit den selektierten Flächen auf.
Zeile 5: Definiert den regulären Ausdruck zum Überprüfen der Trimmung.

[1]Shortcut: (Strg + I)

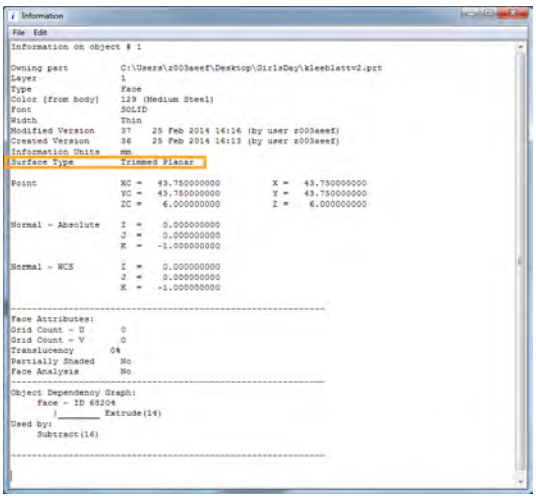


Abbildung 2.5: Informationsdialog in NX

Da der Benutzer an dieser Stelle weiß, dass die Fläche getrimmt ist, besteht die nächste Anforderung darin, die begrenzenden Kurven der Oberfläche zu extrahieren.

2.4 Extraktion von Kurven

Um alle begrenzenden Kurven einer gegebenen Oberfläche im Bildbereich ($\mathbb{R}^3$) zu extrahieren, verwendet man die Funktion

Face.GetEdges()

```
Edges[] GetEdges()
```

Edges: (*Edges[Number of Edges]*):

Rückgabewert dieser Funktion ist ein Array vom Typ *Edges*, in dem alle angrenzenden Kurven der Oberfläche gespeichert sind.

Die Klasse *Edge* beinhaltet eine Vielzahl von Methoden und Eigenschaften, die aber im Rahmen dieser Arbeit keinen analytischen Verwendungszweck finden. Vielmehr soll die Weiterverarbeitung dieser Kurve im Fokus stehen. Um die angrenzenden Kanten in eine analytische Kurve zu transformieren, wird ein Objekt vom Typ *UFEval* benötigt:

UF_ EVAL_ initialize

```
int UF_EVAL_initialize
(
tag_t          tag          Input: ID der Kante
UF_EVAL_p_t*   evaluator    Output: Evaluationsstruktur der Kante
)
```

tag (*tag_ t*):

Eindeutiger Identifier der Kante (NX intern abgespeichert).

evaluator (*int*):

Objektstruktur, mit der die Kante weiter ausgewertet werden kann.

Im nächsten Schritt können mit dem *evaluator* die zugehörigen Spline-Daten abgefragt werden:

UF_ EVAL_ ask_ spline

```
int UF_EVAL_ask_spline
(
UF_EVAL_p_t        evaluator  Input: Evaluationsstruktur der Kante
UF_EVAL_spline_p_t spline     Output: Spline-Daten
)
```

evaluator (*UF_ EVAL_ p_ t*):

Die Evaluationsstruktur der Kante, mit der ein Spline erzeugt werden kann.

spline (*UF_ EVAL_ spline_ p_ t*):

Nach erfolgreicher Erzeugung gibt die Funktion die Spline-Daten (Grad des Splines, etc.) zurück.

Zur vollständigen Beschreibung einer Spline-Kurve benötigt man noch die Kontrollpunkte sowie die Knotenfolge. Da die Signatur der Funktionen zur Extraktion dieser beiden Informationen nahezu identisch ist, wird exemplarisch vorgestellt, wie man die Knotenfolge erhalten kann und darauf verzichtet, den analogen Fall für die Kontrollpunkte durchzuführen. Für nähere Informationen hierzu empfiehlt sich die offizielle API von NXOpen.

UF_ EVAL_ ask_ spline_ knots

```
int UF_EVAL_ask_spline_knots
(
UF_EVAL_p_t    evaluator   Input: Evaluationsstruktur der Kante
int*           n_knots     Output: Anzahl der Knoten
double         knots[]     Output: Knotenfolge
)
```

evaluator (*UF_ EVAL_ p_ t*):

Die Evaluationsstruktur der Kante

n_ knots (*int*):

Knotenanzahl des erzeugten Splines.

knots (*double[n_ knots]*):

Die Knoten des Splines, gespeichert in einem Array der Länge n_ knots.

Die Kontrollpunkte der Splinekurve können gleichermaßen mit der Funktion **UF_ EVAL_ ask_ spline_ control_ pts** extrahiert werden.
Die Informationsgewinnung der begrenzenden Kanten als Spline-Kurven ist komplex und mit mehreren Aufrufen verbunden. Aus diesem Grund soll der gesamte schematische Ablauf nochmals etwas näher erläutert werden (siehe Abbildung 2.6).

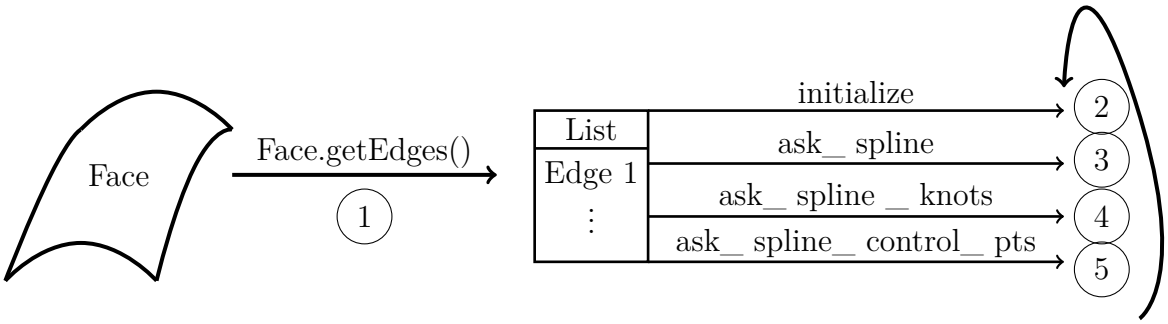


Abbildung 2.6: Ablauf der Kantenextraktion einer Fläche

Im ersten Schritt werden alle an eine Fläche angrenzenden Kanten extrahiert und in eine Liste von Kanten gespeichert (1). Anschließend werden für jedes Element in der Liste nacheinander vier Methoden aufgerufen. Nach der Initialisierung (2) kann der Spline ausgewertet werden und man erhält den Grad des Splines (3). Danach können die Kontrollpunkte (4) sowie der Knotenvektor (5) ausgelesen werden. Mit diesen Informationen ist es nun möglich, die begrenzenden Kurven exakt zu reproduzieren.

Die größte Rolle bei der Extraktion von Geometrie-Informationen spielen jedoch weder der in NX intern verwendete Oberflächentyp noch die begrenzenden Randkurven. Die wichtigste Aufgabe besteht darin, die Fläche selbst analytisch zu extrahieren.

2.5 Extraktion von Flächen

Wie bereits vorgestellt, speichert NX je nach Oberflächentyp verschiedene Daten; bei einem Kegel zum Beispiel wird in einem dreidimensionalen Vektor die Rotationsachse gespeichert, ist die Fläche eine ebene Platte wird in demselben Vektor die Normale der Fläche gespeichert. Um diesen Inkonsistenzen zu begegnen, wurde auch hier ein Workaround entwickelt. Im ersten Schritt wird die gegebene Fläche in eine sogenannte *I-Form* konvertiert. Dadurch werden Iso-Linien erzeugt mit deren Hilfe man eine Beschreibung als B-Spline-Fläche erhält, die über einen weiteren Befehl ausgelesen werden kann. Bei

der Konvertierung in eine *I-Form* wird die standardmäßig voreingestellte Fehlertoleranz von 0.0254 mm eingehalten. Auch wenn diese Vorgehensweise einen Fehler (innerhalb der Toleranz) impliziert, so hat sie den großen Vorteil, dass nun alle Flächen (Kegel, Kugel, Ebene, usw.) im selben Format vorliegen. Damit werden unnötige Fallunterscheidungen vermieden und der Programm-Code deutlich lesbarer. In Abbildung 2.7 ist das Schema einer solchen Flächenextraktion dargestellt:

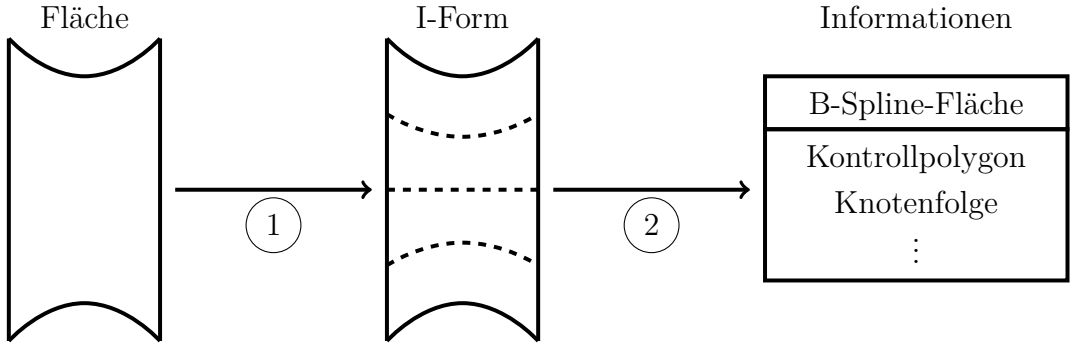


Abbildung 2.7: Extraktionsschema einer beliebigen Fläche als B-Spline-Fläche

Die Konvertierung einer beliebigen Fläche in eine *I-Form* (1) wird über den sogenannten *I-Form Builder* realisiert. Dieser bietet einen Konstruktor zur Erstellung solcher Flächen anhand bestimmter Regeln und Eigenschaften. Eine mögliche praktische Umsetzung könnte wie folgt aussehen:

```
1  Part workPart = theSession.Parts.Work;
2  NXOpen.Features.IForm nullFeatIForm = null;
3  NXOpen.Features.IFormBuilder iFormBuilder1;
4  iFormBuilder1 = workPart.Features.CreateIformBuilder(nullFeatIForm);
5  Face[] faces1 = {taggedFace};
6  FaceDumbRule faceDumbRule1;
7  faceDumbRule1 = workPart.ScRuleFactory.CreateRuleFaceDumb(faces1);
8  SelectionIntentRule[] rules2 = {faceDumbRule1};
9  iFormBuilder1.FaceToDeform.FaceCollector.ReplaceRules(rules2, false);
10 try      {return iFormBuilder1.Commit();}
11 catch    {return null;}
```

Zeile 1-4: Initialisierung des I-Form Builders zur Erzeugung einer I-Form.
Zeile 5-9: Die zu konvertierende Fläche muss zunächst in ein Array umgeschrieben werden. Anschließend wird eine FaceDumbRule erzeugt und ebenfalls in ein Array geschrieben. Dieses wird dann an den I-Form Builder übergeben.
Zeile 10: Nachdem alle Eigenschaften gesetzt sind, wird mit diesem Befehl der I-Form Builder aktiviert und NX versucht nun die Fläche zu erzeugen.
Zeile 11: Für den Fall, dass bei der Konvertierung Fehler aufgetreten sind, wird der Wert *null* zurückgegeben.

Im zweiten Schritt können nun die Knotenfolge sowie das Kontrollpolygon der Fläche mithilfe folgender Methode ausgelesen werden:

UF_ MODL_ ask_ bsurf

```
int UF_MODL_ask_bsurf
(
tag_t                    face       Input: ID der Oberflaeche
UF_MODL_bsurface_p_t     bsurf      Output: B-Spline Informationen
)
```

face (*tag_ t*):

Eindeutiger Identifier der Oberfläche (NX intern abgespeichert).

bsurf (*UF_ MODL_ bsurface_ p_ t*):

Eine von NX intern vorgegebene Struktur, in der alle relevanten Informationen der B-Spline Fläche gespeichert sind.

Die Datenstruktur einer Oberfläche ist komplexer als die einer einzigen Splinekurve. Aus diesem Grund stellt NXOpen für diesen Fall ein Struct zur Verfügung, das es ermöglicht, alle Informationen zur Beschreibung der Fläche in einer Datenstruktur zu speichern. Dieser Struct heißt **UF_ MODL_ bsurface_ p _ t** und beinhaltet folgende Variablen:

UF_ MODL_ bsurface_ p _ t

```
int UF_MODL_bsurface_p_t
(
int          num_poles_u      Output: Anzahl Kontrollpunkte in u
int          num_poles_v      Output: Anzahl Kontrollpunkte in v
int          order_u          Output: Grad in u
int          order_v          Output: Grad in v
int          is_rational      Output: Rationalitaet der Flaeche
double       knots_u[]        Output: Knotenfolge in u
double       knots_v[]        Output: Knotenfolge in v
double       poles[][]        Output: Kontrollpolygon
)
```

num_ poles_ u (*int*):

Gibt die Anzahl an Kontrollpunkten in u-Richtung an.

num_ poles_ v (*int*):

Gibt die Anzahl an Kontrollpunkten in v-Richtung an.

order_ u (*int*):

Gibt die Ordnung $m + 1$ der Oberfläche in u-Richtung an.

order_ v (*int*):

Gibt die Ordnung $m + 1$ der Oberfläche in v-Richtung an.

is_ rational (*int*):

Ist dieser Wert 0 so ist die Fläche polynomial, ist der Wert 1, ist die Fläche rational.

knots_ u (*double[num_ poles_ u]*):

Dieses Array beinhaltet die Knotenfolge in u-Richtung.

knots_ v (*double[num_ poles_ v]*):

Dieses Array beinhaltet die Knotenfolge in v-Richtung.

poles (*double[num_ poles_ u · num_ poles_ v][4]*):

In diesem Array ist das gesamte Kontrollpolygon gespeichert. Zu beachten ist hierbei folgendes Format: **poles[a][b]:**

- a ist Nummer des Kontrollpunktes
- b ist $\begin{cases} 0 & \widehat{=} \text{ x-Koordinate} \\ 1 & \widehat{=} \text{ y-Koordinate} \\ 2 & \widehat{=} \text{ z-Koordinate} \\ 3 & \widehat{=} \text{ Gewicht des Kontrollpunktes} \end{cases}$

2.6 Extraktion am Beispiel eines Straßenschildes

Der gesamte Extraktionsvorgang soll nun am Beispiel des bereits vorgestellten Straßenschildes stückweise erarbeitet werden. Abbildung 2.8 zeigt das Erscheinungsbild des Straßenschildes in der GUI von NX.

Abbildung 2.8: Graphische Darstellung des Straßenschildes in NX

Im ersten Schritt werden allgemeine Daten mithilfe von **UF_ MODL_ ask_ face_ data** ausgelesen:

- Oberflächentyp: 22 (bounded plane)
- Koordinaten der umschließenden Box (Abbildung 2.9(a))
- Koordinaten eines Punktes auf der Oberfläche (Abbildung 2.9(b))
- Richtung der Flächennormalen (Abbildung 2.9(c))

(a) Bounding Box (b) Punkt auf der Fläche (c) Flächennormale

Abbildung 2.9: Extraktion der Geometrieinformation eines Straßenschildes

Da der Workaround zur Überprüfung auf Trimmung den Oberflächentyp **Trimmed Planar** liefert, müssen im nächsten Schritt die begrenzenden Kurven extrahiert werden. Nachdem durch den Aufruf der Methode **Face.getEdges()** die Rohdaten der Kanten vorliegen, werden die in Abbildung 2.6 dargestellten Schritte durchlaufen, um als Ausgabe Splines zu erhalten. Die Kurven können nun in ein für Matlab verträgliches Format gebracht werden:

- S(1).m = 1;
- S(1).D=[5 3.5;8.5 10;0 0];
- S(1).T = [0 0 1 1];
- S(1).W = [1 1];
- S(1).r = 0;

Der Grad des Splines ist $m = 1$, die Kontrollpunkte d geben in diesem Fall den Start- und Endpunkt an. Anhand der Variablen r kann abgelesen werden, ob es sich um eine rationale Splinekurve handelt. Die Knotenfolge ist in T gespeichert und das Array w beinhaltet die Gewichte, die zur Beschreibung nicht-rationaler Splines notwendig sind. Mit diesen Informationen ist es nun möglich, die acht Randkurven in Matlab zu exportieren und plotten zu lassen:

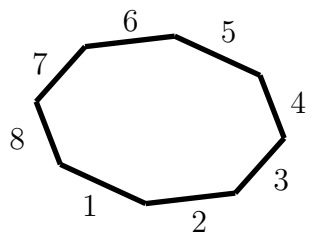


Abbildung 2.10: Begrenzende Splinekurven des Straßenschildes

Im letzten Schritt wird die Fläche mithilfe der Methode **UF_ MODL_ bsurface_ p_ t** ohne Berücksichtigung der Trimmung als B-Spline-Fläche extrahiert und ebenfalls in ein für Matlab kompatibles Format konvertiert:

- F(1).mu =3;
- F(1).mv =3;
- F(1).D(:,:,1) = [0 1.667 3.333 5;0 1.667 3.333 5;0 1.667 3.333 5;0 1.667 3.333 5];
- F(1).D(:,:,2) = [10 10 10 10;8.34 8.34 8.34 8.34;6.67 6.67 6.67 6.67;5 5 5 5];
- F(1).D(:,:,3) = [0 0 0 0;0 0 0 0;0 0 0 0;0 0 0 0];
- F(1).Tu = [-0.0025 -0.0025 -0.0025 -0.0025 0.0025 0.0025 0.0025 0.0025];
- F(1).Tv = [0 0 0 0 0.005 0.005 0.005 0.005];
- F(1).W = [1 1 1 1;1 1 1 1;1 1 1 1;1 1 1 1];
- F(1).r = 0;

Die Variablennamen der Flächen wurden analog zu den Kurven gewählt, das bedeutet m steht für den Grad, T für den Knotenvektor und D für das Kontrollpolygon (für Flächen jeweils in u- und v-Richtung).
In der folgenden Grafik ist die extrahierte Fläche ohne (Abbildung 2.11(a)) und mit (Abbildung 2.11(b)) begrenzenden Splinekurven dargestellt.

(a) Fläche ohne Trimmkurven (b) Fläche mit Trimmkurven

Abbildung 2.11: Matlab-Plot der extrahierten Fläche des Straßenschildes

Damit ist der Extraktionsprozess von geometrischen Eigenschaften aus beliebigen Oberflächen abgeschlossen. Mit dem vorliegenden Datenformat ist es nun möglich, die Flächen als B-Spline-Flächen (z.B. in Matlab) zu exportieren und damit Berechnungen (minimaler Abstand, Ableitungen, usw.) auf ihnen auszuführen.
Die Schnittstelle NXOpen bietet an dieser Stelle noch einige vordefinierten Methoden an, die sich im Umgang mit den Flächen als teilweise sehr nützlich erweisen. Da die Funktionalität dieser Aufrufe mit den hier bereits gesammelten Informationen durch gängige numerische Verfahren implementiert werden kann, muss die nachfolgende Beschreibung klar von der Extraktion von geometrischen Eigenschaften abgekapselt/abgekoppelt werden. Der Vollständigkeit halber werden die Methoden im Kontext der Schnittstellenbeschreibung von NXOpen bezüglich der Geometrie von Oberflächen vorgestellt, jedoch nicht mit Beispielen veranschaulicht.
Die erste nützliche Methode heißt **ask_ face_ parm** und bietet die Möglichkeit, zu einem gegebenen Punkt einer Fläche im Bildbereich (x, y, z) die Parameterwerte (u, v) zu erhalten:

UF_ MODL_ ask_ face_ parm

```
int UF_MODL_ask_face_parm
(
tag_t     face_id      Input: ID der Flaeche
double    ref_pnt[]    Input: Referenzpunkt im Bildbereich
double    parm[]      Output: Parameterwert (u,v) der Flaeche
double    face_pnt[] Output: Zugehoeriger Punkt auf der Flaeche
)
```

face_ id (*tag_ t*):

Eindeutiger Identifier der Fläche.

ref_ pnt (*double[3]*):

Punkt im Bildbereich, zu dem der Parameterwert unbekannt ist.

parm (*double[2]*):

Der Parameterwert (u, v) zum Referenzpunkt.

face_ pnt (*double[3]*):

Der zum Parameterwert gehörende Punkt auf der Fläche.

Im Optimalfall liegt der Referenzpunkt bereits auf der Fläche, so dass sich die Variablen ref_pnt und $face\_pnt$ nicht unterscheiden. Falls der Wert jedoch außerhalb liegt, wird der nächstgelegene Punkt der Fläche im Bildbereich verwendet.
Die nächste Methode, die NXOpen zur Verfügung stellt, bietet dem Benutzer die Möglichkeit, die kürzeste Distanz (inklusive Start- und Endpunkt) zwischen zwei geometrischen Objekten zu berechnen. Das Interface ist so generisch gehalten, dass es keinerlei Rolle spielt, von welchem Typ die beiden Objekte sind. Zulässig sind alle Kombinationen aus Punkten, Kurven, Oberflächen und Körpern.

UF_ MODL_ ask_ minimum_ dist

```
int UF_MODL_ask_minimum_dist
(
tag_t     object1          Input: ID von Objekt 1
tag_t     object2          Input: ID von Objekt 2
int       guess1_given     Input: Indikator fuer Vermutung 1
double    guess1[]         Input: Vermuteter Punkt 1
int       guess2_given     Input: Indikator fuer Vermutung 2
double    guess2[]         Input: Vermuteter Punkt 2
double*   min_dist        Output: Berechnete minimale Distanz
double    pt_on_obj1[]    Output: Punkt auf Objekt 1
double    pt_on_obj2[]    Output: Punkt auf Objekt 2
)
```

object1 (*tag_ t*):

Eindeutiger Identifier von Objekt 1.

object2 (*tag_ t*):

Eindeutiger Identifier von Objekt 2.

guess1_ given (*int*):

Setze Wert auf 1, falls eine naheliegende Vermutung vorliegt, sonst auf 0.

guess1 (*double[3]*):

Punkt im Bildbereich, der auf kürzester Distanz zwischen den zwei Objekten liegt.

guess2_ given (*int*):

Analog zu guess1_ given.

guess2 (*double[3]*):

Analog zu guess1.

min_ dist (*int*):

Die berechnete minimale Distanz zwischen den zwei Objekten.

pt_ on_ obj1 (*double[3]*):

Punkt des minimalen Abstands auf Objekt 1.

pt_ on_ obj2 (*double[3]*):

Punkt des minimalen Abstands auf Objekt 2.

Diese beiden durch NXOpen bereitgestellten Funktionalitäten ersparen die Implementierung des Newton-Verfahrens und der damit verbundenen Fallunterscheidungen der verwendeten Objekte.
Die Methode **UF_ MODL_ ask_ face_ uv_ minmax** bietet die Möglichkeit, die Grenzen des Parameterbereichs auszulesen ohne den Umweg über das erste und letzte Element des jeweiligen Knotenvektors gehen zu müssen.

