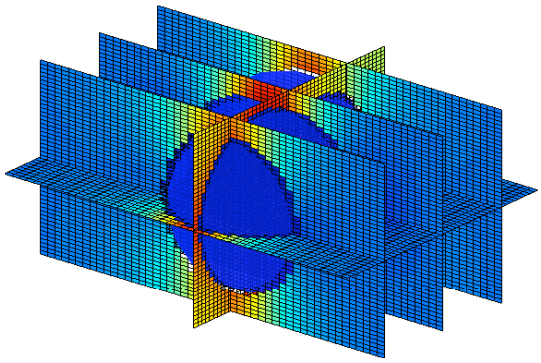
WEB-Approximation auf CSG-Objekten

Von der Fakultät Mathematik und Physik
der Universität Stuttgart
zur Erlangung der Würde eines
Doktors der Naturwissenschaften (Dr. rer. nat.)
genehmigte Abhandlung



Vorgelegt von

Stephan A. Kreitz

geboren in Heidelberg

Hauptberichter:	Prof. Dr. Klaus Höllig
Mitberichter:	Prof. Dr. Ulrich Reif
Tag der mündlichen Prüfung:	14. Juli 2008

Institut für Mathematische Methoden in den Ingenieurwissenschaften,
Numerik und geometrische Modellierung der Universität Stuttgart
2008

Danksagung

Ich möchte mich zu allererst bei Prof. Dr. Klaus Höllig bedanken, dass er mich in Stuttgart aufgenommen hat, mir ermöglicht hat diese Arbeit zu schreiben und für die Betreuung während meiner Zeit an der Universität in Stuttgart. Prof. Dr. Ulrich Reif möchte ich danken für das Erstellen des Gutachtens meiner Arbeit.
Bedanken möchte ich mich auch bei allen Mitarbeitern und Kollegen des IMNG, insbesondere bei Jörg Hörner, für die Hilfe und die vielen geduldig beantworteten Fragen.
Und last but not least möchte ich mich bei meiner Familie und ganz besonders bei meiner Freundin Amra bedanken, die mich die ganze Zeit unterstützt hat und an mich geglaubt hat.

Für meinen Vater Rolf, der das leider nicht mehr erleben durfte.

Inhaltsverzeichnis

1 Einleitung

Die Methode der Finiten Elemente hat in vielen Bereichen der Industrie, wie beispielsweise der Automobil-, Schiff-, Luft- und Raumfahrtindustrie, eine wichtige Rolle eingenommen. Auch die Wärmeausbreitung in einem abgeschlossenen Raum oder die Verteilung von Druck und Belastung bei einem Rad oder einer Felge lassen sich mit Differentialgleichungen modellieren, die mit Finite Elemente Methoden gelöst werden können. Üblicherweise wird das Gebiet, auf dem die Finite Elemente Methode angewendet werden soll, in sehr viele kleine Teile unterteilt, beispielsweise Dreiecke oder Vierecke im Zweidimensionalen oder Tetraeder und Quader im Dreidimensionalen. Man erhält daraus das so genannte Gitter. Bei der Rechenleistung heutiger Computer ist das Lösen des Finite Elemente Problems in kurzer Zeit kein Problem. Ein Gitter zu generieren, das gewissen Ansprüchen genügt, ist dagegen ein zeitaufwändiger Schritt, der vorher durchgeführt werden muss. Bei der hier verwendeten WEB-Methode entfällt dagegen der Schritt der Gittergenerierung dagegen, da ein regelmäßiges Vierecks- beziehungsweise Quadergitter verwendet wird.

1.1 Aufbau der Arbeit

Zu Beginn dieser Arbeit wird in Kapitel 2 eine Einführung in die Theorie der Finiten Elemente gegeben. Mit der Methode der Finiten Elemente sollen elliptische Differentialgleichungen numerisch gelöst werden. Deshalb werden zu Beginn des Kapitels der Begriff der elliptischen Randwertprobleme definiert und die zur weiteren Behandlung notwendigen Definitionen aufgeführt. Mit Hilfe eines Differentialoperators L wird eine elliptische Differentialgleichung, an deren Lösung Nebenbedingungen gestellt werden, durch $Lu = f$ dargestellt. Die gängigsten Nebenbedingungen sind Dirichlet- und Neumann-Randbedingungen. Des Weiteren werden weitere zur Lösung der elliptischen Differentialgleichung notwendige Grundlagen behandelt, und das Modellproblem, die Poisson-Gleichung mit homogenen Dirichlet-Randbedingungen, vorgestellt. Sobolev-Räume sind zur Approximation der Lösung notwendige Räume. Zum Abschluss des Kapitels wird der Ritz-Galerkin-Ansatz zur Lösung der Differentialgleichungen vorgestellt.
Es ist naheliegend, für die Approximation von vorgegebenen Daten Polynome zu verwenden, allerdings haben Änderungen an einem Knoten schon globalen Einfluß. Eine bessere Wahl sind deshalb Splines, stückweise definierte Polynome, die nicht nur bessere Approximationseigenschaften besitzen, sondern auch einfach darstellbar und mathematisch zu behandeln sind. Eine Basis des entsprechenden Spline-Raums bilden die so genannten B-Splines. Die wichtigsten benötigten Eigenschaften und eine Rekursionsdarstellung, die aufzeigt, wie gut B-Splines zur Verarbeitung mit dem Computer geeignet sind, werden in Kapitel 3 für univariate B-Splines angegeben. Durch das Bilden von Tensorprodukten univariater B-Splines lassen sich auf einfache Weise multivariate B-Splines erzeugen, was im weiteren Verlauf des Kapitels gezeigt wird.

Mit diesen Grundlagen wird im darauf folgenden Kapitel 4 die von Höllig, Reif und Wipper entwickelte Methode der WEB-Splines eingeführt. B-Splines scheinen keine gute Wahl für Basis-Funktionen für Finite Elemente Methoden, da ihnen die notwendige Flexibilität fehlt, um Randbedingungen ausreichend zu erfüllen. Hierzu werden die B-Splines mit einer für das gesamte Gebiet definierten Gewichtsfunktion multipliziert. Die Gewichtsfunktion wird aus den Gewichtsfunktionen der im nächsten Kapitel vorgestellten Objekte mit R-Funktionen zusammengesetzt. Das Konzept der Erweiterung und die wichtigsten Eigenschaften der WEB-Splines vervollständigen dieses Kapitel.
Im daran anschließenden Kapitel werden Quadriken als Grundobjekte beziehungsweise Bausteine für komplexere Gebiete in zwei und drei Dimensionen eingeführt. Es wurde hierfür eine elegante Darstellung der Quadriken in homogenen Koordinaten gewählt, was bedeutet, dass sich Quadriken durch $x^t Q x$ mit einer (3×3)- beziehungsweise (4×4)- Matrix Q darstellen lassen. Kombinatorik, die, im Voraus durchgeführt, die Laufzeit der Methode merklich verringert, und die für diese Arbeit wichtigen Eigenschaften der Quadriken bilden einen weiteren Teil dieses Kapitels. Abschließend wird der K-Form Algorithmus zum Lösen nichtlinearer polynomialer Gleichungssysteme vorgestellt, der sich besonders für Quadriken eignet, da diese in sehr wenigen Schritten auf die für das Lösen benötigte K-Form gebracht werden können.
Für das Aufstellen des Ritz-Galerkin-Systems ist es notwendig, über den inneren Zellen und dem innerhalb des Gebietes D liegenden Teil der Randzellen zu integrieren. Hierfür wird auf Techniken der numerischen Integration zurückgegriffen, da meist keine analytische Lösung vorhanden oder diese sehr schwer zu berechnen ist. Damit über den Randzellen möglichst genau integriert werden kann, wird der im Inneren von D liegende Teil in Dreiecke beziehungsweise Tetraeder unterteilt, über welchen dann integriert werden kann. Diese Unterteilung in Dreiecke und Tetraeder wird in Kapitel 6 behandelt. Es werden die verschiedenen Fälle von Randzellenschnitten untersucht und einfache Verfahren zur Triangulierung und Tetraedisierung der Randzellen, die vom Autor dieser Arbeit entwickelt wurden, angegeben, bei deren Entwicklung auf die Besonderheiten der in dieser Arbeit verwendeten Methoden und Objekte eingegangen wurde.
Die numerische Integration über den inneren Zellen des Gitters über D ist mit einfachen Tensorprodukt-Gauß-Formeln realisiert. Eine allgemeine Einführung in die Gauß-Formeln sowie die Integration über den inneren Zellen wird zu Beginn von Kapitel 7 gegeben, bevor die Integration über Dreiecken und Tetraedern der Randzellen behandelt wird. Die Integration über Dreiecken und Tetraedern lässt sich mittels Tensorprodukt-Gauß-Formeln realisieren, wenn die Parameter entsprechend mit einer Transformation abgebildet worden sind. Allerdings gibt es auch spezielle Integrationsparameter für Dreiecke und Tetraeder, die den Vorteil haben, dass weniger Parameter für die gleiche Genauigkeit benötigt werden. Jedoch können diese dann im Gegensatz zu den Tensorprodukt-Formeln nicht auf beliebige Dimensionen verallgemeinert werden. Durch die Unterteilung in Dreiecke beziehungsweise Tetraeder erhält man eine stückweise lineare Approximation des Randes ∂D. Ist diese Approximation nicht genau genug, so besteht die Möglichkeit, die Dreiecke und Tetraeder auf krummlinige und krummflächige Elemente abzubilden, um den Rand genauer zu approximieren. Diese Abbildung ist eine so genannte isoparametrische Abbildung, und entsprechend werden die krummlinigen beziehungsweise krummflächigen Elemente isoparametrische Elemente genannt. Durch die Verwendung von Bézier-Techniken, die in dieser Art wohl noch nicht in der Finite Elemente Theorie verwendet wurden, kann hierbei

eine sehr gute Genauigkeit erreicht werden. Eine neue und vollständig andere Möglichkeit zur Berechnung der Integrale, die im Rahmen dieser Arbeit erarbeitet wurde, bildet den Abschluss dieses Kapitels. Die Berechnung erfolgt durch ein entsprechend trainiertes neuronales Netz. Es wird zunächst eine grobe Einführung in die Theorie und den Aufbau von neuronalen Netzen gegeben, ohne allerdings auf biologische Plausibilität Rücksicht zu nehmen. Anschließend wird ein Vorschlag für den Aufbau eines neuronalen Netzes gemacht.
In Kapitel 8 wird auf die Implementierung der in dieser Arbeit vorgestellten Methode in MATLAB eingegangen. Der Ablauf hierbei lässt sich grob in drei Teile unterteilen: die Zell- und B-Spline Klassifikation, das Aufstellen des Ritz-Galerkin-Systems und das Lösen des Ritz-Galerkin-Systems.
Den Abschluss des Kapitels bildet ein Abschnitt über Beispiele, die mit den im Rahmen dieser Arbeit erstellten Programmen aufgestellt und berechnet wurden. Es werden sowohl ein zweidimensionales als auch ein dreidimensionales Gebiet verwendet und es werden Besonderheiten der entsprechenden Programme im Dreidimensionalen gegenüber ihren Gegenstücken im Zweidimensionalen dargestellt.

1.2 Symbolverzeichnis

$\mathbb{N}, \mathbb{Z}, \mathbb{R}$	Die Mengen der natürlichen, ganzen und reellen Zahlen
$\alpha, \lvert\alpha\rvert$	Ein Multiindex und dessen Ordnung
∂^α	Der Differentialoperator
$\delta_{i,j}$	Das Kronecker-Symbol
supp f	Der Träger der Funktion f
$\overline{M}, M^\circ, M^c$	Für eine Menge M der Abschluss, das Innere und das Komplement
$D, \partial D$	Ein Gebiet im $\mathbb{R}^n$ und dessen Rand
D_P	Die Annäherung des Gebietes D durch ein Polygon
$C^k(D)$	Die Menge der auf dem Gebiet D k-mal stetig differenzierbaren Funktionen
$C_0^\infty(D)$	Die Menge aller Funktionen aus $C^\infty(D)$ mit kompaktem Träger in D
$L_2(D)$	Der Lebesque-Raum der quadratisch Lebesque-integrierbaren Funktionen
$H^\ell(D)$	Der Sobolev-Raum der Ordnung ℓ über dem Gebiet D
$H_0^\ell(D)$	Die Vervollständigung der Menge $C_0^\infty(D)$ bezüglich der Sobolev Norm $\lVert\cdot\rVert_{H^\ell(D)}$
$\asymp, \succeq, \preceq$	Gleichheit beziehungsweise Abschätzung bis auf eine positive Konstante
$<\cdot,\cdot>_H$	Das Skalarprodukt im Hilbertraum H
$<\cdot,\cdot>_0$	Das L_2-Skalarprodukt
$\lVert\cdot\rVert_H$	Die Norm im Hilbertraum H
$\lVert\cdot\rVert_\ell, \lvert\cdot\rvert_\ell$	Sobolev-Norm und Halbnorm der Ordnung ℓ
h	Diskretisierungsparameter beziehungsweise die Gitterweite
G	Die Ritz-Galerkin-Matrix
b^n,$b^n_{k,h}$	B-Spline und skalierter und verschobener B-Spline
$K^n_{D,h}$	Die Indexmenge der relevanten B-Splines
$I^n_{D,h}(j)$	Die Indexmenge der (dem äußeren Index j zugeordneten) inneren Splines
$J^n_{D,h}(i)$	Die Indexmenge der (dem inneren Index i zugeordneten) äußeren Splines
Q_i	Gitterzelle
w	Gewichtsfunktion
$e_{i,j}$	Erweiterungskoeffizient
$B^n_{k,h}$	WEB-Spline
$w\mathbb{B}$	Der Raum der gewichteten Splines über dem Gebiet D
$w^e\mathbb{B}$	Der Raum der WEB-Splines über dem Gebiet D

2 Finite Elemente

Zu Beginn dieser Arbeit wird in diesem Kapitel grundlegend auf die Theorie der Finiten Elemente zur numerischen Behandlung von elliptischen Randwertproblemen eingegangen. Um dieses Anfangskapitel kurz und übersichtlich zu gestalten, wird meist auf Beweise verzichtet. Weiterführende Informationen und Beweise sind in der angegebenen Literatur zu finden, beispielsweise in [3] und [30], woran sich der Aufbau dieses Kapitels orientiert.

2.1 Elliptische Randwertprobleme

Eine offene und zusammenhängende Menge $D \subset \mathbb{R}^n$ mit $n \geq 1$ nennt man Gebiet. Der Gebietsrand wird mit $\partial D := \overline{D} \setminus D^\circ$ bezeichnet. Für eine Funktion $u \in C^k(D)$ schreibt man die partielle Ableitung von u als

$$\partial^\alpha = \frac{\partial^{|\alpha|} u}{\partial x_1^{\alpha_1} \dots \partial x_n^{\alpha_n}},$$

wobei $\alpha = (\alpha_1, \dots, \alpha_n) \in \mathbb{N}_0^n$ ein Multiindex und $|\alpha| = \alpha_1 + \dots + \alpha_n$ ist.
Eine lineare Differentialgleichung zweiter Ordnung mit konstanten Koeffizienten $a_{i,j}, b_i \in \mathbb{R}$ lässt sich folgendermaßen schreiben:

$$-\sum_{i,j=1}^{n} a_{i,j} \frac{\partial^2 u}{\partial x_i \partial x_j} + \sum_{i=1}^{n} b_i \frac{\partial u}{\partial x_i} + cu = f, \tag{2.1}$$

mit einer beliebigen Funktion $f : D \to \mathbb{R}$. Es wird hier vorausgesetzt, dass $a_{i,j} = a_{j,i}$, d.h. die Koeffizientenmatrix $A := (a_{i,j})$ ist symmetrisch, was wegen der Vertauschbarkeit der partiellen Ableitungen leicht zu erreichen ist.

Definition 2.1 *Die Differentialgleichung (2.1) heißt elliptisch, wenn die Koeffizientenmatrix ausschließlich positive Eigenwerte besitzt, wenn also die Koeffizientenmatrix A positiv definit ist. Die elliptische Differentialgleichung wird dann durch*

$$Lu = f$$

dargestellt, mit einem entsprechend definierten Differentialoperator L.

Im Allgemeinen hat eine Differentialgleichung unendlich viele Lösungen, sofern sie überhaupt lösbar ist. Man benötigt wie bei gewöhnlichen Differentialgleichungen auch hier Nebenbedingungen an die Lösung von (2.1). Hier werden die beiden folgenden physikalisch relevanten Nebenbedingungen gewählt:

- Dirichlet-Randbedingung: Auf dem Rand ∂D von D gilt $u = g(x)$.

- Neumann-Randbedingung: Auf ∂D gilt $\frac{\partial u}{\partial n} = g(x)$, wobei $\frac{\partial u}{\partial n}$ die Ableitung in Normalenrichtung ist.

Ein wichtiges Beispiel, das hier als Modellproblem verwendet wird, ist die Poisson-Gleichung.

Definition 2.2 *Seien ein Gebiet $D \subset \mathbb{R}^n$ und eine beliebige Funktion $f : D \to \mathbb{R}$ gegeben. Das Poisson-Problem mit homogenen Dirichlet-Randbedingungen ist gegeben durch*

$$\begin{aligned} -\Delta u &= f \quad in\ D \\ u &= 0 \quad auf\ \partial D\,. \end{aligned} \tag{2.2}$$

Hier ist die Matrix A die Einheitsmatrix.

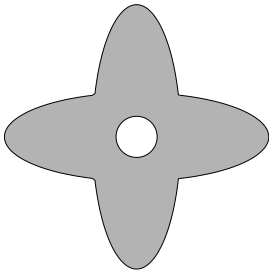


Abbildung 2.1: Beispiel für ein Gebiet D

Ein Beispiel für ein Gebiet D, das im weiteren Verlauf dieser Arbeit zum Darstellen der einzelnen Schritte verwendet wird, ist in Abbildung 2.1 zu sehen.

Definition 2.3 *Die Kombination aus Nebenbedingungen beziehungsweise Randbedingungen und der Differentialgleichung (2.1) nennt man elliptisches Randwertproblem. Eine Funktion $u \in C^2(D) \cap C^0(\overline{D})$, die das Modellproblem unter der Dirichlet-Randbedingung löst, nennt man klassische Lösung.*

Elliptische Randwertprobleme haben unter geeigneten Voraussetzungen eine klassische Lösung, deren Existenz allerdings meist nur schwer zu beweisen ist. Aus diesem Grund sucht man die Lösungen in allgemeineren, geeignet vervollständigten Funktionenräumen.

2.2 Sobolev-Räume

Als Grundlage für diesen Abschnitt dient folgende Definition des Lebesgue-Raumes $L_2(D)$:

Definition 2.4 *Sei $D \subset \mathbb{R}^n$, $(n > 1)$ im folgenden eine offene Teilmenge des $\mathbb{R}^n$. Der Lebesgue-Raum der über D quadratisch Lebesgue-integrierbaren Funktionen wird definiert durch*

$$L_2(D) = \{u : D \to \mathbb{R}\ \textit{meßbar} : |u|^2\ \textit{Lebesgue-integrierbar}\}.$$

Auf diesem Raum wird durch

$$\langle u, v \rangle_{L_2} := \int_D u(x)v(x)dx$$

ein Skalarprodukt und durch

$$\|u\|_{L_2} := \sqrt{\langle u, v\rangle_{L_2}}$$

die zugehörige Norm definiert.

Für zwei Funktionen $u, v \in L_2(D)$ gilt $u = v$ genau dann, wenn sich u und v nur auf einer Nullmenge, also einer Menge vom Maß 0 unterscheiden. Das bedeutet $u = v$ genau dann, wenn $\|u - v\|_{L_2} = 0$. $L_2(D)$ bildet, versehen mit dem Skalarprodukt $\langle u, v\rangle_{L_2}$, einen Hilbertraum, da er bezüglich der Norm $\|\cdot\|_{L_2}$ abgeschlossen ist [5].

Definition 2.5 *Der Träger einer Funktion $u : D \to \mathbb{R}$ wird definiert durch*

$$\text{supp } u := \overline{\{x \in D : u(x) \neq 0\}}.$$

Der Raum der Funktionen mit kompaktem Träger in D wird mit

$$C_0^\infty(D) = \{u \in C^\infty(D) : \text{supp } u \textit{ ist kompakt}\}$$

bezeichnet.

Der Raum $L_2(D)$ ist als Ansatzraum für das Modellproblem (2.2) ungeeignet, da an die Differenzierbarkeit der Funktionen weitere Anforderungen gestellt werden müssen. Die folgende Definition liefert eine Verallgemeinerung der klassischen Ableitung

$$\frac{\partial^{|\alpha|}}{\partial a_1^{\alpha_1} \dots \partial x_n^{\alpha_n}} \quad \text{der Ordnung} \quad |\alpha| := \sum_{k=1}^{n} \alpha_k$$

mit dem Multiindex

$$\alpha := (\alpha_1, \dots, \alpha_n) \in \mathbb{N}_0^n.$$

Definition 2.6 *Die Funktion $u \in L_2(D)$ besitzt die schwache Ableitung $\partial^\alpha u \in L_2(D)$, wenn für alle $v \in C_0^\infty(D)$ gilt:*

$$\langle \partial^\alpha u, v\rangle_{L_2} = (-1)^{|\alpha|} \langle u, \partial^\alpha v\rangle_{L_2}$$

mit dem Multiindex $\alpha := (\alpha_1, \dots, \alpha_n) \in \mathbb{N}_0^n$.

Für Funktionen $u \in L_2(D) \cap C^{|\alpha|}(D)$ stimmen die klassische und die schwache Ableitung überein. Dies lässt sich mittels partieller Integration zeigen [3].
Um die für das Modellproblem benötigten Funktionenräume zu erhalten, wird die folgende Definition aus [30] angegeben. Durch die Einbettung des verallgemeinerten Differentiationsbegriffs und die Modifikation der entsprechenden Normen erhält man die gewünschten Funktionenräume.

Definition 2.7 *Sei $D \subset \mathbb{R}^n$ ein Gebiet und $\ell \in \mathbb{N}_0$. Auf dem Sobolev-Raum*

$$H^\ell(D) := \{u \in L_2(D) : \forall \alpha \in \mathbb{N}_0^n \textit{ mit } |\alpha| \leq \ell \textit{ existiert } \partial^\alpha u\}$$

der Ordnung ℓ wird durch

$$\langle u, v\rangle_{H^\ell(D)} := \sum_{|\alpha| \leq \ell} \langle \partial^\alpha u, \partial^\alpha v\rangle_{L_2}$$

ein Skalarprodukt mit der induzierten Sobolev-Norm

$$\|u\|_{H^\ell(D)} := \sqrt{\langle u, v\rangle_{H^\ell(D)}} = \left(\sum_{|\alpha|\leq\ell} \|\partial^\alpha u\|_{L_2}^2\right)^{1/2} \tag{2.3}$$

definiert. Die Beschränkung auf schwache Ableitungen der Ordnung ℓ ergibt die Sobolev-Halbnorm

$$|u|_{H^\ell(D)} := \left(\sum_{|\alpha|=\ell} \|\partial^\alpha u\|_{L_2}^2\right)^{1/2}. \tag{2.4}$$

Hieraus folgt direkt: Für $\ell = 0$ ist $H^0(D) = L_2(D)$ genauso wie $\langle\cdot,\cdot\rangle_{H^\ell(D)} = \langle\cdot,\cdot\rangle_{L_2}$. Im weiteren Verlauf dieser Arbeit werden für $\ell \geq 0$ die Normen und Halbnormen $\|\cdot\|_{H^\ell(D)}$ mit $\|\cdot\|_\ell$ und $|\cdot|_{H^\ell(D)}$ mit $|\cdot|_\ell$ abgekürzt. Insbesondere wird dann $\|\cdot\|_{L_2}$ mit $\|\cdot\|_0$ bezeichnet und analog dazu das Skalarprodukt $\langle\cdot,\cdot\rangle_{L_2}$ mit $\langle\cdot,\cdot\rangle_0$.
Sobolev-Räume sind auch Hilbert-Räume (siehe beispielsweise [31]).
Die homogenen Dirichlet-Randbedingungen des Modellproblems (2.2) können anhand der folgenden Definition in die Sobolev-Räume integriert werden [30].

Definition 2.8 *Für ein Gebiet $D \subset \mathbb{R}^n$ bezeichne $H_0^\ell(D)$ die Vervollständigung von $C_0^\infty(D)$ bezüglich der Sobolev-Norm $\|\cdot\|_\ell$.*

$H_0^\ell(D)$ ist ein abgeschlossener Teilraum von $H^\ell(D)$ und somit wieder ein Hilbert-Raum. Nach [3] gilt für diese speziellen Sobolev-Räume die Äquivalenz der Normen und Halbnormen.

Satz 2.9 Poincaré-Friedrichssche-Ungleichung
Sei D in einem n-dimensionalen Würfel der Kantenlänge s enthalten. Dann ist

$$\|v\|_0 \leq s|v|_1 \qquad \forall v \in H_0^1(D). \tag{2.5}$$

Mit den in diesem Abschnitt dargestellten Eigenschaften der Sobolev-Räume lässt sich nun mit dem nächsten Abschnitt fortfahren.

2.3 Variationsformulierung

In diesem Abschnitt soll das Randwertproblem in ein Variationsproblem im Hilbert-Raum $H \subset L_2(D)$ überführt werden. H bezeichnet also einen Hilbert-Raum, $\langle\cdot,\cdot\rangle_H$ das entsprechende Skalarprodukt und $\|\cdot\|_H$ die Norm auf H.

Definition 2.10 *Die Menge aller stetigen linearen Funktionale*

$$l : H \to \mathbb{R}$$

wird als Dualraum H' bezeichnet mit der zugehörigen Operatornorm

$$\|l\|_{H'} := \sup_{u\in H\setminus\{0\}} \frac{|l(u)|}{\|u\|_H}.$$

Betrachtet man die Poisson-Gleichung aus dem Modellproblem (2.2), $-\Delta u = f$, und wendet das L_2-Skalarprodukt mit Funktionen $v \in H_0^1(D)$ auf beide Seiten an, so erhält man

$$\int_D -\operatorname{div}(\operatorname{grad} u) v \, dx = \int_D f v \, dx \qquad \forall v \in H_0^1(D).$$

Wegen der homogenen Dirichlet-Randbedingung und aus dem Satz von Green folgt

$$\int_D \operatorname{grad} u \cdot \operatorname{grad} v \, dx = \int_D f v \, dx \qquad \forall v \in H_0^1(D).$$

Definiert man nun die Bilinearform $a : H \times H \to \mathbb{R}$ durch

$$a(u, v) = \int_D \operatorname{grad} u \cdot \operatorname{grad} v \, dx$$

und das lineare Funktional $l : L_2(D) \to \mathbb{R}$ durch

$$l(v) = \int_D f v \, dx \,,$$

so lässt sich die Variationsformulierung schreiben als

$$a(u, v) = l(v) \qquad \forall v \in H. \tag{2.6}$$

Eine Lösung $u \in H_0^1(D)$ der Variationsformulierung bezeichnet man als schwache Lösung von (2.2).
Im folgenden Satz und der anschliessenden Definition werden die Grundlagen für die Existenz und Eindeutigkeit einer solchen Lösung gelegt (siehe [4]).

Satz 2.11 Darstellungssatz von Riesz
Jedes $u \in H$ definiert durch

$$l_u(v) := \langle u, v \rangle_H$$

ein stetiges lineares Funktional $l_u \in H'$. Umgekehrt existiert für jedes $l \in H'$ genau ein Element $u \in H$, so dass

$$l(v) = \langle u, v \rangle_H \qquad \forall v \in H$$

und darüber hinaus

$$\|l\|_{H'} = \|u\|_H$$

ist.

Es ist naheliegend, die Bilinearform aus dem Variationsproblem (2.6) zur Definition des in diesem Satz verwendeten Skalarprodukts $\langle u, v \rangle_H$ zu verwenden. Hierfür zunächst folgende Definition:

Definition 2.12 *Eine Bilinearform $a : H \times H \to \mathbb{R}$ auf einem Hilbert-Raum H heißt beschränkt (oder stetig), wenn eine Konstante $\alpha_b > 0$ existiert, so dass*

$$|a(v,w)| \leq \alpha_b \|v\|_H \|w\|_H \qquad \forall v, w \in H$$

und koerziv (oder elliptisch) auf $V \in H$, wenn eine Konstante $\alpha_k > 0$ existiert, so dass

$$a(v,v) \geq \alpha_k \|v\|_H^2 \qquad \forall v \in V.$$

Eine Überprüfung der Eigenschaften eines Skalarproduktes liefert, dass die Bilinearform alle gewünschten Eigenschaften eines Skalarproduktes auf $V \subset H$ besitzt. Ist die Bilinearform a symmetrisch, beschränkt und koerziv auf $V \subset H$, so erhält man sofort

$$\alpha_k \|v\|_H^2 \leq a(v,v) \leq \alpha_b \|v\|_H^2 \qquad \forall v \in V, \tag{2.7}$$

woraus direkt folgt, dass

$$a(v,v) \geq 0 \qquad \forall v \in V$$

und

$$a(v,v) = 0 \;\Leftrightarrow\; v = 0.$$

Lemma 2.13 *Sei V ein abgeschlossener Teilraum des Hilbert-Raums H, die Bilinearform $a : H \times H \to \mathbb{R}$ symmetrisch und beschränkt auf H sowie koerziv auf V. Dann ist V ein Hilbert-Raum mit dem Skalarprodukt $a(\cdot,\cdot)$ und der zugehörigen Energienorm $\|v\|_a := \sqrt{a(v,v)}$.*

Beweis: Es genügt zu zeigen, dass V bezüglich der Energienorm $\|\cdot\|_a$ abgeschlossen ist. Aus der in (2.7) verwendeten Normäquivalenz und der Abgeschlossenheit von V bezüglich $\|\cdot\|_H$ folgt dies direkt. □

Mit dem in Lemma 2.13 mittels der Bilinearform a angegebenen Skalarprodukt ist der Hilbert-Raum V der Raum, für den die Existenz- und Eindeutigkeitsaussage mit dem Satz von Riesz getroffen werden kann.

Satz 2.14 *Sei H ein Hilbert-Raum, V ein abgeschlossener Teilraum von H, $a : H \times H \to \mathbb{R}$ eine auf H symmetrische, beschränkte und auf V koerzive Bilinearform und $l \in V'$. Dann besitzt das Variationsproblem*

$$a(u,v) = l(v) \qquad \forall v \in V$$

eine eindeutige Lösung $u \in V$.

Für das Modellproblem (2.2) ist $V = H_0^1(D)$ und

$$a(u,v) = \int_D \operatorname{grad} u \cdot \operatorname{grad} v \; dx = \sum_{|\alpha|=1} \|\partial^\alpha u\|_{L_2}^2 = |u|_1^2. \tag{2.8}$$

Wegen der Äquivalenz der Normen und Halbnormen ist $a(u,v) = |u|_1^2 \asymp \|u\|_1^2$ mit einer Konstanten $c > 0$. Analog zum Beweis von Lemma 2.13 impliziert dies die Existenz und Eindeutigkeit einer schwachen Lösung $u \in H_0^1(D)$. Jede klassische Lösung von (2.2) ist auch

eine schwache Lösung. Umgekehrt ist eine schwache Lösung u genau dann eine klassische Lösung, wenn $u \in C^2(D) \cap C^0(\overline{D})$ [4].
Da im Allgemeinen die Symmetrie der Bilinearform a nicht vorausgesetzt werden kann, stellt sich die Frage nach den Bedingungen an a und l, um die Existenz und Eindeutigkeit von Lösungen von (2.6) zu erhalten. Die Antwort gibt das folgende Lemma von Lax-Milgram.

Lemma 2.15 Lemma von Lax-Milgram
Sei V ein Hilbert-Raum, $a : V \times V \to \mathbb{R}$ eine beschränkte koerzive Bilinearform, $l \in V'$. Dann besitzt das Variationsproblem

$$a(u, v) = l(v) \qquad \forall v \in V$$

eine eindeutige Lösung $u \in V$.

2.4 Das Ritz-Galerkin-Verfahren

Das Verfahren beruht darauf, das Variationsproblem näherungsweise zu lösen und somit eine Approximation der Lösung zu erhalten. Man geht dazu folgendermaßen vor: Es wird ein endlich dimensionaler Teilraum $V_h \subset V$ gewählt mit $V_h = \text{span}\{b_1, \ldots, b_{n_h}\}$. Bezüglich der hier gewählten Basis hat die Approximation der schwachen Lösung u_h die Darstellung

$$u_h = \sum_{k=1}^{n_h} c_k b_k \qquad \text{mit } c_k \in \mathbb{R}, \tag{2.9}$$

h bezeichnet den Diskretisierungsparameter. Für $h \to 0$ geht $n_h \to \infty$ und $u_h \in V_h$ konvergiert für entsprechendes V_h gegen die schwache Lösung $u \in V$.
u_h mit der Darstellung (2.9) ist also Lösung des Variationsproblems

$$a(u_h, v) = l(v) \qquad \forall v \in V_h.$$

a und l sind linear, deshalb genügt es, die Elemente b_ℓ der Basis anstelle von allen $v \in V_h$ zu betrachten. Somit ist die folgende Darstellung analog zum Variationsproblem

$$a(u_h, b_\ell) = l(b_\ell) \qquad \ell = 1, \ldots, n_h\,,$$

mit der in (2.9) angegebenen Darstellung für u_h erhält man

$$a\left(\sum_{k=1}^{n_h} c_k b_k, b_\ell\right) = \sum_{k=1}^{n_h} c_k a(b_k, b_\ell) = l(b_\ell) \qquad \text{mit} \qquad \ell \in \{1, \ldots, n_h\}\,. \tag{2.10}$$

Die Approximation u_h lässt sich durch Lösen des Gleichungssystems

$$\sum_{j=1}^{n_h} a(b_j, b_i) c_i = l(b_i)\,, \qquad i = 1, \ldots, n_h \tag{2.11}$$

bestimmen. Dies zeigt das folgende Lemma aus [30].

Lemma 2.16 *Der Koeffizientenvektor $C = (c_k), k \in \{1, \ldots, n_h\}$, der Basisdarstellung (2.9) der schwachen Lösung $u_h \in V_h$ ist Lösung des Ritz-Galerkin-Systems*

$$GC = F \quad mit \quad g_{k,\ell} := a(b_k, b_\ell) \quad und \quad f_k := l(b_k)\,. \tag{2.12}$$

Die Ritz-Galerkin-Matrix $G := (g_{k,\ell}), k, \ell \in \{1, \ldots, n_h\}$, ist positiv definit.

Beweis: Es ist nur die positive Definitheit von G, also $X'GX > 0$ für alle $X \in \mathbb{R}^{n_h} \setminus \{0\}$ zu zeigen:

$$\begin{aligned} X'GX &= \sum_{k=1}^{n_h}\sum_{\ell=1}^{n_h} g_{k,\ell}x_k x_\ell = \sum_{k=1}^{n_h}\sum_{\ell=1}^{n_h} a(b_k, b_\ell)x_k x_\ell \\ &= a\left(\sum_{k=1}^{n_h} x_k b_k\,, \sum_{\ell=1}^{n_h} x_\ell b_\ell\right) \succeq \|\sum_{k=1}^{n_h} x_k b_k\|_0^2 > 0. \end{aligned}$$

Bei der Abschätzung wurde dabei die Koerzivität von a und $X \neq 0 \Rightarrow u_h \neq 0$ verwendet. □

Eine gute Lösbarkeit des Galerkin-Systems ist wünschenswert. Dies wird durch die Verwendung von Basisfunktionen mit kompaktem und möglichst kleinem Träger erreicht, was eine schwachbesetzte Galerkin-Matrix zur Folge hat. Diese Träger oder oft auch die zugehörigen Funktionen werden Finite Elemente genannt und sind Gegenstand des nächsten Abschnittes.

Eine Aussage darüber, wie gut u_h die Lösung u approximiert, gibt das folgende Lemma von Céa:

Lemma 2.17 Lemma von Céa

Sei V_h ein endlicher, abgeschlossener Teilraum des Hilbert-Raumes V, $a : V \times V \to \mathbb{R}$ eine auf V beschränkte koerzive Bilinearform und $l \in V'$. Dann gilt für die schwachen Lösungen $u \in V$ und $u_h \in V_h$ die Abschätzung

$$\|u - u_h\|_V \leq \frac{\alpha_b}{\alpha_k} \min_{u_h \in V_h} \|u - v_h\|_V\,.$$

Dabei bezeichne α_b die Beschränktheitskonstante und α_k die Koerzivitätskonstante von a in V.

Beweis: Aufgrund der Voraussetzungen gilt

$$a(u, v) = l(v) \quad \forall v \in V \qquad \text{und} \qquad a(u_h, v) = l(v) \quad \forall v \in V_h\,.$$

Die Subtraktion dieser Gleichungen in V_h liefert $a(u - v_h, v) = 0$ für alle $v \in V_h$ und mit der Wahl $v = v_h - u_h \in V_h$ für $u_h \in V_h$ insbesondere $a(u - u_h, v_h - u_h) = 0$. Hiermit folgt unter Verwendung der Koerzivität und Beschränktheit von a

$$\begin{aligned} \alpha_k\|u - u_h\|_V^2 &\leq a(u - u_h, u - u_h) \\ &= a(u - u_h, u - V_h) + a(u - u_h, V_H - u_h) \\ &= a(u - u_h, u - V_h) \\ &\leq \alpha_b\|u - u_h\|_V\|u - v_h\|_V\,. \end{aligned}$$

Die Behauptung folgt nach Division durch $\alpha_k \|u - u_h\|_V$ aus der Abgeschlossenheit von V_h.
□
Eine Aussage für eine Abschätzung in einem Hilbert-Raum liefert das Lemma von Aubin-Nitsche. Für den Beweis sei auf die Literatur verwiesen, beispielsweise [3].

