1.6 Implementierungsdetails

1.6.1 Triangulierungen und Polytope

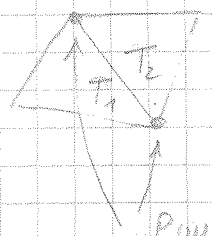
Ein Polytop ist die konvexe Hülle seiner 0-Randpolytope

↪ zur Spezifikation genügt Koordinatenliste

Eine Triangulierung ist durch seine Zellen festgelegt

↪ zur Spezifikation genügen Koordinatenlisten der Zellpolytope

Technisches Problem (Genauigkeit / Speicherknappheit)



Punkte gehören zu T_1 und T_2 → Koordinaten doppelt speichern (teuer!)

Überprüfung der Gleichheit (gefährlich wg. Rundung!)

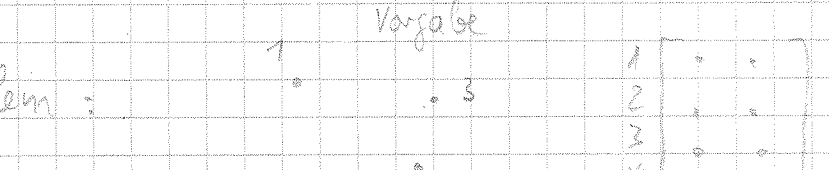
Triangulierung T wird daher beschrieben durch

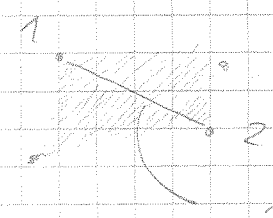
Koordinatenliste zu $\dim_T^{-1}(0)$ und Zellbeschreibung mit Punktnummern

Um z.B. die Fähigkeit $\cdot \dim_T^{-1}$ von T abzubilden

müssen alle Polytope identifiziert werden

technisches Problem:

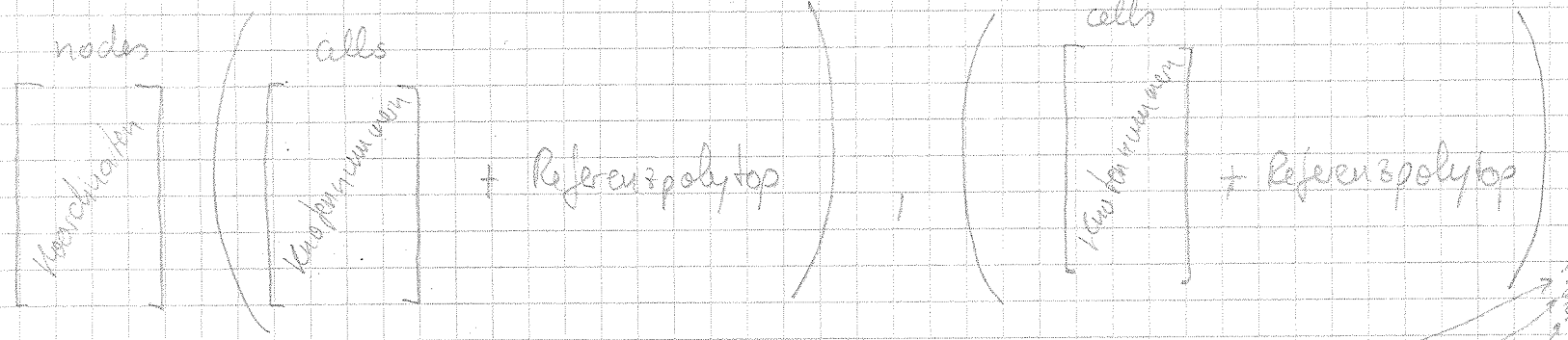


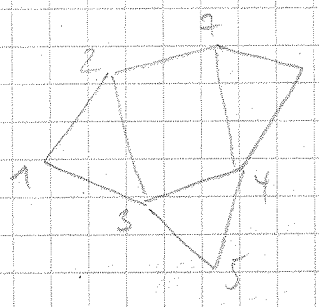


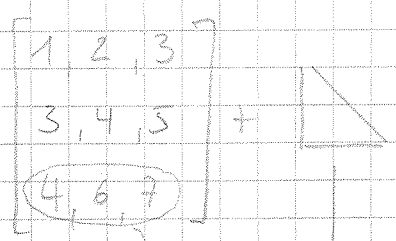
1 → 2 ist kein Randpolytop

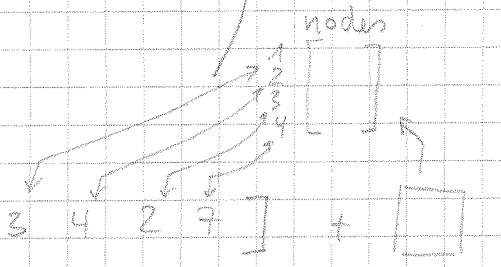
Detektion möglich ... aber aufwendig

Vereinfachung der Beschreibung einer Triangulierung:



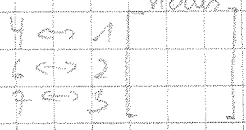




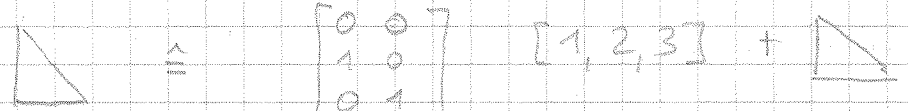


↝ Randpolytope des Referenzpolytops beschreiben Ränder oder Seiten!

ist auch Triangulierung mit nodes



Zwischenstand: Polytop ist spezielle Triangulierung aber Triangulierung braucht Polytope zur Konstruktion



geht nicht!

Ausweg: Rekursion Beschreibe Polytop durch Triangulierung seines Randes

Daher ist Konstruktion von Triangulierungen entscheidend.

Da Polytope spezielle Triangulierungen sind, erben sie alle Fähigkeiten.

Beim Programmieren der Fähigkeiten von Triangulierungen können die entsprechenden Fähigkeiten der Referenzpolytope benutzt werden

Wir unterscheiden: homogene Triangulierung ... alle Zellen äquivalent
heterogene Triangulierung ... sonst

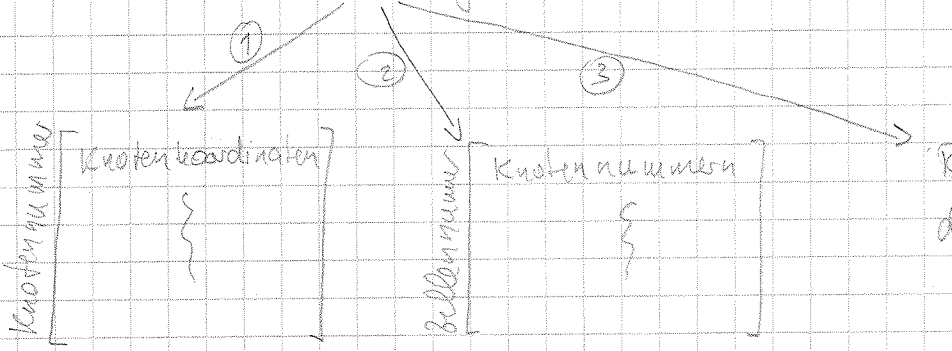
Wir programmieren zunächst den Spezialfall

Klasse die große Datenmengen enthalten von Klasse handle ableiten (*)

homTriangulation < triangulation < handle

Konstruktor hat keine Argumente
Ergebnis ist leere Triangulierung

Konstruktor hat 3 Argumente



* Wird ein Objekt der handle Klasse als Funktionsargument übergeben, so werden nicht die Daten sondern nur ein Zeiger auf die Daten übergeben ... das bedeutet auch, dass Funktionen die Dateninhalte von handle-Objekten modifizieren können (also Vorsicht)

Fähigkeiten der Klasse triangulation (T sei Objekt der Klasse)