UF_ MODL_ ask_ uv_ minmax

```
int UF_MODL_ask_uv_minmax
(
tag_t     face_tag        Input: ID der parametrischen Flaeche
double    uv_min_max[]    Output: Grenzen des Parameterbereichs
)
```

face_ tag (*tag_ t*):

Eindeutiger Identifier der parametrisierten Fläche.

uv_ min_ max (*double[4]*):

Ausgabearray, das folgende Grenzen beinhaltet:
$$\begin{cases} [0] & \widehat{=} & \text{Minimum von u} \\ [1] & \widehat{=} & \text{Maximum von u} \\ [2] & \widehat{=} & \text{Minimum von v} \\ [3] & \widehat{=} & \text{Maximum von v} \end{cases}$$

Die letzte hier vorgestellt Methode dient hauptsächlich zur Bestimmung der ersten und zweiten partiellen Ableitung einer Fläche. Diese Funktionalität wurde im Zuge dieser Masterarbeit jedoch eigenständig implementiert, um auch gemischte partielle Ableitungen berechnen zu können. Da diese Methode für weiterführende Tätigkeiten von Nutzen sein kann, wird sie hier aufgeführt:

UF_ MODL_ ask_ face_ props

```
int UF_MODL_ask_face_props
(
tag_t     face_tag         Input: ID der parametrischen Flaeche
double    param[]          Intpt: Parameterwert (u,v) der Flaeche
double    point[]         Output: Punkt im Bildbereich der Flaeche
double    u1[]            Output: Erste partielle Ableitung in u
double    v1[]            Output: Erste partielle Ableitung in v
double    u2[]            Output: Zweite partielle Ableitung in u
double    v2[]            Output: Zweite partielle Ableitung in v
double    unit_norm[]     Output: Flaechennormale
double    radii[]         Output: Kruemmungsradius

)
```

face_ tag (*tag_ t*):

Eindeutiger Identifier der parametrisierten Fläche.

param (*double[2]*):

Parameterwert (u, v) der Fläche.

point (*double[3]*):

Der zum Parameterwert zugehörige Punkt im Bildbereich.

u1 (*double[3]*):

Erste partielle Ableitung in u-Richtung.

v1 (*double[3]*):

Erste partielle Ableitung in v-Richtung.

u2 (*double[3]*):

Zweite partielle Ableitung in u-Richtung.

v2 (*double[3]*):

Zweite partielle Ableitung in v-Richtung.

unit_ norm (*double[3]*):

Vektor der Flächennormalen.

radii (*double[2]*):

Gibt den Krümmungsradius an diesem Punkt an.
Hinweis: Handelt es sich bei der Fläche um eine ebene Platte (siehe Abbildung 2.8),

wird als Krümmungsradius der Maximalwert 393700786.401575 ausgegeben.

Unter Zuhilfenahme der vorgestellten Methoden ist es dem Benutzer über eine Applikation möglich, Anfragen an die Oberflächengeometrie zu stellen und diese in einer analytischen Form zu extrahieren. Mit diesen Informationen kann nun in einem weiteren Schritt unabhängig von NX mithilfe numerischer Verfahren geprüft werden, welche Beschaffenheit die Flächen und insbesondere die Übergange aufweisen.

3 Mathematische Grundlagen

Das nachfolgende Kapitel liefert die mathematischen Grundlagen, um mit den extrahierten Daten aus NX Aussagen über Oberflächenverläufe und Flächenübergänge treffen zu können. Zu Beginn wird die Theorie von Kurven und B-Splines erläutert und später mithilfe des Tensorprodukts auf Flächen erweitert.

3.1 Kurven, B-Splines und Tensorprodukt-Flächen

In der Mathematik gibt es eine Vielzahl von Anwendungsgebieten, die es erfordern unterschiedliche Darstellungsformen für geometrische Objekte zu verwenden. Der Einheitskreis beispielsweise lässt sich als Lösung des Gleichungssystems

$$x^2 + y^2 - 1 = 0 \text{ (implizite Darstellung)}$$

darstellen. Oftmals sind solche Darstellungsformen für den Einsatz in der Praxis nicht sinnvoll, da das Lösen vieler Gleichungssysteme für einzelne Punkte enorm rechenaufwändig sein kann. Eine Ausnahme stellt zum Beispiel das Raytracing dar: Hierbei werden Strahlen aus einer Quelle ausgesandt und mit Objekten geschnitten. Die Berechnung dieses Schnittes erfolgt über das Lösen eines Gleichungssystems. Eine für die Zwecke dieser Masterarbeit weitaus günstigere Form ist die parametrische Darstellung mithilfe von reellwertigen Vektorfunktionen

$$f(t) = (\cos(t), \sin(t))^T,\ t \in\ [0, 2\pi] \text{ (parametrische Darstellung).}$$

Diese sind im Allgemeinen nicht eindeutig, eignen sich jedoch wesentlich besser um Berechnungen (wie z.B. Ableitungen) durchzuführen. Im Folgenden beziehen sich Kurven und Funktionen immer auf parametrische Darstellungsformen und die Notation der folgenden Kapitel deckt sich weitestgehend mit der aus [5]. Für ausführlichere Informationen zum Thema Kurven empfiehlt sich außerdem [3].

3.1.1 Raumkurven

Definition 4. Eine parametrische (Raum-)Kurve ist eine Abbildung $f : [a, b] \to \mathbb{R}^3$ mit $f(t) = (f_1(t), f_2(t), f_3(t))^T$ wobei $a, b \in \mathbb{R}$ gilt.

Im Folgenden heißt t *Parameter* und das Intervall $[a, b]$ *Parameterbereich* von f. Die nachfolgende Abbildung 3.1 zeigt auf der linken Seite den Parameterbereich $[a, b]$ und die dazugehörige Raumkurve f rechts.

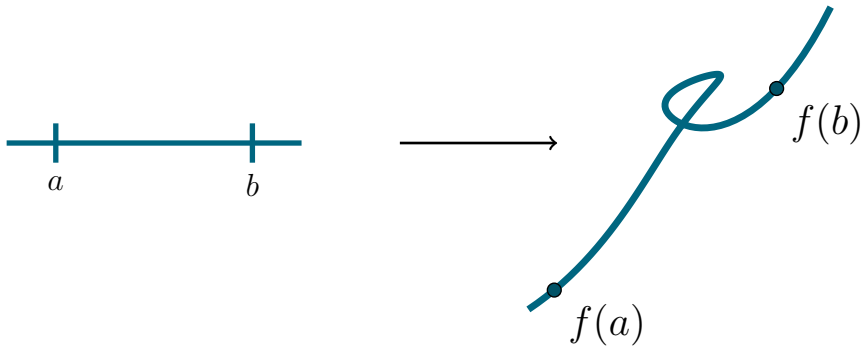


Abbildung 3.1: Raumkurve definiert auf einem Parameterbereich $[a, b]$

Bemerkung 5. Für die nachfolgenden Betrachtungen werden für alle Kurven $f : [a, b] \to \mathbb{R}^3$ folgende Annahmen getroffen:

1. f ist mindestens zwei mal stetig differenzierbar in $[a, b]$
2. $f'(t) \neq (0, 0, 0)^T \quad \forall\, t \in [a, b]$

Ist letztere Bedingung nicht erfüllt, spricht man von einem *kritischen Punkt*. An diesen Stellen t treten Sonderfälle auf, die in einem eigenen Kapitel behandelt werden.

Durch geeignete Abbildungen ist es möglich, den Parameterbereich auf dem f existiert zu transformieren, ohne das Bild von f zu verändern:

Definition 6. Ist $f : [a, b] \to \mathbb{R}^3$ eine parametrische Kurve und $\varphi : [c, d] \to [a, b]$ eine surjektive Abbildung, so heißt φ *reguläre Reparametrisierung*, wenn sie stetig differenzierbar ist und stets $\dot{\varphi} \neq 0$ gilt.

In diesem Zusammenhang fällt oft der Begriff *Bogenlänge*, also die Länge einer Kurve oder eines Kurvenstücks. Diese Eigenschaft besitzt den Vorteil, dass sie bezüglich regulärer Reparametrisierungen invariant ist (vgl. [9]).

Definition 7. Sei $f : [a, b] \to \mathbb{R}^3$ eine Kurve und $a < u \leq b$. Dann heißt $s(u) = \int_a^u \|\dot{f}(t)\|_2 \, \mathrm{d}t$ die *Bogenlänge* des Kurvenstücks von a bis u.

Existiert eine reguläre Reparametrisierung $\varphi : u \to s(u)$, so kann die Kurve nach der Bogenlänge parametrisiert werden. Für die Ableitung der Kurve schreibt man dann $f'(s)$ anstatt $\dot{f}(t)$ und es gilt $f'(s) = \frac{\dot{f}(t(s))}{\|\dot{f}(t(s))\|_2}$. Mit anderen Worten bedeutet dies, dass die Ableitung einer nach der Bogenlänge parametrisierten Kurve stets ein Einheitsvektor ist.

Diese Theorie reicht bereits aus, um die sogenannten B-Splines einzuführen, die eine in dieser Arbeit sehr bedeutende Rolle spielen. Für weitere und detaillierte Informationen zum Thema parametrische Kurven empfiehlt sich [5].

3.1.2 B-Splines

B-Splines sind Kurven, die aus stückweise polynomialen Abbildungen $f : [x, y] \to \mathbb{R}^d$ bestehen und an ihren Endpunkten „zusammengebaut“ werden. Da diese zunächst für sich gesehen jeweils unendlich oft differenzierbar sind, stellt sich die Frage, wie man diese Teilstücke derart zusammensetzt, um auch an den Übergangsstellen ein möglichst gutmütiges Verhalten (im Sinne von Differenzierbarkeit und Stetigkeit) garantieren zu können. Hierfür benötigt man zunächst eine formale Definition der Intervalle und Übergangsstellen:

Definition 8. Eine Menge $T = \{t_1, \ldots, t_{m+n+1}\}$ heißt Knotenfolge der Ordnung m, wenn gilt:

$$t_1 \leq \ldots \leq t_{m+n+1}$$

$$t_i < t_{i+m+1}$$

Bemerkung 9. Eine Knotenfolge der Ordnung m ist anschaulich eine monoton steigende Folge von Punkten, mit der Einschränkung, dass nie mehr als $m + 1$ gleiche Knoten aufeinanderfolgen. In der Fachliteratur wird häufig von einer *einfachen Knotenfolge* gesprochen, wenn $m =$ Polynomgrad gilt.

Auf der oben definierten Menge T ist es nun möglich die B-Splines wie folgt zu definieren:

Definition 10. Für $k = 0, \ldots, m$ sind die B-Splines N_j^k der Ordnung $k + 1$ rekursiv definiert als

$$N_j^0(\cdot|T) = \chi_{[t_j, t_{j+1})}$$

$$N_j^k(\cdot|T) = \frac{\cdot - t_j}{t_{j+k} - t_j} N_j^{k-1}(\cdot|T) + \frac{t_{j+k+1} - \cdot}{t_{j+k+1} - t_{j+1}} N_{j+1}^{k-1}(\cdot|T)$$

wobei $j = 1, \ldots, n + m - k$ gilt und χ die charakteristische Funktion ist (vgl. [2]).

Durch diese rekursive Definition der B-Splines ergeben sich einige sehr nützliche Eigenschaften, die durch genauere Betrachtung klar werden. In Abbildung 3.2 ist eine rekursive Auflösung von $N_i^k(t|T)$ bis zur Auswertung an den einzelnen Intervallen dargestellt. Man erkennt, dass selbst wenn die Knotenmenge T unendlich groß ist, der Träger des jeweiligen B-Splines vom Grad k immer endlich ist (in Abbildung 3.2 blau dargestellt). Diese Beobachtung beantwortet zugleich die Frage, wie sich die Rekursionsformel für einen $(k+1)$-fachen Knoten verhält, denn dann würde der Nenner $t_{j+k+1} - t_j$ zu 0 werden. Da der Träger dieses Splines jedoch die leere Menge ist, kann man diesen Term ignorieren. Durch die Linearkombinationen der B-Splines mit geeigneten Koeffizienten d_j kann man nun Splinekurven der Form

$$S_k \cdot d = \sum_{j=1}^{n} d_j \cdot N_j^m(\cdot|T)$$

erzeugen. Wie die Kurve letztendlich aussieht, hängt von drei verschiedenen Faktoren maßgeblich ab: den Koeffizienten d_j, der Knotenfolge T sowie dem Grad m der stückweisen Polynome.

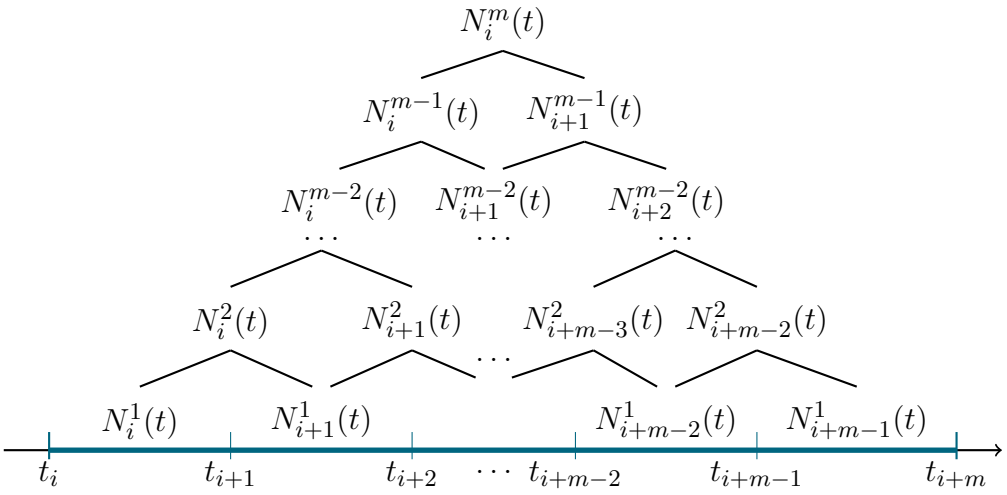


Abbildung 3.2: Rekursionsbaum der B-Splines

Definition 11. Die Koeffizienten $d = (d_j)$ mit $1 \leq j \leq n$ zu einer gegebenen Knotenfolge T und B-Splines vom Grad m heißen *Kontrollpolygon*.

Die nachfolgenden zwei Beispiele sollen die theoretischen Definitionen veranschaulichen. Im ersten Fall werden nur die Basis-Funktionen N_j^m betrachtet und graphisch dargestellt, im zweiten Beispiel wird der Zusammenhang von Kontrollpolygon und Knotenfolge näher erläutert.

Beispiel 12. Zunächst sollen nur die Funktionen $N_1^k, \ldots, N_n^k$ für $m = 3$ ausgewertet und graphisch dargestellt werden. Dazu benötigt man $n+m+1$ viele Knoten. Sei also die Knotenfolge gegeben durch $T = \{0, 1, 2, 3, 4, 5, 6\}$. Daraus ergeben sich drei B-Splines, die in Abbildung 3.3 zu sehen sind. Man kann erkennen, dass jede einzelne Funktion nur in einem Intervall der Länge 4 „lebt“. Erhöht man den Grad m auf 5 und lässt die Knotenfolge unverändert, ergibt sich nur noch eine einzige B-Spline Funktion (Abbildung 3.4), die sich aber auf ein Intervall der Länge 6 ausbreitet.

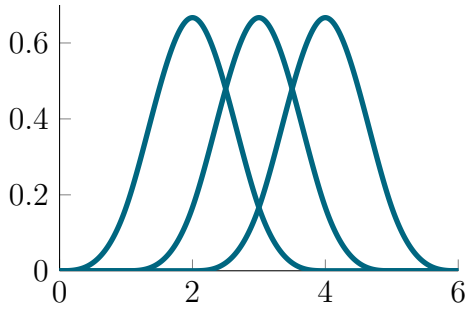


Abbildung 3.3: B-Splines $m = 3$

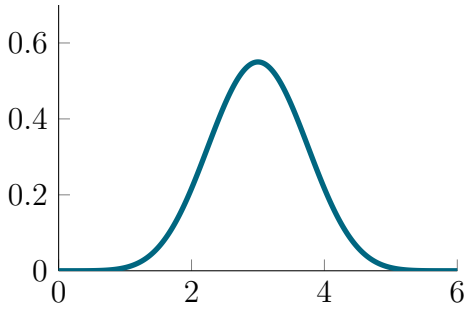


Abbildung 3.4: B-Splines $m = 5$

Man kann erkennen, dass der Grad m eines Splines bei gleichbleibender Knotenfolge Auswirkungen auf die Anzahl der B-Splines hat. Im nachfolgenden Beispiel bleibt der Grad $m = 3$ unverändert, die Knotenfolge jedoch variiert.

Beispiel 13. In diesem Anwendungsbeispiel wird die Funktionsweise der Interpolation mithilfe der Teilung der 1 durch die Basis-Funktionen erklärt. Der Grad der Spline-Funktion sei $m = 3$ und die erste Knotenfolge gegeben durch $T_1 = \{0, 0, 0, 0.5, 0.5, 1, 1, 1\}$. Die Basis-Funktionen zu dieser Knotenfolge sind in Abbildung 3.5 dargestellt. Die im Graphen orange dargestellte Kurve ist die Summe aller Basis-Funktionen im Definitionsbereich von 0 bis 1. Wählt man nun die Knotenfolge $T_2 = \{0, 0, 0, 0, 1, 1, 1, 1\}$, um an Start- und Endpunkt jeweils einen $k + 1$-fachen Knoten erhält, ist die Summe aller Basis-Funktionen stets 1 (Abbildung 3.6).

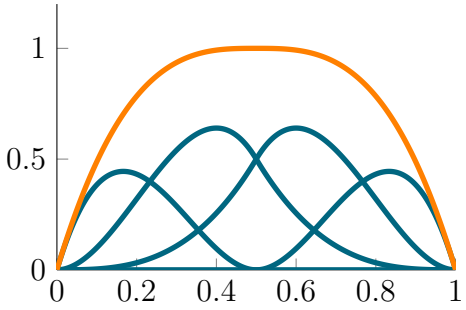


Abbildung 3.5: Basis-Funktionen für $m = 3$ zu T_2

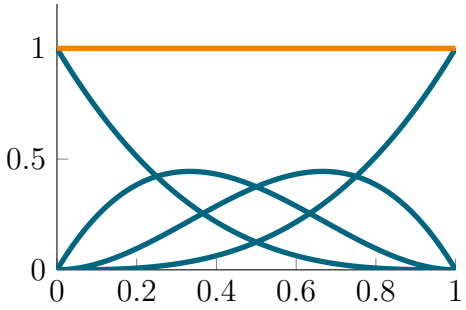


Abbildung 3.6: Basis-Funktionen für $m = 3$ zu T_2

Bemerkung 14. An einem $k+1$-fachen Knoten interpoliert die Spline-Kurve das Kontrollpolygon. Diese Behauptung folgt direkt aus der rekursiven Definition der B-Splines, denn die Auswertung von N_j^k an einer Stelle x eines $k+1$-fachen Knotens ergibt gerade

$$N_j^k(x|T) = N_{j+k+1}^0(x|T),$$

da alle anderen Splines, die in der Rekursion entstehen, leeren Träger haben. Die Interpolation lässt sich nun durch die Betrachtung der Spline-Kurve $S_k \cdot d = \sum_{j=1}^{n} d_j \cdot N_j^k(\cdot|T)$ erklären, da die Summe an der Stelle x des $k+1$-fachen Knotens lediglich einen nichttrivialen Term ungleich 0 enthält. Dieser ist genau $d_j \cdot N_{j+k+1}^0(x|T) = d_j$.

Beispiel 15. Das letzte Beispiel veranschaulicht zwei Splinekurven mit dem Kontrollpolygon

$$d = \begin{pmatrix} 1 & 2 & 4 & 5 \\ 1 & 4 & 4 & 1 \end{pmatrix}$$

und unterschiedlichen Knotenfolgen

$$T_1 = \{0, 0, 0, 0, 0.1, 0.2, 0.8, 0.9, 2, 2, 2, 2\}$$

$$T_2 = \{0, 0, 0, 0, 0.1, 0.1, 0.1, 0.1, 2, 2, 2, 2\}$$

Die beiden Knotenfolgen unterscheiden sich in der Anzahl ihrer $k + 1$-fachen Knoten. Während T_1 jeweils am Anfang und am Ende einen vierfachen Knoten besitzt, besteht

T_2 aus einem zusätzlichen vierfachen Knoten an der Stelle 0.1.
Das Resultat der ersten Splinekurve mit Grad $m = 3$ angewendet mit dem Kontrollpolygon d und die Knotenfolge T_1 ist in Abbildung 3.7 zu sehen. Wie zu erwarten interpoliert die Splinekurve wegen der beiden vierfachen Randknoten den ersten bzw. letzten Punkt des Kontrollpolygons.

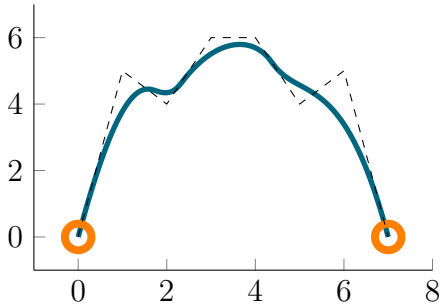


Abbildung 3.7: Splinekurve vom Grad 3 zur Knotenfolge T_1

Die Knotenfolge T_2 hat nun nicht nur an den Randknoten $k + 1$-fache Knoten sondern auch dazwischen. Die resultierende Splinekurve interpoliert damit an 3 verschiedenen Stützstellen (siehe Abbildung 3.8). Man kann erkennen, dass die Kurve an der zweiten Interpolationsstelle nur noch stetig ist.