Lemma 2.18 Lemma von Aubin-Nitsche
Sei H ein Hilbert-Raum mit dem Skalarprodukt $\langle \cdot, \cdot \rangle_H$ und der zugehörigen Norm $\| \cdot \|_H$. V sei ein bezüglich der Norm $\| \cdot \|_V$ abgeschlossener Unterraum von H. Dann gilt für die schwache Lösung u_h im endlichen Teilraum V_h von V

$$\|u - u_h\|_H \preceq \|u - u_h\|_V \sup_{g \in H} \left(\frac{1}{\|g\|_H} \inf_{v_h \in V_h} \|\varphi_g - v_h\|_V \right) , \tag{2.13}$$

wenn jedem $g \in H$ die eindeutige schwache Lösung $\varphi_g \in V$ von

$$a(w, \varphi_g) = \langle g, w \rangle_H \qquad \forall w \in V$$

zugeordnet wird.

2.5 Finite Elemente

Man unterteilt D normalerweise in endlich viele Teilmengen, um die Ansatzfunktionen mit möglichst kleinem Träger zu konstruieren. Im $\mathbb{R}^2$ sind dies üblicherweise Dreiecke oder Rechtecke und im $\mathbb{R}^3$ Tetraeder oder Quader. In diesem Abschnitt wird zur Veranschaulichung das Modellproblem (2.2) auf einer Triangulierung des Gebietes D betrachtet, wobei D durch ein stückweise lineares Gebiet D_P angenähert wird. Im weiteren Verlauf dieser Arbeit wird noch genauer auf Triangulierungen eingegangen, weshalb hier nur die notwendigsten Forderungen an Triangulierungen dargestellt werden (siehe [3]).

Definition 2.19 *Eine Zerlegung $T = \{T_1, \ldots, T_N\}$ von D_P in Dreiecke heißt zulässig, wenn die Schnittmenge zweier Dreiecke T_i und T_j mit $i \neq j$ entweder leer, ein Eckpunkt oder eine Kante ist und $\bigcup_{k=1}^{N} T_k = D_P$ gilt.*
Man schreibt T_h statt T, wenn alle Dreiecke einen Durchmesser von höchstens $2h$ besitzen. Eine Familie solcher Triangulierungen $\{T_h\}$ heißt quasiuniform, wenn es eine Zahl $\kappa > 0$ gibt, so dass jedes Dreieck T von T_h einen Kreis enthält mit Radius r_T mit

$$r_T \geq \frac{h_T}{\kappa},$$

wobei h_T der halbe Durchmesser von T ist. Sie heißt uniform, wenn es eine Zahl κ gibt, so dass jedes Dreieck T von T_h einen Kreis mit dem Radius $r_T \geq \frac{h}{\kappa}$ enthält.

In Abbildung 2.2 ist ein Beispiel für ein Gebiet D und eine Triangulierung dessen zu sehen mitsamt der daraus resultierenden stückweisen Linearisierung D_P des Randes. Die in der Abbildung weiss markierten Knoten x_ℓ werden als innere Knoten bezeichnet, während die auf dem Rand ∂D_P liegenden schwarz markierten Punkte als äußere Knoten bezeichnet werden. Normalerweise ist das Gebiet D genau wie hier im Beispiel nicht als Polygon

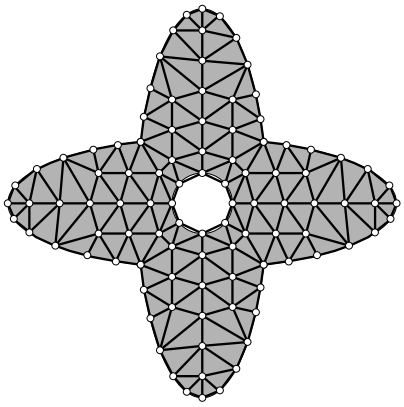


Abbildung 2.2: Triangulierung eines Gebietes D

gegeben. Deshalb wird in der Praxis durch oder vor der Triangulierung der Rand stückweise linear approximiert, was dazu führen kann, dass die Dirichlet-Randbedingungen nicht strikt eingehalten werden, da es vorkommen kann, dass ∂D_P auch Punkte außerhalb von D enthält.
Für jeden inneren Knoten x_i der Triangulierung wird eine lineare Basisfunktion b_k definiert, die an x_i den Wert 1 annimmt und an allen anderen Eckpunkten von Dreiecken, die x_i enthalten, den Wert 0. Die daraus gebildete Basis wird auch nodale Basis genannt (vom englischen node = Knoten), da sie durch die Werte an den Knoten festgelegt ist. Auf D_P bedeutet dies direkt die Einhaltung der Dirichlet-Randbedingungen.
Die Definition der Basisfunktionen führt direkt zu einer Schwachbesetztheit der Galerkin-Matrix, da

$$a(b_i, b_j) = 0\,, \quad \text{wenn} \quad \text{supp}\ (b_i) \cup \text{supp}\ (b_j) = \emptyset\,.$$

Einen Schnitt vom Maß $\neq 0$ haben nur Träger von Funktionen, deren Knoten benachbart sind.
Da die b_k auf jedem Dreieck linear sind, haben sie auf jedem Dreieck konstante partielle Ableitungen. Deshalb gilt

$$\begin{aligned} g_{k,\ell} &= \int_{D_P} \operatorname{grad} b_k \cdot \operatorname{grad} b_\ell \; dx \\ &= \sum_{T \in T_h} \int_T \operatorname{grad} b_k \cdot \operatorname{grad} b_\ell \; dx \\ &= \sum_{T \subset \text{supp}\ b_k \cap \text{supp}\ b_\ell} |T| \operatorname{const}(b_k, T) \operatorname{const}(b_\ell, T)\,, \end{aligned}$$

wobei $|T|$ die Fläche des Dreiecks T bezeichnet. Das bedeutet, dass sich die Einträge der Galerkin Matrix G einfach berechnen lassen. Da die b_k an den inneren Knoten entweder 0 oder 1 sind, also $b_k(x_\ell) = \delta_{k,\ell}$, folgt weiterhin, dass die Funktionswerte von u_h festgelegt sind durch die Koeffizienten der nodalen Basis c_ℓ, d.h. $u_h(x_k) = c_\ell$.

3 Spline-Funktionen

In diesem Kapitel soll eine Einführung in das Konzept der Splines gegeben werden. Zu Anfang werden die wichtigsten Eigenschaften und Verfahren zur Auswertung von B-Splines beschrieben. B-Splines bilden eine Basis des allgemeinen Spline-Raumes der stückweisen Polynome. Es werden hier nur gleichverteilte Knoten betrachtet, da die Vorteile beliebiger Knotenfolgen in mehreren Variablen nicht bestehen bleiben. Stattdessen können zur Verfeinerung hierarchische Basen verwendet werden [16].
Für den Aufbau des Kapitel wurde sich hauptsächlich an [16] orientiert.

3.1 Allgemeiner Spline Ansatz

Polynome liefern gute lokale Approximationsordnungen für glatte Funktionen, jedoch kann auf großen Intervallen die Genauigkeit gering sein. Des Weiteren zeigen numerische Beispiele, dass Poylnome für die Approximation von Funktionen mit Singularitäten in den Ableitungen oder unterschiedlichem lokalen Verhalten (zum Beispiel: $f(x) = \sqrt{x},\ x \in [0,1]$) nicht flexibel genug sind. Ausserdem haben lokale Veränderungen globalen Effekt. Deshalb ist es in der Praxis oft sinnvoll, auf das Prinzip der stückweisen Polynome, der Splines, zurückzugreifen, die auf einer Unterteilung des Parameterintervalls I definiert sind. Sind die Knotenpunkte äquidistant verteilt und gehen die Polynomstücke glatt ineinander über, so führt das zu Schoenbergs klassischer Definition [28].

Definition 3.1 *Ein Spline vom Grad $\leq n$ und Gitterweite h ist $(n-1)$-mal stetig differenzierbar und stimmt mit einem Polynom vom Grad $\leq n$ auf jedem Gitterintervall $[i, i+1]\,h$ des Parameterintervalls D überein.*

Grob gesprochen bedeutet dies, dass ein Spline aus aufeinanderfolgenden Polynomstücken vom Grad n besteht, die an den „Nahtstellen“ $(n-1)$-mal stetig differenzierbar sind.
In Abbildung 3.1 ist ein kubischer Spline, der aus fünf Polynomstücken besteht, und seine Ableitungen zu sehen. Der Spline s ist nach Definition zweimal stetig differenzierbar. Die Unstetigkeitsstellen der zweiten Ableitungen an den markierten Knotenpunkten sind kaum zu sehen. Ebenso scheint die erste Ableitung s', die aus quadratischen Polynomstücken besteht, glatt zu sein. Wenn man aber weiss, dass sie aus Parabelstücken besteht, erkennt man die abrupten Änderungen in den Kurvenverläufen.
Die Definition ist allerdings nicht sehr geeignet für Berechnungen, da sie die freien Parameter nicht hervorhebt. Ohne weiteres ist auch nicht klar, ob die Glattheitsbedingungen an den Knotenpunkten die Approximationsordnung der Polynome beeinträchtigen. Deshalb ist es sinnvoll, eine lokale Basis zu konstruieren, um die Theorie und die numerische Behandlung der Splines anzugehen. Passende Basisfunktionen, die so genannten B-Splines, werden im nächsten Abschnitt definiert.

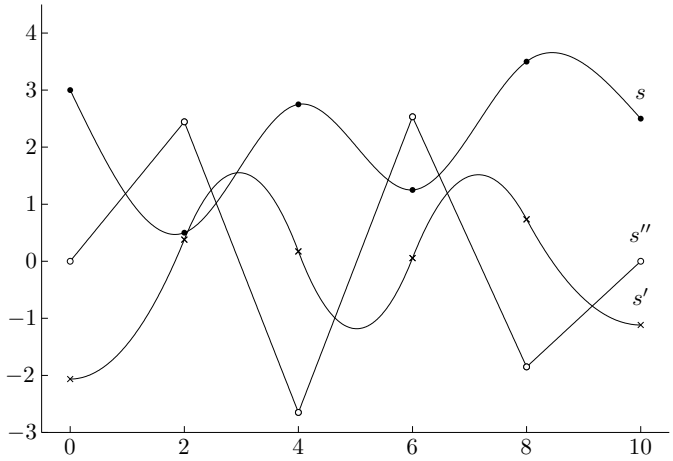


Abbildung 3.1: Ein kubischer Spline s und seine Ableitungen

3.2 B-Splines

Zur Veranschaulichung der Konstruktion von B-Splines betrachtet man zunächst den grundlegenden Fall der stückweise linearen B-Splines. Der stückweise lineare Spline ist von den Werten an den Knotenpunkten $x_i = ih$ eindeutig bestimmt. Also kann er als Linearkombination von Hut-Funktionen b_i eindeutig dargestellt werden, die gleich 1 an x_{i+1} sind und an allen anderen Knotenpunkten verschwinden:

$$p = \sum_i c_i b_i \,, \qquad c_i = p(x_{i+1}) \,. \tag{3.1}$$

Die Summationsweite hängt vom verwendeten Parameterintervall D ab, es muss über alle Hut-Funktionen summiert werden, die einen Träger in D haben. Ein Beispiel für einen stückweise linearen Spline mit den zugehörigen Hut-Funktionen ist in Abbildung 3.2 zu sehen.

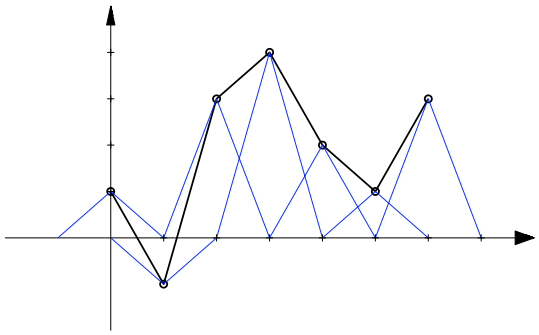


Abbildung 3.2: Stückweise linearer Spline als Linearkombination von Hut-Funktionen

Die Hut-Basisfunktionen sind alle skalierte Translate einer einzigen Funktion

$$b_i(x) = b^1(x/h - i) \,,$$

wobei die Standard Hut-Funktion Träger $[0, 2]$ und Gitterweite 1 hat und bei $x = 1$ den Wert 1 annimmt.
B-Splines können auf verschiedene Weise definiert werden. Hier wird die folgende Definition bevorzugt, da so die grundlegenden Eigenschaften von B-Splines direkt ersichtlich bzw offensichtlich sind:

Definition 3.2 *Der uniforme B-Spline b^n vom Grad n ist definiert durch die Rekursion*

$$b^n(x) = \int_{x-1}^{x} b^{n-1}\,,$$

ausgehend von der charakteristischen Funktion b^0 auf dem Einheitsintervall $[0, 1)$. Äquivalent ist

$$\frac{d}{dx} b^n(x) = b^{n-1}(x) - b^{n-1}(x-1)$$

mit $b^n(0) = 0$. Sei $h > 0$ und $k \in \mathbb{Z}$, dann sind

$$b^n_{k,h}(x) = b^n(x/h - k)$$

die B-Splines vom Grad n zum Knoten kh im Gitter mit der Gitterweite h.

Ausgehend von der charakteristischen Funktion führt der erste Schritt zu den stückweise linearen Hut-Funktionen und mit dem nächsten Schritt erhält man den quadratischen B-Spline b^2. Beispielsweise ist für $x \in [1, 2]$:

$$b^2(x) = \int_{x-1}^{x} b^1 = \int_{x-1}^{1} t\,dt + \int_{1}^{x} 2 - t\,dt = -x^2 + 3x - 3/2\,.$$

Jeder Rekursionsschritt erhöht die Länge des Trägers, die Glattheit und den Grad um 1. In Abbildung 3.3 sind die B-Splines bis zur Ordnung 3 dargestellt. Den Funktionswert $b^3(x_0)$ erhält man nach Definition als Flächeninhalt des unter b^2 gekennzeichneten Bereichs mit dem Einheitsintervall $[x_0 - 1, x_0]$. Im Inneren des Trägers eines Splines vom Grad n müssen mindestens n Knoten liegen. Die B-Splines b^n haben somit minimalen Träger.

3.2.1 Eigenschaften von B-Splines

Die wichtigsten Eigenschaften der B-Splines werden im Folgenden dargestellt:
Die Positivität und der lokale Träger folgen direkt aus Definition 3.2, der B-Spline b^n ist positiv auf dem Intervall $(0, n+1)$ und verschwindet ausserhalb. Des Weiteren wird analog die Glattheit gezeigt, b^n ist auf dem Intervall $(0, n+1)$ $(n-1)$-mal stetig differenzierbar mit Unstetigkeiten der n-ten Ableitung an den Knotenpunkten $0, \ldots, n+1$. Es gilt für die Ableitungen des B-Splines die Differentiationsregel

$$\frac{d}{dx} b^n_{k,h}(x) = \frac{1}{h}\left(b^{n-1}_{k,h}(x) - b^{n-1}_{k+1,h}(x)\right)\,. \tag{3.2}$$

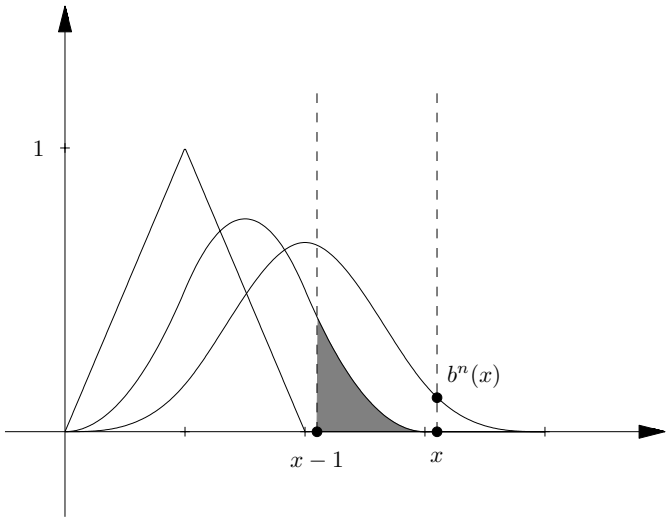


Abbildung 3.3: B-Splines

Die Ableitung eines B-Splines vom Grad n ist also eine Linearkombination aus B-Splines vom Grad $n-1$.
b^n ist ein Polynom vom Grad n auf jedem Intervall $[k, k+1]\,,\ k = 0, \ldots, n$, d.h. b^n ist auf $(0, n+1)$ aus Polynomstücken zusammengesetzt.
Als letztes werden noch die beiden folgenden qualitativen Eigenschaften angegeben:

Definition 3.3 *Der B-Spline vom Grad n ist symmetrisch, d.h.*

$$b^n(x) = b^n(n+1-x)$$

und streng monoton auf $[0, (n+1)/2]$ *und* $[(n+t)/2, n+1]$.

Diese Eigenschaften können einfach durch Induktion nach dem Grad bewiesen werden. Angenommen, dass beide Behauptungen bis zum Grad $n-1$ wahr sind, folgt aus Definition 3.2, dass

$$b^n(n+1-x) = \int_{n-x}^{n+1-x} b^{n-1}(t)dt = \int_{x-1}^{x} b^{n-1}(n-s)ds = \int_{x-1}^{x} b^{n-1}(s)ds\,,$$

woraus sich die Symmetrie ergibt. Um die Monotonie auf dem linken Intervall zu zeigen, bemerkt man an, dass die Ableitung von b^n,

$$b^{n-1}(x) - b^{n-1}(x-1),$$

nach Induktion positiv ist für $x \in \left(0, \frac{n}{2}\right]$. Dies gilt ebenso für $\frac{n}{2} < x < \frac{n+1}{2}$, da in diesem Fall

$$b^{n-1}(x) = b^{n-1}(n-x) > b^{n-1}(\frac{n-1}{2}) > b^{n-1}(x-1)$$

gilt.
Mit dem folgenden Satz wird ein wichtiges Hilfsmittel bereitgestellt, das zeigt, dass Monome durch B-Splines dargestellt werden können.

Satz 3.4 Marsden-Identität
Für $x, t \in \mathbb{R}$ und $n \in \mathbb{N}$ gilt

$$(x-t)^{n-1} = \sum_{k\in\mathbb{Z}} \psi^n_{k,h}(t) b^n_{k,h}(x) \tag{3.3}$$

mit den Koeffizientenfunktionen $\psi^1_{k,h} := 1$ und für $n > 1$

$$\psi^n_{k,h}(t) := \prod_{\ell=1}^{n-1} ((k+\ell)h - t) \ . \tag{3.4}$$

Mit der Marsden-Identität lässt sich auch zeigen, dass die B-Splines eine Partition der Eins bilden:
Die B-Splines bilden eine Partition der Eins, d.h. es gilt

$$\sum_{k\in\mathbb{Z}} b^n_{k,h}(x) = 1 \qquad \forall x \in \mathbb{R} \, .$$

Darüber hinaus folgt unmittelbar auch die lineare Unabhängigkeit der translatierten B-Splines.
Des Weiteren sind die B-Splines $b^n_{k,h}$ mit $h > 0$ und der Knotenfolge $h\mathbb{Z}$ linear unabhängig. Durch die lineare Unabhängigkeit und dadurch, dass auf jedem Intervall $h[\ell, \ell+1), \ell \in \mathbb{Z}$ nur $n+1$ der B-Splines von Null verschieden sind, ist nach Definition der B-Splines gezeigt, dass sie eine Basis der Splines vom Grad $\leq n$ bilden. Alle Polynome vom Grad $\leq n$ auf dem Intervall D können durch Linearkombinationen der $n+1$ B-Splines dargestellt werden, die auf D nicht verschwinden.

3.2.2 B-Spline Rekursion

Die Haupteigenschaften von B-Splines lassen sich aus Definition 3.2 recht einfach ableiten. Allerdings ist sie nicht wirklich gut für Berechnungen geeignet. Ein einfacher Algorithmus, um B-Splines auszuwerten, wird durch die folgende Rekursionsformel bereitgestellt, die von de Boor [7] und Cox [6] bewiesen wurde.

Satz 3.5 *Der B-Spline b^n ist eine gewichtete Kombination von B-Splines vom Grad $n-1$:*

$$b^n(x) = \frac{x}{n} b^{n-1}(x) + \frac{n+1-x}{n} b^{n-1}(x-1) \, .$$

Diese Rekursionsformel wird per Induktion bewiesen, indem man die Äquivalenz zur Formel der Ableitungen in Definition 3.2 zeigt. Da beide Seiten der Gleichung bei $x = 0$ verschwinden, genügt es zu zeigen, dass die Ableitungen übereinstimmen, d.h. dass

$$b^{n-1}_0 - b^{n-1}_1 = \frac{1}{n}(b^{n-1}_0 - b^{n-1}_1) + \{\frac{x}{n}(b^{n-2}_0 - b^{n-2}_1) + \frac{n+1-x}{n}(b^{n-2}_1 - b^{n-2}_1)\} \, , \tag{3.5}$$

wobei die Abkürzung $b^n_k = b^n(x-k)$ verwendet wird. Der Ausdruck in geschweiften Klammern kann in der Form

$$\frac{n-1}{n}\left(\left[\frac{x}{n-1} b^{n-2}_0 + \frac{n-x}{n-1} b^{n-2}_1\right] - \left[\frac{x-1}{n-1} b^{n-2}_1 + \frac{n-(x-1)}{n-1} b^{n-2}_2\right]\right)$$

geschrieben werden. Nimmt man induktiv an, dass die Rekursion für den Grad $n-1$ stimmt, dann sind beide Seiten der Gleichung gleich $b^{n-1}(x)$ beziehungsweise $b^{n-1}(x-1)$. Also ist

$$\{\dots\} = \frac{n-1}{n}\left(b_0^{n-1} - b_1^{n-1}\right)$$

und beide Seiten stimmen überein.
Mit Hilfe dieser Rekursion kann man auch die Polynomstücke der B-Splines berechnen. Dies muss für jedes Gitterintervall einzeln getan werden, und es ist günstig, die Taylor-Entwicklung zum linken Knotenpunkt zu nehmen.

Satz 3.6 *Die $n+1$ Polynomsegmente*

$$a_{k,0}^n + a_{k,1}^n t + \dots + a_{k,n}^n t^n\,, \qquad t = x - k \in [0,1]\,,$$

des B-Splines b^n können mit der Rekursion

$$a_{k,\ell}^n = \frac{k}{n} a_{k,\ell}^{n-1} + \frac{1}{n} a_{k,\ell-1}^{n-1} + \frac{n+1-k}{n} a_{k-1,\ell}^{n-1} - \frac{1}{n} a_{k-1,\ell-1}^{n-1}$$

berechnet werden, angefangen mit $a_{0,0}^0 = 1$ und $a_{k,\ell}^n = 0$ gesetzt, wenn entweder k oder $\ell \notin \{0,\dots,n\}$.

3.3 Splines

Mit Hilfe der vorgestellten B-Splines lassen sich Spline-Funktionen definieren, deren wichtigste Eigenschaften in diesem Abschnitt dargestellt werden.

Definition 3.7 *In einem offenen Intervall $I \subseteq \mathbb{R}$ und mit der Indexmenge*

$$K_{I,h}^n := \{k \in \mathbb{Z} :\ \text{supp}\ b_{k,h}^n \cap I \neq \emptyset\} \tag{3.6}$$

aller Indizes der B-Splines $b_{k,h}^n$, die auf I nicht verschwinden, wird die Spline-Funktion oder auch der Spline über dem Intervall I

$$s(x) := \sum_{k \in K_{I,h}^n} c_k b_{k,h}^n(x) \tag{3.7}$$

mit Koeffizienten $c_k \in \mathbb{R}$, als Linearkombination von B-Splines definiert.

Aus der Positivität der B-Splines folgt für positive c_k mit $k \in K_{I,h}^n$ die Positivität des Splines s. Wie bereits am Anfang des Kapitels erwähnt wurde, ist der Spline s auf jedem Intervall

$$I_\ell := h[\ell, \ell+1) \cap I, \quad \ell \in \mathbb{Z},$$

ein Polynom vom Grad $\leq n$ und an den Knoten mindestens $(n-1)$-mal stetig differenzierbar. Aufgrund der Differentiationsregel für B-Splines (3.2) gilt

$$\begin{aligned} s'(x) = \frac{d}{dx} \sum_{k \in K_{I,h}^n} &= \sum_{k \in K_{I,h}^n} \frac{c_k}{n}\left(b_{k,h}^{n-1}(x) - b_{k+1,h}^{n-1}(x)\right) \\ &= \sum_{k \in K_{I,h}^{n-1}} \frac{c_k - c_{k-1}}{h} b_{k,h}^{n-1}(x)\,. \end{aligned}$$

Durch die kompakten Träger der B-Splines haben Änderungen an den Koeffizienten nur lokalen Einfluss. Ausgehend von der Rekursionsdarstellung der B-Splines und den c_k lässt sich folgendes rekursives Verfahren zur Auswertung von Splines angeben.

Lemma 3.8 *Ausgehend von $c_k^n := c_k$ für $k \in K_{I_\ell,h}^n = \{\ell - n + 1, \ldots, \ell\}$ lässt sich zur Bestimmung des Funktionswertes $c_\ell^1 := s(x_0)$ der Spline-Funktion im Punkt $x_0 := h(\ell+t) \in I_\ell$ mit $t \in [0,1)$ die folgende Rekursion aufstellen*

$$c_k^{n-1} := w_k^n c_k^n + (1 - w_k^n) c_{k-1}^n \quad \text{mit } w_k^n := \frac{\ell + t - k}{n-1}\,. \tag{3.8}$$

3.4 Multivariate Splines

Es gibt mehrere Möglichkeiten univariate B-Splines in multivariate zu verallgemeinern. Die einfachste Methode ist jedoch durch Produkte univariater B-Splines. Die entstehenden multivariaten Tensorprodukt-B-Splines werden in der folgenden Definition eingeführt:

3.4.1 Multivariate B-Splines

Definition 3.9 *Sei $h > 0$, $m \in \mathbb{N}$, $x \in \mathbb{R}^m$, $n \in \mathbb{N}^m$ und $k \in \mathbb{Z}^m$. Der m-variate B-Spline $b_{k,h}^n : \mathbb{R}^m \to \mathbb{R}$ mit Grad n_ℓ in der ℓ-ten Variable mit Index k und Gitterweite h zum Knoten kh ist definiert durch*

$$b_{k,h}^n(x) = \prod_{\ell=1}^{m} b_{k_\ell,h}^{n_\ell}(x_\ell), \tag{3.9}$$

wobei $n \in \mathbb{N}$ geschrieben wird, wenn $n_1 = n_2 = \cdots = n_m$, wovon hier ausgegangen wird, sofern nicht explizit anders angegeben.

Analog zu den univariaten B-Splines, die als Spezialfall $m = 1$ aus der obigen Definition auftreten, lassen sich die Eigenschaften von m-variaten B-Splines zeigen und die univariaten Algorithmen verallgemeinern.
Die Positivität ergibt sich ebenso wie der lokale Träger direkt aus Definition 3.9, $b_{k,h}^n$ ist positiv auf dem Intervall $kh + (0, n+1)^m h$ und verschwindet ausserhalb.
$b_{k,h}^n$ ist bezüglich jeder Variablen $(n-1)$-mal stetig differenzierbar und auf jeder Gitterzelle $I = \ell h + [0,1]^m h$ ein m-variates Polynom vom Grad n.
Die partielle Ableitung $\partial^\alpha b_{k,h}^n$ ist im Fall $m = 2$ die Differenz zweier B-Splines vom Grad $(n_1 - 1, n_2)$. Allgemeiner ist die folgende Definition:

Definition 3.10 *Die ersten partiellen Ableitungen des m-variaten B-Splines $b_{k,h}^{(n_1,\ldots,n_m)}$ sind Differenzen von B-Splines mit kleinerem Grad, d.h. es gilt*

$$\partial^\alpha b_{k,h}^n(x) = \frac{1}{h}\left(b_{k,h}^{n-\alpha}(x) - b_{k+\alpha,h}^{n-\alpha}(x)\right)\,,$$

wobei α jeweils ein Einheitsvektor des $\mathbb{R}^m$ ist.

Höhere Ableitungen können einfach durch Iteration der partiellen Ableitung berechnet werden. Ein Beispiel hierfür wird im Folgenden gegeben [16]:

Beispiel 3.11 *Als Beispiel für die Iteration der partiellen Ableitungen wird der Laplace-Operator* $\Delta = \sum_{\nu} \partial_\nu^2$ *auf einen multivariaten B-Spline angewandt. Differenziert man die ersten Ableitungen aus der obigen Definition ein weiteres Mal, so erhält man*

$$\partial^\alpha \partial^\alpha b_{k,h}^n = \frac{1}{h^2} \left(b_{k,h}^{n-2\alpha} - 2b_{k+\alpha,h}^{n-2\alpha} + b_{k+2\alpha,h}^{n-2\alpha} \right) .$$

$\Delta b_{k,h}^n$ *ist dann die Summe dieser Ausdrücke über alle Einheitsvektoren* α *des* $\mathbb{R}^m$.

Anhand der multivariaten Marsden-Identität im folgenden Satz folgt die lineare Unabhängigkeit der B-Splines und dass sie eine Partition der Eins bilden.

Satz 3.12 Multivariate Marsden-Identität
Für $x, t \in \mathbb{R}^m$, $n \in \mathbb{N}^m$ *und den wie für die univariate Marsden-Identität in Satz 3.4 definierten Koeffizienten-Funktionen* $\psi_{k_\nu,h}^{n_\nu}$ *folgt die Darstellung*

$$\prod_{\nu=1}^{m} (x_\nu - t_\nu)^{n_\nu} = \sum_{k \in \mathbb{Z}^m} \left(\prod_{\nu=1}^{m} \psi_{k_\nu,h}^{n_\nu}(t_\nu) \right) b_{k,h}^n(x) . \tag{3.10}$$

Auf jeder Gitterzelle, die einen nichtleeren Schnitt mit dem Gebiet D besitzt, verschwinden alle bis auf $(n+1)^m$ B-Splines. Da diese nicht verschwindenden B-Splines alle Polynome vom Grad $\leq n$ aufspannen, müssen sie linear unabhängig sein, d.h. sie bilden eine Basis des m-variaten Splineraums.

3.4.2 Multivariate Spline-Funktionen

Analog zu Definition 3.7 werden nun multivariate Splines als Linearkombinationen von multivariaten B-Splines definiert.

Definition 3.13 *In einem Gebiet* $D \subseteq \mathbb{R}^m$ *und mit der Indexmenge*

$$K_{D,h}^n := \{k \in \mathbb{Z}^m : \text{supp } b_{k,h}^n \cap D \neq \emptyset\} \tag{3.11}$$

aller Indizes der m-variaten B-Splines $b_{k,h}^n$, *die auf* D *nicht verschwinden, wird die m-variate Spline-Funktion oder der m-variate Spline über dem Gebiet* D

$$s(x) := \sum_{k \in K_{D,h}^n} c_k b_{k,h}^n(x) \tag{3.12}$$

mit Koeffizienten $c_k \in \mathbb{R}$, *als Linearkombination von B-Splines definiert.*

Bei dieser Definition ist zu beachten, dass es, anders als im Univariaten, Splines geben kann, die nur einen sehr kleinen Teil ihres Trägers in D haben. Weiterhin kann die Indexmenge $K_{D,h}^n$, abhängig von der Form des Gebietes D, sehr unregelmässig sein. Zur Vereinfachung bei Berechnungen kann hier eine rechteckige Indexmenge K gewählt werden, die $K_{D,h}^n$ vollständig enthält.
Ebenfalls analog zum univariaten Fall werden multivariate Splines durch iteratives Ausführen des univariaten Verfahrens in Lemma 3.8 ausgewertet.

4 WEB-Splines

Auf den ersten Blick erscheint es nicht günstig, B-Splines als Finite Elemente Basisfunktionen zu benutzen, da das gleichmässige Gitter nicht dem Rand entspricht. Jedoch ist dieses Problem leicht gelöst, und man muss nicht auf die Vorteile verzichten, die ein regelmäßiges Gitter bietet. Um homogenen Randbedingungen zu genügen, multipliziert man mit einer positiven Gewichtsfunktion w, die auf dem Rand ∂D verschwindet, d.h. der von den relevanten B-Splines aufgespannte Raum wird an die Randbedingungen angepasst [18]. Beispielsweise kann man für ein glattes Gebiet eine Gewichtsfunktion wählen, die äquivalent zur Abstandsfunktion ist, d.h.

$$w(x) \asymp \operatorname{dist}(x, \partial D)\,.$$

Es gibt eine Vielzahl von anderen Möglichkeiten, besonders für Gebiete, die mit einfachen Grundobjekten beschrieben werden können (beispielsweise Geraden, Ebenen, Kreise, Zylinder, etc.). Ein systematischer Ansatz wird im nächsten Abschnitt beschrieben.

4.1 Gewichtung

Definition 4.1 *Der Raum der gewichteten B-Splines ist definiert durch*

$$w\mathbb{B}_h^n(D) = \operatorname{span}_{k\in K} wb_k\,,$$

wobei $\mathbb{B}_h^n$ der Raum der B-Splines vom Grad n zur Gitterweite h ist und K die Indexmenge der relevanten B-Splines.

In der folgenden Definition wird der Begriff der Gewichtsfunktion mit den wichtigsten Eigenschaften dargestellt:

Definition 4.2 *Sei $D \subset \mathbb{R}^m$ ein Gebiet mit glattem Rand und $d : \overline{D} \to \mathbb{R}_0^+$ definiert durch*

$$d(x, \partial D) := \inf_{y\in\partial D} \|x - y\| \tag{4.1}$$

die Abstandsfunktion zum Gebietsrand ∂D. Für $\gamma \in \mathbb{N}_0$ wird eine Funktion $w : \overline{D} \to \mathbb{R}_0^+$, mit

$$w(x) \asymp d(x, \partial D)^\gamma \qquad \text{mit } x \in D\,, \tag{4.2}$$

als Gewichtsfunktion der Ordnung γ bezeichnet. Ist w glatt und verschwindet linear auf ∂D ($\gamma = 1$), so wird w eine Standard-Gewichtsfunktion genannt.

Der Fall $\gamma = 0$ entspricht der trivialen Gewichtsfunktion $w = 1$ und ist hauptsächlich enthalten, um Fallunterscheidungen zu vermeiden. Man kann beispielsweise den Raum $\mathbb{B}$ als Spezialfall des gewichteten Raumes $w\mathbb{B}$ bezeichnen.
Die Positivität der Gewichtsfunktion ist sehr wichtig, denn falls Punkte $x \in D^\circ$ existieren mit $w(x) = 0$, verschwinden alle Funktionen des gewichteten Raums $w\mathbb{B}$ in diesen Punkten, was die Approximationsordnung zerstört. Ebenso wichtig ist es für das Einhalten der Randbedingungen, dass die Gewichtsfunktion mit minimaler Ordnung auf dem Rand verschwindet, da mit dem Raum $w\mathbb{B}$ nur Funktionen modelliert werden können, die mindestens mit der gleichen Ordnung verschwinden. Deshalb werden normalerweise keine Gewichtsfunktionen wie $w(x) = x_1x_2$ oder $w(x) = x_1x_2x_3$ verwendet, da sie Nullstellen höherer Ordnung an Ecken oder Kanten haben. Es ist jedoch möglich, solche Gewichtsfunktionen zu benutzen, sofern man spezielle Erweiterungen verwendet, die das Verhalten der Lösungen an Singularitäten berücksichtigen.
Die Mehrheit der Randbedingungen verlangt, dass w in erster Ordnung auf einem Teil des Gebietsrandes verschwindet, und kann deshalb mit Gewichtsfunktionen erster Ordnung modelliert werden, d.h.

$$w(x) \asymp \operatorname{dist}(x, \partial D)\,.$$

Die Abstandsfunktion selbst kann verwendet werden, wenn sie angemessen in der Nähe ihrer Singularitäten angepasst wird.
Zwei Beispiele für Gewichtsfunktionen für einfache zweidimensionale Gebiete sind in Abbildung 4.1 zu sehen.

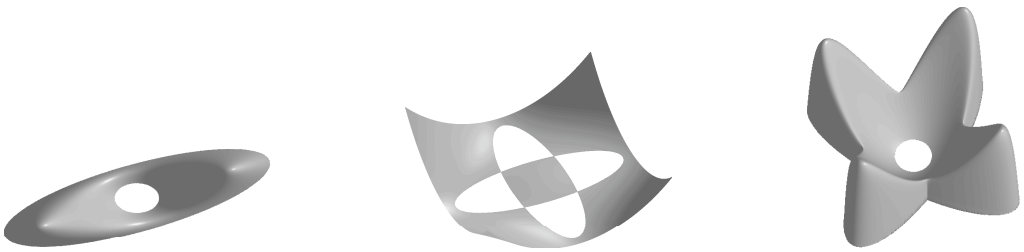


Abbildung 4.1: Links: Gewichtsfunktion durch Multiplikation einer Ellipse mit dem Äußeren eines Kreises. Mitte: Gewichtsfunktionen zweier Ellipsen multipliziert. Rechts: Gewichtsfunktion mittels R-Funktionen für alle drei Objekte.

