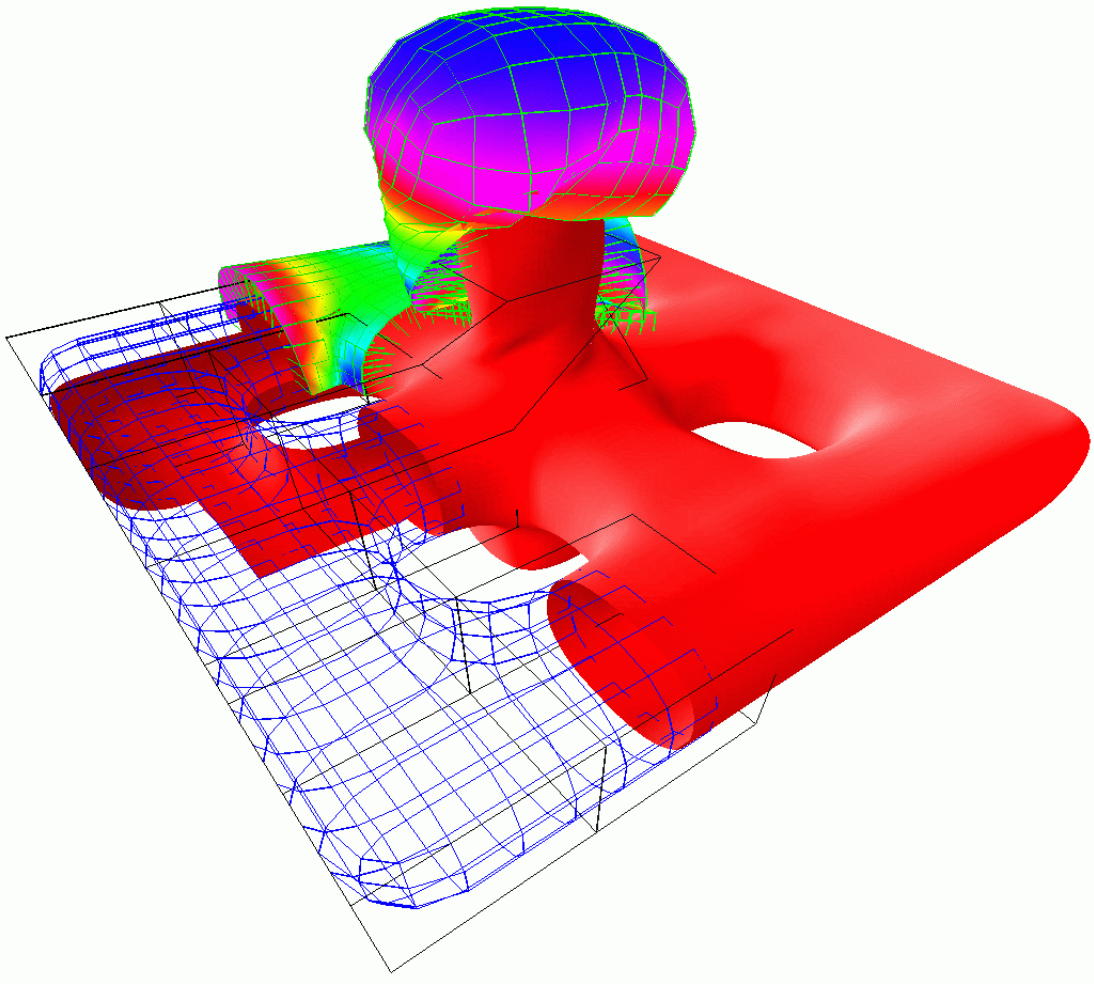
LiLit

Visualisierung von Funktionalen auf Freiformflächen

Frank Langbein
langbein@mathematik.uni-stuttgart.de



24. Juni 1999

Übersicht

- Flächen
  - Biquadratische G–Spline Flächen
  - Doo–Sabin Subdivision
- Funktionen auf Flächen
  - Zusammensetzen von Funktionen auf Flächenstücken
  - Darstellung von Funktionen durch Splines
  - Integration
  - Isolinien
  - Krümmung
- Getrimmte Flächen
- LiLit Programmstruktur

Geometrische Glattheit für biquadratische G–Spline Flächen

- Seien die Flächenstücke p und q gegeben,

$$p: \quad [0,1]^2 \mapsto \mathbb{R}^3, \quad p(u,v) = \sum_{j=0}^{2}\sum_{k=0}^{2} P_{j,k} B_j^2(u) B_k^2(v),$$

$$q: \quad [0,1]^2 \mapsto \mathbb{R}^3, \quad q(u,v) = \sum_{j=0}^{2}\sum_{k=0}^{2} Q_{j,k} B_j^2(u) B_k^2(v).$$

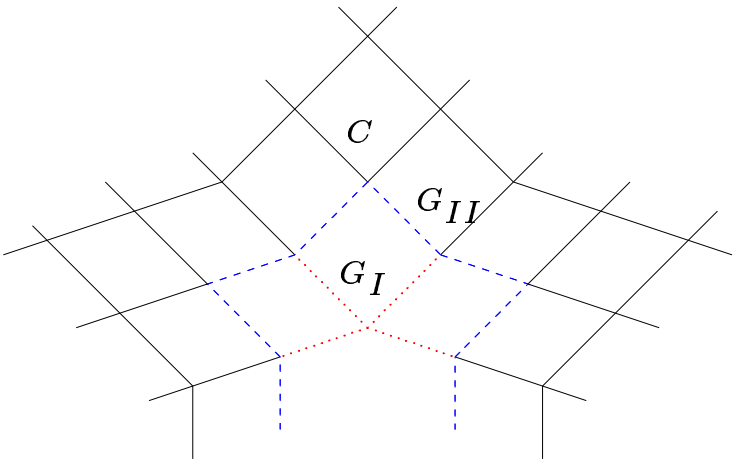
- Die aus p und q zusammengesetze Fläche ist geometrische glatt, wenn