T.npoly	Anzahl Polytope in T
T.nnodes	Anzahl Knotennummern in T (kann größer als Anzahl 0-Polytope sein, wenn nicht alle Knoten genutzt werden)
T.nref	Anzahl Referenzpolytope in T
T.ncells	Anzahl der Zellen in T
T.dim	Dimension von T
T.edim	Einbettungsdimension von T
T.dim2poly(d)	Polytopnummern der Polytope mit Dimension d
T.poly2dim(p)	Dimension des p-ten Polytops
T.ref(r)	r-tes Referenzpolytop
T.poly2cells(p)	Zellennummern in denen p enthalten ist
T.cell2poly(c)	Polytopnummer der Zelle c
T.poly2nodes(p)	Knotennummern aus denen p erzeugt wird
T.nodes2poly([...])	Polytopnummer zu vorgegebener Knotenmenge (NaN falls Polytop nicht in T)
T.poly2ref(p)	Referenzpolytopnummer zu p
T.ref2poly(r)	Nummern der Polytope mit Referenzpolytop r
T.cell2nodes(c)	Knotennummern der Zelle c

T. cell2ref(c)	Referenzpolytopnummer zur Zelle c
T. poly2aff(p)	affine Abbildung, die T.ref(poly2ref(p)) auf p abbildet
T. loc2glob(c, l)	globale Polytopnummer des l-ten lokalen Polytops in Zelle c
T. boundary	Randtriangulierung

Tipps zur Umsetzung

poly2dim : als property (npoly × 1 Vektor) mit Dimensionswerten

dim2poly : mit find-Befehl auf poly2dim

ref : Referenzpolytope in matlab-cell-array abspeichern

cell2poly : als property (ncells × 1 Vektor)

poly2cell : gespeichert als matlab-cell-array: C poly2cell{p} enthält Vektor mit Zellennummern

poly2nodes : verschiedene Polytope können unterschiedliche Knotenanzahl haben, nutze C ref2poly{r}(s,:) um Knotennummern des Polytops p bzw. zugehörigem Referenzpolytop r = poly2ref(p) in Zeile s = Vpoly2ind(p) zu speichern. Der Vektor Vpoly2ind muss dazu angelegt werden.

nodes2poly Speicherung ist hier aufwendig (außer bei Spezialfällen wie konkrete Referenzpolytope). Berechnungsmöglichkeit bei Knotenliste k

$p = T.Vnode2poly(k)$ Vektor der zu jedem Knoten die Polytopnummer enthält

$c \in \bigcap_i T.poly2cell(p(i))$ matlab Befehl intersect

falls kein c existiert, enthält T dieses Polytop nicht,
da keine Zelle die angegebenen Punkte enthält.

K = T.cell2nodes(c) sind Knoten einer Zelle die das Polytop enthält

[dummy, kloc] = ismember(K, K) ergibt Knotennummern im Referenzpolytop der Zelle c.

ploc = T.ref(T.cell2ref(c)).nodes2poly(kloc) lokale Polytopnummer

p = T.loc2glob(c, ploc)

poly2ref als property (vektor)

ref2poly mit find auf poly2ref

cell2nodes als property (Vektor)

cell2ref als property (Vektor)

poly2ref nutze Methode des zugehörigen Referenzpolytops mit Koordinaten der Knoten, die mit poly2nodes erweitert werden können

loc2glob direkt als property (Matrix) wenn nur eine Referenzzelle vorhanden ist
sonst als Methode basierend auf cell-array

boundary:

T.dim = 0 ⇒ leere Triangulierung

pot = T.dim2poly(T.dim − 1) potentielle Zellen

Sammle i in Vektor I mit length(T.poly2cell(pot(i))) == 1

I leer ⇒ leere Triangulierung

p = pot(I) sind neue Zellen

r = T.poly2ref(p) zugehörige Referenzpolytope

R = unique(r) ohne Dubletten

length(R) == 1 homogene Triangulierung; sonst heterogene Triangulierung

im homogenen Fall lokale Indices von p in Cref2poly{R} ermitteln

und Knotennummern herausgreifen und in C speichern

unbenötigte Knoten herausfiltern:

Kold = unique(C); sortierte alte Knotennummern

Knew = 1 : length(Kold); neue Knotennummern

Trans = zeros(1, Kold(end));

Trans(Kold) = Knew; Transformationsvektor

Cnew = Trans(C); neue Zellenbeschreibung

Nnew = T.Vnodes(Kold), neue Knotenkoordinaten ⇒ homTriangulation aufrufen

Grober Ablauf zum Aufbau der globalen Basisfunktionen (im homogenen Fall)