In Abbildung 4.1 ist links zu sehen, dass es durchaus funktionieren kann, durch Multiplikation der Gewichtsfunktionen einfacher Gebiete, hier einer Ellipse und dem Äußeren eines Kreises, eine Gewichtsfunktion für das kombinierte Gebiet zu erhalten. Dies ist aber nicht immer so, wie man an der mittleren Abbildung sehen kann. Das Gebiet hier ist die Vereinigung zweier Ellipsen, allerdings ist das Produkt der Gewichtsfunktionen der beiden Ellipsen auf keinen Fall mehr eine Gewichtsfunktion für das resultierende Gebiet. Da beide Gewichtsfunktionen ausserhalb der jeweilige Ellipse negativ sind, wird das Produkt hier positiv. Im Inneren des Gebietes ist die Gewichtsfunktion nur in der Schnittmenge der beiden Ellipsen positiv. Man sieht also schon an einem solchen einfachen Beispiel, dass allein durch Multiplikation nur selten Gewichtsfunktionen einfacher Gebiete kombinierbar

sind. Eine Gewichtsfunktion für die Kombination aller drei Gebiete ist rechts in der Abbildung zu sehen. Diese wurde mit Hilfe des R-Funktionen Kalküls erstellt, was im nächsten Abschnitt beschrieben wird.

4.2 R-Funktionen

Viele einfache Gebiete liefern ad-hoc Definitionen von Gewichtsfunktionen w, wie bereits kurz an den Beispielen im vorherigen Abschnitt gesehen. Jedoch ist es nicht immer möglich, Funktionen impliziter Gleichungen von Randkurven zu multiplizieren, um Gewichtsfunktionen zu bekommen. Glücklicherweise existiert ein auf Rvachevs Konzept der R-Funktionen basierender systematischer Ansatz [23], [29]. Hier wird ein Spezialfall dieser recht allgemeinen Theorie beschrieben, der angemessen ist für Illustrationszwecke.

Definition 4.3 *Eine Funktion $r : \mathbb{R}^\ell \to \mathbb{R}$ ist eine R-Funktion, wenn ihr Vorzeichen nur vom Vorzeichen ihres Argumentes abhängt.*

R-Funktionen sind nah verwandt mit Booleschen Funktionen. Tatsächlich existieren für jede Boolesche Mengenoperation $\circ$ zugehörige R-Funktionen $r_\circ$, die die entsprechende Operation auf Gewichtsfunktionen definieren. Mit anderen Worten, wenn w_ν eine Gewichtsfunktion für ein Gebiet D_ν ist, dann ist

$$(w_1 \circ w_2)(x) := r_\circ(w_1(x), w_2(x))$$

eine Gewichtsfunktion für $D_1 \circ D_2$. Eine beliebte Wahl von zugehörigen R-Funktionen wird in der folgenden Tabelle gezeigt.

Mengenoperation	*zugehörige R-Funktion*
$D_1 \cap D_2$	$r_\cap(x) = x_1 + x_2 - \sqrt{x_1^2 + x_2^2}$
$D_1 \cup D_2$	$r_\cup(x) = x_1 + x_2 + \sqrt{x_1^2 + x_2^2}$
D^c	$r_c(x) = -x$

Tabelle 4.1: Mengenoperationen und die zugehörigen R-Funktionen

Die explizite Form der Gewichtsfunktion kann kompliziert sein. Jedoch ist dies für Berechnungen irrelevant. Die R-Funktionen-Methode stellt ein voll automatisierbares Verfahren bereit, mit dem man Gewichtsfunktionen für Mengen, die auf Ungleichungen basieren, erstellen kann. Auswertung und Differenziation werden durchgeführt mit den algorithmischen Definitionen und mit Hilfe von automatischer Differenziation [14], [22]. Dieses Verfahren ist Algorithmen aus der constructive solid geometry ähnlich.
Für allgemeine, von Freiformkurven oder -Flächen begrenzte Gebiete müssen Gewichtsfunktionen numerisch konstruiert werden. Dies wird hier basierend auf der Abstandsfunktion getan.

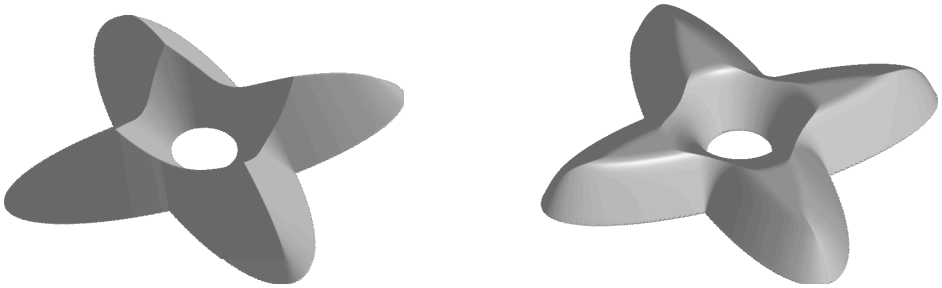


Abbildung 4.2: Konstruktion einer Standard-Gewichtsfunktion anhand der Abstandsfunktion und einem Plateau der Höhe 1

Auf dem Gebietsrand ∂D verschwindet die Abstandsfunktion und sie ist positiv im Inneren. Allerdings ist sie nicht als Gewichtsfunktion geeignet, da sie dadurch, dass sie nur stetig auf D ist, die Differentiationsordnung der Ansatzfunktionen und somit die Approximationsordnung der B-Splines zerstört. Dies lässt sich aber wie folgt umgehen:
Die Abstandsfunktion wird nah am Rand benutzt, wo sie frei von Singularitäten ist und wird glatt in ein Plateau in der Mitte des Gebietes überführt. Genauer definiert man

$$w(x) = 1 - (\max(\delta - \operatorname{dist}(x, \partial D), 0)/\delta)^{\gamma}\,, \tag{4.3}$$

wobei δ die Größe des Plateaus und γ die Glattheit kontrolliert. Das Plateau erleichtert die Benutzung von vorher berechneten Werten beim Aufstellen der Finite Elemente Matrizen. Jedoch sollte es nicht zu groß gewählt werden, um die Ableitungen der Gewichtsfunktion klein zu halten. Ein Beispiel hierfür ist in Abbildung 4.2 zu sehen.
Wie aus den Beispielen oben ersichtlich, steckt viel Flexibilität in der Konstruktion von Gewichtsfunktionen. Speziell können Spline-Approximationen und numerische Techniken effektiv mit der R-Funktionen Methode kombiniert werden. Zum Beispiel können geglättete Abstandsfunktionen als Argument von R-Funktionen verwendet werden. Ausführliche Informationen über verschiedene Arten von Gewichtsfunktionen sind der Literatur zu entnehmen, beispielsweise [11] und [12], sowie den Arbeiten von Rvachev [23], [24] und [25].

4.3 Erweiterung

Die B-Spline Basis ist nicht gleichmässig stabil, wenn sie auf D eingeschränkt wird. Es kann also B-Splines b_k geben, deren Träger nur eine sehr kleine Schnittmenge mit dem Gebiet D besitzt. Dies kann sich beispielsweise negativ auf die Kondition der Galerkin-Matrix und die Lösbarkeit von Ritz-Galerkin-Systemen auswirken. Eine gut konditionierte Basis kann mit Hilfe der Klassifikation der relevanten B-Splines b_k für das Gebiet D in innere und äußere B-Splines konstruiert werden.

Definition 4.4 *Die Menge der Gitterzellen $Q = \ell h + [0,1]^m h$ wird unterteilt in innere, äußere und Randzellen, d.h. Q ist eine innere Zelle, wenn $Q \subseteq \overline{D}$, eine äußere Zelle, wenn $Q \cap D = \emptyset$ und eine Randzelle, wenn das Innere von Q von ∂D geschnitten wird.*

Die relevanten B-Splines $b_k, k \in K$ werden unterschieden in innere B-Splines

$$b_i,\ i \in I^n_{D,h} = \{k \in K^n_{D,h} : \exists\, \ell \in \mathbb{Z}^m \text{ mit } Q_\ell \subset \overline{D} \cap \operatorname{supp} b^n_{k,h}\}\,,$$

die mindestens eine innere Zelle Q_i in ihrem Träger haben und äußere B-Splines

$$b_j,\ j \in J^n_{D,h} = K^n_{D,h} \setminus I^n_{D,h},$$

deren Träger nur aus äußeren und Randzellen besteht.

Die Klassifikation der Randzellen und die Einteilung in innere und äußere B-Splines sind in Abbildung 4.3 dargestellt. Randzellen sind rot unterlegt und innere Zellen grün, während die äußeren Zellen weiss unterlegt sind. Die B-Splines sind mit einem schwarzen Kreis für innere und einem weissen Kreis für äußere B-Splines in der linken unteren Ecke des Trägers markiert.

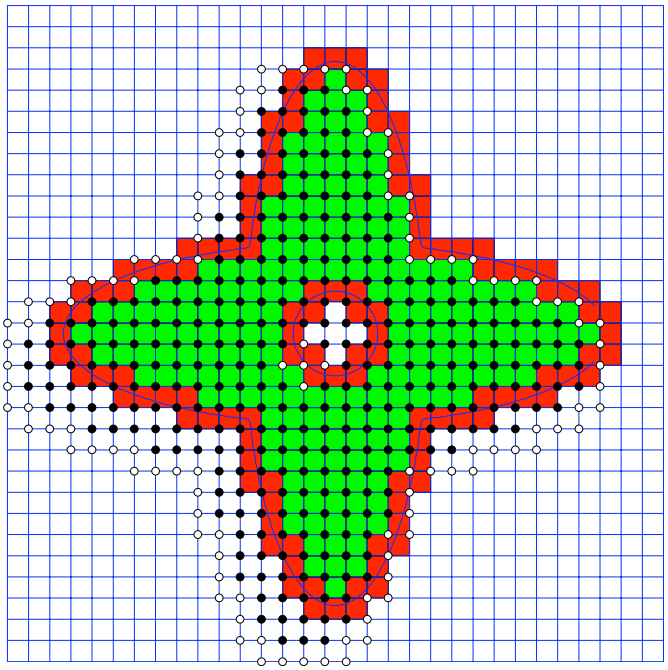


Abbildung 4.3: Klassifizierung der Zelltypen und innere und äußere biquadratische B-Splines

Um die Basis zu stabilisieren, werden die äußeren B-Splines b_j, für die, nach Definition, keine Gitterzelle ihrer Träger ganz in D enthalten ist, an die inneren B-Splines b_i angehängt. Dies muss so geschehen, dass die polynomiale Präzision, welche volle Approximationsordnung garantiert, erhalten bleibt. Deshalb betrachtet man die multivariate Marsden-Identität

$$p(x) = \sum_{i \in I^n_{D,h}} q(i) b_i(x) + \sum_{j \in J^n_{D,h}} q(j) b_j(x)\,, \qquad x \in D\,. \tag{4.4}$$

Da bekannt ist, dass q ein Polynom vom Grad $\leq n$ ist, kann man jeden Koeffizienten $q(j)\,,\ j \in J^n_{D,h}$ aus n^m Koeffizienten $q(i)$, die zu einer Matrix von Indizes

$$I^n_{D,h}(j) := \ell + \{1,\ldots,n\}^m \subset I^n_{D,h}\,, \qquad (\ell = \ell(j))\,,$$

gehören, die am nächsten bei j sind, berechnen. Um $q(j)$ zu berechnen, interpoliert man die Werte $q(i)$ an den Indizes in $I^n_{D,h}(j)$ und wertet den Interpolanden an j aus. Folglich ist, wenn

$$e_{i,j} := \prod_{\mu=1}^{m} \prod_{\alpha=1,\alpha\neq i_\mu-\ell_\mu}^{n} \frac{j_\mu - \ell_\mu - \alpha}{i_\mu - \ell_\mu - \alpha}$$

den Wert des zu $i \in I^n_{D,h}(j)$ gehörigen Lagrange Polynoms bei j beschreibt,

$$q(j) = \sum_{i\in I^n_{D,h}(j)} e_{i,j} q(i)\,.$$

Einsetzen des Ausdrucks für $q(j)$ in (4.4) und Vertauschen der Summen führt zu

$$p(x) = \sum_{i\in I^n_{D,h}} q(i) \left[b_i(x) + \sum_{j\in J^n_{D,h}(i)} e_{i,j} b_j \right]\,, \qquad x \in D\,,$$

wobei $J^n_{D,h}(i) \subset J^n_{D,h}$ die Menge aller Indizes j ist, für die $i \in I^n_{D,h}(j)$. Der Term in eckigen Klammern ist die geeignete Erweiterung der inneren B-Splines b_i.
Verwendet man diese Linearkombinationen als neue Basisfunktionen, können alle Polynome vom Grad $\leq n$ dargestellt werden, ohne explizit auf die äußeren B-Splines zurückzugreifen. Wie vorher kann hier auch wieder mit einer Gewichtsfunktion multipliziert werden, um homogene Randbedingungen einzuhalten. Es wird mit $w(x)/w(x_i)$ multipliziert, wobei x_i der Mittelpunkt der inneren Gitterzelle $Q_i \subset \overline{D} \cap \operatorname{supp} b_i$ ist.
Dies führt zur folgenden Definition:

Definition 4.5 *Sei für einen äußeren Index $j \in J^n_{D,h}$ $I^n_{D,h}(j) = \ell + [0,\ldots,n]^m \subset I^n_{D,h}$ eine m-dimensionale Matrix von Indizes möglichst nahe an j, vorausgesetzt h ist klein genug, so dass eine solche Matrix existiert. Seien weiterhin*

$$e_{i,j} := \prod_{\mu=1}^{m} \prod_{\alpha=1,\alpha\neq i_\mu-\ell_\mu}^{n} \frac{j_\mu - \ell_\mu - \alpha}{i_\mu - \ell_\mu - \alpha}$$

die Werte der zu $I^n_{D,h}(j)$ gehörigen Lagrange-Polynome und $J^n_{D,h}(i)$ die Menge aller j mit $i \in I^n_{D,h}(j)$. Die gewichteten erweiterten B-Splines (WEB-Splines)

$$B_i := \frac{w}{w(x_i)} \left[b_i + \sum_{j\in J^n_{D,h}(i)} e_{i,j} b_j \right]\,, \qquad i \in I^n_{D,h}\,,$$

mit x_i dem Mittelpunkt einer inneren Gitterzelle $Q_i \subset D$ im Träger von b_i bilden eine Basis für den WEB-Raum $w^e\mathbb{B}^n_h(D)$.

Mit Ausnahme der Positivität erben die WEB-Splines alle grundlegenden Eigenschaften der B-Splines. Einige wichtige Eigenschaften werden im Folgenden dargestellt:
Für die Erweiterungskoeffizienten gilt $|e_{i,j}| \preceq 1$ und

$$e_{i,j} = 0 \quad \text{für} \quad \|i - j\| \succeq 1 \,.$$

Des Weiteren müssen nur $\preceq h^{1-m}$ B-Splines in der Nähe des Randes modifiziert werden. Für den Hauptteil der inneren Indizes i ist $B_i = w/w(x_i)b_i$ für kleiner werdendes h. Dies folgt ebenso wie die folgende Eigenschaft direkt aus Definition 4.5.
Die WEB-Splines B_i haben lokalen Träger, supp B_i ist $\preceq 1$, und auf jeder Menge $Q \cap \overline{D}$ verschwinden nur $\preceq 1$ WEB-Splines nicht.
Direkt aus der linearen Unabhängigkeit der B-Splines folgt diese auch für die WEB-Splines

$$B_i(x) = \frac{w(x)}{w(x_i)} b_i(x) \,, \quad x \in Q_i \,.$$

Zusätzlich ist

$$\| \sum_i c_i B_i \|_0 \asymp h^{m/2} \|C\|_2 \,,$$

speziell ist $\|B_i\|_0 \asymp h^{m/2}$.
Der WEB-Raum $w^e \mathbb{B}_h$ enthält gewichtete Polynome vom Grad $\leq n$. Darüber hinaus ist

$$\inf_{u_h \in w^e \mathbb{B}_h} \|u - u_h\|_0 \preceq h^{n+1} \,,$$

wenn w und u/w glatt sind. Bei allen aufgeführten Eigenschaften hängen die Konstanten vom Grad n, dem Gebiet D und der Gewichtsfunktion w ab. Im letzten Fall zusätzlich noch von der approximierten Funktion u.
Aussagen und Beweise beispielsweise zur Stabilität und Approximationsordnung von WEB-Splines sind der Literatur ([16],[30]) zu entnehmen. Wichtige Ergebnisse, die im weiteren Verlauf dieser Arbeit notwendig sind, werden an entsprechender Stelle genannt.

5 Gewichtsfunktionen für einfache Objekte

Bisher wurde der Begriff der Gewichtsfunktion mit dem Beispiel der geglätteten Abstandsfunktion definiert und das Konzept der R-Funktionen vorgestellt. In diesem Kapitel werden implizite Gewichtsfunktionen von einfachen Objekten im Zwei- und Dreidimensionalen vorgestellt, die zum Erstellen von komplexeren Objekten anhand von R-Funktionen zusammengesetzt werden können. Als Grundobjekte werden hier Quadriken in zwei und drei Dimensionen betrachtet. Dies stellt eine flexible Klasse von Objekten dar, um komplexere Objekte daraus zu erstellen. Zum Abschluss des Kapitels wird der K-Form Algorithmus von Jürgen Koch [21] zum Lösen nichtlinearer Gleichungssysteme vorgestellt. Dieser eignet sich besonders, um Schnittpunkte von Kegelschnitten zu berechnen, da nur wenige Schritte notwendig sind, um eine implizite Gleichung eines Kegelschnittes auf K-Form umzuformen. Ein weiterer Vorteil, den man durch die Verwendung von Quadriken erhält, ist die einfache Struktur des Randes der Quadriken. Es kann beispielsweise nicht vorkommen, dass eine Gitterlinie beliebig oft auf einem festen Abschnitt geschnitten wird, was die Behandlung und Berechnung der Schnittpunkte des Gebietsrandes mit den Gitterlinien einfacher macht. Sehr ausführliche Informationen über Quadriken im geometrischen Design bietet [2].
Eine elegante Darstellung der Quadriken sowie wichtige Eigenschaften sind Bestandteil des folgenden Abschnittes.

5.1 Quadriken

Quadriken lassen sich mit Hilfe von homogenen Koordinaten in einer sehr eleganten Form darstellen.

Definition 5.1 *Die euklidischen Koordinaten $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ identifiziert man mit den homogenen Koordinaten in $\mathbb{R}^{n+1}$ durch*

$$(x_1, \ldots, x_n) \sim (\alpha x_1, \ldots, \alpha x_n, \alpha)$$

für beliebiges $\alpha \in \mathbb{R} \setminus 0$, beziehungsweise schreibt man

$$x \sim (\alpha x, \alpha) \sim (x, 1)\,.$$

Im $\mathbb{R}^3$ gilt also

$$(x/w, y/w, z/w) \sim (x, y, z, w)$$

Mit dieser Definition lassen sich Quadriken Q in der folgenden Darstellung angeben.

Definition 5.2 *Eine zweidimensionale Quadrik Q mit der impliziten Darstellung*

$$ax^2 + 2bxy + cy^2 + 2dx + 2ey + f = 0$$

lässt sich in homogener Form schreiben als

$$(x, y, w) \begin{pmatrix} a & b & d \\ b & c & e \\ d & e & f \end{pmatrix} \begin{pmatrix} x \\ y \\ w \end{pmatrix} =: (x, y, w)\ Q\ (x, y, w)^t = 0$$

Analog lässt sich eine Quadrik im $\mathbb{R}^3$ mit der impliziten Darstellung

$$ax^2 + by^2 + cz^2 + 2dxy + 2exz + 2fyz + 2gx + 2hy + 2jz + k = 0$$

darstellen als

$$(x, y, z, w) \begin{pmatrix} a & d & e & g \\ d & b & f & h \\ e & f & c & j \\ g & h & j & k \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = (x, y, z, w)\ Q\ (x, y, z, w)^t = 0.$$

Für $w = 1$ erhält man jeweils den euklidischen Fall.

Betrachtet man w als eine weitere Unbekannte und multipliziert die Matrixdarstellung aus so sieht man, dass durch w alle Summanden Monome vom totalen Grad 2 sind, es wird gewissermaßen mit w aufgefüllt.
Die Punkte auf dem Rand einer Quadrik mit einer waagrechten oder senkrechten Tangente beziehungsweise Tangentialebene werden hier als Extrempunkte bezeichnet. Sie spielen hier eine Rolle, da beispielsweise bei einer Ellipse die vier Tangenten in den Extrempunkten das kleinste achsenparallele Rechteck bilden, dass die Ellipse vollständig enthält. Diese Information wird beispielsweise bei der Gittergenerierung verwendet.
Später im Verlauf dieser Arbeit werden Tangenten an die Gewichtsfunktion benötigt. Mit der Information, welche Quadrik für das entsprechende Randstück herangezogen werden muss, kann man direkt die Tangente angeben:
Die Tangente an die Quadrik im Punkt $(x_0, y_0, 1)$ ist gegeben durch

$$(x, y, 1)\ Q\ (x_0, y_0, 1)^t = 0.$$

5.2 Klassifikation von Quadriken

Die hier verwendeten Quadriken sind natürlich nicht immer in Normalform gegeben, da sie gedreht und verschoben werden. Allerdings ist es wünschenswert, auch anhand der Matrix einer Quadrik in allgemeiner Lage zu sehen, um welchen Typ von Quadrik es sich handelt. Dazu gibt es mehrere Möglichkeiten, allerdings bietet sich bei der hier gewählten Darstellung der Quadriken die folgende Vorgehensweise an. Die Matrix Q wird durch Ähnlichkeitstransformationen auf Diagonalgestalt gebracht. Auf der Diagonalen von Q stehen die Koeffizienten von x^2, y^2, w^2 und gegebenenfalls z^2. Im Grunde genommen ist dies nichts

anderes, als quadratisches Ergänzen beziehungsweise eine Hauptachsentransformation in der Polynomdarstellung.

Die Klassifizierung der so transformierten Quadrikmatrizen erfolgt anhand der Tabelle zur Klassifikation affiner Quadriken, die aus der Literatur wohl bekannt sind.

Eine andere Möglichkeit der Klassifikation von Quadriken ist, die Eigenwerte der Untermatrix $A = Q_{1:n,1:n}$ zu berechnen, denn es ist möglich jede Quadrik durch eine Hauptachsentransformation auf Normalgestalt zu bringen, in der die Quadrik anhand der Eigenwerte von A klassifiziert werden kann.

5.3 Eigenschaften von Quadriken

Im solid modelling spielen Quadriken eine wichtige Rolle, da es dort darauf ankommt schnell und sicher zu testen, ob ein Punkt innerhalb des Gebietes liegt oder nicht. Bei Quadriken ist dies möglich, im Gegensatz zu Objekten, die mit mindestens kubischen Kurven berandet sind, wo dies weder schnell noch absolut sicher möglich ist [9]. In dieser Arbeit werden Quadriken verwendet, um sie zu einer Gewichtsfunktion zu verknüpfen. Somit ist auch für die Gewichtsfunktion sichergestellt, dass schnell und sicher getestet werden kann, ob ein Punkt im Inneren liegt oder nicht.

Einige schöne Beispiele für Quadriken denen man in der Architektur von Gebäuden im Alltag begegnet, sind eingestreut in [1], wie beispielsweise das Los Manantiales Restaurant in Xochimilco, D.F., Mexiko, das aus acht gleichen Teilen eines hyperbolischen Paraboloids besteht.

Ein Beispiel für ein dreidimensionales Objekt, das mit der hier vorgestellten Implementierung anhand von R-Funktionen zusammengesetzt ist sieht man in Abbildung 5.1.

Abbildung 5.1: Einzelteile und das fertige Objekt eines 3d Objektes

5.4 K-Form Algorithmus

Der von J. Koch entwickelte K-Form Algorithmus [21] ist ein Algorithmus zur numerischen Berechnung aller Lösungen eines schwachbesetzten polynomialen Gleichungssystems. Hierbei wird das nichtlineare Gleichungssystem in ein äquivalentes Gleichungssystem in kanonischer Form überführt und sich die Eigenschaft zunutze gemacht, dass Polynome im Gegensatz zu allgemeinen nichtlinearen Funktionen Elemente eines linearen Raumes endlicher Dimension sind. Die Umformung des Gleichungssystems in eine kanonische Form geschieht mit einer Komposition von einzelnen Substitutionsschritten. Mit einem Nullstellentest, mit dem die Existenz und Nichtexistenz von Lösungen eines polynomialen Gleichungssystems überprüft werden kann, berechnet der Algorithmus numerisch alle Lösungen eines polynomialen Gleichungssystems durch sukzessives Unterteilen und Testen der Intervalle. Hier wurde bewußt darauf verzichtet zu sehr ins Detail zu gehen; ausführlichere Informationen, Beweise und Beispiele sowie Anwendungen sind bei J. Koch in [21] zu finden.

5.4.1 Problemstellung

Bestimme numerisch die reellen $x \in X$ mit

$$p(x) = 0, \tag{5.1}$$

wobei $X = [a_1, b_1] \times \cdots \times [a_d, b_d]$ ein Intervall im $\mathbb{R}^d$ und $p = (p_1, \ldots, p_d)$ ein System von d Polynomen vom Grad m ist.

Beispiel 5.3 *Die Bestimmung der Schnittpunkte des Kreises*

$$K : p_1(x) = x_1^2 + x_2^2 - 9$$

und der Ellipse

$$E : p_2(x) = \frac{1}{9}x_1^2 + \frac{1}{3}x_2^2 - 2$$

ist ein Beispiel für ein polynomiales Gleichungssystems $p(x) = 0$. Das Intervall X sollte die Lösungsmenge enthalten und kann hier als $X = [-4, 4] \times [-3, 3]$ gewählt werden.

5.4.2 Schwachbesetzte Polynome

Analog zu Matrizen sind schwachbesetzte Polynome Polynome mit nur wenigen von Null verschiedenen Koeffizienten. Ein Polynom mit t von Null verschiedenen Koeffizienten nennt man t-sparse. Dabei hängt es nicht von einem Polynom, sondern von dessen Darstellung, seiner Basis ab, ob es schwach besetzt ist. Beispielsweise ist das Polynom

$$p(x_1, x_2) = (x_1 - 1)^2(x_2 - 1)^2$$

bezüglich der Basis

$$\{1, x_1 - 1, (x_1 - 1)^2)\} \times \{1, x_2 - 1, (x_2 - 1)^2\}$$

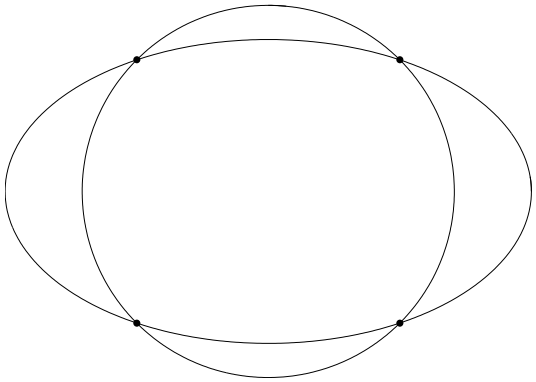


Abbildung 5.2: Schnittpunkte von Kreis und Ellipse

1-sparse, bezüglich der Basis der Monome allerdings

$$p(x_1, x_2) = x_1^2x_2^2 - 2x_1^2x_2 + x_1^2 - 2x_1x_2^2 + 4x_1x_2 - 2x_1 + x_2^2 - 2x_2 + 1$$

vollbesetzt.

5.4.3 Berechnung der K-Form

Die Überführung eines polynomialen Gleichungssystems in ein äquivalentes mit maximal quadratischen Termen erfolgt durch die Einführung neuer Variablen und Gleichungen. Die Intervalle für die neuen Variablen werden anhand einfacher Intervallarithmetik berechnet, dabei ist nur zu beachten, dass

$$[a,b]^2 = [0, \max(a^2, b^2)] \quad \text{kleiner ist als} \quad [a,b] \times [a,b] = [\min(a^2, ab, b^2), \max(a^2, ab, b^2)]$$

wenn a und b unterschiedliche Vorzeichen haben.

Beispiel 5.4 *Das Gleichungssystem aus Beispiel (5.3) lautet in K-Form*

$$\begin{aligned} x_3 + x_4 - 9 &= 0 \\ \frac{1}{9}x_3 + \frac{1}{3}x_4 - 2 &= 0 \\ x_3 - x_1^2 &= 0 \\ x_4 - x_2^2 &= 0 \end{aligned}$$

mit

$$(x_1, x_2, x_3, x_4) \in [-4, 4] \times [-3, 3] \times [0, 16] \times [0, 9]\,.$$

Die Transformation in eine K-Form lässt sich durch einen Substitutionsoperator beschreiben, der ein polynomiales Gleichungssystem durch die Einführung neuer Variablen in ein äquivalentes transformiert, in dem die Nichtlinearität maximal quadratisch ist. Der Substitutionsoperator

$$\mathcal{S} : (p, X) \to (q, X)$$

erzeugt durch Komposition einzelner Substitutionsschritte

$$\mathcal{S}_{ij,} = \mathcal{S}_{i_s,j_s} \circ \cdots \circ \mathcal{S}_{i_1,j_1}$$

die K-Form, wobei jeder Substitutionsschritt mit dem Paar i, j eine neue Gleichung $x_k - x_i x_j = 0$ erzeugt. Jede dieser Substitutionen soll so oft wie möglich ausgeführt werden, so dass $\mathcal{S}_{1,1}$ für x_1^4 nicht $x_1^2 x_2$ liefert, sondern x_2^2. Durch den inversen Substitutionsoperator $\mathcal{S}^{-1}$ kann die K-Form wieder in das ursprüngliche Gleichungssystem überführt werden.
So wird ein polynomiales Gleichungssystem p sukzessive in ein äquivalentes Gleichungssystem q überführt, bei dem die ersten d Gleichungen ein lineares Gleichungssystem

$$Nx + a = 0$$

mit einer schwachbesetzten Matrix N und einem Vektor a bilden und die weiteren Gleichungen die Form

$$0 = x_k - x_i x_j, \quad i \leq j < k$$

haben. Diese weiteren Gleichungen werden durch die Indizes i und j in Vektoren dargestellt. Jedoch ist eine K-Form nicht eindeutig (siehe das folgende Beispiel aus [21]), weshalb sich die Frage nach der besten bezwiehungsweise geeignetsten K-Form stellt.

Beispiel 5.5 *Das polynomiale Gleichungssystem*

$$\begin{aligned} x_1^2 x_2^2 + x_1 x_2 - 1 &= 0 \\ x_1 x_2 + x_1 - 1 &= 0 \end{aligned}$$

wird durch die Substitution $\mathcal{S}_{(1,2,1,3),(1,2,2,4)}$ *in die K-Form*

$$\begin{aligned} x_6 + x_5 - 1 &= 0 \\ x_5 + x_1 - 1 &= 0 \\ x_3 - x_1^2 &= 0 \\ x_4 - x_2^2 &= 0 \\ x_5 - x_1 x_2 &= 0 \\ x_6 - x_3 x_4 &= 0 \end{aligned}$$

transformiert. Die Matrix N *und die Vektoren* a, i *und* j *sind dann*

$$N = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \qquad a = \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \qquad i = \begin{pmatrix} 1 \\ 2 \\ 1 \\ 3 \end{pmatrix}, \qquad j = \begin{pmatrix} 1 \\ 2 \\ 2 \\ 4 \end{pmatrix}.$$

Eine bessere Alternative erzeugt $\mathcal{S}_{(1,3),(2,3)}$*, da für diese Transformation die K-Form*

$$\begin{aligned} x_4 + x_3 - 1 &= 0 \\ x_3 + x_1 - 1 &= 0 \\ x_3 - x_1 x_2 &= 0 \\ x_4 - x_3^2 &= 0 \end{aligned}$$

lediglich aus vier Gleichungen besteht. Die Matrix N und die entsprechenden Vektoren sind dann

$$N = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}, \qquad a = \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \qquad i = \begin{pmatrix} 1 \\ 3 \end{pmatrix}, \qquad j = \begin{pmatrix} 2 \\ 3 \end{pmatrix}.$$

Wie findet man die geeignete Substitution? Eine Möglichkeit wäre es, sämtliche K-Formen zu berechnen und diejenige mit den wenigsten Substitutionen zu wählen. Dies ist allerdings sehr aufwändig und deshalb nicht zu empfehlen. Ein besserer Weg ist es, bei jedem Substitutionsschritt denjenigen zu wählen, der die Nichtlinearität am meisten reduziert. Bei mehreren Substitutionen mit der gleichen Reduktion der Nichtlinearität wählt man diejenige mit dem kleinsten Index. Dies entspricht der lexikographischen Ordnung.
Man kann dies beispielsweise umsetzen, indem man vor jedem Substitutionsschritt eine Matrix R erstellt, die an der Stelle $r_{i,j}$ die Höhe der Reduktion für $x_i^n x_j^m$ enthält. Da die Reduktion von $x_i^n x_j^m$ gleich der Reduktion von $x_j^m x_i^n$ ist, genügt es eine Dreiecksmatrix zu erstellen, um keine doppelten Einträge zu erhalten. Die Reduktion von $x_i^n x_j^m$ beträgt $\frac{nm}{2}$, zu Null hin auf ganze Zahlen gerundet.

Beispiel 5.6 *Die Matrizen R_k für die Substitutionsschritte im obigen Beispiel lauten hiernach*

$$R_1 = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix} \qquad \textit{und} \qquad R_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

5.4.4 K-Form Löser

Ausgehend von disjunkten d-dimensionalen Intervallen approximiert der Algorithmus sämtliche Lösungen des polynomialen Gleichungssystems in K-Form in diesen Intervallen. Der Algorithmus arbeitet eine Liste von Intervallen ab, die sogenannte *todo* Liste. Die Intervalle in dieser Liste werden sukzessive kleiner. Mit einem Nullstellentest, der sich besonders für polynomiale Gleichungssysteme in K-Form eignet, wird für jedes Intervall in der Liste getestet, ob Nullstellen, also Lösungen, in diesem Intervall existieren. Abhängig vom Ergebnis des Tests wird das Intervall aus der Liste entfernt, ein neues Intervall berechnet oder das vereinfachte Newton-Verfahren gestartet, wenn nur eine Lösung im Intervall existiert. Hierbei können schmale Intervalle, d.h. Intervalle, die in mindestens einer Dimension den Radius 0 beziehungsweise einen Radius betragsmässig kleiner als eine gegebene Schranke haben, auftreten, die gesondert bearbeitet werden müssen.

5.4.5 Nullstellentest

Der im Folgenden beschriebene Nullstellentest zur Überprüfung der Existenz oder Nichtexistenz von Lösungen eines polynomialen Gleichungssystems in einem Intervall eignet sich besonders für polynomiale Gleichungssysteme in K-Form, ist aber auch auf beliebige nichtlineare polynomiale Gleichungssysteme anwendbar.
Das polynomiale Gleichungssystem (5.1) wird auf dem Intervall X durch eine affine Funktion

$$Lx = b + A(x - X_m)$$

approximiert. O.B.d.A. ist diese Darstellung von L bezüglich des Intervallmittelpunktes X_m gewählt. Der Approximationsfehler

$$e(x) = Lx - p(x)$$

wird komponentenweise durch

$$\max_{x \in X} |e(x)| \leq c$$

mit einer geeigneten Konstante c abgeschätzt, die nicht von x, sondern lediglich von p, X und L abhängt.
Geometrisch bedeutet dies, dass die Polynome als funktionale Flächen betrachtet durch ihre Tangentialebenen ersetzt werden. Die polynomialen Flächen lassen sich in tangentialer Richtung durch zwei Ebenen einschliessen. Die gemeinsamen Nullstellen liegen im Durchschnitt der Streifen, die die Funktionswerte der Polynome einschliessen, und der Ebene $x_3 = 0$, was im folgenden Lemma beschrieben wird.

Lemma 5.7 *Alle Lösungen des polynomialen Gleichungssystems (5.1) liegen im Spat*

$$S = \{x \mid |Lx| \leq c\}$$

mit dem Mittelpunkt $X_m - A^{-1}b$, der von den Vektoren $c_i A^{-1} e_i$ aufgespannt wird, wobei e_i den i-ten Einheitsvektor bezeichnet, und davon ausgegangen wird, dass A eine reguläre Matrix ist.

Durch die Extremwerte des Spats S ist das neue Intervall Y bestimmt als kleinstes Intervall, das S enthält.

Lemma 5.8 *Das Intervall Y mit dem Mittelpunkt $Y_m = X_m - A^{-1}b$ und dem Radius $Y_r = |A^{-1}|c$ ist das kleinste Intervall, das den Spat S enthält.*

Die Beweise der beiden vorangegangenen Lemmata sind in Kapitel 4.1 von [21] nachzulesen.

5.4.6 Existenz und Nichtexistenz

Anhand der Konstruktion von Y ist zu sehen, dass alle Lösungen des Gleichungssystems (5.1) in Y enthalten sind.

Korollar 5.9 *Alle Lösungen des polynomialen Gleichungssystems (5.1) liegen im Durchschnitt von X und Y. Falls die Ausschlussbedingung*

$$X \cap Y = \emptyset \tag{5.2}$$

erfüllt ist, besitzt (5.1) keine Lösung in X.

