Modifikation von Freiformflächen unter Randbedingungen auf der Basis unregelmäßig über den Flächenbereich verteilter Meßpunkte

Diplomarbeit von Jörg Hörner

22. Dezember 1995

Prüfer: Prof. Dr. K. Höllig

Betreuerin: Dr. Sabine Roth-Koch

Hiermit versichere ich, daß ich die vorliegende Arbeit allein und nur mit den im Literaturverzeichnis angegebenen Hilfsmitteln angefertigt habe.

Jörg Marcus Hörner

Kapitel 1

Motivation und Aufgabenstellung

1.1 Motivation

CAD/CAM-Systeme (Computer Aided Design/Computer Aided Manufacturing) sind aus der Konstruktion und Fertigung von Bauteilen nicht mehr wegzudenken. Dabei ersetzen sie nicht nur das Reißbrett, sondern bieten auch vielfältige Möglichkeiten zur Modellgenerierung und –optimierung. So können einerseits Ausrundungen und Passungen erzeugt und andererseits charakteristische Eigenschaften des Bauteils überprüft werden (z. B. die Oberflächenstruktur anhand von Reflexionslinien). Dabei kommt man ohne physikalisches Modell (Prototyp) aus. Allerdings können die rechnerinternen Modelle die Prototypen nicht vollständig ersetzen, da diese immer noch für Tests notwendig sind, die nicht oder nur schwer simuliert werden können (z. B. Strömungsverhalten im Windkanal).
So kommt es bei der Entwicklung eines Bauteils zu einer Abfolge von Prototypen und CAD-Modellen, die einander entsprechen müssen. Die Erstellung eines Prototyps aus einem CAD-Modell wird mit Hilfe von NC-Fertigungsverfahren oder neuerdings mit generativen Fertigungsverfahren (z. B. Stereolithographie) bewerkstelligt. Die Digitalisierung eines Prototyps erfolgt durch die meßtechnische Erfassung von Punktdaten. Hierbei können durch optische Verfahren, z. B. Laserscanning, große Datenmengen schnell erfaßt werden. Diese Punktdaten ermöglichen es dann, neue CAD-Modelle aufzubauen und vorhandene zu modifizieren. Für den Aufbau von Freiformflächen sind Funktionalitäten in CAD-Systemen gegeben, zur Modifikation bieten die CAD-Systeme allerdings bisher nur wenig Möglichkeiten an.

1.2 Aufgabenstellung

Es ist ein Algorithmus zu entwickeln, der Freiformflächen anhand von Meßpunktdaten modifiziert. Dabei sollen gegebenenfalls Übergänge in nicht modifizierte Modellbereiche erhalten bleiben (bis zur Krümmungsstetigkeit) und die Meßpunkte mit einer vorgegebenen Toleranz approximiert werden.
Der Algorithmus ist in eine Anwenderfunktion des CAD-Systems CATIA von Dassault Systems einzubinden, mit der folgende Funktionen ausgeführt werden können:

- Einlesen von Meßpunktdaten aus einer Datei,
- Auswahl der zu berücksichtigenden Meßpunkte und CAD-Modell-Flächen,
- Feststellung der Bereiche mit Abweichungen, die außerhalb einer zu wählenden Toleranz liegen,
- Auswahl der zu ändernden Flächen,
- Ändern der ausgewählten Flächen unter Berücksichtigung von Randbedingungen,
- Speichern der geänderten Flächen im CAD-Modell oder als Koeffizientendatei.

Kapitel 2

Einführung

In diesem Kapitel werden die später verwendeten mathematischen Begriffe kurz eingeführt.

2.1 Parameterdarstellung von Kurven und Flächen

2.1.1 Parameterdarstellung von Kurven

Das Bild eines reellen Intervalls I unter einer stetigen, lokal injektiven Abbildung in den $\mathbb{R}^n$ heißt *parametrisierte Kurve*. Im $\mathbb{R}^2$ ergibt sich eine ebene Kurve, im $\mathbb{R}^3$ (im allgemeinen) eine Raumkurve.
Die Funktion $K : I \to \mathbb{R}^n; \quad t \mapsto K(t)$ heißt *Parameterdarstellung*, t *Parameter* der Kurve.
Zwei Parameterdarstellungen heißen äquivalent, wenn sie durch eine umkehrbare *Parametertransformation* $T : I \to I'; \quad t \mapsto t'$ auseinander hervorgehen.
Eine Parameterdarstellung, die wenigstens einmal stetig differenzierbar ist und $|K'(t)| \neq 0$ erfüllt, wird *regulär* genannt. Hat die Ableitung von K an der Stelle t_0 eine Nullstelle, so ist die Kurve dort *singulär*.
Die Gerade, die durch den Punkt $K(t_0)$ und den Vektor $K'(t_0)$ gegeben ist, heißt Tangente der Kurve am Punkt $K(t_0)$.
Die Länge L eines Kurvenabschnitts $K(t),\ t \in [t_0, t_1]$, wird berechnet über

$$L = \int_{t_0}^{t_1} |K'(t)|dt.$$

Ist $|K'(t)| \equiv 1$, so heißt K *nach der Bogenlänge parametrisiert*.
Ist K wenigstens zwei mal stetig differenzierbar in t, so heißt

$$\kappa(t) := \frac{|K'(t) \quad \times K''(t)|}{|K'(t)|^3}; K(t) \neq 0$$

die *Krümmung* der Kurve am Punkt $K(t)$.
Die Krümmung ist unabhängig von der Parameterdarstellung der Kurve [2]. Ist $\kappa(t) \equiv 0$, so liegt die Kurve auf einer Geraden. Ist in einem Punkt $t_0 \quad \kappa(t_0) = 0$, so hat die Kurve

hier einen *Flach-* oder *Wendepunkt.*
Da sich für einen Kreis mit Radius r die Krümmung $\kappa = \frac{1}{r}$ ergibt, wird $\frac{1}{\kappa(t_0)}$ der *Krümmungsradius* an der Stelle t_0 genannt.

2.1.2 Parameterdarstellung von Flächen

Ähnlich wie Kurven lassen sich auch Flächen definieren. Das Bild einer stetigen, lokal injektiven Abbildung F eines Gebietes $I_1 \times I_2$ des $\mathbb{R}^2$ nach $\mathbb{R}^n$ heißt Fläche, die Abbildung $F : I_1 \times I_2 \to \mathbb{R}^n; \quad (u, v) \mapsto F(u, v)$ heißt Parameterdarstellung der Fläche, u und v heißen Parameter dieser Darstellung. Die Linien $u = \text{const.}$ und $v = \text{const.}$ beschreiben ein Netz von *Parameterlinien* auf der Fläche.
Eine differenzierbare Parameterdarstellung heißt regulär, wenn

$$\left| \frac{\delta F}{\delta u} \times \frac{\delta F}{\delta v} \right| \neq 0.$$

Hat das Kreuzprodukt der partiellen Ableitungen eine Nullstelle, so hat die Fläche an diesem Punkt im allgemeinen eine Singularität.
Auch bei Flächen heißen zwei Parameterdarstellungen äquivalent, wenn es eine umkehrbare Parametertransformation $u' = u'(u, v); v' = v'(u, v)$ gibt, die sie ineinander überführt.
Die Ebene an einen Punkt $F(u_0, v_0)$, die von den beiden Ableitungsvektoren $\frac{\delta F}{\delta u}(u_0, v_0)$ und $\frac{\delta F}{\delta v}(u_0, v_0)$ aufgespannt wird, nennt man *Tangentialebene* an die Fläche am Punkt $F(u_0, v_0)$.
Eine Kurve im Parametergebiet induziert eine Kurve auf der Fläche. Über solche Flächenkurven lassen sich Flächenpunkten Krümmungen zuordnen. Der Anteil in Richtung der Flächennormalen heißt Normalkrümmung, der Anteil senkrecht zum Tangentenvektor und zur Normalen geodätische Krümmung in Richtung der Flächenkurve. Die Normalkrümmung ist im allgemeinen von der gewählten Flächenkurve abhängig und hat zwei Extremwerte: die beiden *Hauptkrümmungen* κ_1 und κ_2. Das Produkt aus den Hauptkrümmungen heißt *Gaußsche Krümmung.* Die Gaußsche Krümmung läßt eine Aussage über die lokale Gestalt der Fläche zu. Ist sie positiv, so hat die Fläche einen elliptischen Punkt, negativ, so hat die Fläche einen hyperbolischen Punkt, und verschwindet sie, so ist die Fläche an dieser Stelle parabolisch.

2.2 Polynomiale Kurven und Flächen

Sind die Funktionen der Parameterdarstellung Polynome, so heißt die Darstellung *polynomial vom Grad n*, wobei n die größte auftretende Potenz des Parameters in der Darstellung ist. Bei Flächen kann zwischen den beiden Parametern unterschieden werden, man spricht dann von einer polynomialen Fläche vom Grad n in u-Richtung und Grad m in v-Richtung (u,v: Parameter der Fläche).

2.2.1 Monom-Darstellung

Die Funktionen $m_i : \mathbb{R} \to \mathbb{R}; \quad t \mapsto t^i \quad \forall i \in \mathbb{N}$ werden *Monome* genannt. Die Monome $m_0, \ldots, m_n$ bilden eine Basis der Polynome über $\mathbb{R}$ vom Grad $\leq n$, da sie linear unabhängig sind und jedes Polynom vom Grad n als Linearkombination aus Monomen darstellbar ist.
Eine Parameterdarstellung der Form

$$K(t) = \sum_{i=0}^{n} \vec{a}_i m_i(t) = \sum_{i=0}^{n} \vec{a}_i t^i, \quad \vec{a}_i \in \mathbb{R}^3$$

heißt Monom-Darstellung der Kurve K. Die $\vec{a}_i$ heißen *Koeffizienten* der Darstellung.
Entsprechend heißt bei Flächen eine Parameterdarstellung der Form

$$F(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} \vec{a}_{ij} m_{ij}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} \vec{a}_{ij} u^i v^j, \quad \vec{a}_{ij} \in \mathbb{R}^3$$

Monomdarstellung der Fläche F. Hierbei sind die $m_{ij} : \mathbb{R}^2 \to \mathbb{R}; \quad (u,v) \mapsto u^i, v^j$ die Monome in zwei Variabeln.

2.2.2 Bernstein-Polynome

Eine andere Basis für die Polynome vom Grad n bilden die *Bernstein-Polynome vom Grad n*, die definiert sind als:

$$B_i^n(t) = \binom{n}{i} (1-t)^{n-i} t^i, \quad 0 \leq i \leq n.$$

Diese Definition ist hergeleitet aus der binomischen Formel:

$$1 = [(1-t)+t]^n = \sum_{i=0}^{n} \binom{n}{i} (1-i)^{n-i} t^i.$$

Daraus ergeben sich folgende Eigenschaften der Bernstein-Polynome:

1. Teilung der Eins
$$\sum_{i=0}^{n} B_i^n(t) \equiv 1;$$
2. Rekursion
$$B_i^n(t) = (1-t)B_i^{n-1} + tB_{i-1}^{n-1};$$
3. Symmetrie
$$B_i^n(t) = B_{n-i}^n(1-t).$$

Schränkt man den Definitionsbereich auf das Intervall [0,1] ein, so kommen noch folgende Eigenschaften hinzu:

1. $B_0^n(0) = B_n^n(1) = 1; \quad B_0^n(1) = B_n^n(0) = 0;$

2. $B_i^n(0) = B_i^n(1) = 0, \quad 0 < i < n;$

3. $\max B_i^n(t) = B_i^n\left(\frac{i}{n}\right);$

4. $B_i^n(t) \geq 0;$

5. $\frac{\partial^j B_i^n}{(\partial t)^j}(0) = \frac{\partial^j B_{n-i}^n}{(\partial t)^j}(1) = 0, \quad 0 < j < i.$

Diese Eigenschaften lassen sich alle durch Einsetzen sehr einfach nachrechnen.

2.2.3 Bézier-Kurven

Mit den Bernstein-Polynomen als Basisfunktionen ergibt sich nun eine Darstellung

$$K(t) = \sum_{i=0}^{n} \vec{b}_i B_i^n(t), \quad \vec{b}_i \in \mathbb{R}^3, \quad t \in [0,1]$$

als *Bézier-Kurve vom Grad n*. Die Vektoren $\vec{b}_i$ sind dabei die *Bézier-Punkte*. Der Streckenzug $(\overline{b_0 b_1}, \overline{b_1 b_2}, \ldots, \overline{b_{n-1} b_n})$ durch die Bézier-Punkte wird als *Bézier-Polygon* bezeichnet.
Aufgrund der Eigenschaften der Bernstein-Polynome gibt es wichtige Beziehungen zwischen dem Bézier-Polygon und der Bézier-Kurve:

1. Konvexe Hülle

   Da die Bernstein-Polynome die Eins teilen, liegt die Bézier-Kurve immer in der konvexen Hülle des Bézier-Polygons.

2. Graderhöhung

   Aufgrund der Rekursionsformel berechnen sich die Bézier-Punkte vom Grad $n+1$ einer Kurve aus den Bézier-Punkten vom Grad n über folgende Rekursion:

   $$\vec{b}_i' = \frac{i}{n+1}\vec{b}_{i-1} + \left(1 - \frac{i}{n+1}\right)\vec{b}_i; \quad 0 \leq i \leq n+1, \quad \vec{b}_{-1} = \vec{b}_{n+1} = \vec{o},$$

   wobei $\vec{o}$ den Nullvektor bezeichnet.

   Dies ergibt sich, wenn man eine Bézier-Kurve vom Grad n und eine Bézier-Kurve vom Grad $n+1$ allgemein ansetzt und dann einen Koeffizienten-Vergleich durchführt.

Der Kurvenpunkt an einer Stelle t_0 läßt sich aus dem Bézier-Polygon mit Hilfe des *de-Casteljau-Algorithmus* berechnen, der wie folgt abläuft:
Aus den Bézier-Punkten werden neue Punkte berechnet und zwar:

$$\vec{b}_{i-1,i} := (1 - t_0)\vec{b}_{i-1} + t_0\vec{b}_i, \quad 1 \leq i \leq n$$

Mit diesen Punkten wird dasselbe Verfahren wieder angewandt, so daß man zu einer Rekursionsformel gelangt:

$$\vec{b}_{i,\ldots,k} := (1 - t_0)\vec{b}_{i,\ldots,k-1} + t_0\vec{b}_{i+1,\ldots,k}, \quad 0 \leq i < k \leq n$$

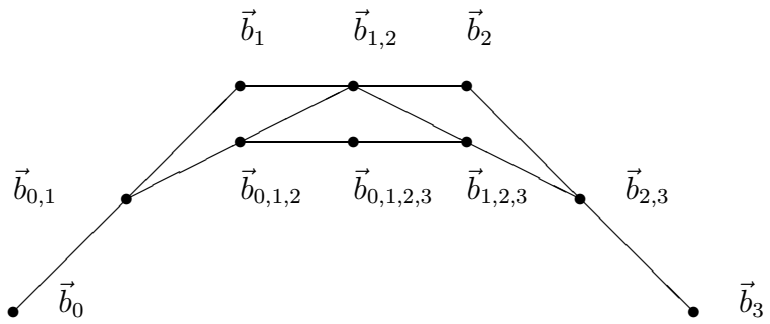


Abbildung 2.1: de-Casteljau-Algorithmus für eine Bézier-Kurve vom Grad 3.

Der letzte Punkt $\vec{b}_{0,\ldots,n}$ entspricht dann der Kurve an der Stelle t_0:

$$K(t_0) = \vec{b}_{0,\ldots,n}$$

(Beweis siehe [1]).
Außerdem bilden die Punkte

$$\vec{b}_0, \vec{b}_{0,1}, \ldots, \vec{b}_{0,\ldots,n}$$

das Bézier-Polygon einer Kurve, die identisch mit der Teilkurve

$$K(t), \quad t \in [0, t_0]$$

ist. Entsprechend bilden die Punkte

$$\vec{b}_{0,\ldots,n}, \vec{b}_{1,\ldots,n}, \ldots, \vec{b}_n$$

ein Bézier-Polygon zur anderen Teilkurve (siehe Abbildung 2.1).

2.2.4 Bézier-Flächen

Setzt man die Bernsteinpolynome als Basis für eine Flächendarstellung an, so erhält man

$$F(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} \vec{b}_{ij} B_i^n(u) B_j^m(v),$$

eine *Bézier-Fläche vom Grad* (n, m).
Die $\vec{b}_{ij}$ heißen auch hier Bézier-Punkte. Sie bilden das *Bézier-Netz*, in dessen konvexer Hülle die Fläche liegt.
Nimmt man aus dem Bézier-Netz die Punkte mit einem festen Index, z.B. $i = 0$, heraus, so bekommt man das Bézier-Polygon der Kurve, die die entsprechende Parameterlinie

beschreibt, also in unserem Beispiel die Parameterlinie $u = \frac{i}{n} = 0$, was einer Randkurve der Fläche entspricht.
Eine Graderhöhung in u-Richtung bei Bézier-Flächen wird durchgeführt, indem man diese Graderhöhung für alle Polygone mit festem Index j durchführt, das heißt die Punkte

$$\vec{b}_{ij}, \quad j = const., \quad 0 \leq i \leq n$$

betrachtet und mit diesen eine Graderhöhung wie unter 2.2.3 beschrieben durchführt.
Ein Flächenpunkt für (u_0, v_0) wird berechnet, indem man den de-Casteljau-Algorithmus zunächst für alle Polygone der einen Richtung durchführt und dann mit dem Polygon aus den daraus resultierenden Punkten den Algorithmus in die andere Richtung anwendet, also zunächst

$$\vec{h}_i := \sum_{j=0}^{m} \vec{b}_{ij} B_j^n(v_0), \quad 0 \leq i \leq n$$

berechnet und dann

$$F(u_0, v_0) = \sum_{i=0}^{n} \vec{h}_i B_i^n(u_0) = \sum_{i=0}^{n} \sum_{j=0}^{m} \vec{b}_{ij} B_i^n(u_0) B_j^m(v_0).$$

Man kann allerdings auch gemischt vorgehen, also z.B. abwechselnd einen Schritt in u- und einen in v-Richtung ausführen. Dabei braucht man die wenigsten Berechnungen, wenn man zunächst die Richtung mit dem kleineren Grad vollständig durchrechnet und dann die andere, da die Rechenschritte, die für den de-Casteljau-Algorithmus durchgeführt werden müssen, proportional zu n^2 wachsen.

2.3 Randbedingungen

Hat man mehrere Kurven oder Flächen, stellt sich häufig die Frage, wie diese ineinander übergehen und wie sich die Tangenten bzw. Krümmungen an diesen Übergängen verhalten.

2.3.1 Stetige Übergänge von Kurven

Zwei Kurven gehen (C^0-)*stetig* ineinander über, wenn der Endpunkt der ersten Kurve mit dem Anfangspunkt der zweiten Kurve identisch ist. Für zwei Bézier-Kurven K_1 vom Grad n_1, K_2 vom Grad n_2 heißt dies, wenn

$$K_1(1) = K_2(0)$$

oder mit Hilfe der Bézierpunkte ausgedrückt

$$\vec{b}_{n_1}^1 = \vec{b}_0^2$$

gilt.
Stimmen in diesem Punkt die Tangentenvektoren beider Kurven überein, so spricht man von *Tangentenstetigkeit* oder C^1-*Stetigkeit*. Stimmen auch noch die Krümmungsvektoren

überein, so nennt man dies *Krümmungs-* oder C^2*-Stetigkeit*. Allgemein gilt: Stimmen die i-ten Ableitungen zweier Kurven für $0 \leq i \leq n$ überein, so nennt man diesen Übergang C^n-stetig. Da bei Bézier-Kurven nur die ersten bzw. letzten $i+1$ Punkte Einfluß auf die i-te Ableitung an einem Randpunkt haben (siehe Eigenschaft 5 der auf $[0,1]$ eingeschränkten Bernstein-Polynome), ergeben sich hier die Übergangsbedingungen
für C^1-Stetigkeit zu:

$$n_1(\vec{b}^1_{n_1} - \vec{b}^1_{n_1-1}) = n_2(\vec{b}^2_1 - \vec{b}^2_0) \tag{2.1}$$

und für C^2-Stetigkeit zu:

$$n_1(n_1-1)(\vec{b}^1_{n_1} - 2\vec{b}^1_{n_1-1} + \vec{b}^1_{n_1-2}) = n_2(n_2-1)(\vec{b}^2_2 - 2\vec{b}^2_1 + \vec{b}^1_0), \tag{2.2}$$

(siehe auch [1]).
Stimmen an einem Randpunkt nicht die Tangentenvektoren, aber die Tangenten selbst überein, so spricht man von *geometrischer Tangentenstetigkeit*.
Werden diese beiden Kurven nach der Bogenlänge parametrisiert, so sind die transformierten Darstellungen C^1-stetig. Es gibt also eine Umparametrisierung einer der beiden Kurven, die aus der geometrischen Tangentenstetigkeit eine C^1-Stetigkeit macht.
Analog spricht man von *geometrischer* C^i*-Stetigkeit*, wenn es eine Umparametrisierung einer der Darstellungen gibt, so daß die Darstellungen C^i-stetig werden bzw. wenn die Kurven nach der Bogenlänge parametrisiert C^i-stetig sind. Man spricht deshalb auch von *Bogenlängenstetigkeit*.

2.3.2 Stetige Übergänge von Flächen

Entsprechend gilt für Flächen: Zwei Flächen gehen (C^0-) stetig ineinander über, wenn sie an einer Randkurve übereinstimmen. Für C^1-Stetigkeit müssen die Tangentenvektoren an allen Randkurvenpunkten gleich sein und für C^2-Stetigkeit auch die Krümmungsvektoren. Dies bedeutet, daß für zwei Bézier-Flächen $F_1(u_1, v_1)$ vom Grad (n_1, m) und $F_2(u_2, v_2)$ vom Grad (n_2, m) die Bedingung erfüllt sein muß, daß alle $m+1$ Bézier-Kurven aus Polygonen mit festem Index j der ersten Fläche in die entsprechenden Kurven der zweiten Fläche übergehen, also für C^0-Stetigkeit

$$\vec{b}^1_{n_1 j} = \vec{b}^2_{0j}, \quad 0 \leq j \leq m,$$

für C^1-Stetigkeit zusätzlich

$$n_1(\vec{b}^1_{n_1 j} - \vec{b}^1_{n_1-1,j}) = n_2(\vec{b}^2_{1j} - \vec{b}^2_{0j}), \quad 0 \leq j \leq m$$

und dann noch für C^2-Stetigkeit

$$n_1(n_1-1)(\vec{b}^1_{n_1 j} - 2\vec{b}^1_{n_1-1,j} + \vec{b}^1_{n_1-2,j}) = n_2(n_2-1)(\vec{b}^2_{2j} - 2\vec{b}^2_{1j} + \vec{b}^1_{0j}), \quad 0 \leq j \leq m$$

gelten muß.

Kapitel 3

Vorgehensweise

In diesem Kapitel wird beschrieben, wie aus einem C^2-stetigen Flächenmodell und den Meßpunkten ein geändertes Flächenmodell entwickelt wird, wobei die C^2-Stetigkeit erhalten bleibt.

3.1 Bereitstellen der Meßpunkte und Flächendaten

Es wird davon ausgegangen, daß die benötigten Daten schon im CAD-Modell vorhanden sind oder in einer Datei vorliegen. Im zweiten Fall werden zwei Dateiformate unterstützt:

- VDAFS 2.0 nach DIN 66301 [12] für Punkte und Flächen, da dieses Format für den Austausch von CAD-Daten in Deutschland weit verbreitet ist und durch die Norm eine eindeutige Beschreibung des Formats vorliegt und
- DMIS 3.0 nach ANSI/CAM-I 101-199X [13], da dieses Format zur Übertragung von Meßdaten von Koordinatenmeßprogrammen im allgemeinen unterstützt wird und auch hier eine Norm vorliegt.

Daten die in anderen Formaten vorliegen, müssen über möglicherweise vorhandene CAD-Schnittstellen eingelesen werden.
Die Daten aus der Datei werden in die CATIA-Datenstruktur eingebunden. Punkte werden in der Datenstruktur durch Koordinatentripel repräsentiert, Flächen durch ein Rechteckgitter aus *Segmenten*. Die Segmente selbst sind in Monomdarstellung gegebene Flächen. Die Datenstruktur einer Fläche besteht daher aus der Information der Segmentanzahlen in u- und v-Richtung (die Gesamtzahl der Segmente ergibt sich als Produkt) und den Daten für die einzelnen Segmente. Zu einem Segment gehören wiederum die Grade der Darstellung und die Koeffizientenmatrizen für die drei Koordinaten.
Aus den vorhandenen Punkten und Flächen werden interaktiv diejenigen ausgewählt, die für die Flächenänderung relevant sein sollen.

3.2 Projektion der Meßpunkte

Um eine Information über das Maß der Flächenabweichung zu haben, muß der Abstand der gemessenen Punkte zu den ausgewählten Flächen bestimmt werden. Die Meßpunkte werden dazu auf alle Flächen normal projiziert und die Projektion mit dem kürzesten Abstand wird ausgewählt. Unterscheiden sich zwei Projektionen um weniger als eine (einstellbare) Konstante, so werden beide Projektionen berücksichtigt. Die Vektoren zwischen Meßpunkt und jeweiliger Projektion, die als *Differenzvektoren* bezeichnet werden, geben an, wie die Fläche geändert werden soll.
Für die Projektion werden im CAD-System vorhandene Funktionen verwendet. In CATIA wird hierbei folgendes Verfahren angewandt:

- Für jedes Segment wird ein Parameterliniengitter berechnet.
- Das Segment wird mit Hilfe der Gitterpunkte in Dreiecke unterteilt.
- Für jedes Dreieck wird die Normale berechnet.
- Es wird festgestellt, ob die Gerade durch den Punkt parallel zur Normalen das Dreieck schneidet. In diesem Fall wird der Schnittpunkt als Ausgangswert für die weitere Berechnung nach dem Newton-Verfahren gewählt.
- Wird für kein Dreieck ein Schnittpunkt gefunden, so gibt es keine Projektion.
- Werden auf verschiedenen Segmenten Lösungen gefunden, wird die mit kürzestem Abstand gewählt.

Das Verfahren ist in [7] beschrieben.
Nachdem die Projektion mit dem kleinsten Abstand gefunden wurde, wird die Abweichung mit der eingegebenen Toleranz verglichen. Ist die Abweichung zu groß, wird der Punkt entsprechend markiert. Die durch die Projektion gewonnenen Daten werden aber in jedem Fall gespeichert.

3.3 Festlegen des Änderungsbereichs

Nun kann vom Benutzer anhand der angezeigten Projektionen eine Auswahl der zu ändernden Flächen vorgenommen werden. Dabei können auch Flächen gewählt werden, die keine Projektion außerhalb der Toleranz oder überhaupt keine Projektion aufweisen, um eine globale Flächenänderung zu erlauben.
Es können auch Punkte markiert werden, die bei der Berechnung unberücksichtigt bleiben sollen. Dies ist sinnvoll, wenn sie offensichtlich falsch projiziert wurden oder ein deutlicher Meßfehler vorliegt. Auch zur Beschleunigung der Berechnungen kann dies sinnvoll sein, da für jeden Meßpunkt sichergestellt werden muß, daß er durch die Änderung der Fläche in den Toleranzbereich kommt bzw. diesen nicht verläßt.
Mit den so ausgewählten Punkten und Flächen wird weitergearbeitet.

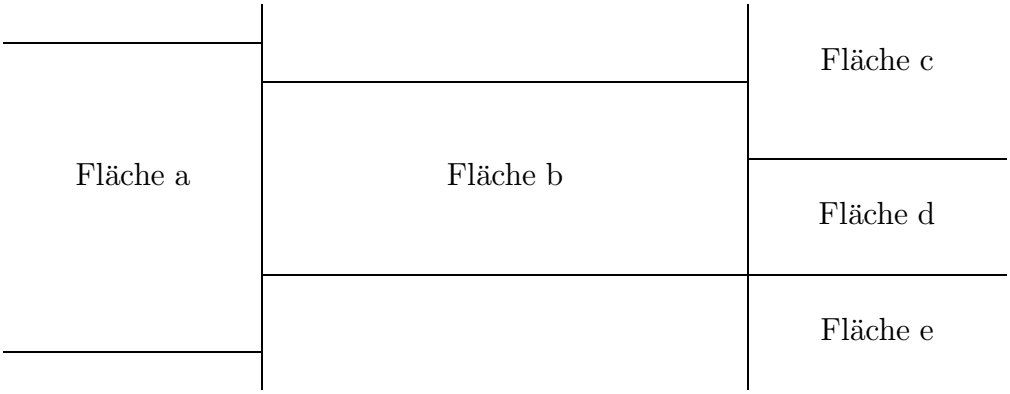


Abbildung 3.1: Mögliche Nachbarschaftsbeziehungen

3.4 Feststellen der Nachbarschaftsbeziehungen

Bei der Berechnung der Flächenänderung sollen Übergänge in benachbarte Flächen bis zur C^2-Stetigkeit erhalten bleiben. Um dies zu gewährleisten, muß zunächst festgestellt werden, wie die Nachbarschaftsbeziehungen aussehen, das heißt, welche Flächen aneinandergrenzen. Hierbei können verschiedenartige gemeinsame Kanten auftreten. (siehe Abbildung 3.1).
Um sämtliche Möglichkeiten zu berücksichtigen, wird folgendermaßen vorgegangen:

- Für eine Fläche werden die Eckpunkte berechnet.
- Diese Eckpunkte werden auf alle anderen Flächen projiziert.
- Fällt ein Eckpunkt mit seiner Projektion zusammen, so liegt dieser Punkt auf beiden Flächen. Die beiden Flächen berühren sich.
- Liegen zwei Eckpunkte einer Fläche auf einer anderen, so haben diese beiden Flächen eine Kante gemeinsam (Fläche a und b).
- Liegt nur ein Eckpunkt auf der zweiten Fläche, so werden die Eckpunkte der zweiten Fläche berechnet und auf die erste projiziert.
- Ergibt diese Rückprojektion zwei Punkte, die auf der ersten Fläche liegen, so ist wieder eine gemeinsame Kante gefunden (Fläche b und d).
- Ist auch bei der Rückprojektion nur ein Punkt auf beiden Flächen, so wird festgestellt, ob sich die Punkte der ersten und zweiten Projektion unterscheiden. Falls ja, ist auch eine gemeinsame Kante gefunden (Fläche b und c).
- Sind die beiden Punkte gleich, so liegt eine Eckenberührung vor (Fläche b und e).

Tritt ein anderer Fall auf, wie z.B. drei oder gar vier Punkte auf beiden Flächen, wird dies als Fehler im Modell angesehen und nicht berücksichtigt.