$$\Phi p_u(\cdot,0) + \Psi p_v(\cdot,0) + q_v(\cdot,0) \equiv 0$$

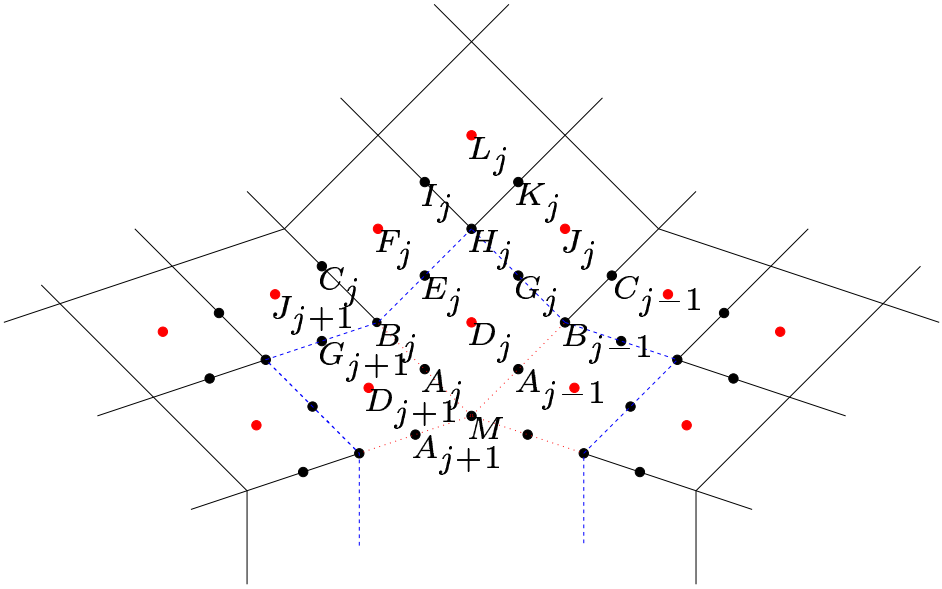
  mit $\Phi : [0,1] \to \mathbb{R}$ und $\Psi : [0,1] \to \mathbb{R}^+$ und der gemeinsamen Randkurve

$$c : [0,1] \to \mathbb{R}^3, \quad c(u) = p(u,0) = q(u,0) \text{ für } t \in [0,1].$$

- Spezielle Typen geometrischer Glattheit:
  i) Typ C, wenn $\Phi \equiv 0$ und $\Psi \equiv 1$;
  ii) Typ G_I, wenn $\Phi \not\equiv 0$ und $\Psi \equiv 1$;
  iii) Typ G_{II}, wenn $\Phi \equiv 0$ und $\Psi \not\equiv 1$.
- Bei biquadratische G–Spline Flächen mit semi–regulären Kontrollnetzen werden an den Irregularitäten obige Glattheitsbedingungen verwendet:



Biquadratische G–Spline Flächen



- Wenn die Kontrollpunkte die folgenden Gleichungen erfüllen

$$\begin{bmatrix} E & -E & -E & E \\ S-E & E & -S & 0 \\ TS+\kappa T & (1-\kappa)T & 0 & 0 \end{bmatrix} \begin{bmatrix} D \\ F \\ J \\ L \end{bmatrix} =: PX = 0,$$

(mit dem Shift Operator $S : A_j \to A_{j+1}$ und einer speziellen $(n-3) \times n$ Matrix T) dann erhält man die restlichen Bézierpunkte aus

$$A = \frac{(S+\kappa E)D + (1-\kappa)F}{2}, \qquad G = \frac{D+J}{2},$$

$$B = \frac{D+F+SD+SJ}{4}, \qquad H = \frac{D+F+J+L}{4},$$

$$C = \frac{F+SJ}{2}, \qquad I = \frac{F+L}{2}, \qquad M = \frac{1}{n}\sum_{j=0}^{n-1} A_j,$$

$$E = \frac{D+F}{2}, \qquad K = \frac{J+L}{2}, \qquad \kappa = \frac{2}{2-\cos\left(\frac{2\pi}{n}\right)}.$$

- Für beliebige Kontrollpunkte $x = [d, f, j, l]^T$ kann ein passender Vektor X über ein Optimierungsproblem gefunden werden:

$$\|X - x\| = \min_{P\tilde{x}=0} \|\tilde{x} - x\| \quad \Rightarrow \quad X = (E - P^+P)x.$$

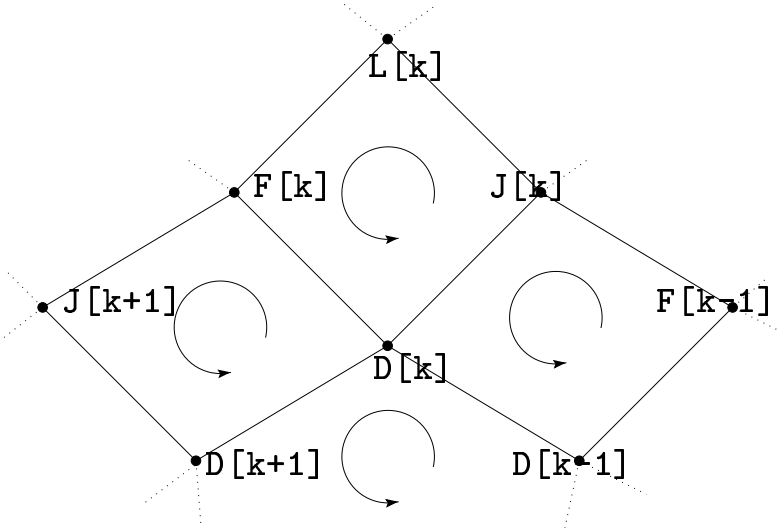
Algorithmen für biquadratische G–Spline Flächen

Datenstrukturen / ODL–Format

- Intern werden alle Objekte als Listen in einer `World` Klasse gepeichert.
- Objekte können als Teilobjekte andere Objekte enthalten.
- Semi–reguläre Kontrollnetze werden als Liste von Polygonen gespeichert.
- Die Polygone werden als Liste von Punkten gespeichert.
- Ein Punkt gilt nur dann als gleich, wenn er über den gleichen Bezeichner angesprochen wird. Die Position wird hierfür nicht berücksichtigt.

G–Spline Algorithmus

- Zunächst wird die Fläche (falls möglich) über die Polygone orientiert:



- Die Irregularitäten werden im Kontrollnetz gesucht und die Bézierpunkte werden erzeugt.
- Für die restlichen regulären Teilen des Kontrollnetzes werden die Bézierpunkte berechnet.

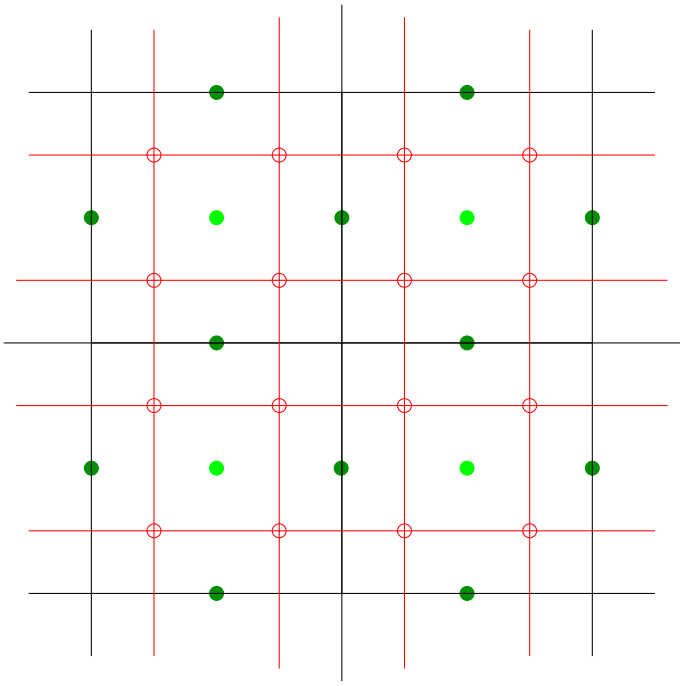
Beispiele [gspline]

