Rheinische Friedrich-Wilhelms-Universität Bonn

Automatisierte parallele Generierung blockstrukturierter Gitter für CAD-erstellte Geometrien

Diplomarbeit
von
Christian Müller aus Ulm

Bonn, im November 2003

Inhaltsverzeichnis

Kapitel 1

Einleitung

In vielen Bereichen der industriellen Produktentwicklung wird heutzutage die numerische Simulation von Strömungen und anderen Naturphänomenen erfolgreich eingesetzt. Dies ist begründet durch die Zeit- und Kostenersparnis, die durch die Nutzung von Computern erreicht wird. Zunehmend ist hohe Rechnerleistung kostengünstig verfügbar, so dass der Einsatz von mittlerweile hoch entwickelten Simulationsprogrammen günstiger ist, als aufwändig zu konstruierende Prototypen einem umfangreichen Praxistest zu unterwerfen. Immer kürzere Produktzyklen werden daher die Bedeutung von Simulationen im Rahmen der Produktentwicklung weiter verstärken und den Einsatz "klassischer Prototypen" in der Produktentwicklung weiter zurückdrängen. Daher kommt der Ausarbeitung zunehmend detailgetreuer und realitätsnaher Simulationen ein immer größeres ökonomisches und wissenschaftliches Interesse zu.

Prinzipiell können derartige numerische Simulationen in drei Teilschritte unterteilt werden: Zuerst gilt es, ein mathematisches Modell zu entwickeln, das das zu simulierende Phänomen durch mathematische Gleichungen beschreibt. Der zweite Schritt ist es, diese kontinuierlichen Gleichungen zu diskretisieren. Das heißt, die Gleichungen nur noch an endlich vielen Punkten zu betrachten, um auf diesen eine Lösung zu finden, die die kontinuierliche Lösung hinreichend genau approximiert. Dabei wird die kontinuierliche Lösung umso besser approximiert, je höher die Auflösung des Simulationsgebiets durch diese Punkte ist. Im dritten Schritt wird das diskretisierte Problem mit Rechnereinsatz möglichst schnell und genau gelöst. Schließlich wird die berechnete Lösung des diskreten Problems zu Analysezwecken visualisiert.

Der Fokus dieser Arbeit richtet sich auf den zweiten Teilschritt der numerischen Simulation. Es wird die Generierung von Gittern behandelt. Dabei ist ein Gitter, stark vereinfacht, die Menge der diskreten Punkte, auf denen die Lösung berechnet wird. Die hier generierten Gitter werden zur Strömungssimulation eingesetzt. Dabei löst der parallele Strömungslöser NSCL auf diesen Gittern die Navier-Stokes-Gleichungen, welche die Strömung modellieren. Eine wesentliche Eigenschaft der erzeugten Gitter ist, dass sie anhand von CAD (**C**omputer **A**ided **D**esign)-Daten generiert werden. Da der Einsatz von CAD-Programmen zur Produktgestaltung heutzutage Standard ist, ist es naheliegend, die Simulation einer Umströmung des im CAD-Programm konstruierten Objekts direkt an das CAD-Programm anzubinden. Die Verbindung der Simulation mit dem Objektentwurf ermöglicht eine unkomplizierte Handhabung der Strömungssimulation für eine große Klasse von Geometrietypen. Das in dieser Arbeit vorgestellte Programm leistet genau diesen Schritt. Dafür

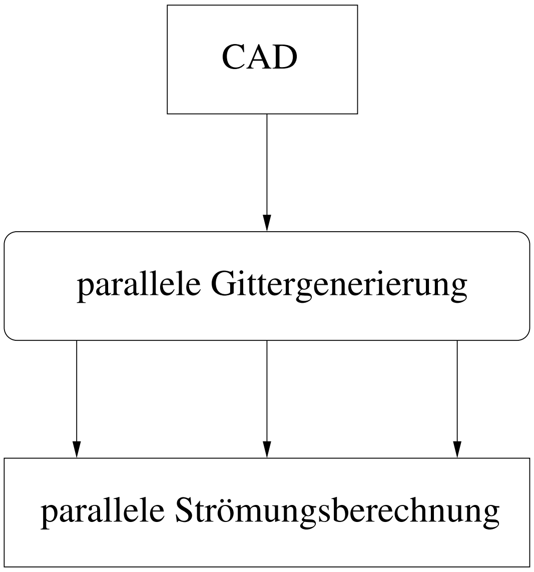

Abbildung 1.1: Datenstrom der parallelen Strömungssimulation.

werden die Geometriedaten des zu umströmenden Objekts in dem weitverbeiteten CAD-Datenstandard IGES eingelesen, um auf dem Simulationsgebiet, also auf der Umgebung des Objektes oder im Inneren des Objektes (bei einer Durchströmung des Objektes) Gitter zu generieren. Auf diesen Gittern wird dann Strömung berechnet. Bei den für den Strömungslöser zu generierenden Gittern handelt es sich um *blockstrukturierte* Gitter. Das heißt, das Gitter ist aus mehreren Blöcken strukturierter Gitter zusammengesetzt. Zur Generierung blockstrukturierter Gitter muss das Simulationsgebiet erst in einzelne Blöcke aufgeteilt werden, auf denen dann jeweils ein strukturiertes Gitter generiert wird.
Bei hoher Auflösung des Simulationsgebietes, um zum Beispiel Turbulenz berechnen zu können, wird die Anzahl der Gitterpunkte und damit die zu verwaltende Datenmenge extrem groß. Um große Datenmengen effizient handhaben zu können, ist es sinnvoll, die Daten *parallel* zu bearbeiten. Dafür werden die Daten auf mehrere Rechner verteilt, die dann gleichzeitig an kleineren Teilproblemen arbeiten. Die

Verteilung der Daten wird beibehalten, um im Anschluss an die parallele Gittergenerierung ohne erneuten Datenaustausch parallel Strömung zu berechnen. Diese Vorgehensweise ist in Abbildung 1.1 angedeutet.
Zur Bearbeitung dieser Aufgabe wurde ein C++ Programm entwickelt. Die vorliegende Arbeit stellt die hierfür benutzten Methoden sowie die erzielten Ergebnisse vor.

Die Arbeit ist wie folgt gegliedert:
In Kapitel 2 werden die Grundlagen für die Gittergenerierung bereitgestellt. Nach einer Einführung in die Gittergenerierung werden die benutzten Methoden zur Generierung strukturierter Gitter vorgestellt. Diese sind Transfinite Interpolation und Methoden mit elliptischen Differentialgleichungen. Es wird das klassische Laplace-System zur Gittergenerierung vorgestellt, anhand dessen die grundlegenden Eigenschaften von Gittern, die mittels elliptischer Differentialgleichungen generiert wurden, aufgezeigt werden. Danach wird eine von Spekreijse [33] entwickelte, verbesserte Methode vorgestellt, die ein Poisson-System verwendet. Schließlich werden die zur internen Geometriedarstellung benutzten NURBS-Flächen und der benutzte CAD-Datenstandard IGES vorgestellt.
Kapitel 3 behandelt die Umsetzung der Generierung blockstrukturierter Gitter. Dabei wird auf die Rekonstruktion der Geometrie aus den CAD-Daten, die Aufteilung des Simulationsgebiets in Blöcke und die Generierung der einzelnen Blöcke eingegangen. Es wird eine Methode entwickelt, die die Blockgrenzen glättet und dadurch zu einer Verbesserung des blockstrukturierten Gitters führt. Anschließend wird ein Überblick über den Aufbau des für diese Arbeit entwickelten Programms gegeben. Danach wird auf die numerische Lösung der bei der Gittergenerierung mittels elliptischer Differentialgleichungen entstehenden nichtlinearen Gleichungssysteme eingegangen. Hierfür wird die Picard-Iteration eingesetzt. Für das dabei entstehende lineare Teilproblem wird die Effizienz verschiedener Löser verglichen. Außerdem wird der Einfluss einer Dämpfung auf die Konvergenzgeschwindigkeit der Picard-Iteration untersucht. Abschließend wird auf die Parallelisierung der Gittergenerierung eingegangen.
Kapitel 4 stellt die Ergebnisse numerischer Simulationen dar, die auf Gittern berechnet wurden, welche mit dem hier entwickelten Gittergenerierer erzeugt wurden.
In Kapitel 5 werden die Ergebnisse dieser Arbeit zusammengefasst und Perspektiven für die Weiterentwicklung aufgezeigt.

Kapitel 2

Theoretische Grundlagen

2.1 Einführung in die Gittergenerierung

Ein Gitter im allgemeinen Sinne ist eine Diskretisierung eines Gebietes $X \subset \mathbb{R}^d, d = 1, 2, 3$. Es dient dazu, auf diesem mathematische Berechnungen durchführen zu können, die wegen seines unendlichen Charakters nicht direkt auf dem Gebiet möglich sind. Dies beinhaltet zum Beispiel das Lösen von Differential- oder Integralgleichungen. Die Diskretisierungsmethoden korrespondieren mit den entsprechenden Lösungsmethoden für die Gleichungen. So wird bei Finite-Elemente-Methoden das Gebiet X in Teilgebiete, sogenannte *Zellen* aufgeteilt. Die Lösung des Problems setzt sich dann aus den Lösungen auf den Teilgebieten zusammen. Finite-Differenzen-Verfahren beruhen auf der Bildung von Differenzen von Funktionswerten zwischen benachbarten Punkten. Die entsprechende Diskretisierung des Gebiets ist eine *Menge von Punkten mit Nachbarschaftsrelationen*. Man kann die zu lösenden Gleichungen also entweder auf einer diskreten Menge von Zellen, oder auf einer Menge von diskreten Punkten betrachten.
Dem entspricht die Unterteilung der Gitter in *strukturierte* und *unstrukturierte Gitter*, die jeweils mit der Diskretisierung des Gebiets als Menge von Punkten mit Nachbarschaftsrelationen beziehungsweise als Menge von Zellen identifiziert sind. Jedes strukturierte Gitter kann auch als unstrukturiertes Gitter betrachtet werden, da man über die Nachbarschaftsrelationen der Punkte das Gebiet auch in Zellen zerlegen kann. Der umgekehrte Fall gilt im Allgemeinen nicht, da an strukturierte Gitter strengere Anforderungen hinsichtlich der Topologie gestellt werden.
Andere Methoden, die sogenannten gitterfreien Methoden, verwenden allgemeinere Diskretisierungskonzepte als Gitter. Auf diese wird hier aber nicht eingegangen. Mehr über diese Methoden findet man zum Beispiel in [11] und [1].

Finite-Differenzen-Verfahren sind, historisch gesehen, eng mit kartesischen Gittern auf rechteckigen Gebieten verbunden, da diese hierauf leicht zu generieren sind, und die Nachbarschaftsrelationen klar vorgegeben sind. Dagegen wurden Finite-Elemente-Methoden für Gebiete mit allgemeineren Geometrien benutzt. Strukturierte Gitter sind aber nicht auf rechteckige Gebiete beschränkt. Einfache Abbildungen, wie die Polarkoordinatenabbildung, erlauben es, die Nachbarschaftsrelationen auf kompliziertere Geometrien zu übertragen: Verallgemeinert man diese Idee, so

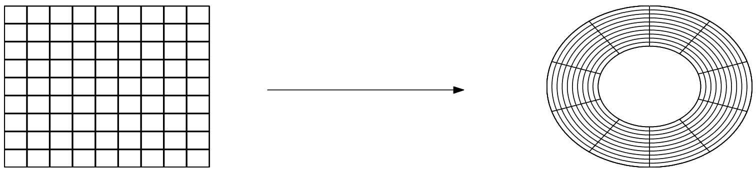

Abbildung 2.1: Ein strukturiertes Gitter gegeben durch die Polarkoordinatenabbildung.

kann man die Generierung strukturierter Gitter auf die Suche einer Abbildung eines rechteckigen Gebietes mit kartesischem Koordinatensystem auf das zu diskretisierende Gebiet zurückführen.
Dazu bezeichne $\Xi \subset \mathbb{R}^d$ ein d-dimensionales Rechteck und $X \subset \mathbb{R}^d$ das zu diskretisierende Gebiet. Dann ist eine Abbildung

$$x : \overline{\Xi} \longrightarrow \overline{X}$$

gesucht. Die Abbildung x muss — ähnlich einer Karte wie sie aus der Differentialgeometrie bekannt ist, siehe [4, Kap. 0.2] — bestimmte Bedingungen erfüllen. Insbesondere muss der Rand von Ξ bijektiv auf den Rand von X abgebildet werden. Im nächsten Kapitel 2.2 wird genauer auf die von x zu erfüllenden Bedingungen und

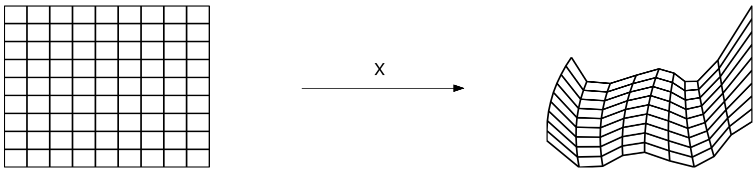

Abbildung 2.2: Gittergenerierung mittels Transformationsabbildung.

die Methoden zur Bestimmung dieser Abbildung eingegangen. Ein Vorteil strukturierter Gitter ist, dass durch ihre feste Topologie eine effiziente Numerik möglich ist. Andererseits ist der Aufwand bei der Gittergenerierung höher als bei unstrukturierten Gittern.

Bei unstrukturierten Gittern werden keine weiteren Bedingungen an die Organisation der Gitterpunkte gestellt — sie sind *un*strukturiert. So sind die entstehenden Formen der Gitterzellen vielfältig. Häufig auftretende Formen der Gitterzellen sind

Tetraeder (in $3D$) oder Dreiecke (in $2D$). Grundsätzlich sind aber keine weiteren Forderungen an die Zellen gestellt, außer dass sie d-dimensionale Volumina sind. Betrachtet man strukturierte Gitter als unstrukturierte Gitter, so sind die sich ergebenden Zellen immer Hexaeder beziehungsweise Vierecke.
Ein Vorteil dieser Methode ist die Flexibilität, sowohl was komplizierte Geometrien als auch was Verfeinerungen und Vergröberungen der Auflösung angeht. Denn Zellen lassen sich leicht durch Hinzufügen von Punkten zerteilen oder durch Entfernen von Punkten mit anderen Zellen verschmelzen, ohne dass andere Teile des Gitters verändert werden müssen. Dies macht die Generierung solcher Gitter relativ einfach. Andererseits steigt durch die fehlende Struktur der Verwaltungsaufwand. Es muss gespeichert werden, welche Zellen welche Punkte beinhalten, und welche Zellen aneinandergrenzen.
Neben diesen beiden Gitterarten gibt es noch Variationen und Kombinationen der

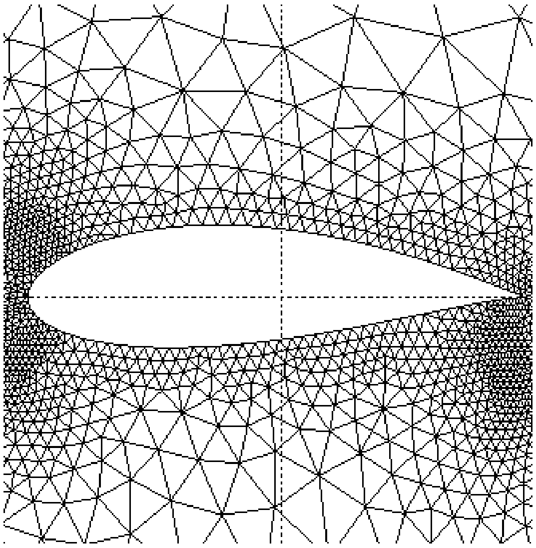

Abbildung 2.3: Unstrukturiertes Gitter um einen Tragflächenquerschnitt.

beiden Gittersorten: blockstrukurierte, überlappende und hybride Gitter.

Blockstrukturierte Gitter bestehen aus mehreren strukturierten Gittern, die wiederum zu einem (möglicherweise unstrukturierten) Gitter zusammengesetzt werden. Diese Methode erlaubt durch eine geeignete Wahl der Blockaufteilung, den Verlauf der Gitterlinien genauer zu steuern. Durch diese Methode können auch für kompliziertere Geometrien noch strukturierte Gitter erzeugt werden. Wesentlicher Bestandteil der Generierung blockstrukturierter Gitter ist die Wahl der geeigneten Blocktopologie. Diese beschreibt, wie die einzelnen Gitterblöcke zu einem Gitter zusammengesetzt werden. Hierauf wird in Kapitel 3.1 eingegangen. Auf der Ebene eines einzelnen Blocks werden strukturierte Gitter, wie in Kapitel 2.2 besprochen, generiert.

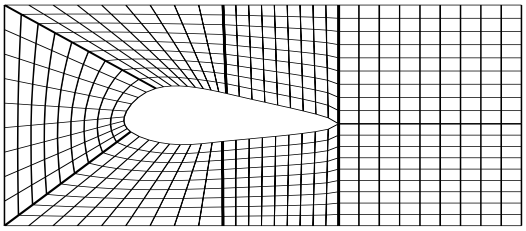

Abbildung 2.4: Blockstrukturiertes Gitter um einen Tragflächenquerschnitt.

Überlappende Gitter bestehen ebenfalls aus mehreren strukturierten Gittern, die sich jedoch (im Gegensatz zu blockstrukturierten Gittern) überlappen können. Sie sind leichter zu generieren als blockstrukturierte Gitter, da man die Blockgrenzen einfacher wählen kann. Allerdings stellen solche Gitter zusätzliche Anforderungen an die numerischen Methoden, da wegen des Überlappens geeignete Interpolationen zwischen den einzelnen Blöcken erforderlich sind.

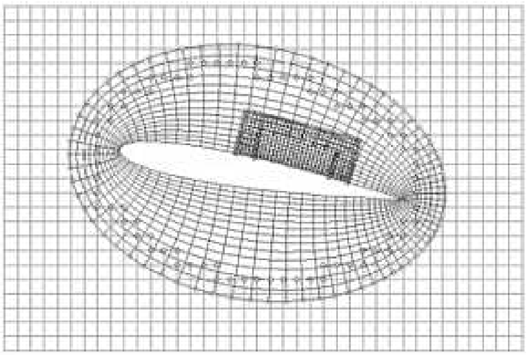

Abbildung 2.5: Überlappendes Gitter um einen Tragflächenquerschnitt.

Hybride Gitter sind aus strukturierten und unstrukturierten Gittern zusammengesetzt. Solche Gitter können sinnvoll sein für Probleme, die heterogene Materialien, zum Beispiel feste und flüssige, behandeln, auf denen unterschiedliche Gleichungen gelöst werden sollen.

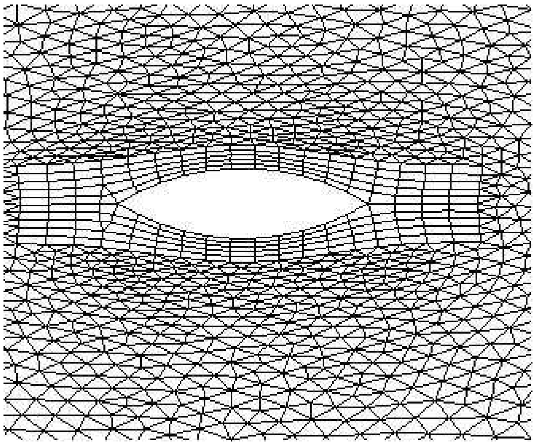

Abbildung 2.6: Hybrides Gitter um einen Tragflächenquerschnitt.

In dieser Arbeit werden, wie eingangs erwähnt, blockstrukturierte Gitter generiert. Die Generierung solcher Gitter zerfällt in zwei Teilschritte: Zuerst muss das ganze Gebiet in mehrere Blöcke zerlegt werden. Danach muss für jeden einzelnen Block ein Gitter erstellt werden.
Die Gittergenerierung auf den jeweiligen Teilblöcken ist identisch mit der Generierung strukturierter Gitter, die im folgenden Abschnitt behandelt wird. Die Blockzerlegung wird in Kapitel 3.1.3 besprochen.

2.2 Generierung strukturierter Gitter

Die verschiedenen Methoden, eine Abbildung $x : \overline{\Xi} \longrightarrow \overline{X}$ zu finden, mit der ein Gitter auf X generiert werden kann, werden nun vorgestellt. Dafür wird eine Notation eingeführt, die in der gesamten Arbeit beibehalten wird. Seien hierzu X ein einfach zusammenhängendes Gebiet in $\mathbb{R}^3$ und $\Xi = (0,1)^3$ der Einheitswürfel in $\mathbb{R}^3$. Die Abbildung x setzt sich dann aus den drei Koordinatenfunktionen $x_i : \overline{\Xi} \longrightarrow \mathbb{R},\ i = 1,2,3$ zusammen, $x = (x_1, x_2, x_3)$. Auf dem Würfel $\overline{\Xi}$ steht ein Gitter trivial zur Verfügung. Seine Gitterpunkte $\xi_p(i,j,k)$ sind gegeben durch $\xi_p(i,j,k) = (\frac{i}{I-1}, \frac{j}{J-1}, \frac{k}{K-1})$. Dabei sind I, J, K die Anzahlen der Gitterpunkte in Richtung der jeweiligen Raumdimension und $0 \leq i < I,\ 0 \leq j < J,\ 0 \leq k < K$. Der Würfel $\overline{\Xi}$ soll nun so auf $\overline{X}$ abgebildet werden, dass man hier genauso wie in $\overline{X}$ auf die Gitterpunkte zugreifen kann. Die Gitterpunkte $x_p(i,j,k)$ in X sind dann gegeben durch $x_p(i,j,k) = x(\xi_p(i,j,k))$. Vergleiche dazu auch Abbildung 2.1 oder 2.2. Dafür muss die Abbildung x die folgenden Bedingungen erfüllen:

1. $x : \overline{\Xi} \longrightarrow \overline{X}$ muss bijektiv sein

2. Der Rand von Ξ muss bijektiv auf den Rand von X abgebildet werden.

Ist die Gitterabbildung nicht bijektiv, kann es zu gefalteten Gittern, das heißt Gittern mit sich schneidenden Gitterlinien, kommen. Diese Gitter sind unbrauchbar.