Durch diese Methode werden Flächen, die nur an Eckpunkten übereinstimmen, aber ansonsten auseinanderklaffen, als benachbart erkannt. Sie wird dennoch so angewandt, da eine genaue Überprüfung der Übergänge zu viel Rechenzeit beanspruchen würde. Allerdings kommen solche absichtlich offenen Stellen nur in wenigen Modellen vor und es ist eine getrennte Betrachtung der beiden Flächenbereiche möglich.
Um die Anzahl der Berechnungen zu verringern, werden am Anfang die Eckpunkte aller Flächen berechnet und die Projektionen nur einmal ausgeführt.

3.5 Berechnen der Flächenänderung

Nachdem die Nachbarschaftsbeziehungen festgestellt worden sind, wird für jede Fläche und innerhalb einer Fläche für jedes einzelne Segment die Änderung bestimmt. Für die Reihenfolge der Bearbeitung der Flächen gibt es drei vernünftige Möglichkeiten:

1. Es wird die Summe über die Differenzvektoren jeder Fläche gebildet und die Fläche bearbeitet, die den längsten Summenvektor aufweist. Hierdurch werden die Flächen mit großen oder vielen Änderungen zuerst berechnet. Dadurch können sich große Änderungen über Segmentgrenzen hinweg fortpflanzen. Das Modell wird weniger lokal verändert.

2. Es wird die Summe über die Differenzvektoren jeder Fläche gebildet und die Fläche zuerst geändert, die den kürzesten Summenvektor aufweist. Flächen ohne oder mit kleiner Änderung werden dadurch zuerst festgelegt. Die Modelländerung ist lokaler.

3. Die Reihenfolge wird durch die Anzahl der Freiheitsgrade bestimmt, das heißt, es wird die Fläche zuerst bearbeitet, die nach Berücksichtigung der Übergangsbedingungen die wenigsten Freiheitsgrade für die Approximation hat. Dies ist zu empfehlen, wenn die Fehler gleichmäßig über die Flächen verteilt sind.

Um die Berechnung der Änderung in zwei voneinander unabhängige Schritte

1. Erhaltung der Stetigkeit und

2. Approximation der Differenzvektoren

einzuteilen, werden als Basisfunktionen $p_{i,j}$ die Produkte von Bernstein-Polynomen vom Grad n in u- und Grad m in v-Richtung verwendet, das heißt, die Segmente werden in Bézier-Darstellung umgerechnet.

$$p_{i,j}(u,v) := B_i^n(u)B_j^m(v) = \binom{n}{i}\binom{m}{j}u^i(1-u)^{n-i}v^j(1-v)^{m-j}, \quad i = 0\ldots n, j = 0\ldots m.$$

Aufgrund der in Kapitel 2 beschriebenen Eigenschaften der Bernstein-Polynome haben auf die C^2-Stetigkeit an Rand $u = 0$ nur die Basisfunktionen $p_{i,j},\ i \leq 2$, am Rand $u = 1$ nur $p_{i,j},\ i \geq n-2$ Einfluß. Entsprechend wird j für die Ränder $v = 0$ und $v = 1$ eingeschränkt. Die anderen Basisfunktionen stehen uneingeschränkt für die Approximation zur Verfügung.

Es ergibt sich also für das Segment S das geänderte Segment $\tilde{S}$

$$\tilde{S} = S + R_1 + R_2 + R_3 + R_4 + E_1 + E_2 + E_3 + E_4 + I$$

mit

$$\begin{aligned}
R_1 &= R_1(u,v) := \sum_{i=0}^{2} \sum_{j=3}^{m-3} \vec{a}_{i,j} B_i^n(u) B_j^m(v), \\
R_2 &= R_2(u,v) := \sum_{i=3}^{n-3} \sum_{j=m-2}^{m} \vec{a}_{i,j} B_i^n(u) B_j^m(v), \\
R_3 &= R_3(u,v) := \sum_{i=n-2}^{n} \sum_{j=3}^{m-3} \vec{a}_{i,j} B_i^n(u) B_j^m(v), \\
R_4 &= R_4(u,v) := \sum_{i=3}^{n-3} \sum_{j=0}^{2} \vec{a}_{i,j} B_i^n(u) B_j^m(v), \\
E_1 &= E_1(u,v) := \sum_{i=0}^{2} \sum_{j=0}^{2} \vec{a}_{i,j} B_i^n(u) B_j^m(v), \\
E_2 &= E_2(u,v) := \sum_{i=0}^{2} \sum_{j=m-2}^{m} \vec{a}_{i,j} B_i^n(u) B_j^m(v), \\
E_3 &= E_3(u,v) := \sum_{i=n-2}^{n} \sum_{j=m-2}^{m} \vec{a}_{i,j} B_i^n(u) B_j^m(v), \\
E_4 &= E_4(u,v) := \sum_{i=n-2}^{n} \sum_{j=0}^{2} \vec{a}_{i,j} B_i^n(u) B_j^m(v), \\
I &= I(u,v) := \sum_{i=3}^{n-3} \sum_{j=3}^{m-1} \vec{a}_{i,j} B_i^n(u) B_j^m(v).
\end{aligned}$$

Die $\vec{a}_{i,j}$ sind noch zu bestimmen. Entsprechenes gilt für C^1 und C^0-Stetigkeit.

3.5.1 Erhaltung der Stetigkeit

Die Vorgehensweise ist für alle vier Ränder gleich, daher wird hier nur der Rand $u = 0$ betrachtet. Außerdem wird vom Erhalt einer C^2-Stetigkeit ausgegangen, für andere Stetigkeiten sind die Indizes entsprechen zu verändern.
Am Rand $u = 0$ des Segments tritt einer der folgenden Fälle auf:

1. Das Segment grenzt an kein Segment aus dem Änderungsbereich.

   Ist dies der Fall, so ist dafür zu sorgen, daß sich die vorhandene C^2-Stetigkeit zum restlichen Modell nicht ändert. Daraus ergeben sich die Faktoren $\vec{a}_{i,j}$

   $$\vec{a}_{i,j} = 0 \quad i = 0 \ldots 2, \quad j = 1 \ldots m.$$

   R_1, E_1 und E_2 sind bestimmt und nicht zur Approximation frei.

2. Das Segment hat eine Grenze mit einem anderen Segment aus dem Änderungsbereich gemeinsam, das noch nicht bearbeitet wurde.

   Da die Stetigkeitsbedingungen bei der Bearbeitung der anderen Segmente berücksichtigt werden, können sie für dieses Segment unberücksichtigt bleiben. R_1 steht zusätzlich zur Approximation zur Verfügung. E_1 und E_2 müssen noch überprüft werden.

3. Das Segment hat eine Grenze mit einem anderen Segment aus dem Änderungsbereich gemeinsam, das schon bearbeitet ist.

   Um die Stetigkeit zu erhalten, müssen die Faktoren entsprechend dem geänderten Segment gewählt werden. Hierbei ist zu berücksichtigen, ob „echte“ Stetigkeit oder nur geometrische Stetigkeit vorlag. Die Koeffizienten ergeben sich zu:

$$\vec{a}_{0,j} = \vec{b}_{n,j}, \tag{3.1}$$

$$\vec{a}_{1,j} = \lambda\frac{n}{m}(\vec{b}_{n,j} - \vec{b}_{n-1,j}) + \vec{a}_{0,j}, \tag{3.2}$$

$$\vec{a}_{2,j} = \lambda^2\frac{n(n-1)}{m(m-1)}(\vec{b}_{n,j} - 2\vec{b}_{n-1,j} + \vec{b}_{n-2,j}) + 2\vec{a}_{1,j} - \vec{a}_{0,j}. \tag{3.3}$$

   wobei

   die $\vec{b}_{i,j}$ die Faktoren des bereits geänderten Segmentes sind,

   m der Grad des momentanen Segmentes ist,

   n der Grad des schon geänderten Segmentes ist,

   λ das Verhältnis der Längen der beiden Tangentenvektoren der ungeänderten Segmente ist, also

$$\lambda = \frac{|\vec{a}'_{1,j} - \vec{a}'_{0,j}|}{|\vec{b}'_{n,j} - \vec{b}'_{n-1,j}|}.$$

   Die Strich-Formen sind die Koeffizienten der ungeänderten Segmente. Für „echte“ Stetigkeit ergibt sich $\lambda = 1$.

   Lag keine Stetigkeit vor und soll diese erreicht werden, muß die Gleichung mit den Summen aus alten Koeffizienten und Änderungskoeffizienten erfüllt sein.

4. Der Rand ist Teil des Randes eines anderen Segmentes aus dem Änderungsbereich.

   Wenn das andere Segment schon bearbeitet wurde, werden die $a_{i,j}$ entsprechend berechnet, andernfalls werden sie bestimmt, nachdem das andere Segment bearbeitet ist. Die Faktoren sind jedenfalls bestimmt und nicht zur Approximation frei.

5. Das Segment grenzt an mehrere Segmente aus dem Änderungsbereich, die alle noch nicht bearbeitet wurden.

   In diesem Fall ist der Rand für die Approximation frei.

6. Das Segment grenzt an mehrere Segmente aus dem Änderungsbereich, die teilweise schon bearbeitet oder fest sind.

   In diesem Fall muß das Segment in zwei Segmente aufgebrochen werden, die dann entsprechend der Fälle 1. – 6., die für sie gelten, behandelt werden.

3.5.2 Approximation der Differenzvektoren

Um die Differenzvektoren $\vec{d_i}$ zu approximieren, wird ein Approximationsansatz nach der *diskreten Fehlerquadratmethode von Gauß* gewählt (siehe [1]). Dazu ist

$$q := \sum_{i=1}^{N} \left(I(u_i, v_i) - \vec{d_i} \right)^2$$

zu minimieren, wobei u_i und v_i die durch die Projektion gewonnenen Parameterwerte des Punktes auf der Fläche sind.
Betrachtet man die einzelnen Koordinaten getrennt, folgt nun:

$$\begin{aligned}
& \tfrac{\partial q}{\partial a_{j,k}} = 2 \sum_{i=1}^{N} \left(I(u_i, v_i) - d_i \right) \left(\tfrac{\partial I}{\partial a_{j,k}}(u_i, v_i) \right) \stackrel{!}{=} 0 && \forall j, k, \\
\Rightarrow \quad & \sum_{i=1}^{N} \left(I(u_i, v_i) - d_i \right) \left(B_j^n(u_i) B_k^m(v_i) \right) = 0 && \forall j, k, \\
\Rightarrow \quad & \sum_{i=1}^{N} I(u_i, v_i) B_j^n(u_i) B_k^m(v_i) = \sum_{i=1}^{N} d_i B_j^n(u_i) B_k^m(v_i) && \forall j, k, \\
\Rightarrow \quad & \sum_{i=1}^{N} \sum_{l_1=3}^{n-3} \sum_{l_2=3}^{m-3} a_{l_1,l_2} B_{l_1}^n(u_i) B_{l_2}^m(v_i) B_j^n(u_i) B_k^m(v_i) = \sum_{i=1}^{N} d_i B_j^n(u_i) B_k^m(v_i) && \forall j, k.
\end{aligned}$$

Setzt man

$$H_{jk}^i = B_j^n(u_i) B_k^m(v_i),$$

ergibt sich ein lineares Gleichungssystem $M\vec{a} = \vec{v}$ mit

$$M = \begin{pmatrix}
\sum_{i=1}^{N} H_{3,3}^i H_{3,3}^i & \sum_{i=1}^{N} H_{3,4}^i H_{3,3}^i & \cdots & \sum_{i=1}^{N} H_{n-3,m-3}^i H_{3,3}^i \\
\sum_{i=1}^{N} H_{3,3}^i H_{3,4}^i & \sum_{i=1}^{N} H_{3,4}^i H_{3,4}^i & \cdots & \sum_{i=1}^{N} H_{n-3,m-3}^i H_{3,4}^i \\
\vdots & \vdots & \ddots & \vdots \\
\sum_{i=1}^{N} H_{n-3,m-3}^i H_{3,3}^i & \sum_{i=1}^{N} H_{n-3,m-3}^i H_{3,4}^i & \cdots & \sum_{i=1}^{N} H_{n-3,m-3}^i H_{n-3,m-3}^i
\end{pmatrix},$$

$$\vec{a} = \begin{pmatrix}
a_{3,3} \\ a_{3,4} \\ \vdots \\ a_{3,m-3} \\ a_{4,3} \\ \vdots \\ a_{n-3,m-3}
\end{pmatrix}, \qquad \vec{v} = \begin{pmatrix}
\sum_{i=1}^{N} d_i H_{3,3}^i \\
\sum_{i=1}^{N} d_i H_{3,4}^i \\
\vdots \\
\sum_{i=1}^{N} d_i H_{3,m-3}^i \\
\sum_{i=1}^{N} d_i H_{4,3}^i \\
\vdots \\
\sum_{i=1}^{N} d_i H_{n-3,m-3}^i
\end{pmatrix},$$

für die Koordinaten x,y und z.

Sind weniger Differenzvektoren gegeben als Freiheitsgrade vorhanden sind, ist das Gleichungssystem unterbestimmt. In diesem Fall wird zusätzlich die Quadratsumme der Koeffizienten minimiert. Es ergibt sich also eine Extremwertberechnung

$$\sum_{ij} a_{i,j}^2 \rightarrow \min$$

mit der Nebenbedingung $M\vec{a} = \vec{v}$.
Diese löst man mit Hilfe des linearen Gleichungssystems

$$\begin{pmatrix} M' & \emptyset \\ id_n & M'^T \end{pmatrix} \begin{pmatrix} \vec{a} \\ \vec{\lambda} \end{pmatrix} = \begin{pmatrix} \vec{v}' \\ \vec{o} \end{pmatrix},$$

wobei
$M'\vec{a} = \vec{v}'$ das aus $M\vec{a} = \vec{v}$ resultierende linear unabhängige Gleichungssystem,
M'^T die transponierte Matrix zu M',
id_n die $n \times n$ Einheitsmatrix,
$\emptyset$ eine Nullmatrix,
$\vec{o}$ der n-dimensionale Nullvektor und
n die Dimension des Vektors $\vec{a}$ sind.
Der Vektor $\vec{\lambda}$ besteht aus den benötigten Hilfsvariablen und hat als Dimension den Rang des unterbestimmten Gleichungssystems $M\vec{a} = \vec{v}$.
Da nach Konstruktion alle Zeilen der erweiterten Matrix linear unabhängig sind, hat dieses Gleichungssystem eine eindeutige Lösung (falls die Nebenbedingung nicht unerfüllbar ist). Das Verfahren kann in [3] nachgesehen werden.
Ist die Zahl der Differenzvektoren größer als die Zahl der Freiheitsgrade, so kann es vorkommen, daß nicht alle Differenzvektoren innerhalb des Toleranzbereiches approximiert werden. Ist dies der Fall, so muß durch Graderhöhung des Segments die Anzahl der Freiheitsgrade vergrößert werden.

3.5.3 Neuberechnung der Projektionen

Im letztgenannten Fall können die Projektionsdaten mit Hilfe des nicht ausreichend approximierenden Segmentes neu berechnet werden. Hierzu wird das berechnete Segment dargestellt, alle für dieses Segment zu berücksichtigenden Punkte nach dem Verfahren aus 3.2 neu projiziert, das Segment wieder gelöscht und die Neuberechnung mit den neu gewonnenen Daten durchgeführt. Da dies eine bedeutende Rechenzeitverlängerung verursacht, wird diese Möglichkeit nur optional angeboten.

3.6 Darstellung der geänderten Flächen

Sind alle Segmente einer Fläche neu berechnet, wird die neue Fläche in die CATIA-Datenstruktur eingebunden und damit dargestellt. Die Darstellung wird von CATIA intern übernommen und wird hier nicht näher beschrieben (siehe auch [6], [8]).

3.7 Abspeichern der geänderten Daten

Die so gewonnenen neuen Flächendaten können nun wieder im VDAFS 2.0 Format in eine Datei geschrieben werden und stehen damit für die Weiterverarbeitung mit anderen Programmen, speziell CAD-Systemen zur Verfügung.

Kapitel 4

Funktionsaufbau in CATIA

In diesem Kapitel wird beschrieben, wie eine Applikation in das CAD-System CATIA integriert wird, und ein Überblick über die Struktur der neuen Applikation gegeben.

4.1 GII - Die Entwicklungsstruktur von CATIA-Funktionen

CATIA bietet die Möglichkeit über das **G***eometric* **I***nteractive* **I***nterface* (GII) eigene Programme als CATIA Funktionen einzubinden. Dazu muß neben den eigentlichen Programmen eine sogenannte FSD-Datei (**F**unction **S**tructure **D**efinition) erstellt werden, in der die Menüstruktur und die einzelnen Interaktionen definiert werden. Diese Datei gliedert sich in drei Abschnitte, die durch die Schlüsselworte `SECTION1`, `SECTION2` und `SECTION3` eingeleitet werden.
Ein Beispiel für eine FSD-Datei steht im Anhang B.

4.1.1 `FSD-SECTION1`: Globale Definitionen für die Funktion

Im ersten Abschnitt wird definiert, was für die gesamte Funktion gelten soll.Hierbei werden auch die Programme definiert, die zur Initialisierung aufgerufen werden müssen. Es können folgende Schlüsselworte verwendet werden:

- `HELPSET`: Hinter dem Schlüsselwort `HELPSET` steht der Name der Datei, in der die Meldungen stehen, die dem Benutzer angezeigt werden sollen.
- `ERRSET`: Hinter `ERRSET` steht der Name der Datei, in der CATIA nach Fehlermeldungen sucht.
- `TASK`: Mit `TASK` werden Prozeduren aufgerufen. Dabei ist zu beachten, daß bei Prozedurnamen eine Länge von sechs Buchstaben nicht überschritten werden darf. Prozeduren, die in der `SECTION1` stehen, werden immer einmal beim Einstieg in die Applikation ausgeführt.

- `CMDEX`: legt die Prozedur fest, die beim Verlassen eines Menüpunktes durchgeführt wird. Die `CMDEX`-Prozedur, die in der `SECTION1` festgelegt ist, wird immer dann aufgerufen, wenn bei der Definition des einzelnen Menüpunktes keine `CMDEX`-Prozedur angegeben wurde.

- `FUNEX`: Die `FUNEX`-Prozedur wird durchgeführt wenn die ganze Applikation verlassen wird. In diesem Fall wird nur die `FUNEX`-Prozedur ausgeführt, die momentane `CMDEX`-Prozedur wird nicht aufgerufen.

4.1.2 `FSD-SECTION2`: Definition der Interaktionstruktur

In der `SECTION2` werden Kommandos definiert. Dies sind die Funktionsabläufe, die bei den verschiedenen Menüpunkten durchgeführt werden sollen. Ein Kommando beginnt mit dem Schlüsselwort `COMMAND` gefolgt vom Kommandonamen. Danach kommen die einzelnen Interaktionen. Um eine Interaktionsstruktur aufzubauen, können folgende Schlüsselworte verwendet werden:

- `CMDEX`: Wird für ein Kommando eine andere Ausstiegsprozedur benötigt als die, die in der `SECTION1` definiert wurde, so kann dies durch die Angabe der `CMDEX`-Prozedur im Kommando erreicht werden.

- `FUNEX`: Ebenso kann eine eigene `FUNEX`-Prozedur angegeben werden. Die Austiegsprozeduren müssen vor der ersten `TASK` und dem ersten `INTLEVEL` stehen.

- `TASK`: Mit dem Schlüsselwort `TASK` werden auch in der `SECTION2` Prozeduren aufgerufen. Diese können entweder innerhalb eines Interaktionsschrittes stehen oder als Initialisierungsprozedur des Kommandos vor allen Interaktionen am Beginn des Kommandos.

- `INTLEVEL`: Durch dieses Schlüsselwort wird eine Interaktion abgefragt. Die Art der Interaktion muß genauer bestimmt werden. Dazu gibt es die Möglichkeiten `KEY, YES, PANEL, IND` und `SEL` (siehe unten). Es können Interaktionen ineinander geschachtelt werden oder parallel laufen. Bei zwei parallelen Interaktionen entscheidet die Aktion des Anwenders wie das Kommando weiter abgearbeitet wird. Bei Tastatureingaben hat `KEY` Vorrang vor `PANEL`.

- `KEY`: Hier wird eine Tastatureingabe erwartet. Die Auswertung dieser Eingabe erfolgt durch Angabe des Typs der erwarteten Eingabe, der aus `*CHAR,*REAL` und `*INT` zu wählen ist. Hinter der Typangabe steht ein Variablenname, in der die Eingabe gespeichert wird.

- `YES`: Diese Interaktion überprüft, ob die sogenannte YES-Taste gedrückt wird. Dies ist eine spezielle Taste auf der Tastatur, die von CATIA mit dieser Funktion belegt wird.

- `PANEL`: In CATIA gibt es die Möglichkeit Eingabefenster (`PANELS`) zu definieren. Soll auf eine Eingabe in einem solchen Fenster reagiert werden, muß das Wort `PANEL` angegeben werden. Außerdem ist es möglich, ein solches Fenster in einzelne Bereiche

(`STRUCTURE`) aufzuteilen. Soll nur auf einen bestimmten Bereich reagiert werden, so kann dies durch weitere Angabe von `*STR` und des Namens des Bereichs vorgeschrieben werden.

- `IND`: Soll der Benutzer eine Stelle auf dem Bildschirm auswählen, wird dies durch die Interaktion `IND` von ihm erwartet. Dies ist z.B. nötig, um Fenster vom Benutzer positionieren zu lassen.

- `SEL`: Durch `SEL` wird vom Benutzer die Auswahl eines geometrischen Elements erwartet. Will man hier eine Mehrfachauswahl zulassen, so stellt man das Wort `MSELW` nach. Sollen nur bestimmte Elementtypen zugelassen werden, so können diese durch ihre Typbezeichnungen angegeben werden (z.B. `*PT, *CRV, *SUR, ...`).

- `PROMPT`: Um zu einer Aktionsabfrage eine erklärende Meldung auszugeben, kann diese nach dem Wort `PROMPT` angegeben werden. Die maximale Länge der Meldungen hängt von der Anzahl der gerade möglichen Interaktionen ab.

- `TASKNO`: Hinter `TASKNO` wird die Prozedur angegeben, die durchgeführt werden soll, wenn der Benutzer eine Aktion rückgängig machen will. Diese Prozedur kann nur am Ende einer Interaktionskette wegfallen und muß dann durch `UNVAL` oder `ACCEPT` ersetzt werden.

- `UNVAL`: Diese Schlüsselwort kann nur am Ende einer Interaktionskette stehen. Es legt fest, daß die Aktionen ohne Bestätigung übernommen werden und nicht rückgängig gemacht werden können.

- `ACCEPT`: Hierdurch wird der Benutzer am Ende einer Interaktionskette aufgefordert, die gemachten Änderungen noch einmal zu bestätigen, bevor sie endgültig werden.

- `RESUME`: ist die schließende Klammer zu `INTLEVEL`. Jede Interaktionsbeschreibung muß mit `RESUME` enden. Ohne zusätzliche Zahl wird hierbei um einen Interaktionsschritt zurückgesprungen, das heißt, man ist in der Interaktionsstruktur wieder am Anfang der Aktion, die `RESUME` schließt. Die angebbare Zahl gibt die Anzahl der Schachteltiefen an, die zusätzlich zurückgesprungen werden soll.

Zusätzlich können an jeder Stelle der FSD-Datei Kommentarzeilen eingefügt werden die dann mit '\' beginnen müssen.
Die hier gegebene Beschreibung beschränkt sich auf das für die Implementierung der Applikation Notwendige. Für eine vollständige Beschreibung der Schlüsselworte und Struktur einer FSD-Datei siehe [5].

4.1.3 `FSD-SECTION3`: Die Menüdefinition

Im 3. Abschnitt der FSD-Datei wird die Menüstruktur definiert. Ein Menü ist umklammert von den Schlüsselworten `MENLEVEL` und `ENDMEN`. Die einzelnen Menüpunkte werden durch `ITEM` eingeleitet. Namen für Menüpunkte dürfen maximal acht Zeichen lang sein. Nach einem Menüeintrag kann ein weiteres Menü stehen. Maximal können vier Menüs

ineinander geschachtelt werden. Hat ein Menüpunkt kein weiteres Untermenü, so steht hinter dem Namen das Schlüsselwort `COMMAND` und ein maximal acht Zeichen langer *Kommandoname*. Ein Kommando ist eine in `SECTION2` definierte Interaktionsstruktur.

4.2 Der Menüaufbau

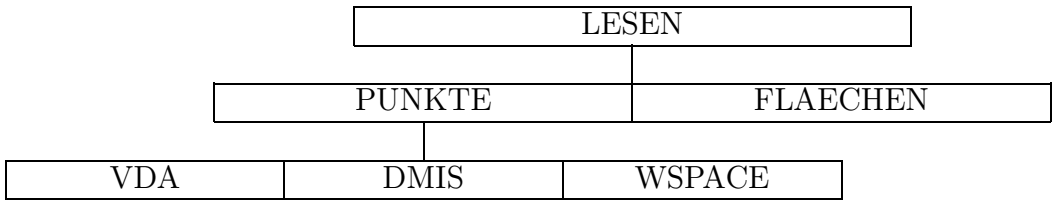
Die in Kapitel 3 beschriebene Vorgehensweise gibt schon eine erste Menüstruktur vor. So muß es einen Menüpunkt zum Einlesen der Daten geben, der in Untermenüs die verschiedenen Dateiformate enthält. Des weiteren wird ein Menüpunkt zur Auswahl der für die Projektionen relevanten Daten benötigt. Auch die Auswahl der für die Flächenänderung relevanten Flächen und Punkte und die Einstellung der Parameter müssen berücksichtigt werden. Weitere Menüpunkte müssen für die Projektion und die Flächenänderung vorgesehen werden. Den Gegenpol zum Lesen bildet der Punkt Schreiben, der die Sicherung der gewonnene Daten zuläßt. Zuletzt müssen die produzierten Geometrie-Elemente auch wieder gelöscht werden können. Wenn man dies alles berücksichtigt, kommt man zu einer Menüstruktur, die wie folgt aussieht:
Das Hauptmenü:

```
LESEN
AUSWAHL
VERGL
AENDER
SCHREIB
LOESCHEN
```

Das Menü ist hier so wiedergegeben wie es auch in der Applikation in CATIA erscheint. Die Namen sind sinnvoll gekürzt, da CATIA nur achtbuchstabige Menüworte zuläßt.

4.2.1 Hauptmenüpunkt LESEN

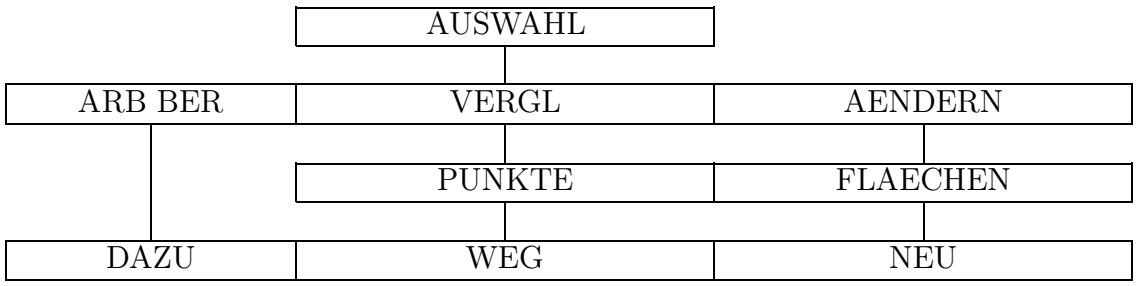
Unter diesem Punkt sind die Routinen zusammengefaßt, die Geometriedaten einlesen. Werden die verschiedenen Quellformate und Zielstrukturen berücksichtigt, ergibt sich eine Untermenüstruktur:



4.2.2 Hauptmenüpunkt AUSWAHL

Hier stehen die Funktionen, die sich mit der Auswahl von Geometrien befassen.
Unter ARB_BER werden die Flächen ausgewählt, die zum Arbeitsbereich gehören sollen. Dieser Bereich besteht aus allen Flächen, die für eine weitere Bearbeitung relevant sind.

Des weiteren können die Punkte und Flächen ausgewählt werden, die für den Vergleich von Meßpunkten und Flächen (VERGL) und die Flächenänderung (AENDERN) berücksichtigt werden sollen. Hierbei ist darauf zu achten, daß sich die Flächen im Arbeitsbereich befinden müssen.
Das ergibt die Untermenüstruktur:

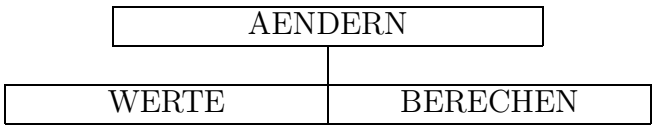


4.2.3 Hauptmenüpunkt VERGL

Dieser Punkt ist für die Durchführung des Vergleichs zwischen Punkten und Flächen vorgesehen. Das einzige was sich VERGL zuordnen ließe, wäre das Löschen der Vergleichsdaten. Da dieser Punkt aber unter LOESCHEN sinnvoll untergebracht ist, gibt es hier keine weiteren Unterpunkte.

4.2.4 Hauptmenüpunkt AENDERN

Die Prozeduren, die unter AENDERN zusammengefaßt sind, beschäftigen sich mit der eigentlichen Flächenänderung. Hierzu gehören die Einstellung der Parameter für die Änderung und die Berechnung selbst. Die Untermenüstruktur sieht wie folgt aus:

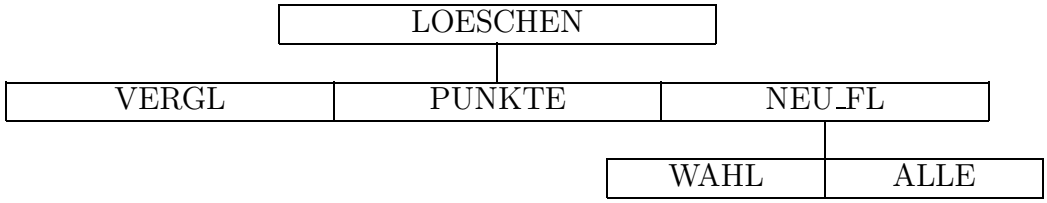


4.2.5 Hauptmenüpunkt SCHREIB

Das Schreiben der Daten bildet den Gegenpol zum Lesen. Auch hier wird in die verschiedenen Formate unterteilt. Es ergibt sich folgender Baum:

4.2.6 Hauptmenüpunkt LOESCHEN

Hier ist alles untergebracht, was benötigt wird, um die produzierten Geometrie-Elemente wieder zu löschen. Daher gliedert sich dieser Punkt folgendermaßen auf:



4.3 Die Datenstruktur

4.3.1 Strukturierte Datentypen

Beim Aufbau der Datenstruktur muß überlegt werden, welche Daten gespeichert werden müssen und wie der Zugriff auf diese Daten vollzogen werden soll. In diese Überlegungen geht auch mit ein, welche Daten in der CAD-Datenstruktur sowieso gehalten werden und deshalb nicht gesondert gespeichert werden müssen. Dies sind z.B. alle Daten der Geometrie-Elemente, die zur Darstellung derselben benötigt werden. Es wurden folgende Strukturen implementiert:

- `Verzeichnis_Eintrag`: In dieser einfach verketteten Liste werden die Dateinamen des aktuellen Verzeichnisses abgespeichert. Daher besteht diese Struktur aus einer Zeichenfolge und einem Zeiger auf das nächste Glied.