- Irregularität der Ordnung 3 / ODL–Format [gspline3_{0,1}]
- Irregularität der Ordnung 5 [gspline5]
- Möbiusband [moebius]

Doo–Sabin Subdivision (1)

Basierend auf den Verfeinerungstechniken für biquadratische, uniforme B–Spline–Flächen entwickelten Donald Doo und Malcolm Sabin einen Subdivision–Algorithmus, der zur Verfeinerung von Kontrollnetzen beliebiger Topologie verwendet werden kann.

- Das alte Kontrollnetz wird vollständig durch ein neues ersetzt.
- Für jedes Polygon und jeden Punkt, werden die neuen Kontrollpunkte als Mittel von 4 durch das jeweilige Polygon bestimmten Punkte berechnet:
  - Alter Kontrollpunkt, für den der neue berechnet wird
  - 2 Kantenpunkte (Mittelpunkte der angrenzenden Kanten des Polygons)
  - Gitterpunkt (Mittel aller Punkte des Polygons)
- Für jedes alte Polygon werden die neuen Kontrollpunkte, die mit diesem Polygon erzeugt wurden, zu einem neuen Polygon zusammengesetzt.
- Für jeden alten Punkt werden die neuen Kontrollpunkte, die mit diesem Punkt erzeugt wurden, zu einem neuen Polygon zusammengesetzt.
- Für jede Kante wird ein neues Polygon aus den neuen Kontrollpunkten erzeugt.



Doo–Sabin Subdivision (2)

- Jedes ursprüngliche Polygon wird verkleinert.
- Für jede Kante ensteht ein neues Viereck, das die verkleinerten Polygone miteinander verbindet.
- Für jeden alten Kontrollpunkt wird ein neues Polygon eingefügt.

Anwendung auf irreguläre Netze

- Liegen zwei irreguläre Polygone eines Kontrollnetzes direkt nebeneinander oder liegt nur ein reguläres Polygon zwischen ihnen, dann kann an dieser Stelle der G–Spline–Algorithmus nicht eingesetzt werden.
- Zwei irreguläre, nebeneinander liegende Polygone des alten Kontrollnetzes werden durch die Anwendung von Doo–Sabin durch ein reguläres Polygon getrennt.
- Führt man den Algorithmus zweimal aus, sind sicher alle Irregularitäten durch wenigstens zwei reguläre Polygone getrennt und der G–Spline–Algorithmus kann auf das Kontrollnetz angewendet werden.

SBW–Format

- Objekte werden durch Zusammensetzen von gleichgroßen Würfel erzeugt.
- Der Doo–Sabin Algorithmus erzeugt hieraus semi–reguläre Netze.

Beispiele [doosabin]

- Einfacher Würfel [cube_{0,1,2,3}]
- Verdrehte Acht [twisted_eight_{sbw,0,1,2,3}]
- Kreuz [cross_{0,1}]
- Verdrehter Würfel [xcube_{sbw,0,1}]

Zusammensetzen von Funktionen auf Flächenstücken (1)

- Wir setzen zunächst zwei Bézierflächen p und q geometrisch glatt zusammen:

$$p: \quad [0,1]^2 \mapsto \mathbb{R}^3, \quad p(u,v) = \sum_{j=0}^{\alpha} \sum_{k=0}^{\beta} P_{j,k} B_j^{\alpha}(u) B_k^{\beta}(v),$$

$$q: \quad [0,1]^2 \mapsto \mathbb{R}^3, \quad q(u,v) = \sum_{j=0}^{\alpha} \sum_{k=0}^{\beta} Q_{j,k} B_j^{\alpha}(u) B_k^{\beta}(v).$$

  Die gemeinsame Randkurve $c : [0,1] \to \mathbb{R}^3$ sei o.B.d.A.

$$c(u) = p(u,0) = q(u,0) \quad \text{für } u \in [0,1].$$

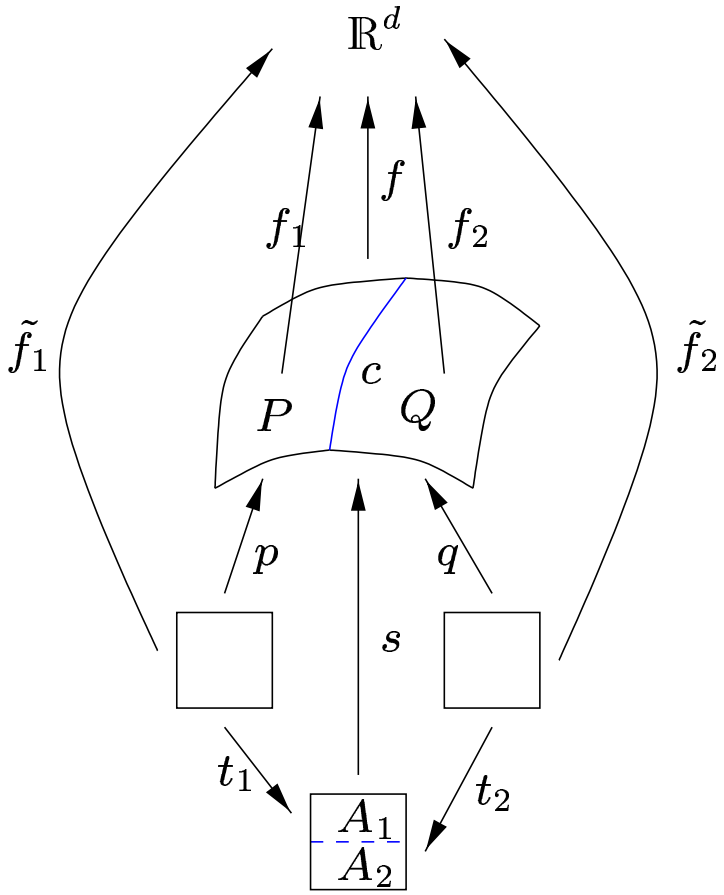
- Auf den beiden Bézierflächen $P := p([0,1]^2)$ und $Q := q([0,1]^2)$ seien zwei stetige differenzierbare Abbildungen gegeben:

$$f_1 : P \to \mathbb{R}^d,$$
$$f_2 : Q \to \mathbb{R}^d.$$

- Diese beiden Funktionen sollen stetig differenzierbar zusammengesetzt werden:

$$f(x) = \begin{cases} f_1(x) & \text{für } x \in P, \\ f_2(x) & \text{für } x \in Q. \end{cases}$$