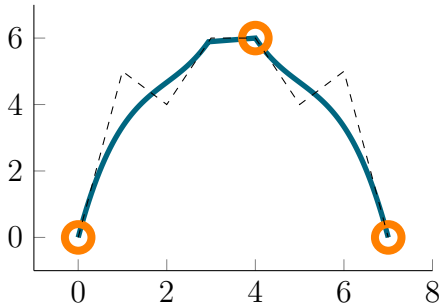


Abbildung 3.8: Splinekurve vom Grad 3 zur Knotenfolge T_2

An dieser Stelle wird deutlich, warum moderne CAD-Systeme standardmäßig auf Splines zur Konstruktion verschiedener Modelle zurückgreift. Spline-Funktionen eignen sich aufgrund ihrer Flexibilität hervorragend zur Modellierung komplexer Formen, ohne dabei Polynomfunktionen sehr hohen Grades zu verwenden. Dies ist vor allem für die Berechnungen innerhalb des CAD/CAM-Systems von essentieller Bedeutung, da Operationen wie die Nullstellenberechnung für Funktionen hohen Grades schnell nicht mehr echtzeitfähig umgesetzt werden können. Darüber hinaus können sie anhand von folgenden drei Eigenschaften eindeutig (Satz von Curry & Schönberg [9]) charakterisiert und rekonstruiert werden:

- Grad der Polynome **m**
- Kontrollpolygon **d**
- Knotenfolge **T**

Möchte man Splinekurven an einer Stelle x im Parameterbereich auswerten, könnte man für jedes $N_j^k(x|T)$ die Rekursionsformel für B-Splines auflösen und mit dem entsprechenden d_j multiplizieren. Dieses naive Verfahren ist für ein großes Kontrollpolygon jedoch sehr ressourcenlastig, da die Rekursion oftmals sehr rechenaufwendig wird. Ein effizienteres Verfahren bietet der von Carl de Boor entwickelte Algorithmus aus dem Jahr 1972 [1]. Dieser ist eine Verallgemeinerung des de Casteljau Algorithmus und basiert auf der Idee an einer Stelle u im Parameterbereich einen $m+1$-fachen Knoten einzufügen. Auch wenn die Verbindung des deBoor Algorithmus mit der Knoteneinfügeoperation erst später entstand, ist diese anschauliche Beschreibung vorteilhaft, da für eine steigende Anzahl an Knoten die Anzahl an nicht-trivialen Basisfunktionen[1] für diesen Punkt sinkt. Eine ausführliche Erläuterung sowie ein Korrektheitsbeweis des de Boor Algorithmus ist in [9] nachzulesen.
Der nächste Schritt in der Analyse von Splinekurven ist die Ableitung an einer Stelle der Kurve zu bestimmen. Im Folgenden wird eine Möglichkeit vorgestellt die Ableitung eines Splines an einer Stelle x des Parameterbereiches zu berechnen:

Lemma 16. Die Ableitung eines B-Splines $N_j^m(x|T)$ ist gegeben durch

$$\frac{\mathrm{d}}{\mathrm{d}x}N_j^m(x|T) = \frac{m}{t_{j+m}-t_j}N_j^{m-1}(x|T) - \frac{m}{t_{j+m+1}-t_{j+1}}N_{j+1}^{m-1}(x|T)$$

Beweis: Siehe [8] Kapitel 8 Splines (S. 158ff).

Diese Ableitungsformel für B-Splines kann nun angewendet werden, um ganze Splinekurven mit gegebenem Knotenvektor T und Kontrollpolygon d abzuleiten:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}x}(S_k \cdot d)(x) &= \sum_{j=1}^{n} d_j \left(\frac{k}{t_{j+k}-t_j}N_j^{k-1}(x|T) - \frac{k}{t_{j+k+1}-t_{j+1}}N_{j+1}^{k-1}(x|T)\right)\\ &= \sum_{j=1}^{n}\frac{k\cdot d_j}{t_{j+k}-t_j}N_j^{k-1}(x|T) - \sum_{j=2}^{n+1}\frac{k\cdot d_{j-1}}{t_{j+k}-t_j}N_j^{k-1}(x|T)\\ &= \sum_{j=1}^{n+1}\frac{k\cdot(d_j-d_{j-1})}{t_{j+k}-t_j}N_j^{k-1}(x|T)\end{aligned}$$

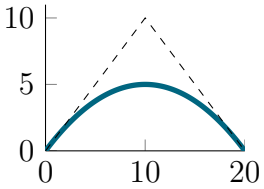
für $d_0 = d_{n+1} = 0$.
Bei der näheren Betrachtung der Ableitungsformel wird ersichtlich, wie der Zusammenhang zwischen der Länge des Ableitungsvektors und der Parametrisierung zustande kommt. Liegen die Knotenpunkte sehr weit voneinander entfernt, ergeben sich große

[1] Basisfunktionen, die ungleich 0 sind

Werte für die Differenzen im Nenner und verkürzen damit die Länge des Ableitungsvektors. Umgekehrt ergeben sich kleine Werte im Nenner, wenn die Punkte der Knotenfolge sehr nahe zusammen liegen.

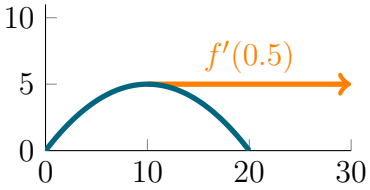
Beispiel 17. Gegeben sei eine Splinekurve f vom Grad $m = 2$ mit Knotenvektor T und dazugehörigen Kontrollpolygon d:

$$T = \{0, 0, 0, 1, 1, 1\}, \qquad d = \begin{pmatrix} 0 & 10 & 20 \\ 0 & 5 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$



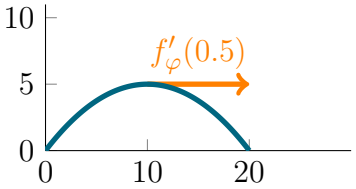
Durch Anwenden des Algorithmus von de Boor und der Ableitungsformel für Splinekurven erhält man für $t = 0.5$ und obiges T:

$$f(0.5) = \begin{pmatrix} 10 \\ 5 \\ 0 \end{pmatrix} \qquad f'(0.5) = \begin{pmatrix} 20 \\ 0 \\ 0 \end{pmatrix}$$



Wählt man nun eine Reparametrisierung $\varphi(t) = 2t$ ergibt sich für $t = 0.5$:

$$f_\varphi(0.5) = \begin{pmatrix} 10 \\ 5 \\ 0 \end{pmatrix} \qquad f'_\varphi(0.5) = \begin{pmatrix} 10 \\ 0 \\ 0 \end{pmatrix}$$



Möchte man die zweite Ableitung einer Splinekurve bestimmen, muss die Ableitungsformel ein weiteres Mal in $\frac{d}{dx} S_k d(x)$ einsetzen:

Definition 18. Die zweite Ableitung eines B-Splines $N_j^m(x|T)$ ist gegeben durch

$$\begin{aligned} \frac{\mathrm{d}^2}{\mathrm{d}x^2} N_j^m(x|T) &= \frac{m}{t_{j+m} - t_j} \left(\frac{k-1}{t_{j+k-1} - t_j} N_j^{k-2}(x|T) - \frac{k-1}{t_{j+k} - t_{j+1}} N_{j+1}^{k-2}(x|T) \right) \\ &- \frac{m}{t_{j+m+1} - t_{j+1}} \left(\frac{k-1}{t_{j+k} - t_{j+1}} N_{j+1}^{k-2}(x|T) - \frac{k-1}{t_{j+k+1} - t_{j+2}} N_{j+2}^{k-2}(x|T) \right). \end{aligned}$$

Auch hier wird die Ableitungsformel für B-Splines verwendet, um die zweite Ableitung einer Splinekurve zu bestimmen. Durch Indexverschiebung kann man die Terme derart

zusammenfassen, dass pro Schleifendurchlauf nur noch eine Basisfunktion ausgewertet werden muss:

$$
\begin{aligned}
&\frac{\mathrm{d}^2}{\mathrm{d}x^2} S_k d(x) = \\
&= \sum_{j=1}^{n+1} \frac{k(d_j - d_{j-1})}{t_{j+k} - t_j} \left(\frac{k-1}{t_{j+k-1} - t_j} N_j^{k-2}(x|T) - \frac{k-1}{t_{j+k} - t_{j+1}} N_{j+1}^{k-2}(x|T) \right) \\
&= \sum_{j=1}^{n+1} \left(\frac{k(k-1)(d_j - d_{j-1})}{(t_{j+k} - t_j)(t_{j+k-1} - t_j)} N_j^{k-2}(x|T) - \frac{k(k-1)(d_j - d_{j-1})}{(t_{j+k} - t_j)(t_{j+k} - t_{j+1})} N_{j+1}^{k-2}(x|T) \right) \\
&= \sum_{j=1}^{n+1} \frac{k(k-1)(d_j - d_{j-1})}{(t_{j+k} - t_j)(t_{j+k-1} - t_j)} N_j^{k-2}(x|T) - \sum_{j=2}^{n+2} \frac{k(k-1)(d_{j-1} - d_{j-2})}{(t_{j+k-1} - t_{j-1})(t_{j+k-1} - t_j)} N_j^{k-2}(x|T) \\
&= \sum_{j=1}^{n+2} \frac{k(k-1)}{t_{j+k-1} - t_j} \left(\frac{d_j - d_{j-1}}{t_{j+k} - t_j} - \frac{d_{j-1} - d_{j-2}}{t_{j+k-1} - t_{j-1}} \right) N_j^{k-2}(x|T)
\end{aligned}
$$

für $d_{-1} = d_0 = d_{n+1} = d_{n+2} = 0$.

Die Auswertung mit dem de Boor Algorithmus und die Bestimmung der ersten und zweiten Ableitungen ermöglichen die Stetigkeit an einem Kurvenübergang zu untersuchen. Hierbei muss zwischen der geometrischen (G-) und der parametrischen (C-) Stetigkeit unterschieden werden. Wie bereits im letzten Beispiel gesehen, hat die Parametrisierung der Kurve Einfluss auf die Länge des Ableitungsvektors. Bei der G-Stetigkeit spielen lediglich die Richtungen der Ableitungen eine Rolle, bei der C-Stetigkeit sind zusätzlich Bedingungen an die Länge der Vektoren gestellt.

Definition 19. Seien f, g zwei Kurven mit $f : [a, b] \to \mathbb{R}^3$, $g : [c, d] \to \mathbb{R}^3$. Dann existiert ein Kurvenübergang von f nach g, wenn $t_1 \in [a, b]$ und $t_2 \in [c, d]$ existieren, so dass

$$f(t_1) = g(t_2) \tag{3.1}$$

gilt. Der Kurvenübergang heißt dann C^0-*stetig*.

Bemerkung 20. Für die nachfolgenden Betrachtungen werden folgende Einschränkungen getroffen:

1. Es gilt $t_1 \in \{a, b\}$ und $t_2 \in \{c, d\}$, das heißt ein Kurvenübergang ist nur am ersten oder letzten Knoten der jeweiligen Kurve erlaubt.

2. Es gibt für zwei Kurven f, g nur einen Kurvenübergang.

Abbildung 3.9 zeigt ein kommutatives Diagramm, das die Definition eines C^0-stetigen Kantenübergangs veranschaulicht. In der unteren Hälfte der Grafik ist der Parameterbereich $[a, d']$ zu sehen. Wendet man nun die Funktion f auf den Bereich $[a, b]$ an, erhält man die erste Splinekurve (blau). Die Reparametrisierung τ bildet das abgeschlossene

Intervall $[b, d']$ auf den urspünglichen Parameterbereich von g ab. τ kann also als Translation des Parameterbereichs von g realisiert werden, so dass ein zusammenhängendes Intervall $[a, d']$ entsteht. Damit erhält man an dem C^0-Übergang $f(b) = g(\tau(b)) = g(c)$.

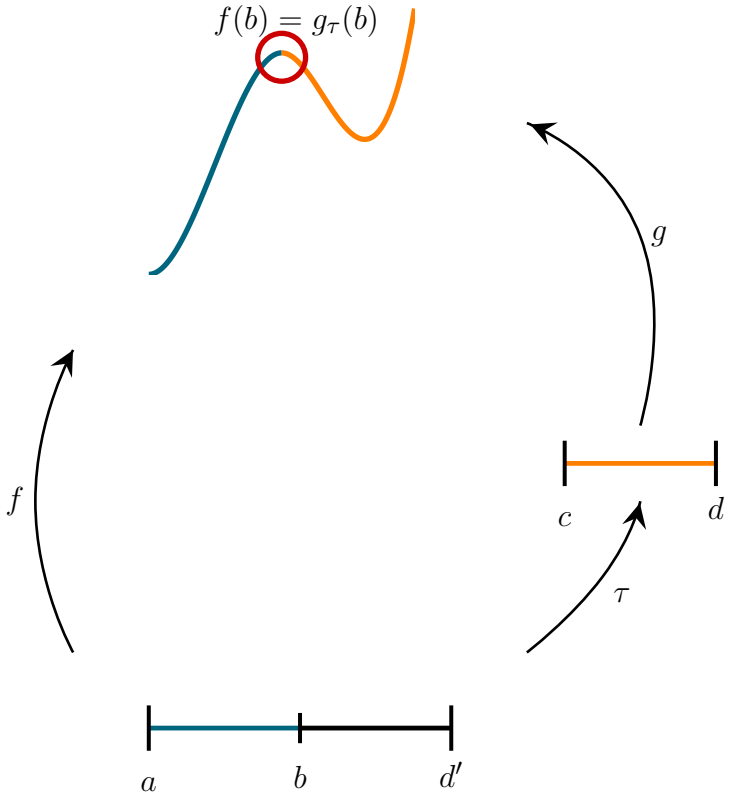


Abbildung 3.9: Diagramm eines C^0-Kantenübergangs

An einem Kurvenübergang treffen die Eckpunkte zweier Kurven aufeinander. In Abbildung 3.10 sind zwei solcher möglichen Kurvenübergänge dargestellt.

Abbildung 3.10: C^0-stetiger Kurvenübergang

Während der linke Kurvenübergang einen Knick aufweist, ist der rechte Übergang glatt. Diese optische Glattheit nennt man G^1-Stetigkeit:

Definition 21. Sei $f(t_1) = g(t_2)$ ein C^0-stetiger Kurvenübergang der beiden Kurven f und g wie in Definition 16. Der Übergang heißt *G^1-stetig*, falls eine Reparametrisierung $\varphi : \mathbb{R} \to \mathbb{R}$ existiert, so dass

$$\dot{f}(t_1) = \dot{g}_{\varphi}(t_2)$$

mit $\dot{\varphi} > 0$ gilt. Ist die Reparametrisierung φ die Identitätsfunktion, so wird der Übergang C^1 genannt.

Da in der Praxis die Reparametrisierung φ in der Regel nicht bekannt ist, liefert der nachfolgende Satz ein Kriterium zur Überprüfung der G^1-Stetigkeit.

Satz 22. *Seien f, g zwei Splinekurven mit zugehörigen Knotenfolgen T_f, T_g und Kontrollpolygonen $d, \tilde{d}$ mit einem C^0-stetigen Übergang $f(t_1) = g(t_2)$. Besitzt folgende Gleichung*

$$\dot{f}(t_1) = \lambda \cdot \dot{g}(t_2) \tag{3.2}$$

eine eindeutige Lösung $\lambda \in \mathbb{R}$, so existiert eine von λ abhängige (reguläre) Reparametrisierung φ, so dass der Übergang unter dieser C^1-stetig ist.

Beweis:
Angenommen es existiert eine Lösung λ für $\dot{f}(t_1) = \lambda \cdot \dot{g}(t_2)$. Da nach Voraussetzung (Bemerkung 17) $\dot{f}(t_1) \neq 0$ und $\dot{g}(t_2) \neq 0$ gilt, folgt $\lambda \neq 0$. Durch Anwendung der Ableitungsformel (Lemma 16) erhält man

$$\dot{f}(t_1) = \lambda \cdot \dot{g}(t_2)$$

$$\sum_{j=1}^{n+1} \frac{m_f \cdot (d_j - d_{j-1})}{(t_{j+m_f} - t_j)} N_j^{m_f-1}(t_1|T_f) = \lambda \cdot \sum_{k=1}^{l+1} \frac{m_g \cdot (\tilde{d}_k - \tilde{d}_{k-1})}{\tilde{t}_{k+m_g} - \tilde{t}_k} N_k^{m_g-1}(t_2|T_g)$$

$$\sum_{j=1}^{n+1} \frac{m_f \cdot (d_j - d_{j-1})}{(t_{j+m_f} - t_j)} N_j^{m_f-1}(t_1|T_f) = \sum_{k=1}^{l+1} \frac{m_g \cdot (\tilde{d}_k - \tilde{d}_{k-1})}{\lambda^{-1} \cdot (\tilde{t}_{k+m_g} - \tilde{t}_k)} N_k^{m_g-1}(t_2|T_g)$$

$$\sum_{j=1}^{n+1} \frac{m_f \cdot (d_j - d_{j-1})}{(t_{j+m_f} - t_j)} N_j^{m_f-1}(t_1|T_f) = \sum_{k=1}^{l+1} \frac{m_g \cdot (\tilde{d}_k - \tilde{d}_{k-1})}{\tilde{t^*}_{k+m_g} - \tilde{t^*}_k} N_k^{m_g-1}(\lambda^{-1} \cdot t_2|\lambda^{-1} \cdot T_g),$$

wobei $\tilde{t^*}_j \in \lambda^{-1} \cdot T_g, \quad j \in \{1, \ldots, m_g + l + 1\}$.
Sei nun $\varphi_\lambda : \mathbb{R} \to \mathbb{R}$ eine Reparametrisierung mit $\varphi_\lambda(x) = \lambda^{-1} \cdot x$. Dann gilt

$$\dot{f}(t_1) = \dot{g}_{\varphi_\lambda}(t_2)$$

□

Während man bei einem C^0-stetigen Übergang eine Reparametrisierung τ finden kann, die die Parameterbereiche in entsprechender Form transliert, beschreibt die Reparametrisierung φ eine Umskalierung der Knoten für die Splines. Existiert solch eine Reparametrisierung ist der Übergang G^1-stetig, nach dessen Anwendung (also der Umskalierung der Knoten) ist er C^1-stetig.

Damit hat φ Auswirkungen auf die Länge der Vektoren, wie folgende Überlegung zeigt: Die Ableitung einer Splinekurve f ist gegeben durch

$$\dot{f}(t) = \sum_{j=1}^{n+1} \frac{m \cdot (d_j - d_{j-1})}{(t_{j+m} - t_j)} N_j^{m-1}(t|T).$$

Betrachtet man nun insbesondere den Nenner, also $(t_{j+m} - t_j)$, wird deutlich, welche Rolle die Reparametrisierung φ spielt. Eine Translation der Knotenfolge T ändert den Wert der Differenzen nicht. Da weder der Grad m noch das Kontrollpolygon d von einer Reparametrisierung beeinflusst werden, bleibt der gesamte Term identisch und damit der Ableitungsvektor gleich. Skaliert man jedoch die Knotenfolge mit einem Faktor λ um, verändert sich der Nenner und damit die Länge des Vektors. Möchte man nun weitere Ableitungen von g bilden, muss die Reparametrisierung φ in die Betrachtung miteinbezogen werden. Bei der Berechnung von $\dot{g}$ war dies nicht notwendig, da die Translation des Knotenvektors keine Auswirkung auf die Ableitungen hatte.

Beispiel 23. Seien zwei Splinekurven f, g mit

$$d_f = \begin{pmatrix} 0 & 5 & 8 \\ 0 & 20 & 20 \end{pmatrix} \text{ und } d_g = \begin{pmatrix} 8 & 11 & 16 \\ 20 & 20 & 0 \end{pmatrix}$$

$$T_f = \{0, 0, 0, 1, 1, 1\} \text{ und } T_g = \{1, 1, 1, 2, 2, 2\}$$

(siehe Abbildung 3.11). Zu den Knotenfolgen soll die G^1-Stetigkeit am Kurvenübergang geprüft werden. Im ersten Schritt werden die beiden Ableitungen $\dot{f}(1)$ und $\dot{g}(2)$ berechnet (siehe Abbildung 3.12).

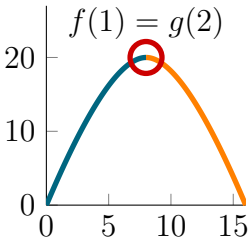


Abbildung 3.11: G^1-Stetigkeitstest

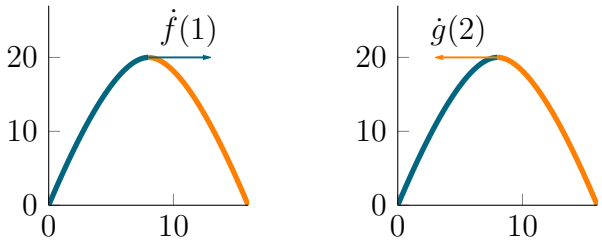


Abbildung 3.12: Ableitungen von f, g

Optisch kann man bereits erkennen, dass beide Kurven entgegengesetzt parametrisiert sind, da die Ableitungen zwar kollinear zueinander liegen, aber in verschiedene Richtungen zeigen. Dieses Problem tritt in der Praxis häufig auf und kann auf sehr einfache Weise anhand des Vorzeichens von λ erkannt werden. Gilt $\lambda > 0$, so sind die Kurven über ihren Übergang hinaus in dieselbe Richtung parametrisiert, ist $\lambda < 0$ sind sie entgegengesetzt.
In diesem Beispiel ergeben sich für die Ableitungen

$$\dot{f}(1) = \begin{pmatrix} 6 \\ 0 \end{pmatrix} \text{ und } \dot{g}(2) = \begin{pmatrix} -6 \\ 0 \end{pmatrix}$$

und damit das Gleichungssystem

$$\dot{f}(1) = \lambda \dot{g}(2)$$

$$\begin{pmatrix} 6 \\ 0 \end{pmatrix} = \lambda \cdot \begin{pmatrix} -6 \\ 0 \end{pmatrix}$$

$$\Rightarrow \lambda = -1$$

Die Reparametrisierung $\varphi_\lambda(t) = t \cdot \lambda^{-1}$ dreht also die Knotenfolge um und es ergibt sich $T_g = \{-1, -1, -1, -2, -2, -2\}$. Da die Knotenfolge einer Splinekurve laut Definition monoton steigend ist, muss dieses Ergebnis derart interpretiert werden, dass das Kontrollpolygon d_g gespiegelt wird. Anschaulich betrachtet ist die Kurve nun also „von hinten“ parametrisiert.

In der Praxis spielt die G^1-Stetigkeit solcher Übergänge eine bedeutende Rolle. Betrachtet man beispielsweise die Bewegung einer Fräsmaschine unter diesem Gesichtspunkt, kann man damit sicherstellen, dass die abgefahrene Fräsbahn keine Knicke aufweist und damit die Maschine die Bahn glatt abfahren kann. Durch die Berechnung einer von λ abhängigen Reparametrisierung ist es darüber hinaus möglich, Aussagen über die Veränderung der ersten Ableitung zu treffen. Hierzu wird das Konzept der G^2-Stetigkeit eingeführt, das in diesem Zusammenhang C^1-Stetigkeit voraussetzt.

Definition 24. Seien f, g zwei Splinekurven mit einem C^1-stetigen Übergang t. Die Kurven stoßen in diesem Punkt $G^2-stetig$ aneinander, wenn eine Funktion $\nu(t)$ existiert, so dass gilt

$$\ddot{f}(t) - \ddot{g}(t) = \nu(t) \cdot \dot{f}(t)$$

Mit anderen Worten ist ein C^1-stetiger Übergang G^2-stetig, wenn die Differenz der zweiten Ableitungen an diesem Punkt parallel zur ersten Ableitung ist (siehe Abbildung 3.13).