Aus diesem Korrolar folgt, dass Y die Lösungen aus X enthält, womit sich folgendes Lemma über die Existenzbedingung formulieren lässt:

Lemma 5.10 *Falls die Existenzbedingung*

$$Y \subset X \tag{5.3}$$

erfüllt ist, besitzt das polynomiale Gleichungssystem (5.1) mindestens eine Lösung in Y.

5.4.7 Eindeutigkeit

Für die Eindeutigkeit einer Lösung von (5.1) sucht man eine geeignete Ergänzung zur Existenzbedingung (5.3). War die Existenz einer Lösung noch unabhängig von der Wahl von A, b und c, so ist hier eine geeignete Wahl notwendig. Naheliegend ist die Wahl der Jacobi-Matrix im Mittelpunkt des Intervalls für A und der Wert des Polynoms am Mittelpunkt des Intervalls für b. Das bedeutet, dass L das Taylor-Polynom vom Grad eins mit dem Entwicklungspunkt X_m ist.
A, b und c lassen sich für ein Polynom in K-Form einfach bestimmen mit

$$A_{1:d} = N\,, \qquad b_{1:d} = NX_m + a\,, \qquad c_{1:d} = 0\,,$$

da die ersten d Gleichungen linear sind und deshalb kein Approximationsfehler entsteht. Die restlichen Polynome $p_k(x) = x_k - x_i x_j$ werden durch

$$L_k x = p_k(X_m) + p_k'(X_m)(x - X_m) \tag{5.4}$$

approximiert. Daraus ergibt sich $b_k = X_{m,k} - X_{m,i}X_{m,j}$ und

$$A_{k,i} = -X_{m,j}\,, \qquad A_{k,j} = -X_{m,i}\,, \qquad A_{k,k} = 1$$

im Fall $i \neq j$, sowie

$$A_{k,i} = -2X_{m,i}\,, \qquad A_{k,k} = 1$$

im Fall $i = j$.
Für die Abschätzung des Approximationsfehlers bzw der Fehlerfunktion

$$e_k(x) = -(x_i - X_{m,i})(x_j - X_{m,j})$$

gilt

$$c_k = \max_{x \in X} |e_k(x)| = X_{r,i}X_{r,j}\,. \tag{5.5}$$

Durch Betrachten des eindimensionalen Falls

$$p(x) = a_1(x - X_m) + a_2(x - X_m)^2, \qquad a_1 \neq 0, \qquad x \in [X_m \pm X_r]\,,$$

und Verallgemeinern auf polynomiale Gleichungssysteme in K-Form mit Hilfsmitteln aus der Linearen Algebra erhält man folgenden Satz. Eine genauere Ausarbeitung dieser Herleitung siehe Kapitel 4.3 in [21].

Satz 5.11 *Verwendet man für ein polynomiales Gleichungssystem in K-Form die Approximation (5.4) mit der Fehlerabschätzung (5.5), dann besitzt das polynomiale Gleichungssystem (5.1) genau eine Lösung in X, falls zusätzlich zur Existenzbedingung (5.3)*

$$Y \subset X$$

auch die Eindeutigkeitsbedingung

$$2Y_r < X_r \tag{5.6}$$

erfüllt ist.

Das bedeutet, dass sich der Intervallradius von Y gegenüber X mehr als halbieren muss.

5.4.8 Konvergenz

Man kann eine Lösung mit dem vereinfachten Newton-Verfahren berechnen, sobald die beiden Bedingungen aus Satz 5.11 erfüllt sind.

Lemma 5.12 *Sind die Bedingungen (5.3) und (5.6) erfüllt, so erhält man mit dem vereinfachten Newton-Verfahren*

$$x_{k+1} = x_k - A^{-1}p(x_k)\,, \quad k = 0, 1, 2, \ldots$$

für jeden beliebigen Startwert $x_0 \in X$ die in X enthaltene Lösung von (5.1).

Der Beweis des Lemmas sowie ein Lemma über die Fehlerabschätzung sind in Kapitel 4.4 von [21] zu finden.

5.4.9 Unterteilungsalgorithmus

Im ersten Schritt wird das aktuelle Intervall betrachtet und untersucht, ob es leer ist. Ist es das nicht, wird geprüft, ob es schmal ist, d.h. der Radius des Intervalls ist in einer Dimension kleiner als eine vorgegebene Schranke *tol*. Schmale Intervalle müssen gesondert behandelt werden, da sie unter Umständen numerisch nicht mehr stabil weiter bearbeitet werden können. Ein geeigneter Wert für *tol* ist die Maschinengenauigkeit. Die Abarbeitung der schmalen Intervalle wird am Ende des Algorithmus vorgenommen, bis dahin werden sie auf eine extra Liste, die *thin*-Liste, verschoben.
Trifft beides nicht zu, so wird ein neues Intervall Y berechnet. In seltenen Fällen ist dies nicht möglich, dann wird das Intervall X unterteilt und die beiden entstandenen Intervalle werden an die *todo*-Liste angehängt. Der genaue Ablauf der Unterteilung und Berechnung der Intervalle ist Kapitel 5 in [21] zu entnehmen.
Das Intervall Z wird als Durchschnitt von X und Y berechnet und nach den folgenden vier Fällen behandelt:

1. Z ist leer. X enthält keine Lösungen und muss nicht weiter bearbeitet werden
2. X enthält eine eindeutige Lösung, die mit dem vereinfachten Newton-Verfahren berechnet wird.
3. Z ist deutlich kleiner als X. Z wird weiter bearbeitet und auf die *todo*-Liste verschoben, da alle Lösungen aus X auch in Z enthalten sind.
4. Z ist nicht deutlich kleiner als X. Z wird unterteilt und die beiden daraus resultierenden Intervalle werden auf die *todo*-Liste verschoben und weiter bearbeitet.

5.4.10 Behandlung schmaler Intervalle

Die gesonderte Bearbeitung der oben auftretenden schmalen Intervalle geschieht wie im Folgenden beschrieben:

```
while todo list ≠ ∅
  pop X from todo list
  if X is empty
    nothing to do
  elseif X is thin
    push X on thin list
  else
    calculate Y
    if calculation of Y failed
      X_left, X_right ← subdivision (X)
      push X_left, X_right on todo list
    else
      Z = X ∩ Y
      if Z = ∅
        no solution, nothing to do
      elseif Y ⊂ X and 2Y_r < X_r
        unique solution, start Newton–iteration
      elseif 2vol(Z) < vol(X)
        push Z on todo list
      else
        Z_left, Z_right ← subdivision (Z)
        push Z_left, Z_right on todo list
      end
    end
  end
end
```

Abbildung 5.3: Unterteilungsalgorithmus [21]

Auftreten können schmale Intervalle aus verschiedenen Gründen, etwa bei Nullstellen, an denen die Jacobi-Matrix singulär ist, was ein in allen Komponenten schmales Intervall zur Folge hat. Bei Problemen wie dem im folgenden Beispiel beschriebenen, treten schmale Intervalle auf, da hier der Unterteilungsalgorithmus erkennt, dass es sich eigentlich um ein lineares Problem handelt.

Beispiel 5.13 *Durch die Substitutionen $\mathcal{S}_{1,1}$ und $\mathcal{S}_{2,2}$ ergibt sich für $p(x)$ aus Beispiel (5.3) die K-Form*

$$\begin{aligned} x_3 + x_4 - 9 &= 0 \\ \frac{1}{9}x_3 + \frac{1}{3}x_4 - 2 &= 0 \\ x_3 - x_1^2 &= 0 \\ x_4 - x_2^2 &= 0 \end{aligned}$$

Daraus ergeben sich

$$a = \begin{pmatrix} -9 \\ -2 \end{pmatrix}, \qquad N = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & \frac{1}{9} & \frac{1}{3} \end{pmatrix}$$

mit $X = [-3,5] \times [-4,8] \times [0,25] \times [0,64]$ *beziehungsweise*

$$X_m = \begin{pmatrix} 1 \\ 2 \\ 12,5 \\ 32 \end{pmatrix} \qquad \textit{und} \qquad X_r = \begin{pmatrix} 4 \\ 6 \\ 12,5 \\ 32 \end{pmatrix}.$$

Die Intervalle wurden hier zur besseren Anschaulichkeit etwas verändert, was keine Auswirkung auf die Lösung hat.
Das neue Intervall Y *berechnet sich aus*

$$A^{-1} = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & \frac{1}{9} & \frac{1}{3} \\ -2 & 0 & 1 & 0 \\ 0 & -4 & 0 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} 0,75 & -2,25 & -0,5 & 0 \\ -0,125 & 1,125 & 0 & -0,25 \\ 1,5 & -4,5 & 0 & 0 \\ -0,5 & 4,5 & 0 & 0 \end{pmatrix}$$

sowie

$$b = \begin{pmatrix} 35,5 \\ \frac{181}{18} \\ 11,5 \\ 28 \end{pmatrix} \qquad \textit{und} \qquad c = \begin{pmatrix} 0 \\ 0 \\ 16 \\ 36 \end{pmatrix}.$$

Man erhält somit

$$Y_m = X_m - A^{-1}b = \begin{pmatrix} 2,75 \\ 2,125 \\ 4,5 \\ 4,5 \end{pmatrix} \qquad \textit{und} \qquad Y_r = |A^{-1}|c = \begin{pmatrix} 8 \\ 9 \\ 0 \\ 0 \end{pmatrix}$$

beziehungsweise $Y = [-5,25\ ;\ 10,75] \times [6,875\ ;\ 11,125] \times [4,5\ ;\ 4,5] \times [4,5\ ;\ 4,5]$.
Man sieht, dass das Intervall in der dritten und vierten Komponente den Radius 0 *hat. Das bedeutet also, das neue Intervall ist schmal.*
Durch Umkehrung der Substitutionen beziehungsweise Auflösen der Gleichungen

$$x_1 = \pm\sqrt{x_3} \qquad \textit{und} \qquad x_2 = \pm\sqrt{x_4}$$

erhält man direkt die Lösungen

$$x_1 = \pm\sqrt{4,5} \qquad \textit{und} \qquad x_2 = \pm\sqrt{4,5}$$

und somit die Schnittpunkte

$$S_{1,2} = \begin{pmatrix} \pm\sqrt{4,5} \\ \sqrt{4,5} \end{pmatrix} \qquad \textit{und} \qquad S_{3,4} = \begin{pmatrix} \pm\sqrt{4,5} \\ -\sqrt{4,5} \end{pmatrix}.$$

Die allgemeine Bearbeitung von schmalen Intervallen läuft folgendermassen ab:
Ist ein Intervall in allen Komponenten schmal, so kann man nach Berechnen von $p(X_m)$ entscheiden, ob man X_m als Lösung akzeptiert.
Ist das Intervall nicht in allen Komponenten schmal, so wählt man den größten Index ℓ, in dessen Komponente das Intervall schmal ist

$$X_{r,\ell} < tol\,, \qquad X_{r,k} > tol \quad \forall\ k > \ell\,.$$

Die Variable x_ℓ bekommt den Wert von $X_{m,\ell}$ zugewiesen, woraus man einen neuen Vektor a berechnen kann mit

$$a = a + X_{m,\ell} N_{:,\ell}\,.$$

Weiterhin wird aus der Matrix N die ℓ-te Spalte gestrichen

$$N = N_{:,1:\ell-1,\ell+1:d+s}\,.$$

Alle Substitutionsgleichungen, in denen x_ℓ auf der rechten Seite auftritt,

$$x_k - x_\ell x_j = 0\,, \qquad x_k - x_i x_\ell = 0\,, \qquad x_k - x_\ell^2 = 0\,,$$

werden durch Einsetzen von x_ℓ zu linearen Gleichungen

$$x_k - X_{m,\ell} x_j = 0\,, \qquad x_k - x_i X_{m,\ell} = 0\,, \qquad x_k - X_{m,\ell}^2 = 0\,.$$

Ist durch $x_\ell = X_{m,\ell}$ der Wert einer der ursprünglichen Variablen festgelegt ($\ell \leq d$), kann man eine der Gleichungen streichen, denn das Gleichungssystem ist überbestimmt mit d Gleichungen und $d-1$ Unbekannten. Hier ist darauf zu achten, eine linear abhängige Zeile zu streichen.
Ansonsten ($\ell > d$) tritt x_ℓ in einer Gleichung der Form

$$x_\ell - x_i^2 = 0 \qquad \text{oder} \qquad x_\ell - x_i x_j = 0$$

auf, da hier eine der durch Substitution entstandenen Variablen festgelegt ist.
Bei Gleichungen des ersten Typs ist $X_{m,\ell} \geq 0$, und man muss lediglich noch überprüfen, welche der beiden Lösungen von

$$x_i = \pm\sqrt{X_{m,\ell}}$$

in Intervall X enthalten ist.
Tritt x_ℓ dagegen in einer Gleichung des zweiten Typs auf, so löst man nach einer Variablen, beispielsweise x_i auf und ersetzt x_i durch $\frac{X_{m,\ell}}{x_j}$. Nach dem Multiplizieren der Gleichungen mit x_j, um wieder polynomiale Gleichungen zu erhalten, ist ein erneutes Berechnen der K-Form notwendig.

6 Triangulierung und Tetraedisierung

Nachdem in Abschnitt 2.5 zu Beginn dieser Arbeit kurz auf Triangulierungen eingegangen wurde, folgt an dieser Stelle eine ausführlichere Betrachtung. Man erzeugt Triangulierungen, indem man in einer Punktmenge durch das Erstellen von Kanten zwischen jeweils zwei Punkten Dreiecke erzeugt. Es entsteht eine zulässige Triangulierung, wenn nur die Punkte der Punktmenge Eckpunkte von Dreiecken sind und sich jeweils zwei Kanten entweder in einem Punkt der Punktmenge oder gar nicht schneiden [10]. Des Weiteren darf es anschliessend nicht mehr möglich sein, weitere Kanten mit diesen Eigenschaften einzufügen, es müssen also schon alle möglichen Kanten erzeugt sein. Eine solche Triangulierung ist in den meisten Fällen nicht eindeutig.

Definition 6.1 *Die Vereinigung von endlich vielen Dreiecken nennt man Triangulierung, wenn jeweils zwei Dreiecke als Durchschnitt entweder*

- *einen Punkt,*
- *eine Kante*
- *oder die leere Menge*

haben, d.h. das Innere der Dreiecke ist disjunkt. Bei einer Triangulierung Δ_I einer Gitterzelle ist I der Index der Zelle, d.h. $I = (i_x, i_y)$ bei einem zweidimensionalen Gitter.

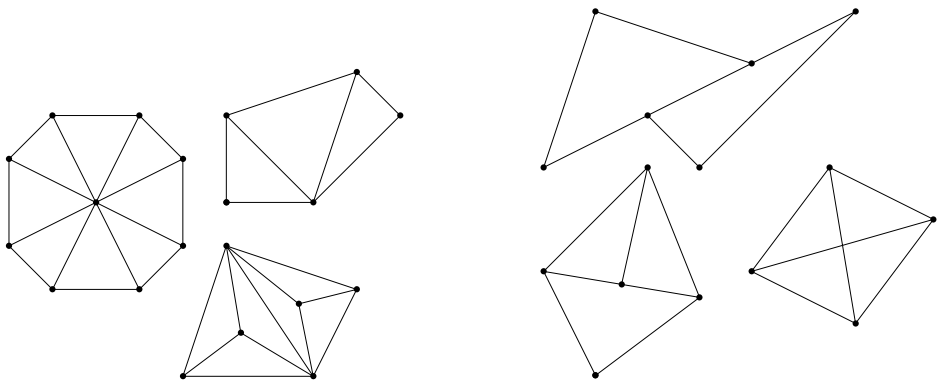


Abbildung 6.1: links: Zulässige Triangulierungen, rechts: Unzulässige Triangulierungen

In Abbildung 6.1 sind links allgemeine Beispiele für zulässige Triangulierungen zu sehen und auf der rechten Seite Beispiele für nicht zulässige Triangulierungen, wobei nur die schwarzen Punkte die Punktmenge markieren.

Üblicherweise bildet die Vereinigung aller Dreiecke die konvexe Hülle der Punktmenge. Bei der Triangulierung der Randzellen ist dies nicht immer der Fall, da nur das Innere des Gebietes D in der Randzelle trianguliert wird und keine Dreiecke außerhalb von D erlaubt sind. Dadurch ist die Triangulierung im eigentlichen Sinn nicht vollständig und bildet auch nicht die konvexe Hülle der Menge der zu triangulierenden Punkte.
Analog kann auch im Dreidimensionalen verfahren werden. Eine Punktmenge im $\mathbb{R}^3$ kann tetraedisiert werden, indem so Kanten zwischen den Punkten erzeugt werden, dass sich jeweils zwei Kanten entweder in einem Punkt der Menge schneiden oder gar nicht. Des Weiteren dürfen die Kanten kein aus drei Kanten bestehendes Dreieck durchstoßen. So entsteht eine Tetraedisierung der Punktmenge, sofern keine weiteren Kanten mit den aufgeführten Eigenschaften erstellt werden können.

Definition 6.2 *Die Vereinigung von endlich vielen Tetraedern nennt man Tetraedisierung, wenn jeweils zwei Tetraeder als Durchschnitt entweder*

- *einen Punkt,*
- *eine Kante,*
- *ein Seitendreieck*
- *oder die leere Menge*

haben. Bei einer Tetraedisierung $\tilde{\Delta}_I$ einer Gitterzelle ist I der Index der Zelle, d.h. $I = (i_x, i_y, i_z)$ bei einem dreidimensionalen Gitter.

6.1 Triangulierung

Der Einfachheit halber wird hier von der Triangulierung der Randzellen gesprochen, wenn der Teil der Randzelle trianguliert wird, der innerhalb des Gebiets liegt. Für die Triangulierung der Randzellen wird hier davon ausgegangen, dass die Gitterweite klein genug gewählt wurde, so dass höchstens zwei Objekte sich in einer Gitterzelle schneiden, beziehungsweise eine Randzelle nicht mehr als zwei Objekte enthält und diese sich gar nicht oder genau einmal in dieser schneiden. Bei den Objekten wird sich, wie bereits erwähnt, auf Halbräume und Quadriken beschränkt, weshalb die Anzahl der Möglichkeiten, wie der Gebietsrand eine Randzelle schneiden kann, noch überschaubar ist.

6.1.1 Fallunterscheidung

Wird die Randzelle von der Kurve eines Quadrikrandes oder von ein oder zwei Geraden geschnitten, so sind, bis auf Symmetrie, die in Abbildung 6.2 dargestellten Fälle möglich. Das Innere beziehungsweise Äußere des Gebietes ist hierbei in jedem der Fälle austauschbar, weshalb es der Übersichtlichkeit nicht markiert wurde. Die Fälle mit einer oder zwei Geraden, die die Randzelle schneiden sind mit gepunkteten Strecken dargestellt und werden analog behandelt, wie die durchgezogen abgebildeten Fälle einer Randkurve. Die Schnittpunkte der beiden Geraden liegen hier der Übersichtlichkeit wegen auf den Extrempunkten

der Kurve. Bei dem unten rechts abgebildeten Fall kann es auch vorkommen, dass der untere Kantenschittpunkt nicht die untere sondern die linke Kante schneidet. Dieser Fall ist analog mit dem darüber abgebildeten Fall. Ebenso können die beiden Kanten mit jeweils zwei Schnittpunkten aus dem Bild unten in der Mitte auch auf benachbarten Kanten liegen.

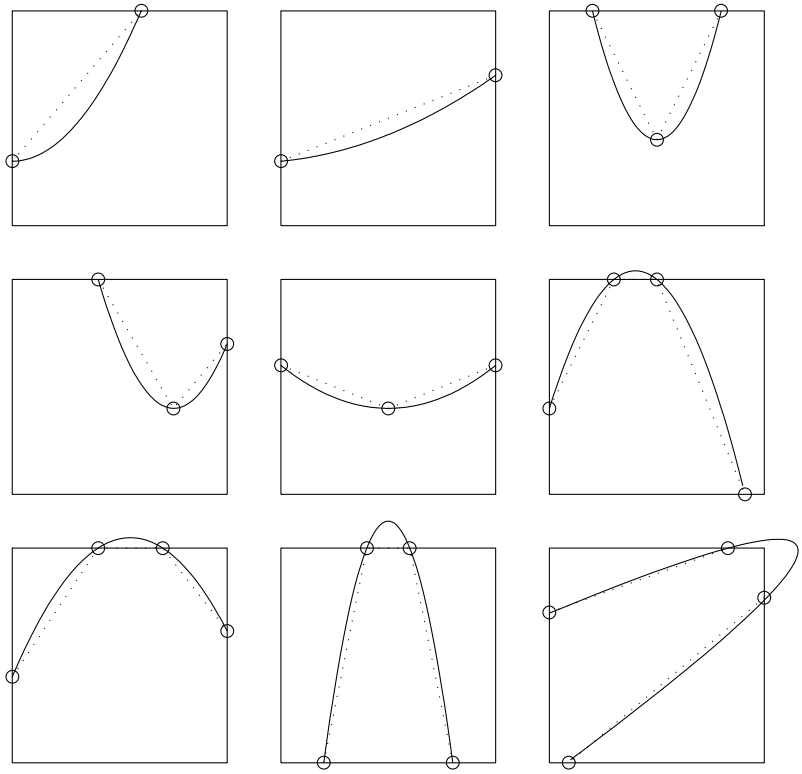


Abbildung 6.2: Einfache Fälle der Randzellen in 2d

Bemerkung 6.3 *Es lässt sich einfach zeigen, dass die Fälle in Abbildung 6.2, bis auf Symmetrie, die einzigen sind, die mit einer Kurve beziehungsweise einer oder zwei Geraden vorkommen können.*

- *Betrachtet man die ersten beide Fälle in der ersten Zeile. Eine Gerade beziehungsweise eine Kurve ohne Extremstelle in der Zelle, die eine Zelle genau zweimal schneidet, schneidet entweder in zwei gegenüberliegenden Kanten oder in zwei Kanten, die einen gemeinsamen Eckpunkt haben, da innerhalb der Zelle keine Richtungsänderungen möglich sind.*

- *Besitzt eine Kurve einen Extrempunkt in der Zelle und schneidet die Zelle genau zweimal, so ist dies analog zu zwei sich in der Zelle schneidenden Halbgeraden. Für die Betrachtung ist der Schnittpunkt analog zu behandeln wie der Extrempunkt. Die Schnittpunkte mit den Zellkanten liegen entweder auf einer einzigen Kante, auf zwei benachbarten Kanten oder zwei gegenüberliegenden Kanten, was durch die drei abgebildeten Fälle dargestellt ist.*

- *Die verbleibenden Fälle von zwei Geraden, die sich nicht innerhalb der Zelle schneiden und einer Kurve, die die Zelle mehr als zweimal schneidet sind in den letzten vier Abbildungen dargestellt. Für die Gerade ist es wieder einfach zu sehen. Es gibt entweder zwei Kanten mit jeweils zwei Schnittpunkten, diese können benachbart oder gegenüberliegend sein, oder es gibt eine Kante mit zwei Schnittpunkten und zwei weitere mit jeweils einem Schnittpunkt. Diese beiden Kanten liegen sich entweder gegenüber oder sind benachbart. Der letzte Fall ist, dass die beiden Geraden alle vier Kanten schneiden, was in der Abbildung unten rechts zu sehen ist.*

  Analog verhält es sich mit einer Kurve, die die Zelle in vier Punkten schneidet. Es können nicht mehr als vier Schnittpunkte bei einer Kurve auftreten, da nur quadratische Kurven verwendet werden und somit kein weiterer Wechsel in der Ableitung möglich ist.

Kantenschnittpunkte in den Zellecken sind möglich, können aber zur Fallbehandlung einer der beiden Kanten zugeordnet werden, so dass sich hierdurch keine neuen Fälle ergeben. Zellen mit drei Schnittpunkten sind nicht möglich, da hierfür entweder ein Geradenschnittpunkt oder ein Extrempunkt der Kurve auf der Gitterkante liegen muss, was bei der Gittergenerierung ausgeschlossen wurde. Zellen mit nur einem Schnittpunkt sind selbstverständlich nicht möglich, da geschlossene Gebiete verwendet werden und die Gerade oder Kurve nicht in der Zelle enden kann. Weitere Fälle können ausgeschlossen werden, wenn die Gitterweite h klein genug gewählt wird.

Die weiteren möglichen Fälle, also mit zwei Kurven in einer Zelle, sind nicht so übersichtlich darstellbar, wenn man allerdings vor der Gittergenerierung einige Einschränkungen an das Gitter beziehungsweise an die Gitterweite macht, lassen sich viele der noch möglichen Fälle ausschließen. Wählt man h klein genug, verlangt man beispielsweise, dass zwischen jeweils zwei Objekt-Schnittpunkten oder Extrempunkten von zwei verschiedenen Objekten mindestens zwei Zellen in mindestens einer Koordinatenrichtung liegen, so lassen sich die noch möglichen Fälle deutlich einschränken.

6.1.2 Triangulierungsalgorithmus

Aufgrund dieser Vorüberlegung lässt sich ein einfacher Triangulierungsalgorithmus formulieren.
Zuerst jedoch einige Bezeichnungen, um die Erklärungen übersichtlicher zu machen. Im folgenden Algorithmus werden die Schnittpunkte des Gebietes mit den Kanten der Zelle mit S_i bezeichnet, die Zellecken, die im Inneren des Gebietes liegen, je nach Anzahl mit I_1 bis I_3 und der Extrem- beziehungsweise Schnittpunkt mit P, wie in der folgenden Abbildung 6.3 zu sehen ist. Der Verlauf der Ränder der Ausgangsobjekte im Inneren des Gebietes ist gestrichelt angedeutet.
Als Erstes wird überprüft, ob in der Zelle ein Extrempunkt des Objekts oder ein Schnittpunkt zweier Objekte liegt. Ist dies der Fall, so wird nach folgendem Schema vorgegangen:

- Erzeuge Dreieckskanten von diesem Punkt P zu den inneren Ecken der Zelle I_k und zu den Schnittpunkten der Randkurve mit den Zellenkanten S_i.

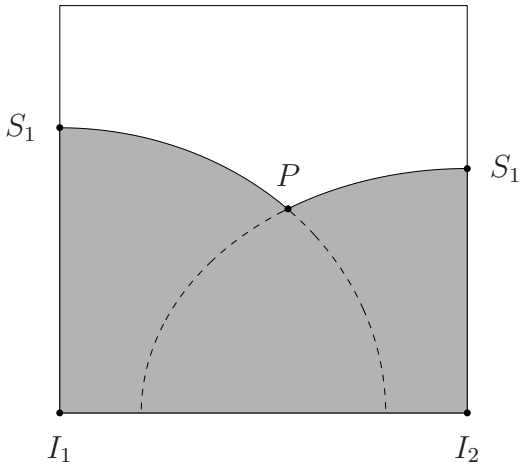


Abbildung 6.3: Bezeichnungen in einer Randzelle

- Schneidet keine der erzeugten Kanten die Randkurve in einem anderen Punkt als den zur Konstruktion verwendeten Schnittpunkten, so ist die Behandlung des Punktes P abgeschlossen.
- Schneidet eine Kante zwischen P und mindestens einem I_k die Randkurve, werden alle solche Kanten verworfen.
- Schneidet eine Kante zwischen P und einem S_1 oder S_2 die Randkurve, so wird diese Kante verworfen und der Schnittpunkt in die Triangulierung mit einbezogen.

Die folgenden Schritte werden durchgeführt, falls kein Extrempunkt oder Schnittpunkt in der Randzelle liegt oder wenn die Behandlung des entsprechenden Punktes abgeschlossen ist.

- Erzeuge eine Kante zwischen S_1 und S_2. Unter Umständen kann hier eine Kante erzeugt werden, die ein Dreieck schliesst, das vollständig ausserhalb des Gebietes liegt, dies muss überprüft werden.
- Schneidet diese Kante die Kurve, so wird die Kante verworfen und neue Kanten zwischen den Ausgangspunkten S_1 und S_2 und dem Schnittpunkt mit der Randkurve werden erzeugt.
- Nun werden sukzessive von S_1, S_2 und dem eventuell im vorigen Schritt entstandenen Schnittpunkt Kanten zu den inneren Zellecken I_k erzeugt, die keine bereits vorhandene Kante schneiden und eine Kante vom Schnittpunkt aus dem vorigen Schritt zum Punkt P. Bei diesem Schritt können Dreiecke entstehen, die vollständig außerhalb von D liegen, dies muss überprüft werden und die Dreiecke bzw die entsprechende Kante verworfen werden.
- Als letzter Schritt werden noch Kanten zwischen benachbarten I_k erzeugt.

Aus den so konstruierten Dreieckskanten können nun alle Dreiecke in dieser Randzelle zusammengesetzt werden. Dies geschieht nach dem folgenden Muster:

- Man sucht für die erste Kante alle Kanten mit dem gleichen Startpunkt.
- Für jede Kombination von gefundenen Kanten sucht man eine dritte Kante, so dass die drei Kanten ein Dreieck bilden.
- Sind alle Kombinationen für diese Ausgangskante erledigt, wird diese Kante als bearbeitet markiert und mit der nächsten Kante fortgefahren.
- Wenn alle Kanten bearbeitet sind, werden doppelt vorkommende Dreiecke aus der Liste der Dreiecke entfernt. Damit ist die Triangulierung der Randzelle abgeschlossen.

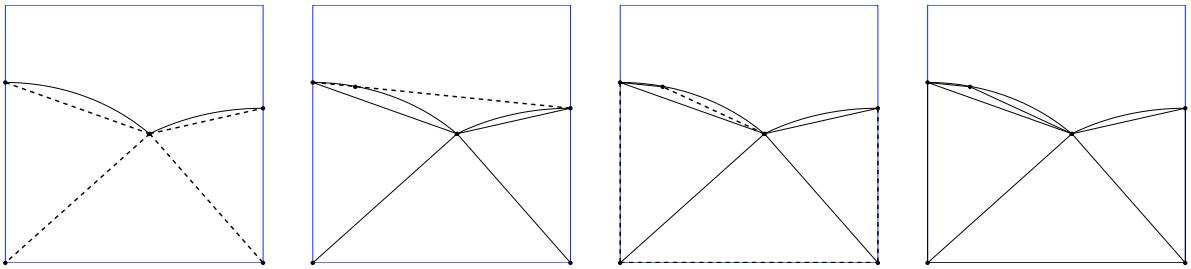


Abbildung 6.4: Beispieldurchlauf des Triangulierungsalgorithmus für eine Randzelle

In Abbildung 6.4 ist ein Beispieldurchlauf des Triangulierungsalgorithmus abgebildet. Die Kanten, die im entsprechenden Schritt erzeugt werden sind gestrichelt dargestellt, auch wenn sie evtl im nächsten Schritt verworfen werden. Im letzten Bild ist dann die vollständige Triangulierung zu sehen.
Eine Möglichkeit zur gesonderten Behandlung von Randzellen, die von zwei (krummlinigen) Objekten geschnitten werden, ist es, die beiden Objekte innerhalb der Zelle nacheinander einzeln zu triangulieren. Es wird als erstes das Innere des ersten Objektes, unter Berücksichtigung aller für beide Triangulierungen notwendigen Punkte, vollständig trianguliert und anschließend das Innere des zweiten Objektes unter Berücksichtigung derjenigen Kanten, die im Schnitt der beiden Objekte liegen. Die überflüssigen Dreiecke, die außerhalb des kombinierten Objektes liegen, können danach verworfen werden, und es bleibt eine vollständige Triangulierung des Inneren des kombinierten Objektes übrig.

6.1.3 Ansatz mit Graphentheorie

Eine weitere Möglichkeit, eine Triangulierung des Inneren der Randzellen zu erhalten, ist, die zur Triangulierung notwendigen Punkte als Knoten eines Graphen zu sehen.
Für Zellen, die keinen Extrempunkt oder Objektschnittpunkt P enthalten lässt sich eine einfache Möglichkeit der Triangulierung angeben, die anschließend leicht erweitert werden kann für Zellen, die einen Punkt P enthalten.

- Hierzu werden die Eckpunkte der Zelle I_i und die Schnittpunkte S_j des Randes ∂D mit den Zellkanten angefangen mit einer inneren Ecke durchnummeriert und als Knoten des Graphen definiert. Existiert keine innere Ecke, so wird ein Kantenschnittpunkt als erster Knoten verwendet, so dass die Verbindung zum zweiten Knoten innerhalb des Objektes liegt.

- Ausgehend vom ersten Knoten werden Kanten zum nächsten Knoten erzeugt. An jedem Schnittpunkt wird entschieden, ob die folgende Kante im Inneren liegt und erzeugt wird oder außen liegt und keine Kante erzeugt wird. An den Kantenschnittpunkten S_j erfolgt ein Wechsel von innen nach außen oder umgekehrt, an den Ecken I_i bleibt es gleich. Wurde also im vorigen Schritt eine Kante zum entsprechenden Punkt erzeugt, so wird bei S_j keine Kante erzeugt, da diese außerhalb liegt, an einem Eckpunkt wird jedoch wieder eine Kante erzeugt. Dies erspart eine Überprüfung jeder einzelnen Kante, ob sie gegebenenfalls vollständig außen liegt.

- So erhält man eine oder mehrere Zusammenhangskomponenten des Graphen, die Kanten enthalten.

- Als nächstes werden innerhalb jeder dieser Zusammenhangskomponenten so lange Kanten erzeugt, die keine vorhandenen Kanten schneiden, bis keine weitere Kante hinzugefügt werden kann.

- Als letztes werden nach dem gleichen Prinzip Kanten zwischen den einzelnen Zusammenhangskomponenten erzeugt, sofern die hierbei erzeugten Kanten nicht den Gebietsrand schneiden. Dies kommt beispielsweise dann vor, wenn zwei disjunkte Objektteile innerhalb einer Zelle existieren, in diesem Fall müssen für eine vollständige Triangulierung keine weiteren Kanten erzeugt werden.

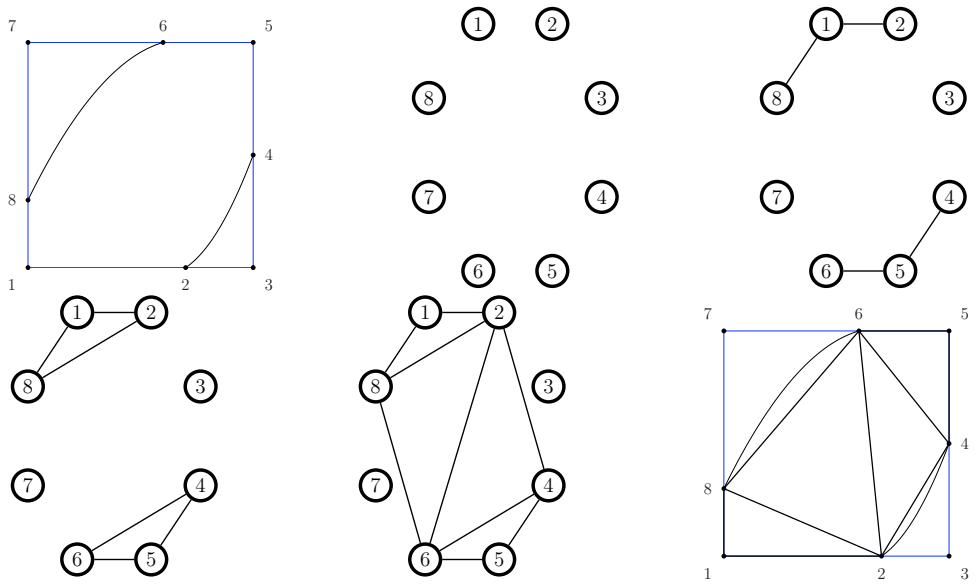


Abbildung 6.5: Links oben: Die zu triangulierende Zelle. Danach der Graph in den verschiedenen Schritten des Algorithmus und schließlich rechts unten die daraus resultierende Triangulierung der Zelle.

Ein Beispiel für einen Durchlauf des Algorithmus ist in Abbildung 6.5 dargestellt. Links oben ist die Randzelle mit der Randkurve und den durchnummerierten Knoten zu sehen.

Die folgenden vier Bilder zeigen den Graphen nach den einzelnen Schritten des Algorithmus und das Bild rechts unten zeigt die daraus resultierende Triangulierung.
Für Zellen, die einen Extrempunkt enthalten lässt sich die Triangulierung ähnlich erstellen.

- Zu Beginn werden wieder, wie oben, Kanten auf dem Zellrand erzeugt.
- Danach wieder die Knoten der einzelnen Zusammenhangskomponenten miteinander mit Kanten verbunden, so dass vollverknüpfte Zusammenhangskomponenten entstehen.
- Schneidet eine dieser Kanten die Kurve, so werden alle diese Kanten verworfen und zwei neue Kanten erzeugt, vom Ausgangspunkt zum Schnittpunkt und vom Schnittpunkt zum Endpunkt, und der Schnittpunkt wird zum Graphen und der Zusammenhangskomponente hinzugefügt. Danach werden nochein mal Kanten in dieser Zusammenhangskomponente erzeugt, so dass Sie dann vollverknüpft ist.
- Als letztes werden alle Knoten mit dem Punkt P verbunden, sofern die Kante keine andere Kante schneidet.
- Schneidet eine dieser Kanten die Kurve, so wird auch diese Kante verworfen und statt derer wieder zwei Kanten hinzugefügt. Vom Schnittpunkt aus werden nun zusätzlich Kanten zu den Zusammenhangskomponenten erzeugt, sofern diese Kanten keine weiteren Kanten schneiden.
- In den letzten beiden Schritten können Dreiecke entstehen, die vollständig ausserhalb des Gebietes liegen, beispielsweise bei zwei disjunkten Teilgebieten. Die hier entstandenen Dreiecke müssen deshalb überprüft werden.