Zusammensetzen von Funktionen auf Flächenstücken (2)



- Damit f stetig ist, muß gelten

$$f_1(c(u)) = f_2(c(u)) \text{ für } u \in [0,1].$$

- Werden p und q geometrisch glatt zusammengesetzt, d.h. wenn

$$\Phi p_u(\cdot,0) + \Psi p_v(\cdot,0) + q_v(\cdot,0) \equiv 0$$

mit $\Phi : [0,1] \to \mathbb{R}$ und $\Psi : [0,1] \to \mathbb{R}^+$ gilt, dann ist f stetig differenzierbar, wenn gilt

$$\Phi \tilde{f_{1,u}}(\cdot,0) + \Psi \tilde{f_{1,v}}(\cdot,0) + \tilde{f_{2,v}}(\cdot,0) \equiv 0.$$

- Die Bedingung für die Differenzierbarkeit von f erhält man über die Parametrisierungen $\tilde{f}_1$, $\tilde{f}_2$ von f_1, f_2 und einer Parametrisierung von f.

Darstellung von Funktionen durch Splines

- Sowohl die Fläche, als auch die Funktionen auf dieser Fläche sollen durch biquadratische Splines dargestellt werden.
- Es sei eine biquadratische Splinefläche gegeben,

$$p: \quad [0,1]^2 \mapsto \mathbb{R}^3, \quad p(u,v) = \sum_{j=0}^{2} \sum_{k=0}^{2} P_{j,k} B_j^2(u) B_k^2(v).$$

- Eine Funktion $f_1 : P \to \mathbb{R}^d$ auf dieser Fläche beschreiben wird als biquadratischen Spline der Form

$$\tilde{f}_1 : [0,1]^2 \to \mathbb{R}^d, \quad \tilde{f}_1(u,v) = \sum_{j=0}^{2} \sum_{k=0}^{2} F_{j,k} B_j^2(u) B_k^2(v).$$

- Für einen Punkt $x = p(u_0, v_0)$ mit $u_0, v_0 \in [0,1]$ des Flächenstückes p gilt $f_1(x) = \tilde{f}_1(u_0, v_0)$, bzw. es gilt $\tilde{f}_1 = f_1 \circ p$.
- Für das stetig differenzierbare Zusammensetzen gelten die gleichen Bedingungen wie für glatte Flächen. Allerdings sind Φ und Ψ bereits durch das Zusammensetzen der Flächenstücke bestimmt.
- Ist die Fläche durch ein semi–reguläres Kontrollnetz gegeben, beschreiben wir eine Funktion auf dieser Fläche, indem wir zu jedem Kontrollpunkt der Fläche einen Kontrollpunkt für die Funktion angeben.
- Die Algorithmen für die Flächen müssen nun neben den Kontrollpunkten der Fläche auch die der Funktion berücksichtigen. Die Flachenkontrollpunkte können jedoch in die gleichen Formeln eingesetzt werden.

Darstellungsmodi für die Funktionen

- Darstellungsmodi für die Funktionen auf den Flächen:
  - Stacheln
  - Gitter über der Fläche
  - Farbdarstellung auf der Fläche
  - Farbdarstellung als Fläche über der Fläche
- Berechnung der Farbwerte:

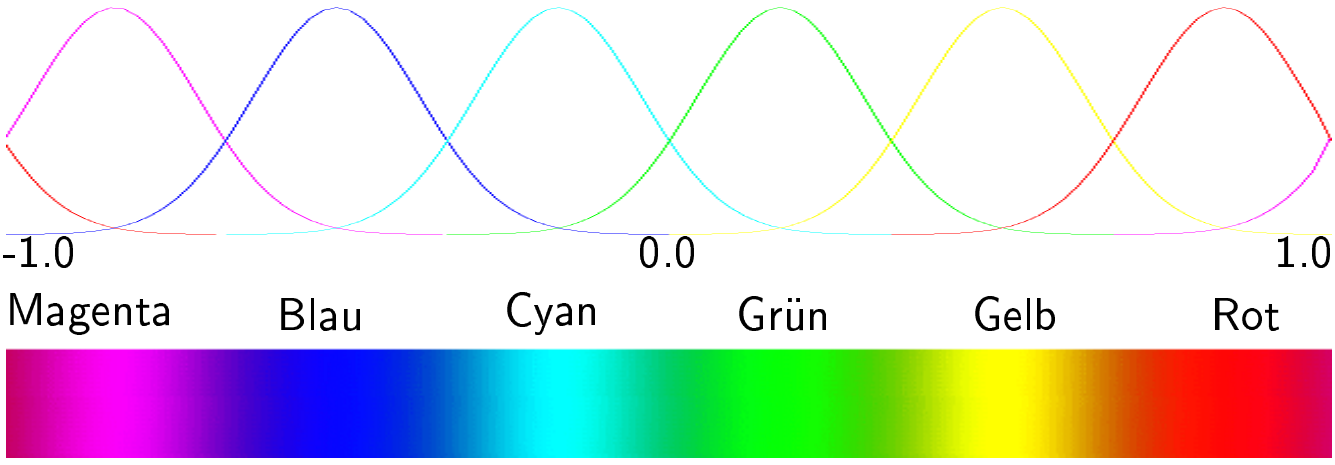
$$w(x) = f(x) f_{\text{scale}} c_{\text{scale}} + c_{\text{delta}},$$

$$C_i(w) = e^{-32\left(\frac{k_i}{6} - w\right)^2}$$

wobei $w(x)$ ins Intervall $[-1, 1]$ verschoben werden muß.

Rot l.:	$i = 1$	$k_i = -7$	Grün:	$i = 5$	$k_i = 1$
Magenta l.:	$i = 2$	$k_i = -5$	Gelb:	$i = 6$	$k_i = 3$
Blau:	$i = 3$	$k_i = -3$	Rot r.:	$i = 7$	$k_i = 5$
Cyan:	$i = 4$	$k_i = -1$	Magenta r.:	$i = 8$	$k_i = 7$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} r \\ 0 \\ 0 \end{bmatrix} C_1(w) + \cdots + \begin{bmatrix} r \\ 0 \\ b \end{bmatrix} C_8(w)$$



Beispiele zur Darstellung von Funktionen

FNC Format

- Reguläres Kontrollnetz für die Fläche und für die Funktion wird durch Parametrisierung erzeugt.
- Irregularitäten können durch späteres identifizieren einzelner Punkte erzeugt werden.