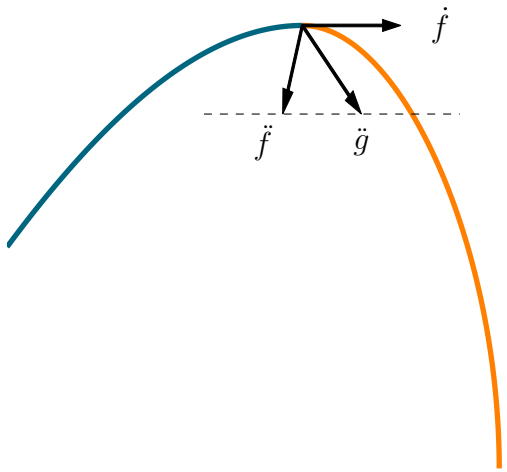
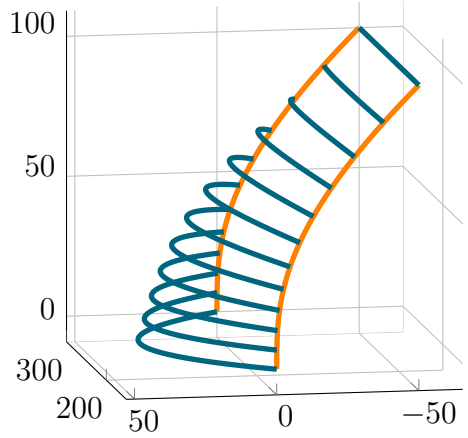


Abbildung 3.13: G^2-Stetigkeit

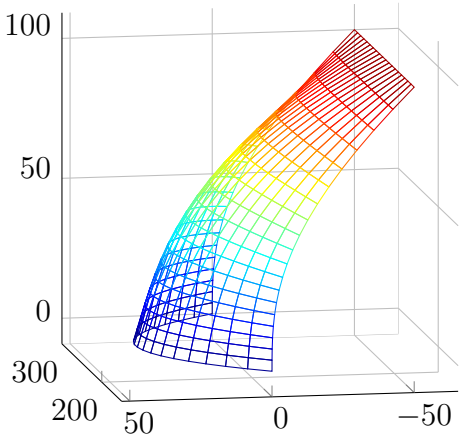
Weitere ausführliche Beschreibungen der hier behandelten Stetigkeiten können in [3] nachgeschlagen werden. Darüber hinaus sind in [7] weitere allgemeine Ansätze und Beispiele für G^k/C^k-Stetigkeiten nachzulesen.

3.1.3 Tensorprodukt-B-Spline-Flächen

Das Konzept der Splinekurven lässt sich nun auf Flächen erweitern. Anschaulich kann man sich vorstellen, dass eine Splinekurve (Abbildung 3.14(a) blau) entlang einer zweiten Splinekurve (Abbildung 3.14(a) orange) durch den Raum bewegt wird und sich dabei verändern kann.



(a) B-Splineflächenerzeugung



(b) B-Spline-Fläche

Abbildung 3.14: Anschauliche Betrachtung von B-Spline-Flächen

Hierzu benötigt man zunächst eine Initialkurve

$$S_m \cdot d(u) = \sum_{j=0}^{m} d_j \cdot N_j^m(u|T_1)$$

mit konstantem Grad m. Nun sei jedes d_j für sich wieder eine Kurve der Form

$$d_j = d_j(v) \sum_{k=1}^{n} d_{j,k} N_k^n(v|T_2)$$

Setzt man nun die beiden obigen Gleichung zusammen, ist es möglich eine Fläche zu charakterisieren.

Definition 25. Eine Fläche $F : \mathbb{R}^2 \to \mathbb{R}^3$ (im Folgenden auch *Patch* genannt) ist definiert durch das Tensorprodukt von B-Splines der Form

$$F(u,v) = \sum_{j=1}^{m} \sum_{k=1}^{n} d_{j,k} \cdot N_j^m(u|T_1) \cdot N_k^n(v|T_2).$$

Bemerkung 26. Für Flächen $f(u,v)$ werden ähnlich wie für Kurven folgende Einschränkungen getroffen:

1. F ist mindestens zwei mal stetig differenzierbar.
2. Für die partiellen Ableitung gilt Rang $\left[\frac{\partial}{\partial u}F, \frac{\partial}{\partial v}F\right] = 2$.

Im nächsten Beispiel wird eine B-Spline-Fläche mit zwei Splinekurven erzeugt, um das Prinzip des Tensorprodukts zu veranschaulichen.

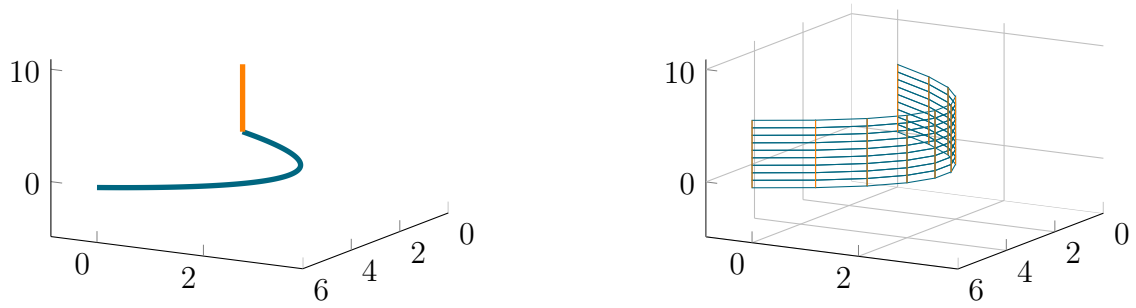
Beispiel 27. Sei S_m die initiale Splinekurve vom Grad $m = 3$ mit Kontrollpolygon

$$d_1 = \begin{pmatrix} 0 & 2 & 4 & 6 \\ 0 & 3 & 3 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Die Knotenfolge sei gegeben durch $T_1 = \{0,0,0,0,1,1,1,1\}$. Die zweite Splinekurve soll nun die Initialkurve S_m entlang der z-Achse ausbreiten. Dazu wählt das Kontrollpolygon der zweiten Splinekurve als

$$d_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 2 & 4 & 6 \end{pmatrix}.$$

Die Knotenfolge sei ebenfalls $T_2 = \{0,0,0,0,1,1,1,1\}$. In Abbildung 3.15(a) sieht man die Initialkurve (blau) und die Ausbreitungskurve (orange). Es ist ersichtlich, dass es keine Rolle spielt, welche der beiden Kurven als Bezugspunkt verwendet wird, es macht also keinen Unterschied mit welcher der beiden Kurven begonnen wird. In Abbildung 3.15(b) ist die resultierende Fläche abgebildet.



(a) Splinekurven einer Fläche

(b) B-Spline-Fläche

Abbildung 3.15: Splinefläche entlang zweier Splinekurven

Um die Tangentialebene an einem Punkt (u, v) einer Fläche F zu berechnen, benötigt man die partiellen Ableitungen in u- und v-Richtung. Hierzu verwendet man die Ableitungsformel für Splines aus Lemma 16 und setzt sie in $F(u, v)$ ein:

Lemma 28. Die partielle Ableitung in u-Richtung ist ergibt sich als

$$
\begin{aligned}
&\frac{\partial}{\partial u} F(u,v) \\
&= \sum_{j=1}^{n_1} \sum_{k=1}^{n_2} d_{j,k} \left(\frac{m_1}{t_{1,j+m_1} - t_{1,j}} N_j^{m_1-1}(u|T) - \frac{m_1}{t_{1,j+m_1+1} - t_{1,j+1}} N_{j+1}^{m_1-1}(u|T_1) \right) N_k^{m_2}(v|T_2) \\
&= \sum_{j=1}^{n_1} \sum_{k=1}^{n_2} d_{j,k} \frac{m_1}{t_{1,j+m_1} - t_{1,j}} N_j^{m_1-1}(u|T_1) N_k^{m_2}(v|T_2) \\
&\quad - \sum_{j=1}^{n_1} \sum_{k=1}^{n_2} d_{j,k} \frac{m_1}{t_{1,j+m_1+1} - t_{1,j+1}} N_{j+1}^{m_1-1}(u|T_1) N_k^{m_2}(v|T_2) \\
&= \sum_{j=1}^{n_1} \sum_{k=1}^{n_2} d_{j,k} \frac{m_1}{t_{1,j+m_1} - t_{1,j}} N_j^{m_1-1}(u|T_1) \\
&\quad - \sum_{j=2}^{n_1+1} \sum_{k=1}^{n_2} d_{j-1,k} \frac{m_1}{t_{1,j+m_1} - t_{1,j}} N_j^{m_1-1}(u|T_1) N_k^{m_2}(v|T_2) \\
&= \sum_{j=1}^{n_1+1} \sum_{k=1}^{n_2} \frac{(d_{j,k} - d_{j-1,k}) \cdot m_1}{t_{1,j+m_1} - t_{1,j}} N_j^{m_1-1}(u|T_1) N_k^{m_2}(v|T_2),
\end{aligned}
$$

wobei $d_{0,x} = 0$ gilt.

Bemerkung 29. Durch analoge Rechnung kann man zeigen, dass sich für die partielle Ableitung in v-Richtung ergibt:

$$
\frac{\partial}{\partial v} F(u,v) = \sum_{j=1}^{n_1} \sum_{k=1}^{n_2+1} \frac{(d_{j,k} - d_{j,k-1}) \cdot m_2}{t_{2,k+m_2} + t_{2,k}} N_k^{m_2-1}(v|T_2) N_j^{m_1}(u|T_1),
$$

wobei $d_{x,0} = 0$ gilt.

Definition 30. Seien an einer Stelle (u, v) im Parameterbereich einer Fläche F die partiellen Ableitungen erster Ordnung gegeben durch $\frac{\partial}{\partial u}F(u,v) = (\alpha_x, \alpha_y, \alpha_z)^T$ und $\frac{\partial}{\partial v}F(u,v) = (\beta_x, \beta_y, \beta_z)^T$. Dann heißt

$$\mathbf{J_F}(u,v) = \begin{pmatrix} \alpha_x & \beta_x \\ \alpha_y & \beta_y \\ \alpha_z & \beta_z \end{pmatrix}$$

die *Jacobi-Matrix* der Fläche F im Punkt (u, v).

Durch die mehrfache Anwendung der Ableitungsformel für B-Splines ist es nun möglich auch die partiellen Ableitungen zweiter Ordnung zu bestimmen.

Lemma 31. Die gemischte partielle Ableitung $\frac{\partial^2}{\partial u \partial v}F(u,v)$ ergibt sich durch nacheinander angewendete partielle Differentiation:

$$\begin{aligned}
&\frac{\partial^2}{\partial u \partial v}F(u,v) \\
&= \frac{\partial}{\partial v}\left(\sum_{j=1}^{n_1+1}\sum_{k=1}^{n_2}\frac{(d_{j,k}-d_{j-1,k})\cdot m_1}{t_{1,j+m_1}-t_{1,j}}N_j^{m_1-1}(u|T_1)N_k^{m_2}(v|T_2)\right) \\
&= \sum_{j=1}^{n_1+1}\sum_{k=1}^{n_2}\frac{(d_{j,k}-d_{j-1,k})\cdot m_1}{t_{1,j+m_1}-t_{1,j}}N_j^{m_1-1}(u|T_1)\left(\frac{m_2}{t_{2,k+m_2}-t_{2,k}}N_k^{m_2-1}(v|T_2)\right. \\
&\quad \left.-\frac{m_2}{t_{2,k+m_2+1}-t_{2,k+1}}N_{k+1}^{m_2-1}(v|T_2)\right) \\
&= \sum_{j=1}^{n_1+1}\sum_{k=1}^{n_2}\frac{(d_{j,k}-d_{j-1,k})\cdot m_1}{t_{1,j+m_1}-t_{1,j}}N_j^{m_1-1}(u|T_1)\frac{m_2}{t_{2,k+m_2}-t_{2,k}}N_k^{m_2-1}(v|T_2) \\
&\quad -\sum_{j=1}^{n_1+1}\sum_{k=1}^{n_2}\frac{(d_{j,k}-d_{j-1,k})\cdot m_1}{t_{1,j+m_1}-t_{1,j}}N_j^{m_1-1}(u|T_1)\frac{m_2}{t_{2,k+m_2+1}-t_{2,k+1}}N_{k+1}^{m_2-1}(v|T_2) \\
&= \sum_{j=1}^{n_1+1}\sum_{k=1}^{n_2}\frac{(d_{j,k}-d_{j-1,k})\cdot m_1}{t_{1,j+m_1}-t_{1,j}}N_j^{m_1-1}(u|T_1)\frac{m_2}{t_{2,k+m_2}-t_{2,k}}N_k^{m_2-1}(v|T_2) \\
&\quad -\sum_{j=1}^{n_1+1}\sum_{k=2}^{n_2+1}\frac{(d_{j,k-1}-d_{j-1,k-1})\cdot m_1}{t_{1,j+m_1}-t_{1,j}}N_j^{m_1-1}(u|T_1)\frac{m_2}{t_{2,k+m_2}-t_{2,k}}N_k^{m_2-1}(v|T_2) \\
&= \sum_{j=1}^{n_1+1}\sum_{k=1}^{n_2+1}\frac{m_1\cdot m_2\cdot(d_{j,k}-d_{j-1,k}-d_{j,k-1}+d_{j-1,k-1})}{(t_{1,j+m_1}-t_{1,j})\cdot(t_{2,k+m_2}-t_{2,k})}N_j^{m_1-1}(u|T_1)N_k^{m_2-1}(v|T_2),
\end{aligned}$$

wobei $d_{0,x} = d_{x,0} = 0$ gilt.

Bemerkung 32. Durch analoge Berechnung ergibt sich für die gemischte partielle Ableitung

$$\begin{aligned}\frac{\partial^2}{\partial v \partial u} F(u,v) &= \\ &= \sum_{j=1}^{n_1+1} \sum_{k=1}^{n_2+1} \frac{m_1 \cdot m_2 \cdot (d_{j,k} - d_{j-1,k} - d_{j,k-1} + d_{j-1,k-1})}{(t_{1,j+m_1} - t_{1,j}) \cdot (t_{2,k+m_2} - t_{2,k})} N_j^{m_1-1}(u|T_1) N_k^{m_2-1}(v|T_2),\end{aligned}$$

wobei $d_{0,x} = d_{x,0} = 0$ gilt.
Die Reihenfolge der partiellen Ableitungen spielen also keine Rolle, es gilt

$$\frac{\partial^2}{\partial u \partial v} F(u,v) = \frac{\partial^2}{\partial v \partial u} F(u,v).$$

Für die Bestimmung eines Flächenübergangs werden für diese Arbeit noch die partiellen Ableitungen $\frac{\partial^2}{\partial u^2} F(u,v)$ und $\frac{\partial^2}{\partial v^2} F(u,v)$ benötigt:

Lemma 33. Die partielle Ableitung zweiter Ordnung $\frac{\partial^2}{\partial u^2} F(u,v)$ ergibt sich als

$$\begin{aligned}\frac{\partial^2}{\partial u^2} F(u,v) &= \\ &= \frac{\partial}{\partial u} \sum_{j=1}^{n_1+1} \sum_{k=1}^{n_2} \frac{(d_{j,k} - d_{j-1,k}) \cdot m_1}{t_{1,j+m_1} - t_{1,j}} N_j^{m_1-1}(u|T_1) N_k^{m_2}(v|T_2) \\ &= \sum_{j=1}^{n_1+1} \sum_{k=1}^{n_2} \frac{(d_{j,k} - d_{j-1,k}) m_1}{t_{1,j+m_1} - t_{1,j}} \left(\frac{m_1 - 1}{t_{1,j+m_1-1} - t_{1,j}} N_j^{m_2-2}(u|T_1) \right. \\ &\quad \left. - \frac{m_1 - 1}{t_{1,j+m_1-1+1} - t_{1,j+1}} N_{j+1}^{m_2-2}(u|T_1) \right) N_k^{m_2}(v|T_2) \\ &= \sum_{j=1}^{n_1+1} \sum_{k=1}^{n_2} \frac{(d_{j,k} - d_{j-1,k}) m_1}{t_{1,j+m_1} - t_{1,j}} \frac{m_1 - 1}{t_{1,j+m_1-1} - t_{1,j}} N_j^{m_2-2}(u|T_1) N_k^{m_2}(v|T_2) \\ &\quad - \sum_{j=1}^{n_1+1} \sum_{k=1}^{n_2} \frac{(d_{j,k} - d_{j-1,k}) m_1}{t_{1,j+m_1} - t_{1,j}} \frac{m_1 - 1}{t_{1,j+m_1-1+1} - t_{1,j+1}} N_{j+1}^{m_2-2}(u|T_1) N_k^{m_2}(v|T_2) \\ &= \sum_{j=1}^{n_1+1} \sum_{k=1}^{n_2} \frac{(d_{j,k} - d_{j-1,k}) m_1}{t_{1,j+m_1} - t_{1,j}} \frac{m_1 - 1}{t_{1,j+m_1-1} - t_{1,j}} N_j^{m_2-2}(u|T_1) N_k^{m_2}(v|T_2) \\ &\quad - \sum_{j=2}^{n_1+2} \sum_{k=1}^{n_2} \frac{(d_{j-1,k} - d_{j-2,k}) m_1}{t_{1,j+m_1-1} - t_{1,j-1}} \frac{m_1 - 1}{t_{1,j+m_1-1} - t_{1,j}} N_j^{m_2-2}(u|T_1) N_k^{m_2}(v|T_2) \\ &= \sum_{j=1}^{n_1+2} \sum_{k=1}^{n_2} \left(\frac{d_{j,k} - d_{j-1,k}}{t_{1,j+m_1} - t_{1,j}} - \frac{d_{j-1,k} - d_{j-2,k}}{t_{1,j+m_1-1} - t_{1,j-1}} \right) \frac{m_1 \cdot (m_1 - 1)}{t_{1,j+m_1-1} - t_{1,j}} N_j^{m_2-2}(u|T_1) N_k^{m_2}(v|T_2),\end{aligned}$$

wobei $d_{-1,x} = d_{0,x} = 0$ gilt.

Bemerkung 34. Analog ergibt sich für die partielle Ableitung $\frac{\partial^2}{\partial v^2}F$

$$\frac{\partial^2}{\partial v^2}F(u,v) = \\ = \sum_{j=1}^{n_1}\sum_{k=1}^{n_2+2}\left(\frac{d_{j,k}-d_{j,k-1}}{t_{2,j+m_1}-t_{2,j}} - \frac{d_{j,k-1}-d_{j,k-2}}{t_{2,j+m_1-1}-t_{2,j-1}}\right)\frac{m_1\cdot(m_1-1)}{t_{2,j+m_1-1}-t_{2,j}}N_j^{m_2}(u|T_1)N_k^{m_2-2}(v|T_2),$$

wobei $d_{x,0} = d_{x,-1} = 0$ gilt.

Definition 35. Seien die partiellen Ableitung erster und zweiter Ordnung einer Fläche F an der Stelle (u,v) gegeben. Dann heißt

$$\mathbf{H_F}(u,v) = \begin{pmatrix} \frac{\partial^2}{\partial u^2}F(u,v) & \frac{\partial^2}{\partial u\partial v}F(u,v) \\ \frac{\partial^2}{\partial v\partial u}F(u,v) & \frac{\partial^2}{\partial v^2}F(u,v) \end{pmatrix}$$

die *Hesse-Matrix* der Fläche F im Punkt (u,v).

Ähnlich wie bei den Splinekurven spielt die Parametrisierung der B-Spline-Flächen eine entscheidende Rolle bei der Berechnung der verschiedenen Ableitungen. Anstatt nur einer Ableitung bei Splines, werden bei einem Patch zwei (partielle) Ableitungen erster Ordnung berechnet. Während im ersten Fall die Ableitung eine Richtungsableitung entlang der Kurve selbst darstellt, wird bei Flächen entlang der u- und v-Richtung abgeleitet. Die beiden resultierenden Vektoren spannen damit eine Tangentialebene auf. In diesem Zusammenhang ist die Wahl der Knotenfolgen $T_{F,u}, T_{F,v}, T_{G,u}, T_{G,v}$ aber nicht der einzige beeinflussende Faktor der Ableitungsvektoren, wie nachfolgendes Beispiel veranschaulicht.

Beispiel 36. Seien zwei Patches gegeben, die in derselben Ebene liegen und eine gemeinsame Kante besitzen (vgl. Abbildung 3.16). Weiter gelte $T_{F,u} = T_{G,u}$ und $T_{F,v} = T_{G,v}$.

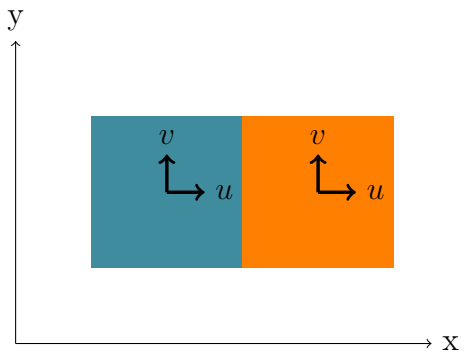


Abbildung 3.16: Zwei Patches mit identischer Parametrisierung

Leitet man die beiden Patches an einem beliebigen Punkt der Kante ab, stimmen die

Richtungen der partiellen Ableitungen überein (diese sind in diesem speziellen Beispiel genau die Richtungen der Parametrisierung u, v). Die Länge der Ableitungen erster Ordnung hängen wie zuvor bei den Splinekurven gesehen von der Wahl des Knotenvektors ab. Da $T_F = T_G$ gilt, stimmen also auch deren Längen überein. Betrachtet man nun selbige Flächen mit der in Abbildung 3.17 zu sehenden Parametrisierung für gleichbleibende Knotenvektoren stimmen die Richtungen offensichtlich nicht mehr überein.

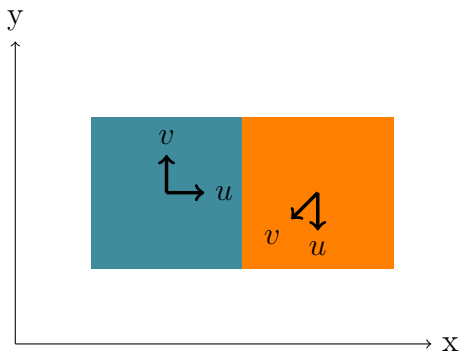


Abbildung 3.17: Zwei Patches mit unterschiedlicher Parametrisierung

Bei der Ableitung von Flächen muss also auch die Richtung der Parametrisierung in der Betrachtung berücksichtigt werden.