- `ppunkt`: Hier werden die Daten der projizierten Punkte gespeichert. Da diese Punkte nicht ständig dargestellt sind, müssen hier auch die Koordinaten vorhanden sein. Diese Struktur enthält eine Indexnummer (gültig, solange der Punkt dargestellt ist), einen Zeiger auf die Fläche, auf der er liegt, die Koordinaten, den Abstand zwischen dem projizierten und ursprünglichen Punkt und einen Zeiger auf den nächsten in der Liste.

- `punkt`: Dies ist die Liste der ausgewählten Punkte. Die Koordinaten dieser Punkte sind in der CAD-Datenstruktur gespeichert und müssen daher nicht in der Struktur sein. Es bleiben dann noch die Indexnummer des Punktes, die ursprüngliche Farbe, ein Zeiger auf den zugehörigen projizierten Punkt, ein Zeiger auf die Liste der Projektionen, ein Schalter, der angibt, ob dieser Punkt bei der Flächenänderung zu berücksichtigen ist und ein Zeiger auf den nächsten Punkt der Liste.

- `projektion`: Diese Struktur speichert die Projektionen, die zwar nicht den kürzesten Abstand haben, aber nur wenig von diesem abweichen. Sie enthält folgende Daten: einen Zeiger auf den zugehörigen Punkt, die Koordinaten des Differenzvektors und dessen Länge, einen Zeiger auf die Basisfläche, die Werte der Projektion in Flächenparametern, einen Zeiger auf die Liste der Projektionen für eine Fläche und einen Zeiger auf die Liste der Projektionen für einen Punktes.

- `face`: Die Daten für die begrenzten Flächen bestehen aus der Indexnummer, der ursprünglichen Farbe, einem Zeiger auf die Basisfläche, einem Zeiger für die Liste der Faces mit gleicher Basis und einen Zeiger auf die Liste aller Faces.

- `flaeche`: Für die Flächen sind die meisten Daten zu speichern. Dazu ist nötig: eine Indexnummer, die ursprüngliche Farbe, die Anzahl der Segmente, die Nummer der zugehörigen Änderungsfläche, ein Schalter, ob die Fläche zu ändern ist, der Flächenfehler, ein Schalter, ob die Fläche erzeugt wurde, ein Zeiger auf die gegebenenfalls vorhandenen Faces, ein Zeiger auf die Projektionen, die Eckpunktnummern, Zeiger auf die Nachbarschaften, die Stetigkeitswerte an den einzelnen Rändern, ein Zeiger auf die neu berechneten Segmente und ein Zeiger für die verkettete Liste.

- `patch`: Die Daten der einzelnen neu berechneten Segmente werden in der Struktur `patch` gespeichert. Diese besteht aus der Ordnung und der Matrix der Parameterdarstellung.

- `neue_flaeche`: Die Liste der dargestellten neuen Flächen besteht nur aus der Flächennummer und dem Zeiger auf die jeweils nächste.

- `ecke`: Hier werden Eck-Eck Übergänge gespeichert. Sie bestehen aus einem Zeiger auf die Fläche, auf der die Ecke liegt, der betroffenen Ecknummer und einem Zeiger auf den nächsten Eck-Eck Übergang.

- `kante`: Entsprechend zu den Ecken werden Kantenübergänge gespeichert. Dazu werden Anfangs- und Endpunkt der Kante, ein Zeiger auf die zugehörige Fläche, der Anfangs- und Endpunkt auf der zugehörigen Fläche, die Kantennummer auf der zugehörigen Fläche, ein Schalter, der angibt, ob die Ausrichtung gleich ist und ein Zeiger auf die nächste Kante benötigt.

- `uebergang`: Zuletzt werden noch für jedes Segment die Daten für die Übergänge benötigt. Diese bestehen aus den Zeigern auf die vier angrenzenden Segmente, die jeweils gültigen Kantennummern,den Schaltern, die angeben, ob die Kanten verschiedene Durchgangsrichtungen haben, und den Stetigkeiten, die an diesen Übergängen zu berücksichtigen sind.

Wie die Strukturen in der C-Syntax aussehen, ist aus Anhang C.1 zu ersehen. Die Datenstrukturen, die CATIA benutzt, sind in [6] beschrieben.

4.3.2 Globale Variabeln

Für die Listen und einige Einstellungen sind Variabeln notwendig, die während der gesamten Lebensdauer der Applikation Gültigkeit haben. Diese wurden als globale Variabeln implementiert, da CATIA das Hauptprogramm beim Einbinden der Applikation erstellt, und somit darauf kein Zugriff besteht. Die Deklaration der globalen Variabeln stehen in `jglobals.h` (siehe Anhang C.2). Definiert werden sie in `jinit.c`. Folgende globalen Variabeln werden benötigt:

Variable	Typ	Inhalt
`jlesepfad`	string	Der aktuelle Pfad zu den Dateien.
`jschreibedatei`	string	Name der Datei, in die geschrieben werden soll.
`jdateiliste`	pointer	Zeiger auf die Liste mit den Dateinamen des in `jlesepfad` gespeicherten Verzeichnisses.
`sollpunkte`	pointer	Zeiger auf die Liste der für den Vergleich ausgewählten Punkte.
`moment_sollpunkt1`	pointer	Zeiger auf den ersten Punkt der letzten Auswahl.
`moment_sollpunkt2`	pointer	Zeiger auf den letzten Punkt.
`punktespeicher`	pointer	Zeiger auf die abgewählten Punkte.
`ppunkte`	pointer	Zeiger auf die Liste der projizierten Punkte.
`flaechen`	pointer	Zeiger auf die Liste der Arbeitsflächen.
`momentflaeche1`	pointer	Zeiger auf die erste Fläche der letzten Auswahl.
`momentflaeche2`	pointer	Zeiger auf die letzte Fläche.
`flaechenspeicher`	pointer	Zeiger auf die abgewählten Flächen.
`faces`	pointer	Zeiger auf die Liste der Faces im Arbeitsbereich.
`momentface1`	pointer	Zeiger auf die erste Face der letzten Auswahl.
`momentface2`	pointer	Zeiger auf die letzte Face.
`facespeicher`	pointer	Zeiger auf die abgewählten Faces.
`neue_flaechen`	pointer	Zeiger auf die Liste der modifizierten Flächen.
`neueflaechenspeicher`	pointer	Zeiger auf die gelöschten modifizierten Flächen.
`stetig[4]`	integer	Stetigkeitsgrade der Übergänge in der Reihenfolge, wie in Kapitel 6 beschrieben.
`maxgrad[2]`	integer	Maximal zulässiger Grad der Flächen in den beiden Parameterrichtungen.
`maxerhöehung[2]`	integer	Maximal zulässige Graderhöhung.
`reihenfolge`	integer	Reihenfolge in der die Flächen bearbeitet werden sollen.
`berechnungsart`	integer	Schalter, der angibt, ob die Punkte nach jedem Iterationsschritt neu projiziert werden sollen.
`soll_ist_Vergleich`	integer	Schalter, der angibt, ob ein Vergleich durchgeführt wurde.

Kapitel 5

Beschreibung der Prozeduren

In diesem Kapitel werden die einzelnen Prozeduren mit ihren Parametern und Aufrufhierarchien beschrieben.

5.1 Von CATIA bereitgestellte Prozeduren

Zuerst werden die vom CAD-System bereitgestellten und in der Applikation verwendeten Routinen beschrieben. Diese Routinen sind in der Programmiersprache FORTRAN implementiert. Um FORTRAN-Routinen aus einem in der Programmiersprache C geschriebenen Programm heraus aufzurufen ist zu beachten, daß

- ein INTEGER$*$4 einem long entspricht,
- ein INTEGER$*$2 einem short entspricht,
- ein REAL$*$8 einem double entspricht,
- ein REAL$*$4 einem float entspricht,
- ein CHAR einem char entspricht,
- FORTRAN nur Zeiger auf Variabeln übergibt und
- ein Rückgabewert größer 0 einem Fehler entspricht, der zum Sprung an eine angegebene Fehlersprungmarke führt.

Die C-Syntax der FORTRAN-Routinen kann man der Header-Datei `catia.h` im Anhang C.3 entnehmen.
Hier wird nur kurz die Aufgabe der Prozeduren beschrieben, für eine genaue Beschreibung siehe [5], [6], [8], [9] und [11].

- **ggan2**

  Aufgabe: Schreibt Text in eine Struktur. Strukturen sind die Elemente, aus denen Fenster zusammengesetzt sind.
  Auftreten: ji1, jlese_Verzeichnis, jlese_Details, jsoll_ist_tabelle, jVDA_Kopffenster, jwertestruktur.
  Beschreibung: in [5] Seite 150.

- **gganfd**

  Aufgabe: Definiert ein Feld in einer Struktur für die Eingabe alphanumerischer Daten.
  Auftreten: ji1, jVDA_Kopffenster, jwertestruktur.
  Beschreibung: in [5] Seite 151.

- **ggarv**

  Aufgabe: Bindet eine Struktur an das Fenster, in dem sie dargestellt wird.
  Auftreten: ji1, ji2, ji5, jsoll_ist_tabelle, jVDA_Kopffenster.
  Beschreibung: in [5] Seite 188.

- **ggclst**

  Aufgabe: Schließt die momentan geöffnete Struktur.
  Auftreten: ji1, ja18, ja29, jsoll_ist_tabelle, jVDA_Kopffenster, jwertestruktur.
  Beschreibung: in [5] Seite 180.

- **ggdlst**

  Aufgabe: Löscht eine Struktur.
  Auftreten: ja2
  Beschreibung: in [5] Seite 180.

- **ggdrav**

  Aufgabe: Löst eine Struktur von einem Fenster. Die Struktur wird danach nicht mehr dargestellt.
  Auftreten: ja2
  Beschreibung: in [5] Seite 189.

- **ggeplb**

  Aufgabe: Positioniert den Zeiger innerhalb einer Struktur an einer Marke.
  Auftreten: ja2, ja18, ja29, ja30
  Beschreibung: in [5] Seite 183.

- **gginlb**

  Aufgabe: Erstellt eine Marke in einer Struktur.
  Auftreten: ji1, jVDA_Kopffenster, jwertestruktur.
  Beschreibung: in [5] Seite 182.

- **ggopst**

  Aufgabe: Öffnet eine Struktur zum Schreiben und Lesen der eingegebenen Daten.
  Auftreten: ji1, ja2, ja18, ji5, ja29, jsoll_ist_tabelle, jVDA_Kopffenster, jwertestruktur.
  Beschreibung: in [5] Seite 150.

- **ggoep**

  Aufgabe: Setzt den Zeiger, der auf das aktuelle Element einer geöffneten Struktur zeigt, um eine bestimmte Anzahl Elemente weiter.
  Auftreten: ja2,ja18,ja29
  Beschreibung: in [5] Seite 184.

- **ggpick**

  Aufgabe: Setzt das Pick-Attribut eines Elementes. Das Element ist dann selektierbar.
  Auftreten: ji1, ji5, jsoll_ist_tabelle, jVDA_Kopffenster, jwertestruktur.
  Beschreibung: in [5] Seite 173.

- **ggpkid**

  Aufgabe: Vergibt eine Nummer für ein anwählbares Element.
  Auftreten: ji1, ji5, jsoll_ist_tabelle, jVDA_Kopffenster, jwertestruktur.
  Beschreibung: in [5] Seite 174.

- **ggpl2**

  Aufgabe: Zeichnet eine Linie in einer Struktur.
  Auftreten: jsoll_ist_tabelle, jwertestruktur.
  Beschreibung: in [5] Seite 153.

- **ggqe**

  Aufgabe: Liest den Inhalt eines alphanumerischen Feldes.
  Auftreten: ja2, ja18, ja29.
  Beschreibung: in [5] Seite 197.

- **ggqste**

  Aufgabe: Überprüft, ob eine Struktur schon existiert.
  Auftreten: je4.
  Beschreibung: in [5] Seite 200.

- **ggsdec**

  Aufgabe: Definiert eine Scroll-Struktur.
  Auftreten: ji1, ji5, jsoll_ist_tabelle.
  Beschreibung: in [5] Seite 207.

- **ggslif**

  Aufgabe: Wählt den Scroll-Modus in einer Struktur.
  Auftreten: ji1, ji5, jsoll_ist_tabelle
  Beschreibung: in [5] Seite 208.

- **ggspag**

  Aufgabe: Wählt den Blätter-Modus in einer Struktur.
  Auftreten: ji1, ji5, jsoll_ist_tabelle.
  Beschreibung: in [5] Seite 209.

- **ggsqws**

  Aufgabe: Berechnet die Grösse des Fensters, in das die gesamte Struktur passen würde.
  Auftreten: ji1, ji5, jsoll_ist_tabelle.
  Beschreibung: in [5] Seite 213.

- **ggsups**

  Aufgabe: Aktualisiert eine Struktur, nachdem gescrollt wurde.
  Auftreten: ja1, ja17.
  Beschreibung: in [5] Seite 211.

- **ggsupv**

  Aufgabe: Aktualisiert ein Fenster, nachdem Strukturen verändert wurden.
  Auftreten: ja1, ja17.
  Beschreibung: in [5] Seite 212.

- **ggswin**

  Aufgabe: Initialisiert eine Scroll-Struktur.
  Auftreten: ji1, ji5, jsoll_ist_tabelle.
  Beschreibung: in [5] Seite 207.

- **ggtxci**

  Aufgabe: Setzt die Textfarbe auf einen bestimmten Wert.
  Auftreten: ji1, jlese_Verzeichnis, jlese_Details, jsoll_ist_tabelle, jVDA_Kopffenster, jwertestruktur.
  Beschreibung: in [5] Seite 163.

- **ggvch**

  Aufgabe: Aktiviert bzw. deaktiviert ein Fenster.
  Auftreten: je1, je4, je5, je9, ji1, jlese_Verzeichnis, jlese_Details, jsoll_ist_tabelle, jVDA_Kopffenster, jvergleich, jwertestruktur.
  Beschreibung: in [5] Seite 176.

- **ggvmp2**

  Aufgabe: Definiert, wie eine Struktur in das Fenster eingebaut wird.
  Auftreten: ji1, jsoll_ist_tabelle, jVDA_Kopffenster.
  Beschreibung: in [5] Seite 193.

- **ggvtr**

  Aufgabe: Definiert das Aussehen eines Fensters (Größe, Farbe, Ecken ...).
  Auftreten: ji1, jsoll_ist_tabelle, jVDA_Kopffenster.
  Beschreibung: in [5] Seite 178.

- **giccol**

  Aufgabe: Definiert die Farbe eines Elements.
  Auftreten: ja3, ja4, ja5, ja6, ja7, ja8, ja9, ja23, je4, jfunex, jneue_flaechen, jsoll_ist_tabelle, jn3, jn4.
  Beschreibung: in [8] Seite 46.

- **gicmas**

  Aufgabe: Macht den Hauptarbeitsbereich zum aktuellen Arbeitsbereich.
  Auftreten: ja31.
  Beschreibung: in [8] Seite 37.

- **gictxt**

  Aufgabe: Definiert die Darstellung eines Elements.
  Auftreten: ja7, ja9, ja20, ja21, ja22, ja23, je4, jfunex, jn3, jn11, jn12.
  Beschreibung: in [8] Seite 47.

- **gidtex**

  Aufgabe: Löscht einen Text an einem Element.
  Auftreten: je4, jn9.
  Beschreibung: in [8] Seite 71.

- **gieras**

  Aufgabe: Löscht ein Element.
  Auftreten: ja16, je3, je4, je8, jkante_teilen, j2kanten_teilen, jn1, jn2, jn9, jneue_flaechen.
  Beschreibung: in [6] Seite 111.

- **giiend**

  Aufgabe: Beendet eine Prozedur nach einem Fehler.
  Auftreten: ja16, ja29, ja31, jflaeche_aufbrechen, jneue_flaechen. jsoll_ist_tabelle, jVDA_Kopffenster, jwertestruktur.
  Beschreibung: in [6] Seite 95.

- **girad1**

  Aufgabe: Sucht nach dem Element zu einem Namen.
  Auftreten: ja30.
  Beschreibung: in [6] Seite 143.

- **girbas**

  Aufgabe: Gibt die Basis einer Face an.
  Auftreten: jbaue_flaeche.
  Beschreibung: in [8] Seite 8.

- **giride**

  Aufgabe: Gibt den Namen eines Elements an.
  Auftreten: jlese_Details.
  Beschreibung: in [6] Seite 142.

- **girmat**

  Aufgabe: Liest den Description-Block eines Elementes.
  Auftreten: ja16, ja29, jbaue_flaeche, jneue_flaechen, jsoll_ist_tabelle.
  Beschreibung: in [6] Seite 78.

- **girsiz**

  Aufgabe: Ermittelt die Größe eines Description-Blocks.
  Auftreten: ja29, jbaue_flaeche, jneue_flaechen.
  Beschreibung: in [6] Seite 77.

- **girtps**

  Aufgabe: Ermittelt den Typ eines Elements (Punkt, Linie, ...).
  Auftreten: ja11, ja12.
  Beschreibung: in [6] Seite 74.

- **girvis**

  Aufgabe: Ermittelt die momentane Darstellungsart eines Elements (Farbe, Form, ...).
  Auftreten: ja5, ja8, ja11.
  Beschreibung: in [8] Seite 48.

- **gisetl**

  Aufgabe: Sucht nach Elementen eines bestimmten Typs.
  Auftreten: ja8.
  Beschreibung: in [6] Seite 75.

- **gislim**

  Aufgabe: Sucht nach den Randkurven einer Face.
  Auftreten: jbaue_flaeche.
  Beschreibung: in [8] Seite 10.

- **giswsp**

  Aufgabe: Ermittelt alle zur Verfügung stehenden Arbeitsbereiche.
  Auftreten: jlese_Details.
  Beschreibung: in [8] Seite 38.

- **giwdet**

  Aufgabe: Erzeugt einen neuen Arbeitsbereich.
  Auftreten: ja31.
  Beschreibung: in [8] Seite 36.

- **giwedt**

  Aufgabe: Kopiert ein Element aus einem Arbeitsbereich in einen anderen.
  Auftreten: ja31.
  Beschreibung: in [8] Seite 42.

- **giwtex**

  Aufgabe: Schreibt einen Text an ein Element.
  Auftreten: jsoll_ist_tabelle.
  Beschreibung: in [8] Seite 69.

- **giwpt**

  Aufgabe: Erzeugt einen Punkt.
  Auftreten: ji1, jlese_Verzeichnis, jlese_Details, ja3, ja4.
  Beschreibung: in [8] Seite 61.

- **giwsur**

  Aufgabe: Erzeugt eine Fläche.
  Auftreten: ja11, jbaue_flaeche, jneue_flaechen, jpatch_darstellen.
  Beschreibung: in [8] Seite 62.

- **gmicgi**

  Aufgabe: Liefert das Ergebnis einer Fenstereingabe.
  Auftreten: ja1, ja17, ja18, jhole_Name.
  Beschreibung: in [5] Seite 99.

- **gmicsl**

  Aufgabe: Liefert den Punkt der in einem Fenster angewählt wurde.
  Auftreten: ja1, ja17.
  Beschreibung: in [5] Seite 100.

- **gmikey**

  Aufgabe: Liefert das Ergebnis einer Tastatureingabe.
  Auftreten: ja16, ja28, ja31.
  Beschreibung: in [5] Seite 91.

- **gmimse**

  Aufgabe: Wertet eine Mehrfachauswahl aus.
  Auftreten: ja8, ja9, ja21, ja22, ja25.
  Beschreibung: in [5] Seite 93.

- **gminte**

  Aufgabe: Gibt an, welche Interaktion durchgeführt wurde.
  Auftreten: ja18.
  Beschreibung: in [5] Seite 90.

- **gsdisf**

  Aufgabe: Analysiert eine Fläche.
  Auftreten: jneue_flaechen.
  Beschreibung: in [9] Seite 12.

- **gsoiof**

  Aufgabe: Stellt fest, ob ein Punkt in einer Face liegt.
  Auftreten: ja16.
  Beschreibung: in [9] Seite 9.

- **gsoisl**

  Aufgabe: Berechnet die Eckpunkte einer Fläche.
  Auftreten: jneue_flaechen.
  Beschreibung: in [9] Seite 125.

- **gsopos**

  Aufgabe: Projiziert einen Punkt auf eine Fläche.
  Auftreten: ja16, jneue_flaechen.
  Beschreibung: in [9] Seite 98.

- **gucele**

  Aufgabe: Ermittelt weitere Ergebnisse von Berechnungen, die mehr als ein Ergebnis haben.
  Auftreten: ja16, jkante_teilen, j2kanten_teilen, jneue_flaechen.
  Beschreibung: in [9] Seite 5.

- **gucsur**

  Aufgabe: Ermittelt die Parameterwerte eines Punktes auf einer Fläche.
  Auftreten: ja16, jkante_teilen, j2kanten_teilen, jneue_flaechen.
  Beschreibung: in [9] Seite 6.

- **guedis**

  Aufgabe: Setzt die Abbruchbedingungen.
  Auftreten: in allen Prozeduren, die Fehler erzeugen können.
  Beschreibung: in [5] Seite 57.

- **gueror**

  Aufgabe: Gibt eine CATIA-Fehlermeldung aus.
  Auftreten: in allen Prozeduren, die CATIA-Routinen aufrufen.
  Beschreibung: in [5] Seite 53.

5.2 Die Header-Dateien

In der Header-Datei `catia.h` stehen die Prototypen der verwendeten CATIA-Routinen wie sie nach der ANSI-NORM für C vorgeschrieben sind. In der Header-Datei `jstructs.h` sind die benötigten Strukturen definiert. In der Datei `jglobals.h` stehen die Deklarationen der globalen Variabeln, die für die Applikation benötigt werden. Die Datei `jfuncts.h` beinhaltet die Prototypen für alle Prozeduren, die zur Applikation gehören.

5.3 Eigene Prozeduren

Hier werden die Prozeduren beschrieben, die vom Hauptprogramm der Applikation aufgerufen werden. An diese Prozeduren werden keine Parameter übergeben und es werden auch keine Werte zurückgegeben. Eine Ausnahme bilden die beiden Prozeduren `jt1` und `jt2`, die 0 oder 1 zurückgeben.
Da die Prozedurnamen des Hauptprogrammes nur aus sechs Zeichen bestehen dürfen, wurde für die Namensvergabe folgende Konvention gewählt:

1. Der erste Buchstabe ist immer „j“, um Konflikte mit anderen Prozeduren zu vermeiden (CATIA-Routinen fangen alle mit „g“ an).

2. Der zweite Buchstabe gibt an, welche Aufgabe die Prozedur hat. Hierbei steht

   - a für ausführende Prozeduren, die eine Eingabe auswerten und Berechnungen durchführen,
   - e für Prozeduren, die beim Verlassen eines Kommandos ausgeführt werden,
   - f für Prozeduren, die beim Verlassen der Applikation ausgeführt werden,
   - i für Initialisierungsroutinen, die beim Einstieg in ein Kommando durchgeführt werden,
   - n für Routinen, die eine Eingabe rückgängig machen und
   - t für Testroutinen, die überprüfen, ob die Eingangsbedingungen erfüllt sind.

3. Danach folgen eine oder zwei Ziffern, die eine laufende Nummer angeben.

Die Prozeduren sind in alphabetischer Reihenfolge angegeben. Bei jeder Prozedur ist angegeben, in welcher Datei sie steht, welche Unterfunktionen und globalen Variabeln sie benutzt und welche Aufgabe sie hat. Außerdem wird eine kurze Beschreibung der Prozedur gegeben. Ist unter dem Punkt Unterfunktionen „keine“ angegeben, so werden von dieser Prozedur nur CATIA-Routinen aufgerufen.

- **ja1**

Datei:	jfenster.c.
Aufgabe:	Reagiert auf die Scroll-Knöpfe im Datei-Fenster und stellt den entsprechenden Teil dar.
Unterfunktionen:	keine.
globale Variabeln:	keine.

- **ja2**

Datei:	jfenster.c.
Aufgabe:	Setzt den Pfad auf das eingegebene Verzeichnis, liest die Dateien in diesem und schreibt die Dateiliste neu.
Unterfunktionen:	keine.
globale Variabeln:	jlesepfad, jdateiliste.

- **ja3**

Datei:	jlese_datei.c.
Aufgabe:	Liest die Punkte der angewählten VDA-Datei ein, nimmt sie in die Sollpunktliste auf und stellt sie dar.
Unterfunktionen:	jhole_Name, jlese_VDA_Zeile.
globale Variabeln:	jlesepfad, moment_sollpunkt1, moment_sollpunkt2, sollpunkte.

- **ja4**

Datei:	jlese_datei.c.
Aufgabe:	Liest die Punkte der angewählten DMIS-Datei ein, nimmt sie in die Sollpunktliste auf und stellt sie dar.
Unterfunktionen:	jhole_Name.
globale Variabeln:	jlesepfad, moment_sollpunkt1, moment_sollpunkt2, sollpunkte.

- **ja5**

Datei:	jlese_wspace.c.
Aufgabe:	Sucht im aktuellen Arbeitsbereich nach Punkten und nimmt diese in die Sollpunktliste auf.
Unterfunktionen:	keine.
globale Variabeln:	moment_sollpunkt1, moment_sollpunkt2, sollpunkte.

- **ja6**

Datei:	jlese_datei.c.
Aufgabe:	Liest die Flächen der angewählten VDA-Datei, nimmt diese in die Arbeitsflächenliste auf und stellt sie dar.
Unterfunktionen:	jhole_Name.
globale Variabeln:	flaechen, jlesepfad, momentflaeche1, momentflaeche2 .

- **ja7**

Datei:	jpunkte.c.
Aufgabe:	Entfernt alle Punkte aus der Sollpunktliste.
Unterfunktionen:	keine.
globale Variabeln:	moment_sollpunkt2, punktspeicher, sollpunkte.

- **ja8**

Datei:	jpunkte.c.
Aufgabe:	Nimmt angewählte Punkte zusätzlich in die Sollpunktliste auf.
Unterfunktionen:	keine.
globale Variabeln:	moment_sollpunkt1, mpment_sollpunkt2, punktspeicher, sollpunkte.

- **ja9**

Datei:	jpunkte.c.
Aufgabe:	Entfernt angewählte Punkte aus der Sollpunktliste.
Unterfunktionen:	keine.
globale Variabeln:	moment_sollpunkt1, moment_sollpunkt2, punktspeicher, sollpunkte.

- **ja10**

Datei:	jflaechen.c.
Aufgabe:	Entfernt alle Flächen aus der Flächenliste.
Unterfunktionen:	keine.
globale Variabeln:	faces, facespeicher, flaechen, flaechenspeicher, momentface2, momentflaeche2.

- **ja11**

Datei:	jflaechen.c.
Aufgabe:	Nimmt angewählte Flächen zusätzlich in die Flächenliste auf.
Unterfunktionen:	jbaue_Flaeche.
globale Variabeln:	faces, facespeicher, flaechen, flaechenspeicher, momentface1, momentface2, momentflaeche1, momentflaeche2.

- **ja12**

Datei:	jflaechen.c.
Aufgabe:	Entfernt angewählte Flächen aus der Flächenliste.
Unterfunktionen:	keine.
globale Variabeln:	faces, facespeicher, flaechen, flaechenspeicher, momentface1, momentface2, momentflaeche1, momentflaeche2.

- **ja13**

  Datei: jflaechen.c.
  Aufgabe: Setzt die Markierung als Änderungsfläche bei allen Flächen zurück.
  Unterfunktionen: keine.
  globale Variabeln: flaechen.

- **ja14**

  Datei: jflaechen.c.
  Aufgabe: Setzt bei den angewählten Flächen die Markierung als Änderungsfläche.
  Unterfunktionen: keine.
  globale Variabeln: flaechen.

- **ja15**

  Datei: jflaechen.c.
  Aufgabe: Setzt bei den angewählten Flächen die Markierung als Änderungsfläche zurück.
  Unterfunktionen: keine.
  globale Variabeln: flaechen.

- **ja16**

  Datei: jvergleich.c.
  Aufgabe: Führt den Vergleich zwischen den Punkten der Sollpunktliste und den als Vergleichsflächen markierten Flächen der Flächenliste durch.
  Unterfunktionen: jsoll_ist_tabelle.
  globale Variabeln: flaechen, jsoll_ist_vergleich, ppunkte, sollpunkte.

- **ja17**

  Datei: jfenster.c.
  Aufgabe: Reagiert auf die Scroll-Knöpfe im Soll-Ist-Vergleich-Fenster und stellt den entsprechenden Teil dar.
  Unterfunktionen: keine.
  globale Variabeln: keine.

- **ja18**

  Datei: jfenster.c.
  Aufgabe: Reagiert auf Eingaben im Werte-Fenster, setzt die Parameter auf die entsprechenden Werte und stellt das Fenster neu dar.
  Unterfunktionen: jwertestruktur.
  globale Variabeln: berechnungsart, maxerhoehung, maxgrad, neueflaechenfarbe, reihenfolge, stetig.

- **ja19**

  Datei: jneue_flaechen.c.
  Aufgabe: Berechnet die Flächenänderung zu den als Änderungsflächen markierten Flächen der Flächenliste, stellt die geänderten Flächen dar und nimmt sie in eine Liste auf.
  Unterfunktionen: jbernstein, jdiff_fuellen, jdiff_korrigieren, jflaeche_aufbrechen, jKante_eintragen, jloese_system, jmon2bez, jpatch_darstellen, jpatch_fuellen, jrandwerte, jTeilkante_eintragen, jvektor_fuellen, jwert_berechnen, jxmat_bauen.
  globale Variabeln: berechnungsart, flaechen, maxerhoehung, maxgrad, neue_flaechen, neueflaechenfarbe, reihenfolge, stetig.

- **ja20**

  Datei: jpunkte.c.
  Aufgabe: Entfernt bei allen Punkten die Markierung, die angibt, daß der Punkt bei der Flächenänderungsberechnung berücksichtigt werden soll.
  Unterfunktionen: keine.
  globale Variabeln: sollpunkte.

- **ja21**

  Datei: jpunkte.c.
  Aufgabe: Setzt bei den angewählten Punkten die Markierung, die angibt, daß der Punkt bei der Flächenänderungsberechnung berücksichtigt werden soll.
  Unterfunktionen: keine.
  globale Variabeln: sollpunkte.