Zu Beginn: npoly = 0, alle relevanten Felder leer,

Schleife über alle Zellen c
  K = T.cell2nodes(c)
  globale Referenzpolytope entsprechen Referenzpolytopen der Referenzzelle
  Schleife über Polytope ploc von T_{ref}
  globale Knoten von ploc k = K(T_{ref}.poly2nodes(ploc))
  p = T.nodes2poly(k)
  Fall 1 p = NaN das Polytop ist noch nicht vorhanden
    npoly = npoly + 1 wird die Nummer p des Polytops
    Dimension T_{ref}.poly2dim(ploc) in T.poly2dim eintragen
    Referenzpolytop entsprechend eintragen in poly2ref
    Wenn Dimension = 0 p in Vnode2poly eintragen
    C_{ref}2poly, Vpoly2 sind anpassen
    Wenn Dimension = T.dim cell2poly anpassen
    poly2cell anpassen
    loc2glob anpassen

Fall 2: Polytop schon vorhanden

p in loc2glob eintragen

c in CPoly2all vermerken

nächsten lokalen Polytop bearbeiten

nächste Zelle bearbeiten

Polytope sind spezielle homogene Triangulierungen (es gibt nur eine Zelle also auch nur eine Referenzzelle)

also polytope < homTriangulation

0-Polytope, 1-Polytope, Simplex(d), Cube(d) sind spezielle Polytope

also point < polytope

segment < polytope

simplex < polytope

cube < polytope

Die Triangulierung zu einem Polytop P ist

$$T = \rho(P) \cup \{P\}$$

Damit kann der Konstruktor allgemein formuliert werden:

Parameter hängen vom Spezialfall ab

```
function p=polytope(varargin)
    p@homTriangulation();      leere homogene Triangulierung erzeugen
    p.rho = p.constructBoundary(varargin{:});   diese Methode muss abhängig vom
    Daten von rho sinngemäß nach                Spezialfall implementiert werden
    p übertragen; ein Referenzpolytop hinzufügen (p selbst)
    eine Zelle eintragen mit Knoten [1:p.nnodes]
    alle anderen Polytope gehören zu dieser Zelle; etc.
end
```

Die Methode boundary aus Triangulierung kann überladen werden:

```
function dp = boundary(p)
    dp = p.rho;
end
```

In polytope ist unklar, was constructBoundary tun soll,
deshalb wird die Methode abstrakt gelassen

```
method (Abstract, Access = protected)
    rho = constructBoundary (p, varargin)
end
```

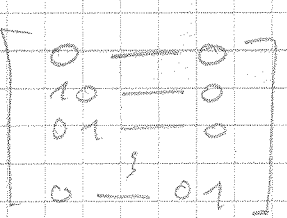
Als Resultat lassen sich von der Klasse polytope keine Objekte erzeugen,
d.h. es kann nur spezielle Polytope geben (wie point, segment etc.)
bei denen constructBoundary implementiert ist

z.B. in der Klasse simplex

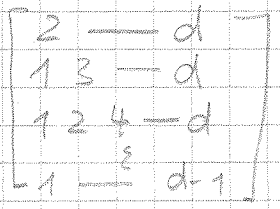
```
methods (Access = protected)
    function rho = constructBoundary (p, d)
        Fall d = 1:   rho = segment;
```

✓ in simplex ist bekannt, dass nur die Dimension d im Konstruktor übergeben werden soll

Fall $d > 1$. Punkte $N =$



Zellenbeschreibung $C =$



rho = homTriangulation(N, C, simplex(d-1));