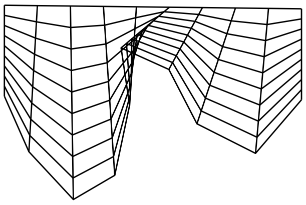

Abbildung 2.7: Gefaltetes Gitter

Bei vielen Diskretisierungsverfahren sind Gitterlinien mit möglichst geringer Krümmung von Vorteil. Der hier benutzte Strömungslöser NSCL interpoliert zum Beispiel die Werte für die Geschwindigkeit linear zwischen zwei Gitterzellen, was bei stärker gekrümmten Gitterlinien zu größeren Fehlern führt [23, Kap. A.1.3].
Unter den Methoden zur Generierung strukturierter Gitter unterscheidet man grundsätzlich zwei Klassen von Methoden: *Algebraische* und *Differentialgleichungs-* beziehungsweise *Variations-Methoden*.
Algebraische Methoden generieren Gitter mittels *Transfiniter Interpolation*. Dabei erhält man die Koordinatenfunktionen durch Interpolation zwischen den Rändern von X. Diese Methoden sind leicht und effizient zu berechnen. Bei Gebieten mit komplizierten Rändern kann die Interpolation jedoch fehlschlagen. Mögliche Folgen sind Gitterpunkte, die außerhalb des Gebietes liegen, oder eine nicht bijektive Abbildung. Diese Fehler lassen sich durch Benutzung von Differentialgleichungsmethoden vermeiden. Da die Transfinite Interpolation effizient zu berechnen ist, werden algebraisch erzeugte Gitter (auch in der vorliegenden Arbeit) als Startwerte für iterative Berechnungen in Differentialgleichungsmethoden benutzt.
Bei den Differentialgleichungsmethoden wählt man die Koordinatenfunktionen x_i, $i = 1, 2, 3$ von x als Lösungen von Differentialgleichungen. Eine entsprechende Wahl der Differentialgeichungen erlaubt es, die gewünschten Eigenschaften der Lösungen, also der Abbildung x, zu erreichen. Bei elliptischen Differentialgleichungen kann man zum Beispiel am gesamten Rand sogenannte Dirichlet-Randwerte vorgeben. Durch diese Bedingungen kann man sicherstellen, dass $\partial\Xi$ auf ∂X abgebildet wird. Aus diesem Grund werden Methoden mit Systemen elliptischer Differentialgleichungen häufig für die Generierung blockstrukturierter Gitter eingesetzt. In vielen Fällen ist hier Bijektivität der Abbildung x, wie man in Kapitel 2.4 sehen wird, durch das Maximumprinzip garantiert. Regularitätssätze, [8, Kap.6.4], können eine zusätzliche Glattheit der Koordinatenfunktionen garantieren.
Stehen andere Eigenschaften des Gitters im Vordergrund, ist es sinnvoll, andere Typen von Differentialgleichungen zu verwenden. Wenn man zum Beispiel nicht am gesamten Rand Werte vorgeben möchte, auf der anderen Seite aber ein möglichst orthogonales Gitter erhalten möchte, ergeben sich Lösungen hyperbolischer Diffe-

rentialgleichungen als geeignete Gitterabbildung (Beispiel: [27]). Ausgehend von elliptischen Methoden sind auch Methoden mit parabolischen Gleichungen entwickelt worden. Sie liefern möglichst orthogonale Gitter und erlauben Vorgaben an das Zellvolumen, zum Beispiel [25] oder [26].
Ein Vorteil dieser Methoden ist, dass die numerischen Verfahren zur Lösung dieser Gleichungen deutlich schneller sind als die für elliptische Differentialgleichungen. Hierbei wird eine Ortskoordinate als Zeitvariable betrachtet und dann ein Anfangswertproblem gelöst. Dadurch erlauben es diese Methoden nicht, am ganzen Rand Werte vorzugeben. Dies schränkt die Anwendungsmöglichkeiten solcher Gitter stark ein, da man in vielen Fällen die Gitterpunkte am Rand vorgeben möchte. Insbesondere bei blockstrukturierten Gittern ist es unverzichtbar, dass sich die Gitterlinien angrenzender Blöcke an den Blockgrenzen treffen. So werden durch hyperbolische oder parabolische Differentialgleichungen generierte Gitter vor allem für überlappende Gitter oder Gitter, die nur Teile des Randes genau auflösen, verwendet. Ein Beispiel für den letzteren Fall ist die Umströmung einer Rakete, wo es wichtig ist, die Oberfläche derselben genau zu treffen, man aber keine Vorgaben haben muss, wie die Gitterlinien am äußeren Rand, der einfach das Ende des Simulationsgebietes kennzeichnet, verlaufen müssen.
Die verschiedenen Methoden mit Differentialgleichungen sind optimal, bezüglich unterschiedlicher Eigenschaften. Wie man in Kapitel 2.4 sehen wird, haben zum Beispiel durch Laplace-Gleichungen generierte Gitter besonders gleichmäßig verteilte Gitterpunkte. Ein anderes Beispiel sind die oben erwähnten hyperbolischen Gleichungen, die möglichst orthogonale Gitter liefern können.
Dies legt die Formulierung der Gittergenerierung mittels Differentialgleichungen als Optimierungsprozess nahe. Bei den Variationsmethoden werden die geforderten Bedingungen als Kostenfunktional formuliert. Dies umfasst Bedingungen wie Glattheit der Koordinatenfunktionen, Abweichung von Orthogonalität der Gitterlinien und Vorgaben für das Gitterzellenvolumen. Man kann diese Bedingungen im Kostenfunktional, je nach Anwendung, unterschiedlich gewichten und so eine flexible Methode zur Generierung strukturierter Gitter mit unterschiedlichen Eigenschaften erhalten. Über die entsprechenden Euler-Lagrange-Gleichungen [8, Kap. 11.5] erhält man die zu lösenden Differentialgleichungen.
In dieser Arbeit wird die Gittergenerierung mittels Systemen bestimmter elliptischer Differentialgleichungen (in Kapitel 2.4) sowie die im folgenden Kapitel beschriebene Transfinite Interpolation näher besprochen.

2.3 Transfinite Interpolation

Die Transfinite Interpolation (TFI) wurde zuerst 1973 von William Gordon beschrieben [9]. Das Ziel der Transfiniten Interpolation ist, die Abbildung $x : \overline{\Xi} \longrightarrow \overline{X}$ durch

Interpolation zu erzeugen. Dabei werden die Funktions- und möglicherweise auch die Ableitungswerte von x wenigstens am Rand von X vorgegeben. Die Interpolation heißt transfinit, da der Rand von X überall exakt aufgelöst wird, im Gegensatz zu anderen Interpolationstechniken, die den Rand nur auf Teilstücken exakt auflösen [34, p. 311]. TFI basiert auf einer *unidirektionalen Interpolation*, deren Ergebnisse (hier P_1, P_2, P_3) durch die *Boolsche Summe* zur TFI, $P[x] : \overline{\Xi} \longrightarrow \mathbb{R}^3$, kombiniert werden:

$$\begin{aligned} P[x] &= P_1[x] \ \oplus \ P_2[x] \ \oplus \ P_3[x] \\ &= P_1[x] + P_2[x] + P_3[x] - \\ &\quad P_1[x] \otimes P_2[x] - P_1[x] \otimes P_3[x] - P_2[x] \otimes P_3[x] + \\ &\quad P_1 \otimes P_2 \otimes P_3. \end{aligned}$$

Im Folgenden wird erst die unidirektionale Interpolation definiert, mit der dann das Tensorprodukt $\otimes$ und die Boolsche Summe $\oplus$ für unidirektionale Interpolationen definiert werden können.

Unidirektionale Interpolation

Die *unidirektionale Interpolation* ist eine Interpolation in einer Koordinatenrichtung, hier in Richtung $\xi_i, i = 1, 2, 3$. Es wird zwischen Stützstellen, die auf Ebenen orthogonal zur jeweiligen Koordinatenrichtung gegeben sind, interpoliert. Eine Ebene orthogonal zur Koordinatenrichtung ξ_i ist dabei die Menge der Punkte $x(\xi_1, \xi_2, \xi_3)$ mit festem ξ_i. Sei die zu interpolierende Funktion x sowie Ableitungen von x bis zum Grad Q auf den Ebenen orthogonal zu ξ_i in $L_i \geq 2$ Positionen $\xi_i = \xi_l^i$, $1 \leq l \leq L_i$ vorgegeben, wobei Fall $L_i = 2$ bedeutet, dass nur Randwerte vorgegeben sind, für $L_i > 2$ gibt es zusätzliche Stützstellen im Inneren. Dann ist die *unidirektionale Interpolation in Richtung* ξ_i gegeben durch

$$\begin{aligned} &P_i[x] : \overline{\Xi} \longrightarrow \mathbb{R}^3 \\ &P_i[x](\xi_1, \xi_2, \xi_3) = \sum_{l=1}^{L_i} \sum_{m=0}^{Q} \alpha_i^{lm}(\xi_i) \beta_i^{lm}(\xi_1, \xi_2, \xi_3). \end{aligned}$$

β_i^{lm} ist dabei der vorgegebene Wert von $\frac{\partial^m x}{\partial \xi_i^m}$ auf der l-ten Ebene orthogonal zur Koordinatenrichtung ξ_i,

$$\beta_i^{lm}(\xi_1, \xi_2, \xi_3) = \frac{\partial^m x}{\partial \xi_i^m}(\xi_1, \xi_2, \xi_3) \quad \text{mit} \quad \xi_i \equiv \xi_i^l, \quad i = 1, 2, 3. \tag{2.1}$$

Die Überblendfunktionen α_i^{lm} sind durch die verwendete Interpolation gegeben. Hierfür werden Standardinterpolationstechniken wie Hermite- oder Lagrange-Interpolation eingesetzt. Allgemein formuliert sind die Überblendfunktionen von der Form

$$\alpha_i^{lm} : [0,1] \longrightarrow \mathbb{R}$$
$$\frac{\partial^{\tilde{m}} \alpha_i^{\tilde{l}m}}{\partial \xi_i^{\tilde{m}}}\Big|_{\xi_i = \xi_i^l} = \delta_{l\tilde{l}} \delta_{m\tilde{m}}$$
$$l, \tilde{l} = 1, \dots, L_i \quad m, \tilde{m} = 0, \dots, Q.$$

δ_{ij} bezeichnet hierbei das Kronecker-Delta. Die einfachste Form mit $L_i = 2$ und $Q = 0$ entspricht genau der Lagrange-Interpolation zwischen den Rändern ohne zusätzliche Stützpunkte. Bei komplizierten Geometrien kann es sinnvoll sein, Stützstellen im Inneren des Gebietes vorzugeben, um zu vermeiden, dass die Koordinatenlinien das Gebiet verlassen. Dies ist insbesondere bei stark nicht-konvexen Gebieten der Fall.

Tensorprodukt

Das *Tensorprodukt* zwischen unidirektionalen Interpolationen $P_i[x]$ und $P_j[x]$ ist für $i \neq j$ definiert durch

$$\begin{aligned} P_i[x] \otimes P_j[x] :=& P_i\big[P_j\,[x]\,\big] \\ =& \sum_{l=1}^{L_i} \sum_{m=0}^{Q} \alpha_i^{lm}(\xi_i) \sum_{k=1}^{L_j} \sum_{n=0}^{R} \alpha_j^{kn}(\xi_j) \beta_{ij}^{klmn}. \end{aligned}$$

Hierbei ist β_{ij}^{klmn} der Wert von $\frac{\partial^{m+n} x(\xi_1,\xi_2,\xi_3)}{\partial \xi_i^m \partial \xi_j^n}$ auf der Schnittmenge der l-ten Ebene orthogonal zur Koordinatenrichtung ξ_i und der k-ten Ebene orthogonal zur Koordinatenrichtung ξ_j,

$$\beta_{ij}^{klmn}(\xi_1, \xi_2, \xi_3) = \frac{\partial^{m+n} x(\xi_1, \xi_2, \xi_3)}{\partial \xi_i^m \partial \xi_j^n} \quad \text{mit} \quad \xi_i \equiv \xi_i^l \text{ und } \xi_j \equiv \xi_j^k, \quad i, j = 1, 2, 3. \tag{2.2}$$

Das Tensorprodukt ist symmetrisch, denn

$$\begin{aligned} P_i[x] \otimes P_j[x] &= \sum_{l=1}^{L_i} \sum_{m=0}^{Q} \sum_{k=1}^{L_j} \sum_{n=0}^{R} \alpha_i^{lm}(\xi_i) \alpha_j^{kn}(\xi_j) \beta_{ij}^{klmn} \\ &= \sum_{k=1}^{L_j} \sum_{n=0}^{R} \sum_{l=1}^{L_i} \sum_{m=0}^{Q} \alpha_j^{kn}(\xi_j) \alpha_i^{lm}(\xi_i) \beta_{ji}^{lknm} = P_j[x] \otimes P_i[x]. \end{aligned} \tag{2.3}$$

Ebenso definiert man das Tensorprodukt zwischen drei unidirektionalen Interpolationen:

$$P_i[x] \otimes P_j[x] \otimes P_k[x] := P_i\Big[P_j\big[P_k\,[x]\,\big]\Big]. \tag{2.4}$$

Boolsche Summe
Betrachte zur Motivation den 2D Fall.

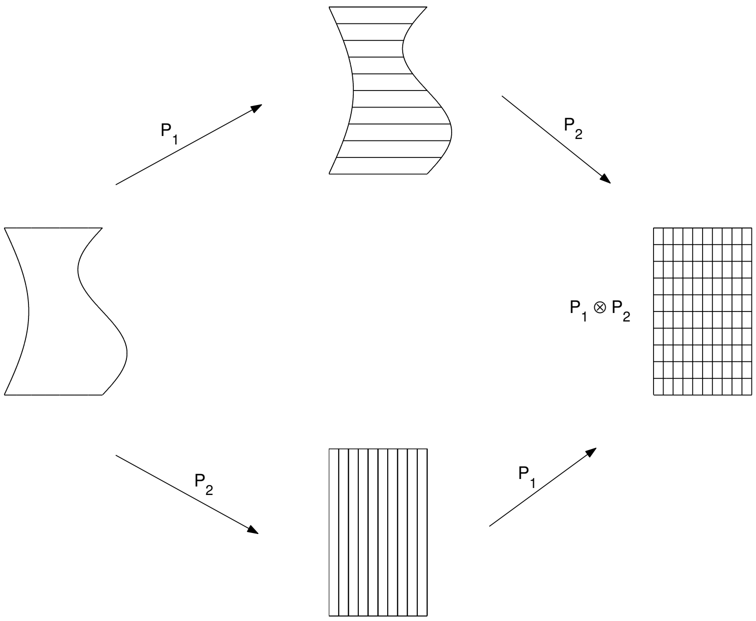

Abbildung 2.8: Unidirektionale Interpolationen und Tensorprodukt.

Das Tensorprodukt $P_1[x] \otimes P_2[x]$ stimmt genau an den Eckpunkten mit x überein, während das für den Rest des Randes im Allgemeinen nicht gilt, wie es in der Abbildung 2.8 an der linken und der rechten Seite zu sehen ist. Das Tensorprodukt liefert also noch keine Interpolation mit den gewünschten Eigenschaften. Dies führt uns zur Boolschen Summe.
Betrachtet man die Summe $S(\xi_1, \xi_2) := P_1[x] + P_2[x]$ zum Beispiel am "$\xi_1 = 0$ - Rand":

$$S(0, \xi_2) = x(0, \xi_2) + P_2[x](0, \xi_2),$$

so stellt man fest, dass $S(0, \xi_2)$ genau um den zweiten Summanden vom Rand abweicht. Dasselbe passiert am "$\xi_1 = 1$ - Rand":

$$S(1, \xi_2) = x(1, \xi_2) + P_2[x](1, \xi_2).$$

Die Idee ist nun, als Interpolation die Summe $S(\xi_1, \xi_2)$ ohne die Fehlerterme zu benutzen. Dafür werden die beiden Fehlerterme in ξ_1 Richtung interpoliert:

$$R(\xi_1, \xi_2) := P_1\big[P_2\,[x]\big].$$

Das Ergebnis dieser Interpolation ist aber genau $P_1[x] \otimes P_2[x]$. Also trifft die Differenz

$$S - R = P_1[x] + P_2[x] - P_1[x] \otimes P_2[x]$$

den Rand genau. Diser Ausdruck wird als Boolsche Summe bezeichnet:

$$P_1[x] \oplus P_2[x] := P_1[x] + P_2[x] - P_1[x] \otimes P_2[x].$$

Er stellt zugleich bereits die Transfinite Interpolation in zwei Raumdimensionen dar.

Rekursive Berechnung

Wendet man die unidirektionale Interpolation rekursiv an, lässt sich die Boolsche Summe als Sequenz unidirektionaler Interpolationen darstellen.

Die unidirektionale Interpolation $P_i[x]$ ist linear in x. Denn aus den Definitionen von β_i^{kl} (2.1) und β_{ij}^{klmn} (2.2) folgt:

$$\begin{aligned}
P_i[x - P_j[x]] = {} & \sum_{l=1}^{L_i}\sum_{m=0}^{Q} \alpha_i^{lm}(\xi_i)\left[\frac{\partial^m x}{\partial \xi_i}|_{\xi_i=\xi_i^l} - \sum_{k=1}^{L_j}\sum_{n=0}^{R} \alpha_j^{kn}(\xi_j)\frac{\partial^{n+m} x}{\partial \xi_j^n \partial \xi_i^m}|_{\xi_i=\xi_i^l \;\, \xi_j=\xi_j^k}\right] \\
= {} & \sum_{l=1}^{L_i}\sum_{m=0}^{Q} \alpha_i^{lm}(\xi_i)\left[\beta_i^{lm} - \sum_{k=1}^{L_j}\sum_{n=0}^{R} \alpha_j^{kn}(\xi_j)\beta_{ij}^{klnm}\right] \\
= {} & \sum_{l=1}^{L_i}\sum_{m=0}^{Q} \alpha_i^{lm}(\xi_i)\beta_i^{lm} - \sum_{l=1}^{L_i}\sum_{m=0}^{Q} \alpha_i^{lm}(\xi_i)\sum_{k=1}^{L_j}\sum_{n=0}^{R} \alpha_j^{kn}(\xi_j)\beta_{ij}^{klnm} \\
= {} & P_i[x] - P_i\big[P_j[x]\big].
\end{aligned} \tag{2.5}$$

Nun ist:

$$P_1[x] \oplus P_2[x] = P_1[x] + P_2[x] - P_1\big[P_2[x]\big],$$

und mit der Linearität (2.5) sowie der Symmetrie (2.3) erhält man die Boolsche Summe als Sequenz unidirektionaler Interpolationen:

$$P_1[x] \oplus P_2[x] = P_1[x] + P_2\big[x - P_1[x]\big].$$

Betrachtet man eine einzelne unidirektionale Interpolation als eindimensinale Transfinite Interpolation, so lässt sich die Transfinite Interpolation rekursiv für höhere Raumdimensionen definieren:

$$F_i := F_{i-1} + P_{i+1}[x - F_{i-1}] \;\text{ für }\; i = 1, 2, 3, \ldots.$$

F_i bezeichne dabei die Transfinite Interpolation in der jeweiligen Raumdimension. Die in dieser Arbeit benutzte Transfinite Interpolation in drei Dimensionen F_3 berechnet sich demnach als

$$\begin{aligned}
F_1[x] = {} & P_1[x] \\
F_2[x] = {} & F_1[x] + P_2\big[x - F_1[x]\big] \\
F_3[x] = {} & F_2[x] + P_3\big[x - F_2[x]\big].
\end{aligned}$$

2.4 Gittergenerierung mittels Lösungen transformierter elliptischer Differentialgleichungen

Gesucht ist eine Abbildung $x : \overline{\Xi} \longrightarrow \overline{X}$, die bijektiv ist und den Rand von Ξ auf den Rand von X abbildet.

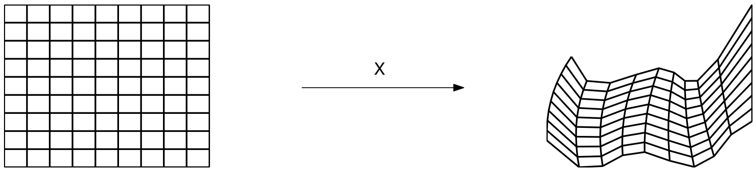

Abbildung 2.9: x bildet $\overline{\Xi}$ auf $\overline{X}$ ab.

Diese Bedingungen können erfüllt werden, wenn man die Koordinatenfunktionen $x_i, i = 1, 2, 3$ als Lösungen eines Systems elliptischer Differentialgleichungen wählt.

Definition 1 *sei $X \subset \mathbb{R}^3$ offen und zusammenhängend, sowie $\xi_l : X \longrightarrow \mathbb{R}$ eine Funktion auf X gegeben. Ein Diffentialoperator $L(\xi_l) = \sum_{i,j=1}^{3} a_{ij} \frac{\partial^2 \xi_l}{\partial x_i \partial x_j}$ heißt* elliptisch im Punkt $x = (x_1, x_2, x_3)$, *falls es eine positive Zahl $\mu(x)$ gibt, so dass*

$$\sum_{i,j=1}^{3} a_{ij}(x)\zeta_i\zeta_j \geq \mu(x) \sum_{i=1}^{3} \zeta_i^2 \tag{2.6}$$

für alle 3-Tupel reeller Zahlen $(\zeta_1, \zeta_2, \zeta_3)$. Der Operator L heißt elliptisch in dem Gebiet X, *wenn L in jedem Punkt aus X elliptisch ist. L heißt* uniform elliptisch in X, *falls L elliptisch auf X ist, und es eine positive Konstante $\mu_0 > 0$ gibt, so dass $\mu(x) \geq \mu_0$ für alle x aus X gilt.*

Für elliptische Operatoren gilt das *Maximumprinzip von E. Hopf*:

Satz 1 *Sei $\xi_l(x_1, x_2, x_3)$, so dass folgende Ungleichung auf einem offenen und zusammenhängenden Gebiet $X \subset \mathbb{R}^3$ erfüllt ist:*

$$L(\xi) = \sum_{i,j=1}^{d} a_{ij}(x) \frac{\partial^2 \xi_l}{\partial x_i \partial x_j} \geq 0. \tag{2.7}$$

Seien außerdem die Koeffizienten a_{ij} uniform beschränkt und L uniform elliptisch auf X.
Falls ξ_l sein Maximum M auf X annimmt, so ist $\xi_l \equiv M$ auf X.

Einen Beweis hierzu findet man in [30, Kapitel 3]. Eine Funktion, die den Bedingungen des Satzes 1 entspricht, nimmt ihr Maximum also auf dem Rand von X an. Bei elliptischen Differentialgleichungen kann man auf dem ganzen Rand Werte für

die Lösung, sogenannte Dirichlet-Randwerte, vorgeben. Dadurch kann man erreichen, dass der Rand von Ξ mittels $x = (x_1, x_2, x_3)$ auf den Rand von X abgebildet wird.
Diese Eigenschaften elliptischer Differentialgleichungen kann man nutzen, um sicherzustellen, dass das Bild von $\overline{\Xi}$ unter $x = (x_1, x_2, x_3)$ in $\overline{X}$ enthalten ist, falls X *quaderförmig* ist.