Das Ergebnis des Algorithmus mit einem Schnittpunkt P in der Zelle ist in Abbildung 6.6 dargestellt, analog zur vorigen Abbildung. Hierbei ist wieder darauf zu achten, dass eine Kante die Randkurve schneidet und somit der zusätzliche Punkt 7 entsteht.

6.1.4 Delaunay-Triangulierung

Bei diesen Triangulierungsalgorithmen wird bisher die Form der entstehenden Dreiecke noch nicht optimiert. Eine mögliche Forderung wäre beispielsweise mindestens ein gewisses Verhältnis zwischen der kürzesten und der längsten Dreieckskante, um „unschöne“Dreiecke zu vermeiden.
Eine geometrisch sehr gute Triangulierung erhält man durch eine Delaunay-Triangulierung, bei der der kleinste Innenwinkel der Dreiecke der Triangulierung maximiert wird. Im Vergleich zu der gerade beschriebenen Methode ist das Erstellen einer Delaunay Triangulierung zeitaufwändiger und sofern keine weiteren Anforderungen an die Dreiecke gestellt werden, ist es nicht notwendig eine Delaunay Triangulierung, nach B. Delaunay [8], zu verwenden.

Definition 6.4 *Eine Delaunay Triangulierung einer konvexen Punktmenge ist eine Triangulierung, bei der der Umkreis jedes Dreiecks keine Ecken anderer Dreiecke enthält. Liegen auch auf dem Kreisrand keine weiteren Eckpunkte, so ist die Delaunay Triangulierung eindeutig.*

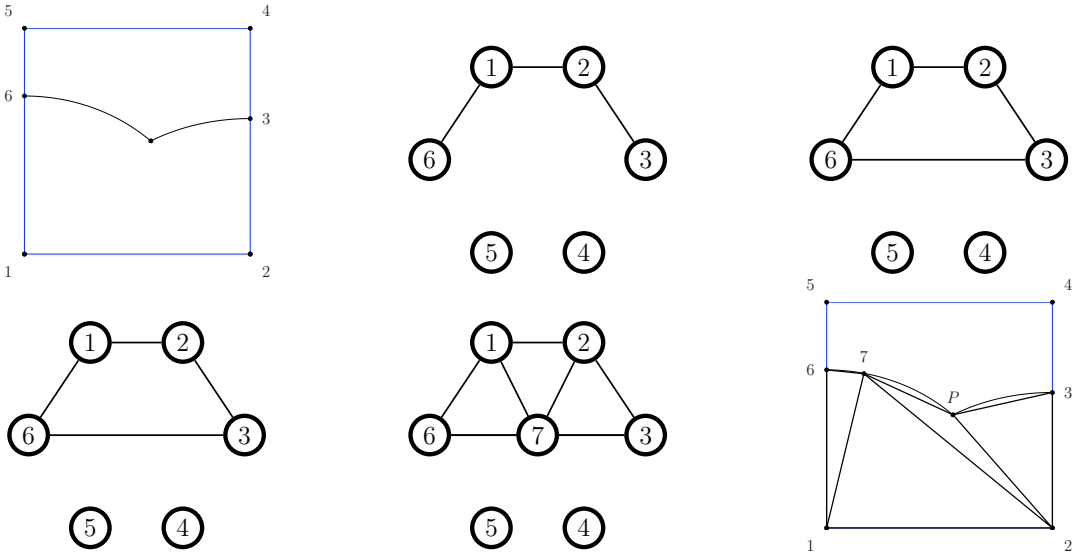


Abbildung 6.6: Links oben: Die zu triangulierende Zelle mit einem Schnittpunkt im Inneren. Danach wieder Bilder des Graphen und als letztes die resultierende Triangulierung.

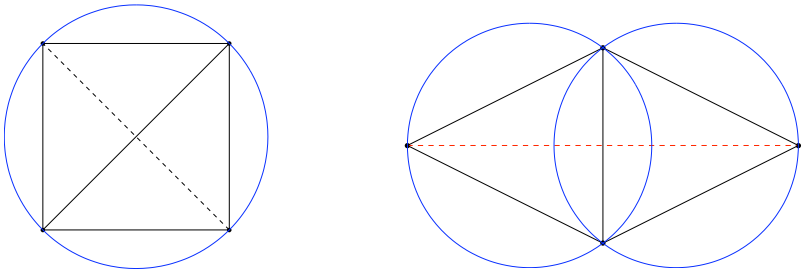


Abbildung 6.7: Beispiele für eine Delaunay-Triangulierung

In Abbildung 6.7 sind zwei Beispiele für Delaunay-Dreiecke zu sehen. Die Dreiecke auf der rechten Seite sind eindeutig, da die Verwendung der gestrichelten Kante nicht erlaubt ist, sie verstößt gegen das Umkreis-Kriterium. Auf der linken Seite ist die Triangulierung nicht eindeutig, auch die gestrichelte Kante ist zulässig.
Die Vereinigung der Dreiecke einer Delaunay Triangulierung bildet die konvexe Hülle der Triangulierungspunkte, allerdings wurde zu Anfang des Kapitels schon erwähnt, dass bei der Triangulierung der Randzellen nicht immer konvexe Punktmengen trianguliert werden, wie beispielsweise in Abbildung 6.3 zu sehen ist. Eine Möglichkeit dies zu umgehen ist es, die ganze Randzelle zu triangulieren und anschließend diejenigen Dreiecke, die außerhalb von D liegen zu entfernen. Dies ist jedoch nicht zu empfehlen, da nicht sichergestellt ist, dass die Dreiecke der Triangulierung entweder komplett innerhalb oder komplett außerhalb von D liegen.
Eine bessere Vorgehensweise ist es die nichtkonvexe Punktmenge in konvexe Teilmengen zu unterteilen und diese dann einzeln zu triangulieren, was im Prinzip die Weiterführung dessen ist, dass bei zwei Objekten in einer Zelle beide Objekte nacheinander trianguliert

werden. Das Einteilen in konvexe Punktmengen kann beispielsweise dadurch erreicht werden, dass wie in Abbildung 6.3 zu sehen, die beiden Randkurven ins Innere von D verlängert werden und zwei weitere Kantenschnittpunkte S_3 und S_4 berechnet werden. Die Punktmenge lässt sich nun durch das Verbinden von P mit den S_i in konvexe Punktmengen unterteilen. Es ist bei diesem Vorgehen auch nicht nötig darauf zu achten, dass eventuell auf einer dieser Strecken hinzugefügte Punkte bei beiden Triangulierungen verwendet werden, da die Dreiecke nur für die numerische Integration verwendet werden und keine weiteren Eigenschaften der Triangulierung gefordert werden müssen.

6.2 Tetraedisierung

Zur Tetraedisierung des Teils einer dreidimensionalen Randzelle, der innerhalb des Gebietes liegt, wird hier ein einfacher Algorithmus angegeben. Im Gegensatz zum zweidimensionalen Fall gibt es deutlich mehr mögliche Fälle, wie die Randfläche durch die Randzelle verlaufen kann, selbst wenn man sich wie hier auf maximal drei Objekte pro Randzelle beschränkt.

6.2.1 Fallunterscheidung

Der Spezialfall von maximal zwei Halbräumen, die eine Randzelle schneiden soll hier näher untersucht werden, um die Vielzahl von Möglichkeiten zu verdeutlichen. Dabei ist für die Überlegungen zu beachten, dass durch Vertauschen von Innen und Außen nicht unbedingt wieder die gleiche Anzahl neuer Fälle entsteht, da somit entweder zwei disjunkte Gebiete entstehen können, die einzeln behandelt werden, oder aber auch ein Gebiet ein anderes enthalten kann, wodurch nur noch ein Objekt in der Zelle behandelt wird. Es wird hier also, im Gegensatz zum Zweidimensionalen, das Innere und Äußere der Halbräume beachtet und mit berücksichtigt. Somit entstehen auch durch Vertauschen von Innen und Außen keine weiteren Fälle, sie sind alle aufgeführt.

7 Fälle Die Fälle von einer einzigen Ebene, die die Zelle schneidet sind trivial und anhand der Anzahl der im Inneren liegenden Ecken bis auf Symmetrie zu unterscheiden.

16 Fälle Zwei Ebenen, die sich innerhalb der Zelle nicht schneiden verlaufen analog zu den sieben Fällen einer Ebene, wobei das Gebiet immer zwischen den beiden Ebenen liegt, da sonst disjunkte Gebiete entstehen würden, die einzeln wie oben behandelt werden können. Zwei Fälle ohne eine Zellecke zwischen den beiden Ebenen sind möglich, die beiden Ebenen schneiden die gleichen Zellflächen und diese sind entweder gegenüberliegend oder benachbart.

Eine weitere Möglichkeit ist, dass genau eine Zellecke im Inneren des Gebietes liegt.

Bei zwei Ecken zwischen den Ebenen gibt es drei Möglichkeiten, die beiden Ecken sind durch eine Kante verbunden, sie liegen auf der gleichen Zellfläche ohne Kante zwischen ihnen oder sie liegen sich diagonal gegenüber.

Änhlich ist es bei drei Ecken, entweder ist eine Ecke mit den beiden anderen Ecken durch eine Kante verbunden, nur mit einer und es existiert keine Kante zur dritten Ecke oder es gibt keine Kanten zwischen den drei Ecken.

Bei vier Ecken gibt es die Möglichkeiten, dass alle Ecken auf einer Zellfläche liegen, drei Ecken liegen auf einer Zellfläche und die vierte Ecke ist durch eine Kante mit einer der ersten verbunden oder es existiert keine solche Kante. Jeweils zwei Kanten sind mit einer Kante verbunden und diese Kanten liegen sich diagonal gegenüber.

Fünf Ecken im Inneren ergeben sich drei Möglichkeiten. Die einfachste sind vier Ecken, die auf einer Zellebene liegen und eine weitere Ecke. Bei drei Ecken auf einer Ebene gibt es die Möglichkeit, dass die beiden weiteren Ecken auf der gegenüberliegenden Zellebene durch eine Kante verbunden sind oder es keine solche Kante auf dieser Fläche gibt.

Bei sechs inneren Ecken lassen sich die drei Fälle am einfachsten anhand der äußeren Ecken identifizieren. Diese sind entweder durch eine Kante verbunden, liegen in der gleichen Zellebene oder diagonal gegenüber. Sieben und acht innere Ecken sind nicht möglich, ohne dass sich die beiden Ebenen in der Zelle schneiden.

17 Fälle Zwei Ebenen, die sich in der Zelle schneiden, schneiden sich immer in einer Geraden. Auch hier erfolgt die Unterscheidung der Fälle wieder anhand der inneren Zellecken. Schließen die beiden Halbräume keine Ecken ein, so gibt es die beiden Möglichkeiten, dass die Schnittgerade zwei gegenüberliegende Zellflächen schneidet oder zwei benachbarte. Bei einer inneren Ecke gibt es die beiden gleichen Fälle.

Zwei innere Ecken können entweder durch eine Kante verbunden sein oder sich auf einer Randfläche diagonal gegenüber liegen, woraus vier Fälle entstehen, wenn die Schnittgerade analog zu oben verläuft.

Bei drei und vier inneren Ecken gibt es jeweils zwei Fälle, die sich wieder nur durch den Verlauf der Schnittgerade unterscheiden. Die inneren Ecken liegen dabei in beiden Fällen auf einer gemeinsamen Zellfläche.

Für fünf innere Ecken gibt es wieder zwei Möglichkeiten. Vier Ecken liegen auf einer Zellebene und die fünfte ist durch eine Kante mit einer der vier Ecken verbunden. Analog bei sechs inneren Ecken.

Eine letzte Möglichkeit gibt es für sieben innere Ecken, hierbei ist es nicht möglich, dass die Schnittgerade zwei gegenüberliegende Zellflächen schneidet.

Dies macht deutlich, dass es schon bei der Verwendung von nur zwei stückweise linearen Objekten eine große Zahl von Möglichkeiten, nämlich 40, gibt, die noch einmal erhöht wird, wenn man Quadriken als weitere Objektklasse verwendet und drei Objekte in einer Zelle liegen, was nicht auszuschließen ist. Aus diesem Grund muss eine Möglichkeit gefunden werden, diese Fälle zu behandeln oder sie in wenige einfachere Fälle zu unterteilen. Hierzu wird im Folgenden ein Algorithmus angegeben, mit dem sich schnell einfache Fälle behandeln lassen.

6.2.2 Tetraedisierung der Randzellen

Zur besseren Übersichtlichkeit des Algorithmus werden hier analog zum zweidimensionalen Fall einige Bezeichnungen eingeführt, die in Abbildung 6.8 zu sehen sind. Die im Inneren des Gebietes liegenden Zellecken werden mit I_ℓ bezeichnet mit $\ell = 1, \ldots, m$ mit $m =$Anzahl

der inneren Ecken. Die Schnittpunkte der Ausgangsobjekte mit den Zellkanten werden mit S_k bezeichnet, wobei $k = 1, \ldots, n$ mit n =Anzahl der Schnittpunkte. Die Schnittpunkte zweier Ausgangsobjekte auf den Randebenen der Zelle werden mit E_m bezeichnet.

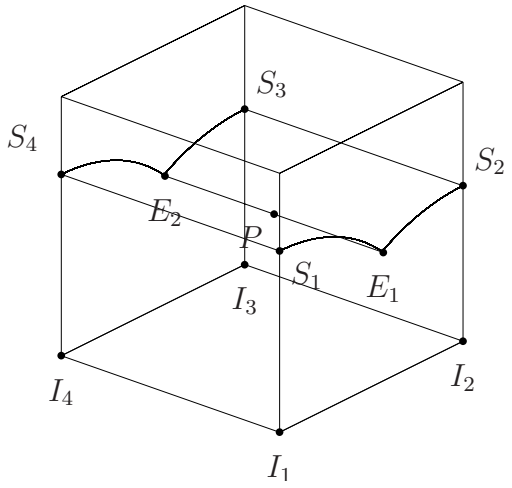


Abbildung 6.8: Bezeichnungen in einer Randzelle

Aufgrund der Vielzahl von auftretenden Fällen für Randzellenschnitte kann es einfacher sein, wie schon im Zweidimensionalen vorgeschlagen, die verschiedenen Objekte einer Zelle nacheinander zu tetraedisieren unter Berücksichtigung aller bekannten Schnittpunkte und Schnittkanten, beispielsweise der Schnittkante einer Ebene mit einer Randfläche des Zellwürfels.
Zur Tetraedisierung wird ein einfacher Algorithmus formuliert, der aus zwei Schritten besteht, als Erstes wird ein zusätzlicher Punkt im Inneren der Zelle gewählt, der die Spitze aller entstehenden Tetraeder werden wird, und im zweiten Schritt werden alle Randebenen der Zelle einfach und schnell trianguliert. Liegen zwei oder drei disjunkte Objektteile in einer Zelle, so werden diese als einzelne Objekte behandelt und nach den einfachen Fällen für einzelne Objekte abgearbeitet und nicht als weiteren Spezialfall für zwei oder drei Objekte in einer Randzelle.

Berechnung der Tetraederspitze

Der zusätzliche Punkt P, der später als Spitze aller Tetraeder gewählt wird, lässt sich auf verschiedene Weise wählen. Hier wird für die verschiedenen Fälle eine Möglichkeit dargestellt, diesen Punkt zu wählen.
Bei einem Objekt, das die Randzelle schneidet, ist im Spezialfall von einer inneren Zellecke gar kein Punkt P zu wählen, denn durch die innere Ecke und die drei Kantenschnittpunkte $S_{1,2,3}$ ist schon ein Tetraeder gegeben und die Tetraedisierung der Zelle ist abgeschlossen. Existiert mehr als eine innere Zellecke, so wird $P = (P_x, P_y, P_z)$ als Mittel der Schnittpunkte des Gebietes mit den Zellkanten berechnet

$$P_x = \frac{1}{i} \cdot \sum_{k=1}^{i} x_k \,, \qquad P_y = \frac{1}{i} \cdot \sum_{k=1}^{i} y_k \,, \qquad P_z = \frac{1}{i} \cdot \sum_{k=1}^{i} z_k \,,$$

wobei $S_k = (x_k, y_k, z_k), k = 1, \ldots, i$ die Schnittpunkte des Gebietes mit den Zellkanten sind und i die Anzahl der Schnittpunkte.
Schneiden sich zwei Objekte in der Zelle und diese beiden Objekte haben auf genau zwei Randebenen der Zelle einen gemeinsamen Punkt, so wird P als der Mittelpunkt der Verbindungsstrecke dieser beiden Punkte gewählt. Haben die beiden Objekte einen oder mehr als zwei Punkte gemeinsam auf den Randebenen der Zelle, so wird P wie oben berechnet. Bei drei Objekten in der Randzelle wird, sofern vorhanden, der gemeinsame Schnittpunkt aller drei Objekte als P gewählt. Schneiden sich die drei Objekte in zwei Geraden, so wird P wieder analog zum Fall mit einem Objekt berechnet. Eine weitere Möglichkeit wäre hier, die Mitte zwischen den beiden Mittelpunkten der Schnittstrecken zu berechnen und diese als P zu verwenden.
Es ist nicht gewährleistet, dass der Punkt P immer im Inneren oder auf dem Rand des Objektes liegt. Allerdings ist die Gitterweite h klein genug gewählt, dass kein großer Abstand zum Rand entstehen kann, da selbst Quadriken in den meisten Fällen innerhalb einer Zelle fast linear sind. Jedoch lässt sich prüfen, ob der Punkt im Inneren oder auf dem Rand liegt und man kann ihn, sollte das nicht der Fall sein, entsprechend verschieben, beispielsweise in Normalenrichtung eines Punktes auf dem Rand, der nah an P liegt, bis er die gewünschte Position hat.
Ist P berechnet, so wird wie folgt mit der Tetraedisierung fortgefahren.

Triangulierung der Randebenen

Die Triangulierung der Randebenen erfolgt äquivalent zum zweidimensionalen Fall, was man auch bei der Implementierung verwenden kann, da die Zellebenen parallel zu den Koordinatenebenen sind und somit eine Koordinate der Punkte fest bleibt. Für die Triangulierungen werden die Punkte S_k, I_ℓ und E_m, die in der jeweiligen Randebene liegen, benötigt. Die Triangulierung analog zum Zweidimensionalen ist mit den angegebenen Methoden möglich, da die Schnittkurve jeder Quadrik mit einer Ebene ein Kegelschnitt ist.
Sämtliche hier erhaltenen Dreiecke werden mit P zu einem Tetraeder ergänzt.

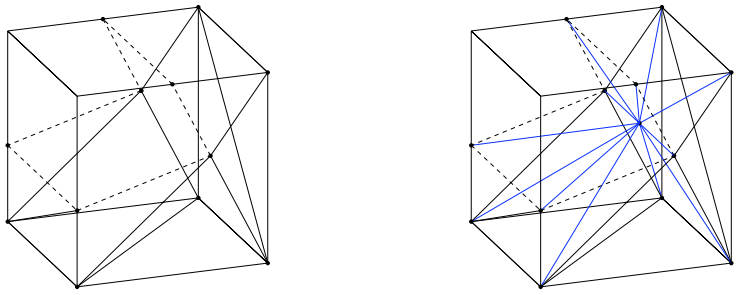


Abbildung 6.9: Triangulierung der Randflächen (links) und die resultierende Tetraedisierung (rechts)

In Abbildung 6.9 ist dies für eine Randzelle dargestellt. Links ist die Triangulierung der Randflächen zu sehen, die Schnittkanten der beiden Ebenen mit den Randflächen der Zelle sind gestrichelt eingezeichnet. Die fertige Tetraedisierung ist rechts dargestellt.

Zwei Spezialfälle eines Randzellenschnittes sind ein oder mehrere Objekte, die in eine Zelle hineinreichen oder durch eine Zelle hindurch gehen, ohne eine Zellkante zu schneiden. Im ersten Fall liegt immer ein Extrempunkt eines oder ein Schnittpunkt mehrerer Objekte in der Zelle, da h nach Voraussetzung klein genug gewählt ist. Die Zelle wird als Randzelle erkannt, da der Extrem- oder Schnittpunkt auf dem Rand des Objekts liegt. Der zweite Fall wird durch den Schnitt der Gitterebenen mit dem zusammengesetzten Objekt erkannt und analog zu den anderen Randzellen behandelt. Ist h klein genug gewählt und sind die Gitterecken geeignet gesetzt tritt dieser Fall sehr selten auf.

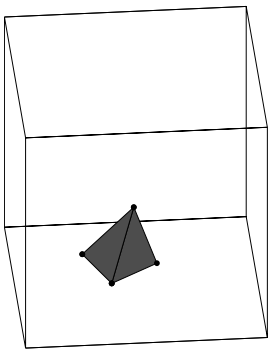


Abbildung 6.10: Spezialfall eines Randzellenschnittes ohne Schnittpunkte auf den Kanten

Bei diesem Verfahren zur Tetraedisierung der Randzellen wurden keine Anforderungen an die Gestalt der Tetraeder gestellt. Im zweidimensionalen Fall wurde die Verwendung einer Delaunay Triangulierung empfohlen, um schönere Dreiecke zu erhalten. Analog ist dies auch mit Tetraedern möglich.

Definition 6.5 *Eine Delaunay-Tetraedisierung einer konvexen Punktwolke im $\mathbb{R}^3$ ist eine Tetraedisierung, bei der jeder Tetraeder der Umkugel-Bedingung genügt, also die Umkugel jedes Tetraeders keine weiteren Tetraederecken enthält. Die Tetraedisierung ist eindeutig, wenn auf keiner Umkugel mehr als vier Eckpunkte liegen.*

Eine Unterteilung der nichtkonvexen Punktwolke innerhalb einer Randzelle in konvexe Teilmengen ist auch hier wieder notwendig, wird aber analog zum Zweidimensionalen Fall durchgeführt.
Eine Delaunay-Tetraedisierung erbt jedoch nicht alle positiven Eigenschaften der Delaunay Triangulierung, so ist es nicht möglich, das Maximieren der kleinsten Winkel vom Zweidimensionalen in mehr Dimensionen zu verallgemeinern. Weiterführende Informationen darüber und über Delaunay Triangulierungen und Tetraedisierungen sind der Literatur zu entnehmen, beispielsweise [27].
Wieder analog zum Zweidimensionalen erfordert dieses Vorgehen einen höheren Zeitaufwand als der vorher vorgestellte Algorithmus.

7 Integration

Für das Aufstellen des Ritz-Galerkin-Systems ist es notwendig, Integrale auf den Gitterzellen über D zu berechnen. Zur Berechnung der Integrale wird auf Methoden der numerischen Integration zurückgegriffen, da in den meisten Fällen keine analytischen Lösungen verfügbar sind. Ziel ist eine Approximation des Integrals mittels einer endlichen Summe

$$\int_a^b f(x)dx \approx Q_N(f) = \sum_{i=1}^{N} w_i f(x_i) \tag{7.1}$$

mit Stützstellen x_i und Gewichten w_i. $Q_N(f)$ nennt man Quadraturformel. Werden N äquidistante Stützstellen x_i verwendet, so spricht man von Newton-Cotes Formeln. Die Gewichte w_i sind bestimmbar, so dass Polynome vom Grad $N-1$ exakt integrierbar sind. Beispiele für Newton-Cotes Formeln sind die Trapez-Regel und die Simpson-Regel. Es ist wünschenswert, Quadraturformeln zu finden, mit denen Polynome von möglichst hohem Grad exakt integriert werden können. Deshalb ist es naheliegend, die Stützstellen nicht als äquidistant festzulegen, sondern diese so zu bestimmen, dass $2N$ Parameter bestimmbar sind. Formeln dieser Form nennt man Gauß-Quadratur Formeln. Diese werden hier verwendet, da glatte Funktionen über kleinen Mengen integriert werden, wo diese zu den effizientesten Approximationen führen [16]. Mit den zu bestimmenden Gewichten und Stützstellen ergeben sich $2N$ Unbekannte, die so gewählt beziehungsweise berechnet werden können, dass eine exakte Integration von Polynomen vom Grad $2N-1$ erreicht werden kann.
Die Parameter der ersten fünf Formeln sind in Tabelle 7.1 für das Intervall $[0,1]$ zu sehen.

N	x_i	w_i
1	$\frac{1}{2}$	2
2	$\frac{1}{2} \pm \frac{1}{6}\sqrt{3}$	$\frac{1}{2}$
3	$\frac{1}{2}$	$\frac{4}{9}$
	$\frac{1}{2} \pm \frac{1}{10}\sqrt{15}$	$\frac{5}{18}$
4	$\frac{1}{2} \pm \frac{1}{70}\sqrt{525-70\sqrt{30}}$	$\frac{1}{4} + \frac{1}{72}\sqrt{30}$
	$\frac{1}{2} \pm \frac{1}{70}\sqrt{525+70\sqrt{30}}$	$\frac{1}{4} - \frac{1}{72}\sqrt{30}$
5	$\frac{1}{2}$	$\frac{64}{225}$
	$\frac{1}{2} \pm \frac{1}{42}\sqrt{245-14\sqrt{70}}$	$\frac{161}{900} + \frac{13}{1800}\sqrt{70}$
	$\frac{1}{2} \pm \frac{1}{42}\sqrt{245+14\sqrt{70}}$	$\frac{161}{900} - \frac{13}{1800}\sqrt{70}$

Tabelle 7.1: Stützpunkte und Gewichte für die Gauss-Quadratur

Meist werden Gauß-Parameter für die Standardintervalle $[-1,1]$ oder $[0,1]$ angegeben. Die Verwendung eines dieser Intervalle bedeutet keine Einschränkung, da sich die Stützstellen und Gewichte leicht mittels

$$x = \frac{1}{2}(b-a)\tilde{x} + \frac{1}{2}(b+a) \quad \text{und} \quad w = \frac{b-a}{2}\tilde{w}$$

von einem allgemeinen Intervall $[a,b]$ auf das Standardintervall $[-1,1]$ transformieren lassen, wodurch für das Integral

$$\int_a^b f(x)dx = \frac{1}{2}(b-a)\int_{-1}^{1} f(\tilde{x})d\tilde{x}$$

gilt. Analog erhält man mit der Transformation

$$x = (b-a)\tilde{x} + a \quad \text{und} \quad w = (b-a)\tilde{w}$$

das Integral über dem Einheitsintervall $[0,1]$

$$\int_a^b f(x)dx = (b-a)\int_{0}^{1} f(\tilde{x})d\tilde{x}.$$

Als Stützstellen x_i der N-Punkt-Formel eignen sich die Nullstellen des Legendre-Polynoms vom Grad N [15] und als Gewichte w_i die Integrale der zugehörigen Lagrange-Polynome. Für die Integration über D ist die Unterscheidung der Zelltypen wichtig, denn über die Randzellen zu integrieren ist aufwändiger als die Integration über die inneren Zellen. Der einfachere Fall wird zunächst betrachtet, bevor die Integration über den Randzellen im nächsten Abschnitt für zweidimensionale und dreidimensionale Randzellen behandelt wird. Analog zum eindimensionalen Fall lassen sich auch allgemeine zwei- und dreidimensionale Rechtecks- beziehungsweise quaderförmige Gebiete auf Standardgebiete wie beispielsweise das Einheitsquadrat oder den Einheitswürfel transformieren, indem man für jede Variable die Transformation durchführt. Analog lautet für das Doppelintegral über dem Rechteck $[a,b]\times[c,d]$

$$\int_a^b \int_c^d f(x_1,x_2)dx_2dx_1 \approx \sum_{i=1}^{N}\sum_{j=1}^{N} w_{1,i}w_{2,j}f(x_i,x_j)$$

die zweidimensionale Transformation auf das Einheitsquadrat $[0,1]^2$

$$(b-a)(d-c)\int_0^1 \int_0^1 f(\tilde{x}_1,\tilde{x}_2)d\tilde{x}_1d\tilde{x}_2 \approx \sum_{i=1}^{N}\sum_{j=1}^{N} \tilde{w}_{1,i}\tilde{w}_{2,j}f(\tilde{x}_i,\tilde{x}_j),$$

wobei sich hier ebenfalls wieder die Gauß-Parameter für eindimensionale Integration in jede Summe einsetzen lassen. Für eine innere Gitterzelle $Q = \ell h + [0,1]^m\, h$, die ∂D also nicht schneidet, lässt sich der oben behandelte eindimensionale Fall also wie beschrieben

mittels Tensorprodukt in zwei und analog auch in m Dimensionen verallgemeinern. Für $m = 3$ lautet diese verallgemeinerte Gauß-Formel beispielsweise:

$$\int_D f \approx h^3 \sum_{i,j,k} w_i w_j w_k \; f(x_i, x_j, x_k) \,,$$

mit, ebenfalls analog zum eindimensionalen Fall, auf das Intervall Q transformierten Gauß-Punkten und Gewichten. Ebenso wie im Univariaten ist diese Integration für Polynome bis zum Grad $2N - 1$ exakt. Die Tensorproduktformel kann direkt angewendet werden, da die quadratische Zelle vollständig im Inneren des Gebietes D liegt. Schneidet ∂D jedoch die Gitterzelle, kann die Tensorproduktformel nicht direkt angewendet werden, und das Integrationsgebiet ist entweder zu transformieren [16] oder das Integrationsgebiet wird in Gebiete unterteilt, über denen direkt integriert werden kann.

7.1 Integration über Dreiecken und Tetraedern

Bei Randzellen ist es nicht ratsam, die zu integrierende Funktion ausserhalb von D auf Null zu setzen, denn dadurch entsteht ein deutlicher Genauigkeitsverlust. Um Standardformeln anzuwenden, muss man die Schnittmenge der Zelle Q mit dem Gebiet D unterteilen. Eine Möglichkeit hierfür im Zweidimensionalen ist es, die Zelle in parallele Streifen durch Extrempunkte, Kantenschnittpunkte und Ecken in ∂D zu unterteilen [16].
Mit den in dieser Arbeit bereits vorgestellten Triangulierungs- und Tetraedisierungsalgorithmen wird der im Inneren von D liegende Bereich $Q \cap D$ in Dreiecke beziehungsweise Tetraeder unterteilt, über welchen anschliessend integriert wird. Integrale sind additiv, deshalb gilt

$$\int_{D \cap Q} f(x) \approx \int_{T_1} f(x) + \cdots + \int_{T_k} f(x) \,,$$

wobei $T_1, \ldots, T_k$ die Dreiecke beziehungsweise Tetraeder sind, die durch den entsprechenden Unterteilungsalgorithmus für die Unterteilung von $D \cap Q$ erhalten wurden. Durch $\approx$ wird angedeutet, dass durch die Unterteilung des Gebietes der Rand ∂D stückweise linear approximiert wird und dadurch ein Approximations-Fehler entsteht, der davon abhängt, wie genau die Kanten beziehungsweise Flächen den Rand approximieren, was von der Gestalt des Randes, der Unterteilung und der Gitterweite h abhängt.
Bei der Triangulierung beziehungsweise Tetraedisierung der Randzellen werden beliebige Dreiecke und Tetraeder erzeugt. Die Parameter für Gauß-Quadratur Formeln sind jedoch nur für Standardgebiete angegeben. Es ist aber nicht notwendig, für jedes Dreieck und jeden Tetraeder die Gauß-Parameter einzeln herzuleiten und zu speichern. Es ist effizienter, die Parameter für ein Standardgebiet zu speichern und eine Transformation des Standardgebietes auf das jeweilige Dreieck beziehungsweise den jeweiligen Tetraeder zu berechnen.
Allgemein kann man einen zweidimensionalen Bereich B_1 auf einen anderen Bereich B_2 wie folgt abbilden.

Lemma 7.1 *Sei $\varphi : B_1 \to B_2$ mit $B_1, B_2 \subset \mathbb{R}^2$ eine stetig differenzierbare bijektive Abbil-*

dung mit der Jacobi-Matrix

$$J(x,y) = \begin{pmatrix} \frac{\partial \varphi_1}{\partial x}(x,y) & \frac{\partial \varphi_1}{\partial y}(x,y) \\ \frac{\partial \varphi_2}{\partial x}(x,y) & \frac{\partial \varphi_2}{\partial y}(x,y) \end{pmatrix}.$$

Falls $\det J(x,y) \neq 0$ *für alle* $(x,y) \in B_1$*, so gilt*

$$\int_{B_1} f(\varphi(x,y)) |\det J(x,y)| \, dxdy = \int_{B_2} f(\tilde{x},\tilde{y}) \, d\tilde{x}d\tilde{y}.$$

Ist φ *affin, so ergibt sich*

$$\varphi(x,y) = A \begin{pmatrix} x \\ y \end{pmatrix} + b, \quad A \in \mathbb{R}^{2\times 2}, \quad \det(A) \neq 0, \quad b \in R^2.$$

Daraus folgt für das Integral:

$$|\det A| \int_{B_1} f(A \begin{pmatrix} x \\ y \end{pmatrix} + b) \, dxdy = \int_{B_2} f(\tilde{x},\tilde{y}) \, d\tilde{x}d\tilde{y}.$$

Analog erfolgt die Transformation im Dreidimensionalen. Es bietet sich an, als Standardgebiet im $\mathbb{R}^2$ das Einheitsdreieck und im Dreidimensionalen den Einheitstetraeder zu verwenden. Die Transformation des Einheitsdreiecks auf ein beliebiges Dreieck $T = (P_1, P_2, P_3)$ lässt sich unmittelbar angeben:

$$\varphi(x,y) = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

mit

$$A = \begin{pmatrix} P_{2x} - P_{1x} & P_{3x} - P_{1x} \\ P_{2y} - P_{1y} & P_{3y} - P_{1y} \end{pmatrix} \quad \text{und} \quad \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} P_{1x} \\ P_{1y} \end{pmatrix}.$$

Betrachtet man die Lösung, sieht man, dass es nicht notwendig ist, das lineare Gleichungssystem für jedes Dreieck zu lösen, da es mit allgemeinen Eckpunkten P_i einfach gelöst ist. Für die Gauß-Quadratur über dem Dreieck T erhält man somit

$$\begin{aligned}
&\int_0^1 \int_0^{1-x} f(\varphi(x,y)) |\det J(x,y)| dxdy = \\
&\int_0^1 \left(\int_0^{1-x} f(\varphi(x,y)) |\det J(x,y)| dy \right) dx = \\
&\sum_{j=1}^{N} w_j \left(\int_0^{1-\xi} f(\varphi(\xi,y)) |\det J(\xi,y)| dy \right) = \\
&\sum_{j=1}^{N} w_j (1-\xi_j) \sum_{k=1}^{N} w_k f(\varphi(\xi_j, (1-\xi_j)\xi_k)) |\det J(\xi_j, (1-\xi_j)\xi_k)|,
\end{aligned}$$

wobei die ξ_i und ξ_j die Transformierten Gauß-Punkte sind.
Für die Integration können wieder die Parameter aus Tabelle 7.1 verwendet werden, da bei den Integralen die Grenzen dem Einheitsdreieck entsprechend gewählt wurden. Allerdings gibt es auch spezielle Gauß-Parameter für Dreiecke und Tetraeder. In der Literatur werden die Stützstellen für die Gauß-Quadratur über speziellen Gebieten wie Dreiecken und Tetraedern oftmals auch in baryzentrischen Koordinaten (α, β, γ) für Dreiecke beziehungsweise $(\alpha, \beta, \gamma, \delta)$ für Tetraeder angegeben.

Definition 7.2 *Sei $T \subset \mathbb{R}^n$ ein nicht degenerierter n-Simplex mit den $(n+1)$ Ecken a_i. Für jeden Punkt $x \in \mathbb{R}^n$ existieren bezüglich dieses Simplexes eindeutige Zahlen $\lambda_i \in \mathbb{R},\ i = 1, \ldots, n+1$, so dass*

$$\begin{aligned} \sum_{i=1}^{n+1} \lambda_i a_i &= x \\ \sum_{i=1}^{n+1} \lambda_i &= 1 \end{aligned}$$

gilt. Sind die $\lambda_i \geq 0$ so liegt x innerhalb oder auf dem Rand des Simplex.
Das Tripel (α, β, γ) legt jeden Punkt $v \in \mathbb{R}^2$ bezüglich des Dreiecks $T = \Delta(v_1, v_2, v_3)$ eindeutig fest. Dabei gilt $v_1 = (1,0,0),\ v_2 = (0,1,0)$ und $v_3 = (0,0,1)$. Man nennt (α, β, γ) die baryzentrischen Koordinaten von v bezüglich des Dreiecks T.