- **ja22**

  Datei: jpunkte.c.
  Aufgabe: Entfernt bei den angewählten Punkten die Markierung, die angibt, daß der Punkt bei der Flächenänderungsberechnung berücksichtigt werden soll.
  Unterfunktionen: keine.
  globale Variabeln: sollpunkte.

- **ja23**

  Datei: jvergleich.c.
  Aufgabe: Löscht die durch den Punkt-zu-Fläche-Vergleich gewonnenen Daten.
  Unterfunktionen: keine.
  globale Variabeln: soll_ist_vergleich, sollpunkte.

- **ja24**

  Datei: jpunkte.c.
  Aufgabe: Löscht die Punkte der Sollpunktliste.
  Unterfunktionen: keine.
  globale Variabeln: moment_sollpunkt1, moment_sollpunkt2, sollpunkte.

- **ja25**

  Datei: jneue_flaechen.c.
  Aufgabe: Löscht die angewählten geänderten Flächen.
  Unterfunktionen: keine.
  globale Variabeln: neue_flaechen, neueflaechenspeicher.

- **ja26**

  Datei: jneue_flaechen.c.
  Aufgabe: Löscht alle geänderten Flächen.
  Unterfunktionen: keine.
  globale Variabeln: neue_flaechen.

- **ja27**

  Datei: jschreibe_datei.c.
  Aufgabe: Wertet die Auswahl eines Dateinamens zum Schreiben aus.
  Unterfunktionen: je1, jhole_Name, jVDA_Kopffenster.
  globale Variabeln: jschreibedatei.

- **ja28**

  Datei: jschreibe_datei.c.
  Aufgabe: Wertet die Eingabe eines Dateinamens zum Schreiben aus.
  Unterfunktionen: je1, jVDA_Kopffenster.
  globale Variabeln: jschreibedatei.

- **ja29**

  Datei: jschreibe_datei.c.
  Aufgabe: Schreibt die Daten der geänderten Flächen in eine VDA-Datei.
  Unterfunktionen: je1, ji1.
  globale Variabeln: jlesepfad, jschreibedatei, neue_flaechen.

- **ja30**

  Datei: jschreibe_datei.c.
  Aufgabe: Kopiert die geänderten Flächen in den ausgewählten Detail-Bereich.
  Unterfunktionen: jhole_Name.
  globale Variabeln: neue_flaechen.

- **ja31**

  Datei: jschreibe_datei.c.
  Aufgabe: Kopiert die geänderten Flächen in einen neuen Detail-Bereich mit dem eingegebenen Namen.
  Unterfunktionen: je9, ji5.
  globale Variabeln: neue_flaechen.

- **ja32**

  Datei: jflaechen.c.
  Aufgabe: Setzt die Markierung als Vergleichsfläche bei allen Flächen zurück.
  Unterfunktionen: keine.
  globale Variabeln: faces, flaechen.

- **ja33**

  Datei: jflaechen.c.
  Aufgabe: Setzt bei den angewählten Flächen die Markierung als Vergleichsfläche.
  Unterfunktionen: keine.
  globale Variabeln: faces, flaechen.

- **ja34**

  Datei: jflaechen.c.
  Aufgabe: Setzt bei den angewählten Flächen die Markierung als Vergleichsfläche zurück.
  Unterfunktionen: keine.
  globale Variabeln: faces, flaechen.

- **je1**

  Datei: jcmdex.c.
  Aufgabe: Schließt die Fenster zur Pfadeingabe und Dateiauswahl und löscht die Dateiliste.
  Unterfunktionen: keine.
  globale Variabeln: jdateiliste.

- **je2**

  Datei: jcmdex.c.
  Aufgabe: Löscht noch gespeicherte und nicht mehr benötigte Punktdaten.
  Unterfunktionen: keine.
  globale Variabeln: moment_sollpunkt1, moment_sollpunkt2, punktspeicher.

- **je3**

  Datei: jcmdex.c.
  Aufgabe: Löscht noch gespeicherte und nicht mehr benötigte Flächendaten.
  Unterfunktionen: keine.
  globale Variabeln: facespeicher, flaechenspeicher, momentface1, momentface2, momentflaeche1, momentflaeche2 .

- **je4**

  Datei: jcmdex.c.
  Aufgabe: Entfernt die Soll-Ist-Tabelle, die Punktnummern und die projizierten Punkte.
  Unterfunktionen: keine.
  globale Variabeln: sollpunkte, toleranz.

- **je5**

  Datei: jcmdex.c.
  Aufgabe: Schließt das Fenster für die Parametereingabe.
  Unterfunktionen: keine.
  globale Variabeln: keine.

- **je6**

  Datei: jcmdex.c.
  Aufgabe: Setzt die noch auf Zwischenwerte gesetzten Werte der Sollpunktliste um.
  Unterfunktionen: keine.
  globale Variabeln: sollpunkte.

- **je7**

  Datei: jcmdex.c.
  Aufgabe: Setzt die noch auf Zwischenwerte gesetzten Änderungsmarkierungen der Flächenliste um.
  Unterfunktionen: keine.
  globale Variabeln: flaechen.

- **je8**

  Datei: jcmdex.c.
  Aufgabe: Löscht die eventuell noch gespeicherten Daten von geänderten Flächen.
  Unterfunktionen: keine.
  globale Variabeln: neueflaechenspeicher.

- **je9**

  Datei: jcmdex.c.
  Aufgabe: Schließt das Fenster zur Detail-Auswahl.
  Unterfunktionen: keine.
  globale Variabeln: keine.

- **je10**

  Datei: jcmdex.c.
  Aufgabe: Setzt die noch auf Zwischenwerte gesetzten Vergleichsmarkierungen der Flächenliste um.
  Unterfunktionen: keine.
  globale Variabeln: faces, flaechen.
  Beschreibung:

- **jf1**

  Datei: jfunex.c.
  Aufgabe: Ruft die benötigten Ausstiegsroutinen auf.
  Unterfunktionen: je1, jfunex.
  globale Variabeln: keine.

- **jf2**

  Datei: jfunex.c.
  Aufgabe: Ruft die benötigten Ausstiegsroutinen auf.
  Unterfunktionen: je2, jfunex.
  globale Variabeln: keine.

- **jf3**

  Datei: jfunex.c.
  Aufgabe: Ruft die benötigten Ausstiegsroutinen auf.
  Unterfunktionen: je3, jfunex.
  globale Variabeln: keine.

- **jf4**

  Datei: jfunex.c.
  Aufgabe: Ruft die benötigten Ausstiegsroutinen auf.
  Unterfunktionen: je4, jfunex.
  globale Variabeln: keine.

- **jf5**

  Datei: jfunex.c.
  Aufgabe: Ruft die benötigten Ausstiegsroutinen auf.
  Unterfunktionen: je5, jfunex.
  globale Variabeln: keine.

- **jf6**

  Datei: jfunex.c.
  Aufgabe: Ruft die benötigten Ausstiegsroutinen auf.
  Unterfunktionen: je6, jfunex.
  globale Variabeln: keine.

- **jf7**

  Datei: jfunex.c.
  Aufgabe: Ruft die benötigten Ausstiegsroutinen auf.
  Unterfunktionen: je7, jfunex.
  globale Variabeln: keine.

- **jf8**

  Datei: jfunex.c.
  Aufgabe: Ruft die benötigten Ausstiegsroutinen auf.
  Unterfunktionen: je8, jfunex.
  globale Variabeln: keine.

- **jf9**

  Datei: jfunex.c.
  Aufgabe: Ruft die benötigten Ausstiegsroutinen auf.
  Unterfunktionen: je9, jfunex.
  globale Variabeln: keine.

- **jf10**

  Datei: jfunex.c.
  Aufgabe: Ruft die benötigten Ausstiegsroutinen auf.
  Unterfunktionen: je10, jfunex.
  globale Variabeln: faces, flaechen, moment_sollpunkt1, moment_sollpunkt2, momentface1, momentface2, momentflaeche1, momentflaeche2, neue_flaechen, neueflaechenspeicher, soll_ist_vergleich, sollpunkte.

- **jfunex**

  Datei: jfunex.c.
  Aufgabe: Löscht alle noch vorhandenen Daten.
  Unterfunktionen: keine.
  globale Variabeln: faces, flaechen, moment_sollpunkt1, moment_sollpunkt2, momentface1, momentface2, momentflaeche1, momentflaeche2, neue_flaechen, neueflaechenspeicher, soll_ist_vergleich, sollpunkte.

- **ji1**

  Datei: jfenster.c.
  Aufgabe: Baut die Fenster zur Dateiauswahl auf und öffnet sie.
  Unterfunktionen: jlese_Verzeichnis.
  globale Variabeln: jlesepfad.

- **ji2**

  Datei: jfenster.c.
  Aufgabe: Baut das Fenster zur Parameterauswahl auf und öffnet es.
  Unterfunktionen: jwertestruktur.
  globale Variabeln: keine.

- **ji3**

  Datei: jinit.c.
  Aufgabe: Bricht die Ausführung des Kommandos ab, falls Daten aus einem Soll-Ist-Vergleich vorhanden sind.
  Unterfunktionen: keine.
  globale Variabeln: soll_ist_vergleich.

- **ji4**

  Datei: jinit.c.
  Aufgabe: Bricht die Ausführung des Kommandos ab, falls keine Daten aus einem Soll-Ist-Vergleich vorhanden sind.
  Unterfunktionen: keine.
  globale Variabeln: soll_ist_vergleich.

- **ji5**

  Datei: jfenster.c.
  Aufgabe: Baut das Fenster zur Detail-Auswahl auf und öffnet es.
  Unterfunktionen: jlese_Details.
  globale Variabeln: keine.

- **ji6**

  Datei: jinit.c.
  Aufgabe: Bricht die Ausführung des Kommandos ab, falls die Flächenliste leer ist.
  Unterfunktionen: keine.
  globale Variabeln: flaechen.

- **jinit**

  Datei: jinit.c.
  Aufgabe: Initialisiert die globalen Variabeln.
  Unterfunktionen: keine.
  globale Variabeln: alle.

- **jn1**

  Datei: jlese_datei.c.
  Aufgabe: Löscht die eingelesen Sollpunkte wieder.
  Unterfunktionen: keine.
  globale Variabeln: moment_sollpunkt1, moment_sollpunkt2, sollpunkte.

- **jn2**

  Datei: jlese_datei.c.
  Aufgabe: Löscht die eingelesenen Flächen wieder.
  Unterfunktionen: keine.
  globale Variabeln: flaechen, momentflaeche1, momentflaeche2.

- **jn3**

  Datei: jpunkte.c.
  Aufgabe: Holt die abgewählten Sollpunkte wieder zurück.
  Unterfunktionen: keine.
  globale Variabeln: punktspeicher, sollppunkte, toleranz.

- **jn4**

  Datei: jpunkte.c.
  Aufgabe: Nimmt die zusätzlichen Sollpunkte wieder aus der Liste.
  Unterfunktionen: keine.
  globale Variabeln: moment_sollpunkt1, moment_sollpunkt2, sollpunkte.

- **jn5**

  Datei: jflaechen.c.
  Aufgabe: Holt die abgewählten Flächen wieder zurück.
  Unterfunktionen: keine.
  globale Variabeln: faces, facespeicher, flaechen, flaechenspeicher, momentface2, momentflaeche2.

- **jn6**

  Datei: jflaechen.c.
  Aufgabe: Nimmt die zusätzlichen Flächen wieder aus der Liste.
  Unterfunktionen: keine.
  globale Variabeln: faces, flaechen, momentface1, momentface2, momentflaeche1, momentflaeche2.

- **jn7**

  Datei: jflaechen.c.
  Aufgabe: Macht das Entfernen der Änderungsflächenmarkierung rückgängig.
  Unterfunktionen: keine.
  globale Variabeln: flaechen.

- **jn8**

  Datei: jflaechen.c.
  Aufgabe: Macht das Setzen der Änderungsflächenmarkierung rückgängig.
  Unterfunktionen: keine.
  globale Variabeln: flaechen.

- **jn9**

  Datei: jvergleich.c.
  Aufgabe: Macht einen Soll-Ist-Vergleich rückgängig.
  Unterfunktionen: keine.
  globale Variabeln: soll_ist_vergleich, sollpunkte.

- **jn10**

  Datei: jneue_flaechen.c.
  Aufgabe: Macht die Berechnung der geänderten Flächen rückgängig.
  Unterfunktionen: keine.
  globale Variabeln: neue_flaechen.

- **jn11**

  Datei: jpunkte.c.
  Aufgabe: Macht das Entfernen der Änderungsmarkierung bei Punkten rückgängig.
  Unterfunktionen: keine.
  globale Variabeln: sollpunkte.

- **jn12**

  Datei: jpunkte.c.
  Aufgabe: Macht das Setzen der Änderungsmarkierung bei Punkten rückgängig.
  Unterfunktionen: keine.
  globale Variabeln: sollpunkte.

- **jn13**

  Datei: jneue_flaechen.c.
  Aufgabe: Holt gelöschte Änderungsflächen zurück.
  Unterfunktionen: keine.
  globale Variabeln: neue_flaechen, neueflaechenspeicher.

- **jn14**

  Datei: jschreibe_datei.c.
  Aufgabe: Macht die Auswahl des Dateinamens zum Schreiben rückgängig.
  Unterfunktionen: je1, ji1.
  globale Variabeln: jschreibedatei.

- **jn15**

  Datei: jflaechen.c.
  Aufgabe: Macht das Entfernen der Vergleichsflächenmarkierung rückgängig.
  Unterfunktionen: keine.
  globale Variabeln: faces, flaechen.

- **jn16**

  Datei: jflaechen.c.
  Aufgabe: Macht das Setzen der Vergleichsflächenmarkierung rückgängig.
  Unterfunktionen: keine.
  globale Variabeln: faces, flaechen.

- **jt1**

  Datei: jvergleich.c.
  Aufgabe: Überprüft, ob Daten aus einem Soll-Ist-Vergleich vorhanden sind.
  Rückgabewert: 0 oder 1 (int).
  Unterfunktionen: keine.
  globale Variabeln: soll_ist_vergleich.

- **jt2**

  Datei: jvergleich.c.
  Aufgabe: Überprüft, ob keine Daten aus einem Soll-Ist-Vergleich vorhanden sind.
  Rückgabewert: 0 oder 1 (int).
  Unterfunktionen: keine.
  globale Variabeln: soll_ist_vergleich.

5.4 Hilfs-Prozeduren

In diesem Abschnitt werden die Hilfs-Prozeduren beschrieben, die von den im letzten Abschnitt genannten Prozeduren aufgerufen werden. Die Rückgabe von Werten ist meist durch die Übergabe eines oder mehrerer Zeiger als Parameter gelöst. Wird ein Wert direkt zurückgegeben, so steht der Typ dieses Wertes in Klammern hinter der Beschreibung der Rückgabe.
Die Beschreibungen der Prozeduren beschränkt sich auf die Aufgabe die sie erfüllen. Eine genauere Beschreibung findet sich in den jeweiligen Quelldateien.

- **j2kanten_teilen**

Datei:	jkanten.c.
Aufgabe:	Unterteilt eine gemeinsame Kante an den Segmentgrenzen beider Flächen, so daß Kanteneinträge nicht mehr über Segmentgrenzen hinweg gehen.
Parameter:	struct kante * Zeiger auf die erste Kante, struct kante * Zeiger auf die zugehörige Kante.
Rückgabewert:	keiner.
Unterfunktionen:	jkante_teilen.
globale Variabeln:	keine.

- **jbaue_Flaeche**

Datei:	jbaue_Flaeche.c.
Aufgabe:	Erzeugt eine Fläche als Basis zu einer planaren Face.
Parameter:	long Nummer der Face, long * Nummer der erzeugten Fläche.
Rückgabewert:	keiner.
Unterfunktionen:	keine .
globale Variabeln:	keine.

- **jbernstein**

Datei:	jbernstein.c.
Aufgabe:	Berechnet den Wert eines Bernsteinpolynoms für einen Parameterwert.
Parameter:	int Index des Polynoms, int Grad des Polynoms, double Parameterwert.
Rückgabewert:	Der Wert des Bernsteinpolynoms (double).
Unterfunktionen:	keine.
globale Variabeln:	keine.

- **jbez2mon**

Datei:	jmon2bez.c.	
Aufgabe:	Transformiert eine Flächendarstellung von Beziér- in Monom-Basis.	
Parameter:	double **	Matrix in Beziér-Basis,
	int	Ordnung in u-Richtung,
	int	Ordnung im v-Richtung.
Rückgabewert:	Geänderte Matrix.	
Unterfunktionen:	jueber, jvorzeichen.	
globale Variabeln:	keine.	

- **jdiff_fuellen**

Datei:	jdiff_fuellen.c.	
Aufgabe:	Füllt die Vektoren, die die zu approximierenden Abstände enthalten.	
Parameter:	struct projektion *	Zeiger auf die Projektionen,
	struct patch	aktuelles Segment,
	int	Nummer des Segments in u-Richtung,
	int	Nummer des Segments in v-Richtung,
	double *	Vektor der u-Parameterwerte,
	double *	Vektor der v-Parameterwerte,
	double **	Differenzenvektoren,
	long *	Feld der Punktnummern.
Rückgabewert:	Gefüllter Vektor der Differenzenvektoren.	
Unterfunktionen:	jbernstein.	
globale Variabeln:	keine.	

- **jdiff_korrigieren**

Datei:	jdiff_fuellen.c.	
Aufgabe:	Korrigiert die Differenzenvektoren anhand der schon festgelegten Teile des Segments.	
Parameter:	struct patch	aktuelles Segment
	int	Anzahl der Punkte
	double *	Vektor der u-Parameterwerte
	double *	Vektor der v-Parameterwerte
	double **	Differenzenvektoren
	double **	korrigierte Differenzenvektoren.
Rückgabewert:	korrigierte Differenzenvektoren.	
Unterfunktionen:	jbernstein.	
globale Variabeln:	keine.	

- **jflaeche_aufbrechen**

Datei:	jflaeche_aufbrechen.c.	
Aufgabe:	Teilt eine Fläche an einem Parameterwert auf und rechnet die zugehörigen Daten um.	
Parameter:	struct flaeche *	Fläche, die aufzubrechen ist,
	double **	Zeiger auf den Description-Block,
	double	Parameterwert, an dem aufzubrechen ist,
	int	Richtung, in der die Fläche gebrochen werden soll,
	struct projektion *	Projektionen, die zu ändern sind,
	int *	Anzahl der Segmente in u-Richtung,
	int *	Anzahl der Segmente in v-Richtung.
Rückgabewert:	geänderte Description, umgerechnete Projektionen, erhöhte Segmentanzahlen.	
Unterfunktionen:	jmon2bez, jbez2mon, jkante_teilen.	
globale Variabeln:	keine.	

- **jhole_Name**

Datei:	jlese_datei.c.
Aufgabe:	Holt den angewählten Namen aus der Dateiliste.
Parameter:	char * ausgewählter Name.
Rückgabewert:	ausgewählter Name.
Unterfunktionen:	keine.
globale Variabeln:	jdateiliste.

- **jKante_eintragen**

Datei:	jkanten.c.	
Aufgabe:	Trägt eine gefundene Kante in die Datenstruktur der betroffenen Flächen ein.	
Parameter:	struct flaeche *	Fläche 1
	int	betroffene Kantennummer der Fläche 1,
	struct flaeche *	Fläche 2,
	double	u-Wert des ersten Punktes auf Fläche 2,
	double	v-Wert des ersten Punktes auf Fläche 2,
	double	u-Wert des zweiten Punktes auf Fläche 2,
	double	v-Wert des zweiten Punktes auf Fläche 2.
Rückgabewert:	keiner.	
Unterfunktionen:	keine.	
globale Variabeln:	keine.	

- **jkante_teilen**

Datei:	jkanten.c.
Aufgabe:	Unterteilt eine gemeinsame Kante an den Segmentgrenzen der ersten Fläche, so daß Kanteneinträge nicht mehr über Segmentgrenzen hinweg gehen.
Parameter:	struct kante * Kante die zu teilen ist, struct kante * zugehörige Kante.
Rückgabewert:	keiner.
Unterfunktionen:	keine.
globale Variabeln:	keine.

- **jlese_Details**

Datei:	jlese_Verzeichnis.c.
Aufgabe:	Sucht nach vorhandenen Detail-Arbeitsbereichen in einem Modell und baut die Detail-Liste auf.
Parameter:	keine.
Rückgabewert:	keiner.
Unterfunktionen:	keine.
globale Variabeln:	jdateiliste.

- **jlese_VDA_Zeile**

Datei:	jlese_datei.c.
Aufgabe:	Liest eine Zeile aus der geöffneten VDA-Datei, die keine Kommentarzeile ist.
Parameter:	FILE * VDA-Datei, char * gelesene Zeile.
Rückgabewert:	Anzahl der gelesen Zeichen (int).
Unterfunktionen:	keine.
globale Variabeln:	keine.

- **jlese_Verzeichnis**

Datei:	jlese_Verzeichnis.c.
Aufgabe:	Liest die Dateien des angewählten Verzeichnisses und baut die Dateiliste auf.
Parameter:	keine.
Rückgabewert:	keiner.
Unterfunktionen:	keine.
globale Variabeln:	jdateiliste, jlesepfad.

- **jloese_system**

Datei:	jloese_system.c.	
Aufgabe:	Löst ein lineares Gleichungssystem und minimiert die Länge des Lösungsvektors.	
Parameter:	double **	Matrix des Systems,
	double *	Vektor des Systems,
	double *	Lösungsvektor des Systems,
	int	Dimension des Systems.
Rückgabewert:	Lösungsvektor.	
Unterfunktionen:	keine.	
globale Variabeln:	keine.	

- **jmon2bez**

Datei:	jmon2bez.c.	
Aufgabe:	Transformiert eine Flächendarstellung von Monom- in Beziér-Basis.	
Parameter:	double **	Matrix in Monom-Basis,
	int	Ordnung in u-Richtung,
	int	Ordnung im v-Richtung.
Rückgabewert:	geänderte Matrix.	
Unterfunktionen:	jueber, jvorzeichen.	
globale Variabeln:	keine.	

- **jpatch_darstellen**

Datei:	jpatch_darstellen.c.	
Aufgabe:	Stellt ein geändertes Segment als eigenständige Fläche dar.	
Parameter:	int	Nummer des Segments in u-Richtung,
	int	Nummer des Segments in v-Richtung,
	struct patch	Änderungsdaten des Segments,
	double *	Description-Block der Fläche,
	long *	Nummer der erzeugten Fläche.
Rückgabewert:	Nummer der erzeugten Fläche.	
Unterfunktionen:	jueber, jvorzeichen.	
globale Variabeln:	keine.	

- **jpatch_fuellen**

Datei:	jpatch_fuellen.c.	
Aufgabe:	Trägt die berechneten Änderungen in die Datenstruktur der Flaeche ein.	
Parameter:	struct patch	Segment das zu füllen ist,
	int *	Startwerte der Grade,
	int *	Endwerte der Grade,
	int *	Verkleinerung Grade,
	double *	Lösungsvektor der Berechnung.
Rückgabewert:	keiner.	
Unterfunktionen:	keine.	
globale Variabeln:	keine.	

- **jrandwerte**

Datei:	jrandwerte.c.	
Aufgabe:	Überträgt die Randbedingungen auf das aktuelle Segment.	
Parameter:	struct patch	aktuelles Segment,
	int	Rand der zu behandeln ist,
	struct patch	Segment von dem der Rand zu übernehmen ist,
	int	Nummer des zu übernehmenden Randes,
	int	zu berücksichtigende Stetigkeit,
	int	Richtung in die zu übernehmen ist,
	int	Schalter, der anzeigt, ob zu extrapolieren ist,
	double	unterer Parameterwert,
	double	oberer Parameterwert.
Rückgabewert:	keiner.	
Unterfunktionen:	keine.	
globale Variabeln:	keine.	

- **jsoll_ist_tabelle**

Datei:	jsoll_ist_tabelle.c.
Aufgabe:	Erstellt die Vergleichstabelle und färbt die Punkte.
Parameter:	keine.
Rückgabewert:	keiner.
Unterfunktionen:	keine.
globale Variabeln:	sollpunkte, toleranz.

- **jTeilkante_eintragen**

Datei:	jkanten.c.	
Aufgabe:	Trägt eine gefundene Kante, die nicht über die gesamte Fläche geht, in die Datenstruktur der entsprechenden Flächen ein.	
Parameter:	struct flaeche *	Fläche 1,
	unsigned char	Betroffene Ecke der Fläche 1,
	long *	Nummer des Segments in u-Richtung auf Fläche 1,
	double *	u-Parameterwert auf Fläche 1,
	long *	Nummer des Segments in v-Richtung auf Fläche 1,
	double *	v-Parameterwert auf Fläche 1,
	struct flaeche *	Fläche 2,
	unsigned char	Betroffene Ecke der Fläche 2,
	long *	Nummer des Segments in u-Richtung auf Fläche 2,
	double *	u-Parameterwert auf Fläche 2,
	long *	Nummer des Segments in v-Richtung auf Fläche 2,
	double *	v-Parameterwert auf Fläche 2.
Rückgabewert:	keiner.	
Unterfunktionen:	j2kanten_teilen.	
globale Variabeln:	keine.	

- **jueber**

Datei:	jueber.c.
Aufgabe:	Berechnet Binominialkoeffizienten (a über b).
Parameter:	int oberer Wert (a),
	int unterer Wert (b).
Rückgabewert:	Binominialkoeffizient (int).
Unterfunktionen:	keine.
globale Variabeln:	keine.

- **jVDA_Kopffenster**

Datei:	jVDA_Kopffenster.c.
Aufgabe:	Erstellt und öffnet das Fenster für den Kopf der VDA-Datei.
Parameter:	keine.
Rückgabewert:	keiner.
Unterfunktionen:	keine.
globale Variabeln:	jschreibedatei.

- **jvektor_fuellen**

Datei:	jvektor_fuellen.c.	
Aufgabe:	Erstellt die rechte Seite des Gleichungssystems.	
Parameter:	int *	Startwerte der Grade,
	int *	Endwerte der Grade,
	int *	Verkleinerung der Grade,
	struct patch	aktuelles Segment,
	double *	zu füllender Vektor,
	int	Anzahl der Punkte,
	double *	Vektor der u-Parameterwerte,
	double *	Vektor der v-Parameterwerte,
	double *	Differenzen der aktuellen Koordinate.
Rückgabewert:	gefüllter Vektor.	
Unterfunktionen:	jbernstein.	
globale Variabeln:	keine.	

- **jvorzeichen**

Datei:	jvorzeichen.c.
Aufgabe:	Liefert 1 für gerade Zahlen und -1 für ungerade Zahlen.
Parameter:	int Zahl.
Rückgabewert:	Vorzeichenwert (int).
Unterfunktionen:	keine.
globale Variabeln:	keine.

- **jwert_berechnen**

Datei:	jwert_berechnen.c.	
Aufgabe:	Berechnet den Wert einer berechneten Änderung an einer bestimmten Stelle.	
Parameter:	struct patch	aktuelles Segment,
	int *	Startwerte der Grade,
	int *	Endwerte der Grade,
	double	u-Parameterwert,
	double	v-Parameterwert,
	double *	Wert an dieser Stelle.
Rückgabewert:	Wert an der angegebenen Stelle.	
Unterfunktionen:	jueber.	
globale Variabeln:	keine.	

- **jwertestruktur**

Datei:	jwertestruktur.c.
Aufgabe:	Erstellt und öffnet das Fenster für die Parametereingabe.
Parameter:	keine.
Rückgabewert:	keiner.
Unterfunktionen:	keine.
globale Variabeln:	berechnungsart, maxerhöhung, maxgrad, neueflaechenfarbe, reihenfolge, stetig

- **jxmat_bauen**

Datei:	jxmat_bauen.c.	
Aufgabe:	Erstellt den Description-Block der geänderten Fläche.	
Parameter:	struct patch **	Änderungsdaten der einzelnen Segments,
	double *	zu berechnender Description-Block.
Rückgabewert:	berechneter Description-Block.	
Unterfunktionen:	jueber, jvorzeichen.	
globale Variabeln:	keine.	

Kapitel 6

Arbeiten mit der Funktion – Anwendungsbeschreibung

In diesem Kapitel wird beschrieben, wie die Funktion zur Flächenmodifikation anzuwenden ist, welche Eingaben erwartet werden und welche Fehler auftreten können. Die einzelnen Menüpunkte werden hier in der Reihenfolge beschrieben, in der sie im Menü aufgelistet sind. Beim Arbeiten mit der Funktion kann es zu anderen Reihenfolgen kommen.
Eine Kurzübersicht über die einzelnen Funktionalitäten der CATIA-Applikation findet sich im Anhang A.

6.1 Einlesen von Daten

Der Menüpunkt LESEN enthält die Prozeduren, die nötig sind, um Geometriedaten aus Dateien einzulesen. Es wird hier unterschieden, ob *Punkte* oder *Flächen* einzulesen sind. Punkte können in drei verschiedenen Formaten vorliegen:

- VDAFS 2.0 (VDA): Dies ist ein Format, das viele CAD-Systeme in Deutschland benutzen, um Geometriedaten auszutauschen. Außerdem genügt dieses Format einer Deutschen Industrie Norm (DIN 66301). Eine genaue Beschreibung dieses Formats findet sich in [12].

- DMIS 3.0 (DMIS): Dieses Format wird von Koordinatenmeßprogrammen benutzt, um Meßpunktdaten auszutauschen. Auch dieses Format ist genormt und zwar vom American National Standards Institute (ANSI). Dieses Format ist in [13] beschrieben.

- CATIA-Workspace (WSPACE): Hierbei werden alle Punkte übernommen, die im aktuellen Arbeitsbereich (Workspace) vorhanden sind. So können Punkte, die in anderen Formaten gegeben sind (z.B. IGES), über CATIA Schnittstellen eingelesen werden und dann in die Funktion übernommen werden. Dieser Punkt ist nur eine Erleichterung für den Anwender. Dasselbe kann auch mit AUSWAHL–PUNKTE–DAZU und Eingabe von `*pt` erreicht werden.

Für Flächen wird nur das Format VDAFS 2.0 unterstützt, da im DMIS-Format keine Flächenbeschreibungen möglich sind und im allgemeinen nicht alle dargestellten Flächen ausgewählt werden sollen.
Wurde VDA oder DMIS angewählt, so erscheinen auf dem Bildschirm zwei Fenster. Im rechten wird der momentane Pfad zu den Dateien angezeigt, der durch Eingabe in das Textfeld geändert werden kann, im linken werden alle Dateien dieses Verzeichnisses angezeigt. Sollten mehr Dateien im Verzeichnis sein, als auf den Bildschirm passen, wird zusätzlich am rechtem Rand ein Scroll-Balken angezeigt. Der Benutzer kann nun mit den Pfeiltasten die Anzeige blättern oder durch Anklicken eines Namens eine Datei zum Einlesen auswählen (siehe Abbildung 6.1).
Ist eine Datei zum Einlesen ausgewählt, so können folgende Fehlermeldungen erscheinen:

- `FORMATFEHLER IN VDA-DATEI` zeigt an, daß eine Datei als VDA-Datei eingelesen werden sollte, die nicht vollständig der Norm entspricht, aber dennoch interpretiert werden konnte (z.B. Fehlende Zeilennummern). Wenn diese Meldung erscheint, sollte allerdings geprüft werden, ob die Datei richtig interpretiert worden ist.