Definition 2 *Ein Gebiet $A \subset \mathbb{R}^3$ heißt* quaderförmig, *falls es $a_i, b_i \in \mathbb{R}$ mit $a_i < b_i$ und $i = 1, 2, 3$ gibt, so dass gilt:*

$$A = \{(x_1, x_2, x_3) \in \mathbb{R}^3 \mid a_i < x_i < b_i\}. \tag{2.8}$$

Satz 2 *Sei X quaderförmig und seien die Koordinatenfunktionen x_i Lösungen des Laplace-Systems $\Delta x_i = 0$ für $i = 1, 2, 3$ der Art, dass mittels $(x_1, x_2, x_3) = x : \overline{\Xi} \longrightarrow \mathbb{R}^3$ der Rand $\partial\Xi$ von Ξ auf den Rand ∂X von X abgebildet wird. Dann gilt $x(\overline{\Xi}) \subset \overline{X}$.*

Beweis:
Falls die Koordinatenfunktionen $x_i,\ i = 1, 2, 3$ die Laplace-Gleichung $\Delta x_i = 0$ erfüllen, gilt das Maximumprinzip sowohl für x_i als auch für $-x_i$. Das heißt, dass die Koordinatenfunktionen ihre Extrema auf dem Rand von Ξ annehmen. Laut Definition des Gebietes X sind die Minima beziehungsweise Maxima der Koordinatenfunktionen x_i genau a_i beziehungsweise b_i, $i = 1, 2, 3$. Das heißt, es gilt für alle $\xi \in \Xi$ und $i = 1, 2, 3 :\ a_i < x_i(\xi) < b_i$, also $x(\xi) \in \overline{X}$ für alle $\xi \in \overline{\Xi}$. □

Nun ist aber im Allgemeinen das Gebiet X nicht quaderförmig. Auf der anderen Seite kann man das Gebiet Ξ frei wählen, hier $(0, 1)^3$ und so die Quaderförmigkeit von Ξ garantieren. Wenn man also sicherstellen möchte, dass das Bild von Ξ in X enthalten ist, ist es naheliegend, die Umkehrabbildung $\xi : \overline{X} \longrightarrow \overline{\Xi}$ zu betrachten.
Die Vorgehensweise ist wie folgt: Wähle die Koordinatenfunktionen der Umkehrabbildung $\xi = (\xi_1, \xi_2, \xi_3)$ als Lösungen elliptischer Differentialgleichungen, die das Maximumprinzip erfüllen. Da aber die (kartesischen) Koordinatenlinien in $\overline{\Xi}$ bekannt sind, während die Koordinatenlinien beziehungsweise Koordinatenfunktionen in $\overline{X}$ gesucht sind, wird eine Koordinatentransformation durchgeführt, um die Koordinatenfunktionen in $\overline{X}$ als abhängige Variablen zu erhalten.
Im Folgenden wird erst der Fall betrachtet, in dem die Abbildung ξ ein Laplace-System erfüllt. Hier kann bewiesen werden, dass die daraus entstehende Gitterabbildung bijektiv und das resultierende Gitter frei von degenerierten Gitterzellen ist. Danach wird auf die Eigenschaften von durch Laplace-Systeme generierten Gittern eingegangen, was zu einer Motivation für die Benutzung von Poisson-Systemen führt.

Anschließend wird eine Modifikation des Laplace-Systems vorgestellt, welches eine bessere Kontrolle über den Verlauf der Gitterlinien im Inneren des Gebietes bietet, ohne dass die Bijektivität der Abbildung verloren geht.

2.4.1 Laplace-System

Hier wird das Laplace-System

$$\Delta\xi_l = 0, \quad l = 1, 2, 3 \tag{2.9}$$

mit Randbedingungen, so dass ∂X bijektiv auf $\partial\Xi$ abgebildet wird, zur Generierung der Koordinatenfunktionen gewählt. Hierbei sind wieder ξ_l die Koordinatenfunktionen der Abbildung ξ. Die Existenz und Eindeutigkeit der Lösung auf $\overline{\Xi}$ sind durch die Sätze 2.14 beziehungsweise 2.2 in [8] gegeben. Diese Sätze gelten auch noch im Fall des im nächsten Kapitel behandelten Poisson-Systems.
Satz 5 liefert die transformierte Differentialgleichung, die gelöst werden muss, um die gesuchte Abbildung x zu erhalten. Für den Beweis von Satz 5 benötigt man die folgenden Sätze 3 und 4, die zuerst bewiesen werden.

Satz 3 *Es gelte* (2.9). *Außerdem sei* $\xi|_{\partial X} : \partial X \longrightarrow \partial\Xi$ *differenzierbar auf* $\partial X \setminus \{Kanten\}$. *Die Kanten sind dabei die Kurven in* $\overline{X}$, *die auf die Kanten des Würfels* $\overline{\Xi}$ *abgebildet werden.*
Dann ist $det(\nabla\xi) \neq 0$ *für alle* $x \in X$.

Den Beweis hierzu findet man in [21]. Wesentliche Aussagen für den Beweis sind in [15, pp. 273-275] nachzulesen.

Satz 4 *Sei* $\xi : \overline{X} \longrightarrow \overline{\Xi}$ *die Transformationsabbildung, deren Koordinatenfunktionen* (2.9) *erfüllen und sei* Ξ *quaderförmig.*
Falls die Determinante der Jacobimatrix $\det(\nabla\xi) \neq 0$ *für alle* $x \in X$ *ist, so ist* ξ *eine bijektive Abbildung.*

Beweis:
Nimm an, ξ wäre nicht injektiv, dann gäbe es zwei Punkte $p_1, p_2 \in X$ mit

$$\xi(p_1) = \xi(p_2) = (u_0, v_0, w_0).$$

Definiere folgende Niveaumengen in $\overline{X}$:

$$L_1 = \left\{p \in \overline{X} | \xi_1(p) = u_0\right\}$$
$$L_2 = \left\{p \in \overline{X} | \xi_2(p) = v_0\right\}$$
$$L_3 = \left\{p \in \overline{X} | \xi_3(p) = w_0\right\}$$

Sei $K = L_2 \cap L_3$ und p ein Punkt in $K \cap X$. Zeige zuerst, dass es für $p \in K$ eine glatte Kurve C durch p mit Endpunkten in $K \cap X$ gibt. Anschließend wird die Beweisannahme zum Widerspruch geführt. Dabei wird mehrmals der Satz über lokale Umkehrbarkeit (Umkehrsatz), nachzulesen in [18, Kap. 3], benutzt.
Eine Folge des Umkehrsatzes ist, dass K in einer Umgebung von p eine zusammenhängende Kurve ist. Seien dazu q_1 und $q_2 \in K$ und $\xi(q_1) = (u_1, v_0, w_0)$ und $\xi(q_2) = (u_2, v_0, w_{,0})$. Dann ist wegen der Konvexität von Ξ $k(t) = (u_1 + t(u_2 - u_1), v_0, w_0), \quad t \in (0,1)$ eine (zusammenhängende) Kurve in Ξ. k ist eine kompakte Teilmenge von Ξ, so dass es eine offene Überdeckung von k aus endlich vielen Teilmengen von Ξ gibt. Jede dieser offenen Mengen lässt sich diffeomorph auf eine offene Teilmenge in X abbilden, und die Vereinigung der Bilder der Teilkurven von k ist eine zusammenhängende Kurve in X. Aus der Definition von k folgt $k \subset K$. Also gibt es für $p \in K \cap X$ eine offene Umgebung $N \subset X$, so dass $K \cap N$ eine glatte Kurve C enthält. Aus der weiteren Argumentation wird folgen, dass $K \cap N$ tatsächlich eine glatte Kurve ist und nicht etwa eine Fläche. Setze C nun stetig nach $\partial X \cap K$ ($\neq \emptyset$ wegen der Randbedingungen) fort. Aufgrund der Randbedingungen enthält $\partial X \cap K$ höchstens zwei Punkte. Wenn die beiden Endpunkte von C übereinstimmen, hat die Funktion ξ_1 ein Maximum in einem Punkt $q \in X \setminus \partial X$, und $\nabla \xi$ ist in keiner Umgebung von q injektiv, im Widerspruch zum Umkehrsatz. Also hat C zwei verschiedene Punkte aus $K \cap \partial X$ als Endpunkte. Aus demselben Grund kann auch $K \cap N$ keine Fläche enthalten. Es gäbe sonst eine geschlossene Kurve in $K \cap N$, die einen kritischen Punkt von ξ_1 enthalten würde.
Betrachte nun die beiden Punkte p_1 und p_2 mit $\xi(p_1) = \xi(p_2)$. p_1 und p_2 liegen in $(L_1 \cap L_2 \cap L_3) \subset K$, also gibt es Kurven C_1 und C_2 in K, die jeweils p_1 beziehungsweise p_2 enthalten und deren Endpunkte in $\partial X \cap K$ liegen. Da K lokal eine glatte Kurve ist, stimmen C_1 und C_2 auf einem Teilstück A überein, sofern sie sich schneiden. Das Teilstück A hat p_1 und p_2 als Endpunkte. Dies führt, wie oben, zum Widerspruch, da ξ_1 einen kritischen Punkt auf A haben müsste.
Nimm nun an, C_1 und C_2 seien derart, dass sie nur Anfangs- und Endpunkte gemeinsam haben. Nun ist L_2 eine Fläche, das folgt aus einer ähnlichen Argumentation, wie sie benutzt wurde, um zu zeigen, dass $K \cap N$ eine Kurve ist: Die Hyperfläche, gegeben durch $\{(\xi_1, \xi_2, \xi_3) \in \overline{\Xi} \mid \xi_2 = v_0\}$, ist eine kompakte Teilmenge von $\overline{\Xi}$. Es gibt also eine offene Überdeckung von endlich vielen Teilmengen von $\overline{\Xi}$, so dass x auf diesen Teilmengen ein Diffeomorphismus ist. Die Vereinigung der Bilder dieser Teilmengen ist die Fläche L_2.
Sei S die durch $C_1 \cup C_2$ begrenzte Teilmenge von L_2. Die Menge $L_1 \cap S$ enthält p_1, also gibt es in einer Umgebung von p_1 eine glatte Kurve C_0 in S mit p_1 als Endpunkt. Setze C_0 als Teilmenge von L_1 in S fort, bis C_0 die Kurve $C_1 \cup C_2$ in einem Punkt p_3 schneidet. Es gilt $p_3 \in L_1 \cap L_2 \cap L_3$ und wegen $L_1 \cap L_2 \cap L_3 \cap \partial X = \emptyset$ liegt p_3 im Inneren von X. Wie vorher führt dies zum Widerspruch, da die $\xi_i, i = 1, 2, 3$ Extrema auf C_0 haben müssten. Damit ist ξ injektiv.

Surjektivität: Seien nun (u_0, v_0, w_0) ein beliebiger Punkt in Ξ und L_1 und L_2 definiert wie vorher, dann enthält $K = L_1 \cap L_2$, wie oben gezeigt, eine glatte Kurve C, deren Endpunkte die beiden Punkte in $\partial X \cap K$ sind. Wegen der Randbedingungen nimmt ξ_1 sein Maximum und sein Minimum auf ∂X an, und wegen der Stetigkeit von ξ_1 wird der Wert u_0 auf dem Kompaktuum C angenommen. $\square$

Satz 5 *ξ_l sind genau dann Lösungen von (2.9), wenn x_l Lösungen des quasilinearen elliptischen Systems*

$$\sum_{i,j=1}^{3} g^{ij} \frac{\partial^2 x_l}{\partial \xi_i \partial \xi_j} = 0 \quad \text{für } l = 1, 2, 3 \tag{2.10}$$

mit Randbedingungen, so dass $\partial\Xi$ bijektiv auf ∂X abgebildet wird, ist. Dabei sind $g^{ij} = \nabla\xi_i \cdot \nabla\xi_j$.

Beweis: "⇒": Seien ξ_l Lösungen von (2.9) und $\Delta_X := \sum_{i=1}^{3} \frac{\partial^2}{\partial x_i \partial x_i}$.
Es gilt mit der Kettenregel:

$$\begin{aligned}
0 = \Delta_X x_l(\xi_1, \xi_2, \xi_3) &\overset{\xi_i = \xi_i(x_1, x_2, x_3)}{=} \sum_{i,j=1}^{3} \frac{\partial}{\partial x_i} \left(\frac{\partial x_l}{\partial \xi_j} \frac{\partial \xi_j}{\partial x_i} \right) \\
&= \sum_{i,j,k=1}^{3} \frac{\partial^2 x_l}{\partial \xi_j \partial \xi_k} \frac{\partial \xi_k}{\partial x_i} \frac{\xi_j}{\partial x_i} + \sum_{i,j=1}^{3} \frac{\partial x_l}{\partial \xi_j} \frac{\partial^2 \xi_j}{\partial x_i \partial x_i} \\
&= \sum_{j,k=1}^{3} g^{jk} \frac{\partial^2 x_l}{\partial \xi_j \partial \xi_k} + \sum_{j=1}^{3} \frac{\partial x_l}{\partial \xi_j} \sum_{i=1}^{3} \frac{\partial^2 \xi_j}{\partial x_i \partial x_i} \\
&= \sum_{j,k=1}^{3} g^{jk} \frac{\partial^2 x_l}{\partial \xi_j \partial \xi_k} + \sum_{j=1}^{3} \frac{\partial x_l}{\partial \xi_j} \underbrace{\Delta_X \xi_j}_{=0} \\
&= \sum_{j,k=1}^{3} g^{jk} \frac{\partial^2 x_l}{\partial \xi_j \partial \xi_k}.
\end{aligned}$$

Zeige nun, dass (g^{jk}) positiv definit ist und damit die Elliptizität von (2.10).

$$g^{jk} = \nabla\xi_j \cdot \nabla\xi_k,$$

also ist g^{jk} symmetrisch. Laut Satz 3 gilt $det(\nabla\xi) \neq 0$, falls $\xi_i, i = 1, 2, 3$ (2.9) erfüllen. Also ist

$$det((g^{jk})) = det^2(\nabla\xi) > 0,$$

und damit ist (g^{jk}) positiv definit.
"$\Leftarrow$": Sei (2.10) erfüllt, dann folgt aus obiger Gleichung auch (2.9):

$$\begin{aligned} 0 &= \Delta_X x_l(\xi_1, \xi_2, \xi_3) \\ &= \underbrace{\sum_{i,j,k=1}^{3} \frac{\partial^2 x_l}{\partial \xi_j \partial \xi_k} \frac{\partial \xi_k}{\partial x_i} \frac{\partial \xi_j}{\partial x_i}}_{=0} + \sum_{i,j=1}^{3} \frac{\partial x_l}{\partial \xi_j} \frac{\partial^2 \xi_j}{\partial x_i \partial x_i} \\ &= \sum_{i,j=1}^{3} \frac{\partial^2 \xi_j}{\partial x_i \partial x_i} \frac{\partial x_l}{\partial \xi_j} \quad \text{für } l = 1, 2, 3. \end{aligned}$$

Schreibt man dies als lineares Gleichungssystem, so erhält man:

$$\begin{pmatrix} \frac{\partial x_1}{\partial \xi_1} & \frac{\partial x_1}{\partial \xi_2} & \frac{\partial x_1}{\partial \xi_2} \\ \frac{\partial x_2}{\partial \xi_1} & \frac{\partial x_2}{\partial \xi_2} & \frac{\partial x_2}{\partial \xi_2} \\ \frac{\partial x_3}{\partial \xi_1} & \frac{\partial x_3}{\partial \xi_2} & \frac{\partial x_3}{\partial \xi_2} \end{pmatrix} \begin{pmatrix} \Delta \xi_1 \\ \Delta \xi_2 \\ \Delta \xi_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}. \tag{2.11}$$

Die Matrix ist die Inverse der Jacobimatrix $\nabla \xi$ und damit regulär.
Damit folgt $\Delta \xi_i = 0$ für $i = 1, 2, 3$. □

2.4.2 Poisson-System

Die durch das transformierte Laplace-System (2.10) generierten Gitter haben Eigenschaften, die die Kontrolle über den Verlauf der Gitterlinien durch Vorgabe von Gitterpunkten am Rand erschweren. Dies geht im Wesentlichen auf zwei Phänomene zurück:

1. Die Lösungen von (2.10) tendieren dazu, die Gitterpunkte gleichmäßig über das Gebiet zu verteilen. Gibt man am Rand also nicht äquidistante Punkte vor, so verlaufen die zugehörigen Koordinatenlinien mit zunehmendem Abstand zum Rand eher äquidistant. In manchen Fällen ist dieser Effekt unerwünscht, so dass das geglättete Gitter sogar schlechter ist als das interpolierte Ausgangsgitter.

2. Die Form des Gebietes, genauer gesagt, der Verlauf des Randes von X hat entscheidenden Einfluss auf die Lösung und damit auf den Verlauf der Gitterlinien.

Das erste Phänomen hängt damit zusammen, dass man (2.9) auch als Euler-Lagrange-Gleichungen des Dirichlet-Problems

$$\textit{minimiere } \frac{1}{2} \int_X \sum_{i=3}^{3} |\nabla \xi_i|^2 dx, \quad \xi_i|_{\partial X} \text{ erfüllen die Randbedingungen}$$

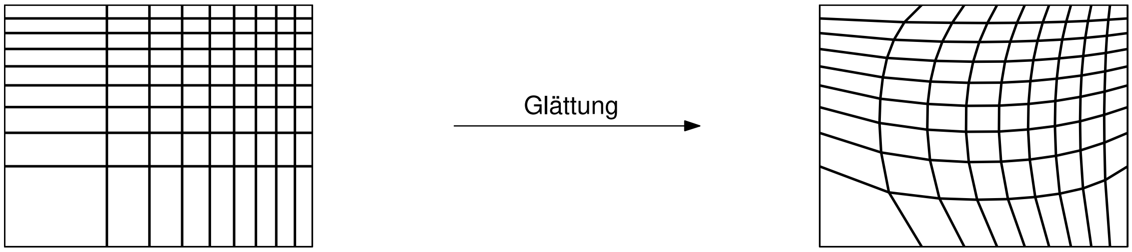

Abbildung 2.9:
links: interpoliertes Gitter
rechts: durch Laplace-System geglättetes Gitter

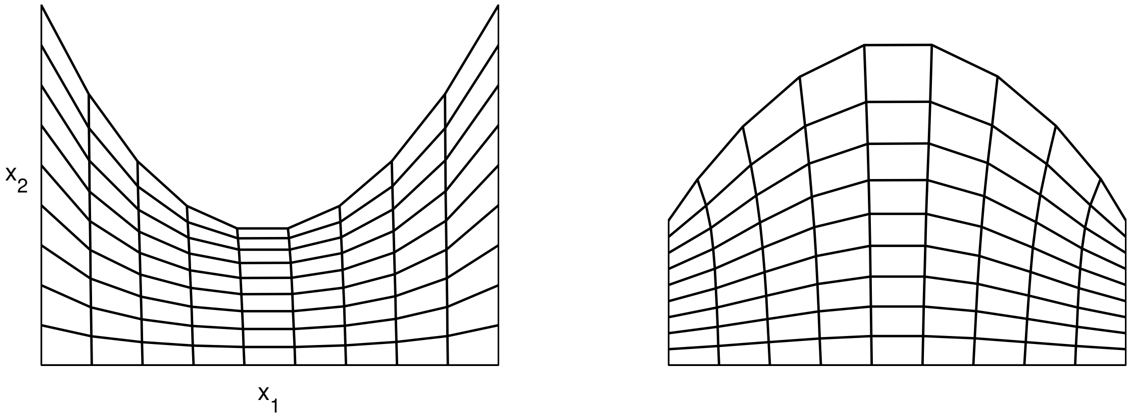

Abbildung 2.9:
links: konkaver Rand, lokale Häufung von Gitterlinien
rechts: konvexer Rand, lokale Ausdünnung von Gitterlinien

erhält. Siehe dazu [8, Kap. 11.5]. Da immer auf ein äquidistantes Gitter in Ξ abgebildet wird, bedeutet ein betragsmäßig großer Gradient von ξ_i in einer Umgebung von x, dass die Koordinatenlinien in ξ_i-Richtung in einer Umgebung von x dichter liegen. Eine Lösung des Minimierungsproblems vermeidet also lokale Häufungen von Gitterlinien und verteilt sie gleichmäßiger. Eine Kontrolle über die Maschenweite des Gitters im Inneren des Gebietes durch Vorgabe von Punkten am Rand ist also nicht gegeben.
Das zweite Phänomen, der Einfluss des Randes auf die Lösung, verhält sich wie folgt: Ist ∂X konkav, so konzentrieren sich Gitterlinien am Rand. Im umgekehrten Fall verlaufen die Gitterlinien eher im Inneren von X. Zur Erklärung reicht es aus, den zweidimensionalen Fall zu betrachten. Schaut man sich ein konkaves Stück Rand von X an, zum Beispiel die Kurve mit $\xi_2 \equiv 1$, so ergibt sich wegen der Konkavität von ∂X hier $\frac{\partial^2 \xi_2}{\partial^2 x_1} > 0$. Damit die Laplace-Gleichung erfüllt ist, muss also $\frac{\partial^2 \xi_2}{\partial^2 x_2} < 0$ gel-

ten. Daraus folgt, dass die Flächen mit $\xi_2 \equiv const.$ mit abnehmenden x_2 (von oben nach unten, siehe linken Teil der Abbildung 2) immer weiter auseinander liegen, die Gitterlinien also am Rand dichter liegen als im Inneren. Dabei wird immer davon ausgegangen, dass in Ξ ein äquidistantes Gitter betrachtet wird. Im konkaven Fall argumentiert man analog mit umgekehrtem Vorzeichen, es muss $\frac{\partial^2 \xi_2}{\partial^2 x_2} > 0$ gelten, so dass die $\xi_2 \equiv const$-Linien mit abnehmendem x_2 dichter zusammen liegen.
In der Praxis möchte man jedoch durch Vorgabe der Gitterpunkte am Rand das ganze Gitter beeinflussen, um zum Beispiel an bestimmten Stellen im Inneren des Gebietes Gitterpunkte zu konzentrieren. Dies kann man erreichen, indem man sich das gerade beschriebene Phänomen zunutze macht: Statt des Laplace-Systems wird nun ein Poisson-System $\Delta x_l = f, \;\; l = 1,2,3$ mit einer Funktion $f : X \longrightarrow \mathbb{R}$ gelöst. Über die Funktion f kann nun der Einfluss des Randes von X kompensiert werden. Im Fall eines konkaven Randes, wie oben beschrieben, kann man dort beispielsweise die rechte Seite lokal vergrößern, um eine andere Bedingung für $\frac{\partial^2 \xi_2}{\partial^2 x_2}$ zu erhalten. Darüber hinaus kann man den Einfluss der rechten Seite auch direkt nutzen, um die Verteilung der Gitterpunkte zu beeinflussen. So ist es möglich, mithilfe der rechten Seite Punkte im Inneren des Gebietes zu konzentrieren, ohne Verteilung der Randpunkte zu ändern. In der Praxis wird dies auch durchgeführt, siehe zum Beispiel [34, Kap. VI 2]. Jedoch kann die Bijektivität der Abbildung x hier nicht mehr garantiert werden, so dass eine Faltung des Gitters nicht ausgeschlossen werden kann [20, p. 166].
Im Folgenden wird eine Methode besprochen, die das erste Phänomen behandelt. Durch Modifikation der rechten Seite wird das Gleichungssystem so geändert, dass die in Abbildung 1 zu sehenden Effekte nicht auftreten. Vom Gebietsrand her vorgegebene nicht äquidistante Abstände der Gitterlinien werden im Inneren also beibehalten. Bei dieser Methode kann weiterhin garantiert werden, dass die erhaltene Gitterabbildung x bijektiv ist.