end

end

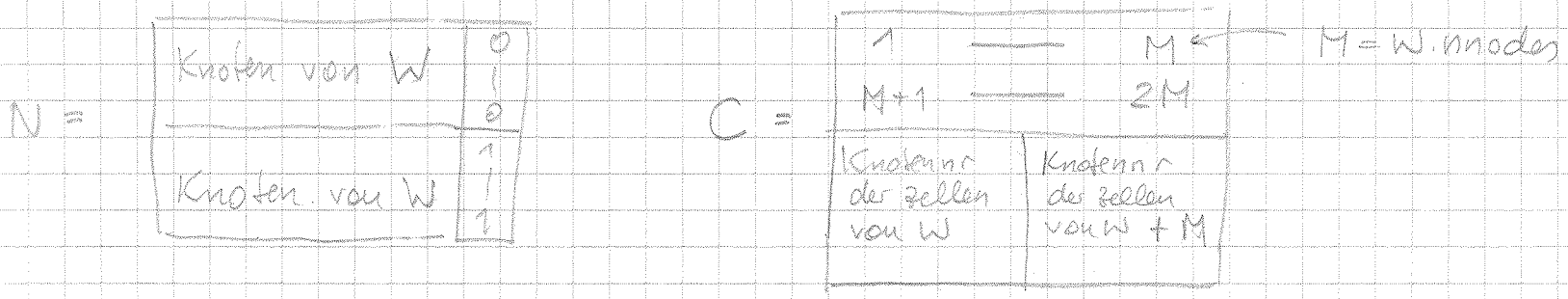
Oder in Klasse cube

methods (Access = protected)

function rho = constructBoundary(p, d)

Fall $d = 1$: rho = segment;

Fall $d > 1$: W = cube(d-1);



rho = homTriangulation (N, C, W);

end

end

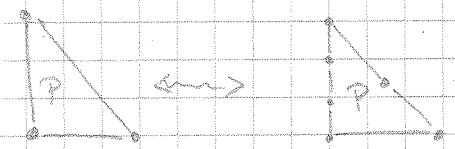
weitere Fähigkeit eines Polytops P

P. nodes2aff (K) wobei K = [Koordinaten ; ; ;] P. nodes bereit hat

Das Ergebnis soll eine affine Abbildung sein, die den i-ten Knoten von P auf K(i,:) abbildet (das geht natürlich nur für spezielle K)

Auch diese Methode ist abstrakt, da die Abbildung nur für konkrete Polytope angegeben werden kann.

Die Randpolytope eines 2-Polytops in $\mathbb{R}^2$ sind nicht eindeutig



aber man kann eine eindeutige kleinstmögliche Randbeschreibung wählen:

Definition 1.16 Sei P ein k-Polytop in $\mathbb{R}^d$ mit $0 < k \leq d$
Wir nennen (Q, A) eine Darstellung von P, wenn Q ein k-Polytop in $\mathbb{R}^k$ ist und $A : \mathbb{R}^k \longrightarrow \mathbb{R}^d$ eine injektive, affine Abbildung ist mit $A(Q) = P$.

Beobachtung: Sei P ein d-Polytop in $\mathbb{R}^d$ und Q ein Randpolytop von P mit Darstellung $(\tilde{Q}, A)$. Dann ist $\text{ext}(Q) = A\left((A^{-1}(\partial P))^0\right)$ unabhängig von der Wahl der Darstellung.
Dabei ist $\text{ext}(Q)$ Bild einer offenen, konvexen, beschränkten nicht leeren Menge und $Q \subset \text{ext}(Q) \subset \partial P$

Beweis: Seien (U, A) (V, B) zwei Darstellungen von Q

Wähle $\bar{q} \in Q$ dann gibt es eindeutige $\bar{u} \in U$ $\bar{v} \in V$ mit $A(\bar{u}) = \bar{q} = B(\bar{v})$

Definiere $a(u) = A(u + \bar{u}) - A(\bar{u})$ $b(v) = B(v + \bar{v}) - B(\bar{v})$

Dann ist $\ker a = \ker b = \{0\}$ $\mathrm{Bild}\, a = \mathrm{Bild}\, b$

Sei $L = (a^T a)^{-1} a^T b : \mathbb{R}^k \to \mathbb{R}^k$

dann $a L = \underline{a (a^T a)^{-1} a^T} \; b = b$

Orthogonalprojektion auf $\mathrm{Bild}\, a = \mathrm{Bild}\, b$

und damit L injektiv also invertierbar.