Wie bereits an diesem sehr einfach konstruierten Beispiel gesehen, ist die Aufgabe einen Stetigkeitsbegriff für Flächen einzuführen komplexer und aufwendiger als für Splinekurven. Das nächste Kapitel behandelt den Übergang zweier Patches und ein numerisches Verfahren zur Analyse sowie die damit verbundene Einordnung in G/C-stetig.

4 Flächenübergänge

Im vorherigen Kapitel wurden Splinekurven eingeführt und die Stetigkeit zweier angrenzender Splines untersucht. Im letzten Abschnitt wurden B-Spline-Flächen definiert und deren partielle Ableitungen bestimmt. Mit diesem Hintergrundwissen ist es nun möglich (ähnlich wie bei den Kurven zuvor) den Übergang zweier Flächen zu charakterisieren. In der Praxis trifft man oftmals auf folgende Fragestellungen:

1. Wie nahe liegen die beiden Flächen aneinander?
2. Die Flächen berühren sich in einem Punkt - Haben sie auch die gleiche Tangentialebene?
3. Wie glatt ist der Übergang in einer bestimmten Richtung?

In Abbildung 4.1 ist ein Flächenübergang, der einen Knick aufweist, zu sehen. Während die blaue Fräsbahn jedes Mal einen Knick abfährt, ist die orange Bahn „glatt“.

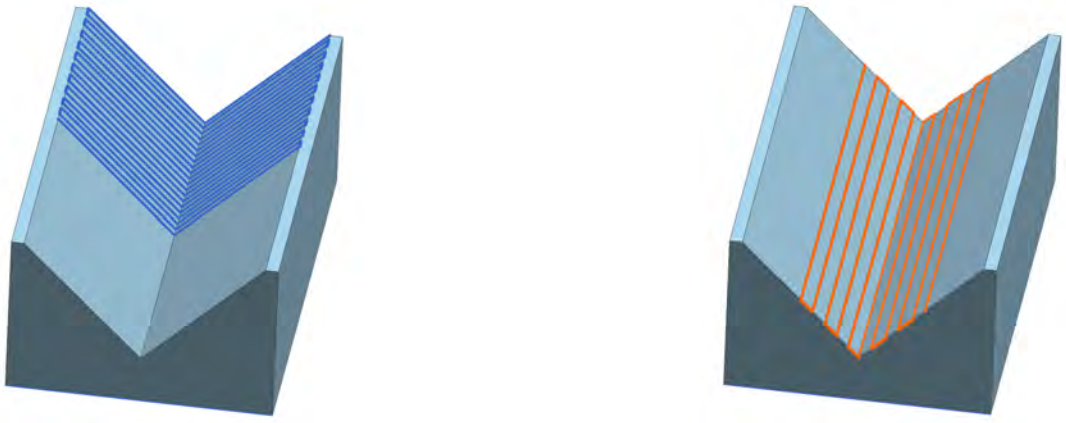


Abbildung 4.1: Verschiedene Frässtrategien zweier Flächenübergänge

Um diese und weitere Fragen zu beantworten, wird das Konzept der G/C-Stetigkeit im Zusammenhang mit Flächenübergängen erklärt und ein Verfahren angegeben, welches eine eindeutige Klassifizierung ermöglicht. Zusätzliche Informationen zum Thema stetige Flächenübergänge können in [3] sowie in [7] nachgeschlagen werden. Für eine ausführliche Beschreibung der in diesem Kapitel verwendeten numerischen Verfahren bietet sich die Literatur von [11] sowie [6] als Nachschlagewerk an.

Definition 37. Seien F, G zwei B-Spline-Flächen. Die beiden Patches besitzen einen Flächenübergang, falls $(u, v) \in T_{F,u} \times T_{F,v}$ und $(\tilde{u}, \tilde{v}) \in T_{G,u} \times T_{G,v}$ existieren, so dass

$$F(u, v) = G(\tilde{u}, \tilde{v})$$

gilt. Der Übergang heißt dann C^0-stetig.

In der praktischen CAD-Konstruktion ist es in vielen Fällen nicht möglich, einen exakten C^0-stetigen Übergang entlang einer Kante zu konstruieren. Ein solcher Problemfall tritt beispielsweise auf, wenn versucht wird, eine polynomiale B-Spline-Fläche an die Oberfläche eines Zylinders zu setzen (vgl. Abbildung 4.2). Während die B-Spline-Fläche mit Polynomen den Kreisbogen (orange Kante) nur näherungsweise approximiert, ist die Anschlussstelle des aufgetrennten Zylinders (blaue Kante) ein exakter Kreisbogen. Es folgt, dass das Zusammensetzen der beiden Flächen entlang einer gemeinsamen Kante nur innerhalb einer gewissen Toleranzgrenze möglich ist.

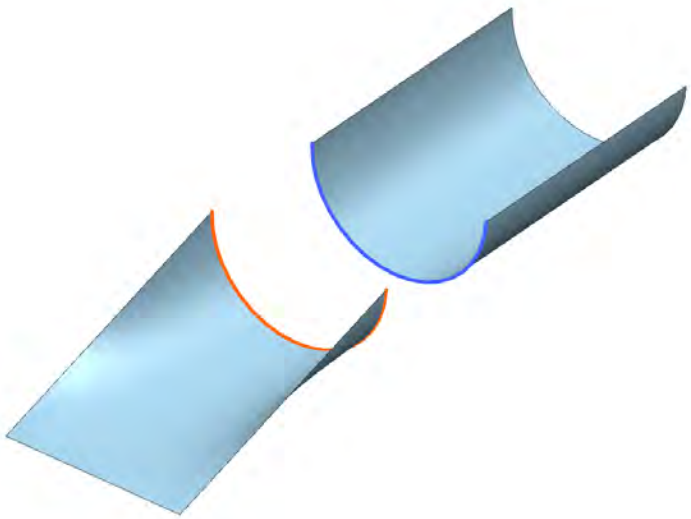


Abbildung 4.2: Beispiel eines nicht exakten C^0-stetigen Übergangs

Da in der Praxis eingesetzte Werkzeugmaschinen (z.B. eine Fräsmaschine mit Fräserradius 1mm) selbst eine Ungenauigkeit innerhalb einer Toleranz implizieren, macht es Sinn einen solchen Übergang trotzdem als C^0-stetig zu klassifizieren. Es ist eine gewisse Robustheit des CAD-Systems gefordert, welche leider nicht immer erfüllt wird und teilweise stark vom verwendeten CAD-System abhängt. Dieses einfache Beispiel zeigt bereits die Komplexität der Problemstellung und die Notwendigkeit, die numerischen Resultate zu verifizeren und im Kontext richtig einzuordnen. Zur Prüfung dieser Sachverhalte werden mehrere Schritte benötigt. Ausganspunkt des Verfahrens ist ein aus der MRA gewonnener Punkt, der als Indikator für einen Flächenübergang fungiert. Im ersten Schritt müssen die beiden Flächen, deren Übergang untersucht werden soll, identifiziert werden. Anschließend muss geprüft werden, ob es sich dabei um einen C^0-stetigen Übergang (innerhalb einer vorgegebenen Toleranzgrenze $\epsilon > 0$) handelt. Um in weiterführenden Schritten die berechneten Fräsbahnen optimieren zu können, wird anschließend ein C^1- / G^2-Stetigkeitstest durchgeführt. Für die praktische Umsetzung in NX wurde ein Verfahren entwickelt, das im Folgenden dargestellt wird.

4.1 Prüfung auf C^0-Stetigkeit

Im ersten Schritt müssen zu einem gegebenen Punkt der MRA zwei Flächen sowie ein Übergang assoziiert werden, der in späteren Schritten genauer untersucht wird. Hierzu werden zu allen Flächen des Werkstücks die minimalen Abstände zu einem vorgegebenen Punkt der MRA berechnet und gespeichert (vgl. UF_ MODL_ ask_ minimum_ dist Kapitel 2). Anschließend werden die zwei Flächen mit den kleinsten minimalen Abständen selektiert und man erhält die zwei Patches F und G, sowie die daraufliegenden Punkte p_1, p_2 (vgl. Abbildung 4.3). Mit anderen Worten findet man die zwei nächstgelegenen Flächen zu einem Punkt der MRA.

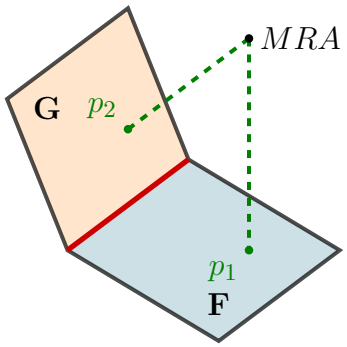


Abbildung 4.3: C^0-Stetigkeitstest Teil I

Im nächsten Schritt wird der Punkt $\tilde{p}_1$ mithilfe der minimalen Distanz von p_1 zur gegenüberliegenden Fläche G berechnet (vgl. Abbildung 4.4).

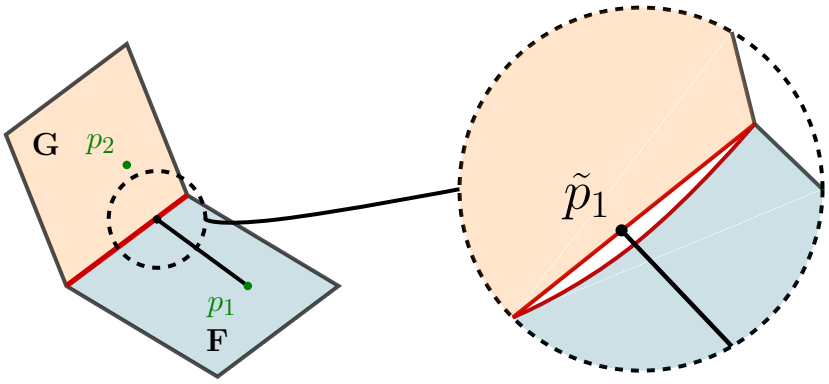


Abbildung 4.4: C^0-Stetigkeitstest Teil II

Diese Vorgehensweise wird für $\tilde{p}_1$ und F wiederholt und man erhält den ersten Untersuchungspunkt x_1 (vgl. Abbildung 4.5).

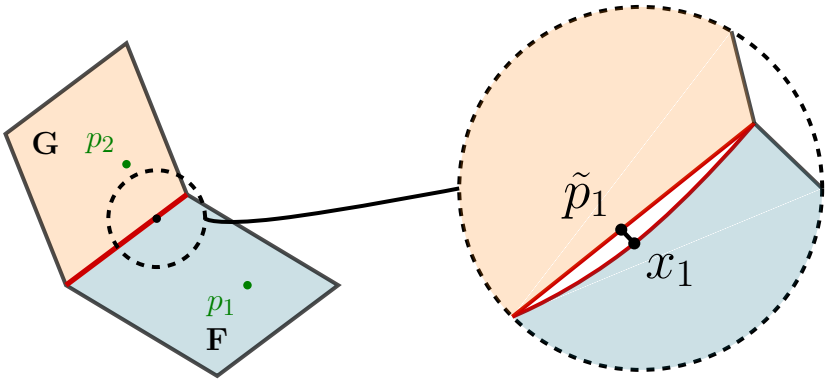


Abbildung 4.5: C^0-Stetigkeitstest Teil III

Analog wird mit dem Punkt p_2 auf der Fläche G verfahren und man erhält den zweiten Untersuchungspunkt x_2, wie in Abbildung 4.6 zu sehen ist.

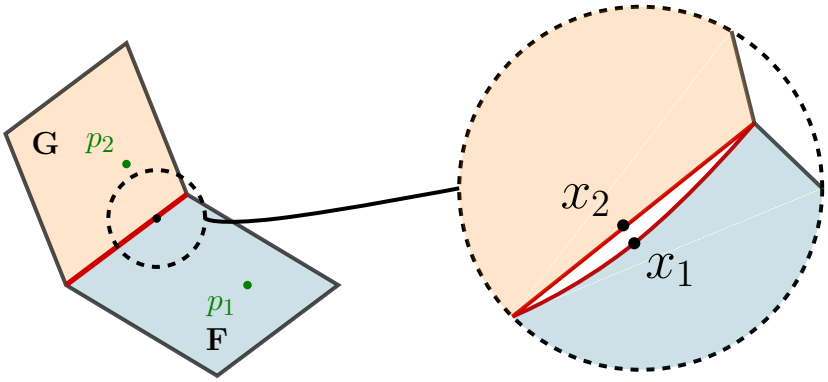


Abbildung 4.6: C^0-Stetigkeitstest Endresultat

Im letzten Schritt wird der Abstand von x_1 und x_2 mithilfe der euklidischen Norm berechnet und folgende Ungleichung geprüft:

$$\|x_1 - x_2\|_2 < \epsilon$$

Unterscheidet sich die Lage der beiden Punkte im $\mathbb{R}^3$ maximal um einen Schwellwert ϵ, so wird der Übergang als C^0-stetig klassifiziert.

Dieses Verfahren realisiert den Rückschritt/die Rücktransformation von einer erzeugten Fräsbahn zur Geometrie im CAM-System. Mit dem identifizierten Flächenübergang lassen sich mithilfe numerischer Verfahren zur Prüfung auf G^1/G^2-Stetigkeit nun weitere Aussagen über den optimalen Verlauf der Fräsbahn treffen.

4.2 Prüfung auf C^1-Stetigkeit

Definition 38. Seien F, G zwei B-Spline-Flächen mit einem C^0-stetigen Übergang. Dieser heißt G^1-stetig, falls eine reguläre Reparametrisierung φ existiert, so dass für die Jacobi-Matrizen gilt

$$\mathbf{J_F}(u, v) = \mathbf{J_{G_\varphi}}(\tilde{u}, \tilde{v}).$$

Falls die Abbildung φ die Identität ist, so heißt der Flächenübergang C^1*-stetig.*

Satz 39. *Seien F und G zwei Patches mit einem C^0-stetigen Flächenübergang $F(u^*, v^*) = G(\tilde{u}^*, \tilde{v}^*)$. Existiert eine invertierbare Lösung $\mathbf{X} \in \mathbb{R}^{2\times 2}$ zur Gleichung*

$$\mathbf{J_F}(u^*, v^*) \cdot \mathbf{X} = \mathbf{J_G}(\tilde{u}^*, \tilde{v}^*),$$

so gibt es eine von $\mathbf{X}$ abhängige affine Reparametrisierung $\varphi : (u, v) \mapsto \mathbf{X} \cdot \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$ so dass der Flächenübergang nach Anwendung von φ C^1-stetig ist.

Beweis: Angenommen es existiert eine Lösung $\mathbf{X}$ für $\mathbf{J_F}(u^*, v^*) \cdot \mathbf{X} = \mathbf{J_G}(\tilde{u}^*, \tilde{v}^*)$. Sei $\varphi : (\tilde{u}, \tilde{v}) \mapsto \mathbf{X} \cdot \begin{pmatrix} \tilde{u} \\ \tilde{v} \end{pmatrix} + \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$ die Reparametrisierung der Fläche G. Damit ergibt sich für die Jacobi-Matrix der reparametrisierten Fläche G

$$\mathbf{J_{G\circ\varphi}}(\tilde{u}^*, \tilde{v}^*) = \mathbf{J_G}(\mathbf{X} \cdot \begin{pmatrix} \tilde{u}^* \\ \tilde{v}^* \end{pmatrix} + \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}) \cdot \mathbf{J}_\varphi(\tilde{u}, \tilde{v})$$

In obige Gleichung eingesetzt ergibt sich

$$\mathbf{J_F}(u^*, v^*) \cdot \mathbf{X} = \mathbf{J_G}(\mathbf{X} \cdot \begin{pmatrix} \tilde{u}^* \\ \tilde{v}^* \end{pmatrix} + \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}) \cdot \mathbf{X}.$$

Durch die Invertierbarkeit von $\mathbf{X}$ folgt nun

$$\mathbf{J_F}(u^*, v^*) = \mathbf{J_G}(\mathbf{X} \cdot \begin{pmatrix} \tilde{u}^* \\ \tilde{v}^* \end{pmatrix} + \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}).$$

Die zwei Parameter w_1, w_2 sind für die Translation des Paares $(\tilde{u}^*, \tilde{v}^*)$ verantwortlich. Sie können wie folgt berechnet werden:

$$\begin{pmatrix} u^* \\ v^* \end{pmatrix} = \mathbf{X} \cdot \begin{pmatrix} \tilde{u}^* \\ \tilde{v}^* \end{pmatrix} + \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$$

$$\begin{pmatrix} u^* \\ v^* \end{pmatrix} - \begin{pmatrix} x_1 & x_2 \\ x_3 & x_4 \end{pmatrix} \cdot \begin{pmatrix} \tilde{u}^* \\ \tilde{v}^* \end{pmatrix} = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$$

$$\Rightarrow w_1 = u^* - x_1 \cdot \tilde{u}^* - x_2 \cdot \tilde{v}^*$$

$$\Rightarrow w_2 = v^* - x_3 \cdot \tilde{u}^* - x_4 \cdot \tilde{v}^*$$

□

Die Grundidee zur Bestimmung der Reparametrisierung φ für Flächenübergänge ist also ähnlich zu dem Verfahren für Splinekurven. Die zu lösende Gleichung ist nun jedoch von zwei Variablen u und v abhängig und die Lösung selbst ist eine 2×2-Matrix anstatt eines Skalars. Diese Problemstellung kann der Klasse der linearen Ausgleichsprobleme zugeordnet werden. Der Hauptbestandteil der praktischen Umsetzung zur Überprüfung eines C^1-stetigen Übergangs ist also die Lösung und Verifikation dieser Gleichung.

Im ersten Schritt werden die Jacobi-Matrizen $\mathbf{J_F}$ und $\mathbf{J_G}$ an den jeweiligen Stellen im Parameterbereich berechnet. Die Realisierung erfolgt durch die Implementierung der in Kapitel 2 vorgestellten Ableitungsformeln für B-Spline-Flächen. Anschließend müssen die beiden resultierenden Matrizen auf maximalen Rang geprüft werden. Dies ist notwendig, um Singularitäten, an denen eine Ableitung $(0,0,0)^T$ wird, auszuschließen.

Die Hauptaufgabe besteht nun darin, die Gleichung

$$\mathbf{J_F}(u,v) \cdot \mathbf{X} = \mathbf{J_G}(\tilde{u},\tilde{v})$$

für gegebene Matrizen $\mathbf{J_F}(u,v)$ und $\mathbf{J_G}(\tilde{u},\tilde{v})$ zu lösen. Existiert ein gültiges $\mathbf{X}$, dann gibt es (lokal) die gesuchte Reparametrisierung. Um diese Lösung $\mathbf{X}$ (und die Reparametrisierung φ) zu berechnen und diese bewerten zu können, wird im Folgenden das Konzept der linearen Ausgleichsrechnung eingeführt. Die Rolle der beiden Flächen F, G ist nicht symmetrisch, falls eine der beiden Flächen eine Singularität aufweist (siehe hierzu Erklärung Seite 74).

Das Lineare Ausgleichsproblem (least squares)

Problemstellung

Gegeben sind m Messpunkte bestehend aus Zeitpunkten t_i und Messwerten b_i. Der Abhängigkeit von t und b soll ein Zusammenhang zugrunde liegen, so dass sie in einer Funktion φ wie folgt darzustellen ist:

$$b(t) = \varphi(t; x_1, \ldots, x_n)$$

wobei x_i unbekannte Parameter sind. Da die Messungen i.A. fehlerbehaftet sind, definieren wir die jeweiligen Abweichungen als

$$\Delta_i = b_i - \varphi(t_i; x_1, \ldots, x_n) \text{ für } i = 1, \ldots, m.$$

Die gesuchte Lösung erhält man nun durch die Minimierung der quadratischen Abweichungen Δ_i^2. Mit anderen Worten sind diejenigen Parameter $x_1, \ldots, x_n$ gesucht, für die

$$\sum_{i=1}^{m} \Delta_i^2 = \min$$

gilt. Ist φ linear, d.h.

$$\varphi(t; x_1, \ldots . x_n) = a_1(t)x_1 + \ldots + a_n(t)x_n,$$

wobei $a_1, \ldots, a_n$ beliebige Funktionen von $\mathbb{R} \to \mathbb{R}$ sind, kann obige Problemstellung folgendermaßen formuliert werden:
Gesucht sind Parameter $x_1, \ldots, x_n \in \mathbb{R}^n$, so dass

$$\|\mathbf{A} \cdot x - b\|_2 = \min_x,$$

wobei $b = (b_1, \ldots, b_m)^T, x = (x_1, \ldots, x_n)^T$ und $\mathbf{A} \in \mathbb{R}^{m \times n}$ mit $a_{ij} = a_j(t_i)$.
Hier wird nur der Fall eines überbestimmten Gleichungssystems (also $m \geq n$) betrachtet. Das macht an dieser Stelle Sinn, denn anschaulich bedeutet diese Annahme, dass mehr Daten vorhanden sind, als Parameter bestimmt werden müssen, was in der Praxis keine Seltenheit darstellt.

Satz 40. *Die Lösung des linearen Ausgleichsproblems ist äquivalent zur Lösung der Normalengleichung:*

$$\min_x(\|\mathbf{A} \cdot x - b\|) \Leftrightarrow \mathbf{A^T} \cdot \mathbf{A} \cdot x = \mathbf{A^T} \cdot b$$

Beweis: Siehe [6] Kapitel 3 Lineare Ausgleichsprobleme (S. 71ff)

Definition 41. (Pseudoinverse)
Stellt man obige Normalengleichung nach x um, erhält man:

$$x = (\mathbf{A^T} \cdot \mathbf{A})^{-1} \cdot \mathbf{A^T} \cdot b$$

Der Term $(\mathbf{A^T} \cdot \mathbf{A})^{-1} \cdot \mathbf{A^T}$ heißt **Pseudoinverse** der Matrix $\mathbf{A}$, falls $\det(\mathbf{A^T} \cdot \mathbf{A}) \neq 0$ gilt und wird mit $\mathbf{A}^+$ bezeichnet, wenn der Vektor $x = \mathbf{A}^+ \cdot b$ die kleinste Lösung (bezüglich der euklidischen Norm $\| \cdot \|_2$) von $\|\mathbf{A} - b \cdot x\| = \min$ für alle $b \in \mathbb{R}^m$ ist.

Bemerkung 42. Die obige Einschränkung ist notwendig, wenn man den Begriff der Pseudoinversen für beliebige $m \times n$-Matrizen ($m \geq n$) verallgemeinern möchte. Die Lösung x hängt von der Matrix $\mathbf{A}$ sowie von b ab und für den Fall $m > n$ ist die Lösung nicht mehr eindeutig bestimmt.