2.4.3 Modifiziertes Laplace-System

Um die Gitterabbildung zu konstruieren, wird ein zusätzlicher *Parameterraum* $P = (0,1)^3$ eingeführt. Die Gitterabbildung $x : \overline{\Xi} \longrightarrow \overline{X}$ wird nun durch Komposition einer Abbildung $p : \overline{\Xi} \longrightarrow \overline{P}$ und einer Abbildung $\tilde{\xi}^{-1} : \overline{P} \longrightarrow \overline{X}$ erzeugt. Dabei ist $\tilde{\xi}$ eine Abbildung, wie sie im vorherigen Abschnitt besprochen wurde. Das heißt, die Koordinatenfunktionen der Umkehrabbildung $\tilde{\xi}$ erfüllen die Laplace-Gleichung

$$\begin{aligned} &\Delta\tilde{\xi}_l = 0, \quad l = 1,2,3 \\ &\text{mit Randbedingungen}: \\ &\tilde{\xi}|_{\partial X} : \partial X \longrightarrow \partial P. \end{aligned} \tag{2.12}$$

Jedoch ist das der Berechnung von $\tilde{\xi}$ zugrundeliegende Gitter in P möglicherwei-

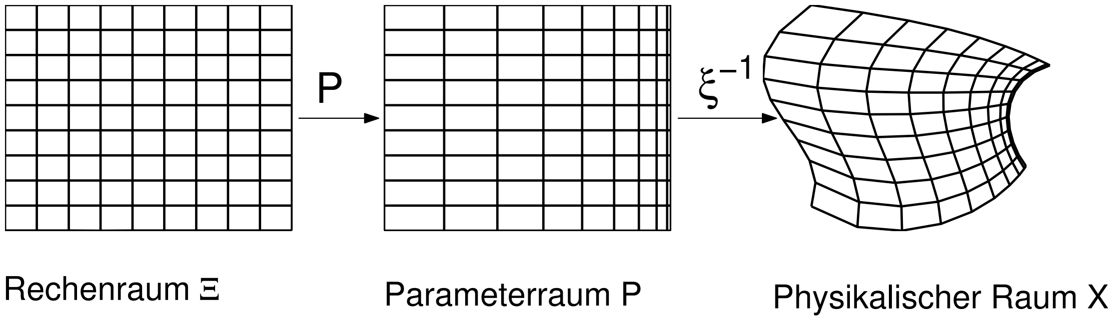

Abbildung 2.10: $\overline{\Xi}$ wird erst auf einen Parameterraum P abgebildet.

se nicht äquidistant. Es ist gegeben durch die Abbildung p, welche in Abhängigkeit von der Randpunkteverteilung in $\overline{X}$ kontstruiert wird. Entscheidend hierbei ist, dass die Abbildung p bijektiv ist und damit die Gitterabbildung $x = \tilde{\xi}^{-1} \circ p$ ebenfalls. Des Weiteren muss $p|_{\partial\Xi} = \partial P$ gelten, damit sichergestellt ist, dass $\partial\Xi$ auf ∂X abgebildet wird. Wie schon zuvor wird das Gleichungssystem transformiert, um die Koordinatenfunktionen der Gitterabbildung x_i als abhängige Variablen zu erhalten. Der Unterschied zum vorher besprochenem Laplace-System (2.9) besteht in der Zwischenschaltung der Abbildung p, die das Laplace-System (2.12) mittels p^{-1} auf $\overline{\Xi}$ transformiert. Betrachtet man das Gleichungssystem (2.12) auf $\overline{\Xi}$, so wird hier ein Poisson-System gelöst.
Im Folgenden wird zuerst für eine gegebene bijektive Abbildung p die Transformation des Gleichungssystems $\Delta\tilde{\xi}_l = 0, \quad l = 1, 2, 3$ auf $\overline{X}$ hergeleitet. Anschließend wird die Berechnung der bijektiven Abbildung p aus der Randpunkteverteilung in $\overline{X}$ erklärt.

Sei wie gehabt $\Delta_X := \sum_{i=1}^{3} \frac{\partial^2}{\partial x_i^2}$ und $\xi : \overline{X} \longrightarrow \overline{\Xi}$. Es gilt dann mit der Kettenregel für $l = 1, 2, 3$

$$\Delta_X p_l \circ \xi = \sum_{j,k=1}^{3} g^{jk} \frac{\partial^2 p_l}{\partial \xi_j \partial \xi_k} + \sum_{j=1}^{3} \frac{\partial p_l}{\partial \xi_j} \Delta_X \xi_j. \tag{2.13}$$

Da $\xi, \tilde{\xi}$ und p alle bijektiv sind, gilt

$$p_l \circ \xi = \tilde{\xi}_l.$$

Es gilt also wegen (2.12)

$$0 = \sum_{j,k=1}^{3} g^{jk} \frac{\partial^2 p_l}{\partial \xi_j \partial \xi_k} + \sum_{j=1}^{3} \frac{\partial p_l}{\partial \xi_j} \Delta_X \xi_j. \tag{2.14}$$

Mit

$$P_{ij} = -T^{-1} \begin{pmatrix} \frac{\partial^2 p_1}{\partial\xi_i \partial\xi_j} \\ \frac{\partial^2 p_2}{\partial\xi_i \partial\xi_j} \\ \frac{\partial^2 p_3}{\partial\xi_i \partial\xi_j} \end{pmatrix} \tag{2.15}$$

und der 3×3 Matrix T, definiert als

$$T = \left(\frac{\partial p_i}{\partial \xi_j} \right)_{ij}, \tag{2.16}$$

erhält man aus (2.14) das Poisson-System

$$\begin{pmatrix} \Delta_X \xi_1 \\ \Delta_X \xi_2 \\ \Delta_X \xi_3 \end{pmatrix} = \sum_{i,j=1}^{3} g^{ij} P_{ij}. \tag{2.17}$$

Die insgesamt 27 Einträge der neun Vektoren P_{ij} sind die sogenannten *Kontrollfunktionen.* Wegen (2.15) gilt $P_{ij} = P_{ji}$. Mit (2.17) und (2.16) sind sie für eine gegebene Abbildung $p \in C_2(\Xi), \quad p : \overline{\Xi} \longrightarrow \overline{P}$ wohldefiniert.
Nun wird das Poisson-System, wie im vorherigen Abschnitt das Laplace-System, transformiert, um x_l als abhängige Variablen zu erhalten.
Es gilt

$$\Delta_X x_l = 0 \quad \text{für} \quad l = 1, 2, 3.$$

Und damit auch

$$0 = \Delta_X x_l \circ \xi = \sum_{j,k=1}^{3} g^{jk} \frac{\partial^2 x_l}{\partial \xi_j \partial \xi_k} + \sum_{j=1}^{3} \Delta_X \xi_j \frac{\partial x_l}{\partial \xi_j}.$$

Mit (2.17) erhält man schließlich das für die Gittergenerierung zu lösende System

$$0 = \sum_{i,j=1}^{3} g^{ij} \frac{\partial^2 x_l}{\partial \xi_i \partial \xi_j} + \sum_{k=1}^{3} \sum_{i,j=1}^{3} g^{ij} P_{ij}^k \frac{\partial x_l}{\partial \xi_k} \quad l = 1, 2, 3. \tag{2.18}$$

Dabei ist P_{ij}^k die k-te Komponete des Vektors P_{ij}. Falls alle Kontrollfunktionen gleich null sind, stimmt (2.18) mit dem Laplace-System (2.9) überein. Dies ist genau dann der Fall, wenn die Abbildung p gerade die Identität ist.

Berechnung der Kontrollfunktionen

Die nachfolgend konstruierten Kontrollfunktionen sollen sicherstellen, dass die am Rand vorgegebene Verteilung der Gitterpunkte im Inneren des Gebietes erhalten bleibt. Dies geschieht, indem man die Parameterabbildung wählt, wie in Abbildung 2.10. Das heißt, man passt den Parameterraum so an, dass die Effekte, die zur Verzerrung des Gitters führen, nicht mehr auftreten. Anschaulich gesprochen, wählt man das Gitter im Parameterraum so, dass es im selben Maße nicht äquidistant ist, wie das zu generierende Gitter in $\overline{X}$. Konstruiere nun die Koordinatenfunktionen $p_i : \overline{\Xi} \longrightarrow \overline{P}$ des Parameterraumes P. Aus diesen können dann mittels (2.15) und (2.16) die Kontrollfunktionen berechnet werden. Die Bedingungen an p sind am Rand durch zwei Anforderungen gegeben. Erstens durch die zu erfüllenden Randbedingungen der komponierten Abbildung $\tilde{\xi}^{-1} \circ p$ und zweitens durch die Bedingung, die Maschenweiten der Randpunkteverteilung in $\overline{X}$ zu übernehmen. So wird die Abbildung p auch erst an den Kanten berechnet und dann mittels Interpolation auf ganz $\overline{P}$ fortgesetzt. Diese Art der Konstruktion der Kontrollfunktionen geht auf Spekreijse [33] zurück.

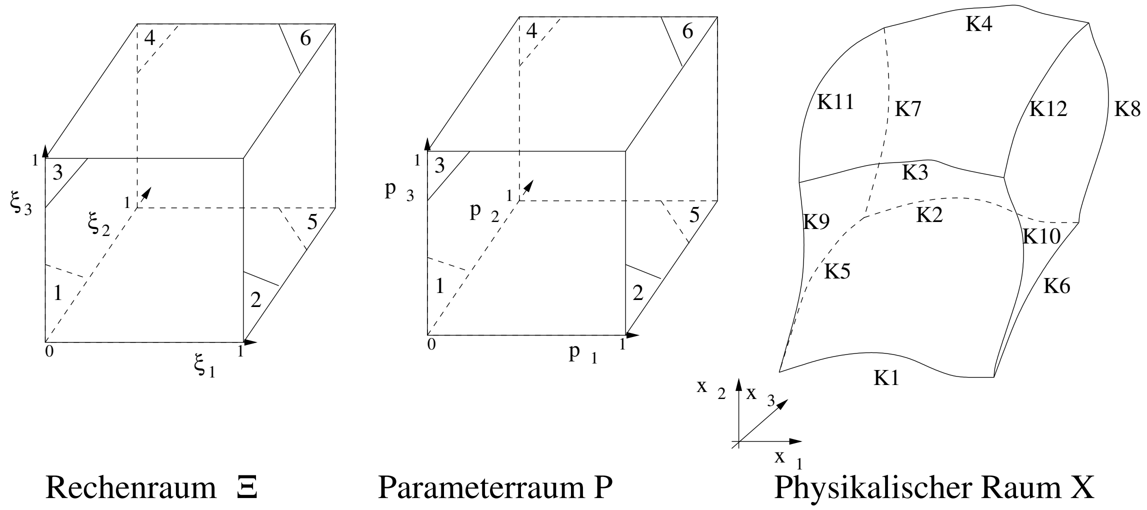

Abbildung 2.11: Bezeichnungen für die Ecken, Kanten und Seiten.

Die Koordinatenfunktionen müssen folgende Bedingungen erfüllen:

1.

- $p_1 = 0$ auf Seite 1 und $p_1 = 1$ auf Seite 2
- $p_2 = 0$ auf Seite 3 und $p_2 = 1$ auf Seite 4
- $p_3 = 0$ auf Seite 5 und $p_3 = 1$ auf Seite 6,

damit die zusammengesetzte Abbildung $\tilde{\xi}^{-1} \circ p$ die Randbedingungen $\tilde{\xi}^{-1} \circ p|_{\partial\Xi} = \partial X$ erfüllen kann.

2.

- p_1 ist die normalisierte Bogenlänge an den Kanten K1, K2, K3, K4

- p_2 ist die normalisierte Bogenlänge an den Kanten K5, K6, K7, K8
- p_3 ist die normalisierte Bogenlänge an den Kanten K9, K10, K11, K12.

Die normalisierte Bogenlänge bezieht sich hier auf die Bogenlänge der Kanten von X als Kurven betrachtet. Durch diese Bedingung wird das Gitter in P so verzerrt, dass das Gitter der Randpunkteverteilung des Gitters in X entspricht.
Aus den ersten drei Bedingungen ergibt sich

$$\begin{aligned} p_1(0,\xi_2,\xi_3)=0 \;\text{ und }\; p_1(1,\xi_2,\xi_3) &= 1 \\ p_2(\xi_1,0,\xi_3)=0 \;\text{ und }\; p_2(\xi_1,1,\xi_3) &= 1 \\ p_3(\xi_1,\xi_2,0)=0 \;\text{ und }\; p_3(\xi_1,\xi_2,1) &= 1. \end{aligned}$$

$x|_{\partial\Xi} : \partial\Xi \longrightarrow \partial X$ ist bijektiv aufgrund der Randbedingungen an x. Dadurch ist jedem Punkt in $\partial\Xi$ ein eindeutiger Punkt in ∂X zugeordnet. Zusammen mit den weiteren drei Bedingungen ist dann jedem Punkt auf den Kanten von X ein eindeutiger Punkt auf den Kanten von P zugeordnet. Damit kann man nun die Koordinatenfunktionen p_i eindeutig auf den Kanten definieren:

$$\begin{aligned} p_1(\xi_1,0,0) &= p_{1K_1}(\xi_1),\; p_1(\xi_1,1,0)=p_{1K_2}(\xi_1) \\ p_1(\xi_1,0,1) &= p_{1K_3}(\xi_1),\; p_1(\xi_1,1,1)=p_{1K_4}(\xi_1) \\ p_2(0,\xi_2,0) &= p_{2K_5}(\xi_2),\; p_2(1,\xi_1,0)=p_{2K_6}(\xi_2) \\ p_2(0,\xi_2,1) &= p_{2K_7}(\xi_2),\; p_2(1,\xi_2,1)=p_{2K_8}(\xi_2) \\ p_3(0,0,\xi_3) &= p_{3K_9}(\xi_3),\; p_3(1,0,\xi_3)=p_{3K_{10}}(\xi_3) \\ p_3(0,1,\xi_3) &= p_{3K_{11}}(\xi_3),\; p_3(1,1,\xi_3)=p_{3K_{12}}(\xi_1) \end{aligned}$$

Die zwölf Kantenfunktionen $p_{1K_1},...,p_{3_K12}$ bilden das Intervall $[0,1]$ auf die normalisierte Bogenlänge der jeweiligen Kante ab. Sie sind monoton steigend und durch die Randpunkteverteilung an den zwölf Kanten von X definiert. Die Abbildung $p:\overline{\Xi}\longrightarrow\overline{P}$ wird nun durch bilineare Interpolation zwischen den Kanten definiert:

$$\begin{aligned} p_1 = \; & p_{1K_1}(\xi_1)(1-p_2)(1-p_3)+p_{1K_2}(\xi_1)p_2(1-p_3)+ \\ & p_{1K_3}(\xi_1)(1-p_2)p_3+p_{1K_4}(\xi_1)p_2p_3 \end{aligned} \tag{2.19}$$

$$\begin{aligned} p_2 = \; & p_{2K_5}(\xi_2)(1-p_1)(1-p_3)+p_{2K_6}(\xi_2)p_1(1-p_3)+ \\ & p_{2K_7}(\xi_2)(1-p_1)p_3+p_{2K_8}(\xi_2)p_1p_3 \end{aligned} \tag{2.20}$$

$$\begin{aligned} p_3 = \; & p_{3K_9}(\xi_3)(1-p_1)(1-p_2)+p_{3K_{10}}(\xi_3)p_1(1-p_2)+ \\ & p_{3K_{11}}(\xi_3)(1-p_1)p_2+p_{3K_{12}}(\xi_3)p_1p_2 \end{aligned} \tag{2.21}$$

Zur Berechnung der $p_i,\ i = 1,2,3$ im Inneren des Gebietes muss in jedem Gitterpunkt das Gleichungssystem (2.19) – (2.21) gelöst werden. In der vorliegenden Implementierung wird hierzu das Newton-Verfahren benutzt. Siehe zum Newton-Verfahren auch in Kapitel 3.3.1. Gleichung (2.19) bedeutet, dass eine Ebene in Ξ mit konstantem ξ_1 auf eine bilineare Fläche in P abgebildet wird. $p_1(\xi_1)$ ist bilinear in p_1 und p_2. Auf dieselbe Art und Weise werden durch (2.20) und (2.21) Flächen mit konstantem ξ_2 und ξ_3 auf bilineare Flächen in P abgebildet. So wird für einen gegebenen Punkt in Ξ der entsprechende Punkt im Parameterraum (p_1, p_2, p_3) als Schnittpunkt der drei bilinearen Flächen berechnet. Für die Bijektivität der Abbildung p bleibt die Eindeutigkeit zu zeigen.

Lemma 1 *Die durch das Gleichungssystem* (2.19) – (2.21) *gegebene Abbildung* $p:\ \overline{\Xi}\ \longrightarrow\ \overline{P}$ *ist bijektiv.*

Beweis:
Zeige dazu, dass zwei Flächen $p_1(\xi_1)$ und $p_1(\xi_1')$ mit $\xi_1 \neq \xi_1'$ sich nicht schneiden. Dieselbe Argumentation gilt auch für p_2 und p_3. Damit erhält man, dass der Schnittpunkt der drei bilinearen Flächen und damit die Abbildung p eindeutig ist.
Sei $p_1(\xi_1) = p_1(\xi_1')$ mit $\xi_1 \neq \xi_1'$, dann gilt

$$\begin{aligned}
& p_{1K_1}(\xi_1)(1-p_2)(1-p_3) + p_{1K_2}(\xi_1)p_2(1-p_3) \\
&\qquad + p_{1K_3}(\xi_1)(1-p_2)p_3 + p_{1K_4}(\xi_1)p_2p_3 \\
= {} & p_{1K_1}(\xi_1')(1-p_2)(1-p_3) + p_{1K_2}(\xi_1')p_2(1-p_3) \\
&\qquad + p_{1K_3}(\xi_1')(1-p_2)p_3 + p_{1K_4}(\xi_1')p_2p_3\ ,
\end{aligned}$$

also

$$\begin{aligned}
& [p_{1K_1}(\xi_1) - p_{1K_1}(\xi_1')](1-p_2)(1-p_3) + [p_{1K_2}(\xi_1) - p_{1K_2}(\xi_1')]\, p_2(1-p_3) \\
+ & [p_{1K_3}(\xi_1) - p_{1K_3}(\xi_1')](1-p_2)p_3 + [p_{1K_4}(\xi_1) - p_{1K_4}(\xi_1')]\, p_2p_3 = 0.
\end{aligned} \tag{2.22}$$

Die Kantenfunktionen $p_{iK_j}\ i = 1,2,3\ j = 1,2,3,4$ sind monoton steigend. Daher haben die Terme in den eckigen Klammern alle dasselbe Vorzeichen. Die anderen Faktoren der Summanden sind alle größer oder gleich null und können nicht gleichzeitig verschwinden. Somit kann die Summe (2.22) nicht verschwinden, was ein Widerspruch ist . □

2.5 NURBS

In CAD-Systemen benötigt man zur Darstellung und Modellierung der Geometrie parametrisierte Kurven und Flächen. NURBS (**N**on **U**niform **R**ational **B**-**S**plines) sind der de-facto-Standard zur Darstellung parametrisierter Kurven und Flächen und werden auch von dem hier benutzten CAD-Ausgabeformat IGES unterstützt. Des Weiteren gibt es Bibliotheken, wie die in dieser Arbeit benutzte Nurbs++-Bibliothek [19], in denen die wichtigsten Manipulationen von NURBS zur Verfügung gestellt werden. So ist es naheliegend, auch zur internen Darstellung der Geometrie im Gittergenerator NURBS zu verwenden. Die Gründe für die weite Verbreitung von NURBS sind die folgenden:

- hohe Flexibilität, es ist möglich alle Arten von Flächen darzustellen, insbesondere Niveaulinien bestimmter Funktionen, wie zum Beispiel Kugeln, die in anderen Formaten nicht darstellbar sind
- intuitive Handhabung bei Manipulationen
- NURBS-Algorithmen sind schnell und numerisch stabil
- NURBS können effizient gespeichert werden
- Invarianz unter affinen Transformationen

Eine ausführliche Beschreibung findet man in [29]. NURBS sind eine Verallgemeinerung von B-Splines, die wiederum eine Verallgemeinerung von Bézier-Kurven sind.

Bézier-Kurven
Eine Bézier-Kurve vom Grad n ist gegeben durch

$$C(u) = \sum_{i=0}^{n} B_{i,n}(u) P_i, \quad a \leq u \leq b.$$

Dabei sind $P_i \in \mathbb{R}^d, d = 1, 2, 3$ und $B_{i,n}$ das Bernstein-Polynom n-ten Grades [3]. Das Bernstein-Polynom ist gegeben durch

$$B_{i,n}(u) = \frac{n!(u-a)^i(b-u)^{n-i}}{i!(n-i)!(b-a)^n}.$$

Die Punkte P_i sind die sogenannten *Kontrollpunkte*. Sie bilden das *Kontrollpolygon*, das den Verlauf der Kurve wesentlich bestimmt, wie man es der Abbildung 2.12 entnehmen kann. Für Kontrollpolygone mit vielen Kontrollpunkten, also komplexeren Kurven, wird es unpraktisch, die Kurve durch ein einziges Bézier-Polynom darzustellen, da der Polynomgrad groß wird, was zu numerischen Instabilitäten führt. Außerdem wird die Modellierung der Kurve durch Änderung einzelner Kontrollpunkte schwieriger, da lokale Änderungen der Kontrollpunkte sich auf den globalen Verlauf der Kurve auswirken. Eine Lösung für dieses Problem bieten die *B-Splines*.

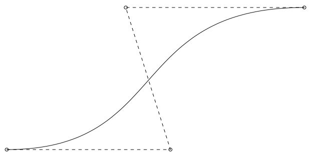

Abbildung 2.12: Bézierkurve vom Grad 3 als parametrisierte Kurve in $\mathbb{R}^2$ und Kontrollpolygon (gestrichelte Linie).