Beispiele [function]

- Einheitsnormalenfeld der Acht–Fläche [eightsurf_{fnc,odl}]
- $(x, y, z) \mapsto (x^2 + y^2)^{0.2}$ auf dem Whitneyschen Regenschirm [whitney_umbrella]
- Dipolpotential [dipol]

Oberflächenintegrale von Skalarfeldern

- **Oberflächenintegral von Skalarfeldern.**
  Sei $S : \mathbb{R}^2 \supset K \to \mathbb{R}^3$ eine Fläche und f ein stetiges Skalarfeld auf $S(K)$. Das Oberflächenintegral von f über S ist

$$\int_S f d\sigma := \int_K f(S(u,v)) \left\| \frac{\partial S}{\partial u} \times \frac{\partial S}{\partial v} \right\| d(u,v)$$

- f und S werden bereits durch biquadratischen Bézierkontrollnetze beschrieben. Das Integral über die gesamte Fläche kann als Summe der Integrale über die Bézierflächen dargestellt werden,

$$\int_S f d\sigma = \sum_{i \in I} \int_{S(K_i)} f \, d\sigma.$$

- Jedes einzelne Integral kann über

$$\iint_{[0,1]^2} f_i(u,v) \left\| \frac{\partial S_i}{\partial u} \times \frac{\partial S_i}{\partial v} \right\| d(u,v)$$

  berechnet werden, wobei die Funktion f_i und die Fläche S_i in Bézierform gegeben sind.

Oberflächenintegrale von Vektorfeldern

- **Oberflächenintegral von Vektorfeldern.**
  Sei $S : \mathbb{R}^2 \supset K \to \mathbb{R}^3, (u,v) \mapsto \left[X(u,v), Y(u,v), Z(u,v)\right]^T$ eine Fläche mit dem Parameterbereich K und $f : S \to \mathbb{R}^3, \quad p \mapsto \left[P(p), Q(p), R(p)\right]^T$ ein stetiges drei–dimensionales Vektorfeld auf $S(K)$. Man definiert das Oberflächenintegral von f über S als

$$\int_S P dy \wedge dz + Q\, dz \wedge dx + R\, dx \wedge dy :=$$

$$\int_S P\, dy \wedge dz + \int_S Q\, dz \wedge dx + \int_S R\, dx \wedge dy :=$$

$$\int_K P(S(u,v)) \frac{\partial(Y,Z)}{\partial(u,v)}\, d(u,v) + \int_K Q(S(u,v)) \frac{\partial(Z,X)}{\partial(u,v)}\, d(u,v) +$$

$$\int_K R(S(u,v)) \frac{\partial(X,Y)}{\partial(u,v)}\, d(u,v)$$

- Dieses Oberflächenintegral läßt sich auch als skalares Oberflächenintegral darzustellen:

$$\int_S P\, dy \wedge dz + Q\, dz \wedge dx + R\, dx \wedge dy =$$

$$\int_K \left\langle f(S(u,v)), \frac{\partial S}{\partial u} \times \frac{\partial S}{\partial v} \right\rangle d(u,v) =$$

$$\int_K \langle f(S(u,v)), n(u,v) \rangle \left\| \frac{\partial S}{\partial u} \times \frac{\partial S}{\partial v} \right\| d(u,v) =$$

$$\int_S \langle f, n \rangle\, d\sigma$$

  $n(u,v)$ ist der Einheitsnormalenvektor der Fläche S an $S(u,v)$.

Univariater Romberg–Algorithmus

- Die univariate Trapezregel für das Integral $If := \int_a^b f(x)\,dx$ lautet

$$If \approx T_k f := \frac{b-a}{k}\left(\frac{1}{2}f(x_0) + \sum_{j=1}^{k-1} f(x_j) + \frac{1}{2}f(x_k)\right).$$

  mit $x_j = a + j\frac{b-a}{k}$.
- **Euler–Maclaurin–Summationsformel.**
  Sei $f \in C^{2m}([a,b])$. Dann gilt

$$If - T_k f = \sum_{j=1}^{m-1} c_{2j} \left(f^{(2j-1)}(b) - f^{(2j-1)}(a)\right) h^{2j} + c_{2m} f^{(2m)}(t)(b-a)h^{2m}$$

  für $t \in [a,b]$ und den von f unabhängigen Konstanten c_{2j}.
- Damit kann der Fehler der Trapezregel dargestellt werden als

$$If - T_k f = K_2 h^2 + K_4 h^4 + K_6 h^6 + K_8 h^8 + \cdots + K_{2m} h^{2m}$$

  Dabei ist $m \in \mathbb{N}$ beliebig, solange $f \in C^{2m}([a,b])$.
- Die Konstanten K_{2j} hängen nicht von der Schrittweite h, sondern nur von f selbst ab. Damit lassen sich einzelne Fehlerterme über die Richardson–Extrapolation eliminieren,

$$\left.\begin{aligned} T_i^0 &:= T_{2^i k} f, \\ T_i^{j+1} &:= \frac{4^{j+1} T_i^j - T_{i-1}^j}{4^{j+1} - 1} \end{aligned}\right\} \quad \text{mit } j = 0 : i \text{ für } i \in \mathbb{N}_0.$$

Bivariater Romberg–Algorithmus

- Die bivariate Trapezregel für $If := \iint_A f(x,y)\,d(x,y)$ wobei A ein zwei–dimensionales Intervall ist, erhalten wir durch zweimaliges Anwenden der univariaten Trapezregel,

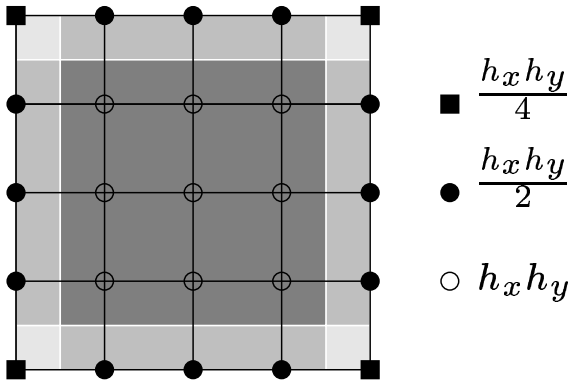
$$If \approx \int_a^b \underbrace{T_k f(x,\cdot)}_{=:F(x)}\,dx$$

$$= \int_a^b h_y \left(\frac{1}{2}f(x,c) + \sum_{j=1}^{k_y-1} f(x,c+jh_y) + \frac{1}{2}f(x,d)\right)\,dx \approx T_k F$$

$$= \frac{1}{2}h_x h_y \left(\frac{1}{2}f(a,c) + \sum_{j=1}^{k_y-1} f(a,c+jh_y) + \frac{1}{2}f(a,d)\right) +$$

$$\sum_{i=1}^{k_x-1} h_x h_y \left(\frac{1}{2}f(a+ih_x,c) + \sum_{j=1}^{k_y-1} f(a+ih_x,c+jh_y) + \frac{1}{2}f(a+ih_x,d)\right) +$$

$$\frac{1}{2}h_x h_y \left(\frac{1}{2}f(b,c) + \sum_{j=1}^{k_y-1} f(b,c+jh_y) + \frac{1}{2}f(b,d)\right).$$



- Den Fehler können wir durch zweimaliges Anwenden von der Euler–Maclaurin–Summationsformel bestimmen. Dies ergibt dieselbe Form wie für die univariate Trapezregel und wir können die Richardson Extrapolation genau wie im univariaten Fall anwenden.
- Im Algorithmus wenden wir also den bivariaten Romberg–Algorithmus auf jedes einzelne Flächenstück an und summieren die Ergebnisse.