- Falls eine VDA-Datei keine Interpretation mehr zuläßt, wird dies durch `FEHLER IN VDA-DATEI` gemeldet.

- `KEINE VDA-DATEI` erscheint, wenn eine Datei als VDA-Datei eingelesen werden soll, die falsch beginnt (Schlüsselwort `HEADER` fehlt).

- Entsprechend erscheint bei DMIS-Dateien die Fehlermeldung `KEINE DMIS DATEI`, falls der Dateikopf nicht der Norm entspricht und

- `FEHLER IN DMIS-DATEI`, falls während des Einlesens ein Fehler auftritt.

Beim Einlesen der Punkte aus dem Arbeitsbereich können nur CATIA-interne Fehler auftreten, die in [5] beschrieben sind.
Nach dem Einlesen sind die Geometrie-Elemente dargestellt und für die Bearbeitung markiert. Die Fenster werden erneut dargestellt, um eine weitere Datei einzulesen.
Mit der NO-Taste kann das Einlesen einer Datei zurückgenommen werden. Dann werden alle Geometrien wieder gelöscht. Das Einlesen des Workspace kann nicht rückgängig gemacht werden, allerdings kann die Auswahl durch AUSWAHL-PUNKTE-NEU und YES-Taste wieder gelöscht werden.

6.2 Auswahl der relevanten Geometrien

Zunächst können die Flächen ausgewählt werden, die für das weitere Vorgehen benutzt werden sollen. Diese Flächen werden *Arbeitsbereich* genannt. Die Flächen, die für den Punkt-Fläche-Vergleich oder die Flächenänderung ausgewählt werden sollen, müssen alle im Arbeitsbereich liegen. Durch den Unterpunkt VERGL können die Punkte und Flächen ausgewählt werden, die beim Punkt-Fläche-Vergleich benutzt werden sollen. Dieser Punkt

steht nur zur Verfügung, wenn noch kein Vergleich durchgeführt oder ein bereits durchgeführter wieder gelöscht wurde. Unter FLAECHEN sind auch „begrenzte Flächen“, sogenannte Faces, anwählbar, allerdings müssen sie – wie auch die Flächen – bereits im Arbeitsbereich sein. Über den Unterpunkt AENDERN können die Flächen gewählt werden, die verändert werden sollen. Auch diese Flächen müssen bereits im Arbeitsbereich sein. Es können auch Flächen angewählt werden, für die keine Projektionen errechnet worden sind, um ein „Ausklingen“ einer Änderung zu ermöglichen. Für die Änderung sind keine Faces anwählbar, da diese nicht unabhängig von ihrer Basisfläche geändert werden können.
Wurde schon ein Vergleich durchgeführt, so können auch die Punkte gewählt werden, die in die Flächenänderung eingehen sollen.
Bei allen Menüpunkten erscheint ein Untermenü, das aus DAZU, WEG und NEU besteht. Durch DAZU werden die gewählten Elemente dem jeweiligen Bereich hinzugefügt, falls sie nicht schon enthalten sind. Mit WEG können Elemente aus einem Bereich entfernt werden, wobei das Entfernen einer Fläche aus dem Arbeitsbereich auch die Entfernung aus Vergleichs- und Änderungsbereich zur Folge hat. Bei NEU werden zuerst alle Elemente aus dem Bereich herausgenommen. Dies muß mit der YES-Taste bestätigt werden. Danach können wie bei DAZU Elemente angewählt werden, die zu diesem Bereich gehören sollen.
Bei der Auswahl der Geometrie-Elemente kann es zu folgenden Fehlerausgaben kommen:

- `PT SCHON IN LISTE`: Ein Punkt wurde doppelt angewählt. Die doppelte Auswahl wird ignoriert.
- `PT NICHT ANWAEHLBAR`: Ein Punkt soll aus einer Liste genommen werden, in der er nicht ist, oder ein Punkt, für den keine Vergleichsdaten vorhanden sind, soll in die Änderungsliste genommen werden. Die Auswahl wird ignoriert.
- `FLAECHE ZUR FACE SCHON IN LISTE`: Es wurde eine Face angewählt, deren Basisfläche auch schon angewählt wurde (kommt oft bei `*trap` vor). Die Auswahl einer Face dient dazu, den Projektionsbereich einer Fläche auf den Face-Bereich zu beschränken. Da die Punkte auf alle angewählten Flächenelemente projiziert werden, hat die Auswahl einer Face nur dann eine einschränkende Wirkung, wenn die Basisfläche nicht angewählt wird. Es sollte daher entweder die Face oder die Basisfläche aus der Liste genommen werden.
- `FLAECHE DOPPELT ANGEWAEHLT`: Es trat eine Mehrfachauswahl derselben Fläche auf. Die Mehrfachauswahl wird ignoriert.
- `FLAECHE NICHT ANWAEHLBAR`: Es wurde eine Fläche für den Änderungsbereich oder Vergleichsbereich angewählt, die nicht im Arbeitsbereich ist oder eine Fläche aus einem Bereich abgewählt, die nicht in diesem Bereich war. Die Auswahl wird ignoriert.

Die jeweils letzte Auswahl kann mit Hilfe der NO-Taste rückgängig gemacht werden.

6.3 Durchführen eines Soll-Ist-Vergleichs

Hiermit wird der Punkt-Fläche-Vergleich zwischen den ausgewählten Punkten und Flächen des Vergleichsbereichs durchgeführt. Vom Benutzer wird die Eingabe der Toleranz verlangt, die zur farblichen Kennzeichnung der Punkte innerhalb (grün) und außerhalb (rot) dieser Toleranz dient. Nach der Berechnung erscheint ein Fenster mit den Vergleichsdaten. Hier sind für jeden Punkt eine Nummer, seine Koordinaten, die Koordinaten der Projektion und die Abweichung angegeben (siehe Abbildung 6.2).
Im Modell sind die Punkte und ihre Projektionen (falls vorhanden) dargestellt und jeweils mit der entsprechenden Nummer versehen. Beim Verlassen des Menüpunktes werden das Fenster, alle Projektionen und die Nummern gelöscht. Die Punkte, die projiziert werden konnten, werden als für die Änderung ausgewählt markiert (Stern).
Ein weiterer Soll-Ist-Vergleich kann nur durchgeführt werden, wenn der alte explizit gelöscht wurde (Menüpunkt LOESCHEN), um eine Datenverwirrung zu vermeiden.
Durch Drücken der NO-Taste werden alle Vergleichsdaten und die Markierungen der Punkte (Farbe und Form) wieder entfernt.

6.4 Berechnen der Änderungsflächen

Der Menüpunkt AENDERN hat zwei Unterpunkte: WERTE und BERECHEN. Beim Punkt WERTE erscheint ein Fenster, in dem die Parameter für die Flächenänderung eingestellt werden können (siehe Abbildung 6.3). Dieses Fenster ist in fünf Sektionen geteilt. In der ersten Sektion kann die Reihenfolge der Flächenbearbeitung eingestellt werden. Drei Versionen stehen zur Wahl.

- *Nach Fehler absteigend*: Die Flächen werden in der Reihenfolge der größten Fehlersumme abgearbeitet, d.h. die Fläche mit der größten Fehlersumme wird zuerst bearbeitet. Dies hat zur Folge, das sich die Fehler globaler über den Flächenverband ausbreiten, also eine größere Auswirkung haben.

- *Nach Fehler aufsteigend*: Die Flächen mit dem kleinsten Fehlersumme bzw. ohne Projektionen werden zuerst bearbeitet und festgelegt. Dadurch bleiben die Fehler auf wenige Flächen begrenzt und die Änderung ist lokaler. Bei dieser Reihenfolge bleiben Flächen ohne Projektionen unverändert, müßten also gar nicht erst ausgewählt werden (Rechenzeitverkürzung).

- *Nach Bindungen absteigend*: Hier wird festgestellt, wieviele Freiheitsgrade für die Approximation der Änderungen zur Verfügung stehen. Dann werden die Flächen mit weniger Freiheitsgraden vor denen mit vielen berechnet. Dies hat zur Folge, daß bei einem zusammenhängenden Flächenbereich vom Rand nach innen gearbeitet wird. Diese Methode ist zu empfehlen, wenn die Fehler sehr gleichmäßig über den Änderungsbereich verteilt sind, da es in diesem Fall bei den anderen Methoden zu ungleichmäßigen Änderungen kommen kann.

In der zweiten Sektion können die Stetigkeiten eingestellt werden, die zu erhalten bzw. zu erzwingen sind. Hier werden vier Nachbarschaftsarten unterschieden:

- *Stetigkeit zwischen zwei Patches einer Fläche*: In CATIA sind Flächen aus einzelnen Segmenten aufgebaut, den sogenannten Patches. Hier kann nun eingestellt werden, welche Stetigkeit zwischen zwei solchen Segmenten erzeugt oder nicht verändert werden soll.

- *Stetigkeit zwischen zwei Flächen des Änderungsbereichs*: Hier wird angegeben wie die Stetigkeit zwischen zwei Flächen, die geändert werden, zu behandeln ist. Dabei wird bei der ersten Fläche, die behandelt wird, der Rand mitverändert und die zweite Fläche dann daran angepaßt.

- *Stetigkeit zwischen Änderungsbereich und Arbeitsbereich*: Die Stetigkeit, die am Rand einer Änderungsfläche nicht geändert werden darf, falls sie an eine Arbeitsbereichsfläche grenzt.

- *Stetigkeit zwischen Änderungsbereich und Modell*: Entsprechend. Die Stetigkeit, die am Rand einer zu ändernden Fläche erhalten bleiben soll, wenn sie an keine Änderungs- oder Arbeitsfläche grenzt, der Rand also in eine andere Fläche übergeht oder dort keine Fläche mehr ist.

Liegt ein Rand an mehreren Flächen unterschiedlichen Typs, so wird die höchste in Frage kommende Stetigkeit genommen.
Für die zu berücksichtigende Stetigkeit kann jeweils aus vier Werten gewählt werden:

- *keine*: Es werden keine Übergänge berücksichtigt, die Flächen können auseinanderklaffen.

- C^0-*stetig*: Die Flächen haben gemeinsame Randkurven.

- C^1-*stetig*: Die Flächen haben zusätzlich zur gemeinsamen Randkurve auch gleiche Tangenten an jedem Punkt dieser Randkurve.

- C^2-*stetig*: An den Randkurven gehen auch noch die Krümmungen stetig ineinander über.

Zusätzlich kann noch eingestellt werden, ob die Stetigkeiten innerhalb des Änderungsbereiches erzwungen werden sollen oder nicht.
Wird `erzwingen` gewählt, so werden die Anschlüsse „echt“ stetig berechnet.
Andernfalls wird eine zumindest geometrische Stetigkeit angenommen und diese erhalten. Konkreter: Falls C^1-Stetigkeit ausgewählt ist, spannen die Tangentialebenen der geänderten Flächen immer genau den Winkel auf, den die Tangentialebenen der ursprünglichen Flächen aufspannen. Bei geometrischer Tangentenstetigkeit ist dieser Winkel 0.
Der dritte Abschnitt des Fensters beschäftigt sich mit den Graden der Flächendarstellung. Hier können Maximalzahlen eingegeben werden. Zunächst wird der maximale Grad der Flächen getrennt nach u-, und v-Richtung angegeben. Da in CATIA nur Flächen bis zum Grad 15 vorgesehen sind, ist dies die Obergrenze (und Defaultwert). Dann kann noch eine maximale Graderhöhung (Standard: 3) eingegeben werden. Diese Zahl gibt an, wie stark die Grade der Fläche erhöht werden dürfen, falls dies zur Approximation der Differenzvektoren innerhalb der Toleranz nötig wird. Eine Angabe einer maximalen Erhöhung

kann sinnvoll sein, um unerwünschte Schwingungen der geänderten Fläche zu vermeiden. Allerdings sollte, um überhaupt eine Approximation zu ermöglichen, bei niedergradigen Flächen ein Graderhöhung zugelassen werden (vor allem bei C^2-Stetigkeit). Die Flächenberechnung wird bei nicht genügend guter Approximation solange mit höherem Grad wiederholt, bis einer der Maximalwerte erreicht wird.
Im vierten Abschnitt kann die *Neuberechnung der Projektionen nach jedem Iterationsschritt* eingeschaltet werden. Dann werden vor jeder Neuberechnung nach einer Graderhöhung die Projektionen neu berechnet. Dabei werden alle Punkte, die für diese Fläche relevant sind, auf die zuvor berechnete, aber nicht ausreichend gut approximierende Fläche projiziert. Mit den sich daraus ergebenden Differenzenvektoren wird die nächste Berechnung durchgeführt. Diese Option sollte mit Vorsicht angewandt werden, da das Projizieren sehr viel Rechenaufwand erfordert (siehe Kapitel 3).
Im letzten Abschnitt kann die Farbe eingestellt werden, in der die modifizierten Flächen dargestellt werden. Hier sind die Farben 1 bis 127 der CATIA-Palette möglich.
Die momentan gültigen Einstellungen sind hervorgehoben bzw. eingetragen. Sie können durch Anklicken der jeweils gewünschten Option geändert werden. Die Maximalwerte für die Grade werden durch Eingabe in das entsprechende Textfeld geändert (Abschließen mit Ausführen).
Durch Auswahl des Unterpunktes BERECHEN kann nun mit der YES-Taste die Berechnung der neuen Flächen gestartet werden. Hierzu werden die in WERTE eingestellten Parameter benutzt. Außerdem wird versucht, die im Soll-Ist-Vergleich eingegebene Toleranz einzuhalten. Die neu berechneten Flächen werden dargestellt. Wird bei der Flächenberechnung einer der maximalen Grade erreicht, ohne daß die Toleranz von allen Punkten eingehalten wurde, wird diese Fläche dargestellt und durch eine Fehlerausgabe `FLAECHE AUSSER TOLERANZ ABER MAXGRAD ERREICHT` darauf aufmerksam gemacht. Die Flächenänderung kann mehrmals hintereinander mit verschiedenen Parametereinstellungen ausgeführt werden. Bei jedem Durchlauf entstehen neue Flächen, die vorher berechneten werden nicht entfernt. So können verschiedene Verfahren miteinander verglichen werden.
Mit der NO-Taste können die jeweils zuletzt berechneten Flächen entfernt werden. Andere Flächen können im Menüpunkt LOESCHEN entfernt werden.
Die Flächenberechnung kann einige Zeit in Anspruch nehmen, da größere Gleichungssysteme gelöst und möglicherweise noch neue Projektionen berechnet werden müssen. Ein Abbruch der Berechnungen mit der INTERRUPT-Taste kann zu Inkonsistenzen in der Datenstruktur führen. Wenn eine solche bemerkt wird, wird dies mit der Fehlerausgabe `INKONSISTENTE DATENSTRUKTUR` gemeldet. Die Funktion sollte dann verlassen und neu aufgerufen werden.

6.5 Sichern der neu gewonnenen Daten

Mit dem Menüpunkt SCHREIB können die neu erzeugten Flächen gespeichert werden. Hierbei wird entweder in eine Datei geschrieben (VDA) oder ein sogenannter *Detail-Workspace* angelegt (DETAIL). Ein Detail-Workspace, kurz Detail genannt, ist ein abgeteilter Arbeitsbereich, in dem spezielle Elemente oder auch Teilmodelle gespeichert werden, von

denen dann eine oder mehrere Kopien in den Hauptarbeitsbereich übernommen werden können.
Beim Unterpunkt VDA werden die schon von LESEN bekannten Fenster zur Datei-Auswahl angezeigt. Hier kann wie bei LESEN eine Datei angewählt werden, die dann überschrieben wird. Zusätzlich kann ein neuer Datei-Name eingegeben werden. Wurde ein Datei-Name gewählt, erscheint ein weiteres Fenster, in dem die Daten für den Kopf der VDA-Datei eingegeben werden können. In diesem Fenster steht auch der Name der Datei, die geschrieben wird. Ist dieser nicht korrekt, so kann mit der NO-Taste zurückgegeangen und ein anderer gewählt werden. Wird die Fenstereingabe abgeschlossen, werden die Daten aller noch dargestellten neu berechneten Flächen in die Datei geschrieben. Die Darstellung wird dadurch nicht berührt.
Beim Unterpunkt DETAIL wird nur ein Fenster dargestellt, in dem die Namen der bisher angelegten Details aufgelistet sind. Auch hier kann eines zum Überschreiben ausgewählt oder ein neuer Detail-Name eingegeben werden (Zu beachten: Detail-Namen haben maximal 14 Zeichen – der eingegebene Name wird entsprechend gekürzt). Die noch vorhandenen neuen Flächen werden in das Detail kopiert.
Das Schreiben einer Datei oder eines Details kann nicht widerrufen werden.

6.6 Herstellen der ursprünglichen Situation

Der Menüpunkt LOESCHEN dient dazu, den Zustand des Modells beim Einstieg in die Funktion wiederherzustellen. Es gibt drei Gruppen von Elementen, die zu löschen sind:

- VERGL: Wird der Unterpunkt Vergleich mit der YES-Taste bestätigt, werden die Daten, die sich beim Soll-Ist-Vergleich ergeben haben, gelöscht. Dieses Kommando muß durchgeführt werden, bevor ein weiterer Soll-Ist-Vergleich (z.B. mit anderen Flächen) durchgeführt werden kann, da sich sonst Vergleichsdaten vermischen können.

  Wenn keine Vergleichsdaten vorhanden sind, wird eine Fehlermeldung angezeigt.

- PUNKTE: Wird der Unterpunkt PUNKTE mit der YES-Taste bestätigt, werden alle markierten Punkte gelöscht, um die über LESEN produzierten Punkte wieder zu entfernen. Dies geschieht beim Ausstieg aus der Funktion nicht automatisch.

- FLAECHEN: Hierdurch lassen sich die neu erzeugten Flächen wieder löschen und zwar entweder ALLE oder mit AUSWAHL nur ausgewählte. Das mit YES-Taste bestätigte Löschen aller Flächen ist endgültig, beim selektiven Löschen kann jeweils die letzte Auswahl rückgängig gemacht werden. Mit LOESCHEN–FLAECHEN lassen sich nur neu berechnete Flächen löschen. Entsprechend erscheint bei Auswahl einer anderen Fläche die Fehlermeldung `FLAECHE NICHT AUSWAEHLBAR`.

6.7 Beispiele

In diesem Abschnitt werden die Auswirkungen der einzelnen Parametereinstellungen anhand einiger einfacher Beispiele beschrieben.

Zunächst soll die ebene Fläche aus Abbildung 6.4 anhand der Punkte aus Abbildung 6.5 geändert werden. Gibt man hier keine Randbedingungen vor und läßt eine Graderhöhung (1 in beide Richtungen) zu, so ergibt sich die Fläche aus Abbildung 6.6. Wird keine Gradierhöhung zugelassen, so ergibt sich eine Ausgleichsebene (Abbildung 6.7).
Werden als Randbedingung C^0- oder C^1-Stetigkeit angegeben, so ergeben sich die Flächen in Abbildung 6.8 bzw. Abbildung 6.9. Diese Flächen können die Punkte im Randbereich nicht approximieren, da die Randkurven nicht geändert werden dürfen.
Der Unterschied zwischen C^1- und C^2-Stetigkeit wird an der Änderung der Ebene als Flächenverband aus zwei Flächen (Abbildung 6.10) deutlich. In Abbildung 6.11 wird die zweite Fläche nicht geändert, da sie die tangentiale Fortsetzung darstellt. Bei C^2-Stetigkeit wird die zweite Fläche allein durch die Stetigkeitsbedingungen gebildet, da für sie keine Punkte zu berücksichtigen sind (Abbildung 6.12).
Die Auswirkung der Reihenfolge der Bearbeitung wird bei der Änderung des Flächenverbandes aus Abbildung 6.13 deutlich. Wird hier bei C^0-Stetigkeit am Rand und C^1-Stetigkeit innerhalb des Verbandes die Änderung fehlerabsteigend durchgeführt, so führt dies nur bei den Flächen am Rand zu ungenügender Approximation (Abbildung 6.14), während bei fehleraufsteigender Behandlung die Randbedingungen auch bei inneren Flächen zu Abweichungen führen (Abbildung 6.15).
Wird bei der Änderung dieses Verbandes eine Graderhöhung größer als zwei zugelassen, so ergeben sich an den äußeren Flächen Schwingungen (Abbildung 6.16).
Mit der Applikation lassen sich auch Flächenverbände mit „T-Kreuzungen“ wie der in Abbildung 6.17 behandeln. Abbildung 6.18 zeigt die Änderung eines solchen Flächenanschlusses, Abbildung 6.19 die Änderung des gesamten Verbandes. Hierbei wurde innerhalb des Änderungsbereichs C^2-Stetigkeit und am Rand keine Stetigkeit erhalten. Ferner wurde eine Graderhöhung bis maximal Grad 5 zugelassen.
Da die Flächen in Abbildung 6.17 nur geometrische Stetigkeit aufweisen, ergibt sich beim Erzwingen der C^1-Stetigkeit ein abweichendes Ergebnis (Abbildung 6.20).
Zum Abschluß noch ein Beispiel aus einer Anwendung. Ein Schaltknauf wurde mit Hilfe eines Laserscanners digitalisiert. Aus den Meßpunktdaten wurden mit Hilfe von CAD-Funktionen Flächen generiert und diese durch Zwischenflächen C^1-stetig verbunden (Abbildung 6.21). Da die Verbindungsflächen nicht aus den Meßpunktdaten erzeugt wurden, treten hier Fehler auf. Bei der Zwischenfläche aus Abbildung 6.22 liegt die größte Länge der Differenzvektoren bei 0.64, die Summe der Längen aller 242 Vektoren ist 71.17. Da die Fläche nur etwas zu flach ist, empfiehlt es sich relativ wenige Punkte in der Mitte der Fläche zur Änderung zu verwenden. Die verwendeten Punkte sind aus Abbildung 6.23 ersichtlich. Unter der Bedingung, die C^1-Stetigkeit zu den beiden Nachbarflächen und innerhalb der Fläche beizubehalten, maximalem Grad 5 in beiden Parameterrichtungen und einer maximalen Graderhöhung von 2 Stufen in beiden Parameterrichtungen ergibt sich eine modifizierte Fläche, bei der die maximale Länge der Differenzvektoren bei 0.27 und die Summe bei 21.33 liegt (Abbildung 6.24). Bei anderen Parametereinstellungen werden teilweise Flächen berechnet, die ungewollte Schwingungen aufweisen. Ein Beispiel hierfür ist die Fläche aus Abbildung 6.25, die ensteht, wenn weitere Punkte angewählt werden und der maximale Grad auf 6 erhöht wird.
Am letzten Beispiel wird deutlich, daß das Aussehen der modifizierten Flächen sehr stark

von der Parameterwahl abhängt. Diese sollten daher sorgfältig gewählt werden. Es sollte nur verlangt werden, was benötigt wird, aber auch zu viel Freiheiten führen zu unerwarteten Ergebnissen.

6.8 Abbildungen

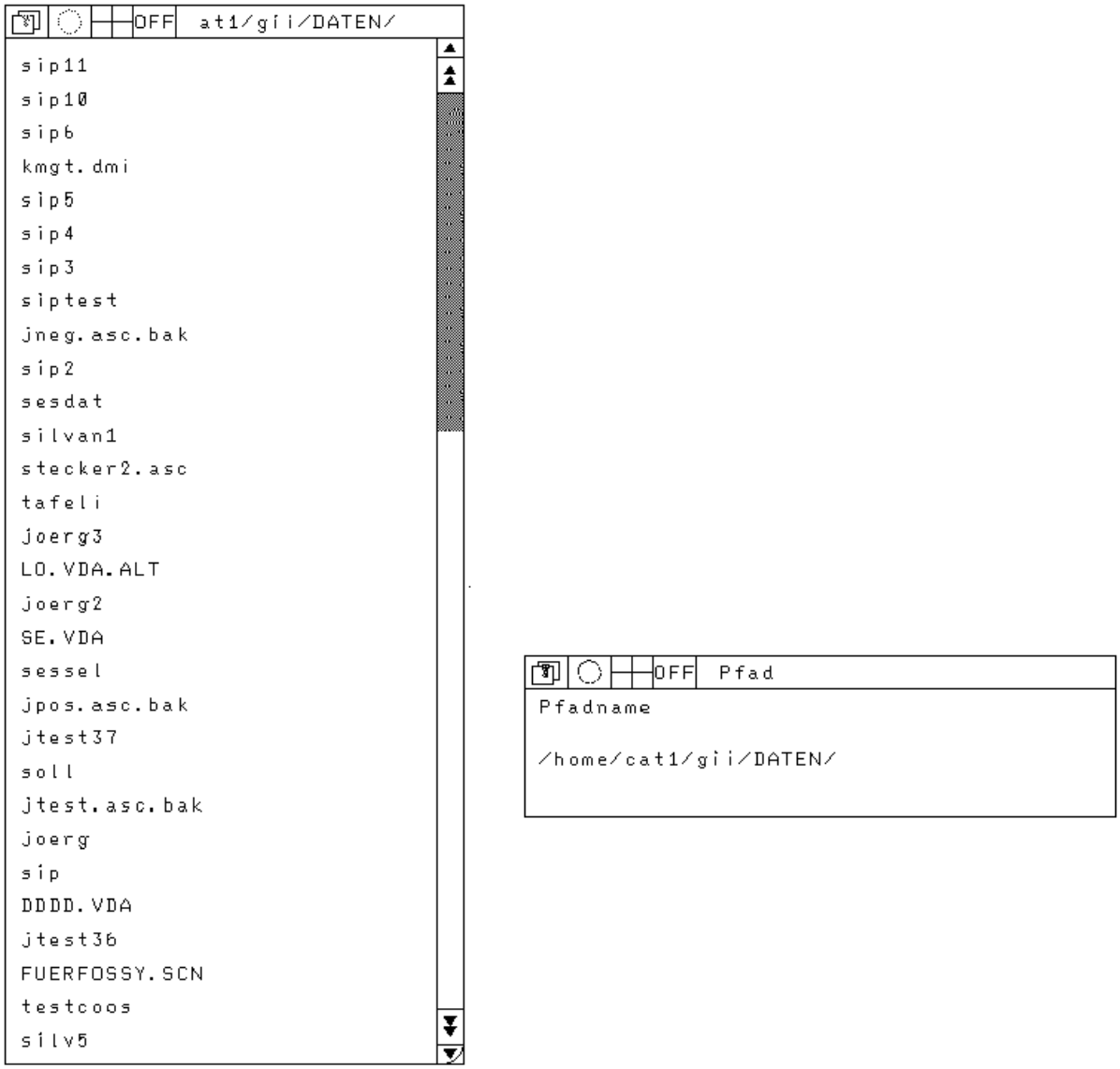


Abbildung 6.1: Fenster zur Dateiauswahl

OFF SOLL-IST-VERGLEICH

Markierte Punkte haben eine Abweichung > 0.1000

Nummer	Punkt X	Punkt Y	Punkt Z	Projekt.X	Projekt.Y	Projekt.Z	Abweichung
1	-100.000	-100.000	0.000	-100.000	-100.000	0.000	0.00000
2	-80.000	-100.000	10.800	-80.000	-100.000	0.000	10.80000
3	-60.000	-100.000	19.200	-60.000	-100.000	0.000	19.20000
4	-40.000	-100.000	25.200	-40.000	-100.000	0.000	25.20000
5	-20.000	-100.000	28.800	-20.000	-100.000	0.000	28.80000
6	0.000	-100.000	30.000	-0.000	-100.000	0.000	30.00000
7	20.000	-100.000	28.800	20.000	-100.000	0.000	28.80000
8	40.000	-100.000	25.200	40.000	-100.000	0.000	25.20000
9	60.000	-100.000	19.200	60.000	-100.000	0.000	19.20000
10	80.000	-100.000	10.800	80.000	-100.000	0.000	10.80000
11	100.000	-100.000	0.000	100.000	-100.000	0.000	0.00000
12	-100.000	-80.000	10.800	-100.000	-80.000	0.000	10.80000
13	-80.000	-80.000	21.600	-80.000	-80.000	0.000	21.60000
14	-60.000	-80.000	30.000	-60.000	-80.000	0.000	30.00000
15	-40.000	-80.000	36.000	-40.000	-80.000	0.000	36.00000
16	-20.000	-80.000	39.600	-20.000	-80.000	0.000	39.60000
17	0.000	-80.000	40.800	-0.000	-80.000	0.000	40.80000
18	20.000	-80.000	39.600	20.000	-80.000	0.000	39.60000
19	40.000	-80.000	36.000	40.000	-80.000	0.000	36.00000
20	60.000	-80.000	30.000	60.000	-80.000	0.000	30.00000
21	80.000	-80.000	21.600	80.000	-80.000	0.000	21.60000
22	100.000	-80.000	10.800	100.000	-80.000	0.000	10.80000
23	-100.000	-60.000	19.200	-100.000	-60.000	0.000	19.20000
24	-80.000	-60.000	30.000	-80.000	-60.000	0.000	30.00000
25	-60.000	-60.000	38.400	-60.000	-60.000	0.000	38.40000
26	-40.000	-60.000	44.400	-40.000	-60.000	0.000	44.40000
27	-20.000	-60.000	48.000	-20.000	-60.000	0.000	48.00000
28	0.000	-60.000	49.200	-0.000	-60.000	0.000	49.20000
29	20.000	-60.000	48.000	20.000	-60.000	0.000	48.00000
30	40.000	-60.000	44.400	40.000	-60.000	0.000	44.40000
31	60.000	-60.000	38.400	60.000	-60.000	0.000	38.40000
32	80.000	-60.000	30.000	80.000	-60.000	0.000	30.00000
33	100.000	-60.000	19.200	100.000	-60.000	0.000	19.20000
34	-100.000	-40.000	25.200	-100.000	-40.000	0.000	25.20000
35	-80.000	-40.000	36.000	-80.000	-40.000	0.000	36.00000
36	-60.000	-40.000	44.400	-60.000	-40.000	0.000	44.40000
37	-40.000	-40.000	50.400	-40.000	-40.000	0.000	50.40000
38	-20.000	-40.000	54.000	-20.000	-40.000	0.000	54.00000
39	0.000	-40.000	55.200	-0.000	-40.000	0.000	55.20000
40	20.000	-40.000	54.000	20.000	-40.000	0.000	54.00000
41	40.000	-40.000	50.400	40.000	-40.000	0.000	50.40000
42	60.000	-40.000	44.400	60.000	-40.000	0.000	44.40000
43	80.000	-40.000	36.000	80.000	-40.000	0.000	36.00000
44	100.000	-40.000	25.200	100.000	-40.000	0.000	25.20000
45	-100.000	-20.000	28.800	-100.000	-20.000	0.000	28.80000