B-Splines

Eine B-Spline-Kurve ist aus mehreren Bézier-Kurven an sogenannten *Knoten* zusammengesetzt. An den Knoten werden zusätzlich Bedingungen an die Glattheit gestellt, so dass man insgesamt eine glatte Kurve erhält. Die Knoten sind diejenigen Werte des Parameters u, an denen die Bézier-Segmente aneinanderstoßen. Sie bilden eine nicht absteigende Folge reeller Zahlen $u_0, ...u_m$, für die gilt $a \leq u_i \leq u_{i+1} \leq b$. Dabei sind a, beziehungsweise b der minimale, beziehungsweise maximale Parameterwert der Kurve. Der B-Spline vom Grad k ist nun definiert durch

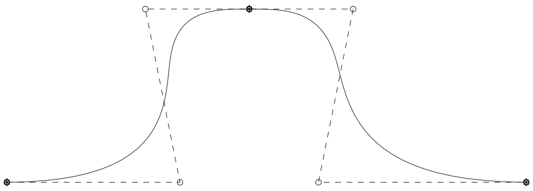

Abbildung 2.13: B-Spline-Kurve vom Grad 3 in $\mathbb{R}^2$ mit Kontrollpolygon und Knoten (fettgedruckte Punkte).

$$C(u) = \sum_{i=0}^{n} N_{i,k}(u) P_i \quad a \leq u \leq b.$$

Dabei sind $N_{i,k}$ die B-Spline-Basisfunktionen vom Grad k. Sie bilden, wie der Name schon sagt, eine Basis des Raumes der B-Splines vom Grad k auf den Knoten $u_0, \ldots, u_m$. Des Weiteren haben die $N_{i,k}$ folgende Eigenschaften

- $suppN_{i,k} \subset [u_i, u_{i+k}]$ (lokaler Träger)
- $N_{i,k}(u) \geq 0$ für alle $u \in \mathbb{R}$ (nicht negativ)
- $\sum_{i=1}^{n} N_{i,k}(u) = 1$ für $u \in [a, b]$ (Zerlegung der Eins)
- $N_{i,k}$ ist ein stückweises Polynom von Grad $\leq k - 1$, bezüglich der Intervalle $[u_i, u_{i+k}]$

Es gibt verschiedene Möglichkeiten, die $N_{i,k}$ zu berechnen, unter anderem Dividierte Differenzen und rekursive Berechnung, wie sie in [6, Kap. 7.4] beschrieben sind. Für die numerische Berechnung eignet sich am besten die rekursive Berechnung der $N_{i,k}$:

$$N_{i,0} = \begin{cases} 1 & \text{für } u_i \leq u \leq u_{i+1}, \\ 0 & \text{sonst.} \end{cases} \tag{2.23}$$

$$N_{i,k} = \frac{u - u_i}{u_{i+k} - u_i} N_{i,k-1}(u) + \frac{u_{i+k+1} - u}{u_{i+k+1} - u_{i+1}} N_{i+1,k-1}(u).$$

Für diese Berechnung wird der Knotenvektor um die Randknoten erweitert, die hier $k+1$-fach eingetragen werden. Man erhält den *nicht uniformen Knotenvektor* U:

$$U = \underbrace{a, ..., a}_{k+1}, u_{k+1}, ... u_{m-k+1}, \underbrace{b, ..., b}_{k+1}.$$

Dabei ist $m = n + k + 1$. Hierbei kann es in der Formel (2.23) zu einer Division durch null kommen, deren Ergebnis null gesetzt wird.
Detailliertere Aussagen über Bézierkurven und B-Splines sind in [6, Kap. 7.3 und 7.4] zu finden. Mit B-Splines ist es möglich, viele Punkte durch eine möglichst glatte Kurve zu interpolieren. Niveaulinien bestimmter Funktionen, wie zum Beispiel Kreise oder Ellipsen, können mit B-Splines jedoch nicht exakt dargestellt werden. Hierzu benötigt man rationale Funktionen.

NURBS
NURBS sind definiert durch

$$C(u) = \frac{\sum_{i=0}^{n} N_{i,k}(u) w_i P_i}{\sum_{i=0}^{n} N_{i,k}(u) w_i} \quad a \leq u \leq b.$$

Dabei sind P_i die Kontrollpunkte, $N_{i,k}$ die auf dem nicht uniformen Knotenvektor definierten B-Spline Basisfunktionen und w_i Gewichte, die den Einfluss der einzelnen Knoten auf den Kurvenverlauf steuern. Kleinere Werte von w_i ziehen die Kurve näher an P_i und größere Werte bewegen die Kurve weg von P_i. Sind alle Gewichte gleich eins, so stimmt die NURBS-Kurve mit einem B-Spline überein. Mit

$$R_{i,k}(u) = \frac{N_{i,k}(u) w_i}{\sum_{j=0}^{n} N_{j,k}(u) w_j}$$

erhält man die häufig benutzte Darstellung

$$C(u) = \sum_{i=0}^{n} R_{i,k}(u) P_i. \tag{2.24}$$

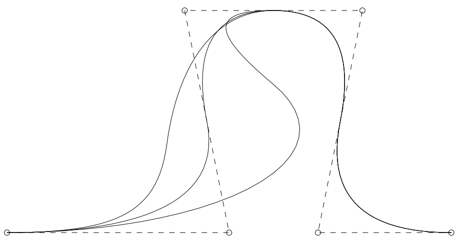

Abbildung 2.14: NURBS-Kurve mit $w_1 = 0.5, 1, 1.5$.

Die Flächendarstellung von NURBS lautet

$$C(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,k}(u)N_{j,l}(v)w_{ij}P_{ij}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,k}(u)N_{j,l}(v)w_{ij}} \quad a \leq u \leq b,\ c \leq v \leq d.$$

Dabei sind die Kontrollpunkte zu einem bidirektionalen *Kontrollnetz* P_{ij} zusammengefasst. Die $N_{i,k}(u)$ und $N_{j,l}(v)$ sind auf den Knotenvektoren

$$U = \underbrace{a, ..., a}_{k+1}, u_{k+1}, ...u_{r-k+1}, \underbrace{b, ..., b}_{k+1}$$
$$V = \underbrace{c, ..., c}_{l+1}, v_{l+1}, ...v_{s-l+1}, \underbrace{d, ..., d}_{l+1}$$

definiert, wobei $r = n + k + 1, s = m + l + 1$. Üblicherweise werden die Grenzen $[a, b]$ und $[c, d]$ auf $[0, 1]$ gesetzt. Mit

$$R_{i,j}(u,v) = \frac{N_{i,k}(u)N_{j,l}(v)w_{ij}}{\sum_{r=0}^{n}\sum_{s=0}^{m} N_{r,k}(u)N_{s,l}(v)w_{rs}}$$

erhält man wieder

$$C(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} R_{i,j}(u,v)P_{ij}.$$

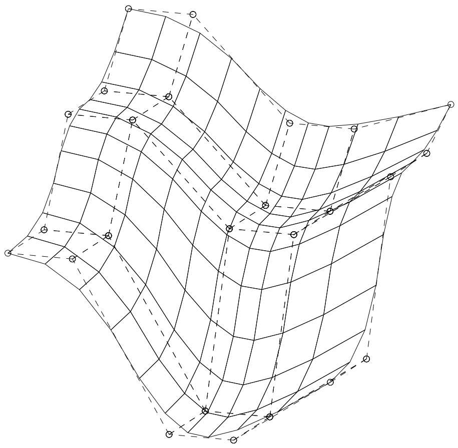

Abbildung 2.15: NURBS-Fläche mit Kontrollnetz.

2.6 IGES — Ein Standard zum Austausch von CAD-Daten

IGES ("**I**nitial **G**raphics **E**xchange **S**pecification") ist ein standardisiertes Dateiformat zum Austausch von CAD-Daten. Es zählt zu den weitverbreitetsten und wird auch von dem für diese Arbeit benutztem CAD-Programm Pro/Engineer [31] unterstützt. Mit IGES ist es möglich, alle Elemente der Informationen, die von heutigen CAD-Systemen bereitgestellt werden, darzustellen. Dies beinhaltet neben der Beschreibung der physikalischen Form (Geometrie- und Topologiedaten) auch Informationen über die Funktionsweise des Objektes, Daten für die Produktion und Ähnliches. Für diese Arbeit ist lediglich die Beschreibung der physikalischen Form relevant. Hier ist es unter anderem möglich, die Geometrie mittels NURBS-Flächen darzustellen. Diese Eigenschaft wird von Pro/Engineer sowie auch von vielen anderen CAD-Programmen unterstützt und in der vorliegenden Arbeit benutzt.

2.6.1 Entities

Die Daten werden innerhalb von IGES durch *Entities* dargestellt. Entsprechend der verschiedenen Arten von Informationen, die ausgetauscht werden, gibt es unterschiedliche Arten von Entities, die diese Informationen enthalten. So gibt es zum Beispiel Entities für Geometriedaten, zusätzliche Objektbeschreibung und Präsentationsformen. Eine Entity entspricht dann einem allen CAD Systemen gemeinsamen Konzept, welches unabhängig von der internen Darstellung ist. Für Geometriedaten gibt es zum Beispiel Entities, um jeweils Punkte, Linien, NURBS-Kurven oder Flächen zu beschreiben. Die Entities sind teilweise hierarchisch aufgebaut. Im Folgenden wird ein kurzer Überblick über die benutzen Entities und deren Hierarchie gegeben. Eine genaue Auflistung aller IGES-Entities, sowie eine detaillierte Beschreibung des IGES-Standards findet man in [32].

2.6.2 Benutzte IGES Entities

Die benutzten Entities sind in Abbildung 2.16 dargestellt. Entity 144 ist eine Fläche mit Rand, entsprechend verweist sie auf eine Entity, die eine Fläche enthält, und auf eine Entity, die den Rand der Fläche darstellt, nämlich eine Kurve auf einer parametrisierten Fläche. Hat die Fläche (Entity 144) ein oder mehrere Löcher, so wird auf mehrere (geschlossene) Kurven verwiesen, entsprechend der Anzahl der Ränder. Die geschlossene Kurve ist in vielen Fällen eine zusammengesetzte Kurve.

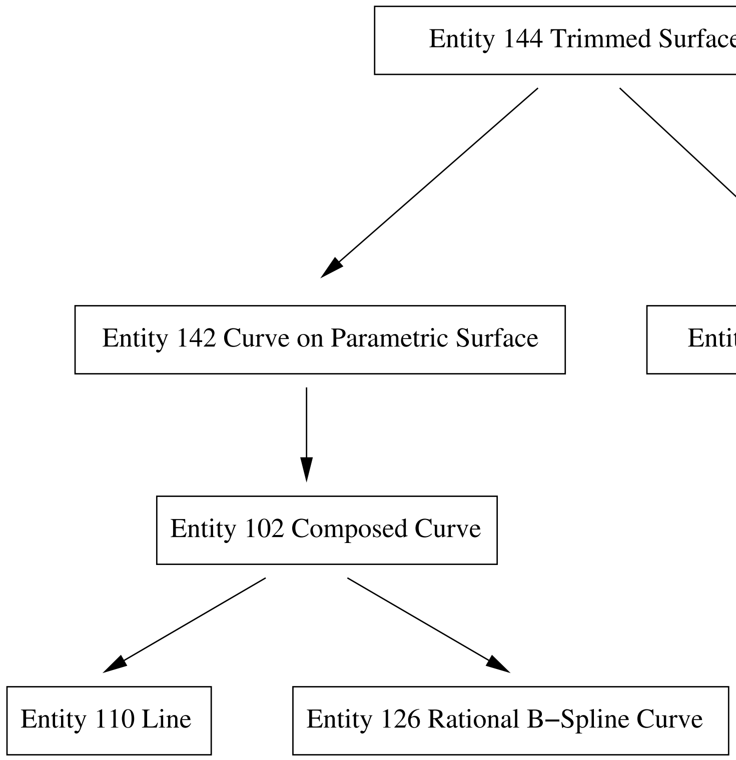

Abbildung 2.16: Hierarchie von IGES-Entities.

In solch einem Fall verweist sie auf eine Entity für zusammengesetzte Kurven, die wiederum auf mehrere Kurven (Linien oder NURBS) verweist. Oft reichen diese Entities aus, um die Geometrie komplett zu beschreiben.

2.6.3 Dateistruktur

Eine IGES (Ascii) Datei (vergleiche Abbildung 2.17) besteht aus fünf Sektionen: Start-, Global-, Directory-, Parameter- und End-Sektion, die durch die entsprechenden Buchstaben S, G, D, P und T in Spalte 72 der jeweiligen Zeile der zugehörigen Sektion gekennzeichnet sind. Der letzte Eintrag jeder Zeile ist die Zeilennummer innerhalb der Sektion (beginnend mit 1).

Start-Sektion
Hier ist ein beliebiger Text enthalten, der dazu genutzt werden kann, die Datei zu beschreiben.
Global-Sektion
Dieser Abschnitt gibt Auskunft darüber, mit welchem System IGES Datei erzeugt

```
Dateiname: bsp.igs                                                      S      1
Dies ist ein Beispiel, bestehend aus einer Linie (Entity 110)           S      2
1H,,1H;,9HCHRISTIAN,25HH:/w/diplom/bsp.igs,                             G      1
49HPro/ENGINEER by Parametric Technology Corporation,7H2001200,32,38,7, G      2
38,15,9HCHRISTIAN,1.,1,4HINCH,32768,0.5,13H021105.140358,0.0259451,     G      3
259.462,7Hmueller,7HUnknown,10,0,13H021105.140358;                      G      4
     110       1       1       1       0       0       0       001010000D      1
     110       0       0       2       0                    LINE       1D      2
110,-1.083343116898D2,0D0,-5.819136671545D1,-1.083343116898D2,         1P      1
0D0,4.537350986851D1;                                                  1P      2
S      2G     4D      4P       2                                        T      1
```

Abbildung 2.17: IGES-Datei.

wurde und welche Einheiten benutzt wurden. Außerdem beinhaltet er auch die benutzte Darstellungsgenauigkeit und noch einige andere allgemeine Informationen.

Directory-Sektion

Die Directory-Sektion liefert einen Index der Entities, aus denen das beschriebene Objekt besteht. Jede Entity belegt zwei Zeilen. Der erste Eintrag jeder Zeile sagt, um welche Art von Entity es sich handelt (normalerweise eine ganze Zahl zwischen 0 und 599). Der zweite Eintrag der ersten Zeile enthält die Zeilennummer der Entity in der Parameter-Sektion (siehe Parameter-Sektion). Die restlichen Einträge beinhalten allgemeine Attribute der Entity, wie zum Beispiel Farben oder Verweise auf Entities mit Kommentaren.

Parameter-Sektion

Hier befinden sich die Daten, die die Entity konkret definieren. Dies sind zum Beispiel die Koordinaten bei Punkten, Anfangs- und Endpunkte von Linien und so weiter. Bei zusammengesetzten Entities können hier auch Verweise auf die enthaltenen Entities stehen. Ein solcher Verweis besteht aus der Zeilennummer der entsprechenden Entity in der Directory-Sektion. Der drittletzte Eintrag einer Zeile (vor dem P) ist ein Verweis auf den eigenen Directory-Eintrag.

Terminate-Sektion

Dies ist die letzte Zeile der Datei. Sie besteht aus den Buchstaben S, G, D, P und T für die entsprechenden Sektionen, gefolgt von der Anzahl der Zeilen in jeder Sektion.

Kapitel 3

Praktische Umsetzung

In Kapitel 2 wurde im Detail die Generierung strukturierter Gitter, aus denen sich ein blockstrukturiertes Gitter zusammensetzt, behandelt. Abschnitt 2.3 widmete sich den algebraischen Methoden und in Abschnitt 2.4 wurden Methoden vorgestellt, die Gitter mittels Lösungen elliptischer Differentialgleichungen generieren. Schließlich sind in Abschnitt 2.5 die parametrisierten Kurven und Flächen eingeführt worden, die benutzt werden, um den Rand des zu vergitternden Gebiets zu beschreiben. Die Beschreibung des benutzten Dateiformats der CAD-Daten wurde in Kapitel 2.6 geliefert.
Der folgende Teil dieser Arbeit beschäftigt sich mit der Erzeugung der Blöcke aus den Geometriedaten sowie der Lösung der bei der Gittergenerierung mittels Differentialgeleichungen entstehenden nichtlinearen Gleichungssysteme.

3.1 Generierung der Blöcke

Die Generierung der Blöcke ist ein wesentlicher Bestandteil blockstrukturierter Gittergenerierung. Es ist der bis jetzt am wenigsten automatisierte Teil der Gittergenerierung und erfordert viel menschliche Arbeit. In der Regel stellt dieser Teil sogar den größten Anteil an menschlicher Arbeit bei der Gittergenerierung, wie Vatsa, Sanetrik und Parlette in einem Überblick über blockstrukturierte Gittergenerierer [36] herausgestellt haben. Die Blockgenerierung besteht aus drei Schritten, die nachfolgend beschrieben werden.

1. Rekonstruktion der Geometrie aus den CAD-Daten

2. Festlegen der Blocktopologie

3. Umsetzung der vorgegebenen Blocktopologie mit der rekonstruierten Geometrie

3.1.1 Rekonstruktion der Geometrie

Die Rekonstruktion der Geometrie zerfällt in drei Schritte

1. Einlesen der IGES-Datei

2. Generierung der NURBS-Flächen

3. Zusammenfügen der Flächen zu einer gut zu verwaltenden Geometrie-Datenstruktur, so dass die Geometrie am Ende in Form einer oder mehrerer Instanzen einer Block-Klasse vorhanden ist.

Nach dem Einlesen der IGES-Datei stehen die IGES-Entities (siehe Kapitel 2.6.1) zur Verfügung. Aus diesen werden nun anhand ihrer Parameter (Koordinaten der Punkte und NURBS Parameter) die Flächen rekonstruiert, aus denen sich die Oberfläche des zu rekonstruierenden Objekts zusammensetzt.
Die Generierung der Flächen geschieht nach der in Kapitel 2.6.2 beschriebenen Hierarchie: Suche zuerst die Entities 144, die die berandeten Flächen darstellen und füge dann schrittweise die untergeordneten Entities hinzu. Die Entities werden im Programm durch entsprechende Klassen dargestellt:

Entity 144 Bounded Surface	$\widehat{=}$	`Seite`
Entity 142 Curve on Parametric Surface +		
Entity 102 Composed Curve	$\widehat{=}$	`Rand`
Entity 110 Line bzw. Entity 126 Rational B-Spline	$\widehat{=}$	`Kurve`

Abbildung 3.0: IGES-Entities und die entsprechenden Klassen.

Die Geometrie wird dann als Instanz der Klasse `Block` abgespeichert. Diese Block ist nicht zu verwechseln mit den später erzeugten `Block`-Instanzen, die zußatlich noch Gitterdaten enthalten.
Ein `Block` hat also einen Vektor sechs zusammenhängender `Seiten`, i. e. `Seiten`, deren Flächen zusammen den Rand eines `Block`s darstellen. Eine `Seite` wiederum besteht aus einer NURBS-Fläche und einem `Rand`, der sich aus zwei oder vier `Kurven` zusammensetzt.
Um die Geometrie als `Block` abspeichern zu können, muss die Oberfläche der Geometrie auf sechs paarweise gegenüberliegende Flächen verteilt werden. Im Allgemeinen liegt die Oberfläche der Geometrie jedoch nicht in Form von sechs gegenüberliegenden Flächenstücken vor. Die Oberfläche muss also zerteilt werden, und möglicherweise werden die zerteilten Flächenstücke wieder zu neuen Flächen vereinigt, so dass man sie am Ende als `Seiten` eines `Blocks`, dessen Oberfläche mit dem Geometrierand übereinstimmt, abspeichern kann. Die einzelnen Bestandteile des `Block`s sind

so orientiert und sortiert, dass sie leicht mit Bestandteilen anderer Blöcke zu neuen Blöcken kombiniert werden können. Dafür muss bei jedem Objekt genau die absolute Orientierung im Raum sowie die relative Orientierung zum übergeordnetem Objekt definiert sein. Bei einer `Kurve` heißt das , dass bekannt sein muss, welche die vorherrschende Richtung im Raum ist (x_1, x_2 oder x_3) und wie die Position innerhalb des `Rand`es ist (links, rechts, oben oder unten). Ausgehend von den absoluten Orientierungen der `Kurve`n wird dann die absolute Orientierung des `Rand`es bestimmt, über die dann wiederum die Orientierung der entsprechenden `Seite` gegeben ist. Die relativen Orientierungen innerhalb eines übergeordneten Objekts werden durch Vergleiche der Subobjekte bestimmt.

3.1.2 Blocktopologie

Unter den drei erwähnten Schritten bei der Blockgenerierung ist sicherlich das Festlegen der Blocktopologie derjenige, der am schwierigsten zu automatisieren ist, und daher viel Handarbeit beinhaltet. Die Blocktopologie gibt sowohl die Anzahl der Blöcke vor, als auch deren Verbund im blockstrukturierten Gitter. Sie bestimmt letztendlich das gesamte Design des Gitters. Um ein strukturiertes Gitter zu generieren, benötigt man immer sechs paarweise logisch gegenüberliegende Seiten, deren Vereinigung den Rand des zu vergitternden Gebiets beziehungsweise Blocks ergibt. Zwischen den gegenüberliegenden Seiten verlaufen die Gitterlinien. Oft ist nicht von vornherein klar, welche Teile des Randes des zu vergitternden Gebietes zu welchen Seiten gehören sollen. Das einfache Beispiel einer einspringenden Ecke zeigt, dass es hier viele Möglichkeiten gibt, die zu unterschiedlichen Ergebnissen führen.

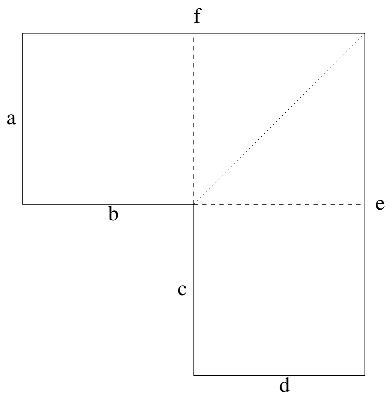

Abbildung 3.1: Gebiet mit einer einspringenden Ecke.