Flächeninhalt und Volumen

- Der **Flächeninhalt** ist definiert als

$$A(S) := \int_S 1 \, d\sigma,$$

d.h. pro Bézierfläche S_i berechnen wir das Integral

$$\iint_{[0,1]^2} \left\| \frac{\partial S_i}{\partial u} \times \frac{\partial S_i}{\partial v} \right\| \, d(u,v).$$

- Das **Volumen** einer geschlossenen Fläche ist

$$V(S) = \int_V 1 \, dV = \int_V \operatorname{div} \begin{pmatrix} x \\ 0 \\ 0 \end{pmatrix} \, dV = \int_S \left\langle \begin{pmatrix} x \\ 0 \\ 0 \end{pmatrix}, n \right\rangle \, d\sigma.$$

Analog gilt

$$V(S) = \int_S \left\langle \begin{pmatrix} 0 \\ y \\ 0 \end{pmatrix}, n \right\rangle \, d\sigma = \int_S \left\langle \begin{pmatrix} 0 \\ 0 \\ z \end{pmatrix}, n \right\rangle \, d\sigma.$$

Also

$$V(S) = \frac{1}{3} \iint_{[0,1]^2} \langle S, N \rangle \, d(u,v).$$

Schwerpunkt und Trägheitsmoment

- Der **Schwerpunkt** einer Fläche S ist

$$c(S) := \frac{1}{m(S)} \begin{pmatrix} \int_S x\rho \, d\sigma \\ \int_S y\rho \, d\sigma \\ \int_S z\rho \, d\sigma \end{pmatrix}$$

  mit der Masse

$$m(S) := \int_S \rho \, d\sigma.$$

- Das **Trägheitsmoment** $J(S)$ einer Fläche S mit der Dichte ρ bzgl. einer Rotationsachse ist

$$J(S) := \int_S r^2 \rho \, d\sigma.$$

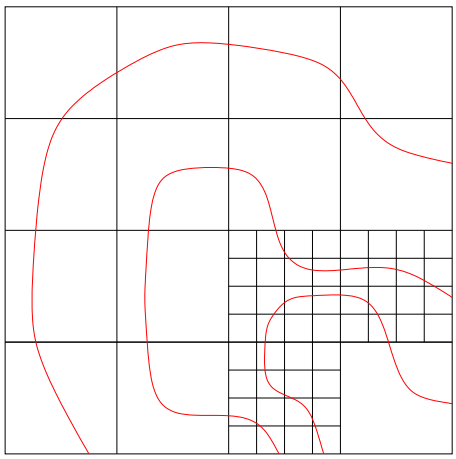
  $r : S \to \mathbb{R}$ weist jedem Punkt der Fläche seinen senkrechten Abstand zu der Rotationsachse zu. Dabei geben wir $r^2\rho$ als G–Spline–Funktion auf der Fläche an.

Integrale und Flächenschwerpunkte verschiedener Objekte

Obj.	Flächenst.	Volumen	[Iter.]	Schwerpunktintegrale		Schwerpkt.
	prec	Fläche	[Iter.]	x	[Iter.]	x
				y	[Iter.]	y
				z	[Iter.]	z
Würfel	96	0.61	[3.1354]	1.78	[3.5625]	0.5
	1.042e-05	3.57	[4.25]	1.78	[3.5625]	0.5
				1.78	[3.5625]	0.5
Verd.	992	13.44	[4.1069]	65.43	[4.5847]	1.38
Würfel	1.008e-06	47.54	[4.2480]	66.01	[4.5796]	1.39
				63.15	[4.4516]	1.33
Acht	800	11.23	[3.9888]	100.36	[4.6563]	2.50
	1.25e-06	40.15	[3.975]	60.22	[4.3975]	1.50
				20.07	[3.8625]	0.50
Verd.	928	13.04	[4.8114]	113.48	[5.3179]	2.50
Acht	1.078e-06	45.39	[4.6789]	68.09	[5.0129]	1.50
				68.09	[5.0065]	1.50
Kreuz	480	5.84	[3.7125]	30.37	[4.0583]	1.50
	1.724e-06	20.25	[3.95]	30.37	[4.0375]	1.50
				30.37	[4.0583]	1.50
T	1408	4.01	[3.6235]	24.15	[3.8274]	1.50
	7.102e-07	16.10	[3.5866]	30.35	[3.9517]	1.89
				8.05	[3.1719]	0.50

Isolinien Algorithmus

- Falls die Funktion schon wenigstens einmal rekursiv aufgerufen wurde, dann überprüfe, ob in dem ausgewählten Quadrat im Parameterbereich Linien gezeichnet werden können:
  - Geht durch jede Seite des Quadrats maximal eine Isolinie, dann zeichne die Linien für das Quadrat.
  - Falls die maximale Rekursionstiefe erreicht wurde, unterteile das verbleibende Quadrat im Parameterbereich mittels bilinearer Interpolation in Teilquadrate.
    Pro Teilquadrat werden maximal `maxlines` Linien gezeichnet.
- Zerlege den Parameterbereich in gleichgroße Teilquadrate und rufe den Algorithmus rekursiv für jedes dieser Quadrate auf.