Abbildung 6.2: Fenster mit Vergleichsdaten

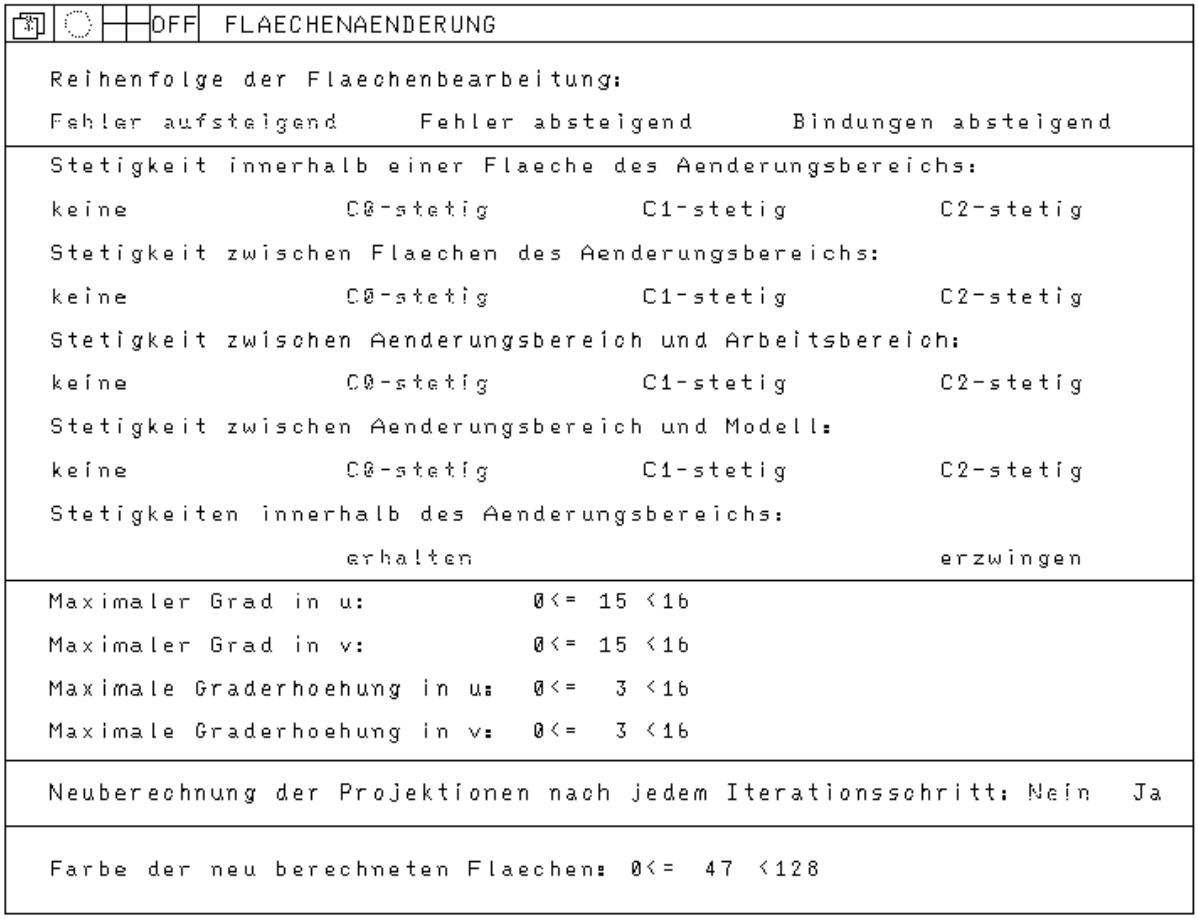


Abbildung 6.3: Fenster zur Parameterauswahl

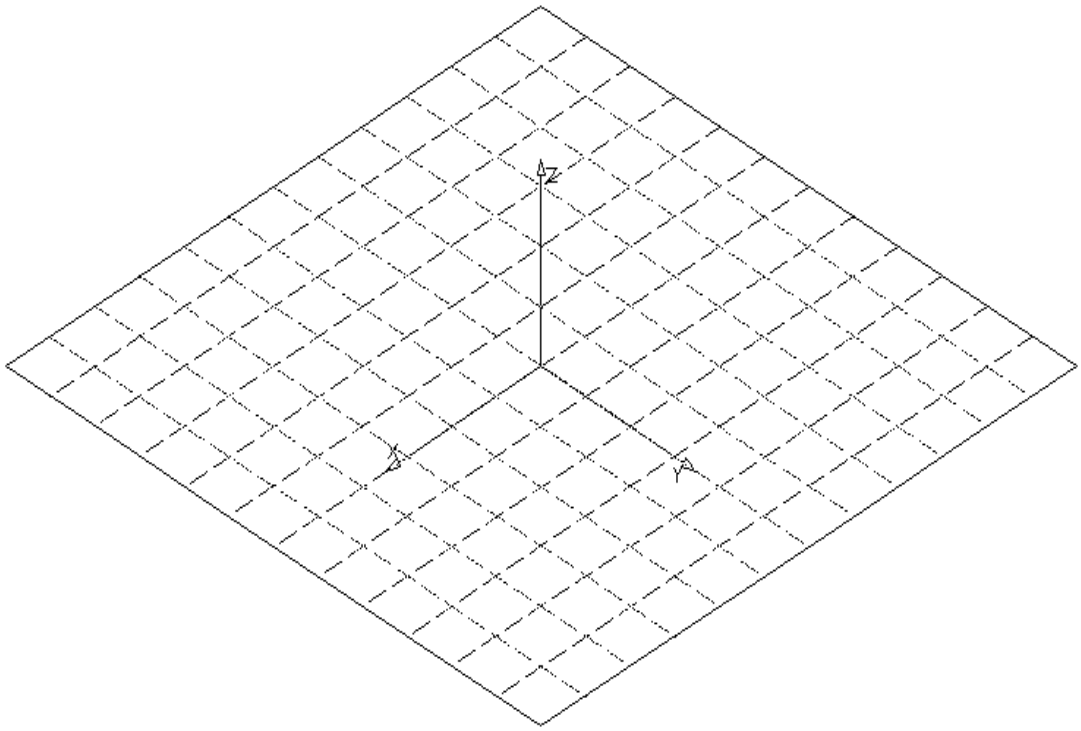


Abbildung 6.4: Ebene als parametrisierte Fläche vom Grad 1,1

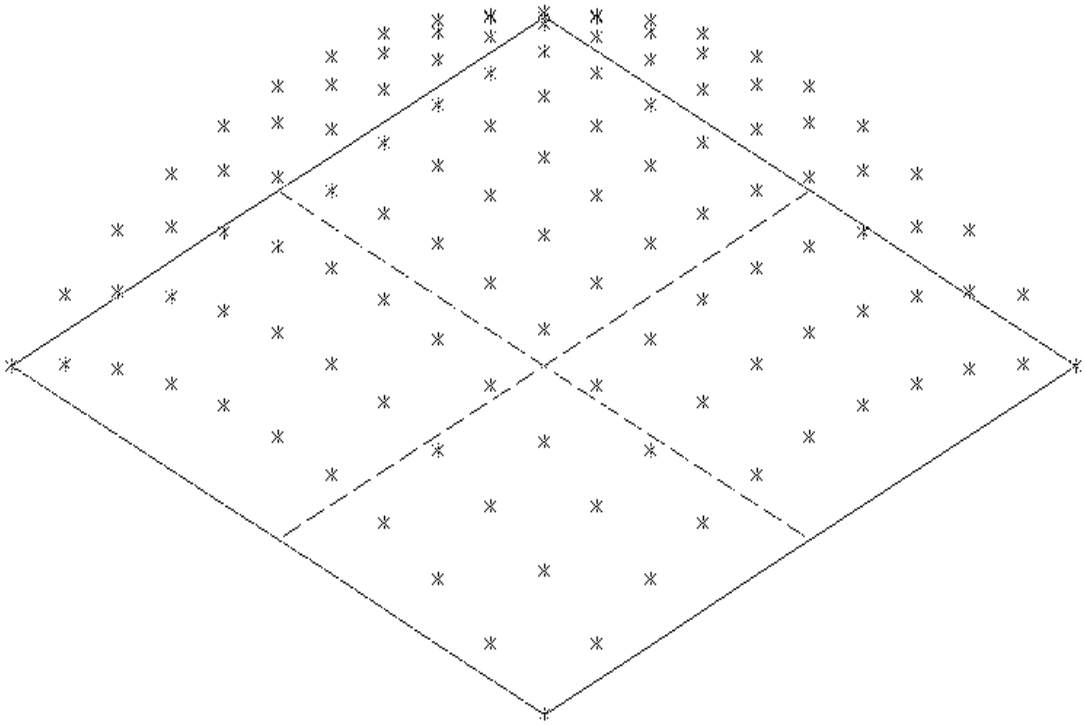


Abbildung 6.5: Punkte die zur Modifikation verwendet werden

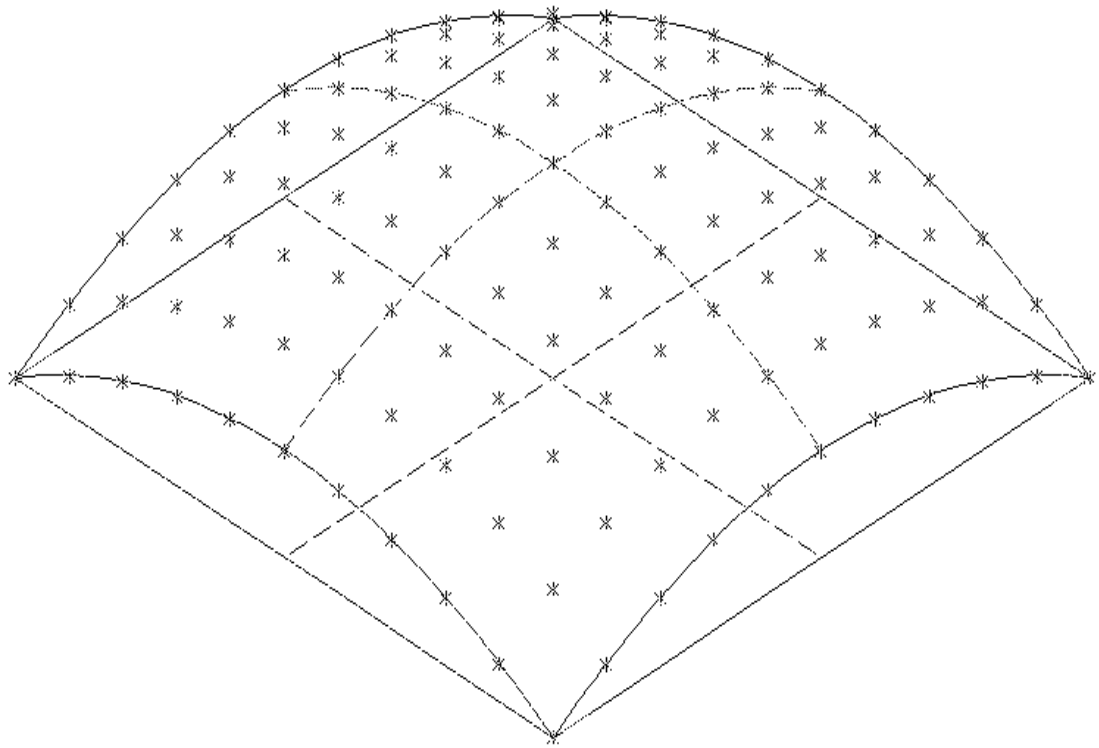


Abbildung 6.6: Änderung der Fläche aus 6.4 ohne Randbedingungen

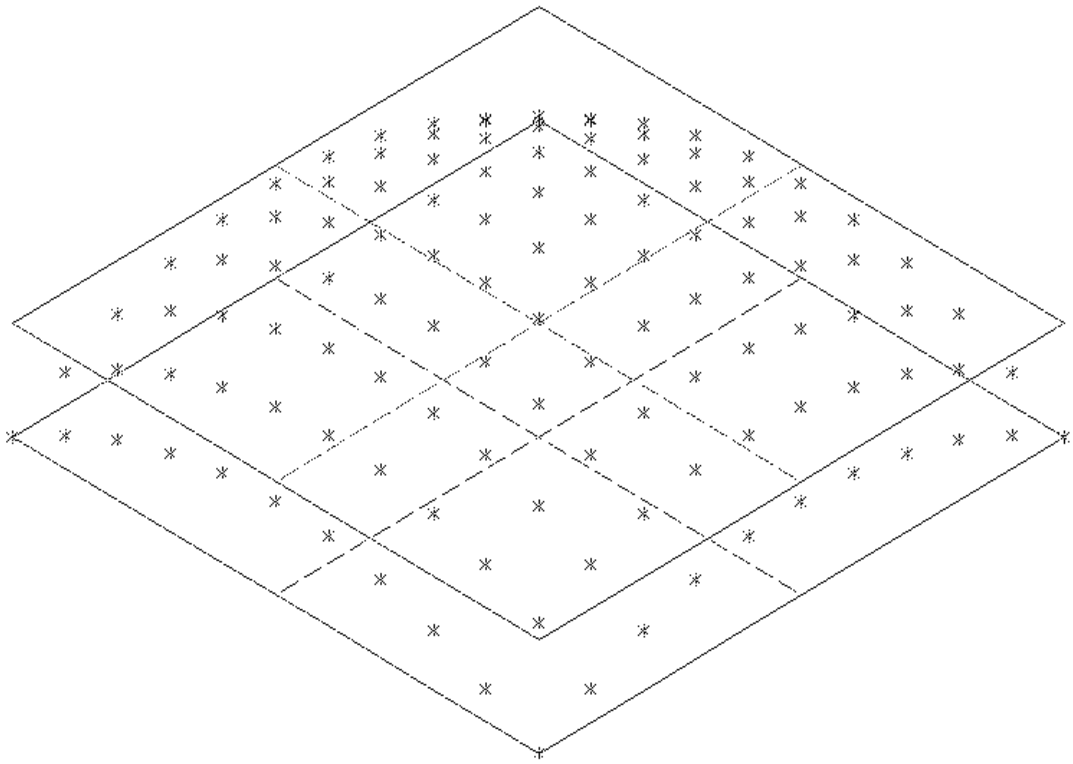


Abbildung 6.7: Änderung der Fläche aus 6.4 ohne Graderhöhung (Ausgleichsebene)