Die erste Frage ist, ob man das Gebiet in mehrere Blöcke aufteilen möchte oder nicht. Möchte man dies nicht tun, bleibt zu entscheiden, welche Randstücke des Gebiets gegenüberliegende Seiten bilden sollen. Man kann zum Beispiel die Seiten b und c, sowie e und f jeweils zu einer Seite zusammenfassen, so dass a und d sowie die beiden

zusammengefassten Seiten gegenüberliegen. Genauso ist es möglich b, c und d zusammenzufassen, um a und e gegenüberliegend zu haben. Ähnlich viele Möglichkeiten gibt es bei der Unterteilung in mehrere Blöcke. Man kann das Gebiet entlang der gepunkteten Linie in zwei Blöcke unterteilen oder entlang der gestrichelten Linien in drei Blöcke oder entlang nur einer der gestrichelten Linien in zwei unterschiedlich große Blöcke. Wann welche Wahl für den Verlauf der Gitterlinien richtig ist, hängt vom zu berechnenden Problem und der verwendeten Diskretisierung ab. Bei den hier ausschließlich betrachteten Strömungsproblemen und der von NSCL verwendeten Diskretisierung ist es von Vorteil, wenn die Gitterlinien parallel zur Strömungsrichtung verlaufen [23, Kap. 2.1]. So wird man also den Verlauf der Gitterlinien nach dem Strömungsverlauf ausrichten. Bestimmend für den Strömungsverlauf sind die Randbedingungen. Hat man keine Vorstellungen über den Verlauf der Gitterlinien, oder erwartet man Turbulenz, so wird man orthogonale Gitter wählen, da diese nicht an einen bestimmten Strömungsverlauf angepasst sind. Außerdem ist die von NSCL verwendete Diskretisierung hierauf konsistent von zweiter Ordnung, im Gegensatz zu einer Konsistenz von erster Ordnung bei allgemeinen kurvilinearen Gittern [23, Kap. 2.1.7].
Bei komplizierten Geometrien kann es schwierig werden, überhaupt Aufteilungen des Gebietes zu finden, die zu möglichen Blocktopologien führen. Eine Technik zur vollständigen Automatisierung dieses Prozesses für beliebige Geometrien gibt es nicht. Dannenhoffer [5] hat eine Technik vorgestellt, die eine gegebene Blocktopologie unter gewissen Gesichtspunkten optimiert, was den Prozess vereinfachen kann. In der vorliegenden Arbeit sind bestimmte Blocktopologien vorgegeben. In dem implementierten Programm kann der Nutzer unter verschiedenen Topologien auswählen, die fest implementiert sind. Diese Topologien sind sinnvoll für bestimmte Anwendungen, wie die Umströmung von Tragflächen, Säulen, Türmen oder ähnlichen Gegenständen. Aus diesen Blocktopologien werden dann automatisch die Blöcke generiert (siehe Kapitel 3.1.3).
Ein blockstrukturiertes Gitter ist immer aus Teilen mit Standardtopologien zusammengesetzt. Diese sind:

1. Gebietsteile ohne Loch, wie zum Beispiel das Gebiet mit einer einspringenden Ecke.

2. Gebietsteile mit Loch. Hier kann man grundsätzlich drei verschiedene Typen von Blocktopologien unterscheiden: H-, O- und C-Gitter.

Das H-Gitter hat eine kartesische Anordnung der Blöcke, das heißt, es werden nur ξ_1-Linien (Linien mit $\xi_2, \xi_3 \equiv$ konstant) mit ξ_1-Linien verbunden (für ξ_2, ξ_3 analog). Die Orientierung dieser Linien über die Blockgrenzen hinaus ändert sich nicht. Im Inneren dieses Gitters gibt es einen Schlitz. Das heißt, zwei aneinanderliegende Blöcke sind an den sich berührenden Seiten nicht verbunden, wodurch eine innere

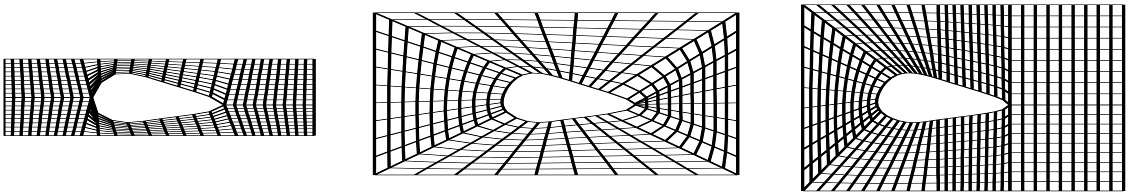

Abbildung 3.2: Querschnitte von H-, O- und C-Gittern um eine Tragfläche.

Gebietsgrenze entsteht. Diese Gebietsgrenze stellt das zu umströmende Objekt dar. Man benutzt diese Art von Gittern, um schmale Gegenstände wie Rotorblätter oder dünne Tragflächen zu umströmen.
Bei O-Gittern verlaufen die Koordinatenlinien einer Dimension kreisförmig. Es gibt also eine Klebestelle, an der in einer Koordinatenrichtung die Punkte mit maximalem Parameterwert des einen Blocks mit den Punkten mit minimalem Parameterwert des anderen Blocks identisch sind. In der Abbildung 3.2 sind zum Beispiel vier Blöcke zu einem O-Gitter verklebt. Solche Gitter finden Anwendung bei dicken, rundlichen Geometrien, wie sie manche Flugzeugrümpfe haben.
Eine Kombination aus H- und O-Gitter stellt das C-Gitter dar. Hier werden die Gitterlinien auf einer Seite, wie beim O-Gitter, um das Objekt herumgeführt, während das Objekt auf der anderen Seite, wie beim H-Gitter, "zwischen das Gitter geschoben" wird. Dadurch kommt es auf der Seite, wo das Objekt zwischen das Gitter geschoben wird, dazu, dass parallel verlaufende Gitterlinien aneinandergrenzender Blöcke entgegengesetzt orientiert sind.

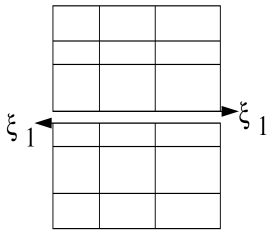

Abbildung 3.3: Zwei aneinandergrenzende Blöcke mit unterschiedlichen Orientierungen für ξ_1.

C-Gitter sind sinnvoll für Geometrien wie in der Abbildung 3.2, wo es auf einer Seite eine scharfe Kante und auf der anderen eine Rundung gibt.
Bei komplizierteren Geometrien werden oft Kombinationen dieser Basistopologien benutzt, um den einzelnen Details der Geometrie gerecht zu werden. Bei Flugzeugen werden beispielsweise häufig unterschiedliche Gitter für Rumpf, Tragflächen und Triebwerke benutzt.

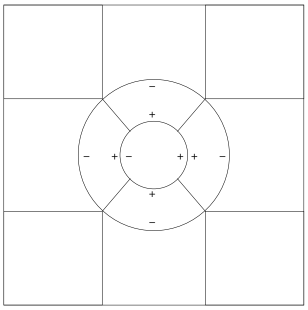

Abbildung 3.4: Die (hinteren) Blockgrenzen der vor dem Torus liegenden Blöcke bei Verwendung eines O-Gitters.

Einschränkungen

Das Lösen von Differentialgeichungen auf Gittern mit aufwändigen Blocktopologien erfordert einen gewissen Mehraufwand in der Verwaltung der Gitter. Für die Berechnungen an den Blockgrenzen müssen Daten zwischen den angrenzenden Blöcken ausgetauscht werden. Je nach Löser gibt es daher Einschränkungen der Möglichkeiten für Blocktopologien. Der hier verwendete Löser NSCL stellt folgende Bedingungen an das Gitter:

- Nur ganze Seiten dürfen miteinander verklebt werden
- Es dürfen nur ξ_1- mit ξ_1-, ξ_2- mit ξ_2-Linien etc. verklebt werden
- Die Orientierung der Gitterlinien aneinandergrenzender Blöcke muss gleich sein

Was ist mit diesen Einschränkungen möglich?

Durch die dritte Bedingung ist die Benutzung von C-Gittern grundsätzlich ausgeschlossen, da man hier immer aneinandergrenzende Blöcke mit unterschiedlich orientierten Gitterlinienen hat. H- und O-Gitter sind möglich, doch auch hier gibt es Einschränkungen, wie am folgenden Beispiel des Torus verdeutlicht wird.
Es soll also das einen Torus umgebende Gebiet so in Blöcke aufgeteilt werden, dass die Generierung eines blockstrukturierten Gitters ermöglicht wird. Betrachte zuerst die Variante mit einem O-Gitter. Es reicht aus, die 13 vor dem Torus liegenden Blöcke zu betrachten (siehe Abbildung 3.4). Von innen nach außen sind das: ein Block vor dem Loch in der Mitte, vier Blöcke vor dem Torus, welche zu einem O-Gitter verklebt werden, und acht kartesisch angeordnete Blöcke um das O-Gitter. Diese Blocktopologie ist unter den geforderten Bedingungen nicht möglich. Es reicht aus, nur das O-Gitter (die vier Blöcke vor dem Torus) und den innen liegenden Block zu betrachten. Untersucht man die innenliegende Grenze der Blöcke des O-Gitters,

so stellt man fest, dass dort, wegen der geforderten Gleichheit der Orientierung der Gitterlinien über die Blockgrenzen hinweg, immer die Punkte mit maximalen (minimalen) Parameterwerten einer Dimension liegen müssen. In Abbildung 3.4 ist dies durch die Pluszeichen angedeutet. Nun kann der Block in der Mitte nicht auf zwei gegenüberliegenden Seiten minimale Parameterwerte haben, da die Gitterabbildung dann nicht bijektiv wäre. So müssten also innen rechts zwei Plus-Seiten verklebt werden, was eine Änderung der Orientierung der Gitterlinien über die Blockgrenzen hinaus bedeuten würde. Außerdem müssten an zwei Seiten des inneren Blocks ξ_1- mit ξ_2-Linien verklebt werden, da die radialen Linien im O-Gitter entweder alle ξ_1- oder alle ξ_2-Linien sein müssen, während die horizontalen und vertikalen Linien im inneren Block Linien verschiedener Parameterdimensionen sind. Neben den Problemen mit dem Block in der Mitte gibt es hier auch noch ein Problem mit den Blöcken um das O-Gitter herum: Die vier Blöcke, die mit ganzen Seiten an das O-Gitter grenzen, übernehmen dessen Orientierung, was zur Folge hat, dass die beiden Blöcke oben und unten, relativ zu den beiden Blöcken rechts und links, um 90° gedreht sind. Diese Blöcke sind also unterschiedlich orientiert. Die Blöcke in den Ecken grenzen also an jeweils zwei unterschiedlich orientierte Blöcke, was aber nicht sein darf, da aneinandergrenzende Blöcke gleich orientiert sein müssen.
Untersuche jetzt eine Blocktopologie mit komplett kartesischer Anordnung der Blöcke. Auch hier ist es ausreichend, die vor dem Torus liegende Schicht zu betrachten. Diese besteht hier aus 25 Blöcken.

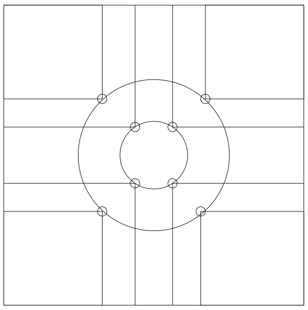

Abbildung 3.4: Die (hinteren) Blockgrenzen der 25 vor dem Torus liegenden Blöcke bei Verwendung eines kartesischen Gitters.

Zwar erfüllt diese Anordnung alle drei von NSCL geforderten Bedingungen, jedoch ist dieses Gitter aus anderen Gründen schlecht: An den gekennzeichneten Stellen haben Zellen des mittleren Blockes, sowie Zellen der den Ecken zugewandten Blöcken vor dem Torus, Winkel von nahezu 180°. Solche Winkel führen bei der Berechnung von finiten Differenzen zu numerischen Instabilitäten, da es vorkommen kann, dass Ableitungen nahezu orthogonal zu beiden (in 3D allen drei) Koordinatenlinien berechnet werden müssen.
Es ist also nicht, bzw. nur eingeschränkt möglich, mit NSCL einen Torus zu umströmen. Aus dem ersten Beispiel kann man zwei Einschränkungen von NSCL im

Umgang mit O-Gittern herleiten.
Erstens: O-Gitter sind nur möglich, wenn die inneren Seiten der Blöcke des O-Gitters nicht mit anderen Seiten verklebt werden müssen, sie also eine (innere) Grenze des Simulationsgebietes darstellen. (Ein O-Gitter um ein Objekt herum ist also möglich.)
Zweitens: Man kann einem O-Gitter nicht beliebig Blöcke anfügen, da es dann leicht zu Problemen mit der Orientierung der Blöcke kommen kann.
Kartesische Anordnungen von Blöcken, wozu auch H-Gitter zählen, sind von Natur aus konform mit den von NSCL geforderten Bedingungen. Im zweiten Beispiel wird jedoch deutlich, dass diese, für NSCL unproblematischen Geometrien, schnell an ihre Grenzen stoßen können. Nichtsdestoweniger zeigt das Beispiel auch, dass die Umströmung von Geometrien mit Loch grundsätzlich möglich ist, sofern die Geometrie die Generierung eines "guten" Gitters zulässt. Eine solche Geometrie wäre zum Beispiel ein Quader mit rechteckigem Loch.

3.1.3 Generierung von Blöcken

Nach der Rekonstruktion der Geometrie steht die Geometrie als Instanz der `Block`-Klasse zur Verfügung. Nun soll, je nach Blocktopologie, entweder der Geometrie-`Block` in kleinere Blöcke aufgeteilt werden oder neue `Block`-Objekte um den Geometrie-`Block` generiert werden. In beiden Fällen besteht die Generierung neuer Blöcke aus zwei Teilen:

1. Generierung neuer `Seiten`, insbesondere neuer Flächen

2. Reorganisation der neuen und alten `Seiten` zu neuen `Block`-Objekten

Die Generierung neuer `Seiten` geschieht durch Zerschneiden und Verschmelzen vorhandener `Seiten` sowie durch Interpolation vorgegebener Randkurven. Für die hierzu nötigen Manipulationen von NURBS-Flächen und Kurven wurde die Nurbs++ Bibliothek um die nötigen Funktionen erweitert. Das folgende Beispiel verdeutlicht den Ablauf der Blockgenerierung:
Die aus der IGES-Datei eingelesene Geometrie ist ein (deformierter) Würfel mit sechs `Seiten` und ist als `Block`-Objekt abgespeichert. Die Umgebung ist ebenfalls ein `Block` mit sechs `Seiten` und besteht aus einem größeren Würfel, der die Geometrie umgibt. Aus diesen beiden `Block`-Objekten sollen nun sechs `Block`-Objekte gemacht werden, die das Gebiet zwischen den beiden Ausgangsblöcken beinhalten sollen. Für einen neuen `Block` werden als Ausgangsseiten gegenüberliegende `Seiten` der beiden `Block`-Objekte, zum Beispiel die linke `Seite` der Geometrie und die linke `Seite` der Umgebung, genommen. Die verbleibenden `Seiten` werden nun durch Transfinite Interpolation zwischen den entsprechenden Randkurven der gegenüberliegenden `Seiten` generiert. Für die linken `Seiten` heißt das, die Oberseite des `Blocks`

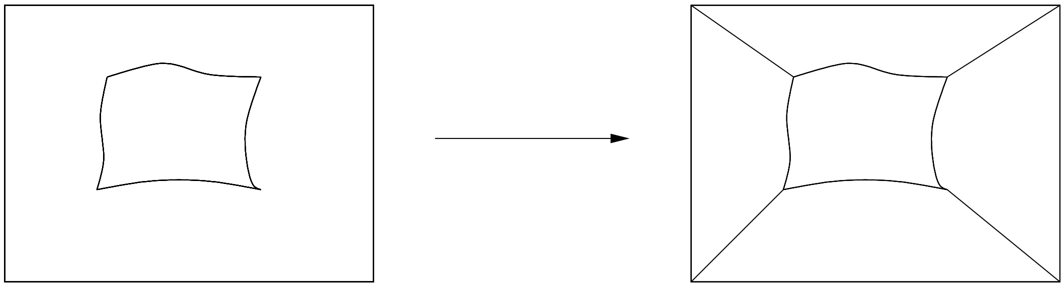

Abbildung 3.4: Generierung neuer Blöcke durch Einfügen von Seiten.
links: 2Blöcke. rechts: 4 Blöcke um die Geometrie.

entsteht durch Interpolation der Randkurven oben, die Unterseite durch Interpolation zwischen den Randkurven unten, und so weiter. Bei komplizierten Rändern kann die Transfinite Interpolation fehlschlagen, so dass man Flächen erhält, deren Ränder nicht mit den vorgegebenen Randkurven übereinstimmen. Dies stellt eine Einschränkung an die zu bearbeitenden Geometrien dar. Diese Einschränkung kann überwunden werden, indem man die neuen, durch Interpolation erzeugten `Seiten` mit den besprochenen Differentialgleichungsmethoden glättet, um so eine Übereinstimmung mit dem vorgegebenen Rand zu erreichen. Da diese Probleme in den betrachteten Fällen nicht auftreten, ist es ausreichend die Transfinite Interpolation zu benutzen.

3.1.4 Glättung an den Blockgrenzen

Aus der von NSCL verwendeten Diskretisierung der Kontinuitätsgleichung [23][Kap. 2.1.3] und den zugehörigen Interpolationsregeln [23][Kap. A.1.3] ergibt sich, dass die Diskretisierung umso besser ist, je weniger die Gitterlinien, im Verlauf von einer Zelle zur nächsten, geknickt werden. Innerhalb eines Blocks werden die Gitterlinien, wie in Kapitel 2.4 beschrieben, geglättet, so dass Knicke minimiert werden. An den Blockgrenzen kann es jedoch zu weniger glatten Übergängen kommen, da die Gitterpunkte dort (als Randbedingungen der Differentialgleichungen) vorgegeben sind, und somit hier nicht geglättet wird. Es gibt verschiedene Möglichkeiten, diese Stellen zu glätten. Man kann zum Beispiel mehrere Blöcke zusammenfassen und die Glättung mittels einer Schwarz-Iteration durchführen. Dadurch werden Blockgrenzen de facto abgeschafft und es gibt auch keine Unglattheiten mehr. Allerdings geht auch die Kontrolle über den Verlauf der Gitterlinien verloren, da man über die Randwerte an den Blockgrenzen Punkte vorgibt, durch die Gitterlinien laufen müssen. Eine andere Möglichkeit ist, die Blöcke sich überlappen zu lassen, so dass die eigentlichen Blockgrenzen im Innern der Blöcke liegen und mitgeglättet wer-

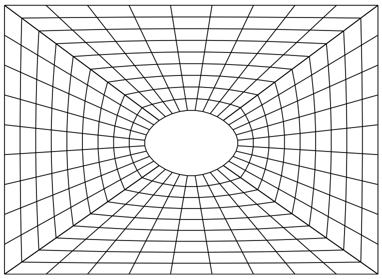

Abbildung 3.5: Schnitt durch ein O-Gitter mit vier Blöcken.

den. In den überlappenden Bereichen hat man dann verschiedene Lösungen bzw. Koordinatenwerte, insbesondere für die Randpunkte. Hier muss dann zwischen den verschiedenen Lösungen interpoliert werden, um eine eindeutige Lösung zu erhalten, die dann die Gitterpunkte am Rand beider angrenzender Blöcke darstellt. Diese Methoden machen die Lösung der Gleichungssysteme aufwändiger und erfordern im Fall blockweiser, paralleler Berechnung zusätzliche Kommunikation. Ein anderer Nachteil dieser Methoden ist, dass echte Randpunkte, also Randpunkte eines Blocks, die gleichzeitig auch Gebietsrand sind, nicht geglättet werden.

Eine einfache Methode, bei der diese Probleme nicht auftauchen, ist die folgende: Aufgrund der zugrundeliegenden Blocktopologie ist abzusehen, wo es zu stärkeren Knicken der Gitterlinien kommt. Im Beispiel des O-Gitters sind es die O-förmig verlaufenden Gitterlinien. An diesen im Vorhinein bekannten Stellen wird nun geglättet. Dazu werden in einem Block zusätzlich zu den eigenen Gitterpunkten noch jeweils eine Schicht der Gitterpunkte des jeweils angrenzenden Blocks gespeichert. Abhängig von der Blocktopologie werden jetzt bestimmte Koordinatenwerte der Randpunkte zwischen den angrenzenden inneren Punkten der jeweiligen Blöcke interpoliert. Im Falle des O-Gitters sind das die x-Werte (vorausgesetzt die x-Achse verläuft horizontal), in der Grafik unten sind es die z-Werte. Den geglätteten Randpunkt erhält man dann durch Projektion des Interpolierten Koordinatenwertes auf den Rand des Blocks.

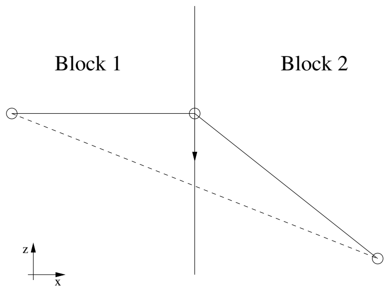

Abbildung 3.6: Glättung durch Interpolation zwischen den Blöcken.

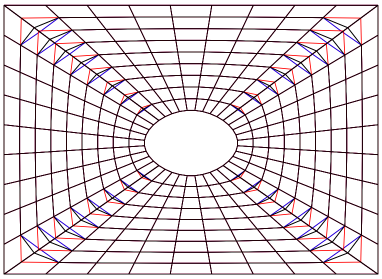

Abbildung 3.7: O-Gitter mit $\lambda_1 = 0, 0.7, 1$ (rot, schwarz, blau).

Dabei muss sichergestellt sein, dass die geglätteten Punkte nicht außerhalb der Blockgrenzen liegen. Zur Kontrolle werden Parameter λ_1, λ_2 und $\lambda_3 \in [0, 1]$ benutzt, die die Interpolation in x-, y- und z-Richtung regeln. $\lambda_i = 0$ bedeutet keine und $\lambda_i = 1$ bedeutet volle Interpolation ($i = 1, 2, 3$). Bei der Glättung des Randes ist Folgendes zu beachten. Es gibt Zellen, bei denen sich die Glättung negativ auf die Qualität der Diskretisierung, die NSCL verwendet, auswirkt. Die Konsistenzordung der Diskretisierung des Divergenzoperators in NSCL ist abhängig von der Form der Gitterzellen. Lediglich für parallelogrammartige Zellen (Zellen mit parallelen Seiten gleicher Länge) ist die Konsistenzordnung zwei, ansonsten ist die Diskretisierung konsistent erster Ordnung [23] Kap. A.1.4. Im Falle des O-Gitters werden durch die Glättung an den Blockgrenzen besonders die Zellen in den Ecken deformiert, da die Blockgrenzen im Innern geglättet werden, während die Gebietsgrenze unverändert bleibt. Einen negativen Einfluss auf die Lösung hat das jedoch nur bei grober Auflösung. Bei feiner Auflösung überwiegt der Anteil der positive der durch die Glättung positiv beeinflussten Zellen.