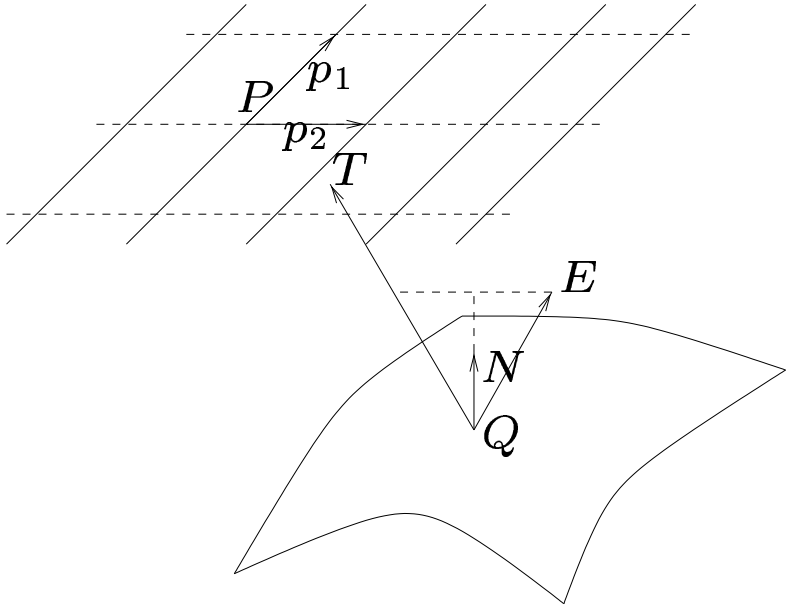
Beachte

- Haben wir die maximale Rekursionstiefe erreicht, dann legen wir über das Quadrat im Parameterbereich noch einmal ein Gitter mittels bilinearer Interpolation. Damit erhalten wir wieder Quadrate, durch deren Seiten maximal eine Linie geht.
- Es gibt zusätzlich den Parameter `maxlines` zur Begrenzung der Anzahl der Linien, die pro Quadrat gezeichnet werden. Damit wird verhindert, daß ein zu feines Gitter erzeugt wird.

Isolinien Beispiele [isoline]

- Contourplot der Affensattels [affencontour]
- Höhenlinien des Affensattels [affencontour2]
- Isolinien auf der verdrehten Acht [twisted_eight]
- Isolinien des Kreuzes [xcross]

Reflektionslinien



P	Punkt in der Ebene der Lichtquellen
p_1	Richtung der Lichtquellen
p_2	Abstand zwischen den parallelen Lichtquellen
E	Beobachtungspunkt
Q	Punkt auf der Fläche
T	Schnittpunkt des Reflektionstrahls mit der Ebene
N	Normalenvektor von S bei Q
q	Ausfallstrahl $(E - Q)$
r	Richtung des Reflektionstrahls
t_1, t_2, s	Unbekannte

$$r = q - 2\left(q - \frac{\langle q, N\rangle}{\|N\|^2}N\right) = 2\frac{\langle q, N\rangle}{\|N\|^2}N - q,$$

$$P + p_1 t_1 + p_2 t_2 = T = Q + rs$$

- Ist t_2 ganzzahlig, dann trifft der Reflektionstrahl eine Lichtquelle.
- Ist s negativ, dann findet die Reflektion "hinter" der Fläche statt.

Reflektionslinien Beispiele [reflection]

- Reflektionslinien für Irregularität der Ordnung 3 [3ref]
- Reflektionslinien für Irregularität der Ordnung 5 [5ref]
- Reflektionslinien für Irregularität der Ordnung 6 [6ref]
- Reflektionslinien auf der Acht [eight_ref]

Krümmung

- **Gauß–Krümmung**:

$$K = \det(dn_p) = \frac{eg - f^2}{EG - F^2}$$

- **Mittlere Krümmung**:

$$H = -\frac{1}{2}\,\text{spur}(dn_p) = \frac{1}{2}\frac{eG - 2fF + gE}{EG - F^2}$$

- **Hauptkrümmungen**:

$$k_i = H \pm \sqrt{H^2 - K}$$

(E, F, G sind die Koeffizienten der ersten Normalform, e, f, g sind die Koeffizienten der zweiten Normalform)

Beachte

- Nachdem die Formeln für biquadratische G–Splines an den Irregularitäten nur aus den Bedingungen für tangentiales Zusammensetzen der Flächenstücke erzeugt wurden, ist die Krümmung nicht stetig, da e, f, g von den zweiten Ableitungen der Parametrisierung abhängen.

Beispiele [curv]

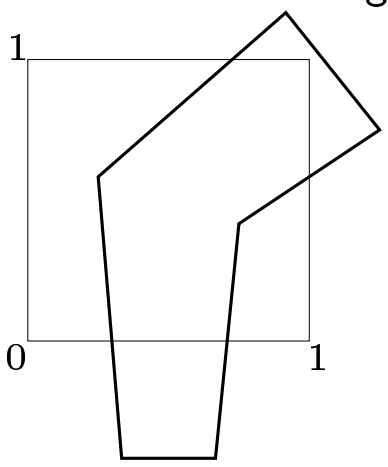
- Gauß'sche Krümmung für Irregularitäten der Ordnung 3, 4, 5 [{3,5,6}curv]
- Mittlere Krümmung des Whitneyschen Regenschirms [whitney_umbrella]
- Gauß'sche Krümmung des T [T_curv]
- Gauß'sche Krümmung des Affensattels [affen_{gauss,func}]

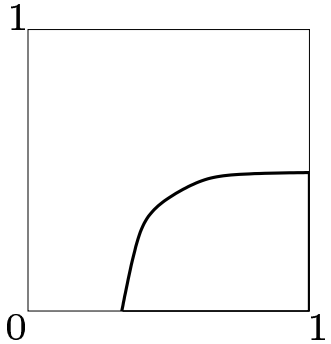
Trimming (1)

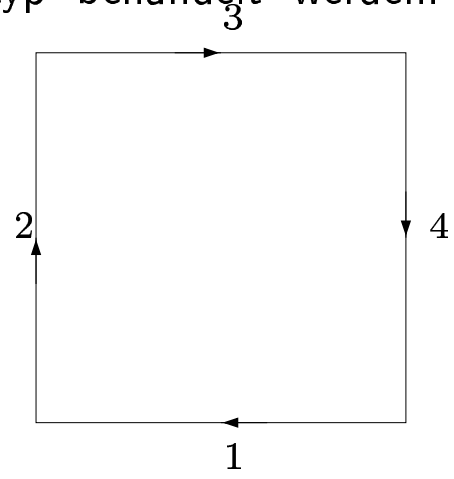
- Ausschneiden von implizit gegebenen Volumina aus der Fläche:

$$x^T M x + v^T x < c$$

- Trimmkurve im Parameterbereich $[0,1]^2$, die über einen Kontrollpunkt der quadratischen G–Spline Fläche einen Teil aus einem einzigen Bézierflächenstück ausschneidet:
  - Polygonzug:
    - Die Kontrollpunkte der Trimmkurve legen die Ecken des Polygonzuges fest.
  - Quadratische Splinekurve:
    - Kontrollpunkte werden über die Bedingungen der geometrischen Stetigkeit in jeweils 3 Bézierpunkte pro Kurvensegment umgewandelt.
    - Ist die Kurve nicht geschlossen, wird sie über die Schnittpunkte mit dem Rand von $[0,1]^2$ erweitert. Existieren keine Schnittpunkte, werden die Enden der Kurve in Richtung der Tangente verlängert, bis ein Schnittpunkt entsteht.
    - Die Splinekurve wird in einen Polygonzug umgewandelt und kann damit analog zum ersten Trimkurventyp behandelt werden.