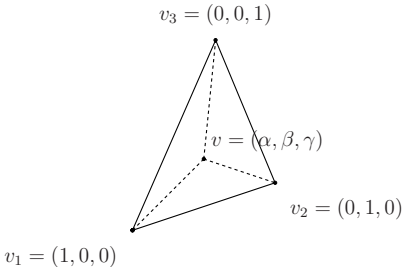


Abbildung 7.1: Baryzentrische Koordinaten bezüglich eines Dreiecks T

Baryzentrische Koordinaten sind gegenüber affinen Transformationen invariant.
Für die Integrationsformel für das Dreieck bedeutet das

$$\int\int f(\alpha, \beta, \gamma) dA \approx A \sum_{i=1}^{N} w_i f(\alpha_i, \beta_i, \gamma_i),$$

wobei A der Flächeninhalt des Dreiecks ist. Analog bedeutet dies für den Tetraeder

$$\int\int\int f(\alpha, \beta, \gamma, \delta) dV \approx V \sum_{i=1}^{N} w_i f(\alpha_i, \beta_i, \gamma_i, \delta_i),$$

wobei V das Volumen des Tetraeders ist.
Gegenüber den Parametern der Tensorprodukt-Formeln haben diese speziellen Parameter den Nachteil, dass sie nicht einfach in weitere Dimensionen verallgemeinert werden können, allerdings ist der Vorteil der speziellen Parameter, dass oftmals weniger Gauß-Parameter benötigt werden. In den Tabellen 7.2 und 7.3 ist eine Übersicht dargestellt über die verwendeten Stützstellen der Newton-Cotes Formeln im Vergleich zu Produktformeln, die auf der eindimensionalen Gauß Quadratur basieren, und die Anzahl der besten bisher bekannten Formeln [26].

Grad	*Produktformel*	*Newton-Cotes Formel*	*Beste bekannte*
1	1	3	1*
2	4	6	3*
3	4	10	4*
4	9	15	6*
5	9	21	7*
6	16	28	12
7	16	36	12*
8	25	45	15*
9	25	55	19
10	36	66	22
11	36	78	27
12	49	91	33
13	49	105	36
14	64	120	42
15	64	136	48
16	81	153	52
17	81	171	61
18	100	190	70
19	100	210	73
20	121	231	79

* Es ist bekannt, dass dies die minimale mögliche Anzahl für den gegebenen Grad ist.

Tabelle 7.2: Anzahl der Stützpunkte für Dreiecksformeln

7.2 Integration über isoparametrischen Elementen

Affine Abbildungen bilden das Einheitsquadrat immer auf ein Parallelogramm ab und das Einheitsdreieck wieder auf ein Dreieck, wie in Abbildung 7.2 zu sehen.

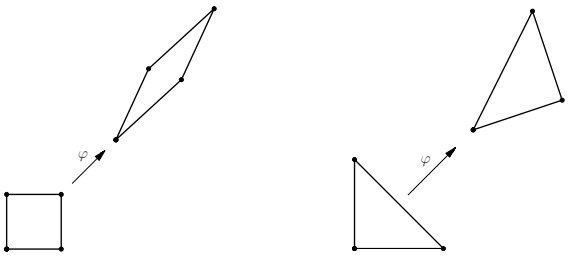


Abbildung 7.2: Affine Abbildung des Einheitsquadrates und Einheitsdreiecks

Ist mehr Flexibilität notwendig und eine stückweise lineare Approximation des Randes nicht ausreichend, kann man durch die Verwendung von isoparametrischen Elementen, d.h. krummlinige Dreiecke beziehungsweise krummflächige Tetraeder, eine bessere Approximation des Randes ∂D erreichen. Der Einfachheit halber wird sich in diesem Abschnitt auf

Grad	*Produktformel*	*Newton-Cotes Formel*	*Beste bekannte*
1	1	4	1*
2	8	10	4*
3	8	20	5*
4	27	35	11
5	27	56	14
6	64	84	24
7	64	120	31
8	125	165	43
9	125	220	53

* Es ist bekannt, dass dies die minimale mögliche Anzahl für den gegebenen Grad ist.

Tabelle 7.3: Anzahl der Stützpunkte für Tetraederformeln

krummlinige beziehungsweise krummflächige Elemente direkt am Rand beschränkt. Solche Elemente sind auch im Inneren von D möglich, aber hier nicht notwendig. Die krummlinige Kante beziehungsweise Fläche wird hier durch ein Polynom interpoliert. Anstelle der affinen Abbildung φ verwendet man hierzu eine Abbildung $p : E \to K$ vom Einheitsdreieck beziehungsweise Einheitstetraeder auf ein entsprechendes krummliniges oder krummflächiges Element mit

$$p = \begin{pmatrix} p_1(x,y) \\ p_2(x,y) \end{pmatrix} \quad \text{beziehungsweise} \quad p = \begin{pmatrix} p_1(x,y,z) \\ p_2(x,y,z) \\ p_3(x,y,z) \end{pmatrix}$$

wobei die p_i Polynome sind, deren Grad abhängig ist vom Grad der Polynome, durch die die krummlinige Kante oder Fläche interpoliert wird. Zusammenfassend erhält man somit folgende Definition:

Definition 7.3 *Man nennt die Dreiecke oder Tetraeder T einer Familie von Elementen T_h isoparametrische (Dreiecks-/Tetraeder-) Elemente, wenn durch eine bijektive Abbildung $F : T_{ref} \to T$ mit*

$$(x,y) \mapsto (x',y') = p(x,y) = (p_1(x,y), p_2(x,y))$$

beziehungsweise

$$(x,y,z) \mapsto (x',y',z') = p(x,y,z) = (p_1(x,y,z), p_2(x,y,z), p_3(x,y,z))$$

das Einheitsdreieck oder den Einheitstetraeder auf jedes $T \in T_h$ abgebildet wird, wobei die p_i polynomiale Koordinationfunktionen sind.

7.2.1 Isoparametrische Dreieckselemente

Im Fall eines Dreiecks mit quadratischer Approximation der krummlinigen Kante erhält man p aus einem linearen Gleichungssystem. Die p_i sind quadratische Polynome der Form

$$p_i(x,y) = ax^2 + by^2 + cxy + dx + ey + f.$$

Zur Bestimmung der Koeffizienten sind weitere Kontrollpunkte als die im affinen Fall verwendeten erforderlich. Zusätzlich zu den Ecken des Dreiecks werden hierfür noch die Kantenmittelpunkte verwendet. Die Bildpunkte dieser können eigentlich beliebig bestimmt werden, allerdings beeinflußt deren Lage den Verlauf der interpolierenden Kurve. Es empfiehlt sich, die Punkte nahe dem Kantenmittelpunkt zu wählen. Dies kann beispielsweise durch den Schnitt einer nach außen gerichteten, senkrecht zur Kante stehenden Gerade durch den Kantenmittelpunkt mit dem Rand geschehen. Wählt man die Bildpunkte der Kantenmittelpunkte genau in der Mitte der Bildeckpunkte, erhält man den Spezialfall der affinen Abbildung aus dem vorangegangenen Abschnitt.
Die entsprechenden Punkte des Einheitsdreiecks werden mit der Abbildung p auf die Punkte P_i des krummlinigen Dreiecks abgebildet. Daraus und aus

$$\begin{aligned} p_1(x,y) &= a_1x^2 + a_2y^2 + a_3xy + a_4x + a_5y + a_6 \\ p_2(x,y) &= b_1x^2 + b_2y^2 + b_3xy + b_4x + b_5y + b_6 \end{aligned}$$

erhält man anhand der x-Werte der P_i für p_1

$$\begin{aligned} x_1 &= a_6 \\ x_2 &= a_1 + a_3 + a_6 \\ x_3 &= a_2 + a_4 + a_6 \\ x_4 &= \frac{1}{4}a_1 + \frac{1}{2}a_3 + a_6 \\ x_5 &= \frac{1}{4}a_1 + \frac{1}{4}a_2 + \frac{1}{2}a_3 + \frac{1}{2}a_4 + \frac{1}{4}a_5 + a_6 \\ x_6 &= \frac{1}{4}a_2 + \frac{1}{2}a_4 + a_6 \end{aligned}$$

und ein analoges System mit den y-Werten für p_2. Die Koeffizienten von p_1 und p_2 lassen sich also jeweils aus einem solchen System bestimmen. In Matrixschreibweise lauten die beiden Systeme also

$$x = Ma \quad \text{und} \quad y = Mb\,,$$

wobei x und y die Koordinatenvektoren mit den x- beziehungsweise y-Koordinatten der Punkte P_i sind und a und b die Koeffizientenvektoren von p_1 und p_2. Man kann den Grad der Koordinatenfunktionen erhöhen, es werden entsprechend mehr Kontrollpunkte P_i benötigt. Für Koordinatenpolynome vom Grad m sind $\frac{(m+2)(m+1)}{2}$ Kontrollpunkte notwendig.
Ein anderer Ansatz die Abbildung zu wählen, die die Dreiecke auf die krummlinigen Elemente abbildet, ist der folgende geometrische Ansatz, der in Abbildung 7.3 illustriert ist. Wie in der Abbildung zu sehen, sind die schwarz markierten Punkte P_{200}, P_{020} und P_{002} die Bildpunkte der Ecken des Einheitsdreiecks. Die mit Kreisen markierten Punkte P_{110}, P_{101} und P_{001} sind die Bildpunkte der Seitenmittelpunkte.
Mit diesen Punkten lässt sich die Abbildung

$$\varphi(x) = \sum_{i+j+k=2} P_{ijk} b^2_{i,j,k}(x),$$

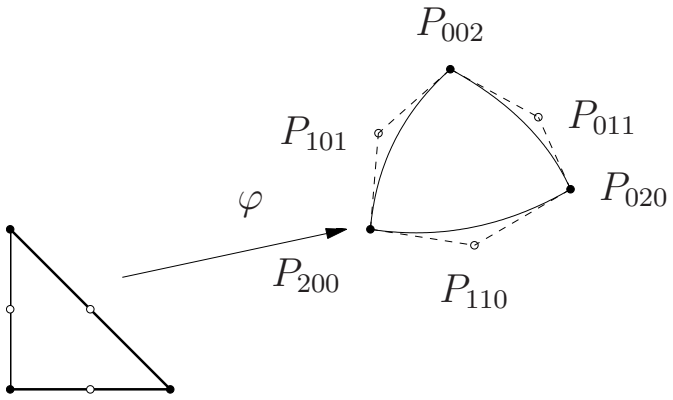


Abbildung 7.3: Geometrischer Ansatz zur Konstruktion der Abbildung φ

konstruieren, wobei die

$$b^2_{i,j,k}(x) = \frac{n!}{i!\,j!\,k!} u^i v^j w^k \qquad \text{mit} \qquad x = (u, v, w)^t$$

die Bernsteinpolynome vom Grad zwei für die baryzentrischen Koordinaten von $x \in \mathbb{R}^2$ sind [20].
Die Bildpunkte der Kantenmittelpunkte lassen sich geometrisch konstruieren. Für Kanten, die nicht auf eine krummlinige Kante abgebildet werden, weil sie nicht direkt am Gebietsrand ∂D liegen, wird als Bildpunkt des Kantenmittelpunktes wieder die Mitte der Bildpunkte der entsprechenden Ecken gewählt. Bei Kanten, die auf eine krummlinige Kante abgebildet werden, liegen die Eckpunkte auf dem Rand ∂D. Konstruiert man in diesen Eckpunkten die Tangente an die Randkurve und wählt den Schnittpunkt der beiden Tangenten als Bildpunkt der Mittelpunkte des Einheitsdreieck, erhält man eine bessere Genauigkeit von $\mathcal{O}(h^4)$ statt $\mathcal{O}(h^3)$ beim vorherigen Ansatz [17].

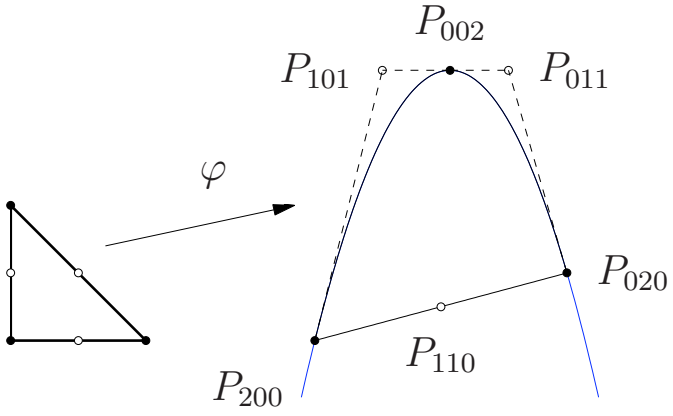


Abbildung 7.4: Abbildung auf ein krummliniges Dreieck mit einer Parabel als Randkurve

Die Abbildungen 7.4 und 7.5 zeigen zwei Möglichkeiten von Dreiecken, die auf ein krummliniges Dreieck abgebildet werden und die Randkurve, die approximiert werden soll. Während

die Parabel exakt approximiert wird, da sie durch ein Bernstein-Polynom darstellbar ist, ist beim Halbkreis noch ein deutlicher Unterschied zu sehen.

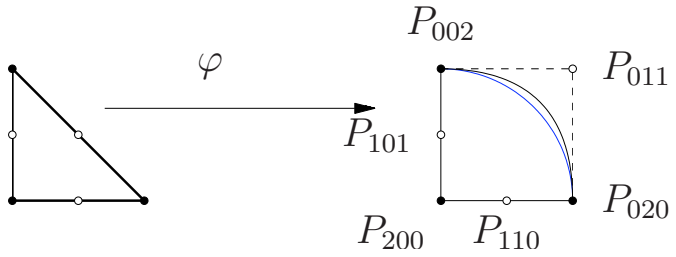


Abbildung 7.5: Abbildung auf ein krummliniges Dreieck mit einem Viertelkreis als Randkurve

Es werden soweit möglich geradlinige Dreiecke verwendet, weshalb nur Dreiecke mit einer oder maximal zwei krummlinigen Seiten, die direkt am Rand ∂D liegen, vorkommen. Dies hat zur Folge, dass man immer einen stetigen Übergang zu benachbarten Dreiecken hat. Es kann aber durchaus vorkommen, dass beide Eckpunkte einer Kante auf der Randkurve liegen, die Kante aber trotzdem nicht auf eine krummlinige Kante abgebildet werden soll. Dies kommt bei Dreiecken vor, bei denen alle drei Eckpunkte auf dem Rand liegen. Dies ist bei der Implementierung zu beachten, da durch die Abbildung ansonsten ein degeneriertes Dreieck entsteht.

7.2.2 Isoparametrische Tetraederelemente

Durch die Verwendung von krummlinigen Tetraederelementen kann man allgemeine räumliche Gebiete erfassen. Analog zu den Dreiecken lässt sich auch mit Tetraedern verfahren. Die Abbildung p mit den drei Koordinatenfunktionen

$$\begin{aligned} p_1(x,y,z) &= a_1x^2 + a_2y^2 + a_3z^2 + a_4x + a_5y + a_6z + a_7xy + a_8xz + a_9yz + a_{10} \\ p_2(x,y,z) &= b_1x^2 + b_2y^2 + b_3z^2 + b_4x + b_5y + b_6z + b_7xy + b_8xz + b_9yz + b_{10} \\ p_3(x,y,z) &= c_1x^2 + c_2y^2 + c_3z^2 + c_4x + c_5y + c_6z + c_7xy + c_8xz + c_9yz + c_{10} \end{aligned}$$

bildet wieder Kontrollpunkte vom Einheitstetraeder T_{ref} auf den entsprechenden Tetraeder T der Familie T_h ab. Für einen quadratischen Ansatz benötigt man zehn Kontrollpunkte. Zur Bestimmung der Koeffizienten der drei Koordinatenfunktionen können drei äquivalente Gleichgungssysteme aufgestellt werden. Analog zum Zweidimensionalen kann man wieder die drei Systeme in Matrixschreibweise angeben durch

$$x = Ma\,, \quad y = Mb \quad \text{und} \quad z = Mc$$

wobei x, y und z die Vektoren mit den x-, y- und z-Koordinaten der Kontrollpunkte P_i sind und a, b und c die Koeffizientenvektoren der Koordinatenpolynome p_i.
Der Ansatzgrad lässt sich auch hier erhöhen, was allerdings mit einer schnell steigenden Knotenanzahl einher geht, so benötigt man für einen kubischen Ansatz schon 20 Kontrollpunkte.

Der Ansatz mit Bernsteinpolynomen ist auch im Dreidimensionalen dem oben vorgestellten Ansatz vorzuziehen. In Abbildung 7.6 ist analog zum Zweidimensionalen die Abbildung φ des Einheitstetraeders auf einen krummkantigen Tetraeder dargestellt.

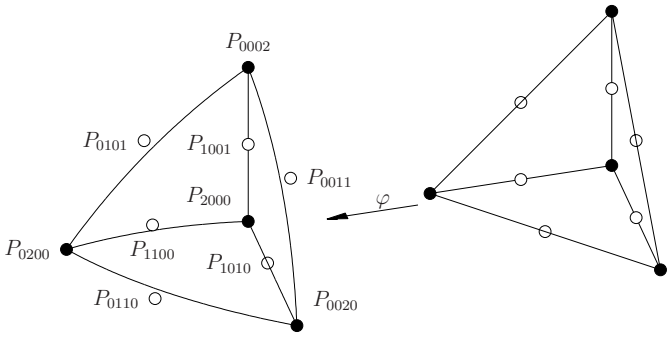


Abbildung 7.6: Geometrischer Ansatz zur Abbildung des Einheitstetraeders

Die Abbildung setzt sich zu

$$\varphi(x) = \sum_{i+j+k+l=2} P_{ijkl} b^2_{i,j,k,l}(x)$$

aus den Bernsteinpolynomen

$$b^2_{i,j,k,l}(x) = \frac{n!}{i!\,j!\,k!\,l!} u^i v^j w^k t^l \qquad \text{für} \qquad x = (u,v,w,t)^t$$

vom Grad 2 in baryzentrischen Koordinaten bezüglich des Einheitstetraeders im $\mathbb{R}^3$ zusammen.

7.2.3 Integration

Analog zum affinen Fall wird über den krummlinigen Elementen mit der Gauß-Quadratur integriert. In der Literatur werden Gauß-Stützstellen, wie bereits erwähnt, oftmals in baryzentrischen Koordinaten angegeben, vor allem, wenn sie speziell für Simplizes angegeben werden.
Deshalb ist in diesen Fällen eine andere Darstellungsform von p sinnvoll. Für quadratische Approximation lautet diese

$$p = \sum_{i=1}^{n} \lambda_i(x)(2\lambda_i(x) - 1)a_i + \sum_{i<j} 4\lambda_i(x)\lambda_j(x)a_{ij}$$

und für kubische Approximation

$$\begin{aligned} p &= \sum_{i=1}^{n} \frac{\lambda_i(x)(3\lambda_i(x) - 1)(3\lambda_i(x) - 2)}{2} a_i \\ &+ \sum_{i\neq j} \frac{9\lambda_i(x)\lambda_j(x)(3\lambda_i(x) - 1)}{2} a_{ij} + \sum_{i<j<k} 27\lambda_i(x)\lambda_j(x)\lambda_k(x)a_{ijk}, \end{aligned}$$

wobei die $\lambda_i(x)$ die baryzentrischen Koordniaten des Punktes x sind und die a_i, a_{ij} und a_{ijk} die Kontrollpunkte zur Bestimmung der Abbildung sind.
Diese Darstellung ist im Grunde die Bernstein-Darstellung, die oben vorgestellt wurde.
Analog zum affinen Fall lautet das Integral über dem Einheitsdreieck hier

$$\begin{aligned}
&\int_0^1 \int_0^{1-x} f(p(x,y))|\det J(x,y)|dxdy = \\
&\int_0^1 \left(\int_0^{1-x} f(p(x,y))|\det J(x,y)|dy \right) dx = \\
&\sum_{j=1}^{N} w_j \left(\int_0^{1-\xi_j} f(p(\xi_j,y))|\det J(\xi_j,y)|dy \right) = \\
&\sum_{j=1}^{N} w_j(1-\xi_j) \sum_{k=1}^{N} w_k f(p(\xi_j,(1-\xi_j)\xi_k))|\det J(\xi_j,(1-\xi_j)\xi_k)|,
\end{aligned}$$

wobei im isoparametrischen Fall die Determinante von J nicht konstant ist und deshalb nicht vor das Integral gezogen werden kann.

7.3 Neuronale Netze

Ein vollkommen anderer und neuer Ansatz zur Berechnung der Integrale ist die Idee, diese mit Hilfe eines neuronalen Netzes zu berechnen. Es wird in diesem Abschnitt nur im benötigten Rahmen auf neuronale Netze und deshalb auf viele Aspekte dieses Themas nicht eingegangen. Die Geschichte der neuronalen Netze sowie ausführlichere Erklärungen zu den hier angesprochenen Bereichen sind der Literatur, beispielsweise [32], zu entnehmen, ebenso Informationen zu den hier ausgelassenen Bereichen. In der Literatur existiert bisher kein einheitlicher Standard für Bezeichnungen und Indizierungsreihenfolge. Für diesen Abschnitt wird sich an [32] orientiert, deshalb kann es vorkommen, dass an anderer Stelle beispielsweise das Gewicht w_{ij} zwischen zwei Neuronen umgekehrt indiziert ist, es ist also Vorsicht geboten beim Vergleich mit anderer Literatur.
Zu Anfang dieses Abschnittes soll zunächst auf die Frage eingegangen werden, was ein neuronales Netz ist und welches seine Bestandteile sind.

7.3.1 Bestandteile neuronaler Netze

Man verwendet (künstliche) neuronale Netze bei Problemen, die man nicht in einen Algorithmus fassen kann oder die von zu vielen Faktoren abhängig sind. Man unterscheidet generell zwischen neuronalen Netzen, die entworfen werden, um das menschliche Gehirn und dessen Funktionsweise besser verstehen zu können, und neuronalen Netzen, mit denen spezifische Anwendungsprobleme gelöst werden sollen.
In Anlehnung an das menschliche Gehirn, jedoch in stark vereinfachter Form, besteht ein neuronales Netz aus vielen einfachen Einheiten beziehungsweise Zellen, den so genannten Neuronen, und gerichteten Verbindungen, über die Neuronen Aktivierungsinformationen an andere Neuronen übermitteln. Genau wie das menschliche Gehirn muss das neuronale Netz lernen, um die ihm gestellten Aufgaben lösen zu können.
Im allgemeinen sind neuronale Netze aus den folgenden Bestandteilen aufgebaut [32]:

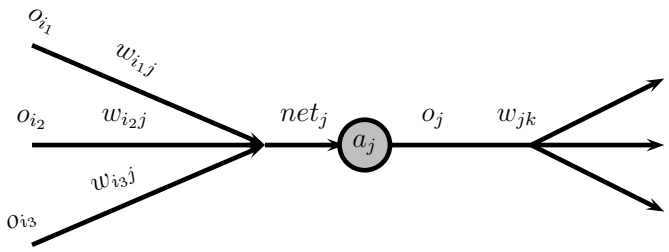


Abbildung 7.7: Aufbau eines Neurons

1. Zellen beziehungsweise Neuronen bestehen aus den folgenden Teilen:
   - Schwellenwert Θ: Gibt an, ab wann ein Neuron aktiv ist.
   - Aktivierungszustand $a_i(t)$: Bezeichnet den Grad der Aktivierung des Neurons
   - Aktivierungsfunktion f_{act}: Funktion, die in der allgemeinen Formel meist mit

     $$a_j(t+1) = f_{act}(a_j(t), net_j(t), \Theta_j(t))$$

     angegeben ist und aus den gegebenen Parametern, dem alten Aktivierungszustand a_i, der Netzeingabe $net_j(t)$ und dem Schwellenwert des Neurons $\Theta_j(t)$, die neue Aktivierung berechnet.
   - Ausgabefunktion f_{out}: Funktion $o_j = f_{out}(a_j)$, die aus der Aktivierung der Zelle deren Ausgabe berechnet.

   Es wird häufig eine nichtlineare Ausgabefunktion verwendet. Je nach Umsetzung ist diese Ausgabefunktion auch Bestandteil der Aktivierungsfunktion, wodurch die Ausgabefunktion dann als Identität gewählt werden kann. Häufig zur Verwendung kommende Aktivierungs- beziehungsweise Ausgabefunktionen sind die binäre Schwellenwertfunktion, sigmoide Funktionen wie die logistische Funktion $1/(1+exp(-x))$, der Tangens Hyperbolicus auf dem Intervall $[-1, 1]$ oder radiale Funktionen. Meistens sind die Aktivierungs- und Ausgabefunktion für alle oder sehr viele Neuronen gleich und es ist nur der Schwellenwert Θ_i unterschiedlich.

2. Verbindungsnetzwerk zwischen den Neuronen. Über die gerichteten Verbindungen zwischen den Neuronen werden Daten übertragen, die durch die Verbindungsgewichte w_{ij} verstärkt oder geschwächt werden. Die Matrix der Gewichte aller Verbindungen wird mit $W = (w_{ij})$ bezeichnet, wobei w_{ij} das Gewicht der Verbindung von Neuron i zu Neuron j bezeichnet.

3. Propagierungsfunktion. Durch die eingehenden Verbindungen erhält ein Neuron die Ausgaben o_i anderer Neuronen, die noch mit dem Gewicht w_{ij} gewichtet wurden. Daraus berechnet die Propagierungsfunktion die Netzeingabe des Neurons mittels

   $$net_j(t) = \sum_i o_i(t) w_{ij}$$

   als gewichtete Summe.

4. Lernregel. Das neuronale Netz lernt, gemäß eines Algorithmus für eine vorgegebene Eingabe das gewünschte Ergebnis auszugeben. Lernen bedeutet, dass das neuronale Netz verändert wird, eine Modifikation der Verbindungsgewichte verändert die Ausgabe des Netzes. Ziel ist es, durch Trainieren des Netzes den Fehler zwischen der Ausgabe und dem gewünschten Ergebnis zu minimieren.

In den hier verwendeten Definitionen und Notationen wurde an manchen Stellen durch ein (t) die Abhängigkeit der Größe von der Zeit dargestellt. Die Zeit wird hier in Zeitschritten gemessen, so bedeutet t die aktuelle Zeit, $t-1$ den vorherigen und $t+1$ den nächsten Zeitschritt.

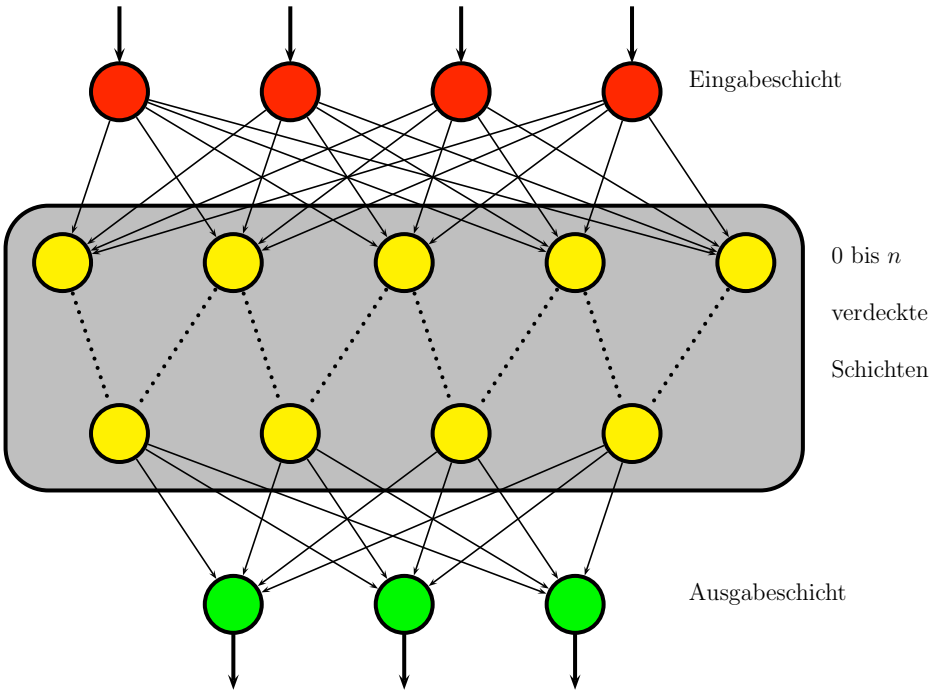


Abbildung 7.8: Zelltypen eines neuronalen Netzes

Die eingeführten Bezeichnungen sind in der Übersicht über einen Spezialfall eines so genannten feedforward-Netzes in den Abbildungen 7.7 und 7.8 zu sehen, ebenso wie die Einteilung der Zelltypen in Eingabeneuronen, verdeckte Neuronen und Ausgabeneuronen je nach Lage im neuronalen Netz. Hier werden in der Literatur verschiedene Arten von neuronalen Netzen unterschieden, beispielsweise Netze mit und ohne Rückkopplungen und shortcut connections, vollständig verknüfpte und nicht vollständig verknüpfte Netze. Im Gegensatz zu den meisten anderen Größen des neuronalen Netzes sind die Aktivierungsfunktion und die Ausgabefunktion meist global und werden in diesem Fall auch global definiert.

7.3.2 Eigenschaften neuronaler Netze

Neuronale Netze werden auch wegen ihrer positiven Eigenschaften zur Problemlösung eingesetzt.

- Fehlertoleranz gegenüber Fehlern innerhalb des Netzes. Neuronale Netze reagieren tolerant auf einen oder wenige Fehler innerhalb des Netzes, die beispielsweise durch fehlerhafte Neuronen oder Verbindungen auftreten. Das neuronale Netz kompensiert diese Fehler durch die verteilte Wissensspeicherung innerhalb des Netzes.

- Fehlertoleranz gegenüber Eingabefehlern. Im Gegensatz zu programmierten Algorithmen besitzen neuronale Netze eine Fehlertoleranz gegenüber ungenauen, beispielsweise durch ein Rauschen, fehlerhaften und widersprüchlichen Eingaben.

- Parallelität. Aufgrund ihrer Struktur bieten neuronale Netze ideale Möglichkeiten zur parallelen Verarbeitung. Auf der entsprechenden für paralleles Rechnen ausgelegten Hardware liefern sie deshalb eine sehr gute Performance, was sogar den Einsatz in Echtzeitanwendungen möglich macht.

- Einfache Elemente. Neuronale Netze bestehen aus sehr einfachen Bausteinen, ihre Komplexität entsteht durch die Interaktion dieser einfachen Elemente.

- Lernfähigkeit und Generalisierungsfähigkeit. Die wohl hervorstechendste Eigenschaft von neuronalen Netzen ist die Selbstorganisation, beziehungsweise die Fähigkeit zu lernen, und die Assoziationsfähigkeit, also die Möglichkeit zur Generalisierung von den Beispielen der Lernphase zu weiteren Problemen der gleichen Art.

- Adaptivität. Neuronale Netze verändern während der Lernphase selbstständig ihre Gewichte und je nach Topologie auch ihre Neuronen und Verbindungen. Sie können sich weiterhin gut an sich ändernde Problemstellungen durch Nachlernen anpassen.

Neben all diesen Vorteilen gibt es bei neuronalen Netzen jedoch auch einige Nachteile, die berücksichtigt werden sollten.

- Wissen. Im Gegensatz zu anderen Ansätzen der künstlichen Intelligenz kann man bei einem neuronalen Netz das Wissen nicht einsehen, da das Wissen in den Neuronen und Gewichten gespeichert ist und nicht etwa in einer Datenbank.

- Kein Grundwissen. Es ist nicht oder nur sehr beschränkt möglich, einem neuronalen Netz vor der Trainingsphase ein gewisses Grundwissen mitzugeben, sämtliches Wissen muss erlernt werden. Bei anderen Ansätzen der künstlichen Intelligenz ist dies beispielsweise durch eine Wissensdatenbank realisierbar.

- Fehleranalyse. Aufgrund der Verteilung des Wissens in den Verbindungen und Neuronen des Netzes ist die Fehlersuche bei einem neuronalen Netz schwierig, da die Entscheidungen des Netzes nur beschränkt nachvollziehbar sind. Es ist immer nur die Ausgabe des Netzes zu sehen, nicht aber deren Entstehung.

- Trial and Error. Es gibt keine festen Richtlinien, nach denen ein neuronales Netz aufgebaut werden muss, um eine bestimmte Problemstellung ausreichend gut zu lösen. Die Größe und Struktur, sowie die ideale Wahl für die meisten weiteren Komponenten, müssen nach dem Trial and Error Prinzip gefunden werden. Dies macht es zeitaufwändig, ein ideales Netz zu erstellen.

7.3.3 Aufbau eines neuronalen Netzes

Zur Integration von Polynomen mit neuronalen Netzes ist es zunächst einmal erforderlich, das Netz, das die Aufgabe lösen soll zu erstellen und anschließend zu trainieren. Ziel ist es, die für das Aufstellen des Ritz-Galerkin-Systems benötigten Integrale auf beliebigen aus den in dieser Arbeit verwendeten einfachen Objekten erzeugten Gebieten D durch das neuronale Netz berechnen zu lassen. Eine entsprechende Wahl der Netzart, der verschiedenen benötigten Funktionen und der Lernart und Lernregel wird in diesem Abschnitt dargestellt.
Die Frage nach der Lernart ist zunächst schnell beantwortet, da hierfür nur überwachtes Lernen Sinn macht. Ziel ist es, das Netz mit Integralen, deren Werte bekannt sind, zu trainieren und daraus dann eine Generalisierung auf weitere, auch komplexere Integrale zu erreichen. Da die Werte der Integrale bekannt sind, ist es nur sinnvoll, diese zum Trainieren des neuronalen Netzes auch zu verwenden, da das überwachte Lernen gegenüber bestärkendem Lernen und unüberwachtem Lernen den schnellsten Lernerfolg erzielt und hier auf biologische Plausibilität nicht geachtet wird. Während der Lernphase spielt Rechenzeit eine untergeordnete Rolle, weshalb durchaus vorher aufwändig berechnete Integrale zur Trainingsmenge gehören können und sollten.

Aufbau des neuronalen Netzes

Der Aufbau des Netzes kann auf verschiedene Arten gelöst werden, es gibt nicht das neuronale Netz, sondern eine Vielzahl von Möglichkeiten ein solches zu realisieren, abhängig von der Aufgabe, die es erfüllen soll. Ohne eine entsprechende Implementierung ist es schwer, Aussagen über den besten Aufbau, die Größe und weitere Beschaffenheit des neuronalen Netzes für die Integration und die verwendeten Funktionen und Verfahren zu treffen. Als Basis einer Implementierung und entsprechenden Tests kann das hier entworfene neuronale Netz dienen.
Es wird vorgeschlagen, ein so genanntes Multilayer Perceptron, ein FeedForward Netz ohne shortcuts und Rückkopplungen, zu verwenden, denn es bildet aufgrund seiner einfachen Topologie eine solide Basis und ermöglicht einfaches Weiterarbeiten ohne größere Einschränkungen.

Definition 7.4 *Ein Perceptron ist ein FeedForward Netz, bei dem jede Schicht zur folgenden vollverknüpft ist. Die Datenaufnahmeschicht, die Retina, ist fest mit der ersten Schicht, der Eingabeschicht, verknüpft. Darauf folgt mindestens eine trainierbare Schicht von Gewichten. Ein Perceptron mit genau einer trainierbaren Schicht nennt man Single-Layer-Perceptron (SLP). Analog wird ein Perceptron mit mehreren trainierbaren Schichten Multi-Layer-Perceptron (MLP) genannt.*

Bei vielen neuronalen Netzen ist nicht die Anzahl der trainierbaren Schichten ausschlaggebend, sondern deren Breite, so dass oft drei bis fünf Schichten ausreichend sind.
Zum Einsparen von Rechenzeit ist es bei FeedForward Netzen möglich, ausgehend von der Topologie des Netzes die Aktivierungsreihenfolge der Neuronen im Voraus zu bestimmen und diese während der gesamten Zeit unverändert zu lassen. Dies ist nur möglich, wenn durch das Lernen die Topologie des Netzes nicht verändert wird.

Wahl der Neuronenfunktionen

Bei der Implementierung eines neuronalen Netzes ist es oft einfacher und sinnvoller, den Schwellenwert Θ der Neuronen als Gewicht der Verbindung eines besonderen on-Neurons oder Bias-Neurons zum entsprechenden Neuron zu realisieren. Ein on-Neuron sendet immer den Wert 1 über die Verbindung zu jedem Neuron. Diese Verbindung zum i-ten Neuron hat das Gewicht $-\Theta_i$. Setzt man den Schwellenwert innerhalb jedes Neurons dann auf 0, verlagert man durch die Verwendung des Gewichtes den Einfluss des Schwellenwerts von der Aktivierungsfunktion in die Propagierungsfunktion, der Schwellenwert wird also von der Netzeingabe subtrahiert. In Abbildung 7.9 ist dies für einen einfachen Ausschnitt eines Netzes dargestellt, bei der Darstellung größerer Netze wird der Übersichtlichkeit wegen darauf verzichtet, das Bias Neuron darzustellen.