Lösung des linearen Ausgleichsproblems

Eine stabile Lösung des Problems $\|\mathbf{A} - b \cdot x\| = \min$ erhält man, wenn man die Matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ mit $\operatorname{rang}(\mathbf{A}) = n$ mithilfe einer orthogonalen Matrix $\mathbf{Q} \in O(m)$ in die Gestalt einer oberen Dreiecksmatrix bringt:

$$\mathbf{Q} \cdot \mathbf{A} = \begin{pmatrix} * & \ldots & * \\ & \ddots & \vdots \\ & & * \\ & & \end{pmatrix} = \begin{pmatrix} \mathbf{R} \\ 0 \end{pmatrix}$$

Satz 43. *Sei* $\mathbf{A} \in \mathbb{R}^{m \times n}, m \geq n$ *mit rang(*$\mathbf{A}$*)* $= n$ *und* $b \in \mathbb{R}^m$*. Weiter sei* $\mathbf{Q} \in O(m)$ *eine orthogonale Matrix mit*

$$\mathbf{Q} \cdot \mathbf{A} = \begin{pmatrix} \mathbf{R} \\ 0 \end{pmatrix} \text{ und } \mathbf{Q} \cdot b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

Dann ist $x = \mathbf{R^{-1}} b_1$ *die Lösung des Minimierungsproblems* $\|b - \mathbf{A} \cdot x\| = \min_x$.

Beweis: Siehe [6] Kapitel 3.2 Orthogonalisierungsverfahren (S. 77ff)

Nun stellt sich zwangsläufig die Frage, wie man Q bestimmen kann. Hier gibt es prinzipiell zwei Ansätze: Der erste Ansatz ist mithilfe von Rotationsmatrizen, sogenannten Givens-Rotationen, der zweite funktioniert über Spiegelungen an Hyperebenen (Householder-Transformationen). Da Drehungen und Spiegelungen beide invariant bezüglich der euklidischen Norm $\| \cdot \|_2$ sind, eignen sich beide zum Diagonalisieren der Matrix $\mathbf{A}$ und letztendlich zum Lösen des linearen Ausgleichsproblems. In dieser Arbeit werden die Householder-Transformationen verwendet:

Householder-Transformationen

Das Householder-Verfahren arbeitet mit Spiegelungen an Hyperebenen durch den Ursprung. Diese Ebenen lassen sich durch einen Normalenvektor $v \in \mathbb{R}^n, v \neq 0$ beschreiben. Die Transformation ist definiert durch

$$\mathbf{Q_v} = \mathbf{I} - 2\frac{v \cdot v^T}{v^T \cdot v}.$$

Die Grundidee dieser Transformation ist zu einem Vektor $y \in \mathbb{R}^n$, $y \notin \text{span}(e^1)$, ein $v \in \mathbb{R}^n$ zu finden, so dass y gespiegelt an der Hyperebene H_v in Richtung e^1 zeigt. Es muss also

$$\mathbf{Q_v} \cdot y = \pm \|y\|_2 e^1$$

gelten.
Die Herleitung von Seite 103 aus [11] liefert:

$$\alpha = \text{sign}(y_1) \cdot \|y\|_2$$

$$v = y + \alpha e^1$$

$$\mathbf{Q_v} \cdot y = -\alpha e^{-1}$$

Die praktische Vorgehensweise zur Berechnung von $\mathbf{Q}$ und $\mathbf{R}$ wird am nachfolgenden Beispiel verdeutlicht:

Beispiel 44. Gegeben sei die Matrix $\mathbf{A} \in \mathbb{R}^{3\times 2}$ mit

$$\mathbf{A} = \begin{pmatrix} -3 & 2 \\ 0 & 4 \\ 4 & 4 \end{pmatrix}$$

Gesucht sind nun die beiden Matrizen $\mathbf{Q}$ und $\mathbf{R}$, so dass gilt

$$\mathbf{Q} \cdot \mathbf{A} = \begin{pmatrix} \mathbf{R} \\ 0 \end{pmatrix}$$

wobei $\mathbf{R}$ eine obere Dreiecksmatrix ist.
Im ersten Schritt sucht man eine orthogonale Matrix $\mathbf{Q}$, so dass der erste Spaltenvektor von $\mathbf{A}$ (im Folgenden mit y bezeichnet) ein Vielfaches vom Einheitsvektor e_1 wird. Dazu berechnet man nach obiger Vorschrift:

$$y = \begin{pmatrix} -3 \\ 0 \\ 4 \end{pmatrix}$$

$$\alpha = \text{sign}(y_1) \cdot \|v\| = (-1) \cdot \sqrt{(-3)^2 + 0^2 + 4^2} = -5$$

$$v = y + \alpha \cdot e_{3\times 1} = \begin{pmatrix} -3 \\ 0 \\ 4 \end{pmatrix} - 5 \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} -8 \\ 0 \\ 4 \end{pmatrix}$$

$$\mathbf{Q_v} = \mathbf{I} - 2 \cdot \frac{v \cdot v^T}{v^T \cdot v} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \frac{2}{80} \cdot \begin{pmatrix} 64 & 0 & -32 \\ 0 & 0 & 0 \\ -32 & 0 & 16 \end{pmatrix} = \begin{pmatrix} -0.6 & 0 & 0.8 \\ 0 & 1 & 0 \\ 0.8 & 0 & 0.6 \end{pmatrix}$$

Man erhält also die erste orthogonale Matrix $\mathbf{Q}$, die durch Rechtsmultiplikation mit $\mathbf{A}$ die erste Spalte so umformt, dass unterhalb des Diagonalelements nur noch Einträge gleich 0 stehen:

$$\mathbf{Q_v} \cdot A = \begin{pmatrix} 5 & 2 \\ 0 & 4 \\ 0 & 4 \end{pmatrix}$$

Im zweiten Schritt verfährt man genauso mit dem zweiten Spaltenvektor von $\mathbf{A}$ (im Folgenden mit z bezeichnet). Dazu betrachtet man aber nur noch den Vektor unterhalb der Diagonalen, also:

$$z = \begin{pmatrix} 4 \\ 4 \end{pmatrix}$$

$$\alpha = sign(z_1) \cdot \|z\| = (1) \cdot \sqrt{(4^2 + 4^2)} = \sqrt{32}$$

$$w = z + \alpha \cdot e_{2\times 1} = \begin{pmatrix} 4 \\ 4 \end{pmatrix} + \sqrt{32} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 9.65 \\ 4 \end{pmatrix}$$

$$\tilde{\mathbf{Q}}_\mathbf{w} = \mathbf{I} - 2 \cdot \frac{w \cdot w^T}{w^T \cdot w} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 1.70 & 0.70 \\ 0.70 & 0.29 \end{pmatrix} = \begin{pmatrix} -0.70 & -0.70 \\ -0.70 & 0.70 \end{pmatrix}$$

Um die Matrixmultiplikation zu ermöglichen, ohne dass die erste Spalte (die ja bereits bereinigt wurde) davon betroffen ist, bettet man nun die 2×2-Matrix $\tilde{\mathbf{Q}}_\mathbf{w}$ in eine 3×3-Matrix $\tilde{\mathbf{I}}$ ein:

$$\mathbf{Q}_\mathbf{w} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & \tilde{\mathbf{Q}}_\mathbf{w} \end{pmatrix}$$

Damit ergibt sich für die gesamte Berechnung

$$\mathbf{Q}_\mathbf{w} \cdot \mathbf{Q}_\mathbf{v} \cdot \mathbf{A} = \begin{pmatrix} \mathbf{R} \\ 0 \end{pmatrix} = \begin{pmatrix} 5 & 2 \\ 0 & -5.66 \\ 0 & 0 \end{pmatrix}$$

mit

$$\mathbf{Q} = \mathbf{Q}_\mathbf{w} \cdot \mathbf{Q}_\mathbf{v}$$

Die Lösung der QR-Zerlegung liefert die beiden Matrizen:

$$\mathbf{R} = \begin{pmatrix} 5 & 2 \\ 0 & -5.66 \end{pmatrix}, \qquad \mathbf{Q} = \begin{pmatrix} -0.6 & 0 & 0.8 \\ -0.57 & -0.70 & -0.42 \\ 0.57 & -0.70 & 0.42 \end{pmatrix}$$

Dieser Ansatz kann nun dazu verwendet werden, eine Gleichung der obigen Form

$$\mathbf{J_F}(u, v) \cdot \mathbf{X} = \mathbf{J_G}(\tilde{u}, \tilde{v})$$

mit $\mathbf{J_F}(u, v) = \begin{pmatrix} f_{x1} & f_{y1} \\ f_{x2} & f_{y2} \\ f_{x3} & f_{y3} \end{pmatrix}$ und $\mathbf{J_G}(\tilde{u}, \tilde{v}) = \begin{pmatrix} g_{x1} & g_{y1} \\ g_{x2} & g_{y2} \\ g_{x3} & g_{y3} \end{pmatrix}$ zu lösen. Hierzu teilt man die Gleichung in zwei Teile auf:

$$\min_\mathbf{X} = \|\mathbf{J_F} \cdot \mathbf{X} - \mathbf{J_G}\|_F^2 = \min_{x_1, x_2} \|\mathbf{J_F} x_1 - \mathbf{J_{G_{(:,1)}}}\|_2^2 + \|\mathbf{J_F} x_2 - \mathbf{J_{G_{(:,2)}}}\|_2^2$$

Damit erhält man zwei lineare Ausgleichsprobleme, die separat mithilfe der Householder-

$$\begin{pmatrix} f_{x1} & f_{y1} \\ f_{x2} & f_{y2} \\ f_{x3} & f_{y3} \end{pmatrix} \cdot \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix} = \begin{pmatrix} g_{x1} & g_{y1} \\ g_{x2} & g_{y2} \\ g_{x3} & g_{y3} \end{pmatrix}$$

$$\begin{pmatrix} f_{x1} & f_{y1} \\ f_{x2} & f_{y2} \\ f_{x3} & f_{y3} \end{pmatrix} \cdot \begin{pmatrix} x_{11} \\ x_{21} \end{pmatrix} = \begin{pmatrix} g_{x1} \\ g_{x2} \\ g_{x3} \end{pmatrix} \qquad \begin{pmatrix} f_{x1} & f_{y1} \\ f_{x2} & f_{y2} \\ f_{x3} & f_{y3} \end{pmatrix} \cdot \begin{pmatrix} x_{12} \\ x_{22} \end{pmatrix} = \begin{pmatrix} g_{y1} \\ g_{y2} \\ g_{y3} \end{pmatrix}$$

Transformationen näherungsweise gelöst werden können. Anschließend setzt man die beiden Lösungsvektoren zur Matrix $\mathbf{X}$ zusammen. An dieser Stelle ist es von großer Bedeutung die Korrektheit der Lösung zu verifizieren, da $\mathbf{X}$ lediglich eine minimale Lösung bezüglich der euklidischen Norm darstellt. Nachfolgendes Beispiel soll diesen Sachverhalt verdeutlichen:

Beispiel 45. Sei ein C^0-stetiger Übergang wie in Abbildung 4.7 gegeben. Es soll nun geprüft werden, ob der Flächenübergang an der Stelle x G^1-stetig ist und damit eine Reparametrisierung existiert, nach dessen Anwendung dieser Übergang C^1-stetig ist.

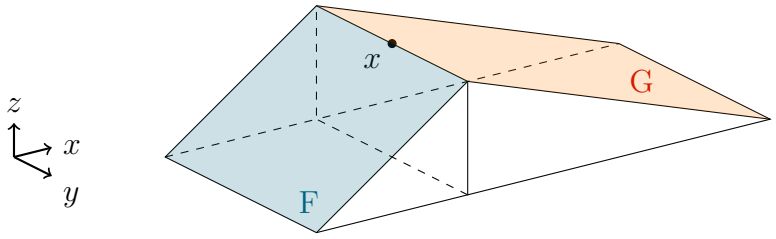


Abbildung 4.7: Nicht G^1-stetiger Flächenübergang

Zunächst werden die Ableitungen in u, v-Richtung der Flächen F, G bestimmt (siehe Abbildung 4.8). Die Flächen seien derart parametrisiert, dass die Ableitung der beiden Flächen in v-Richtung $v_{F,G}$ identisch ist. Die Ableitungsvektoren seien gegeben durch

$$u_F = \begin{pmatrix} -1 \\ 0 \\ -1 \end{pmatrix}, \; u_G = \begin{pmatrix} 1.5 \\ 0 \\ -0.5 \end{pmatrix}, \; v_{F,G} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}.$$

Die dazugehörigen Jacobi-Matrizen haben nun folgende Form:

$$\mathbf{J_F} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \\ -1 & 0 \end{pmatrix}, \; \mathbf{J_G} = \begin{pmatrix} 1.5 & 0 \\ 0 & 1 \\ -0.5 & 0 \end{pmatrix}.$$

Wie bereits beschrieben, besteht der nächste Schritt der G^1-Prüfung in der Lösung der Gleichung

$$\mathbf{J_F} \cdot \mathbf{X} = \mathbf{J_G}.$$

Gelöst wird diese durch Umformulierung in eine Problemstellung der linearen Ausgleichsrechnung. Mit anderen Worten ist im nächsten Schritt die Matrix $\mathbf{X}$ gesucht, die den Term $\|\mathbf{J_F} \cdot \mathbf{X} - \mathbf{J_G}\|_2^2$ minimiert. Unter Zuhilfenahme der Householdertransformationen[1] erhält man

$$\mathbf{X} = \begin{pmatrix} -0.5 & 0 \\ 0 & 1 \end{pmatrix}.$$

[1] Oder einem anderen Ansatz zur Lösung dieser Problemklasse

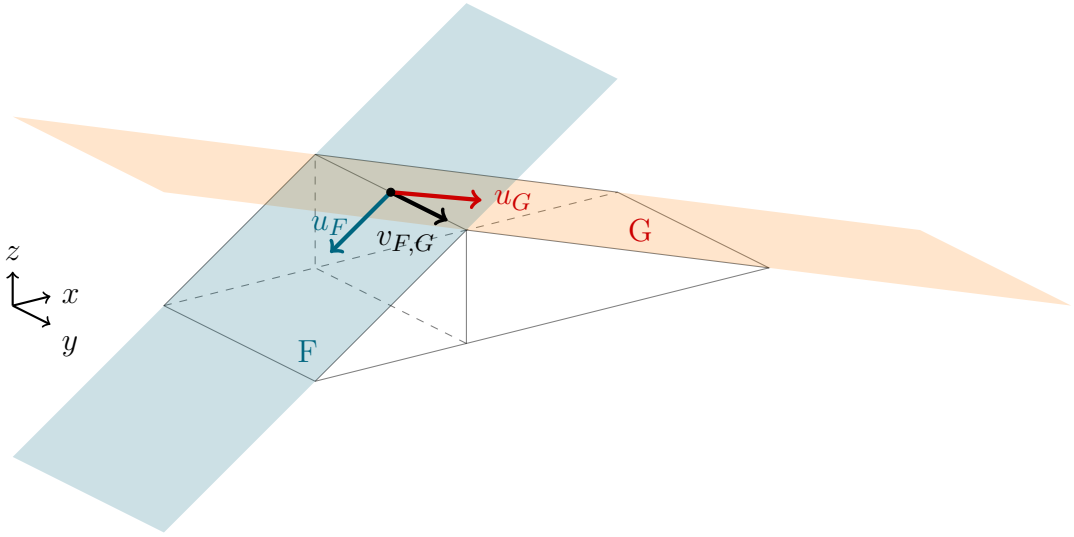


Abbildung 4.8: Richtungsableitungen und Tangentialebenen eines Flächenübergangs

Auch wenn die Matrix $\mathbf{X}$ eine gültige Lösung im Sinne von Invertierbarkeit ist, kann man nicht auf G^1-Stetigkeit des Flächenübergangs im Punkt x schließen. Setzt man $\mathbf{X}$ in obige Gleichung ein und führt eine Korrektheitsprüfung dieser Lösung durch, stellt man Folgendes fest:

$$\mathbf{J_F} \cdot \mathbf{X} = \begin{pmatrix} 0.5 & 0 \\ 0 & 1 \\ 0.5 & 0 \end{pmatrix} \neq \begin{pmatrix} 1.5 & 0 \\ 0 & 1 \\ -0.5 & 0 \end{pmatrix} = \mathbf{J_G}$$

Diese Diskrepanz lässt sich durch die Tatsache erklären, dass die Lösung des linearen Ausgleichsproblems minimal bezüglich der euklidischen Norm ist und immer existiert. Betrachtet man die geometrische Bedeutung von $\mathbf{X}$, so stellt man fest, dass eine „korrekte“ Lösung (Wert des Minimums gleich 0) in diesem Fall nicht zu finden ist, da versucht wird, die beiden Vektoren $u_G, v_{F,G}$ (die die Tangentialebene (rot) der Fläche G im Punkt x aufspannen) als Linearkombination der beiden Vektoren $u_F, v_{F,G}$ (die die Tangentialebene (blau) der Fläche F im Punkt x aufspannen) darzustellen. Da die beiden Ebenen jedoch nicht übereinstimmen, kann nur eine näherungsweise Lösung gefunden werden. Es ist damit unabdingbar, die berechnete Lösung zu verifizieren. Dieser Übergang ist damit nicht G^1-stetig.

Bemerkung 46. Über die Lage der partiellen Ableitungsvektoren zweier aneinandergrenzenden Flächen lassen sich im Allgemeinen keine Aussagen treffen. Es gilt weder die Orthogonalität der Paare (u_F, v_F) zu (u_G, v_G), noch die der partiellen Ableitungen innerhalb einer Fläche. Dies begründet sich in der beliebigen Form der internen Parametrisierung von NX. Daraus folgt auch, dass die Spalten der Lösungsmatrix $\mathbf{X}$ nicht zwingend Einheitsvektoren sind. In Abbildung 4.9 sind drei unterschiedliche Parametrisierungen eines Quadrates mithilfe der angedeuteten Kontrollpolygone abgebildet. Auch

wenn das optische Endresultat der drei Quadrate stets identisch ist, so ist die mittlere Parametrisierung die effizienteste der drei Möglichkeiten, eine Fläche zu verwalten.

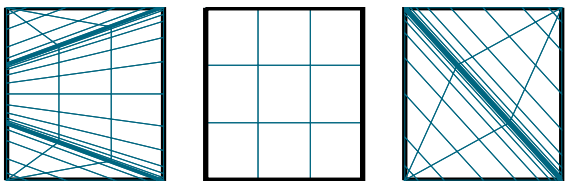


Abbildung 4.9: NX-interne Parametrisierung eines Quadrats

Der letzte Schritt in der Analyse von Flächenübergängen besteht nun in der Prüfung auf G^2-Stetigkeit. In diesem Kontext muss zwischen der sogenannten *Richtungs-G^2-Stetigkeit* und *allgemeinen G^2-Stetigkeit* (kurz G^2-Stetigkeit) unterschieden werden. Im ersten Fall existiert eine Richtung y entlang dieser der Übergang G^2-stetig ist. Ist die Patchgrenze allgemein G^2-stetig, so gilt für alle beliebigen Richtungen, dass der Übergang G^2-stetig ist.

Bemerkung 47. Im Folgenden wird die zweite Ableitung einer Fläche entlang einer Richtung y benötigt. Diese wird in diesem Zusammenhang nicht explizit hergeleitet, sondern vorausgesetzt. Detaillierte Informationen zum Thema Richtungsableitungen können in [5] nachgelesen werden.
Die zweite partielle Ableitung einer Fläche F im Punkt (u, v) entlang einer Richtung $y \in \mathbb{R}^2$ kann mit

$$D_y^2 F(u, v) = y^T \cdot \mathbf{H_F}(u, v) \cdot y$$

berechnet werden.

4.3 Prüfung auf G^2-Stetigkeit

Definition 48. Ein C^1-stetiger Flächenübergang $F(u, v) = G(u, v)$ ist *Richtungs-G^2-stetig* wenn für ein $y \in \mathbb{R}^2$ ein $\lambda \in \mathbb{R}$ existiert, sodass

$$y^T \cdot (\mathbf{H_F}(u, v) - \mathbf{H_G}(u, v)) \cdot y = \lambda \cdot \mathbf{J_F}(u, v) \cdot y$$

in allen drei Komponenten x, y, z übereinstimmt.

Um eine bessere Unterscheidung der jeweiligen Koordinaten x, y und z zu ermöglichen, ist es sinnvoll folgende Notation festzulegen:

Bemerkung 49. Da die Hesse-Matrix einer Fläche F eine 2×2-Matrix ist, die als Komponenten wiederum Vektoren aus $\mathbb{R}^3$ beinhaltet wird folgende Notation für die $x-, y-$ und $z-$Komponente eingeführt:

Sei $\mathbf{H_F}(u,v)$ die Hesse-Matrix gegeben durch

$$\mathbf{H_F}(u,v) = \begin{pmatrix} \frac{\partial^2}{\partial u^2}F(u,v) & \frac{\partial^2}{\partial u \partial v}F(u,v) \\ \frac{\partial^2}{\partial v \partial u}F(u,v) & \frac{\partial^2}{\partial v^2}F(u,v) \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} & \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} \\ \begin{pmatrix} x_3 \\ y_3 \\ z_3 \end{pmatrix} & \begin{pmatrix} x_4 \\ y_4 \\ z_4 \end{pmatrix} \end{pmatrix}.$$

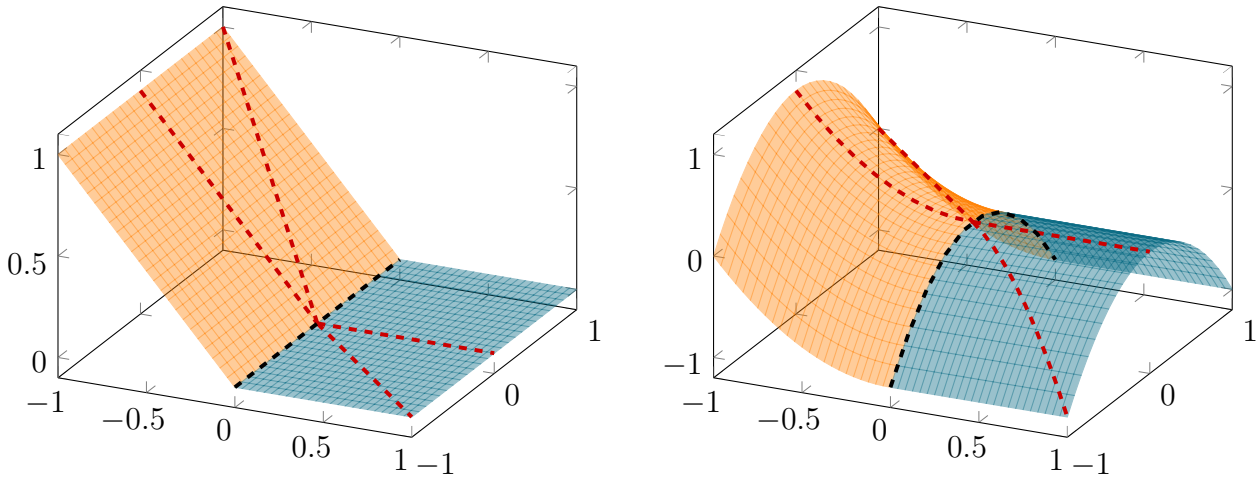
Die Matrix $\mathbf{H_{F_x}}(u,v)$ bezeichnet die x-Komponente der Hesse-Matrix

$$\mathbf{H_{F_x}} := \mathbf{H_{F_{(x,:,:)}}}(u,v) = \begin{pmatrix} x_1 & x_2 \\ x_3 & x_4 \end{pmatrix}.$$

Analog werden im Folgenden $\mathbf{H_{F_y}}(u,v)$ und $\mathbf{H_{F_z}}(u,v)$ für die weiteren Komponenten, sowie $\mathbf{J_{F_x}}, \mathbf{J_{F_y}}$ und $\mathbf{J_{F_z}}$ für die entsprechenden Einträge der Jacobi-Matrix, verwendet.