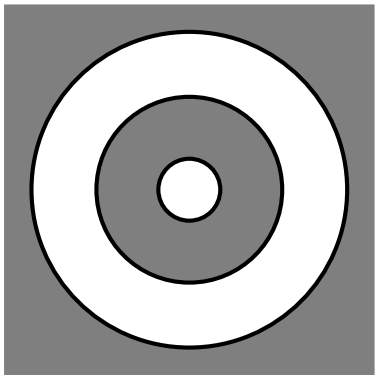




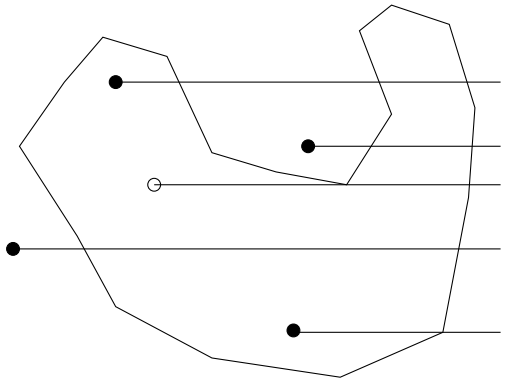
  - Wir gehen davon aus, daß der Polygonzug immer im Uhrzeigersinn durchlaufen wird und alles, was rechts vom Polygonzug liegt, ausgeschnitten wird.

Trimming (2)

- Überlappende Trimmbereiche
  - Liegt ein Punkt im Parameterbereich in einer geraden Anzahl von Trimmbereichen, dann wird er nicht ausgeschnitten.
  - Liegt er in einer ungerade Anzahl von Trimmbereichen wird er ausgeschnitten.
  - Implizite Trimmbereiche im Raum der Fläche werden immer ausgeschnitten.



Trimmkurventest



- Zum Testen, ob ein Punkt $(\mathtt{u}, \mathtt{v})$ im Trimmbereich liegt, zählen wir, wie oft eine Linie von dem Punk in Richtung $(1, 0)$ den Polygonzug schneidet.
  - Ist die Anzahl der Schnitte gerade, sind wir außerhalb des Trimmbereichs.
  - Ist die Anzahl der Schnitte ungerade, sind wir innerhalb des Trimmbereichs.
- Schnittbedingung für $\{(u, v)+t(1, 0) : t > 0\} \cap \mathsf{line}((x_1, y_1), (x_2, y_2))$:

$$\begin{aligned} \mathtt{v} &> \min\{y_1, y_2\} && \wedge \\ \mathtt{v} &\leq \max\{y_1, y_2\} + \mathtt{Epsilon} && \wedge \\ \mathtt{u} &\leq \max\{x_1, x_2\} + \mathtt{Epsilon} && \wedge \\ |y_1 - y_2| &\geq \mathtt{Epsilon} && \wedge \\ \left(|x_1 - x_2| < \mathtt{Epsilon} \vee \mathtt{u} < (x_2 - x_1)\frac{\mathtt{v} - y_1}{y_2 - y_1} + x_1\right). \end{aligned}$$

Beachte

- Es gibt Fälle, bei denen dieses Verfahren nicht funktioniert.
- Durch ändern der Auflösung des Testgitters kann dies umgangen werden.
- Durch zusätzliche Tests, etwa das Zählen der Schnitte mit Linien in andere Richtungen, lassen sich solche Fälle weitere reduzieren.

Trimming Algorithmus

- Lege über das Intervall $[\texttt{u0}, \texttt{u1}] \times [\texttt{v0}, \texttt{v1}]$ ein Gitter mit $\texttt{resolution}^2$ Punkten.
- Lege ein boolean Array an, daß für jeden Gitterknoten angibt, ob er innerhalb oder außerhalb der Trimmbereiche liegt:
  - Zähle die Anzahl der Trimmbereiche, in denen der Knoten liegt. Ist die Anzahl ungerade, dann liegt der Knoten innerhalb eines Trimmbereichs.
  - Überprüfe, ob der zu dem Knoten gehörende Punkt durch eine implizite Funktion aus der Fläche ausgeschnitten wird.
- Bearbeite nun jedes Quadrat im Parameterbereich, das durch die zwei Gitterknoten mit den Indizes `[ui,vi]` und `[ui+1,vi+1]` bzgl. des boolean Arrays bestimmt ist:
  - Liegt keine der Ecken in einem Trimmbereich, dann bearbeite den zu diesem Quadrate gehörenden Teil der Fläche analog zum nicht–getrimmten Fall.
  - Liegt wenigstens eine der Ecken des Quadrats in einem Trimmbereich:
    - Ist die maximale Rekursionstiefe für den Trimmalgorithmus noch nicht erreicht, rufe den Algorithmus rekursiv für dieses Quadrat auf.
    - Sonst überprüfe, ob eines der durch die Ecken des Quadrats bestimmten Dreiecke außerhalb des Trimmbereichs liegt und bearbeite es analog zum nicht–getrimmten Fall.

Beachte

- Die Auflösung des Testgitters muß groß genug gewählt werden, da es sonst möglich ist, daß kleine Trimmbereiche ausgelassen werden.
- Bei großer maximaler Rekursionstiefe arbeitet der Algorithmus am Rand der Trimmbereiche sehr lange.

Trimming Beispiele [trim]

- Getrimmte Flächen
  - Getrimmtes Kreuz [tcross]
  - Getrimmter Würfel [tcube]
  - Getrimmte G–Spline–Fläche mit einer Irregularität der Ordnung 5 [tgspline5]
- Funktionen auf getrimmten Flächen
  - Funktion auf einer getrimmten Fläche mit einer Irregularität der Ordnung 3 [3net_gspline]
  - Funktion auf dem getrimmten Würfel [cube_func]

LiLit

- Einlesen der ODL Datei über einen mit bison++ erzeugten Parser. Hierbei werden die Objekte in der ODL Datei in die interne Darstellung durch C++ Klassen umgewandelt.
- Bearbeiten der Objekte, die angezeigt bzw. integriert werden sollen. Zusätzlich müssen auch Objekte bearbeitet werden, die indirekt von diesen Objekten benutzt werden. In diesem Schritt werden z.B. die Flächenkontrollnetze zusammen mit den zugehörigen Funktionskontrollnetzen in Bézierkontrollnetze umgewandelt.
- Berechnen der Oberflächenintegrale.
- Anzeigen der Objekte mit OpenGL. Hier werden die Bézierkontrollnetze, Linien, etc. mit Hilfe der im "Bearbeiten"–Schritt erzeugten Daten gezeichnet. In diesem Schritt werden auch die Isolinien gezeichnet.
  Alternativ kann der OpenGL Feedback Buffer verwendet werden, um eine encapsulated PostScript Datei aus den noch nicht gerasterten OpenGL Primitives zu erzeugen.
- Interaktiver Modus.

Verbesserte Struktur für LiLit