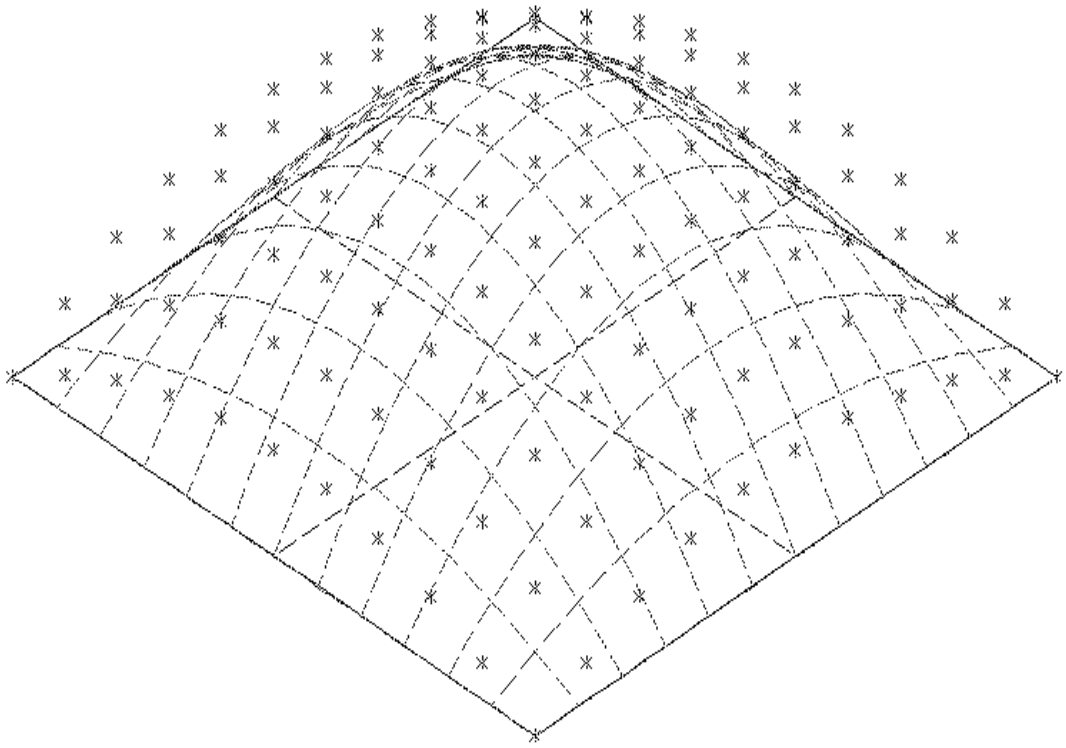


Abbildung 6.8: Änderung der Fläche aus 6.4 mit C^0-stetiger Erhaltung der Ränder

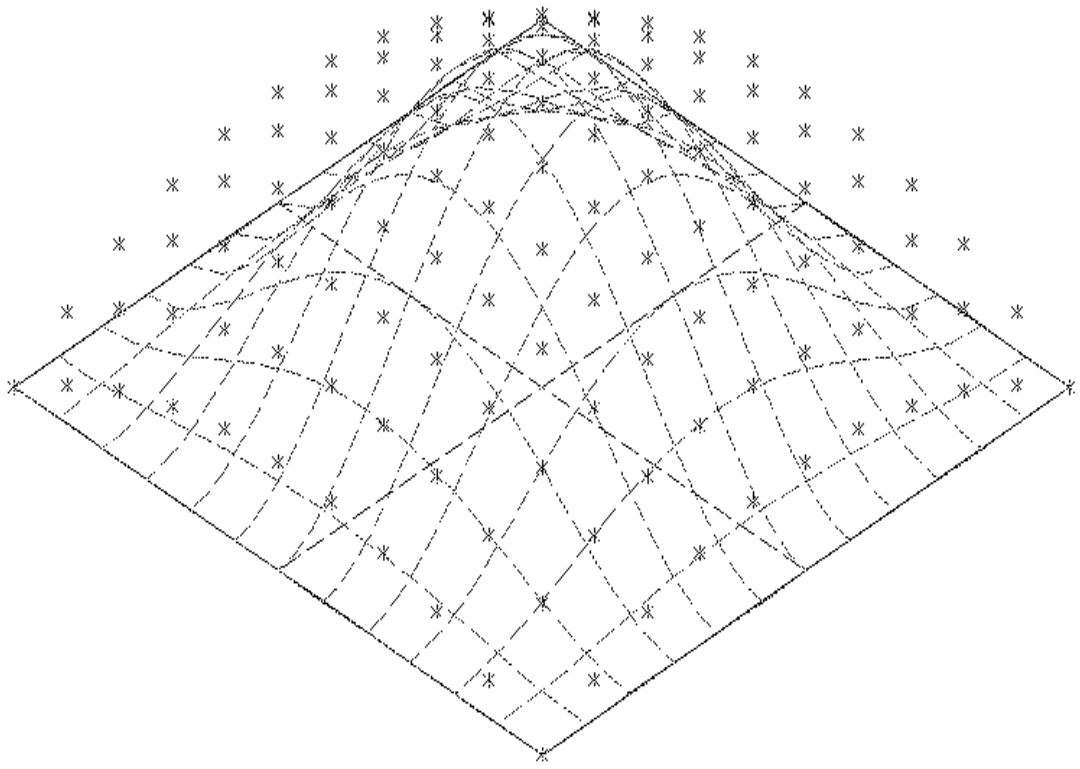


Abbildung 6.9: Änderung der Fläche aus 6.4 mit C^1-stetiger Erhaltung der Ränder

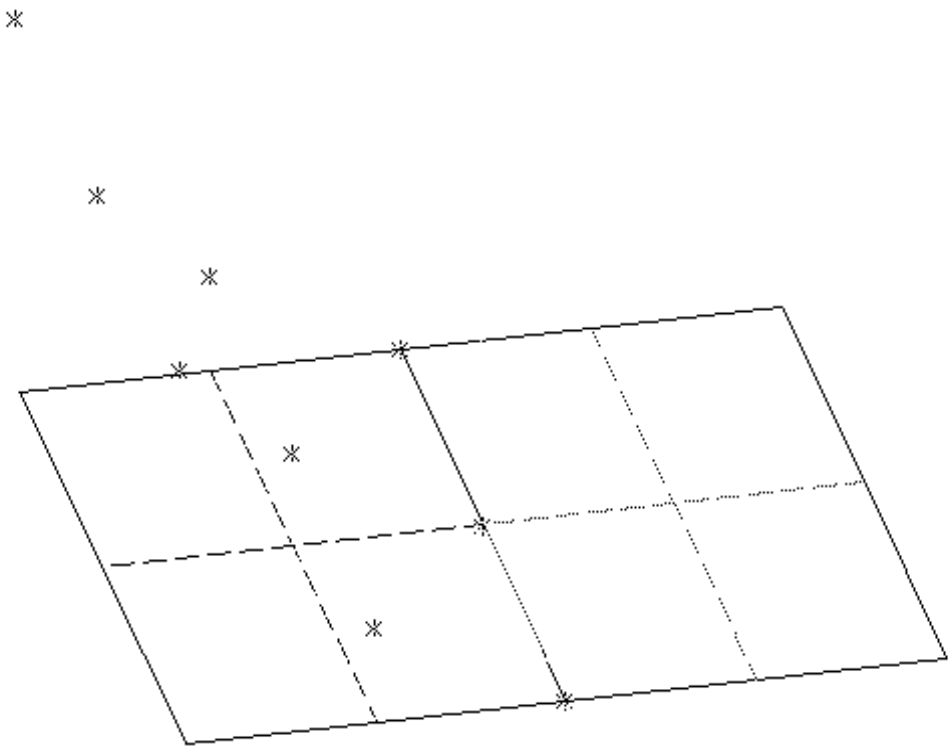


Abbildung 6.10: Ebene als Flächenverband aus zwei Flächen

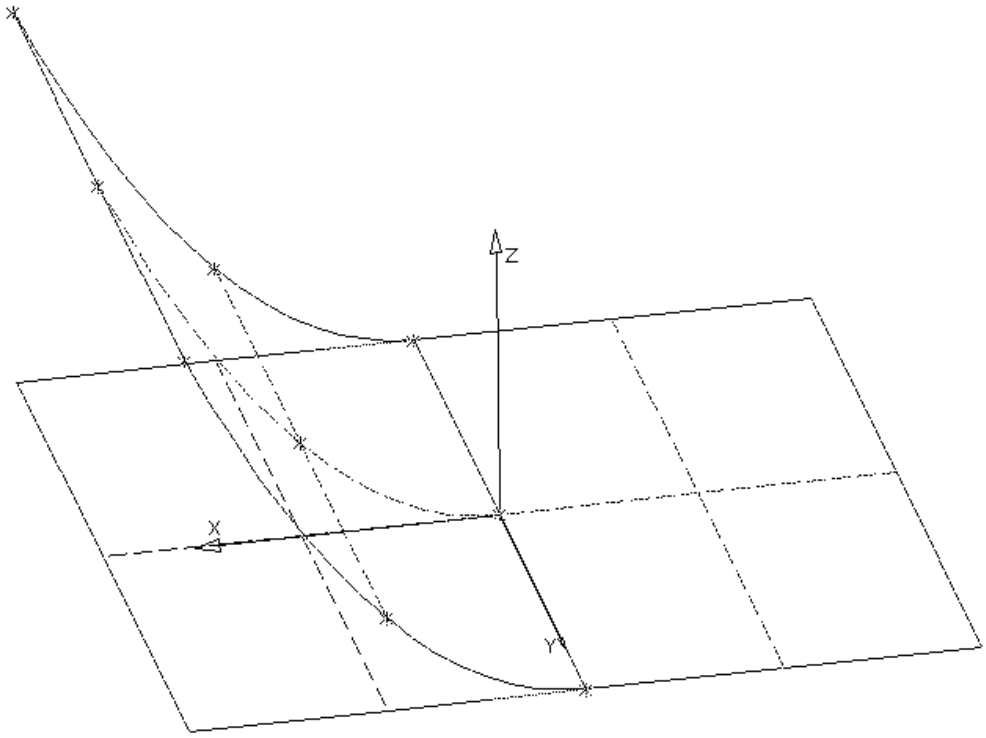


Abbildung 6.11: Änderung des Verbandes aus 6.10 mit C^1-stegigem Übergang

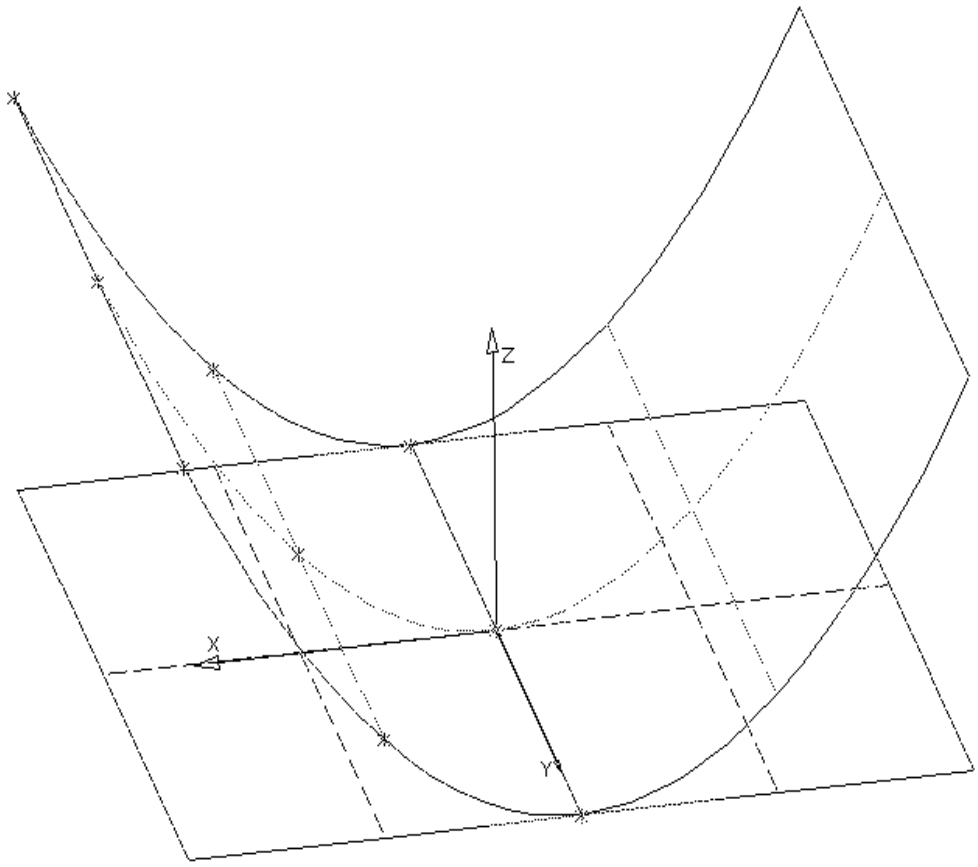


Abbildung 6.12: Änderung des Verbandes aus 6.10 mit C^2-stegigem Übergang

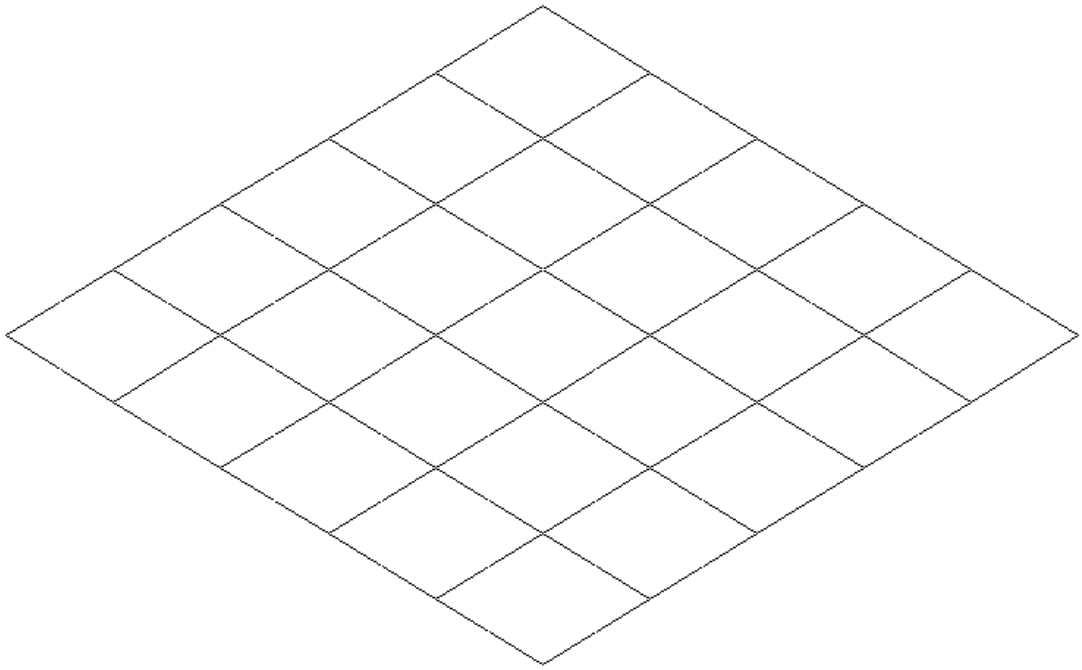


Abbildung 6.13: Ebene als Flächenverband aus 25 Flächen

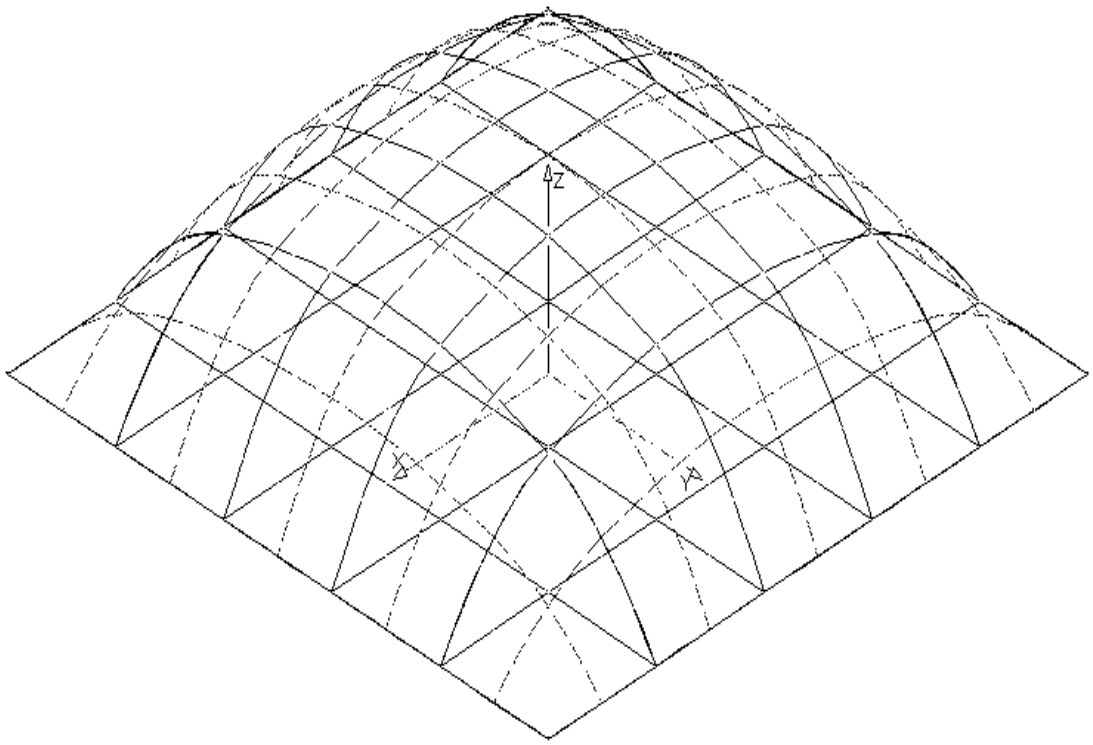


Abbildung 6.14: Änderung des Verbandes aus 6.13 nach Fehler absteigend

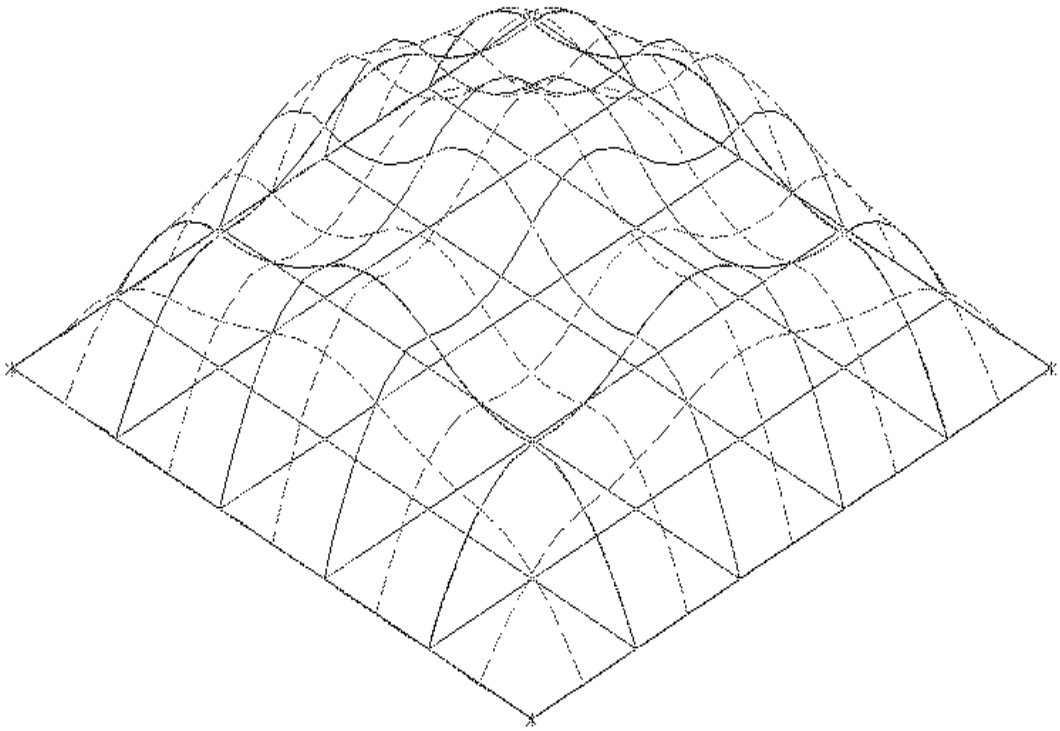


Abbildung 6.15: Änderung des Verbandes aus 6.13 nach Fehler aufsteigend

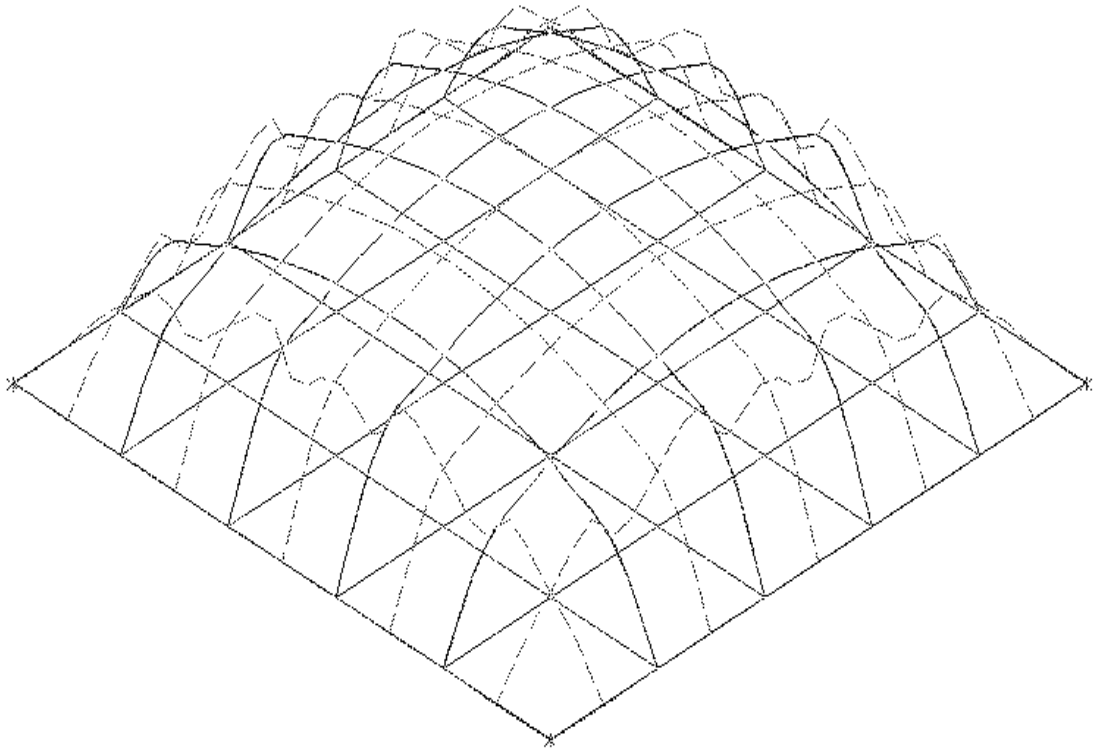


Abbildung 6.16: Änderung des Verbandes aus 6.13 mit Graderhöhung 4

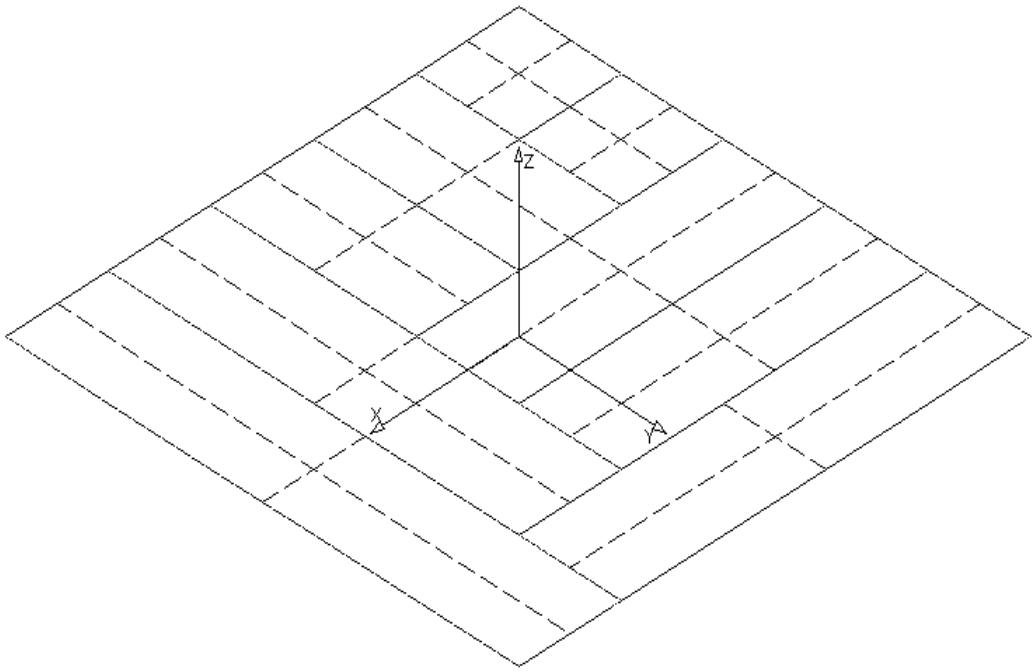


Abbildung 6.17: Ebene als Flächenverband mit „T-Kreuzungen“

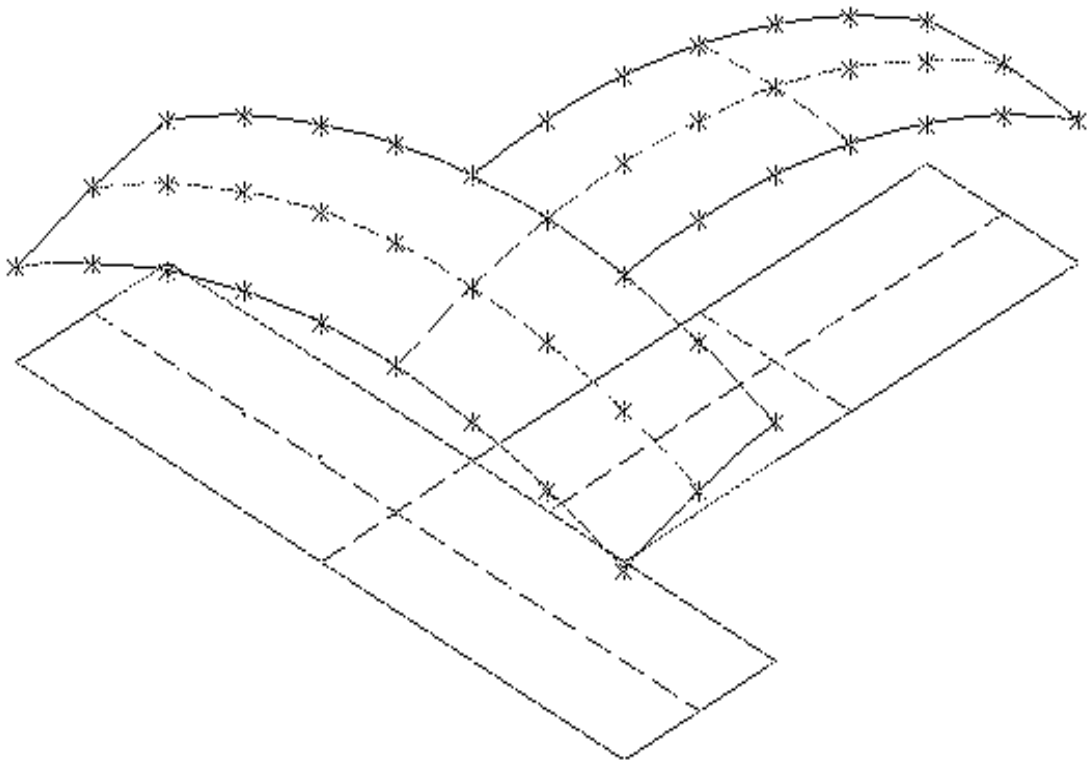


Abbildung 6.18: Änderung einer „T-Kreuzung“

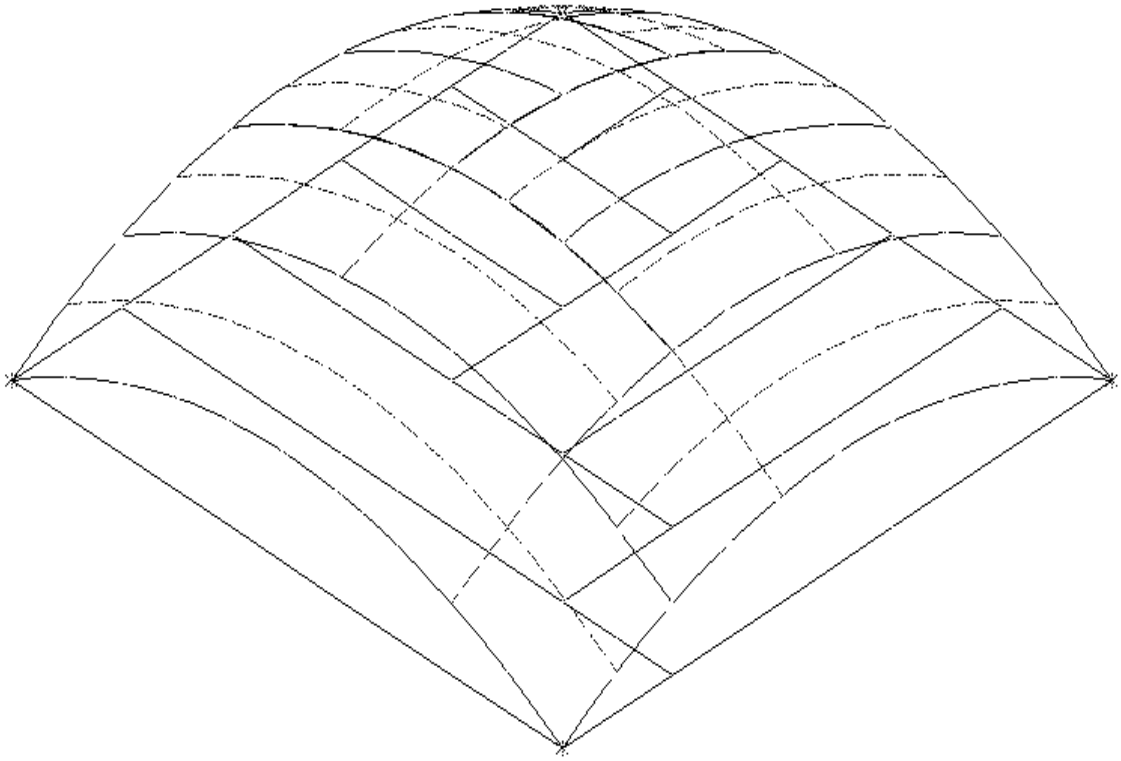


Abbildung 6.19: Änderung des Verbandes aus 6.17

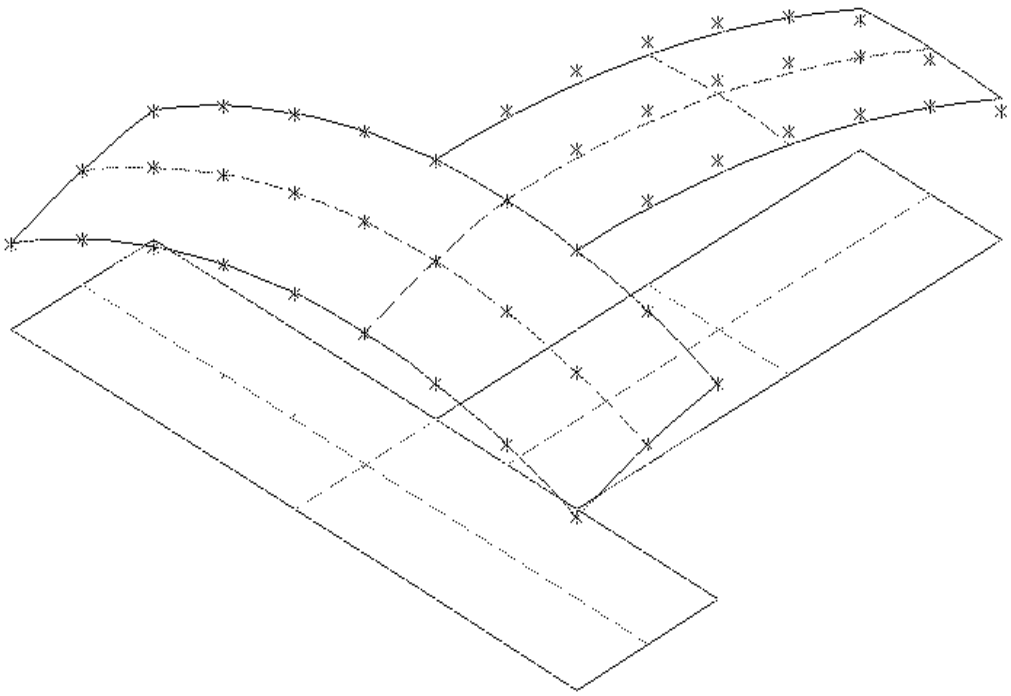


Abbildung 6.20: Änderung einer „T-Kreuzung“mit erzwungener C^1-Stetigkeit

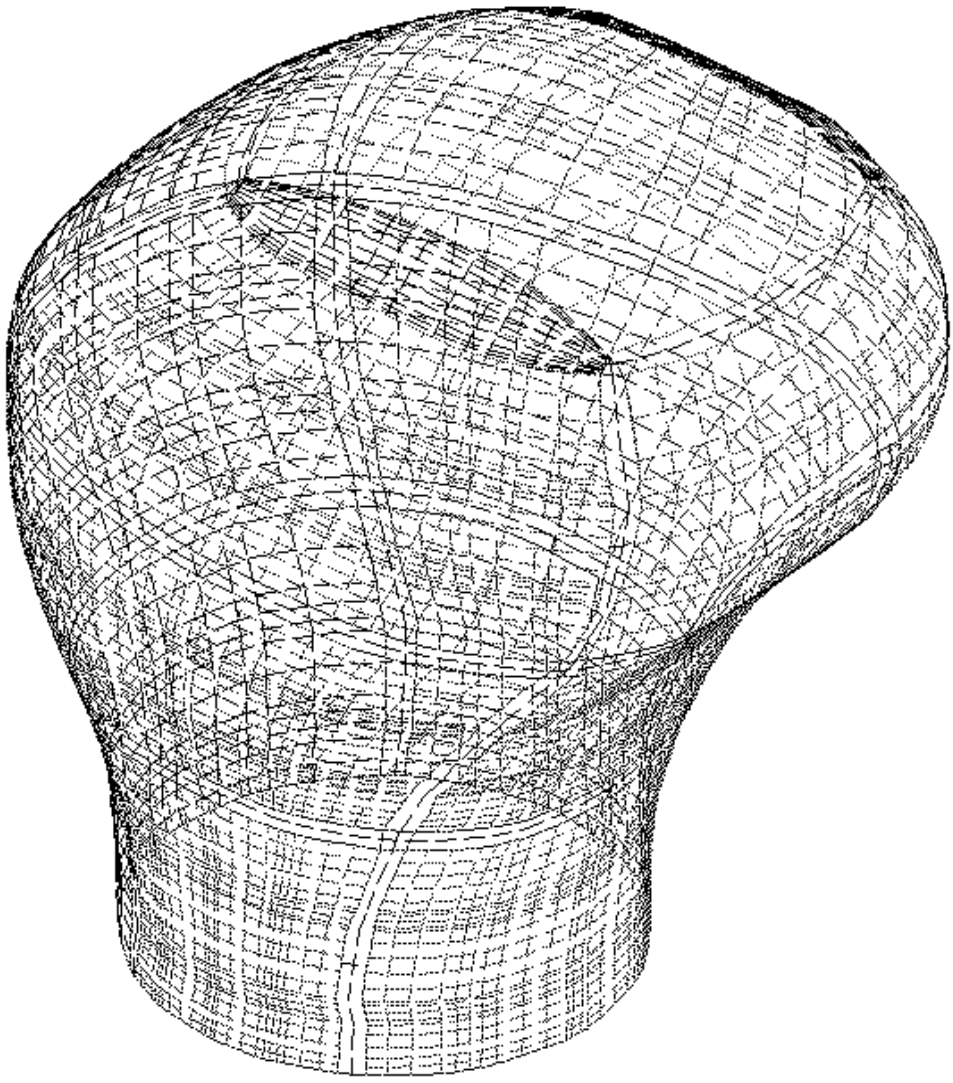


Abbildung 6.21: Mit CAD-Funktionen generierter Schaltknauf

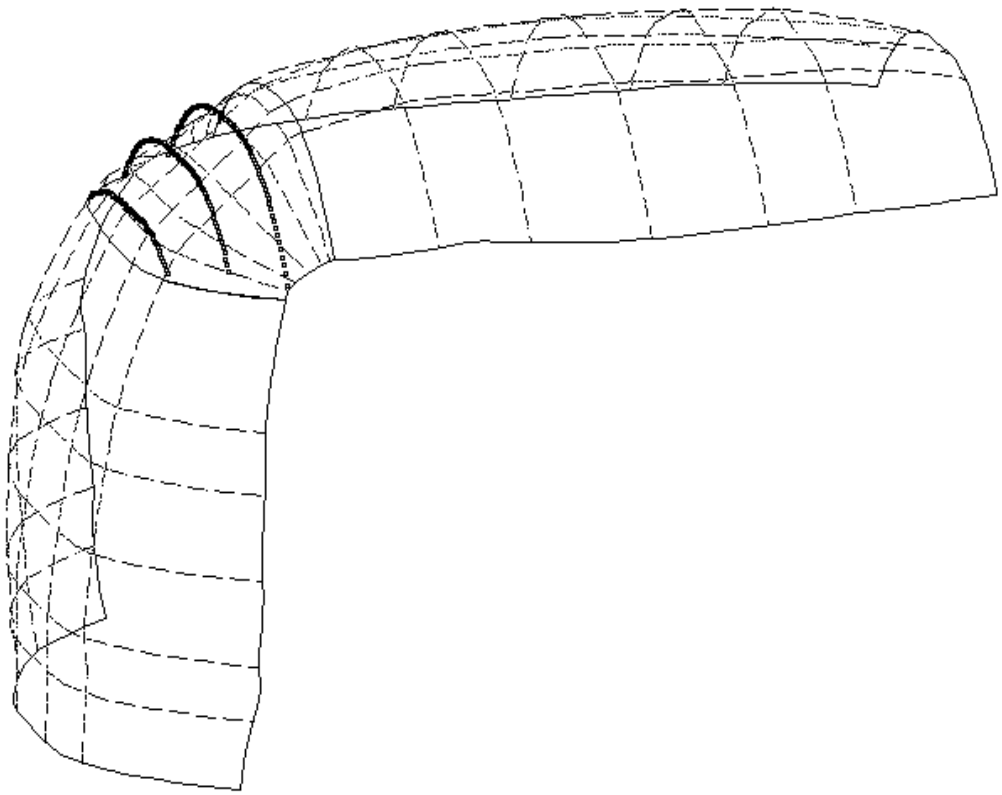


Abbildung 6.22: Zwischenfläche mit digitalisierten Punkten

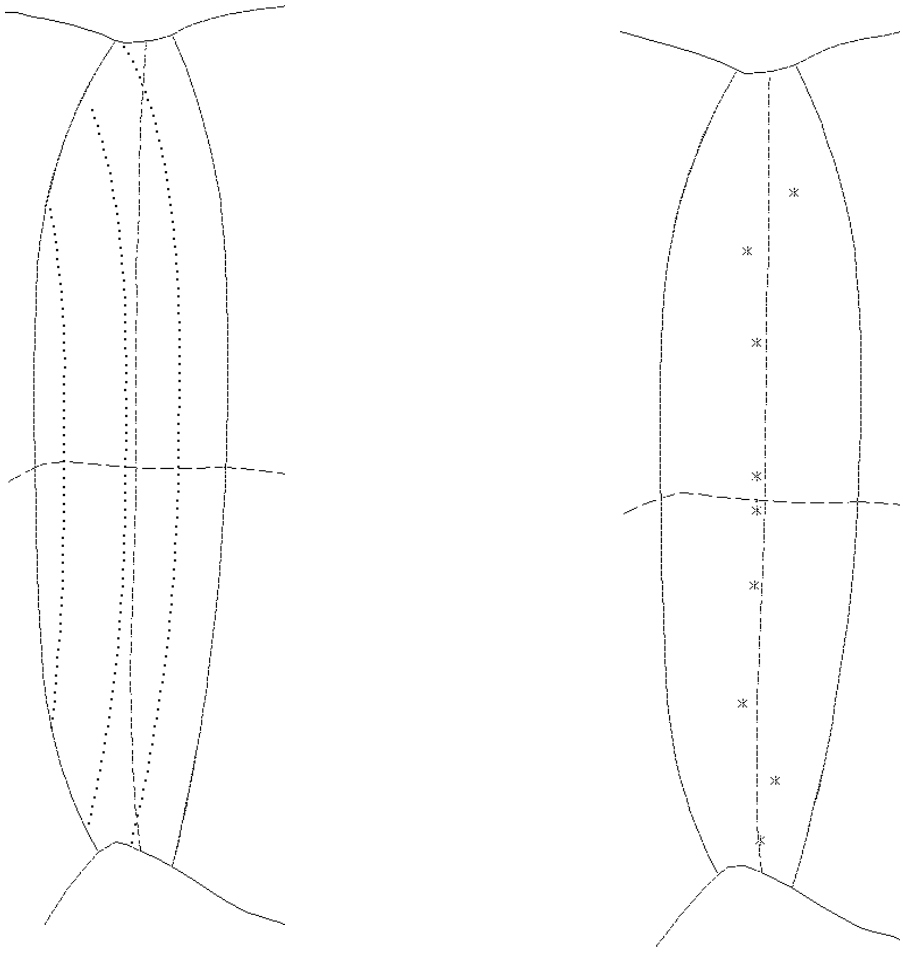


Abbildung 6.23: Alle und die zur Modifikation verwendeten Punkte

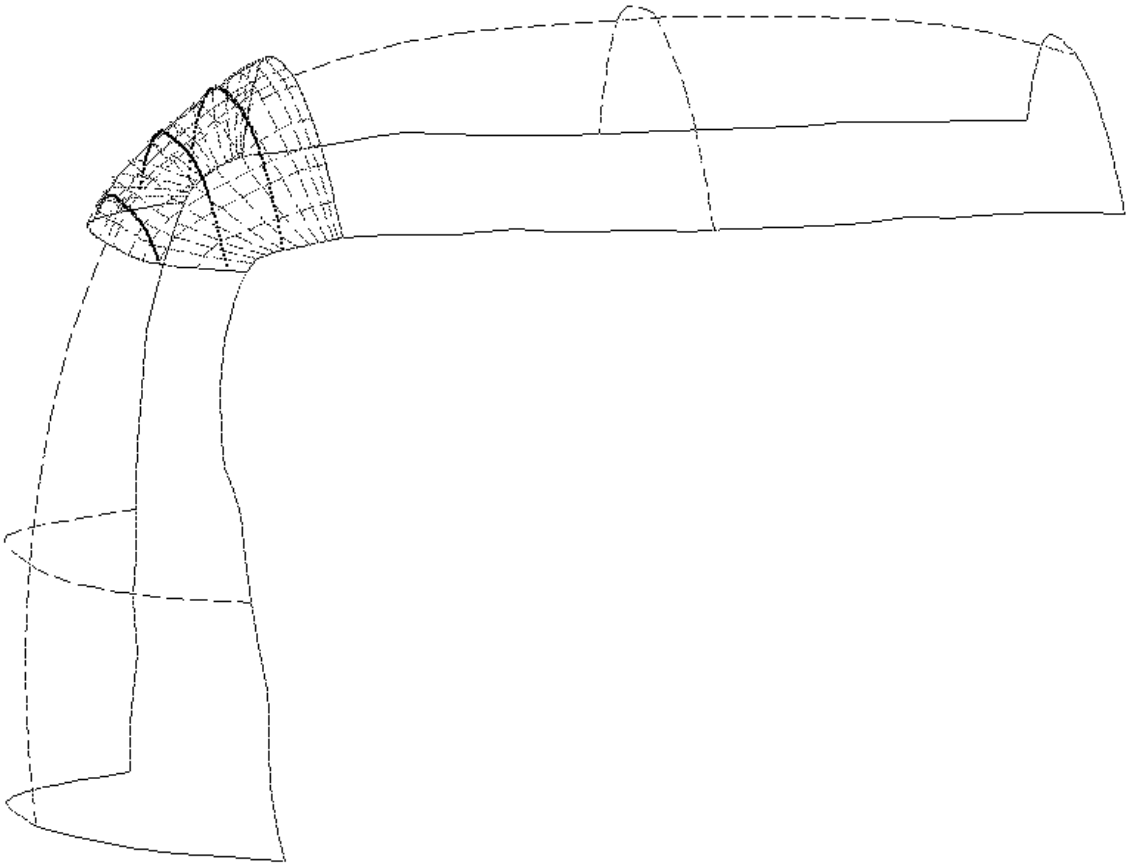


Abbildung 6.24: Änderung der Fläche aus 6.22

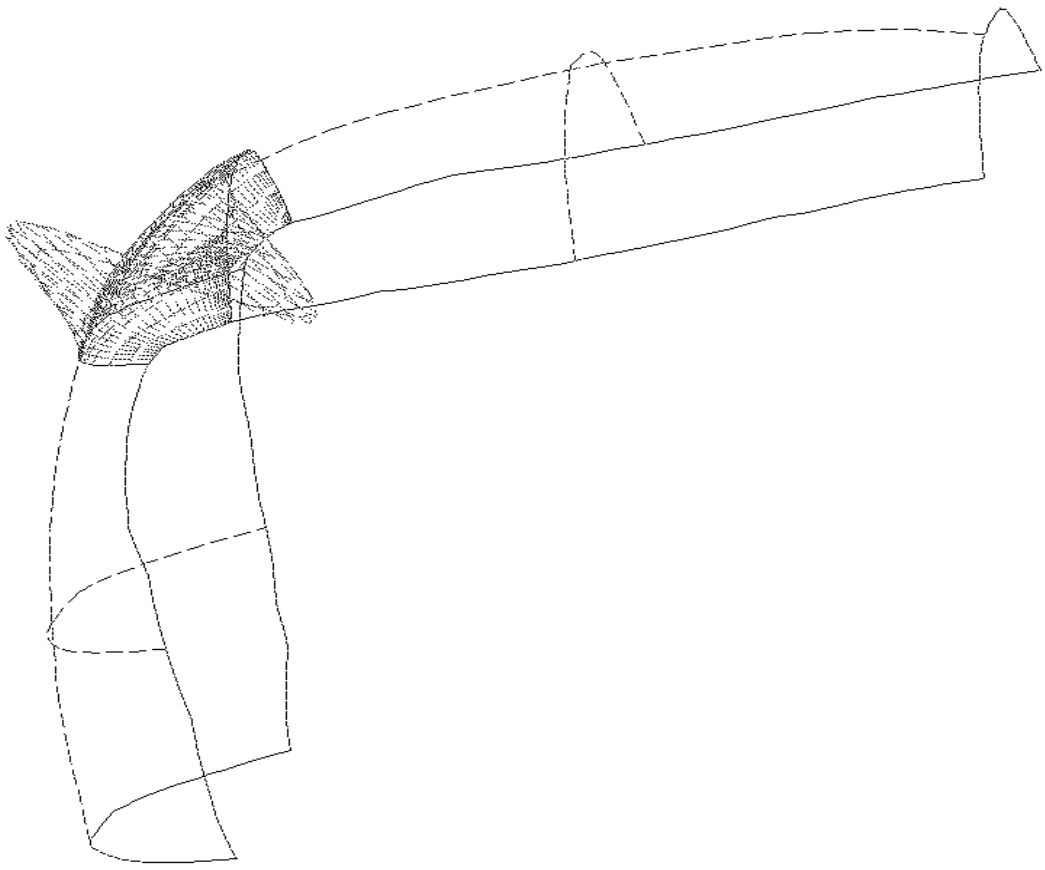


Abbildung 6.25: Modifikation von 6.22 mit ungewollten Schwingungen

Kapitel 7

Zusammenfassung und Ausblick

Mit den Funktionen zur Flächenmodifizierung stehen im CAD-System bisher nicht vorhandene Möglichkeiten zur Verfügung. Es können nun Flächenverbände anhand von Meßpunktdaten geändert werden. Die Daten müssen dazu keine bestimmte Struktur aufweisen. Dadurch werden auch Datensätze abgedeckt, die eine Punkthäufung an interessanten Stellen haben.
Wie die Beispiele in Kapitel 6 zeigen, können durch die einstellbaren Parameter verschiedenartige Änderungen erzeugt werden. So kann die Art der Änderung den Gegebenheiten angepaßt werden. Allerdings kann meist nicht vorhergesagt werden, wie sich die einzelnen Parameter auswirken. Daher ist es notwendig, verschiedene Parameterstellungen auszuprobieren, um die erwünschte Änderung zu erhalten. Da es für die Güte einer Freiformfläche keine eindeutigen Kriterien gibt, kann die Auswahl der Parameter nicht automatisiert werden. Desweiteren sind für unterschiedliche Flächen meist unterschiedliche Parameter notwendig. Daher ist es sinnvoll, einen großen Flächenverband nicht auf einmal zu ändern, sondern in kleinere Bereiche aufzuteilen und diese getrennt zu behandeln.
Bei Änderungen, die die innere Flächenstruktur nicht beeinflussen und fehlerfreien Meßdaten kommt man mit dieser Applikation zu guten Ergebnissen. Da Meßpunktdaten aber im allgemeinen mit Fehlern behaftet sind, sollten zuerst Ausreißer aussortiert werden und dann die Änderung mit möglichst wenig Punkten durchgeführt werden. Für größere Änderungen, vor allem für Änderungen, die eine Umstrukturierung des Flächenverbands notwendig machen, ist diese Applikation nur eingeschränkt zu verwenden. Hier müssen durch interaktives Aufbrechen neue Flächenränder in den Verband eingefügt werden, bevor die Änderung durchgeführt wird.
Um die Möglichkeiten der Applikation zu vergrössern, könnte sie in die neue CATIA-Version 4 eingebunden und um die Behandlung von NURBS (**N**on-**U**niform **R**ational **B**-**S**plines) erweitert werden. Damit wären auch lokale Änderungen von einzelnen Segmenten möglich. In CATIA-Version 3 stehen noch keine NURBS zur Verfügung. Desweiteren könnte ein Algorithmus entwickelt werden, der anhand der Punktdaten feststellt, an welchen Stellen neue Flächengrenzen notwendig sind. Diese werden dann eingefügt oder überflüssige Flächengrenzen entfernt, um auch solche Änderungen durchführen zu können, bei denen dies notwendig wird.