$$A\left(A^{-1}(\partial P)^\circ\right) = \bar{q} + a\left(A^{-1}(\partial P)^\circ - \bar{u}\right) = \bar{q} + b\left(L^{-1} A^{-1}(\partial P)^\circ - L^{-1}\bar{u}\right)$$

$$u \in A^{-1}(\partial P) \iff A(u) \in \partial P \iff \bar{q} + a(u - \bar{u}) \in \partial P \iff u \in \bar{u} + a^{-1}(\partial P - \bar{q})$$

$$L^{-1} A^{-1}(\partial P)^\circ = L^{-1}\bar{u} + L^{-1} a^{-1}(\partial P - \bar{q})^\circ = L^{-1}\bar{u} + b^{-1}(\partial P - \bar{q})^\circ$$

$$A\left(A^{-1}(\partial P)^\circ\right) = \bar{q} + b\left(b^{-1}(\partial P - \bar{q})^\circ\right)$$

$$v \in b^{-1}(\partial P - \bar{q}) \iff b(v) + \bar{q} \in \partial P \iff B(v + \bar{v}) \in \partial P \iff v \in B^{-1}(\partial P) - \bar{v}$$

$$A\left(A^{-1}(\partial P)^\circ\right) = \bar{q} + b\left(B^{-1}(\partial P)^\circ - \bar{v}\right) = B\left(B^{-1}(\partial P)^\circ\right)$$

$T = A(\partial P)$ enthält $\tilde{Q}$ und $\tilde{\tilde{Q}}$ offen $\Rightarrow$ $T^\circ \neq \emptyset$

Da $\bar{P}$ beschränkt, $R \geq \|Au\| = \|a(u-\bar{u}) + \bar{q}\| \geq \|a(u-\bar{u})\| - \|\bar{q}\| \geq \left(\langle a^T a (u-\bar{u}), u-\bar{u} \rangle \right)^{1/2} - \|\bar{q}\|$

$\geq \underbrace{\lambda_{min}}_{>0} \|u-\bar{u}\| - \|\bar{q}\|$ für $u \in T$

also $\|u-\bar{u}\| \leq \frac{1}{\lambda_{min}} (R + \|\bar{q}\|)$ also T beschränkt

zur Konvexität zeige $A(\mathbb{R}^k) \cap T = \emptyset$ dann $\underbrace{\underbrace{A(\mathbb{R}^k)}_{\text{konvex}} \cap \underbrace{\bar{P}}_{\text{konvex}}}_{\text{konvex}} = A(\mathbb{R}^k) \cap \partial P = A(T)$

$u, v \in T$

$\lambda A(u) + (1-\lambda) A(v) \in A(T)$ d.h. $\lambda u + (1-\lambda) v \in T$

$\lambda \bar{q} + \lambda a(u-\bar{u}) + (1-\lambda)\bar{q} + (1-\lambda) a(v-\bar{u})$

$= \bar{q} + a(\lambda u + (1-\lambda) v - \bar{u}) = A(\lambda u + (1-\lambda) v)$

wobei noch $u, v \in T^\circ \Rightarrow \lambda u + (1-\lambda) v \in T^\circ$ zu zeigen ist ... beides folgt aus

Beobachtung: Sei $C \subset \mathbb{R}^d$ konvex $x \in C^\circ$, $y \in C$ $\lambda \in (0,1]$

Dann ist $y + \lambda(x-y) \in C^\circ$

Beweis: $x \overset{F}{\longmapsto} y + \lambda(x-y)$

Es gibt $\varepsilon > 0$: $B_\varepsilon(x) \subset C^\circ$ $(x \in C^\circ)$

außerdem $F(B_\varepsilon(x)) \subset C$ (Konvexkombination)

außerdem $F(B_\varepsilon(x)) = y + \lambda(B_\varepsilon(0) + x - y) = F(x) + B_{\lambda\varepsilon}(0) = B_{\lambda\varepsilon}(F(x))$

da: $B_{\lambda\varepsilon}(F(x)) \subset C$ ist $F(x) \in C^\circ$ ∎

$u, v \in T^\circ \Rightarrow u + \lambda(v-u) \in T^\circ$ für alle $\lambda \in (0,1)$ ✓

Ann: $x \in \underbrace{A(\mathbb{R}^k)}_{\text{konvex}} \cap \bar{P}$ sei $q \in Q \subset \partial P \cap A(\mathbb{R}^k) \subset A(\mathbb{R}^k) \cap \bar{P}$