Bemerkung 50. Für alle nachfolgenden Betrachtung wird $y \neq \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ angenommen, da es sich hierbei um keinen sinnvollen Richtungsbegriff handelt.

Mithilfe der Richtungs-G^2-Stetigkeit ist es bereits möglich für eine vordefinierte Bahnrichtung zu bestimmen, ob diese am Flächenübergang G^2-stetig ist. Im Hinblick auf die Optimierung von G-Code erweist es sich in der Praxis jedoch als nützlich darüber hinaus zu prüfen, ob der Flächenübergang in diesem Punkt für alle Richtungen G^2-stetig ist.

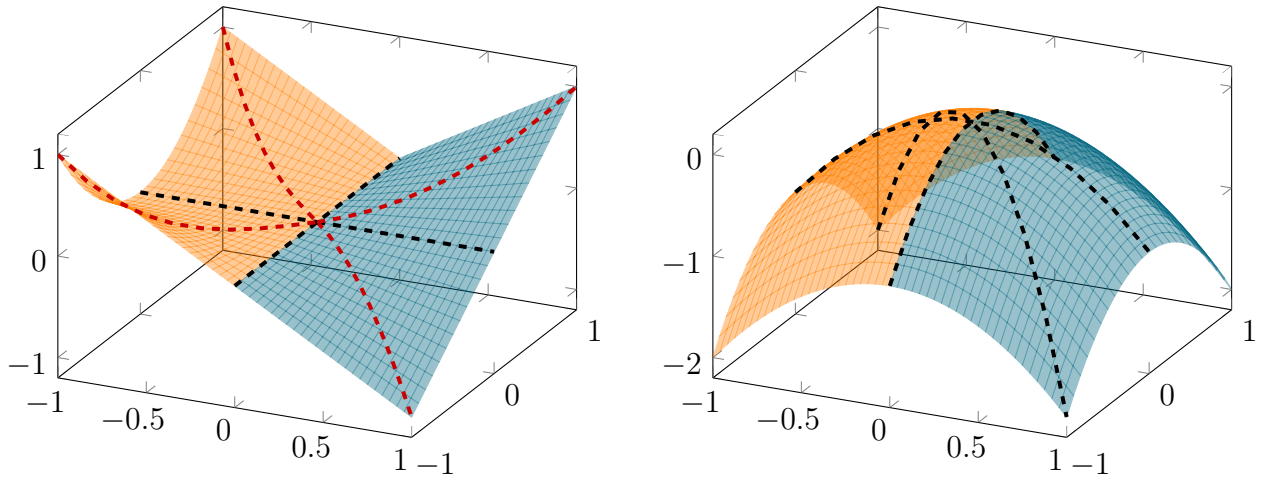


(a) C^0-stetiger Übergang

(b) C^1-stetiger Übergang

Abbildung 4.10: Verschiedene nicht G^2-stetige Bahnverläufe

In Abbildung 4.10(a) ist ein C^0-stetiger Übergang zu sehen, der nicht C^1-stetig ist und damit auch in keiner Richtung G^2-stetig sein kann[2]. Auf der rechten Seite (Abbildung 4.10(b)) ist der Übergang bereits C^1-stetig. Auch hier sind die ausgewählten Richtungen jedoch nicht G^2-stetig. Lediglich die Richtung entlang der Kante erfüllt die notwendigen Kriterien in der zweiten Ableitung. Die die Fläche in diesem Fall jedoch nicht verlassen wird, kann in diesem Fall nicht von einem G^2-stetigen Übergang gesprochen werden.



(a) nicht G^2-stetiger Übergang

(b) G^2-stetiger Übergang

Abbildung 4.11: Verschiedene Bahnverläufe von Flächenübergängen

Die Flächen F (orange) und G (blau) in Abbildung 4.11(a) wurden mit folgender Abbildungsvorschrift konstruiert:

$$F(u,v) = \begin{pmatrix} u \\ v \\ -u \cdot v^2 \end{pmatrix} \text{ und } G(u,v) = \begin{pmatrix} u \\ v \\ u \cdot v \end{pmatrix}$$

Berechnet man nun die Jacobi- und Hesse-Matrizen für die beiden Flächen ergibt sich

$$\mathbf{J_F} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ -v^2 & -2 \cdot u \cdot v \end{pmatrix} \text{ und } \mathbf{J_G} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ v & u \end{pmatrix},$$

$$\mathbf{H_F} = \begin{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ -2 \cdot v \end{pmatrix} \\ \begin{pmatrix} 0 \\ 0 \\ -2 \cdot v \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ -2 \cdot u \end{pmatrix} \end{pmatrix} \text{ und } \mathbf{H_G} = \begin{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \\ \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \end{pmatrix}$$

[2]Ausgenommen ist die Richtung entlang der Kurve.

Der Flächenübergang soll nun im Punkt $(u, v) = (0, 0)$ näher untersucht werden. Die bereits bestimmten Matrizen ergeben sich damit zu

$$\mathbf{J_F}(0,0) = \mathbf{J_G}(0,0) = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 0 \end{pmatrix}$$

und

$$\mathbf{H_F}(0,0) = \begin{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \\ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \end{pmatrix} \text{ und } \mathbf{H_G}(0,0) = \begin{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \\ \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \end{pmatrix}.$$

Anschließend setzt man diese Ergebnisse in Definition 13 ein und erhält für die $x-, y-$ und $z-$ Komponente:

$$f_x := y^T \cdot \left(\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} - \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \right) \cdot y = \lambda \cdot \begin{pmatrix} 1 & 1 \end{pmatrix} \cdot y$$

$$f_y := y^T \cdot \left(\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} - \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \right) \cdot y = \lambda \cdot \begin{pmatrix} 1 & 1 \end{pmatrix} \cdot y$$

$$f_z := y^T \cdot \left(\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} - \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \right) \cdot y = \lambda \cdot \begin{pmatrix} 0 & 0 \end{pmatrix} \cdot y$$

In den ersten beiden Komponenten x, y ist die Richtung y nicht von Belang, da die linke Seite der Gleichung für beliebiges $y \in \mathbb{R}^2$ 0 wird. Damit folgt auch, dass $\lambda = 0$ die einzige Lösung liefert.
Es bleibt also noch die Überprüfung der dritten Komponente f_z. Für ein beliebiges $\lambda \in \mathbb{R}$ wird die rechte Seite der Gleichung 0. Durch die symmetrische Form der Hesse-Matrix $\mathbf{H_G}(0,0)$ wird ersichtlich, dass es nur zwei Richtungen geben kann, für die dieser Übergang G^2-stetig ist, nämlich

$$y = \begin{pmatrix} y_1 \\ 0 \end{pmatrix} \text{ und } y = \begin{pmatrix} 0 \\ y_2 \end{pmatrix}.$$

Dieser Flächenübergang ist also nur entlang der schwarzen Richtungen G^2-stetig. Entlang der roten Richtungen ist der Übergang lediglich C^1-stetig. Auf der rechten Seite (Abbildung 4.11(b)) ist ein Paraboloid der Form

$$F(u,v) = \begin{pmatrix} u \\ v \\ -u^2 - v^2 \end{pmatrix}$$

konstruiert. Dieser Flächenübergang ist in jeder Richtung G^2-stetig[3]. Um genau diesen Sachverhalt bewerten zu können, wird ein Weg benötigt, obige Formel für beliebige Richtungen $y \in \mathbb{R}^2$ zu lösen.

Definition 51. Ein C^1-stetiger Flächenübergang $F(u,v) = G(u,v)$ heißt *G^2-stetig*, wenn ein $\lambda \in \mathbb{R}$ existiert, so dass

$$y^T \cdot (\mathbf{H}_{\mathbf{F_x}}(u,v) - \mathbf{H}_{\mathbf{G_x}}(u,v)) \cdot y = \lambda \cdot \mathbf{J}_{\mathbf{F_x}}(u,v) \cdot y$$

$$y^T \cdot (\mathbf{H}_{\mathbf{F_y}}(u,v) - \mathbf{H}_{\mathbf{G_y}}(u,v)) \cdot y = \lambda \cdot \mathbf{J}_{\mathbf{F_y}}(u,v) \cdot y$$

$$y^T \cdot (\mathbf{H}_{\mathbf{F_z}}(u,v) - \mathbf{H}_{\mathbf{G_z}}(u,v)) \cdot y = \lambda \cdot \mathbf{J}_{\mathbf{F_z}}(u,v) \cdot y$$

für alle $y \in \mathbb{R}^2$ gilt.

In der praktischen Umsetzung des hier entworfenen G^2-Stetigkeitstests spielt die Reparametrisierung abermals eine entscheidende Rolle. Die theoretische Definition der G^2-Stetigkeit setzt die C^1-Stetigkeit voraus. Wie bereits beschrieben, wird genau jene Stetigkeit mithilfe der Reparametrisierung φ sichergestellt. Leitet man nun die reparametrisierte Fläche G_φ partiell ab, so muss durch Anwendungen der Kettenregel auch φ abgeleitet werden. Mit der Fläche F und der reparametrisierten Fläche G_φ ergibt sich damit in der Praxis folgende Gleichung:

$$y^T \cdot \left(\mathbf{H}_{\mathbf{F}}(u^*,v^*) - \mathbf{X}^T \cdot \mathbf{H}_{\mathbf{G}}(\varphi(\tilde{u}^*,\tilde{v}^*)) \cdot \mathbf{X}\right) \cdot y = \lambda \cdot \mathbf{J}_{\mathbf{F}} \cdot y$$

Um nun zu prüfen, ob diese Gleichung für alle $y \in \mathbb{R}^2$ eine gültige Lösung $\lambda \in \mathbb{R}$ besitzt, wird sie im ersten Schritt in ihre drei Komponenten zerlegt und anschließend nach λ aufgelöst:

$$f_x := \lambda = \frac{y^T \cdot \left(\mathbf{H}_{\mathbf{F_x}} - \mathbf{X}^T \cdot \mathbf{H}_{\mathbf{G_x}} \cdot \mathbf{X}\right) \cdot y}{\mathbf{J}_{\mathbf{F_x}} \cdot y}$$

$$f_y := \lambda = \frac{y^T \cdot \left(\mathbf{H}_{\mathbf{F_y}} - \mathbf{X}^T \cdot \mathbf{H}_{\mathbf{G_y}} \cdot \mathbf{X}\right) \cdot y}{\mathbf{J}_{\mathbf{F_y}} \cdot y}$$

$$f_z := \lambda = \frac{y^T \cdot \left(\mathbf{H}_{\mathbf{F_z}} - \mathbf{X}^T \cdot \mathbf{H}_{\mathbf{G_z}} \cdot \mathbf{X}\right) \cdot y}{\mathbf{J}_{\mathbf{F_z}} \cdot y}$$

Im Zähler dieser Gleichungen steht ein quadratische Polynom der Form $y^T \cdot \mathbf{A} \cdot y$, wobei $\mathbf{A} = \mathbf{H}_{\mathbf{F}} - \mathbf{X}^T \cdot \mathbf{H}_{\mathbf{G}} \cdot \mathbf{X}$ für die jeweilige Komponente ist. Um nun zu prüfen, ob dieses polynomiale Gleichungssystem unabhängig von y zu lösen ist, wird die Nullstelle der ersten partiellen Ableitungen gesucht und mithilfe eines Koeffizientenvergleichs berechnet. Hierzu wird die Quotientenregel für gebrochenrationale Funktionen verwendet und der Nenner in dieser Betrachtung bewusst ignoriert.
Sei $y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \mathbf{A} = \mathbf{H}_{\mathbf{F_x}}(u^*,v^*) - \mathbf{X}^T \cdot \mathbf{H}_{\mathbf{G_x}}(\tilde{u}^*,\tilde{v}^*), \mathbf{J}_{\mathbf{F_x}} = (x_1, x_2)$.

[3]Dies gilt sogar entlang der gesamten gemeinsamen Kante.

Dann ergibt sich für die Zähler der Ableitung der Funktion f_x nach y:

$$\begin{pmatrix} \frac{\partial}{\partial y_1} f_x \\ \frac{\partial}{\partial y_2} f_x \end{pmatrix} := \begin{pmatrix} 2 \cdot (x_1 \cdot y_1 + x_2 \cdot y_2) \cdot \mathbf{A}_{(\mathbf{1},:)} \cdot y - y^T \cdot \mathbf{A} \cdot y \cdot x_1 \\ 2 \cdot (x_1 \cdot y_1 + x_2 \cdot y_2) \cdot \mathbf{A}_{(\mathbf{2},:)} \cdot y - y^T \cdot \mathbf{A} \cdot y \cdot x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Durch geeignetes Zusammenfassen erhält man für die drei Komponenten

$$2\mathbf{J_{F_x}} y \left(\mathbf{H_{F_x}}(u^*, v^*) - \mathbf{X}^T \mathbf{H_{G_x}}(\tilde{u}^*, \tilde{v}^*) \right) y - y^T \left(\mathbf{H_{F_x}}(u^*, v^*) - \mathbf{X}^T \mathbf{H_{G_x}}(\tilde{u}^*, \tilde{v}^*) \right) y \mathbf{J_{F_x}}^T = 0$$

$$2\mathbf{J_{F_y}} y \left(\mathbf{H_{F_y}}(u^*, v^*) - \mathbf{X}^T \mathbf{H_{G_y}}(\tilde{u}^*, \tilde{v}^*) \right) y - y^T \left(\mathbf{H_{F_y}}(u^*, v^*) - \mathbf{X}^T \mathbf{H_{G_y}}(\tilde{u}^*, \tilde{v}^*) \right) y \mathbf{J_{F_y}}^T = 0$$

$$2\mathbf{J_{F_z}} y \left(\mathbf{H_{F_z}}(u^*, v^*) - \mathbf{X}^T \mathbf{H_{G_z}}(\tilde{u}^*, \tilde{v}^*) \right) y - y^T \left(\mathbf{H_{F_z}}(u^*, v^*) - \mathbf{X}^T \mathbf{H_{G_z}}(\tilde{u}^*, \tilde{v}^*) \right) y \mathbf{J_{F_z}}^T = 0.$$

Diese Gleichungen liefern das gesuchte Kriterium für die G^2-Stetigkeit eines Flächenübergangs an einem Punkt (u^*, v^*) in beliebige Richtungen y. An dieser Stelle muss noch geprüft werden, ob λ in den jeweiligen Komponenten übereinstimmt. Auf der Basis eines Koeffizientenvergleichs ist es nun möglich einen stabilen G^2-Stetigkeitstest zu implementieren. Die Charakterisierung bietet sich in der Praxis exzellent an, da wie bereits beim C^0- und C^1-Test eine Quantifizierung mithilfe einer vorgegebenen Toleranz möglich ist. Umso kleiner die Koeffizienten des Polynoms sind, desto mehr ist der Flächenübergang G^2-stetig.

Im nächsten Beispiel wird der gesamte Vorgang der G/C-Stetigkeitsprüfung exemplarisch durchgeführt und die Ergebnisse bewertet.

Beispiel 52. Untersucht werden soll der Flächenübergang zweier Flächen F, G wie in Abbildung 4.12 zu sehen. Beide seien mithilfe des in Abbildung 4.12(a) dargestellten Kontrollpolygons derart parametrisiert, dass an der Übergangsstelle gilt $F(0, 0.5) = G(0, 0.5)$.

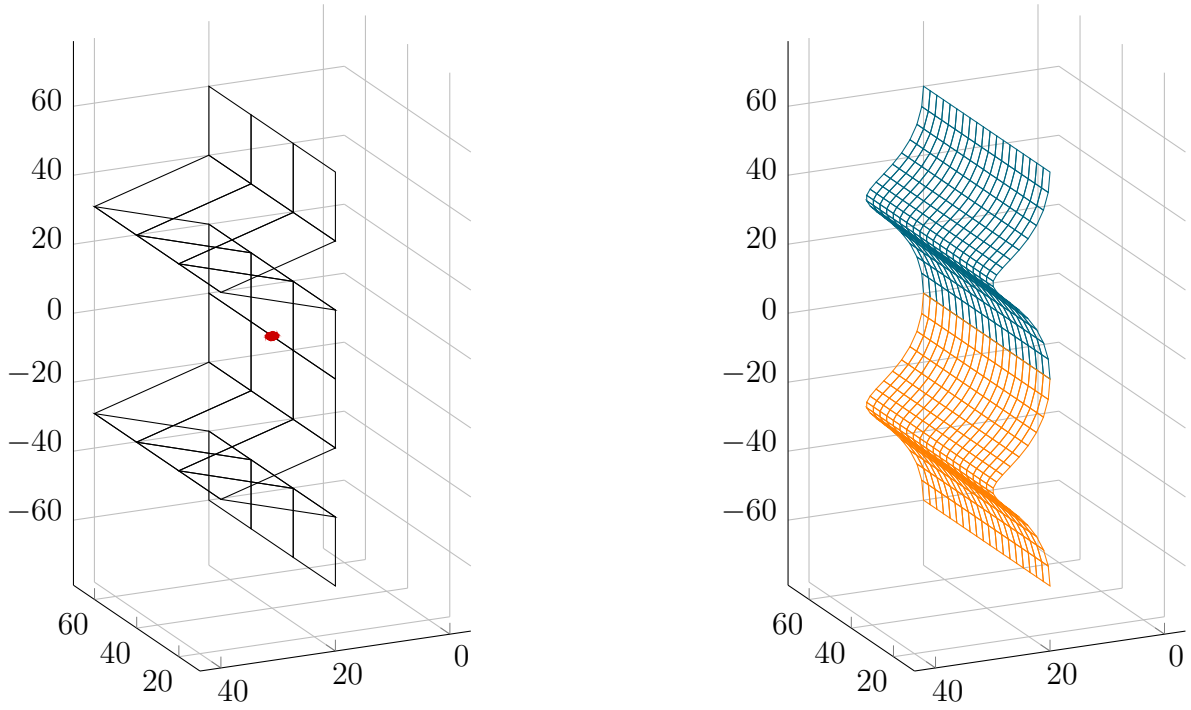
C^0-Stetigkeit

Ausgangspunkt der Betrachtung ist der Punkt $p = (40, 20, 0)$. Da sich die beiden Flächen in diesem Punkt aufgrund der Endpunktinterpolation des Kontrollpolygons exakt treffen, liefert der C^0-Stetigkeitstest zwei identische Werte x_1, x_2 für den gemeinsamen Punkt. Der Übergang wird damit als C^0-stetig identifiziert und es folgt die Prüfung auf C^1-Stetigkeit.

C^1-Stetigkeit

Die beiden Jacobi-Matrizen berechnen sich mithilfe der in Kapitel 2 vorgestellten Ableitungsregeln für B-Spline-Flächen:

$$\mathbf{J_F}(0, 0.5) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 0 \end{pmatrix} \text{ und } \mathbf{J_G} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \\ -1 & 0 \end{pmatrix}$$



(a) Kontrollpolygon

(b) Brückenübergang

Abbildung 4.12: G/C-Stetigkeitstest eines Flächenübergangs

Anschließend setzt man das lineare Ausgleichsproblem $\mathbf{J_F} \cdot \mathbf{X} = \mathbf{J_G}(0, 0.5)$ an

$$\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \cdot \mathbf{X}_{(:,1)} = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix},$$

$$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \cdot \mathbf{X}_{(:,2)} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

und erhält mithilfe der Householder-Transformationen die Lösung

$$\mathbf{X} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Nun muss überprüft werden, ob die Lösung des Minimierungsproblems eine exakte Lösung liefert und damit die Gleichung

$$\mathbf{J_F} \cdot \mathbf{X} - \mathbf{J_G} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

erfüllt ist. Da dies hier offensichtlich der Fall ist, kann der Flächenübergang als G^1-stetig klassifiziert werden und darüber hinaus mit $\mathbf{X}$ die Reparametrisierung

$$\varphi : (u, v) \mapsto \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} u \\ v \end{pmatrix}$$

angegeben werden, um einen C^1-stetigen Flächenübergang zu erhalten. An diesem Beispiel wird auch die geometrische Interpretation der Matrix $\mathbf{X}$ deutlich. In Abbildung 4.13 sind die zwei unterschiedlichen Parametrisierungen an der Übergangsstelle zu sehen. Während die Parametrisierung beider Flächen entlang der v-Richtung identisch verläuft, ist sie in u-Richtung entgegengesetzt. Dieser Zusammenhang ist auch an der Matrix $\mathbf{X}$ abzulesen, da die erste Spalte $\mathbf{X}_{(:,1)} = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$ genau die u-Richtung transformiert und die zweite Spalte $\mathbf{X}_{(:,2)} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ die v-Richtung auf sich selbst abbildet.