Anhang A

Kurzübersicht über die Funktion

LESEN	PUNKTE	VDA		Einlesen von Punkten aus einer VDAFS 2.0 Da
		DMIS		Einlesen von Punkten aus einer DMIS 3.0 Date
		WSPACE		Einlesen aller vorhandenen Punkte im Workspa
	FLAECHEN			Einlesen von Flächen aus einer VDAFS 2.0 Da
AUSWAHL	ARB_BER	DAZU		Auswahl von zusätzlichen Flächen für den Arb
		WEG		Abwahl von Flächen für den Arbeitsbereich
		NEU		Neue Auswahl von Flächen für den Arbeitsbere
	VERGL	PUNKTE	DAZU	Auswahl von zusätzlichen Punkten für den Soll
			WEG	Abwahl von Punkten für den Soll-Ist-Vergleich
			NEU	Neue Auswahl von Punkten für den Soll-Ist-Ve
		FLAECHEN	DAZU	Auswahl von zusätzlichen Flächen für den Soll-
			WEG	Abwahl von Flächen für den Soll-Ist-Vergleich
			NEU	Neue Auswahl von Flächen für den Soll-Ist-Ver
	AENDERN	PUNKTE	DAZU	Auswahl von zusätzlichen Punkten für die Bere
			WEG	Abwahl von Punkten für die Berechnung der F
			NEU	Neue Auswahl von Punkten für die Berechnung
		FLAECHEN	DAZU	Auswahl von zusätzlichen Flächen für die Bere
			WEG	Abwahl von Flächen für die Berechnung der Fl
			NEU	Neue Auswahl von Flächen für die Berechnung
VERGL				Durchführung des Soll-Ist-Vergleichs mit einzu
AENDERN	WERTE			Einstellung der Werte für die Flächenänderung
	BERECHEN			Start der Berechnung der geänderten Flächen a
SCHREIB	VDA			Schreiben der neuen Flächen in eine VDAFS 2
	DETAIL			Schreiben der neuen Flächen in ein CATIA-De
LOESCHEN	VERGL			Löschen der Daten eines vorhergegangenen Sol
	PUNKTE			Löschen der markierten Punkte
	FLAECHEN	AUSWAHL		Selektives Löschen von neu berechneten Fläche
		ALLE		Löschen von allen noch dargestellten neu berec

Anhang B

Die FSD-Datei

```
/
SECTION1
ERRSET JFEHL
TASK JINIT
FUNEX JFUNEX
/
SECTION2
/
COMMAND CMD111
/VDA PSETS einlesen
TASK JI1
/Panel zur Dateiauswahl darstellen
CMDEX JE1
FUNEX JF1
/Panel wieder entfernen
   INTLEVEL
       PANEL
       *STRC SCR1
       TASK JA1
       UNVAL
   RESUME
   INTLEVEL
      PANEL
      *STRC PFAD
      TASK JA2
      UNVAL
   RESUME
   INTLEVEL
       PANEL
       *STRC VERZ
       PROMPT DATEIAUSWAHL
       TASK JA3
       TASKNO JN1
   RESUME
/
COMMAND CMD112
/DMIS Punktefile einlesen
TASK JI1
/Panel zur Dateiauswahl darstellen
CMDEX JE1
FUNEX JF1
/Panel wieder entfernen
   INTLEVEL
       PANEL
       *STRC SCR1
       TASK JA1
       UNVAL
   RESUME
   INTLEVEL
      PANEL
      *STRC PFAD
      TASK JA2
      UNVAL
   RESUME
   INTLEVEL
       PANEL
       *STRC VERZ
       PROMPT DATEIAUSWAHL
       TASK JA4
       TASKNO JN1
   RESUME
/
COMMAND CMD113
/Workspace Punkte einlesen
   INTLEVEL
      YES
      PROMPT WORKSPACE LESEN
      TASK JA5
      UNVAL
   RESUME
/
COMMAND CMD12
```

```
/VDA SURFS einlesen
TASK JI1
/Panel zur Dateiauswahl darstellen
CMDEX JE1
FUNEX JF1
/Panel wieder entfernen
   INTLEVEL
       PANEL
       *STRC SCR1
       TASK JA1
       UNVAL
   RESUME
   INTLEVEL
      PANEL
      *STRC PFAD
      TASK JA2
      UNVAL
   RESUME
   INTLEVEL
       PANEL
       *STRC VERZ
       PROMPT DATEIAUSWAHL
       TASK JA6
       TASKNO JN2
   RESUME
/
COMMAND CMD211
/ Arbeitsflaechen addieren
   INTLEVEL
      SEL MSELW
      *SUR FLAECHE
      *FAC ABSCHNITT
      TASK JA11
      TASKNO JN6
   RESUME
/
COMMAND CMD212
TASK JI3
CMDEX JE3
FUNEX JF3
/ Arbeitsflaechen wegnehmen
   INTLEVEL
      SEL MSELW
      *SUR FLAECHEN
      *FAC ABSCHNITT
      TASK JA12
      UNVAL
   RESUME
/
COMMAND CMD213
TASK JI3
CMDEX JE3
FUNEX JF3
/ Arbeitsflaechenauswahl neu
   INTLEVEL
      YES
      PROMPT NEUE AUSWAHL
      TASK JA10
      TASKNO JN5
      INTLEVEL
         SEL MSELW
         *SUR FLAECHEN
         *FAC ABSCHNITT
         TASK JA11
         TASKNO JN6
      RESUME
   RESUME
/
COMMAND CMD2211
TASK JI3
/ Punkte addieren
   INTLEVEL
      SEL MSELW
      *PT PUNKTE
      TASK JA8
      TASKNO JN4
   RESUME
/
COMMAND CMD2212
TASK JI3
CMDEX JE2
FUNEX JF2
/ Punkte wegnehmen
   INTLEVEL
      SEL MSELW
      *PT PUNKTE
      TASK JA9
      TASKNO JN3
   RESUME
/
COMMAND CMD2213
TASK JI3
/ Punkteauswahl neu
   INTLEVEL
```

```
      YES
      PROMPT NEUE AUSWAHL
      TASK JA7
      TASKNO JN3
      INTLEVEL
         SEL MSELW
         *PT PUNKTE
         TASK JA8
         TASKNO JN4
      RESUME
   RESUME
/
COMMAND CMD2221
/ Vergleichsflaechenauswahl dazu
TASK JI3
TASK JI6
CMDEX JE10
FUNEX JF10
   INTLEVEL
      SEL MSELW
      *SUR FLAECHE
      TASK JA33
      TASKNO JN16
   RESUME
/
COMMAND CMD2222
/ Vergleichsflaechenauswahl weg
TASK JI3
TASK JI6
CMDEX JE10
FUNEX JF10
   INTLEVEL
      SEL MSELW
      *SUR FLAECHEN
      TASK JA34
      TASKNO JN15
   RESUME
/
COMMAND CMD2223
/ Vergleichsflaechenauswahl neu
TASK JI3
TASK JI6
CMDEX JE10
FUNEX JF10
   INTLEVEL
      YES
      PROMPT NEUE AUSWAHL
      TASK JA32
      TASKNO JN15
      INTLEVEL
         SEL MSELW
         *SUR FLAECHEN
         TASK JA33
         TASKNO JN16
      RESUME
   RESUME
/
COMMAND CMD2311
TASK JI4
CMDEX JE6
FUNEX JF6
/ Aenderungspunkte addieren
   INTLEVEL
      SEL MSELW
      *PT PUNKTE
      TASK JA21
      TASKNO JN12
   RESUME
/
COMMAND CMD2312
TASK JI4
CMDEX JE6
FUNEX JF6
/ Aenderungspunkte wegnehmen
   INTLEVEL
      SEL MSELW
      *PT PUNKTE
      TASK JA22
      TASKNO JN11
   RESUME
/
COMMAND CMD2313
TASK JI4
CMDEX JE6
FUNEX JF6
/ Aenderungspunkteauswahl neu
   INTLEVEL
      YES
      PROMPT NEUE AUSWAHL
      TASK JA20
      TASKNO JN11
      INTLEVEL
         SEL MSELW
         *PT PUNKTE
```

```
         TASK JA21
         TASKNO JN12
      RESUME
   RESUME
/
COMMAND CMD2321
TASK JI6
CMDEX JE7
FUNEX JF7
/ Aenderungsflaechen addieren
   INTLEVEL
      SEL MSELW
      *SUR FLAECHE
      TASK JA14
      TASKNO JN8
   RESUME
/
COMMAND CMD2322
TASK JI6
CMDEX JE7
FUNEX JF7
/ Aenderungsflaechen wegnehmen
   INTLEVEL
      SEL MSELW
      *SUR FLAECHEN
      TASK JA15
      TASKNO JN7
   RESUME
/
COMMAND CMD2323
TASK JI6
CMDEX JE7
FUNEX JF7
/ Aenderungsflaechenauswahl neu
   INTLEVEL
      YES
      PROMPT NEUE AUSWAHL
      TASK JA13
      TASKNO JN7
      INTLEVEL
         SEL MSELW
         *SUR FLAECHEN
         TASK JA14
         TASKNO JN8
      RESUME
   RESUME
/
```

```
COMMAND CMD3
TASK JI3
CMDEX JE4
FUNEX JF4
/ Soll-Ist Vergleich durchfuehren
   INTLEVEL JT1
      KEY
      *REAL TOLERANZ
      TASK JA16
      TASKNO JN9
      INTLEVEL
         PANEL
         *STRC SCR2
         TASK JA17
         UNVAL
      RESUME
   RESUME
/
COMMAND CMD41
/ Werte fuer die Flaechenaenderung
/                          bestimmen
TASK JI2
CMDEX JE5
FUNEX JF5
   INTLEVEL
      PANEL
      PROMPT WERTE AUSWAHL
      TASK JA18
      UNVAL
   RESUME
/
COMMAND CMD42
/ Neue Flaechen berechnen
TASK JI4
INTLEVEL
   YES
   PROMPT AENDERUNGSFLAECHEN BERECHNEN
   TASK JA19
   TASKNO JN10
RESUME
/
COMMAND CMD51
/VDA SURFS schreiben
TASK JI1
/Panel zur Dateiauswahl darstellen
CMDEX JE1
FUNEX JF1
```

```
/Panel wieder entfernen
   INTLEVEL
       PANEL
       *STRC SCR1
       TASK JA1
       UNVAL
   RESUME
   INTLEVEL
      PANEL
      *STRC PFAD
      TASK JA2
      UNVAL
   RESUME
   INTLEVEL
       PANEL
       *STRC VERZ
       PROMPT DATEIAUSWAHL
       TASK JA27
       TASKNO JN14
         INTLEVEL
            PANEL
            PROMPT KOPF FUELLEN
            TASK JA29
            UNVAL
         RESUME 1
   RESUME
   INTLEVEL
       KEY
       *CHAR NAME
       PROMPT DATEINAME
       TASK JA28
       TASKNO JN14
         INTLEVEL
            PANEL
            PROMPT KOPF FUELLEN
            TASK JA29
            UNVAL
         RESUME 1
   RESUME
/
COMMAND CMD52
/Detail schreiben
TASK JI5
/Panel zur Dateiauswahl darstellen
CMDEX JE9
FUNEX JF9
/Panel wieder entfernen
   INTLEVEL
       PANEL
       *STRC SCR1
       TASK JA1
       UNVAL
   RESUME
   INTLEVEL
       PANEL
       *STRC VERZ
       PROMPT DETAILAUSWAHL
       TASK JA30
       UNVAL
   RESUME
   INTLEVEL
       KEY
       *CHAR NAME
       PROMPT DETAILNAME
       TASK JA31
       UNVAL
   RESUME
/
COMMAND CMD61
TASK JI4
   INTLEVEL JT2
      YES
      PROMPT DATEN LOESCHEN
      TASK JA23
      UNVAL
   RESUME
/
COMMAND CMD62
   INTLEVEL
      YES
      PROMPT ALLE AUSGEW. PT LOESCHEN
      TASK JA24
      UNVAL
   RESUME
/
COMMAND CMD631
CMDEX JE8
FUNEX JF8
/ Neue Flaechen loeschen mit Auswahl
   INTLEVEL
      SEL MSELW
      *SUR FLAECHEN
      TASK JA25
      TASKNO JN13
```

```
   RESUME
/
COMMAND CMD632
/ Alle neuen Flaechen loeschen
   INTLEVEL
      YES
      PROMPT ALLE NEUEN FL LOESCHEN
      TASK JA26
      UNVAL
   RESUME
/
SECTION3
/
MENLEVEL
  ITEM LESEN
    MENLEVEL
      ITEM PUNKTE
        MENLEVEL
          ITEM VDA              CMD111
          ITEM DMIS             CMD112
          ITEM WSPACE           CMD113
        ENDMEN
      ITEM FLAECHEN             CMD12
    ENDMEN
  ITEM AUSWAHL
    MENLEVEL
      ITEM ARB_FL
        MENLEVEL
          ITEM DAZU             CMD211
          ITEM WEG              CMD212
          ITEM NEU              CMD213
        ENDMEN
      ITEM VERGL
        MENLEVEL
          ITEM PUNKTE
            MENLEVEL
              ITEM DAZU         CMD2211
              ITEM WEG          CMD2212
              ITEM NEU          CMD2213
            ENDMEN
          ITEM FLAECHEN
            MENLEVEL
              ITEM DAZU         CMD2221
              ITEM WEG          CMD2222
              ITEM NEU          CMD2223
            ENDMEN
        ENDMEN
      ITEM AENDERN
        MENLEVEL
          ITEM PUNKTE
            MENLEVEL
              ITEM DAZU         CMD2311
              ITEM WEG          CMD2312
              ITEM NEU          CMD2313
            ENDMEN
          ITEM FLAECHEN
            MENLEVEL
              ITEM DAZU         CMD2321
              ITEM WEG          CMD2322
              ITEM NEU          CMD2323
            ENDMEN
        ENDMEN
    ENDMEN
  ITEM VERGL                    CMD3
  ITEM AENDERN
    MENLEVEL
      ITEM WERTE                CMD41
      ITEM BERECHEN             CMD42
    ENDMEN
  ITEM SCHREIB
    MENLEVEL
      ITEM VDA                  CMD51
      ITEM DETAIL               CMD52
    ENDMEN
  ITEM LOESCHEN
    MENLEVEL
      ITEM VERGL                CMD61
      ITEM PUNKTE               CMD62
      ITEM NEUE_FL
        MENLEVEL
          ITEM AUSWAHL          CMD631
          ITEM ALLE             CMD632
        ENDMEN
    ENDMEN
ENDMEN
/
```

Anhang C

Header-Dateien

C.1 Definition der Strukturen und Typen

```
/***** Definition der Strukturen fuer die Fl"achenmodifikation  *****/
/* Verzeichnisliste */
struct Verzeichnis_Eintrag {char *name;
                            struct Verzeichnis_Eintrag *naechster;
                            };
typedef struct Verzeichnis_Eintrag VEintr;
/* Projezierte Punkte */
struct ppunkt {long nummer;
               struct flaeche *flaeche;
               double koord[3];   /* x-,y-,z-Koordinate */
               double abstand;
               struct ppunkt *naechster;
              };
typedef struct ppunkt Ppunkt;
/* Punkteliste */
struct punkt {long nummer;
              long farbe;
              char beruecksichtigen;
              Ppunkt *ppunkt;
              struct projektion *projektionen;
              struct punkt *naechster;
             };
typedef struct punkt Punkt;
/* Projektionen */
struct projektion {Punkt *punkt;
                   struct flaeche *flaeche;
                   double abstand[4];/*in x-,y-,z- und Normalenrichtung*/
                   long   patch[2]; /* Patch in u- und v-Richtung */
                   double werte[2];/*Parameterwert in u- und v-Richtung*/
```

```
                  struct projektion *naechste_des_Punkts;
                  struct projektion *naechste_der_Flaeche;
                 };
typedef struct projektion Projektion;
/* Faceliste */
struct face {long nummer;
             long farbe;
             struct flaeche *flaeche;
             struct face *gleiche_basis;
             struct face *naechste;
            };
typedef struct face Face;
/* Flaechenliste */
struct flaeche {long nummer;
                long farbe;
                long patchanzahl[2];
                long aendernummer;
                char aender;
                double fehler;
                char gebaut;
                Face *face;
                Projektion *projektion;
                long eckpunkte[4];
                struct ecke *ecken[4];
                struct kante *kanten[4];
                int stetigkeitswert[8];
                struct patch **neue_patches;
                struct flaeche *naechste;
               };
typedef struct flaeche Flaeche;
/* Neue Flaechen */
struct neue_flaeche {long nummer;
                     struct neue_flaeche *naechste;
                    };
typedef struct neue_flaeche Neue_flaeche;
/* Nachbarschaftsbeziehungen Ecke */
struct ecke {Flaeche *flaeche;
             int ecke;
             struct ecke *naechste;
                };
typedef struct ecke Ecke;
/* Nachbarschaftsbeziehungen Kante */
struct kante {double hier_von;
              double hier_bis;
```

```
               struct flaeche *flaeche;
               int kante;
               double dort_von;
               double dort_bis;
               char verdreht;
               struct kante *naechste;
              };
typedef struct kante Kante;

/* Patchdescription */
struct patch {short ordnung[2];
              double **matrix[3];
             };
typedef struct patch Patch;
/* Flaechenuebergaenge */
struct uebergang {Patch *patch[4];
                  int kante[4];
                  char invers[4];
                  int stetig[4];
                 };
typedef struct uebergang Uebergang;
```

C.2 Deklaration der globalen Variablen

```
extern char jlesepfad[];
extern char jschreibedatei[];
extern VEintr *jdateiliste;
extern Punkt *sollpunkte;
extern Punkt *moment_sollpunkt1;
extern Punkt *moment_sollpunkt2;
extern Punkt *punktspeicher;
extern Ppunkt *ppunkte;
extern Flaeche *flaechen;
extern Flaeche *momentflaeche1;
extern Flaeche *momentflaeche2;
extern Flaeche *flaechenspeicher;
extern Face *faces;
extern Face *momentface1;
extern Face *momentface2;
extern Face *facespeicher;
extern double toleranz;
extern int stetig[4];
extern int maxgrad[2];
extern int maxerhoehung[2];
```

```
extern int berechnungsart;
extern int reihenfolge;
extern int soll_ist_vergleich;
extern Neue_flaeche *neue_flaechen;
extern Neue_flaeche *neueflaechenspeicher;
```

C.3 Deklarationen der CATIA-Routinen

```
/*****              Prototypen der CATIA-Funktionen              *****/

long ggan2(float*,long*,char*,long*);
long gganfd(float*,long*,char*,long*);
long ggarv(long*,long*,char*,float*,long*);
long ggdlst(char*,long*);
long ggdrav(long*,char*,long*);
long ggclst(long*);
long ggeplb(long*,long*);
long gginlb(long*,long*);
long ggopst(char*,long*);
long ggoep(long*,long*);
long ggpick(long*,long*);
long ggpkid(long*,long*);
long ggpl2(long*,long*,float*,long*);
long ggqe(long*,long*,long*,char*,long*);
long ggqste(char*,long*,long*);
long ggsdec(char*,char*,long*,long*);
long ggslif(char*,long*,long*);
long ggspag(char*,long*,long*);
long ggsqws(long*,long*,long*,float*,long*);
long ggsups(char*,long*,long*,float*,long*);
long ggsupv(long*,long*,long*);
long ggswin(long*,long*,char*,float*,float*,float*,
            float*,float*,float*,long*);
long ggtxci(long*,long*);
long ggvch(long*,long*,long*,long*,long*,long*,long*,
            long*,long*,long*,long*);
long ggvmp2(long*,long*,float*,float*,long*);
long ggvtr(long*,long*,long*,char*,long*,long*);
long giccol(long*,long*,long*,long*);
long gicmas(long*,long*);
long gictxt(long*,long*,long*,long*);
long gidtex(long*,long*,long*,long*);
long gieras(long*,long*,long*);
void giiend(void);
```

```
long girad1(long*,long*,long*,char*,long*,long*);
long girbas(long*,long*,long*,long*,long*);
long giride(long*,long*,long*,char*,long*);
long girmat(long*,long*,long*,double*,long*);
long girsiz(long*,long*,long*,long*);
long girtps(long*,long*,long*,long*,long*,long*);
long girvis(long*,long*,long*,long*,long*,long*,long*,long*,long*);
long gisetl(long*,long*,long*,long*,long*,long*,long*,long*);
long gislim(long*,long*,long*,long*,long*,long*);
long giswsp(long*,long*,long*,long*,long*,long*);
long giwdet(long*,long*,long*,char*,long*,long*);
long giwedt(long*,long*,long*,long*,long*);
long giwtex(long*,long*,long*,char*,double*,long*,long*,long*);
long giwpt(long*,double*,long*,long*);
long giwsur(long*,double*,long*,long*);
long gmicgi(long*,long*,long*,long*,long*);
long gmicsl(long*,long*,float*,long*,long*,long*);
long gmikey(long*,long*,long*,long*,double*,long*,char*,long*);
long gmimse(long*,long*,long*,long*,long*);
long gminte(long*,long*,long*);
long gmisel(long*,long*,long*,long*);
long gsdisf(long*,long*,long*,long*,long*,long*,double*,double*,long*);
long gsoiof(long*,long*,long*,double*,double*,long*,long*,long*);
long gsoisl(long*,long*,long*,long*);
long gsopos(long*,long*,long*,long*,long*,long*);
long gucele(long*,long*,long*);
long gucsur(long*,long*,double*,long*,double*,long*);
void guedis(long*,long*);
void gueror(long*);
```

C.4 Deklarationen der eigenen Prozeduren

```
/***** Prototypen der Funktionen fuer die die Flaechenmodifikation *****/

void ja1(void);
void ja2(void);
void ja3(void);
void ja4(void);
void ja5(void);
void ja6(void);
void ja7(void);
void ja8(void);
void ja9(void);
void ja10(void);
```

```
void ja11(void);
void ja12(void);
void ja13(void);
void ja14(void);
void ja15(void);
void ja16(void);
void ja17(void);
void je1(void);
void je2(void);
void jentw(void);
void jf1(void);
void jf2(void);
void jfunex(void);
void ji1(void);
void jinit(void);
void jn1(void);
void jn2(void);
void jn3(void);
void jn4(void);
void jn5(void);
void jn6(void);
void jn7(void);
void jn8(void);
void jn9(void);

void    j2kanten_teilen(Kante*,Kante*);
void    jbaue_Flaeche(long,long*);
double  jbernstein(int,int,double);
void    jbez2mon(double**,int,int);
void    jdiff_fuellen(Projektion*,Patch,int,int,double*,
                      double*,double**,long*);
void    jdiff_korrigieren(Patch,int,double*,double*,double**,double**);
void    jflaeche_aufbrechen(Flaeche*,double**,double,int,
                            Projektion*,int*,int*);
void    jhole_Name(char*);
void    jKante_eintragen(Flaeche*,int,Flaeche*,double,double,
                         double,double);
void    jkante_teilen(Kante*,Kante*);
int     jlese_VDA_Zeile(FILE*,char*);
long    jlese_Verzeichnis(void);
void    jloese_system(double**,double*,double*,int);
void    jmon2bez(double**,int,int);
void    jpatch_darstellen(int,int,Patch,double*,long*);
void    jpatch_fuellen(Patch,int*,int*,int*,double*,int);
```

```
void    jrandwerte(Patch,int,Patch,int,int,int,int,double,double);
void    jsoll_ist_tabelle(void);
void    jTeilkante_eintragen(Flaeche*,unsigned char,long*,
                             double*,long*,double*,
                             Flaeche*,unsigned char,long*,
                             double*,long*,double*);
int     jueber(int,int);
void    jVDA_Kopffenster(void);
void    jvektor_fuellen(int*,int*,int*,Patch,double*,int,
                        double*,double*,double*);
void    jwert_berechnen(Patch,int*,int*,double,double,double*);
double* jxmat_bauen(Patch**,double*);
double  scal_prod(double*,double*,int);
void    test_ptr(double*);
```

Literaturverzeichnis

[1] J. Hoschek/D. Lasser; Grundlagen der geometrischen Datenverarbeitung; 2. Auflage; Teubner Stuttgart; 1992

[2] do Carmo; Differentialgeometrie von Kurven und Flächen; 3. Auflage; Vieweg-Verlag; 1993

[3] H. Heuser; Lehrbuch der Analysis – Teil 2; 8. Auflage; Teubner Stuttgart; 1993

[4] B. W. Kernigham/ D. M. Ritchie; Programmieren in C; 2. Auflage; Carl Hanser Verlag; 19990

[5] CATIA Graphics Interactive Interface Version 3 – Reference Manual; 4. Auflage; IBM Publication Number SH50-0020-03; Mai 1991

[6] CATIA Base Version 3 – Geometry Interface – Reference Manual; 5. Auflage; IBM Publication Number SH50-0091-04; Juni 1992

[7] CATIA Base Version 3 – Mathematical Subroutines Package – Reference Manual; 4. Auflage; IBM Publication Number SH50-0089-03; November 1991

[8] CATIA 3D Design Version 3 – Geometry Interface – Reference Manual; 3. Auflage; IBM Publication Number SH50-0068-02; Dezember 1990

[9] CATIA 3D Design Version 3 – Mathematical Subroutines Package – Reference Manual; 3. Auflage; IBM Publication Number SH50-0067-02; Dezember 1990

[10] CATIA Interactive User Acces Commands; 3. Auflage; IBM Publication Number SH50-0069-02; Dezember 1990

[11] CATIA Base Version 3 – Master Index; 4. Auflage; IBM Publication Number SH50-0158-03; Mai 1991

[12] Deutsche Industrie Norm 66301: Industrielle Automation, Rechnerunterstütztes Konstruieren, Format zum Austausch geometrischer Informationen; Juli 1986

[13] American National Standards Institute; ANSI/CAM-I 101-199X

[14] Vieweg Mathematik Lexikon; 2. Auflage; Vieweg-Verlag; 1993

[15] H. Kopka; LaTeX – Eine Einführung; 2. Auflage; Addison-Wesley; 1990

Danksagung

Mein Dank gilt allen Mitarbeitern des Instituts für Produktionstechnik und Automatisierung (IPA) der Fraunhofer Gesellschaft (FhG), insbesondere Frau Dr. Sabine Roth-Koch, die mir als Betreuerin dieser Arbeit zur Seite stand. Desweiteren möchte ich mich bei Herrn Prof. Dr. Klaus Höllig von der Mathematischen Fakultät der Universität Stuttgart für die Anerkennung dieser Arbeit bedanken. Ferner bei meinen Eltern und meiner Ehefrau für die finanzielle und moralische Unterstützung meines Studiums. Außerdem gilt mein Dank allen, die während meines Studiums in der Fachschaft Mathematik der Universität Stuttgart aktiv waren, die mir und vielen anderen Studenten durch Rat und Tat bei allen kleinen und größeren Problemen mathematischer und anderer Natur geholfen haben.

Jörg Marcus Hörner