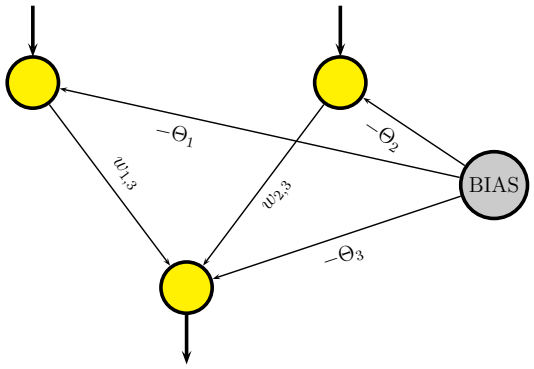


Abbildung 7.9: Ein kleines neuronales Netz mit Bias Neuron

Die Verwendung eines Bias-Neurons hat den weiteren Vorteil, dass die Schwellenwerte einfacher während des Lernens modifiziert werden können, da Lernregeln die Gewichte der Verbindungen verändern und die Veränderung weiterer Gewichte keinen großen Aufwand darstellt, während die Veränderung der Schwellenwerte unter Umständen eine neue Routine erfordert.
Die Propagierungsfunktion wird meist einfach als gewichtete Summe realisiert, wodurch sich der Schwellenwert gut darin integrieren lässt.
Die Aktivierung eines Neurons hängt zu einem bestimmten Zeitschritt immer von seiner vorigen Aktivierung und den Eingaben des aktuellen Zeitschrittes ab.
Die Aktivierungsfunktion in der Eingabeschicht liegt direkt fest. Für die verdeckten Neuronen wird der Tangens Hyperbolicus mit dem Wertebereich $(-1, 1)$ empfohlen und in der

Ausgabeschicht eine lineare Aktivierungsfunktion, damit kein begrenztes Ausgabeintervall entsteht.
Üblicherweise wird die Identität als Ausgabefunktion gewählt, und es wird somit einfach die Aktivierung weitergegeben.

Lernregel

Die Lernregel ist ein Algorithmus, der in einer Programmiersprache implementiert werden kann und der festlegt, wie das neuronale Netz lernt. Grundsätzlich lernt ein neuronales Netz durch die folgenden Modifikationen:

- Veränderung von Verbindungsgewichten
- Erstellen oder Löschen von Verbindungen
- Veränderung der Schwellenwerte von Neuronen
- Veränderung von mindestens einer der drei Neuronenfunktionen
- Erstellen oder Löschen von Neuronen

Die am häufigsten verwendete Methode ist das Verändern der Verbindungsgewichte. Werden Neuronen hinzugefügt oder entfernt, so ist darauf zu achten, dass die Reihenfolge, in der die Neuronen abgearbeitet werden, daran angepasst wird, was bei der Implementierung ein spürbares Maß an Rechenzeit erfordert.
Die meisten Lernregeln basieren auf der Hebb-Regel, die in der folgenden Definition angegeben ist [32].

Definition 7.5 *Wenn die Zelle j eine Eingabe von Zelle i erhält und beide gleichzeitig stark aktiviert sind, dann erhöhe das Gewicht w_{ij}.*
Mathematisch lautet die Hebb-Regel

$$\Delta w_{ij} = \eta o_i a_j.$$

Für das Lernen mit der Backpropagation-Regel für mehrstufige Perceptrons, die hier als Ansatz empfohlen wird, sind semilineare, das heißt monotone und differenzierbare Aktivierungsfunktionen notwendig, was durch die oben gewählten Aktivierungsfunktionen erfüllt ist. Mathematisch kann die Backpropagation-Regel wie folgt definiert werden:

Definition 7.6 *Die Änderung des Gewichtes w_{ij} wird berechnet durch*

$$\Delta w_{ij} = \eta \delta_j,$$

wobei

$$\delta_j = \begin{cases} f_j'(net_j)(t_j - o_j) & \textit{falls } j \textit{ eine Ausgabezelle ist} \\ f_j'(net_j) \sum_k (\delta_k w_{jk}) & \textit{falls } j \textit{ eine verdeckte Zelle ist.} \end{cases}$$

Das Backpropagation Lernverfahren ist ein Gradientenabstiegsverfahren. Mit der Fehlerfunktion, die alle Gewichte des Netzes als Argument erhält und den Fehler des Netzes über alle Trainingsmuster aufsummiert, lässt sich der Fehler als Fehlerfläche darstellen. Ein Gradientenverfahren funktioniert generell nach dem Prinzip, dass ausgehend von einem Punkt die Richtung des steilsten Abstiegs gesucht wird, bis man in einem lokalen Minimum gelandet ist. Das bedeutet, es wird der Gradient der Fehlerfunktion berechnet und sich in Richtung des Gradienten bewegt, bis ein Minimum erreicht ist. Bei einer zweidimensionalen Fehlerfunktion lässt sich das sehr gut bildlich vorstellen. Man geht immer in die Richtung nach unten, in der es am steilsten ist, bis es in alle Richtungen nur noch bergauf geht. Das Problem hierbei ist wie bei allen Gradientenverfahren, dass man eine geeignete Schrittweite finden muss, damit man einerseits keine Minima überspringt und andererseits auf flachen Plateaus nicht zu lange braucht, um weiter zu kommen.
Fasst man die Gewichtsmatrix W als einen langen Gewichtsvektor auf, so kann man die Änderung der Gewichte mit dem Gradienten der Fehlerfunktion wie folgt berechnen:

$$\Delta W = -\eta \nabla E(W),$$

wobei η als Lernfaktor oder auch Schrittweite bezeichnet wird. Die Fehlerfunktion $E(W)$ wird durch

$$E = \sum_p E_p$$

als Summe der Fehler aller Trainingsmuster p

$$E_p = \frac{1}{2} \sum_j (t_{pj} - o_{pj})^2$$

als quadratischer Abstand zwischen der Lerneingabe t_{pj} und der Ausgabe o_{pj} realisiert. Der Term für E_p wurde etwas angepasst, um die Berechnungen zu erleichtern, denn zur Bestimmung der optimalen Gewichte ist es nicht ausschlaggebend, ob der Fehler oder der halbe Fehler minimiert wird und auch nicht, ob der euklidische Abstand oder dessen Quadrat minimiert werden. Wie schon erwähnt gibt es keine optimale Lernrate, sie hängt sehr von der Problemstellung und den Trainingsdaten ab. Bei komplexen Problemen ist es sinnvoll, mit Werten um 0.9 zu starten und diesen schrittweise auf beispielsweise 0.1 zu verringern. Des Weiteren kann durch die Verwendung einer kleinen Lernrate in der Ausgabeschicht verhindert werden, dass gute Minima durch zu große Lernschritte übersprungen werden.
Das Backpropagation Verfahren ist gut erweiterbar und die meisten Erweiterungen sind sehr gut als optionale Features implementierbar. Mögliche Erweiterungen sind beispielsweise:

- Momentum-Term. Fügt einen Trägheitsmoment zur Schrittweite hinzu.
- Flat Spot Elimination. Modifikation der Ableitung des Tangens Hyperbolicus und ähnlicher Funktionen, da diese außerhalb einer gewissen Umgebung um Θ fast Null ist.
- Weight Decay. Bestrafung größerer Gewichte, so dass die Fehlerfunktion weniger Schwankungen aufweist.

- Pruning. Zurechtstutzen des Netzes nach Weight Decay. Entfernt ein Neuron, wenn alle seine Nachfolgewerte Null oder fast Null sind.
- Optimal Brain Damage. Weiteres Verfahren, um nicht benötigte Gewichte und Neuronen zu entfernen.

Trainieren des neuronalen Netzes

Das neuronale Netz wird, wie bereits erwähnt, bei seinem Lernen überwacht. Prinzipiell läuft ein überwachtes Lernverfahren nach den folgenden Schritten ab:

1. Eingabeneuronen werden aktiviert, um das Testmuster einzugeben.
2. Vorwärtspropagierung erzeugt eine Ausgabe für dieses Testmuster.
3. Berechnung des Fehlervektors aus der Ausgabe und dem teaching input, dem gewünschten Ergebnis.
4. Rückwärtspropagierung des Fehlers, um diesen zu verringern. Dies liefert Änderungen der Gewichte der Verbindungen.
5. Anwenden der Änderungen, die im vorigen Schritt berechnet wurden.

Die Trainingsmenge, also die Menge der Lerneingaben, wird zusammengesetzt aus vorher berechneten Integralwerten von einfachen und komplexen Integralen. Auf diese Weise lassen sich ausreichend viele Trainingsmuster erstellen, die die gesamte Schwierigkeitsspanne der zu berechnenden Integrale enthalten. Wie bereits erwähnt spielt während der Trainingsphase Rechenzeit eine untergeordnete Rolle, so dass komplexere Integrale rechenzeitaufwändig in der gewünschten Genauigkeit berechnet werden können.
Es wird nicht die gesamte Trainingsmenge zum Trainieren des Netzes verwendet, man unterteilt diese in zwei Mengen etwa im Verhältnis 70:30, in eine Testmenge und eine Validierungsmenge. Mit der Testmenge wird das neuronale Netz wirklich trainiert, um anschließend, wenn das Netz gut genug trainiert ist, mit der Validierungsmenge zu prüfen, ob das Netz gut genug generalisiert. Damit soll überprüft und verhindert werden, dass das neuronale Netz die Trainingsmuster auswendig lernt und gar nicht mehr generalisiert. In diesem Zusammenhang ist schon beim Aufbau des Netzes darauf zu achten, dass man kein zu großes oder zu kleines Netz verwendet, da sonst die Gefahr besteht, dass das Netz auswendig lernt oder nicht in der Lage ist, gut genug zu lernen, um das gestellte Problem zu lösen und später allgemeinere Probleme ebenso gut zu behandeln. Es ist also ein guter Mittelweg zu finden. Man spricht hier von Repräsentierbarkeit und Lernbarkeit, das bedeutet, dass das Netz in der Lage sein muss, das Problem zu lösen, man aber das Netz auch in endlicher Zeit trainieren können muss. Es empfiehlt sich hier, in verschiedenen Teilen der Trainingsphase die Testmenge und Validierungsmenge zu verändern, also Elemente der beiden Mengen zu tauschen und das Größenverhältnis der Mengen zu verändern. Idealerweise verwendet man alle möglichen Kombinationen von Test- und Validierungsmenge, um das bestmögliche Ergebnis zu erzielen, allerdings ist dies in der Praxis wegen zu großem Zeitaufwand nicht umzusetzen.
Es empfiehlt sich bei der Implementierung, zuerst ein Netz mit wenigen, beispielsweise drei, trainierbaren Schicht zu trainieren und wenn dies nicht gut genug lernt, um eine

weitere Schicht zu erweitern. Die Anzahl der Eingabe- und Ausgabeneuronen sind durch die Problemstellung vorgegeben, und die Anzahl der restlichen Neuronen entspricht den Freiheitsgraden des Problems. Um den Mittelweg zu finden, gilt hier so viele Neuronen wie notwendig sind und so wenige wie möglich zu verwenden. Ein guter Ansatz zum Aufbau eines Netzes ist hier der Bottom-Up-Ansatz. Man beginnt mit einem Netz mit wenigen Neuronen und trainiert dann so lange neue Netze, bis das Lernergebnis nicht mehr deutlich besser wird.
Ein Maß für erfolgreiches Lernen und den Abschluß der Lernphase ist einfach zu finden: Es genügt zu testen, ob der Fehler bei der Berechnung aller Integrale kleiner ist, als eine vorher bestimmte Toleranzschwelle. Nach Abschluss der Lernphase lässt sich die Generalisierungsfähigkeit des Netzes überprüfen, indem man einige komplexere Beispiele, die man vorher auf andere Weise berechnet hat, vom Netz lösen lässt und diese mit dem vorher berechneten Ergebnis vergleicht.
Bei der Initialisierung der Gewichte eines neuronalen Netzes gilt es, bei Verwendung des Backpropagation Verfahrens darauf zu achten, dass die Gewichte nicht alle gleich initialisiert werden, da sonst die einzelnen Schichten unter Umständen keine unterschiedlichen Gewichte ausbilden können. Eine Möglichkeit dieses Problem zu umgehen nennt man symmetry breaking. Man verwendet als Initialisierung der Gewichte kleine Zufallswerte zwischen -1 und 1, was den schönen Nebeneffekt hat, dass der Durchschnitt der Netzeingabe nahe bei Null liegt.

7.3.4 Konkrete Umsetzung

Ein letzter Schritt für die konkrete Umsetzung eines neuronalen Netzes für die hier benötigten Integrationen ist die Festlegung der Eingabeparameter. Die Anzahl der Eingabeparameter wird, während das neuronale Netz lernt, nicht verändert und es muss eine Möglichkeit gefunden werden, ihre Anzahl unabhängig von der Komplexität des zu berechnenden Integrals fest zu lassen.
Betrachtet man wie bisher jede Zelle einzeln, so kann man mit wenigen Eingabeparametern auskommen und die Flexibilität bei der Gittergenerierung erhalten.
Man kann sich die durch die Klassifikation der Zellen gewonnenen Informationen zu Nutze machen kann, da ausschließlich über Randzellen integriert wird.
Als Eingabeparameter bei einer einzelnen Randzelle werden hier als erstes die Funktionswerte der Gewichtsfunktion an den Zellecken verwendet. Weitere Parameter sind die Koordinaten der Schnittpunkte des Gebietes mit den Zellkanten in baryzentrischen Koordinaten auf der entsprechenden Kante bezüglich deren beiden Eckpunkten: Jeder Punkt v zwischen zwei Punkten v_1 und v_2 lässt sich in Abhängigkeit von den beiden Punkten als Summe

$$v = (1-t)v_1 + tv_2 = \alpha v_1 + \beta v_2$$

schreiben. Die Anzahl der Zellkantenschnittpunkte ist allerdings nicht in jeder Zelle gleich, so dass man eine maximale Anzahl festlegen muss. Ausgehend von den Überlegungen im Kapitel über Triangulierung der Randzellen, kann man im Zweidimensionalen zwei Kantenschnittpunkte pro Zellkante als ausreichend betrachten. Das bedeutet, man erhält durch die zwei Koordinaten der acht Schnittpunkte weitere 16 Eingabeparameter, wobei die Parameter auf -1 gesetzt werden, wenn der entsprechende Kantenschnittpunkt nicht

existiert. Es ist hierbei jedoch zu prüfen, ob es eine sinnvollere Wahl für einen Wert für die Parameter von nicht existierenden Kantenschnittpunkten gibt. Ebenso ist es möglich die Schnittpunkte in baryzentrischen Koordinaten bezüglich der Zelle anzugeben und nicht von zwei Schnittpunkten pro Kante, sondern maximal von vier Schnittpunkten insgesamt auszugehen. Zusätzlich kann man weitere Funktionswerte der Gewichtsfunktion als Parameter verwenden, zum Beispiel 4×4 Punkte, die in der Zelle gleichverteilt sind.
Die Gitterweite h muss nicht berücksichtigt werden, da die Zelle immer auf das Einheitsquadrat beziehungsweise den Einheitswürfel skaliert werden kann.
Das ergibt für Randzellen insgesamt 36 Eingabeparameter.

7.3.5 Radiale Basisfunktionennetze

Eine weitere Möglichkeit, die Integration mit einem neuronalen Netz anzugehen, ist die Verwendung eines radialen Basisfunktionennetzes (RBF-Netz). Diese in der Mustererkennung eingesetzten Netze sind vollverknüpfte Netze ohne shortcuts mit genau einer Schicht versteckter Neuronen. Der Unterschied zu den bisher beschriebenen Perceptrons liegt in der Art der Informationsverarbeitung und dem Aufbau der verstecken Neuronen. Grob gesprochen summieren RBF-Netze gestreckte beziehungsweise gestauchte und verschobene Gaußglocken auf, um eine Funktion zu approximieren. Im Folgenden soll kurz auf RBF-Netze eingegangen werden, da sie genau wie die zuvor vorgestellten Multilayer Perceptrons allgemeine Funktionsapproximatoren sind, die sich zur Verwendung bei der Integration anbieten.

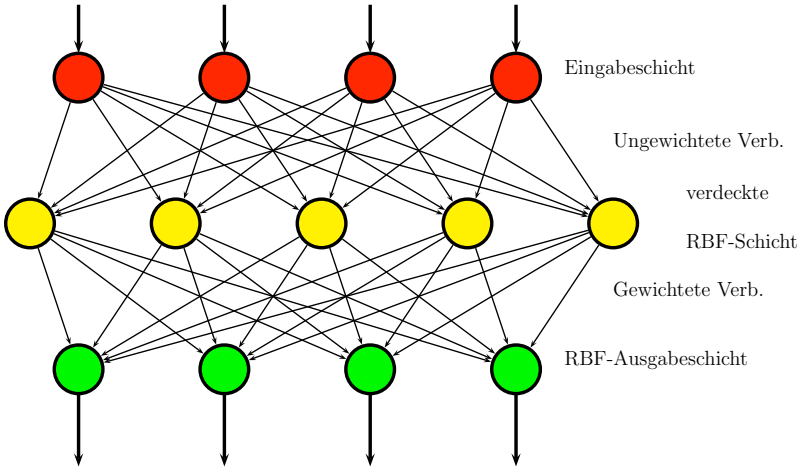


Abbildung 7.10: Aufbau eines RBF-Netzes

In Abbildung 7.10 ist der grundlegende Aufbau eines RBF-Netzes zu sehen, die Verbindungen der ersten zur zweiten Schicht sind ungewichtet, die Verbindungen von der RBF-Schicht zur Ausgabeschicht sind gewichtet. Während sich die Eingabeneuronen nicht von denen eines Perceptrons unterscheiden, gibt es bei den Neuronen der verdeckten Schicht Unterschiede:

Ein RBF-Neuron, also ein Neuron der versteckten Schicht in einem RBF-Netz, berechnet mit einer Norm, der Propagierungsfunktion des Neurons, den Abstand der Eingabe zu seinem Zentrum. Die so berechnete Netzeingabe wird in eine radiale Funktion f_{act}, beispielsweise eine Gaußglocke, eingesetzt, die die Ausgabe des Neurons berechnet. Das Zentrum des Neurons ist der Punkt, an dem das Neuron im Eingaberaum liegt. Ein RBF-Netz summiert skalierte und verschobene Gaußglocken auf und hat deshalb immer eine Ausgabe ≥ 0, was wünschenswert für ein Netz, das zur Integration eingesetzt wird, ist.
RBF-Ausgabeneuronen besitzen die gewichtete Summe als Propagierungsfunktion und die Identität als Aktivierungsfunktion. Ihre Aufgabe besteht also lediglich darin, die Eingabe gewichtet aufzusummieren und wieder auszugeben. Ein BIAS-Neuron wird bei RBF-Netzen nicht verwendet.

8 Implementierung

Der Ablauf des in dieser Arbeit beschriebenen Verfahrens wird im Folgenden an einem Stück beschrieben, dabei wird auf die Implementierung in MATLAB und auf den Ablauf der WEB-Methode eingegangen.
Die WEB-Methode kann grob in drei Teile unterteilt werden: Die Klassifikation der Zellen und der B-Splines, das Aufstellen des Galerkin-Systems und das Lösen des Systems. Bevor man mit der WEB-Methode jedoch starten kann, wird zunächst einmal das Gebiet aus den vorgestellten einfachen Grundobjekten zusammengesetzt.

8.1 Zell- und B-Spline-Klassifikation

Die Klassifizierung der Gitterzellen erfolgt im Dreidimensionalen analog zum Zweidimensionalen, weshalb hier der Übersichtlichkeit halber der zweidimensionale Fall dargestellt wird. Zur Bestimmung der inneren und äußeren Gitterzellen wird je eine Matrix mit den Koordinaten der Schnittpunkte der Gitterlinien gebildet und in die Gewichts- beziehungsweise Randfunktion des Gebietes eingesetzt. Mit einem einfachen Vorzeichentest wird der Funktionswert in der daraus entstandenen Matrix überprüft und die Gitterzelle mit dem Index (k, ℓ) wird als innere Gitterzelle markiert, wenn der Matrixeintrag $M(k, \ell) > 0$ und als äußere Gitterzelle markiert, wenn $M(k, \ell) < 0$. Dadurch werden auch die eigentlichen Randzellen als innere beziehungsweise äußere Gitterzellen markiert, was aber im nächsten Schritt wieder korrigiert wird, womit die Klassifizierung der Gitterzellen abgeschlossen ist.
Im zweiten Schritt zur Klassifizierung der Gitterzellen werden die Schnittpunkte der Gitterlinien mit dem Objekt berechnet. Aufgrund der Form der Randfunktion als Verknüpfung der Funktionen der Ausgangsobjekte mit R-Funktionen ist es sinnvoll, die einzelnen Grundobjekte mit den Gitterlinien zu schneiden und das Ergebnis anschließend durch Einsetzen in die Gewichtsfunktion zu überprüfen. Ist $w(x_s, y_s) = 0$, so ist das Ergebnis auch ein Schnittpunkt des Gebietsrandes von D mit den Gitterlinien, für $w(x_s, y_s) \neq 0$ handelt es sich um einen Punkt, der durch die Verknüpfung der Objekte innerhalb beziehungsweise außerhalb von D liegt. Wird mit den Ausgangsobjekten gerechnet, so lässt sich Rechenaufwand vermeiden, indem man die Gitterlinien in die allgemeine Gleichung des Randes der Objekte einsetzt und einige Vereinfachungen schon vorweg vornimmt, bevor die konkreten Koeffizienten und Gitterwerte eingesetzt werden. Für einen zweidimensionalen Kegelschnitt mit der allgemeinen Gleichung

$$Ax^2 + By^2 + Cxy + Dx + Ey + F = 0$$

kann man durch Einsetzen der Gitterlinien und Vereinfachen die Schnittpunktberechnung auf die einfache Berechnung der Nullstellen einer univariaten quadratischen Gleichung

$$x_{1/2} = -\frac{p}{2} \pm \sqrt{\frac{p^2}{4} - q}$$

mit

$$p = \frac{Cy_g + D}{A} \qquad \text{und} \qquad q = \frac{By_g^2 + Ey_g + F}{A}$$

bringen, in diesem Fall für den Schnitt mit einer zur x-Achse parallelen Gitterlinie. Dies ist möglich, da die allgemeinen Gleichungen der Objekte bekannt sind und die Gitterlinien immer in Koordinatenrichtung verlaufen. Beim Rechnen mit der Gewichtsfunktion von D wäre dies nicht möglich, da diese abhängig von der Anzahl der Ausgangsobjekte und der Art der Verknüpfungen eine andere Gestalt aufweist.
Während der Klassifikation der Zellen können weitere Informationen gespeichert werden, die eventuell später benötigt werden. Beispielsweise kann für jede Randzelle das oder die Objekte gespeichert haben, die zum Randstück in dieser Zelle beitragen. Damit lässt sich direkt die Tangente an der Kurve angeben, wie in Kapitel 5 beschrieben.
Sind die Schnittpunkte berechnet, werden für jeden Schnittpunkt die Gitterzellen, auf deren Rand er liegt, als Randzellen markiert. Es gilt lediglich darauf zu achten, dass diejenigen Zellen richtig markiert werden, deren Zellecken ein Schnittpunkt mit ∂D sind. Für alle Extrempunkte eines Objektes oder Schnittpunkte von mindestens zwei Objekten, die auf ∂D liegen, ist noch zu prüfen, ob die entsprechenden Zellen, die sie enthalten, als Randzellen markiert sind. Es kann vorkommen, dass eine Zelle einen solchen Punkt enthält, aber bisher nicht als Randzelle markiert wurde.
Bei der Verwendung einer geglätteten Gewichtsfunktion sind zusätzlich die sogenannten Offset-Zellen zu markieren. Dies sind innere Zellen Q mit $d(Q, \partial D) < \delta$ und sie werden markiert, da die numerisch ausgewertet werden müssen. Bei allen anderen inneren Zellen ist die Gewichtsfunktion identisch 1.
Die Klassifikation der Tensorprodukt-B-Splines schliesst sich direkt an. Hier werden alle B-Splines $b^n_{k,h}$ als innere B-Splines gekennzeichnet, deren Träger supp $b^n_{k,h}$ mindestens eine innere Zelle enthalten. Enthält der Träger nur äußere und Randzellen, so wird der B-Spline als äußerer B-Spline markiert, B-Splines, deren Träger nur äußere Zellen enthalten, sind für die Lösung nicht relevant.

8.2 Aufstellen des Ritz-Galerkin-Systems

Zum Aufstellen des Ritz-Galerkin-Systems müssen für das Modellproblem 2.2 die folgenden Integrale über allen inneren Zellen und Randzellen berechnet werden:

$$\begin{aligned} g_{k,\ell} &= \int_D \operatorname{grad} B^n_{k,h} \cdot \operatorname{grad} B^n_{\ell,h} dx \\ f_k &= \int_D f\, B^n_{k,h} dx. \end{aligned}$$

Über den inneren Zellen kann direkt mit der Tensorproduktformel integriert werden, wie sie im Kapitel über Integration vorgestellt wurde. Die Randzellen müssen vorher entsprechend mit den vorgestellten Verfahren in Teilgebiete unterteilt werden. Während die Triangulierung im Zweidimensionalen recht problemlos verläuft gestaltet es sich im Dreidimensionalen etwas zeitaufwändiger.

Eine Möglichkeit, dies wieder etwas zu beschleunigen ist es, die auftretenden Fälle der Randzellen und deren Tetraedisierungen zu untersuchen und in Typen einzuteilen. Diese Typen können in Testläufen in einer Liste gespeichert werden, so dass sie später nur noch aus der Liste ausgelesen werden können und lediglich die entsprechenden Koordinaten der Punkte eingesetzt werden müssen. Zur Verbesserung des Ablaufes ist hier also ein Lernprozess notwendig, der mit Trainingsbeispielen durchgeführt wird, um die auftretenden Fälle zu untersuchen. Dies ist allerdings ein anderer Lernprozess als bei den neuronalen Netzen im vorigen Kapitel, da hier eine Wissensdatenbank in Form einer Liste erstellt wird. Diese Wissensdatenbank ist auch dann noch von Nutzen, wenn neue Objekttypen hinzugefügt werden und muss lediglich erweitert werden. Das Wissen ist in einer Form speicherbar, die überprüfbar und einsehbar ist und in der man nach Fehlern suchen kann.
Über den entstandenen Dreiecken und Tetraedern kann integriert werden, wie im vorigen Kapitel beschrieben. Umgesetzt wurde die gesamte Integration mit einer Schleife über die inneren Zellen und Randzellen. Mit Ausnahme der Randzellen, bei denen die einzelnen Integrale über die entsprechenden Dreiecke in jedem Schritt aufsummiert werden, wurde sich bei der Implementierung der Integration und der folgenden Schritte nach [30] gerichtet und analog verfahren. Aus diesem Grund wird hier nicht zu sehr ins Detail gegangen und lediglich der grobe Ablauf dargestellt.
Für das Aufstellen des Ritz-Galerkin-Systems werden zunächst die Räume der gewichteten Splines betrachtet. Das bedeutet, es werden für alle inneren Zellen und Randzellen Q_i die Integrale

$$\alpha_{i,k,\ell} = \int\limits_{Q_i \cap D} \operatorname{grad}(wb^n_{k,h}) \cdot \operatorname{grad}(wb^n_{\ell,h}) dx \qquad \text{und} \qquad \beta_{i,k} = \int\limits_{Q_i \cap D} f wb^n_{k,h} dx$$

berechnet. Daraus ergeben sich die Einträge des Ritz-Galerkin-Systems $\tilde{G}\tilde{C} = \tilde{F}$ zur Basis des WB-Raumes

$$\tilde{g}_{k,\ell} = \sum_{Q_i \subset \operatorname{supp}\ b^n_{k,h} \cap \operatorname{supp}\ b^n_{\ell,h}} \alpha_{i,k,\ell} \qquad \text{und} \qquad \tilde{f}_k = \sum_{Q_i \subset \operatorname{supp}\ b^n_{k,h}} \beta_{i,k}.$$

Durch eine einfache Transformation lässt sich dieses System in das Ritz-Galerkin-System zur WEB-Basis transformieren. Dies stellt eine Vereinfachung der Programmstruktur für die Implementierung dar. Die Transformationsmatrix erhält man nach folgendem Lemma [30]:

Lemma 8.1 *Bezeichne $\tilde{G}\tilde{C} = \tilde{F}$ das Ritz-Galerkin-System zur Gitterweitte h für die WB-Basis und $GC = F$ das System auf demselben Gitter zur WEB-Basis, dann gilt*

$$G = T\tilde{G}T^t \qquad \textit{und} \qquad F = T\tilde{F}$$

mit der Transformationsmatrix $T = (t_{i,k}), i \in I^n_{D,h}, k \in K^n_{D,h}$,

$$t_{i,k} := \begin{cases} (w(x_i)h)^{-1} & \textit{für } k = i, \\ (w(x_i)h)^{-1} e_{i,j} & \textit{für } k = j \in J^n_{D,h}(i), \\ 0 & \textit{sonst.} \end{cases}$$

Hierbei bezeichnen $w(x_i)$ und $e_{i,j}$ die aus der Konstruktion der WEB-Basis gemäß Definition 4.5 bekannten Gewichte und Erweiterungskoeffizienten.

8.3 Lösen des Ritz-Galerkin-Systems

Zur Lösung des Ritz-Galerkin-Systems bieten sich verschiedene Möglichkeiten an. Eine bezüglich der Laufzeit sehr gute Methode mit allerdings erhöhtem Speicher- und Implementierungsaufwand aufgrund von Gitterdaten und Ritz-Galerkin-Systemen für mehrere Gitterebenen ist das Lösen mit einem Mehrgitterverfahren [19]. Bei dieser Methode kann man einen Vorteil daraus ziehen, wenn man bei der Implementierung, wie oben beschrieben, zuerst das Ritz-Galerkin-System zur WB-Basis aufstellt und anschliessend in das System zur WEB-Basis transformiert [30].
Eine weitere Möglichkeit stellt die Methode der konjugierten Gradienten (cg-Verfahren) dar [15], [13]. Die Lösung des Ritz-Galerkin-Systems wird hierbei als Ergebnis des Minimierungsproblems der Funktion

$$f(C) = \frac{1}{2}C^t GC - F^t C$$

in Gradientenrichtung aufgefasst werden. Mit Hilfe des G-Skalarprodukts $< \cdot, \cdot >_G$ und unter Ausnutzung der positiven Definitheit und Symmetrie von G wird das Minimierungsproblem in einer Folge von affinen Räumen gelöst. Grob gesprochen wird für die Lösung X_* eine Folge von Approximationen $X_1, X_2, \ldots$ in einer Folge von affinen Räumen mit größer werdender Dimension berechnet. Die Wahl der Räume erfolgt so, dass man möglichst schnell und mit geringem Aufwand die Lösung X_i im entsprechenden Raum erhält.
Beispielsweise mit der SSOR-Iteration lässt sich dieses Verfahren noch vorkonditionieren, um eine bessere Konvergenz zu erhalten. Man nennt dies dann PCG (preconditioned conjugate gradients).

8.4 Testbeispiele

Zum Abschluß dieses Kapitels werden Testbeispiele der im Rahmen dieser Arbeit vom Autor implementierten Methoden dargestellt.
Als erstes Testgebiet wird das im Laufe dieser Arbeit häufiger verwendete Gebiet das noch einmal in Abbildung 8.1 links zu sehen ist verwendet.

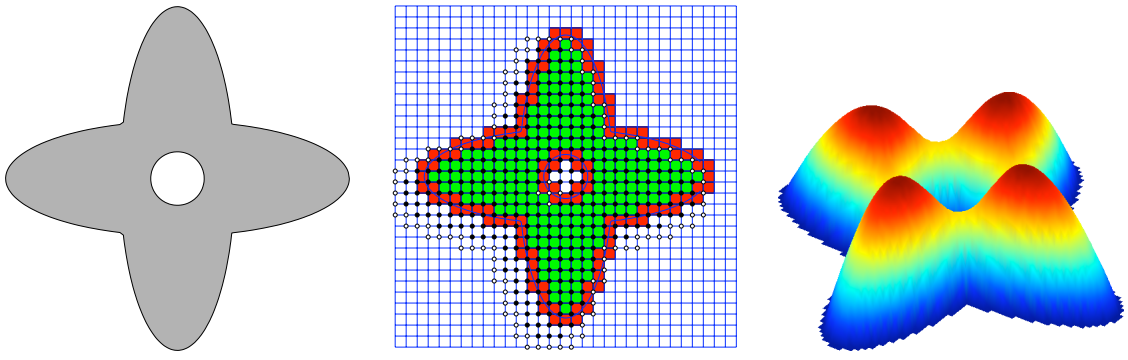


Abbildung 8.1: Links: Beispielgebiet. Mitte: Klassifikation der Zellen und B-Splines. Rechts: Lösung des Modellproblems

Die Klassifizierung der Zellen ergibt 200 innere Zellen und 120 Randzellen, die anschließend in 250 Dreiecke unterteilt werden. Die Einteilung der quadratischen B-Splines ergibt 117 äußere und 320 innere B-Splines.
Die Lösung in der Abbildung rechts zu sehen wurde in 0.037 Sekunden mit 55 Iterationen des preconditioned configured gradients Verfahrens mit $f \equiv 1$ als rechter Seite berechnet.

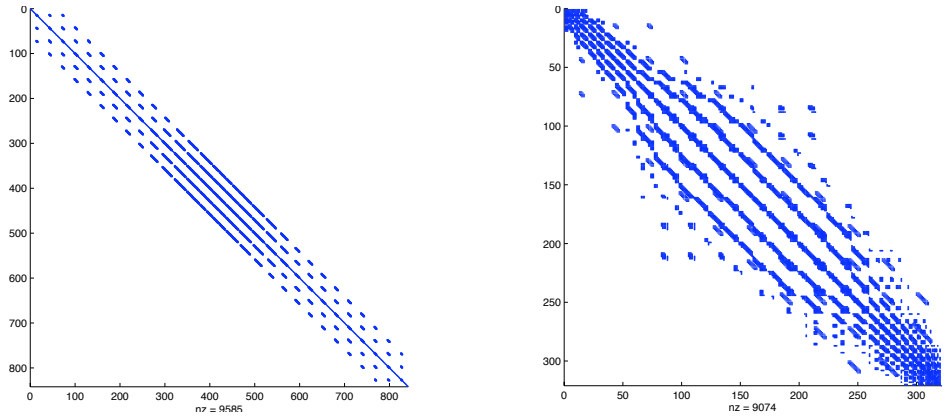


Abbildung 8.2: Belegungsstruktur der Ritz-Galerkin-Matrix des nicht erweiterten (links) und des erweiterten Systems (rechts)

Die Belegungsstruktur der Ritz-Galerkin-Matrizen der nicht erweiterten und des erweiterten Systems ist in Abbildung 8.2 zu sehen.
Als erster einfacher Fall wird ein einfaches dreidimensionales Objekt, die Vereinigung von zwei Kugeln mit dem Ursprung und dem Punkt $(1, 2, 0)$ als Mittelpunkt und jeweils dem Radius 2, erzeugt und über diesem Gebiet das Modellproblem berechnet, um Unterschiede zum Zweidimensionalen zu veranschaulichen. Das Gebiet wird in $12 \times 14 \times 10$ Zellen unterteilt mit 277 Randzellen und 484 inneren Zellen.

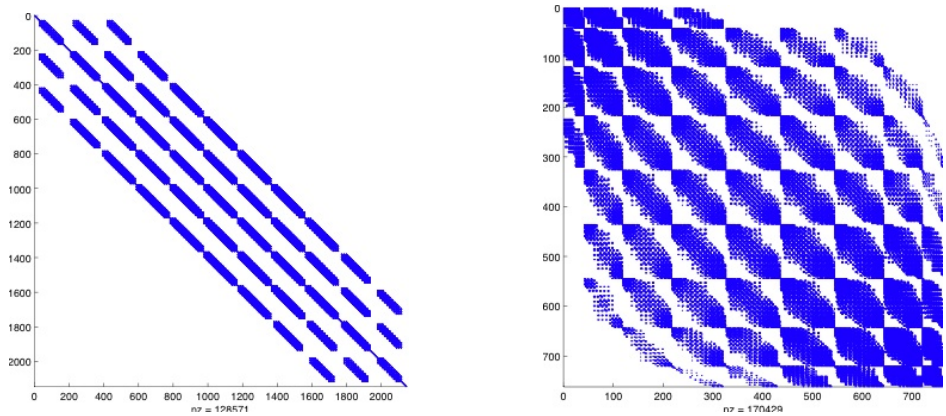


Abbildung 8.3: Belegungsmatrizen der Ritz-Galerkin-Matrix des nicht erweiterten Systems (links) und des erweiterten Systems (rechts)

Beim Aufstellen des Ritz-Galerkin-Systems ist zu beachten, dass für jede Zelle mehr Integrale berechnet werden müssen, da eine Zelle im Dreidimensionalen mehr benachbarte Zellen besitzt als im Zweidimensionalen und deshalb auch mehr B-Splines existieren, deren Träger die entsprechende Zelle enthalten. Aus diesem Grund sind in der erweitertne Matrix auch deutlich mehr Einträge ungleich 0 als im Zweidimensionalen. Die Belegungsmatrizen des Beispiels sind in Abbildung 8.3 zu sehen.

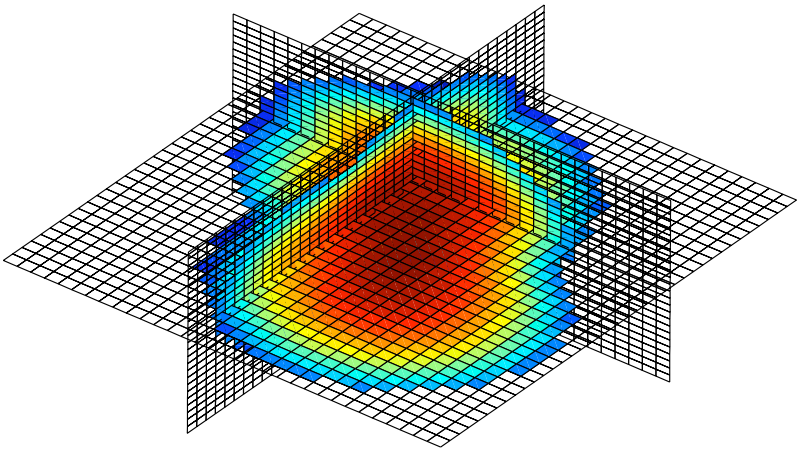


Abbildung 8.4: Lösung des Modellproblems auf dem dreidimensionalen Beispielgebiet

Die Schnitte mit den Koordinatenebenen der Lösung des Modellproblems analog zum zweidimensionalen Beispiel sind in Abbildung 8.4 zu sehen.
In Tabelle 8.1 sind die verschiedenen Schritte des Lösens des Gleichungssystems im Zweidimensionalen und Dreidimensionalen gegenübergestellt.