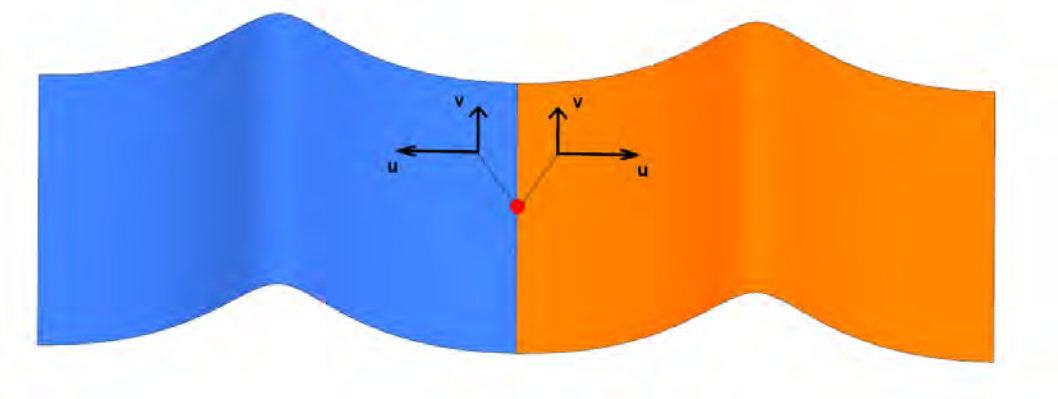


Abbildung 4.13: Parametrisierung des Brückenübergangs

G^2-Stetigkeit

Der letzte Schritt besteht nun darin, die G^2-Stetigkeit zu prüfen. Zunächst müssen die Hesse-Matrizen der beiden Flächen berechnet werden:

$$\mathbf{H_F}(0, 0.5) = \begin{pmatrix} \begin{pmatrix} 0 \\ 240 \\ -360 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \\ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \end{pmatrix} \text{ und } \mathbf{H_G}(0, 0.5) = \begin{pmatrix} \begin{pmatrix} 0 \\ 240 \\ 360 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \\ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \end{pmatrix}.$$

An dieser Stelle müssen die hergeleiteten Gleichungen für die jeweiligen Komponenten aufgestellt und gelöst werden.

x-Komponente:

$$2\begin{pmatrix}0 & 1\end{pmatrix} y \left(\begin{pmatrix}0 & 0\\ 0 & 0\end{pmatrix} - \mathbf{X}^T \begin{pmatrix}0 & 0\\ 0 & 0\end{pmatrix}\right) y - y^T \left(\begin{pmatrix}0 & 0\\ 0 & 0\end{pmatrix} - \mathbf{X}^T \begin{pmatrix}0 & 0\\ 0 & 0\end{pmatrix}\right) y \begin{pmatrix}0\\ 1\end{pmatrix} = \begin{pmatrix}0\\ 0\end{pmatrix}$$

y-Komponente:

$$2\begin{pmatrix}0 & 0\end{pmatrix} y \left(\begin{pmatrix}240 & 0\\ 0 & 0\end{pmatrix} - \mathbf{X}^T \begin{pmatrix}240 & 0\\ 0 & 0\end{pmatrix}\right) y - y^T \left(\begin{pmatrix}240 & 0\\ 0 & 0\end{pmatrix} - \mathbf{X}^T \begin{pmatrix}240 & 0\\ 0 & 0\end{pmatrix}\right) y \begin{pmatrix}0\\ 0\end{pmatrix} = \begin{pmatrix}0\\ 0\end{pmatrix}$$

z-Komponente:

$$2\begin{pmatrix}1 & 0\end{pmatrix} y \underbrace{\left(\begin{pmatrix}-360 & 0\\ 0 & 0\end{pmatrix} - \mathbf{X}^T \begin{pmatrix}360 & 0\\ 0 & 0\end{pmatrix}\right)}_{\begin{pmatrix}0 & 0\\ 0 & 0\end{pmatrix}} y - y^T \underbrace{\left(\begin{pmatrix}-360 & 0\\ 0 & 0\end{pmatrix} - \mathbf{X}^T \begin{pmatrix}360 & 0\\ 0 & 0\end{pmatrix}\right)}_{\begin{pmatrix}0 & 0\\ 0 & 0\end{pmatrix}} y \begin{pmatrix}1\\ 0\end{pmatrix} = \begin{pmatrix}0\\ 0\end{pmatrix}$$

Die Koeffizienten in allen drei Komponenten sind 0 und damit alle identisch. Somit ist die Gleichung zur Bestimmung der G^2-Stetigkeit unabhängig von y lösbar und es gilt die G^2-Stetigkeit in beliebige Richtungen im Punkt $F(0, 0.5)$ des Flächenübergangs der Flächen F und G.

5 Probleme des CAD-Systems

Bei der Erforschung der Schnittstelle NXOpen sowie dem Umgang mit NX selbst sind während der Bearbeitungsphase dieser Master-Arbeit einige Unwägbarkeiten aufgetreten, die in diesem Kapitel kurz vorgestellt und zusammengefasst werden. Dies kann vor allem im Hinblick auf weitere zukünftige Arbeiten in diesem Bereich nützlich werden und von Vorteil sein.

5.1 NX-interne Parametrisierung von Freiformflächen

NX bietet dem Benutzer die Möglichkeit mithilfe des Befehls *Through Curves*[1] Freiformflächen zu konstruieren. Hierbei versucht NX zwei Mengen von zusammenhängenden Kurven aufeinander abzubilden. Wie in Abbildung 5.1 dargestellt, kann man damit z.B. die Kurve a mit der Kurve b verbinden. Je nach gewählten Orientierungssinn der Kurven ergeben sich zwei unterschiedliche Lösungen. Während man im ersten Fall ein aus mathematischer Sicht betrachtet gutmütiges Vieleck erhält, ist das Resultat im zweiten Fall eine sich selbst überschneidende Fläche.

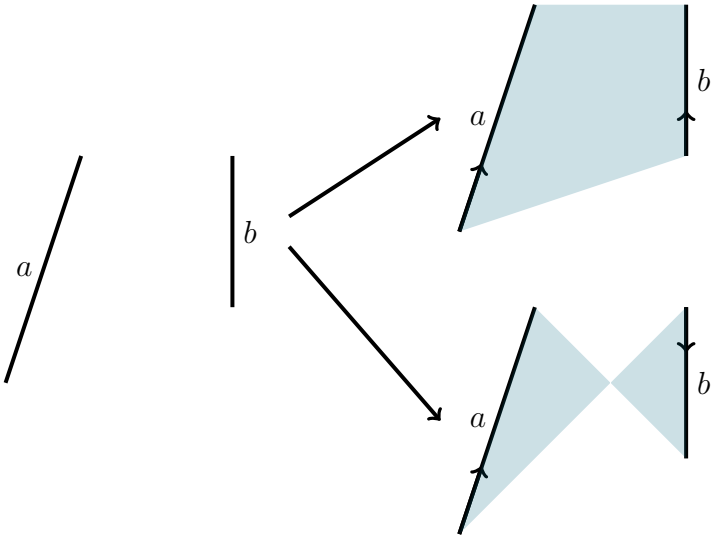


Abbildung 5.1: Erzeugung eines Quadrats durch *Through Curves*

[1] Menü: **Insert** ⇒ **Mesh Surface** ⇒ **Through Curves**

Generell sind solche Freiformflächen selbstverständlich wesentlich komplexer, so dass zwei Kurven oft nicht ausreichen, um die Fläche vollständig zu beschreiben.
Die zweite Problematik, die bei der Konstruktion von Freiformflächen mithilfe des Befehls *Through Curves* auftritt, betrifft die korrekte Wahl der Kantenmengen. Es ist beispielsweise möglich ein Quadrat zu erstellen, in dem man, wie in Abbildung 5.2 dargestellt, die Kurven a_1, a_2 und a_3 mit der Kurve b verbindet oder einfach nur a_2 und b verwendet. Auch wenn das optische Resultat in der Mitte identisch ist, so weisen die beiden Kontrollpolygone deutliche Unterschiede auf.

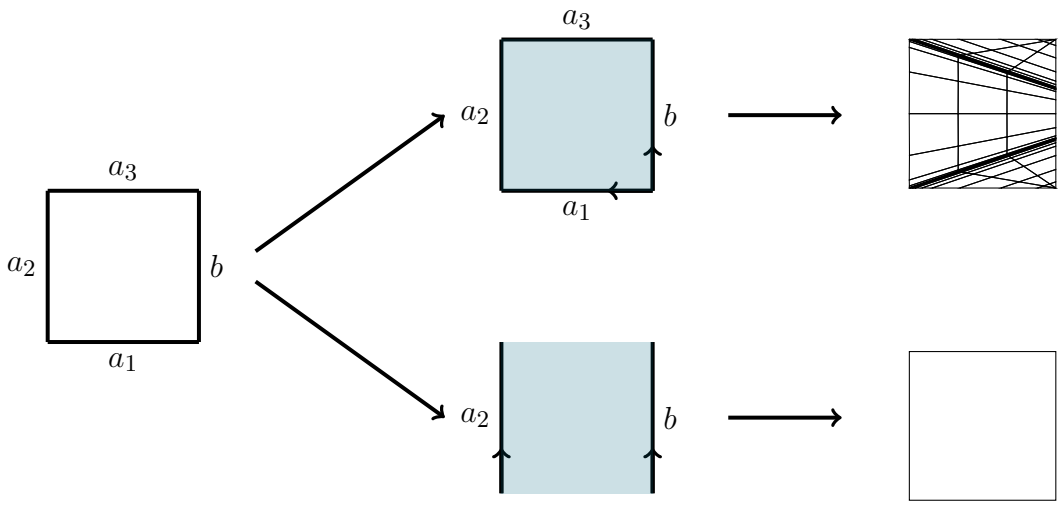


Abbildung 5.2: Verschiedene Parametrisierungen eines Quadrats

Der Grund dafür liegt bei der Eingabe des Benutzers: Im ersten Fall wurden die drei begrenzenden Kurven a_1, a_2 und a_3 der ersten Kantenmenge zugeordnet. NX baut nun intern **eine** Splinekurve aus den drei Kanten und versucht diese linear mit b zu verbinden. In Abbildung 5.3 ist das eigentliche Quadrat in orange angedeutet, die erzeugte Splinekurve ist blau und das dazugehörige Kontrollpolygon schwarz. Es ist zu erkennen, dass das Kontrollpolygon außerhalb der eigentlich Fläche liegt und die Splinekurve an den 90° -Ecken nicht interpoliert. Es ist NX nicht möglich an dieser Stelle einen mehrfachen Knoten einzufügen. Diese Approximation passiert zwar innerhalb einer globalen Toleranz[2], jedoch kann dies insbesondere bei Ableitungen im Bereich dieser Ecke zu numerischen Probleme führen.
Setzt man nun die beiden erzeugten Flächen derart aneinander, dass sie in einer gemeinsamen Ebene liegen, müsste der resultierende Flächenübergang entlang der gesamten gemeinsamen Kante G^2-stetig sein. Die schlechte Parametrisierung einer der beiden Fläche hat jedoch an den Eckpunkten zur Folge, dass die zweite Ableitung hier enorm groß (> 150000) wird und damit das hier entwickelte numerische Verfahren zur Bestimmung der G/C-Stetigkeit nicht funktioniert.

[2]Ein Tausendstel inch.

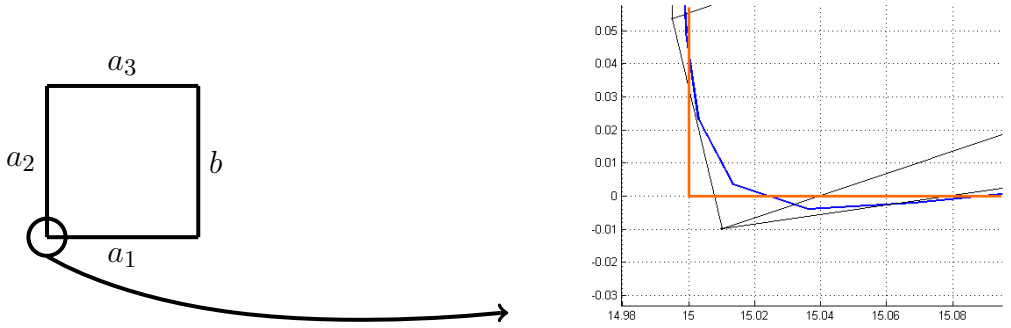


Abbildung 5.3: Untersuchung der Splinekurve an einem Eckpunkt

Bei der Verwendung des Befehls *Through Curves* ist es daher ratsam sorgfältig darauf zu achten, dass einerseits die Orientierung der Kurvenmengen korrekt gesetzt ist und andererseits dass möglichst wenig begrenzenden Kurven verwendet werden.

5.2 Singularitäten

Ein weiteres Problem taucht bei der Betrachtung von B-Spline-Flächen auf, wenn sogenannte *Singularitäten* auftreten. Dieser Fall tritt auf, wenn die Kontrollpunkte entlang einer Richtung auf einen Punkt zusammenfallen und hat zur Folge, dass die erste Ableitungen in diesem Punkt 0 ist. Eine solche Singularität entsteht bei dem Versuch, einen zweidimensionalen rechteckigen Parameterbereich auf ein Dreieck abzubilden. Damit „fällt“ eine Kante des Rechtecks auf einen Punkt im Dreieck zusammen (siehe Abbildung 5.4).

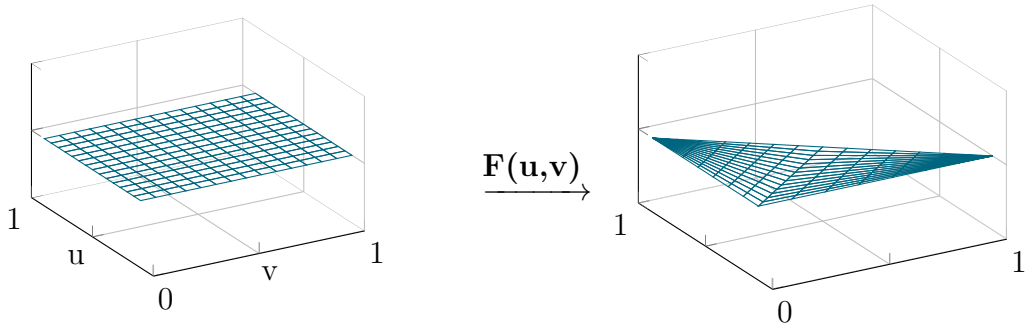
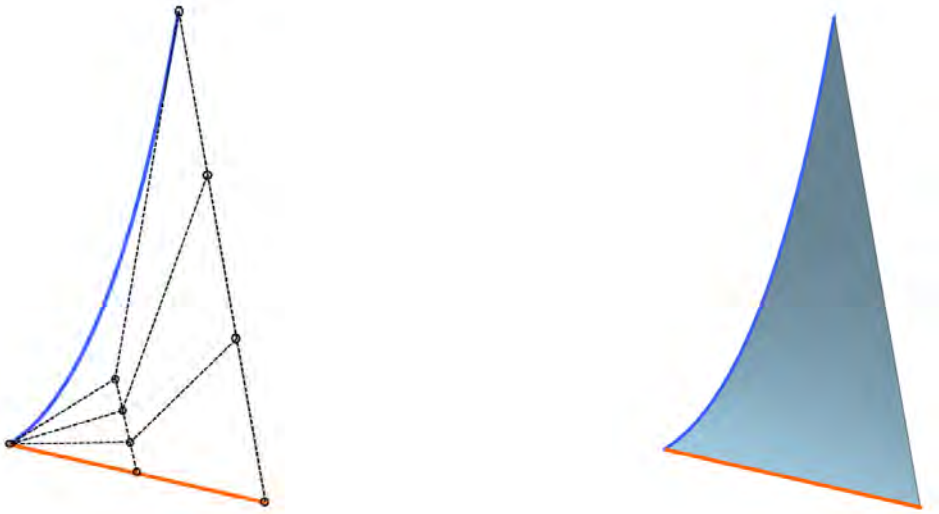


Abbildung 5.4: Zusammenfall des Kontrollpolygons in einer Richtung

Im CAD-System NX lässt sich eine Fläche mit einer Singularität durch Anwendung des Befehls *Through Curves* bewerkstelligen.



(a) Kontrollpolygon (b) Optisches Resultat

Abbildung 5.5: Erzeugung einer Singularität

Die blaue Kurve in Abbildung 5.5 ist ein quadratisches Polynom und die orange Kurve ist eine einfach konstruierte Strecke, die beide senkrecht aufeinander stehen. Durch Anwendung des Befehls *Through Curves* entsteht am Schnittpunkt der blauen und orangen Kurve eine Singularität, in der das Kontrollpolygon entlang einer Kante degeneriert. NX ist an diesem Punkt nicht in der Lage eine Tangentialebene in diesem Punkt zu erzeugen, da die Ableitungen in diesem Punkt 0 sind.

Grenzt an die blaue Fläche F nun eine weitere Fläche G (siehe Abbildung 5.6) ist es trotz der Singularität prinzipiell möglich eine Stetigkeitsprüfung durchzuführen. Während bei der Untersuchung eines Flächenübergangs ohne Singularität die Rollen der beiden Flächen F, G symmetrisch sind, müssen im Falle einer Singularität die Flächen derart gewählt werden, dass Fläche F **keine** Singularität im zu untersuchenden Punkt aufweist. Damit folgt nämlich, dass die Lösung des linearen Ausgleichsproblems während der C^1-Stetigkeitsprüfung eine korrekte Lösung haben kann. Der Übergang kann damit von der „gutmütigen“ Seite untersucht werden. Steht die 0-Spalte auf der rechten Seite der Gleichung

$$\begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \\ z_1 & z_2 \end{pmatrix} \cdot \mathbf{X} = \begin{pmatrix} x_3 & \mathbf{0} \\ y_3 & \mathbf{0} \\ z_3 & \mathbf{0} \end{pmatrix}, \text{ anstatt } \begin{pmatrix} x_1 & \mathbf{0} \\ y_1 & \mathbf{0} \\ z_1 & \mathbf{0} \end{pmatrix} \cdot \mathbf{X} = \begin{pmatrix} x_2 & x_3 \\ y_2 & y_3 \\ z_2 & z_3 \end{pmatrix}$$

kann die Lösung des Minimierungsproblems für diese Gleichung sogar korrekt sein. Tauscht man jedoch die Rolle der beiden Flächen, existiert keine korrekte Lösung für

obige Gleichung. Damit kann ein solcher Übergang nur mit geeigneter Wahl der beiden Flächen untersucht werden. Wie dieses Kapitel gezeigt hat, kann es durch falsch gewählte

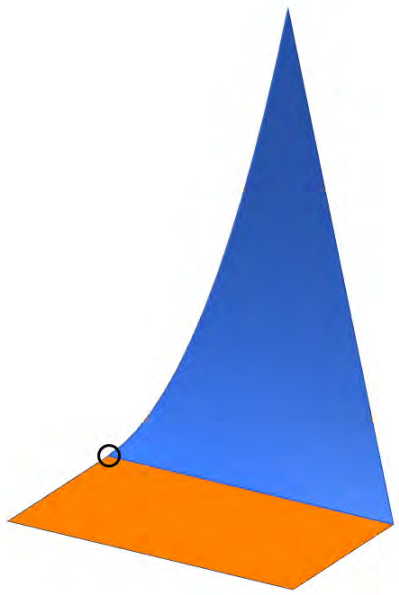


Abbildung 5.6: Flächenübergang mit einer Singularität

Parametrisierungen und unerwünschten Singularitäten zu approximativen Fehlern kommen, die zu unvorhersehbaren Problemstellungen führen können.

6 Fazit

Der erste Teil dieser Arbeit legt den Grundstein für die Verwendung der Schnittstelle NXOpen im Kontext der Fräsbahnoptimierung. Es ist möglich detaillierte Informationen aus dem CAD-Systemen NX zu extrahieren und diese in weiteren Schritten zu verarbeiten. Im zweiten Teil wird die bekannte Theorie der Stetigkeit vom Eindimensionalen ins Mehrdimensionale übertragen und erweitert. Damit ist es nun erstmals möglich Flächenübergänge in einzelnen Punkten im CAD-/CAM-System zu charakterisieren und einen geeigneten Stetigkeitsbegriff einzuführen. Bei den in dieser Arbeit vorgestellten Methoden wurde besonders auf die numerische Stabilität geachtet. Die Stetigkeitstests liefern auch bei leicht gestörten Daten sinnvolle Ergebnisse und eignen sich damit für den Einsatz in der Praxis. Die Beispiele im letzten Kapitel zeigen eindrucksvoll die Bedeutung dieser Forderung für Algorithmen, da kleinste Unterschiede in der Konstruktion zu ungewollten Abweichungen führen können. Der Umgang mit den im letzten Kapitel angesprochenen Unzulänglichkeiten des CAD-Systems NX erweist sich aufgrund von teilweise unvorhersehbaren Approximationen oftmals als sehr anspruchsvoll und damit als eine der Kernanforderungen dieser Arbeit. Eine Besonderheit der hier entwickelten Verfahren ist die Tatsache, dass es möglich ist, den Stetigkeitsbegriff mithilfe von Schwellwerten zu definieren. Die Abweichung der Lösung des linearen Ausgleichsproblems bei der Überprüfung der C^1-Stetigkeit sowie die Größe der Koeffizienten beim G^2-Stetigkeitstest erlauben eine detaillierte Quantifizierung.

Es ist denkbar, die Stetigkeitsprüfung eines Flächenübergangs nicht nur punktuell zu betrachten, sondern entlang der gemeinsamen Kante. Damit wird die Transformationsmatrix $\mathbf{X}$ der Jacobi-Matrizen zu einer Funktion und ermöglicht damit Aussagen über die Stetigkeit entlang des gesamten Kantenverlaufes zu tätigen. Darüber hinaus könnte die Schnittstelle NXOpen dazu verwendet werden, weitere Informationen (nicht nur an Kantenübergängen) zu extrahieren und damit das Werkstück zu optimieren. Durch die enorme Größe der Schnittstelle NXOpen könnte in weiterführenden Arbeiten aktiv, sowohl in den Modeling-Prozess als auch in den Manufacturing-Prozess, eingegriffen werden, um dem Benutzer weitere Optimierungen mithilfe eines programmierten Interfaces zu ermöglichen.

Ich erkläre an Eides Statt, dass ich meine Masterarbeit mit dem Titel *Extraktion von Geometriedaten aus CAD/CAM Systemen* selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe und dass ich alle Stellen, die ich wörtlich oder sinngemäß aus Veröffentlichungen entnommen habe, als solche kenntlich gemacht habe. Die Arbeit hat bisher in gleicher oder ähnlicher Form oder auszugsweise noch keiner Prüfungsbehörde vorgelegen.

Passau, den 2. Mai 2014

_____

(Florian Lorenz)

Literaturverzeichnis

[1] Carl de Boor. On Calculating with B-Splines. *Journal Of Approximation Theory*, 6:50–62, 1972.

[2] Carl de Boor. *A Practical Guide to Splines*. Springer-Verlag, New York, 2001.

[3] Gerald Farin. *Curves and Surfaces for Computer Aided Geometric Design - A Practical Guide*. Academic Press, San Diego, CA, 1988.

[4] Joerg Handeck. 3D-Geometrie Daten Verarbeitung in CNC Steuerungen. *Justus-Liebig-Universitaet*, 2013.

[5] Erwin Kreyszig. *Differential Geometry*. Dover, 1991.

[6] Andreas Hohmann Peter Deuflhard. *Numerische Mathematik 1*. de Gruypter Verlag, Berlin, 2008.

[7] Joerg Peters. *Handbook Geometric Continuity*. University of Florida, 2001.

[8] Tomas Sauer. Numerische Mathematik im Vorlesungsskript, Universitaet Passau.

[9] Tomas Sauer. Splinekurven und -flaechen in CAGD und Anwendung. 2013.

[10] Joerg Handeck Tomas Sauer, Carsten Hamm. Spline multiresolution and wavelet-like decomposition. *Preprint*, 2014.

[11] Arnold Reusken Wolfgang Dahmen. *Numerik fuer Ingenieure und Naturwissenschaftler*. Springer-Verlag, Aachen, 2007.