3.2 Aufbau des Programms

Aus einer im IGES-Format gegebenen Geometrie soll ein blockstrukturiertes Gitter generiert werden, welches dann von dem Programm NSCL zur Strömungsberechnung benutzt wird. Ein blockstrukturiertes Gitter wird innerhalb von NSCL durch die Klasse `GridCollectionNS` dargestellt. Der Gittergenerierer erzeugt also aus einer IGES-Datei eine Instanz der Klasse `GridCollectionNS`. Diese wird dann mittels der zur Verfügung stehenden Ausgaberoutinen in eine Datei geschrieben, welche von NSCL eingelesen werden kann. Alternativ kann auch direkt nach der Gittergenerierung die Strömungsberechnung gestartet werden. Außer dem Dateinamen für die IGES-Daten werden zur Gittergenerierung noch andere Daten benötigt. Diese, wie auch der Name der IGES-Datei, werden aus einer Datei namens "input" eingelesen, die im Anhang B.1 genauer beschrieben wird. Sie beinhaltet unter anderem die

Spezifikation der Blocktopologie (siehe auch Kapitel 3.1) sowie Gitter- und NSCL-Parameter für die Strömungsberechnung. Über die Gitterparameter können die Anzahl der Gitterpunkte in jeder Richtung sowie die Gradierung des Gitters festgelegt werden. Gradierte Gitter konzentrieren Gitterlinien an bestimmten Stellen des Gitters, um dort eine höhere Auflösung zu erreichen.
Die Erzeugung der Instanz von `GridCollectionNS` geschieht schrittweise. Im Folgenden werden die einzelnen Schritte erläutert und anschließend die Datenstrukturen beschrieben. Genauere Beschreibungen der einzelnen Schritte findet man in den Kapiteln 3.1.1 und 3.1.3. Der erste Schritt ist die Rekonstruktion der Geometrie.

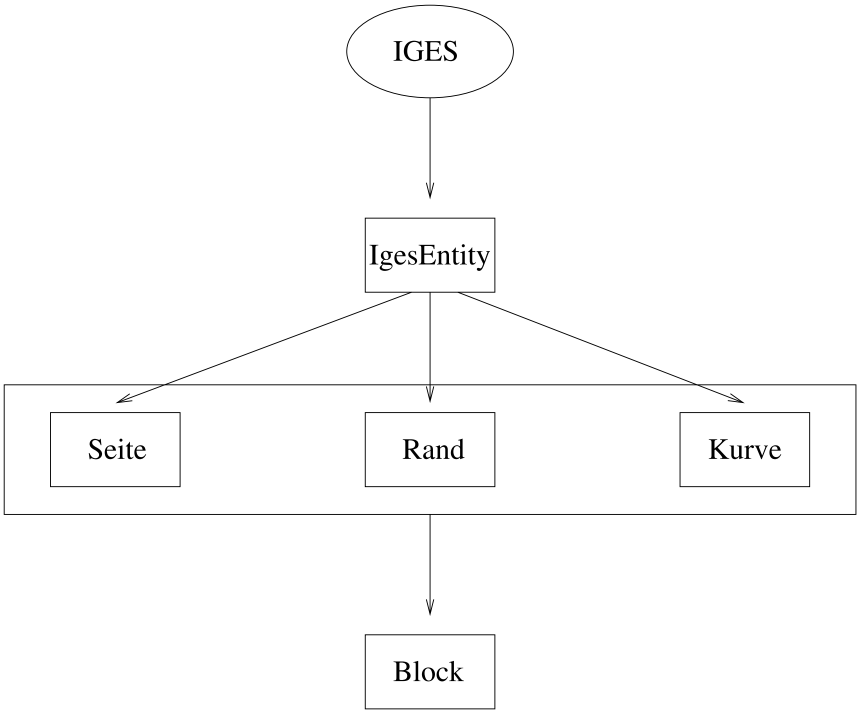

Abbildung 3.8: Schritt 1: Rekonstruktion der Geometrie.

Hier werden Objekte der Klassen `IgesEntity, Seite, Rand, Kurve` und `Block` erzeugt. Die Funktion `readIges()` füllt einen Vektor mit `IgesEntity`-Objekten. Dieser wird an die Funktion `lookForHierachy()` übergeben, die diesen Vektor sortiert und die Entities, falls sie Verweise auf weitere Entities haben, verknüpft. Anschließend werden die Ergebnisse als Objekte der Klassen `Kurve, Rand` und `Seite` abgespeichert, die schließlich in einem `Block`-Objekt zusammengefasst werden. Siehe hierzu auch in Kapitel 3.1.1.
Der zweite Schritt besteht aus der Generierung der einzelnen Gitterblöcke aus der Geometrie, die als `Block`-Objekt abgespeichert ist, sowie aus Informationen über die Blocktopologie, die als Input gegeben werden. Für jeden `Block` aus der `BlockSammlung` wird ein Gitter generiert. Dazu werden Gitterparameter benötigt, die ebenfalls

als Input gegeben werden. Schließlich wird das blockstrukturierte Gitter in einem `GridCollectionNS`-Objekt abgelegt. Die Parameter zur Strömungsberechnung, die für die `GridCollectionNS` benötigt werden, werden ebenfalls aus der Inputdatei eingelesen.

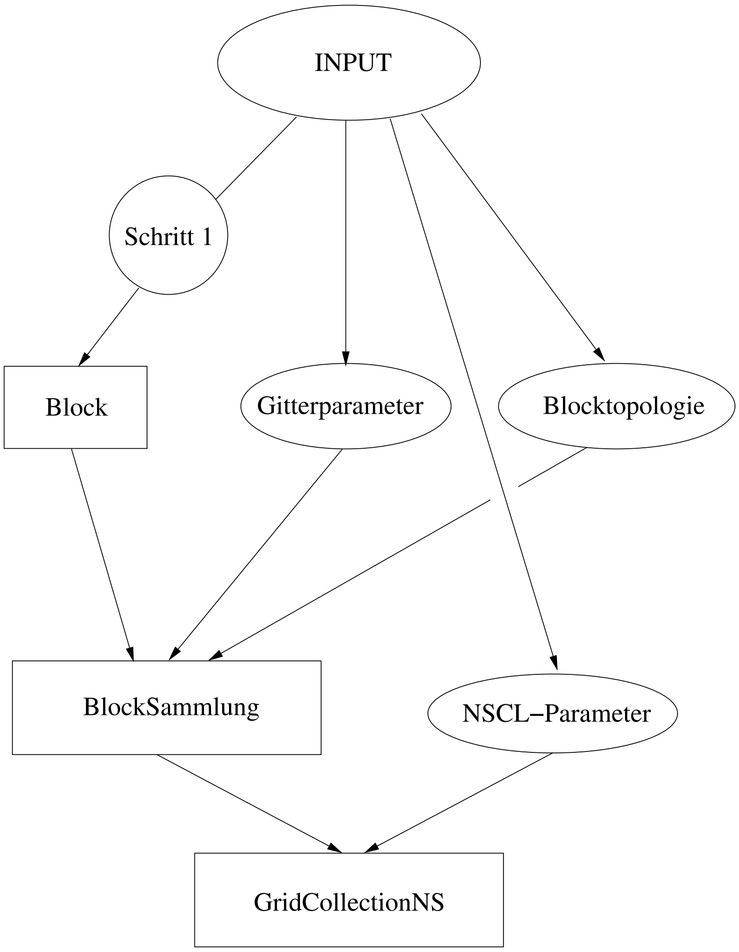

Abbildung 3.9: Schritt 2: Generierung der Gitterblöcke aus der rekonstruierten Geometrie.

Beschreibung der wichtigsten Klassen

IgesEntity repräsentiert eine IGES-Entity, wie sie im Kapitel 2.6.1 beschrieben ist. Dazu hat sie im Wesentlichen eine eindeutige Identitätsnummer, eine Nummer, die den Typ festlegt, und einen Vektor mit typspezifischen Daten. Diese Daten können zum Beispiel NURBS-Parameter oder Verweise auf andere IGES-Entities sein.

Die Klasse **Kurve** enthält ein `NurbsCurve`-Objekt aus der Nurbs++-Bibliothek [19] und zusätzlich eine Position, die beschreibt, in welche Richtung die Kurve vornehmlich verläuft (x_1, x_2, oder x_3).
Ein **Rand** besteht aus mehreren Kurven, die zusammengesetzt eine geschlossene Kurve ergeben. Auch er hat eine eindeutige Position, die besagt, ob er zu einer rechten, linken, vorderen, hinteren, oberen oder unteren Seite eines `Block`s gehört.
Eine **Seite** enthält ein `NurbsSurface`-Objekt aus der Nurbs++-Bibliothek und auch wieder eine Position. Hier bestimmt die Position, ob die `Seite` linke, rechte, obere, untere, vordere oder hintere Seite eines `Block`s ist. Des Weiteren enthält sie mindestens ein `Rand`-Objekt. Gibt es mehrere `Rand`-Objekte, so hat die `Seite` mindestens einen inneren `Rand`, der ein Loch in der `Seite` beschreibt. Jede `Seite` hat genau einen äusseren `Rand`.
Gitter ist die Klasse, für ein einzelnes strukturiertes Gitter. Sie enthält die Anzahl der Gitterpunkte in jeder Richtung, die Parameter zur Gradierung des Gitters, sowie einen Vektor mit dreidimensionalen Punkten, der die Gitterpunkte enthält.
Ein **Block** besteht aus einem Vektor von `Seiten`, deren Flächen aneinandergrenzen und insgesamt die ganze Oberfläche des `Block`s beschreiben. Des Weiteren hat ein `Block` eine eindeutige Identitätsnummer und einen Vektor mit Identitätsnummern von Nachbarn, von denen es an jeder `Seite` einen gibt. Außerdem hat ein `Block` ein `Gitter`.
In der **BlockSammlung** werden die einzelnen `Block`-Objekte zusammengefasst. Sie besteht aus einem Vektor von `Block`-Objekten.

3.3 Lösung des transformierten elliptischen Differentialgleichungssystems

Das quasilineare und damit nichtlineare Gleichungssystem (2.10) bzw. (2.18) wird nun iterativ gelöst. Oft werden zur Lösung nichtlinearer Gleichungssysteme Newton- oder newtonähnliche Verfahren verwendet. Eine Übersicht über solche Methoden wird in [28, Kap. 7] gegeben. Bei dem Newton-Verfahren wird ein Gleichungssystem $G(x) = 0,\ G = (g_1, \ldots, g_n)$ mit Funktionen $g_i : \mathbb{R}^n \longrightarrow \mathbb{R}$ durch eine Iteration $x^{(i+1)} = x^{(i)} - [\nabla G(x^{(i)})]^{-1} G(x^{(i)})$ gelöst. Für größere Werte von n verwendet man zur Invertierung von ∇G wiederum Iterationsverfahren. Hierzu werden Standard-Löser für lineare Gleichungssysteme wie das Jacobi- oder SOR-Verfahren eingesetzt. Dadurch kommt es zu einer geschachtelten Iteration mit einer äußeren Schleife, dem eigentlichen Newton-Schritt, und einer inneren Schleife zur Invertierung von $\nabla G(x^{(i)})$. In jedem Newton-Schritt, also der äußeren Iteration, muss die Jacobimatrix ∇G berechnet werden, was sehr aufwändig sein kann. So haben Versuche mit PETSC [2] gezeigt, dass die Lösung von (2.18) mit dem Newton-Verfahren, trotz quadratischer Konvergenz, die dem Newton-Verfahren zu Eigen ist, unakzeptabel lange dauert. Eine andere, einfachere, aber in diesem Fall völlig ausreichende Methode, ist die nachfolgend beschriebene. Es handelt sich um eine vereinfachte Form des Newton-Verfahrens, bei der die Matrix $\nabla G(x^{(i)})$ nicht berechnet werden muss. Um diese Methode zu definieren, wird nochmal ein iterativer Prozess, auch *Picard-Iteration* genannt, im Allgemeinen definiert.

3.3.1 Picard-Iteration

Die Picard-Iteration ist gegeben durch einen *iterativen Prozess*, der nachfolgend definiert wird.

Definition 3 *Eine Familie von Operatoren Φ_k,*

$$\Phi_k : D_k \subset (\mathbb{R}^n)^{p+k} = \underbrace{\mathbb{R}^n \times \cdots \times \mathbb{R}^n}_{k+p\ mal} \longrightarrow \mathbb{R}^n, \quad k = 1, 2, \ldots$$

definiert einen iterativen Prozess $It = (\{\Phi_k\}, D^*, p)$ *mit* p Anfangspunkten *und mit* Gebiet $D^* \subset D_0$, *falls D^* nicht leer ist, und es einen Punkt $(x^{(0)}, \ldots, x^{(p-1)}) \in D^*$ gibt, so dass die Folge $x^{(k)}$, generiert durch*

$$x^{(k+1)} = \Phi_k(x^{(0)}, \ldots, x^{(k+p-1)}), \quad k = 0, 1, 2, 3, \ldots$$

existiert. Das heißt, dass $(x^{(0)}, \ldots, x^{(k+p-1)}) \in D_k$ für alle $k \geq 0$. Ein Punkt x^* mit $x^* = \lim_{k\to\infty} x_k$ heißt Fixpunkt *des iterativen Prozesses.*

Die in der Praxis vorkommenden iterativen Prozesse sind meistens von der folgenden Art.

Definition 4 *Ein iterativer Prozess $It = (\{\Phi_k\}, D^*, p)$ ist ein* m-Schrittverfahren, *wenn $p = m$ und die Abbildungen Φ_k von der Form*

$$\Phi_k : D_k \subset (\mathbb{R}^n)^m \longrightarrow \mathbb{R}^n, \quad k = 1, 2, 3, \ldots$$

sind. It wird stationär *genannt, wenn es eine Abbildung Φ mit $\Phi_k \equiv \Phi gibt$, und $D_k \equiv D$ für $k = 0, 1, 2, \ldots$ gilt.*

Wesentlich bei iterativen Prozessen ist die Bestimmung des Konvergenzgebietes , das heißt des Gebietes $D_{conv} \subset D^*$, für dessen Elemente der Prozess gegen einen Fixpunkt konvergiert. So wird im Folgenden die Familie $\{\Phi_k\}$ bestimmt und dann versucht das Gebiet D_{conv} zu charakterisieren.
Das Newton Verfahren sowie auch die hier benutzte Methode sind nicht stationäre 1-Schrittverfahren. Die Abbildung Φ_k ändert sich also mit jedem Iterationsschritt

und hängt dabei nur von der aktuellen Iterierten x^k ab.
Sei F der, wie nachfolgend beschrieben diskretisierte, Differentialoperator des Systems (2.18), dann lässt sich das Gleichungssystem (2.18) diskretisiert darstellen als

$$F(x) = A(x)x = b, \tag{3.1}$$

mit einer $n \times n$-Matrix $A(x)$, deren Koeffizienten $a_{ij}(x)$ von x abhängen und $\mathbb{R}^n \ni b = 0$, bis auf die mit den Randwerten korrespondierenden Stellen. Für die Newton-Iteration lautet Φ_k dann

$$\begin{aligned} \Phi_k(x^{(k)}) &= x^{(k)} - \nabla[A(x^{(k)})x^{(k)}]^{-1}\,[A(x^{(k)})x^{(k)} - b] \\ &= x^{(k)} - [\nabla A(x^{(k)})x^{(k)} + A(x^{(k)})]^{-1}\,[A(x^{(k)})x^{(k)} - b]. \end{aligned} \tag{3.2}$$

Die Vereinfachung besteht nun darin, den aufwändig zu berechnenden Teil in (3.2), $\nabla A(x^{(k)})x^{(k)}$, wegzulassen, so dass die Iterationsabbildung

$$\begin{aligned} \Phi_k(x^{(k)}) &= x^{(k)} - A(x^{(k)})^{-1}\,[A(x^{(k)})x^{(k)} - b] \\ &= A(x^{(k)})^{-1}\,b \end{aligned} \tag{3.3}$$

lautet. Dies kommt einer Linearisierung des Gleichungssystems $F(x) = b$ gleich. An Stelle des nichtlinearen Systems (3.1) mit nicht konstanten Koeffizienten, wird in jedem Iterationsschritt das lineare System (3.3), dessen Koeffizienten mit der aktuellen Näherung an die Lösung $x^{(k)}$ berechnet wurden, gelöst. Man muss also in jedem Schritt der Picard-Iteration das lineare Gleichungssystem $A(x^k)x = b$ lösen. So kommt man auch hier, wie beim Newton-Verfahren, zu einer geschachtelten Iteration:
In der äußeren Schleife wird die Matrix $A(x^{(k)})$ mit dem aktuellen Wert von x aufgestellt. In der inneren Schleife wird dann (3.3) gelöst, um $x^{(k+1)}$ zu erhalten.
Nun zur Konvergenz.

Definition 5 *Sei $\Phi : D \subset \mathbb{R}^n \longrightarrow \mathbb{R}^n$. x^* heißt* Attraktionspunkt *der Abbildung Φ, falls es eine Umgebung $U(x^*) \subset D$ von x^* gibt, so dass die durch $x^{(k+1)} = \Phi(x^{(k)})$ Iterierten $\{x^{(k)}\}$ für jedes $x^{(0)} \in U$ in D liegen und gegen x^* konvergieren.*

Laut dem Ostrofski Theorem [28, 10.1.3] ist ein Fixpunkt auch Attraktionspunkt, falls Φ in x^* differenzierbar ist und gilt

$$\rho(\nabla\Phi(x^*)) < 1.$$

ρ ist dabei der Spektralradius. Für die hier verwendete Iterationsabbildung gilt laut [28, Satz 10.2.1]

$$\begin{aligned} \nabla\Phi(x^*) &= I - A(x^*)^{-1}\nabla F(x^*), \quad \text{was sich ergibt zu} \qquad (3.4) \\ &= I - A(x^*)^{-1}[\nabla A(x^*)x^* + A(x^*)] \\ &= I - A(x^*)^{-1}A(x^*) - A(x^*)^{-1}\nabla A(x^*)x^* \\ &= -A(x^*)^{-1}\nabla A(x^*)x^*. \end{aligned}$$

Um Aussagen über die lokale Konvergenz der Picard-Iteration machen zu können, muss also der Spektralradius $A(x^*)^{-1}\nabla A(x^*)x^*$ abgeschätzt werden. Diese Abschätzung wurde nicht durchgeführt. Trotzdem wird davon ausgegengen, dass $\rho(\nabla\Phi(x^*)) < 1$ für ausreichend gute Startwerte gilt. Bei allen während dieser Arbeit durchgeführten Gittergenerierungen konvergierte die Picard-Iteration für mittels Transfiniter Interpolation (siehe 2.3) erzeugte Startwerte.
Die Fixpunkte der Picard-Iteration sind auch diskrete Lösungen von (2.18):

$$x^* = \phi(x^*) = A(x^*)^{-1}(b).$$

Siehe zur Konvergenz auch in den Abschnitten 3.3.3, 3.3.4 und 3.3.6.

3.3.2 Diskretisierung

Das Gleichungssystem (2.18)

$$0 = \sum_{i,j=1}^{3} g^{ij}\frac{\partial^2 x_l}{\partial\xi_i\partial\xi_j} + \sum_{k=1}^{3}\sum_{i,j=1}^{3} g^{ij}P_{ij}^k\frac{\partial x_l}{\partial\xi_k} \quad l = 1, 2, 3.$$

soll nun diskretisiert werden. Wie im vorigen Abschnitt beschrieben, werden die Koeffizienten $g^{ij}(x)$ zum aktuellen Wert von x berechnet. Dies geschieht nach der Formel

$$\begin{aligned} g^{ij} &= \nabla\xi_i \cdot \nabla\xi_j, \quad \text{wobei} \quad \nabla\xi_i \quad \text{berechnet wird durch} \\ \nabla\xi_i &= \frac{1}{det(\nabla x)}\frac{\partial x}{\partial\xi_j} \times \frac{\partial x}{\partial\xi_k}. \end{aligned}$$

Dabei gilt $i, j, k = 1, 2, 3$ in zyklischer Reihenfolge. Siehe zur Herleitung dieser Formeln in den Anhang A.1.
Die Berechnung der Kontrollfunktionen P_{ij}^k erfolgt, wie in Abschnitt 2.4.3 beschrieben. Die partiellen Ableitungen werden mit finiten Differenzen diskretisiert. Für die ersten Ableitungen zur Berechnung der g^{ij}, sowie für die partiellen Ableitungen in dem Term erster Ordnung werden zentrale Differenzen benutzt. Die reinen zweiten Ableitungen werden durch eine dreiminsionale Form des 3-Punkte Sterns $\frac{1}{h^2}[1\ \text{-}2\ 1]$ diskretisiert.
Für die gemischten Ableitungen wurden zwei verschiedene Arten von Sternen implementiert. Erstens ein 4-Punkte Stern, in zweidimensionaler Schreibweise,

$$\frac{1}{h^2}\begin{bmatrix} -\frac{1}{4} & 0 & \frac{1}{4} \\ 0 & 0 & 0 \\ \frac{1}{4} & 0 & -\frac{1}{4} \end{bmatrix}, \tag{3.5}$$

und die beiden 7-Punkte Sterne,

$$(a)\ \frac{1}{h^2}\begin{bmatrix} -1 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -1 \end{bmatrix} \quad \text{oder (b)}\ \frac{1}{h^2}\begin{bmatrix} 0 & -1 & 1 \\ -1 & 2 & -1 \\ 1 & -1 & 0 \end{bmatrix}. \tag{3.6}$$

h ist hierbei die Maschenweite des zugrunde liegenden Gitters in Ξ. Der Einfachheit halber wird hier und in den folgenden Beweisen Maschenweite in jeder Richtung h_i gleich h gesetzt. Dies stellt aber keine Einschränkung der Allgemeinheit dar. Der Stern (a) wird benutzt, wenn der entsprechende Koeffizient negativ ist, der Stern (b) im anderen Fall.
Der 4-Punkte Stern ist der, den man durch wiederholte Anwendung einer zentralen Differenz in der jeweiligen Richtung erhält. Die Sterne (3.6) ermöglichen, dass die entstehende Matrix die M-Matrix-Eigenschaft haben kann, Satz[12, 2.5]. Eine M-Matrix zeichnet sich durch positive Diagonaleinträge, nichtnegative Außerdiagonaleinträge und eine elementweise nichtnegative Inverse aus. Der Satz[12, 2.5] ist im Zweidimensionalen formuliert, die Argumente gelten aber auch im Dreidimensioanalen. Es wird gezeigt, dass falls $g^{ii} > g^{ij}\frac{h}{2}P_{ij}^k$ für $i, j, k = 1, 2, 3$ gilt, die unter Verwendung von (3.6) entstehende Matrix eine M-Matrix ist und das Verfahren konsistent von zweiter Ordnung ist. Aufgrund der Elliptizität des verwendeten Differentialoperators und der per Konstruktion moderaten Werte von P_{ij}^k ist die Bedingung an die Koeffizienten schon für relativ große Werte von h erfüllt. Mittels Taylorapproximation rechnet man nach, dass auch die anderen hier aufgeführten Differenzensterne konsistent von zweiter Ordnung sind. Tests haben gezeigt, dass im Fall äquidistanter Randpunkteverteilung die Diskretisierung mit dem 4-Punkte Stern zu besseren Ergebnissen geführt hat, während bei nicht äquidistanter Randpunkteverteilung die 7-Punkte Sterne bessere Ergebnisse lieferten.