$\tilde{x} = A^{-1}(x)$ $\tilde{q} = A^{-1}(q) \in \tilde{Q}$ offen d.h. $u = \tilde{x} + (1+\varepsilon)(\tilde{q} - \tilde{x}) = \tilde{q} + \varepsilon(\tilde{q} - \tilde{x}) \in \tilde{Q}$

für ε genügend klein

wegen $\tilde{q} = \tilde{x} + \frac{1}{1+\varepsilon}(u - \tilde{x}) = u + \frac{\varepsilon}{1+\varepsilon}(\tilde{x} - u)$

ist $q = A(u) + \frac{\varepsilon}{1+\varepsilon}(x - A(u)) \in P$ ↯

∂P

Ist P ein d-Polytop in $\mathbb{R}^d$ und ist R eine zugehörige Menge von Randpolytopen

Dann $R_{ext} := \{ ext(S) \mid S \in R \}$ (identische Erweiterungen entfernen) endlich

und $R_{min} := \{ T \in R_{ext} \mid \forall S \in R_{ext} \setminus \{T\} : T \not\subset S \}$ (Erweiterungen die in anderen enthalten sind entfernen)
endlich

Beobachtung: $\partial P = \bigcup R \subset \bigcup R_{ext} = \bigcup R_{min} \subset \partial P$ also $\bigcup R_{min} = \partial P$

Beobachtung: $S, T \in R_{min}$ $S \neq T$ $\Rightarrow$ $S \cap T = \emptyset$

Beobachtung: Sei $S \in R_{min}$ mit $S = A(A^{-1}(\partial P)^\circ)$ wobei $(\tilde{Q}, A)$ Darstellung von $Q \in R$

Dann ist $A(\partial \tilde{Q}) = \bigcup_{\substack{T \in R_{ext} \\ T \cap A(\partial \tilde{Q}) \neq \emptyset}} T$ also S Polytop und R_{min} Menge von Randpolytopen

Beobachtung: R^1, R^2 Mengen von Randpolytopen zu P

Dann $R^1_{min} = R^2_{min}$.

Seien $S_1, S_2 \in R_{min}$ $\dim S_i \neq 1$ $S_i = ext(Q_i)$ $Q_i : (\tilde{Q}_i, A_i)$ $H_i := A_i(R^{k_i})$

$\tilde{Q}_i : R^{k_i} \to R^d$

Sei $x \in S_1 \cap \overline{S_2}$

Dann $x \in H_1 \cap H_2$ und $H_i = x + U_i$ $\dim U_i = k_i$

Da $S_1 \not\subset S_2$, $S_2 \not\subset S_1$ ist $U_1 \not\subset U_2$, $U_2 \not\subset U_1$ und daher

sind U_1, U_2 echte Teilräume von $U = U_1 + U_2$ $H := x + U$

Wähle Basis $b_1, \dots, b_\ell, b_{\ell+1}, \dots, b_m$ von U

so dass $b_{\ell+1}, \dots, b_m$ Basis von U_2 und $b_1, \dots, b_\ell \in U_1 \setminus U_2$

Sei γ_k duale Basis d.h. $\gamma_k(b_\ell) = \delta_{k\ell}$

Betrachte $\Phi_1 = (\gamma_1, \dots, \gamma_\ell)$ $\Phi_2 = (\gamma_{\ell+1}, \dots, \gamma_m)$

dann $\operatorname{Kern} \Phi_1 = U_2$, $\operatorname{Kern} \Phi_2$ $\operatorname{span}\{b_1, \dots, b_\ell\}$

Da $x \in S_1$ und S_1 offen in H_1 gibt es $\varepsilon > 0$ mit $B_\varepsilon(x) \cap H_1 \subset S_1$

damit gibt es $\mu > 0$ so dass $x \pm \mu b_i \in S_1$ $i = 1, \dots, \ell$

Da S_2 rel offen in H_2 gibt es $q_2 \in B_\delta(q_2) \cap H_2 \subset S_2$ also $q_2 \pm \nu b_i \in S_2$ $i = \ell+1, \dots, m$