	$2d$	$3d$
Anzahl innere Zellen	200	277
Anzahl Randzellen	120	484
Lösen des Gleichungssystems	0.0699 sek	2.4339 sek
Iterationen des pcg Lösers	35	461
Dimension der Matrix	1089	2145
Lösen des erweiterten Gleichungssystems	0.0367 sek	3.2519 sek
Iterationen des pcg Lösers	55	529
Dimension der erweiterten Matrix	320	761

Tabelle 8.1: Gegenüberstellen der einzelnen Schritte des Lösungsvorgangs

Anhand der Daten in der Tabelle lassen sich Unterschiede zwischen den Fällen im Zweidimensionalen und Dreidimensionalen sehen. Die Anzahl der Randzellen bei solchen einfachen kompakten Gebieten ist im Dreidimensionalen deutlich höher, was sich auf die Erweiterung der Matrix auswirkt. Die Dimension der Matrizen und somit auf die Anzahl

der nötigen Iterationen des Lösers steigt im Dreidimensionalen, da schon bei einfachen Beispielen mehr Zellen auftreten.
Eine Anwendungsmöglichkeit der hier dargestellten Methoden wird im folgenden dargestellt und für das zweidimensionale Beispielgebiet und ein dreidimensionales Gebiet berechnet.

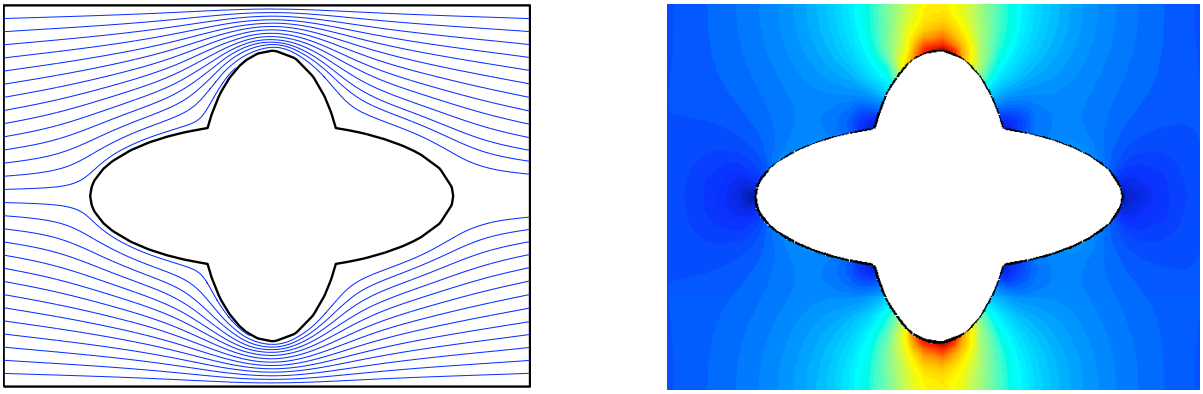


Abbildung 8.5: Strömungslinien (links) und Strömungsgeschwindigkeit (rechts)

Ein reibungsfreies Strömungsfeld einer inkompressiblen Flüssigkeit lässt sich über einem Gebiet D ohne Quellen mit der Laplace-Gleichung mit Neumann-Randbedingungen modellieren. Das bedeutet, dass für die Potentialfunktion u

$$\begin{aligned} -\Delta\, u &= 0 \qquad \text{in } D \\ \frac{\partial u}{\partial n} &= g \qquad \text{auf } \partial D \end{aligned}$$

gilt.
Zusätzlich müssen die Kompatibilitätsbedingung $\int\limits_{\partial D} g ds = 0$ erfüllt sein [16], damit die Lösung eindeutig ist die Forderung $\int\limits_{D} u = 0$.
Eine Gewichtsfunktion ist hier nicht notwendig, weshalb erweiterte B-Splines verwendet werden oder WEB-Splines mit $w \equiv 1$. Bei der Implementierung kann hier somit wiederum Laufzeit eingespart werden, indem man für alle inneren Zellen auf einmal berechnete Werte in einer Tabelle zurückgreift. Bei den Randzellen ist dies nicht möglich, hier mßsen weiterhin Integrale berechnet werden.
Für das Beispielgebiet sind die Strömungslinien $\operatorname{grad} u = c$ und die Strömungsgeschwindigkeit $v = \| \operatorname{grad} u \|$ in Abbildung 8.5 zu sehen. Der ausgeschnittene Kreis in der Mitte des Gebietes ist nicht zu sehen, da die Strömung ihn nicht erreicht und er deshalb nicht ins Gewicht fällt. Die Flüssigkeit fliesst am linken Rand parallel zu den beiden waagrechten Rändern ein und hat am linken Rand die Eintrittsgeschwindigkeit $v_0 = -3$ und am rechten Rand die Austrittsgeschwindigkeit $v_0 = 3$. Die Flussgeschwindigkeit ist an den blauen Stellen am niedrigsten und ist an den roten Stellen direkt am Gebietsrand oben und unten am schnellsten, dort sind deshalb auch die Strömungslinien am dichtesten.

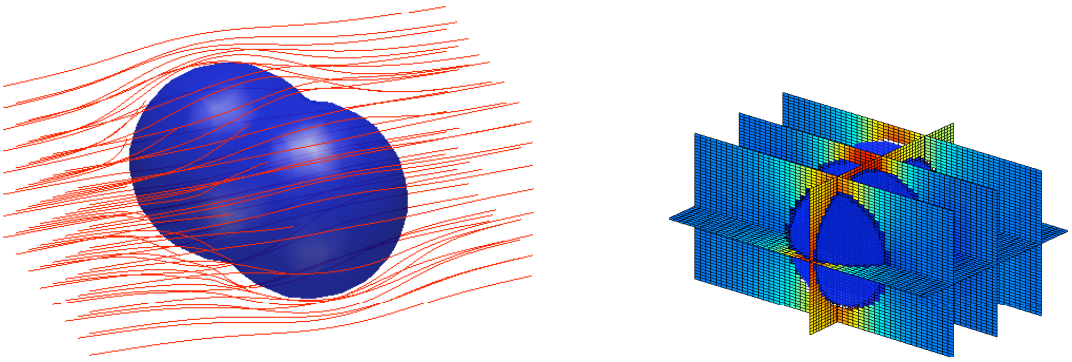


Abbildung 8.6: Strömungslinienfeld (links) und Querschnitt entlang der Koordinatenachsen der Strömungsgeschwindigkeit (rechts)

Analog dazu ist in Abbildung 8.6 das Strömungslinienfeld und ein Schnitt entlang der Koordinatenebenen durch das Bild der Strömungsgeschwindigkeit für das dreidimensionale Beispielobjekt zu sehen. Die Strömung fliesst wieder von links nach rechts durch ein Rohr mit rechteckigem Querschnitt mit $v_0 = \pm 3$.

9 Zusammenfassung

Die WEB-Methode ist ein neues vernetzungsfreies Finite Elemente Verfahren, das die Vorteile konventioneller gitterbasierter Finite Elemente und uniformer B-Splines verbindet [18]. Entscheidend für eine effiziente Realisierung sind die Konstruktion von Gewichtsfunktionen und die Berechnung von Randintegralen. Letztere sind Gegenstand dieser Dissertation. Es werden Methoden entwickelt, um Integrale der Form

$$\begin{aligned} g_{k,\ell} &= \int_D \operatorname{grad} B^n_{k,h} \cdot \operatorname{grad} B^n_{\ell,h} dx \\ f_k &= \int_D f\, B^n_{k,h} dx. \end{aligned}$$

zu berechnen, wobei Q eine Gitterzelle, das heißt ein Quadrat im Zweidimensionalen und ein Würfel im Dreidimensionalen, und D ein durch Quadriken berandetes Gebiet ist, wie es etwa in CSG-Modellen auftreten kann.
Die Grundidee ist einfach: Das Innere der Randzellen, also $Q \cap D$, wird trianguliert, das bedeutet in Dreiecke beziehungsweise Tetraeder unterteilt, über denen mit Gauß-Quadraturformeln integriert wird. Dies liefert eine stückweise lineare Approximation des Randes. Ist eine genauere Approximation gewünscht, so werden isoparametrische Elemente, krummlinig berandete Dreiecke beziehungsweise krummflächig berandete Tetraeder, verwendet.
Der Schlüssel zu sehr effizienten Algorithmen liegt jedoch in der Ausnutzung der speziellen Struktur. Zur Berechnung der Schnittpunkte von Quadriken eignet sich besonders der K-Form Algorithmus von J.Koch [21], der nichtlineare polynomiale Gleichungssysteme durch Substitution von Variablen in äquivalente Gleichungssysteme überführt, in denen die Nichtlinearität höchstens quadratisch auftritt, da die impliziten Gleichungen der Quadriken sehr schnell durch wenige Schritte auf K-Form gebracht werden können.
Betrachtet man für eine klein genug gewählte Gitterweite h die Schnitte der zu einem Objekt kombinierten Quadriken, so lässt sich im Zweidimensionalen eine recht geringe Anzahl von möglichen Schnittfällen feststellen. Im Dreidimensionalen ist dies anders. Die vom Autor durchgeführte Untersuchung der möglichen Fälle für maximal zwei Ebenen, die eine Randzelle schneiden, ergab schon eine recht große Anzahl von Möglichkeiten, die sich noch weiter erhöht, wenn man maximal drei Objekte pro Randzelle zulässt und Quadriken in die Untersuchung mit einbezieht. Eine Klassifizierung dieser Schnitttypen macht dann durchaus Sinn, denn man kann automatisch kanonische Tetraedisierungen für die einzelnen Typen generieren und in einer Wissensdatenbank speichern. Die Verwendung einer solchen Datenbank für die Tetraedisierung kann den Ablauf des Algorithmus deutlich beschleunigen, denn es muss nicht für jede Randzelle explizit eine Tetraedisierung berechnet werden. Selbstverständlich ist dies auch im Zweidimensionalen möglich, die Laufzeiteinsparung ist allerdings nicht wesentlich geringer. Die vom Autor entwickelten Algorithmen

zur Triangulierung beziehungsweise Tetraedisierung liefern für die einzelnen Fälle schnell eine Unterteilung des entsprechenden Teils der Randzellen.
Liefern linear berandete Elemente nicht die gewünscht gute Approximation, so kann, wie bereits erwähnt, eine bessere Approximation durch die Verwendung von isoparametrischen Elementen verwendet werden. Durch Bézier-Techniken, die vermutlich noch nicht in diesem Bereich eingesetzt wurden, lässt sich hier eine sehr gute Approximationsordnung schon für quadratische isoparametrische Abbildungen erzielen. Hierzu wird ein geometrischer Zugang gewählt, der die Konstruktion der zusätzlich für die isoparametrische Abbildung benötigten Kontrollpunkte, hier im Zweidimensionalen, anschaulich darstellt.
Für jede Kante, die auf eine krummlinige Kante abgebildet wird, gilt, dass beide Ecken auf dem Gebietsrand ∂D liegen. Zur Konstruktion des dritten für diese Kante notwendigen Punktes berechnet man den Schnittpunkt der Tangenten an die Randkurve in den beiden Eckpunkten. Man erreicht hiermit eine Genauigkeit von $\mathcal{O}(h^4)$.
Zur Berechnung der benötigten Integrale verwendet man Techniken der numerischen Integration, genauer Tensorprodukt-Gauß-Formeln, da eine analytische Lösung der Integrale, sofern sie überhaupt existiert, meist nur sehr schwer zu berechnen ist. Das bedeutet, dass das Integral durch eine endliche Summe

$$\int_a^b f(x)dx \approx Q_N(f) = \sum_{i=1}^{N} w_i f(x_i)$$

mit Stützstellen x_i und Gewichten w_i approximiert wird. Diese Gauß-Formeln können durch das Bilden von Tensorprodukten in beliebige Dimensionen verallgemeinert werden. Für Dreiecke und Tetraeder existieren spezielle Formeln, die mit weniger Parametern auskommen, allerdings ist hier die Verallgemeinerung auf weitere Dimensionen nicht möglich.
Ein neuer und vollständig anderer Ansatz, der vom Autor dieser Arbeit erarbeitet wurde, ist die Verwendung eines neuronalen Netzes zur Berechnung der Integrale über den Randzellen. Dies bringt einige Vorteile mit sich, allerdings auch Nachteile. Ein Problem besteht darin, ein entsprechendes neuronales Netz zu erstellen und zu trainieren. Es gibt keine festen Regeln, nach denen sich das ideale neuronale Netz erstellen lässt, einige Entscheidungen lassen sich nur durch das Trial-and-Error-Prinzip wählen. Einmal erstellt und ausreichend trainiert ist das neuronale Netz vor allem für eine parallele Implementierung ein großer Vorteil, da es aufgrund seiner Struktur ideal für parallele Hardware ausgelegt ist.
Kapitel 8 widmet sich der Implementierung der in dieser Arbeit vorgestellten Methoden und Algorithmen in MATLAB. Die Implementierung lässt sich grob in drei Abschnitte unterteilen: die Zell- und B-Splineklassifikation, das Aufstellen des Ritz-Galerkin-Systems und das Lösen des Ritz-Galerkin-Systems. Bei der Zellklassifikation werden zwei Schritte durchgeführt. Mit Einsetzen der Gittereckpunkte in die Gewichtsfunktion und einem einfachen Vorzeichentest lassen sich die Zellen in innere und äußere Zellen einteilen. Anschließend werden die Schnittpunkte der Zellkanten mit dem Objekt verrechnet. Hier zeigt sich direkt ein Vorteil der Verwendung von Quadriken als Grundobjekte, denn die Berechnung der Schnittpunkte mit den einzelnen Quadriken gestaltet sich deutlich einfacher als mit der gesamten Gewichtsfunktion. Durch ein wenig vorweggenommene Kombinatorik lassen sich die Berechnungen auf das Lösen quadratischer Gleichungen reduzieren, und es

muss lediglich im Anschluss geprüft werden, ob der Punkt auf ∂D liegt oder nicht. Darauf folgt die Einteilung der B-Splines in innere B-Splines mit mindestens einer inneren Zelle im Träger und äußere B-Splines, deren Träger nur äußere und mindestens eine Randzelle enthalten.
Für das Aufstellen des Ritz-Galerkin-Systems empfiehlt es sich, zunächst das System zur Basis des WB-Raumes aufzustellen und das erhaltene System anschließend zu einem System zur WEB-Basis zu transformieren.
Zur Lösung der Systems bieten sich mehrere Möglichkeiten, beispielsweise Mehrgitter-Verfahren [19] oder (vorkonditionierte) konjugierte Gradienten Methoden [15], [13].

10 Summary

In the first chapter an overview of the theory of finite elements method is given as far as it is needed for this work. With the method of finite elements, elliptical differential equations can be solved numerically. A linear differential equation with constant coefficients $a_{i,j}, b_i \in \mathbb{R}$ is given by

$$-\sum_{i,j=1}^{n} a_{i,j} \frac{\partial^2 u}{\partial x_i \partial x_j} + \sum_{i=1}^{n} b_i \frac{\partial u}{\partial x_i} + cu = f,$$

with an arbitrary function $f : D \to \mathbb{R}$ over a region D. It is supposed that the matrix of the coefficients A is symmetric, i.e., $a_{i,j} = a_{j,i}$, which is easily achieved by substituting the partial derivatives. This differential equation is called elliptic if A has only positive eigenvalues. It is written as

$$Lu = f$$

with an accordingly defined differential operator. Usually these differential equations do not have a unique solution. Therefore, constraints are required. The most popular constraints are

- Dirichlet boundary condition: $u = g(x)$ on the boundary ∂D of D and
- Neumann boundary condition: $\frac{\partial u}{\partial n} = g(x)$ on ∂D, where $\frac{\partial u}{\partial n}$ is the derivative in perpendicular direction.

The basics for solving elliptic differential equations are discussed in the first chapter and the Poisson's equation with homogeneous Dirichlet boundary conditions is introduced as the model problem:

$$\begin{aligned} -\Delta u &= f \ in \ D \\ u &= 0 \ on \ \partial D. \end{aligned}$$

Sobolev spaces are the appropriate choice as function spaces for approximating the solution. The solution is obtained with the Ritz-Galerkin approximation by solving the linear system

$$\sum_{j=1}^{n_h} a(b_j, b_i) c_i = l(b_i), \qquad i = 1, \ldots, n_h.$$

It is obvious to use polynomials for approximating given data, but changes on one knot have global influence. So another way has to be found. A good choice are the piecewise polynomial functions, the so-called spline functions. Roughly spoken, a spline of degree n consists of piecewise consecutive polynomial segments of degree n that are $(n-1)$-times differentiable at the knots.

Splines have better approximation properties than polynomials and are easier to present and to handle mathematically.
This definition of splines is not appropriate for computations. So it makes sense to construct a local basis to approach the theory and the numerical computations. Based on the characteristic function b^0 of the unit interval $[0, 1)$, B-splines are defined by

$$b^n(x) = \int_{x-1}^{x} b^{n-1}$$

as the desired basis functions.
For a grid width h and $k \in \mathbb{Z}$, the B-splines of degree n for the knot kh is defined as

$$b^n_{k,h}(x) = b^n(x/h - k).$$

The most important properties of B-splines shown in chapter 3 are

- positivity,
- local support of b^n,
- smoothness, b^n is $(n-1)$-times continuously differentiable, and
- the derivative of a B-splines of degree n is a linear combination of B-splines of degree $n-1$.

Furthermore, the B-splines $b^n_{k,h}$ with $h > 0$ and the knot sequence $k\mathbb{Z}$ are linearly independent.
For computations the following definition of B-splines is more adequate:

$$b^n(x) = \frac{x}{n} b^{n-1}(x) + \frac{n+1-x}{n} b^{n-1}(x).$$

Based on this, spline functions can be defined as

$$s(x) = \sum_{k \in K^n_{D,h}} c_k b^n_{k,h}(x),$$

with

$$K^n_{D,h} = \{k \in \mathbb{Z} : \text{supp } b^n_{k,h} \cap D \neq \emptyset\}$$

and an open interval $D \subseteq \mathbb{R}$. For positive c_k the positivity of s follows from the positivity of the B-splines.
Because of the local support of the B-splines, changes of the coefficients have only local influence.
Based on these univariate B-splines multivariate B-splines can be easily constructed by forming tensor products. For $h > 0$, $m \in \mathbb{N}$, $x \in \mathbb{R}^m$, $n \in \mathbb{N}^m$ and $k \in \mathbb{Z}^m$ the m-variate B-spline $b^n_{k,h} : \mathbb{R}^m \to \mathbb{R}$ of degree n_ℓ in the ℓ-th variable with index k and grid width h is defined as

$$b^n_{k,h}(x) = \prod_{\ell=1}^{m} b^{n_\ell}_{k_\ell,h}(x_\ell).$$

If $m = 1$, the univariate case is obtained.
The first partial derivatives of m-variate B-splines are computed as a difference of B-splines of lower degree,

$$\partial^\alpha b^n_{k,h}(x) = \frac{1}{h}\left(b^{n-\alpha}_{k,h}(x) - b^{n-\alpha}_{k+\alpha,h}(x)\right),$$

where α is a unit vector of $\mathbb{R}^n$. Higher derivatives are obtained by iteratively computing the partial derivatives.
Again m-variate spline functions are computed as linear combinations of B-splines. For a domain $D \subseteq \mathbb{R}^m$ and the set of indices

$$K^n_{D,h} = \{k \in \mathbb{Z}^m : \text{supp } b^n_{k,h} \cap D \neq \emptyset\}$$

of all B-splines $b^n_{k,h}$ that do not vanish on D, the m-variate spline over D is defined by

$$s(x) = \sum_{k \in K^n_{D,h}} c_k b^n_{k,h}(x)$$

with $c_k \in \mathbb{R}$. Unlike the univariate case in more dimensions there may exist splines where only a small part of the support lies in D.
WEB-Splines introduced by Höllig, Reif and Wipper [18] are subject of the next chapter. B-splines do not seem to be a good choice as basis functions for finite element methods because of their lack of flexibility to statisfy boundary conditions. To achieve the boundary conditions the B-splines are multiplied with a weight function of the domain D.
In this work, weight functions are constructed using quadrics as primitive objects and combining them via R-functions introduced by Rvachev. R-functions are functions that combine the weight functions of the primitive objects in such a way that the resulting function again is a weight function, i.e. it is strictly positive inside the domain D, zero at ∂D, and strictly negative outside of D. For every set operation, there exists a corresponding R-function as seen in table 10.1.

set operation	*corresponding R-function*
$D_1 \cap D_2$	$r_\cap(x) = x_1 + x_2 - \sqrt{x_1^2 + x_2^2}$
$D_1 \cup D_2$	$r_\cup(x) = x_1 + x_2 + \sqrt{x_1^2 + x_2^2}$
D^c	$r_c(x) = -x$

Table 10.1: Set operations and their corresponding R-functions

For quadrics a very elegant representation is chosen: Every two-dimesional quadric with the implicit representation

$$ax^2 + 2bxy + cy^2 + 2dx + 2ey + f = 0$$

can be written as

$$(x,y,w)\begin{pmatrix} a & b & d \\ b & c & e \\ d & e & f \end{pmatrix}\begin{pmatrix} x \\ y \\ w \end{pmatrix} =: (x,y,w)\ Q\ (x,y,w)^t = 0$$

where (x,y,w) are the homogeneous coordinates of a point in $\mathbb{R}^2$. Quadrics in $\mathbb{R}^3$ can be represented in the same way with a 4×4-matrix Q. A big advantage of quadrics is that their equations are already weight functions. So, no weight function has to be computed and they can be combined directly using R-functions. Further advantages are the possibility of preprocessing combinatorial steps because their general equations are known and the fact that these equations have nearly K-form. So, the K-form algorithm by J.Koch [21] is a good choice for computing intersection points of the quadrics.

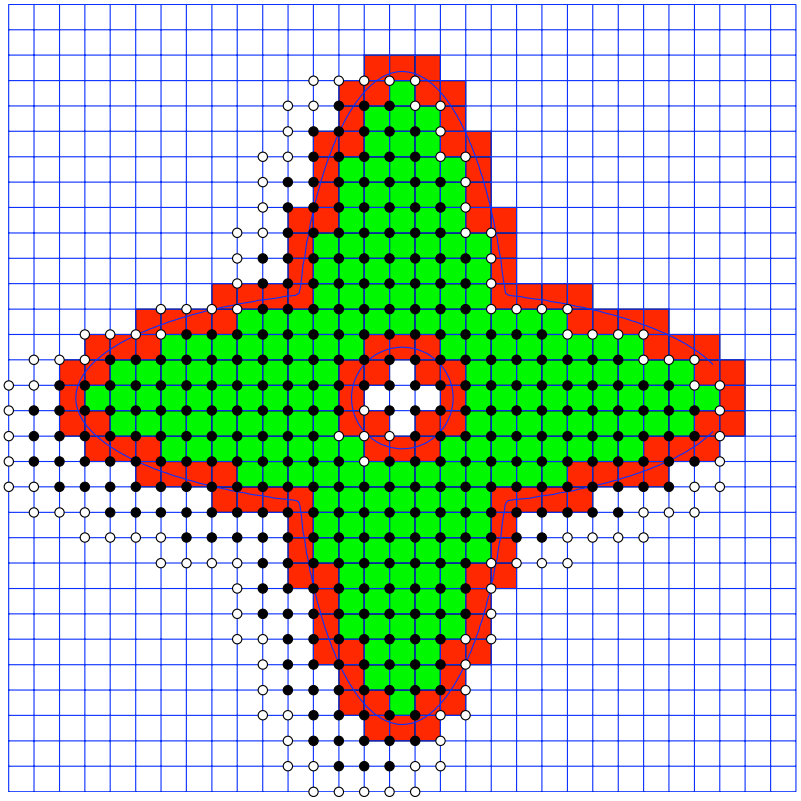


Figure 10.1: Classification of grid cells and B-splines

The WEB-method requires classification of the grid cells in inner, outer, and boundary cells and classification of the B-splines in inner B-splines with at least one inner cell in the support and outer B-splines whose supports contain only outer and boundary cells. A classification of the grid cells and B-splines is shown in figure 10.1 where inner grid cells are green, boundary cells red, and outer cells white. B-splines are marked with a black dot for inner B-splines and a white dot for outer B-splines in the lower left corner of their support.

Integrating over inner cells is easily done via tensor product Gauss quadrature formulas for the unit square or the unit cube respectively. This is not possible for boundary cells in such an easy way because it is integrated only over the inner part of the cell. In this work, the inner part of the boundary cells is subdivided in triangles or tetrahedra, respectively. Over these simplices it is possible to integrate using Gauss formulas. As a side effect, this leads to a piecewise linear approximation of ∂D. Easy algorithms for creating the triangles or tetrahedra are given in chapter 6.
First, the two-dimensional case is handled and several possibilities for creating triangles in the boundary cell are shown. Assuming that the grid width h is small enough so that at most two objects intersect a boundary cell, only a few cases of boundary cell intersections occur. This is different in the three-dimensional case and a detailed study of the occuring cases with only one or two linear objects in one boundary cell shows the complexity of this problem in three dimensions.
The tetrahedra are created by a relatively easy algorithm based on the algorithms for triangulating the cells in two dimensions. For every plane of the cell cube a triangulation of the inner part is created and all these triangles are completed to tetrahedra by connecting the vertices with a specially computed point inside the cell.
Setting up the Ritz-Galerkin-system for the model problem the integrals

$$\begin{aligned} g_{k,\ell} &= \int_D \operatorname{grad} B^n_{k,h} \cdot \operatorname{grad} B^n_{\ell,h} dx \\ f_k &= \int_D f\, B^n_{k,h} dx. \end{aligned}$$

have to be computed over all inner and boundary cells. As already mentioned above, it is easy to compute these for the inner cells using Gauss quadrature formulas.
Numerical integration is used because most integrals either do not have an analytical solution or it is difficult to compute them. Using techniques of numerical integration, such as the Gauss quadrature formulas used here, means that the integral is approximated via a finite sum

$$\int_a^b f(x)dx \approx Q_N(f) = \sum_{i=1}^{N} w_i f(x_i)$$

with nodes x_i and weights w_i.
It is not necessary to compute the nodes and weights for every cell. Every cell is mapped to the unit square in two dimensions and the unit cube in three dimensions, respectively, and for these the Gaussian parameters are well known and can be found in literature about numerical analysis.
For boundary cells it is not advisable to set the function to zero outside of D and use the same tensor product Gaussian formulas because of a loss in accuracy and approximation order. There are better solutions like the one used in [16]. A new approach to this is used in this work. The inner part of each cell is subdivided in triangles oder tetrahedra, respectively, like mentioned above using the algorithms presented in chapter 6. Numerical integration is possible over these triangles and tetrahedra using Gaussian quadrature again.

Integrals are additive, so

$$\int_{D\cap Q} f(x) \approx \int_{T_1} f(x) + \cdots + \int_{T_k} f(x)$$

is already a good approximation for the desired integral where $T_1, \ldots, T_k$ are all triangles or tetrahedra inside the boundary cell.
If this approximation is not sufficient, it is possible to achieve a better accuracy by using isoparametric elements.

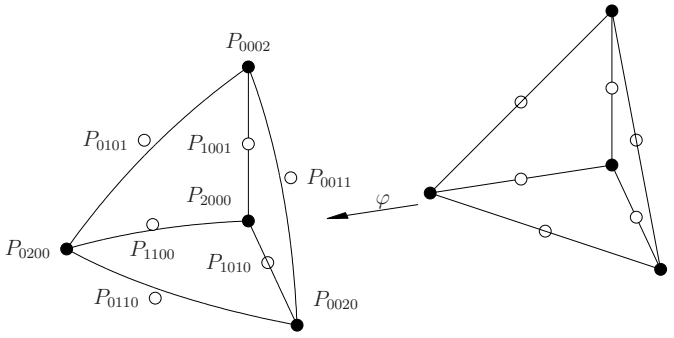


Figure 10.2: Isoparametric mapping for a tetrahedron

Figure 10.2 shows the idea behind this for a tetrahedron. Isoparametric elements are triangles or tetrahedra with curved edges next to the boundary ∂D to achieve a higher accuracy.
A completely different and new approach presented in this work is the use of a neural network for integration over the boundary cells. The idea behind neural networks is the attempt to simulate the functionality of the human brain in the computer. In the meantime, there are two categories for neural networks: neural networks to simulate the human brain and neural networks to solve technical problems. Neural networks of the second category are used here. No biological plausibility is needed here, so it was concentrated on creating and training a neural network for solving the integration problem.
Neural networks consist of simple units:

- Cells
- Connection network
- Propagations function
- Learning rule.

Using neural networks for integration is a new idea and naturally has advantages and disadvantages.

- Fault tolerance against errors of the input and even against errors in the structure of the net

- Parallelism
- Simple elements
- Ability to learn and generalise
- Adaptivity.

These are the most important advantages, but there are also some disadvantages that have to be considered:

- Knowledge is not apparent and not reviewable
- No basic knowledge
- Difficult error analysis
- Trial and error when building a new neural network.

When constructing a neural network the right setup for all components has to be found. While some choices are easy and obvious, there is no given rule to follow for other components. The following components have to be chosen:

- Number of input- and output parameters
- Number of layers
- Number of cells
- General structure and type of the neural network
- Neuronfunctions: activation function, propagation function, output function
- Starting values for the connection weights
- Type of learning and learning rule.

In chapter 8 an overview over the implementation of the methods used in this work is given. The implementation can be divided in three blocks: Cell- and B-spline-classification, setting up the Ritz-Galerkin-system, and solving the Ritz-Galerkin-system.
The classification of the grid cells works in two and three dimensions in an analogous way and is explained for two dimensions only. In the first step a matrix with the coordinates of the grid line intersections is created, and with a simple sign-test with the intersection points and the weight function the cells are classified in inner and outer cells. In the second step, the intersections of the object with the grid lines are computed and at the same time the cells that have been wrongly marked as inner or outer cells are marked as boundary cells. To simplify the second step intersections of each quadric and the grid lines are computed and tested with the weight function since it is much easier and some combinatoric steps can be pre-computed. A two-dimensional quadric has the general equation

$$Ax^2 + By^2 + Cxy + Dx + Ey + F = 0,$$

inserting the grid lines which are parallel to the coordinate axes, and simplifying the equation leads to solving the following quadratic equation

$$x_{1/2} = -\frac{p}{2} \pm \sqrt{\frac{p^2}{4} - q}$$

with

$$p = \frac{Cy_g + D}{A} \qquad \text{and} \qquad q = \frac{By_g^2 + Ey_g + F}{A}$$

for the case of a grid line that is parallel to the x-axis.
The classification of the B-splines follows directly from this: B-splines with only outer and boundary cells in their support are marked as outer B-splines and B-splines with at least one inner cell in their support are marked as inner B-splines. B-splines with no boundary and inner cell in their support are not needed and not classified.
For setting up the Ritz-Galerkin-system, first only the spaces of the weighted B-splines are used. This means that for each inner cell or boundary cell Q_i the integrals

$$\alpha_{i,k,\ell} = \int\limits_{Q_i \cap D} \operatorname{grad}(wb_{k,h}^n) \cdot \operatorname{grad}(wb_{\ell,h}^n) dx \qquad \text{and} \qquad \beta_{i,k} = \int\limits_{Q_i \cap D} f w b_{k,h}^n dx$$

have to be computed. This results in the Ritz-Galerkin-system $\tilde{G}\tilde{C} = \tilde{F}$ for the basis of the WB-space

$$\tilde{g}_{k,\ell} = \sum_{Q_i \subset \operatorname{supp}\, b_{k,h}^n \cap \operatorname{supp}\, b_{\ell,h}^n} \alpha_{i,k,\ell} \qquad \text{and} \qquad \tilde{f}_k = \sum_{Q_i \subset \operatorname{supp}\, b_{k,h}^n} \beta_{i,k}.$$

With an easy transformation it is possible to transform this system into a system for the WEB-basis. Implementation becomes easier when using this transformation.
There are several possibilities for solving the Ritz-Galerkin-system, for example a multigrid solver [19] or a (preconditioned) conjugate gradients method.

Literaturverzeichnis

[1] Marcel Berger. *Geometry II.* Universitext, Springer Verlag, Berlin, 1987.

[2] Wolfgang Boehm and Hartmut Prautzsch. *Geometric Concepts for Geometric Design.* A K Peters, Wellesley, Massachusetts, 1994.

[3] Dietrich Braess. *Finite Elemente.* Springer, 2003.

[4] Susanne C. Brenner and L. Ridgway Scott. *The Mathematical Theory of Finite Element Methods.* Springer, 2002.

[5] Victor I. Burenkov. *Sobolev Spaces on Domains.* B.G. Teubner, 1998.

[6] M.G. Cox. The numerical evaluation of b-splines. *J. Inst. Maths. Applies.*, 10:134–149, 1972.

[7] Carl de Boor. On calculating with b-splines. *Journal of Approximation Theory*, 6:50–62, 1972.

[8] Boris N. Delaunay. Sur la sphère vide. *Bulletin of Academy of Sciences of the USSR*, pages 793–800, 1934.

[9] Gerald Farin. *Kurven unf Flächen im Computer Aided Geometric Design.* vieweg, 1994.

[10] Gerald Farin and Dianne Hansford. *Lineare Algebra - ein geometrischer Zugang.* Springer, 2003.

[11] Michael Floater and Chris Dyken. Transfinite mean value interpolation. *Comp. Aided Geom, Design*, 2007, to appear 2008.

[12] Winfried Geis. Fem-verfahren mit web-spline-basis. Wissenschaftliche arbeit, Universität Stuttgart, 2001.

[13] Anne Greenbaum. *Iterative methods for solving linear systems*, volume 17 of *Frontiers in Applied Mathematics.* 1997.

[14] Andreas Griewank. *Evaluating Derivatives.* Frontiers in applied mathematics. SIAM, 2000.

[15] Klaus Höllig. *Grundlagen der Numerik.* MathText, Zavelstein, 1998.

[16] Klaus Höllig. *Finite Element Method with B-Splines.* SIAM, 2003.

[17] Klaus Höllig and Jürgen Koch. Geometric Hermite Interpolation with maximal order and smoothness. *Computer Aided Geometric Design*, 13:681–695, 1996.

[18] Klaus Höllig, Ulrich Reif, and Joachim Wipper. Weighted extended b-spline approximation of dirichlet problems. *SIAM J. Numer. Ana.*

[19] Klaus Höllig, Ulrich Reif, and Joachim Wipper. Multigrid methods with web-splines. *Numerische Mathematik*, 91:237–256, 2002.

[20] Josef Hoschek and Dieter Lasser. *Grundlagen der geometrischen Datenverarbeitung*. B.G. Teubner, Stuttgart, 1989.

[21] Jürgen Koch. *Parallele Algorithmen zur Lösung schwachbesetzter polynomialer Gleichungssysteme*. PhD thesis, Math.Inst.A, Univ. Stuttgart, 1996.

[22] Louis B. Rall. *Automatic Differentiation: Techniques and Applications*, volume 120 of *Lecture Notes in Computer Science*. Springer-Verlag, 1981.

[23] Vladimir L. Rvachev and Tatyana I. Sheiko. R-functions in boundary value problems in mechanics. *Appl.Mech.Rev.*, 48(4):151–188, 1995.

[24] Vladimir L. Rvachev, Tatyana I. Sheiko, Vadim Shapiro, and Igor G. Tsukanov. On completeness of rfm solution structures. *Comp.Mech.*, 25:305–316, 2000.

[25] Vladimir L. Rvachev, Tatyana I. Sheiko, Vadim Shapiro, and Igor G. Tsukanov. Transfinite interpolation over implicitly defined sets. Technical Report SAL 2000-1, Spatial Automation Laboratory, 2000.

[26] J. Scott Savage and Andrew F. Peterson. Quadrature rules for numerical integration over triangles and tetrahedra. *IEEE Antennas and Propagation Magazine*, 38:100–102, 1996.

[27] Jonathan Richard Schewchuk. *Delaunay Refinement Mesh Generation*. PhD thesis, Carnegie Mellon University, 1997.

[28] Isaac Jacob Schönberg. Contributions to the problem of approximation of equidistant data by analytic functions. *Quart.Appl.Math.*, 4:45–99 und 112–141, 1946.

[29] Vadim Shapiro. Theory of r-functions and applications: A primer. Techn. Report CPA88-3, Cornell Programmable Automation, Sibley School of Mechanical Engineering, Ithaca, NY., 1988.

[30] Joachim Wipper. *Finite-Elemente-Approximation mit WEB-Splines*. PhD thesis, IMNG, Univ. Stuttgart, 2006.

[31] Eberhard Zeidler. *Nonlinear Functional Analysis and Its Applications*. Springer, 1990.

[32] Andreas Zell. *Simulation neuronaler Netze*. Oldenbourg Wissenschaftsverlag, Muenchen, 4., unveraenderter nachdruck edition, 2003.