Konvergenz des Verfahrens

Seien Ξ_p der Raum der Gitterpunkte $\xi_p$in Ξ und X_h^l der Raum der Gitterfunktionen von Ξ_p nach X^l, $l = 1, 2, 3$. X^l bezeichnet hier die Projektion von X auf die l-te Koordinate. Um die kontinuierlichen Lösungen x^l von (2.18) mit den diskreten Lösungen x_h^l vergleichen zu können, wird eine *Restriktion* definiert. Diese sei gegeben durch $R_h : X^l \longrightarrow X_h^l$, mit $R_h(x_h^l) = x_h^l$ für eine Gitterfunktion x_h^l. Die Konvergenz der diskreten Lösung gegen die kontinuierliche Lösung für h gegen 0 ist dann im Fall der 7-Punkt-Diskretisierugegeben durch den folgenden Satz.

Satz 6 *Für die diskreten Lösungen x_h^l von (2.18) gilt mit der oben beschrieben Diskretisierung im Fall der 7-Punkte-Sterne (3.6) $||x_h^l - R_h(x^l)||_\infty = O(h^2)$, falls $x^l \in C^4(\Xi)$. Das Verfahren ist dann also konvergent von zweiter Ordnung.*

Beweis:
Die Konvergenz von der Ordnung 2 folgt aus der Konsistenzordnug 2 der verwendeten Differenzensterne und der Stabilität der zum Differenzenoperator gehörenden Matrix L_h ([13, Satz 4.5.3]). □
Damit ist die Konvergenz für die Diskretisierung durch die 7-Punkte-Sterne bewiesen. Für die 4-Punkt-Diskretisierung der zweiten Ableitungen wurde die Stabilität der entstehenden Matrix nicht formal nachgewiesen. Eigenwertanalysen mit Matlab haben in den betrachteten Fällen eine Beschränkung der Beträge der Eigenwerte von Null weg ergeben, so dass auch für diese Matrix Stabilität angenommen wird.

Regularität der diskreten Lösung

Im Folgenden wird eine Abschätzung für die Regularität der diskreten Lösung x_h^l von (2.10) beziehungsweise (2.18) in der diskreten H_2-Norm hergeleitet. Diese Abschätzung wird in Abschnitt 3.3.6 benutzt, um numerische Ergebnisse theoretisch zu untermauern.
Die diskrete H_2-Norm ist für $u_h \in X_h^l$ ist definiert durch

$$||u_h||_{H_2} := (||u_h||_{H_1}^2 + \sum_{\xi_p \in \Xi_p} \sum_{i,j=1}^{3} h^3 |D_{ij} u_h(\xi_p)|)^{\frac{1}{2}}. \tag{3.7}$$

Dabei ist D_{ij} der der Diskretisierung entsprechende Differenzenoperator zweiter Ordnung. Die diskrete H_1- beziehungsweise H_0-Norm werden analog definiert. Die Regularitätsabschätzung wird aus der oben bewiesenen Konvergenz von zweiter Ordnung und der *inversen Abschätzung* für finite Differenzen folgen.

Lemma 2 *Mit der durch* (3.7) *definierten Norm gilt für* $x_h^l \in X_h^l$ *die* inverse Abschätzung

$$||x_h^l||_{H_2} \leq h^{-2} c ||x_h^l||_{H_0}.$$

Beweis:
Der Beweis wird für Diskretisierungen mit zentralen Differenzen geführt. Die Aussage gilt aber genauso für Diskretisierungen mit Vorwärts-, Rückwärts- oder kombinierten Differenzen.
Für D_{ij} gilt:

$$\begin{aligned} h^3 \sum_{\xi_p \in \Xi_p} |D_{ij} x_h^l(\xi_p)| &= h^3 \sum_{\xi_p \in \Xi_p} \frac{|D_j x_h^l(\xi_p^{i+}) - D_j x_h^l(\xi_p^{i-})|}{2h} \\ &\leq h^3 \sum_{\xi_p \in \Xi_p} \frac{|D_j x_h^l(\xi_p)|}{h} \\ &\leq h^{-1} ||x_h^l||_{H_1}. \end{aligned} \tag{3.8}$$

Daraus erhält man

$$||x_h^l||_{H_2} \leq h^{-1}c||x_h^l||_{H_1}$$

und durch Wiederholung der Argumente aus (3.8) schließlich

$$||x_h^l||_{H_2} \leq h^{-2}c||x_h^l||_{H_0}.$$

Hierbei sind in (3.8) ξ_p^{i-} beziehungsweise ξ_p^{i+} die zu $\xi_p(i,j,k)$ benachbarten Gitterpunkte $\xi_p(i-1,j,k)$ beziehungsweise $\xi_p(i+1,j,k)$. Die Konstanten c hängen von der Dimension der betrachteten Räume ab. □

Mit Satz 6 und Lemma 2 folgt nun Satz 7.

Satz 7 *Für die diskreten Lösungen x_h^l von (2.9) und (2.18) gilt die Abschätzung*

$$||R_h(x^l) - x_h^l||_{H_2} \leq c$$

mit einer Konstanten $c > 0$.

Beweis:
Lemma 2 liefert

$$||R_h(x^l) - x_h^l||_{H_2} \leq h^{-2}c||R_h(x^l) - x_h^l||_{H_0}.$$

Wegen

$$\begin{aligned} ||x_h^l||_{H_0} &= (h^3 \sum_{\xi_p \in \Xi_p} |x_h^l(\xi_p)|^2)^{\frac{1}{2}} \\ &\leq h^3|\Xi_p| \max_{\xi_p \in \Xi_p} |x_h^l(\xi_p)| \\ &= \max_{\xi_p \in \Xi_p} |x_h^l(\xi_p)| \\ &= ||x_h^l||_\infty \end{aligned}$$

gilt mit Satz 6

$$||R_h(x) - x_h^l||_{H_2} \leq h^{-2}ch^2 = c,$$

und damit die Behauptung. $|\Xi_p|$ ist dabei die Anzahl der Elemente in Ξ_p. □

Das Ergebnis wird benutzt, um bei der diskreten Lösung des transformierten Laplace-Systems auch Aussagen über die Lösung des nicht transformierten Laplace-Systems machen zu können. Es wird abgeschätzt, wie genau das Laplace-System erfüllt ist.

3.3.3 Konvergenzanalyse des linearen Problems

Hier werden die Ergebnisse der verschiedenen implementierten Löser für das lineare Teilproblem verglichen. Es wird das lineare Gleichungssystem des ersten Iterationsschrittes der Picard-Iteration betrachtet, und dort das Gleichungssystem für die

x_3-Koordinate. Außerdem werden zwei unterschiedliche Geometrien betrachtet, eine mit stark nicht konvexem Rand, so dass der durch Transfinite Interpolation erzeugte Startwert relativ schlecht ist, und eine "gemäßigtere" Geometrie. Die Querschnitte der Geometrien sind in Abbildung 3.10 dargestellt. Es handelt sich um "2 1/2 dimensionale" Geometrien, das heißt alle Querschnitte bezüglich einer Koordinatenrichtung (hier die Schnitte mit $x_2 = const.$) sehen gleich aus. Bei beiden Geometrien werden die beiden in Abschnitt 2.4 besprochenen Gleichungssysteme, Laplace- und Poisson-System, gelöst. Damit sich die Gleichungen für die beiden Gleichungstypen unterscheiden, werden gradierte Gitter betrachtet, deren Gitterlinien oben dichter liegen als unten (siehe Abbildung 3.10). Dafür werden die Randpunkte an den entsprechenden Stellen dichter vorgegeben. Da die Kontrollfunktionen (2.15) über die Abweichung der Randkurven von Bogenlängenparametrisierung berechnet werden, erhält man so unterschiedliche Gleichungen für die beiden Systeme (siehe Abschnitt 2.4.3).
Die Gleichungssysteme wurden jeweils berechnet für Gitter mit $8^3, 16^3, 32^3$ und 64^3 Gitterpunkten. Die Größe des zugehörigen linearen Gleichungssystems ist $n \times n$, wobei n die Anzahl der Gitterpunkte beschreibt. Das Residuum wird gemessen in der Norm $\|r\| = (\sum_{i=1}^{n} r_i^2/n)^{\frac{1}{2}}$ für $r \in \mathbb{R}^n$. Abbruchkriterium für die Iterationen ist $\|Ax - b\| < 10^{-14}$. In den folgenden Grafiken ist die Konvergenz der verschiedenen

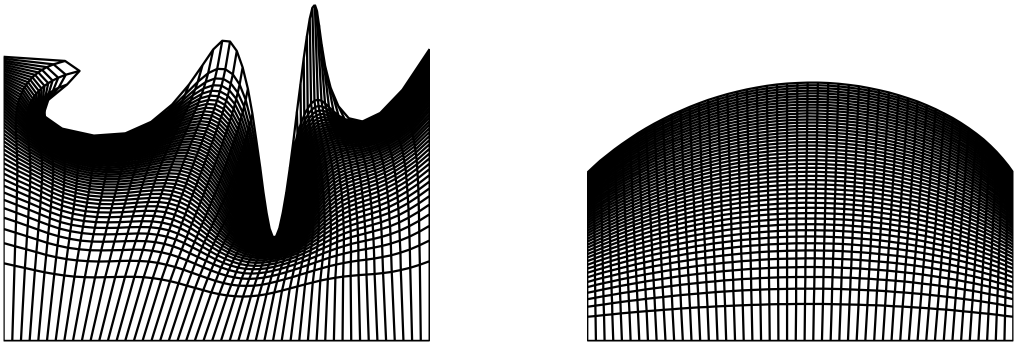

Abbildung 3.10: Querschnitt durch gradierte Poisson Gitter für komplizierte und einfache Geometrie. Die x_3-Koordinatenwerte beschreiben die vertikale Position und die x_1-Koordinatenwerte die horizontale Position der Gitterpunkte.

Verfahren in den unterschiedlichen Fällen dargestellt. Die Kurven zeigen die Residuumsnorm zur entsprechenden Anzahl der Iterationen. Bei dem SOR-Verfahren wurde, je nach Maschenweite, der Abbildung 3.15 zu entnehmende optimale Wert für den Relaxationsparameter ω benutzt.

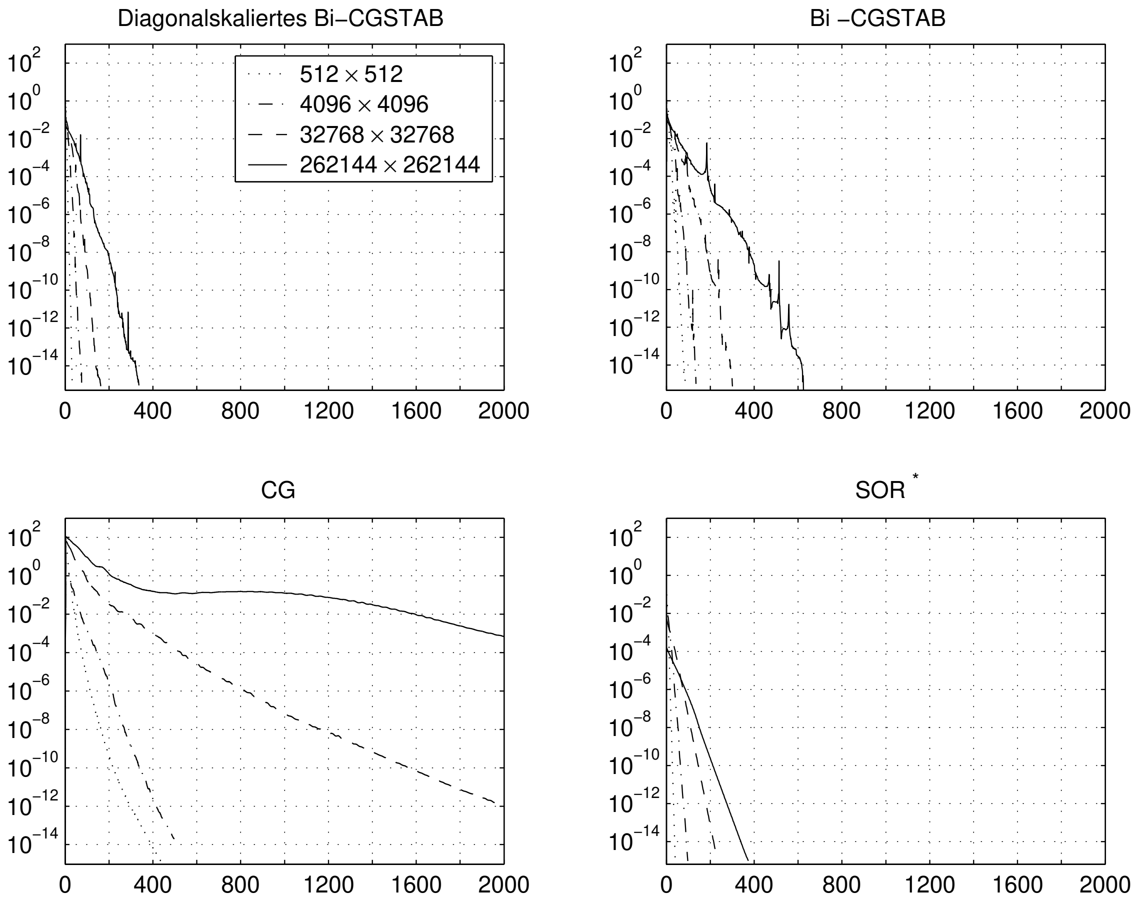

Abbildung 3.11: Konvergenz der verschiedenen Verfahren für das Laplace-System bei einfacher Geometrie. x-Achse: Anzahl der Iterationen, y-Achse: 2-Norm des Residuums.

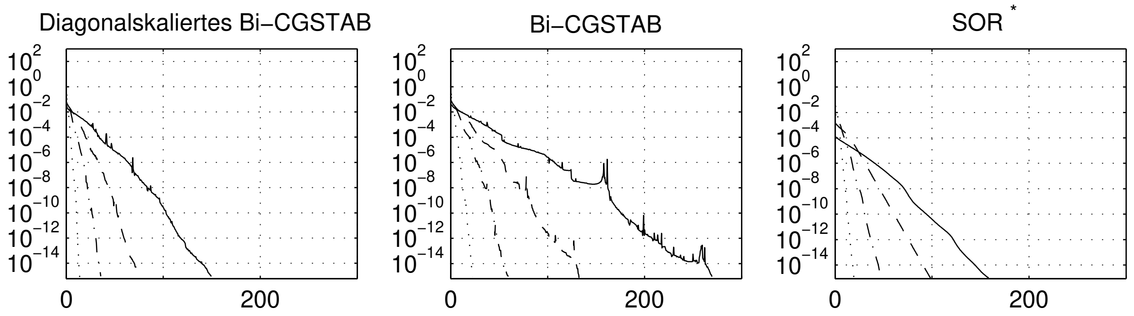

Abbildung 3.12: Konvergenz der verschiedenen Verfahren für das Poisson-System bei einfacher Geometrie. x-Achse: Anzahl der Iterationen, y-Achse: 2-Norm des Residuums.

*Das SOR-Verfahren wurde mit dem jeweils für die entsprechende Gittergröße optimalen Wert für den Relaxationsparameter ω benutzt. Die jeweiligen Werte sind Abbildung 3.15 zu entnehmen.

Matrixgröße	Löser	Iterationen	Gesamtzeit in Sekunden	Zeit/Iteration in Sekunden
512×512	SOR 1.4	40	0.00987	3.108e-04
512×512	CG	162	0.02747	3.030e-04
512×512	BICGSTAB	54	0.01642	3.410e-04
512×512	PBICGSTAB	33	0.01260	2.850e-04
4096×4096	SOR 1.7	98	0.76076	8.404e-03
4096×4096	CG	649	2.74609	7.746e-03
4096×4096	BICGSTAB	134	1.21698	8.855e-03
4096×4096	PBICGSTAB	75	0.86323	1.245e-02
32768×32768	SOR 1.8	233	24.02268	1.714e-01
32768×32768	CG	2716	154.65589	9.574e-02
32768×32768	BICGSTAB	300	40.55018	1.309e-01
32768×32768	PBICGSTAB	161	27.22674	1.659e-01
262144×262144	SOR 1.9	407	389.77375	9.545e-01
262144×262144	CG	8258	4573.04146	9.292e-01
262144×262144	BICGSTAB	624	807.23422	1.036e+00
262144×262144	PBICGSTAB	335	554.48750	1.660e+00

Tabelle 3.1: Vergleich der verschiedenen Löser bei unterschiedlicher Matrixgröße für das Laplace-System bei einfacher Geometrie.

Matrixgröße	Löser	Iterationen	Gesamtzeit in Sekunden	Zeit/Iteration in Sekunden
512×512	SOR 1.4	39	0.00938	3.449e-04
512×512	BICGSTAB	45	0.06120	1.969e-04
512×512	PBICGSTAB	28	0.07478	2.620e-04
4096×4096	SOR 1.7	95	0.75612	8.447e-03
4096×4096	BICGSTAB	119	1.07477	9.547e-03
4096×4096	PBICGSTAB	71	0.76864	1.105e-02
32768×32768	SOR 1.8	197	19.33903	9.985e-02
32768×32768	BICGSTAB	265	35.11232	1.374e-01
32768×32768	PBICGSTAB	151	24.36397	1.579e-01
262144×262144	SOR 1.9	317	300.01077	1.057e+00
262144×262144	BICGSTAB	539	694.98675	1.332e+00
262144×262144	PBICGSTAB	299	469.81583	1.554e+00

Tabelle 3.2: Vergleich der verschiedenen Löser bei unterschiedlicher Matrixgröße für das Poisson-System bei einfacher Geometrie.

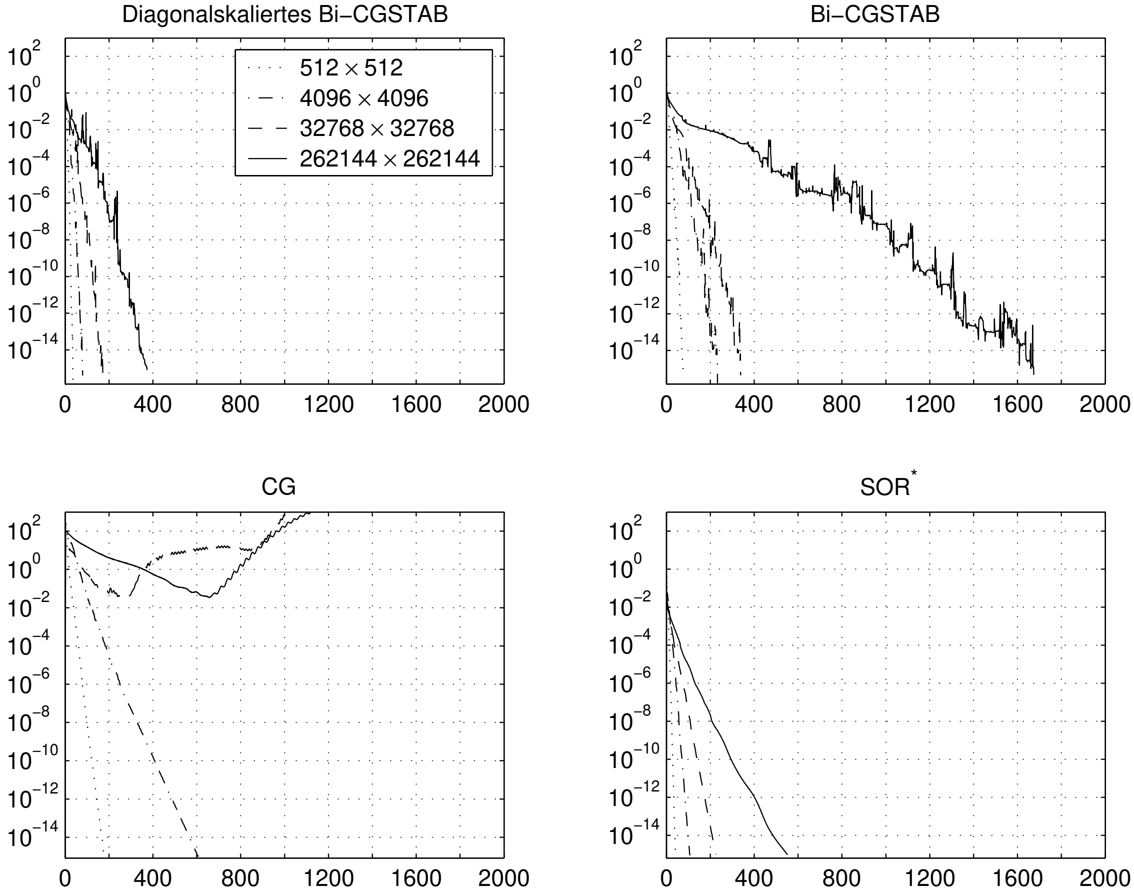

Abbildung 3.13: Konvergenz der verschiedenen Verfahren für das Laplace-System bei komplizierter Geometrie. x-Achse: Anzahl der Iterationen, y-Achse: 2-Norm des Residuums.

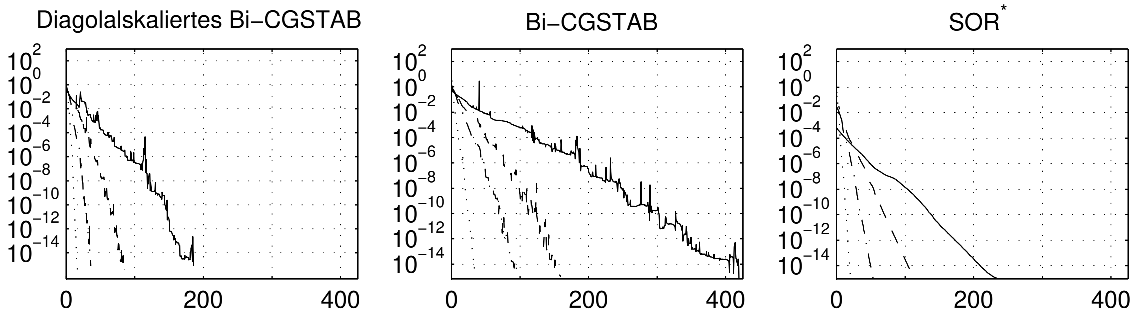

Abbildung 3.14: Konvergenz der verschiedenen Verfahren für das Poisson-System bei komplizierter Geometrie. x-Achse: Anzahl der Iterationen, y-Achse: 2-Norm des Residuums.

*Das SOR-Verfahren wurde mit dem jeweils für die entsprechende Gittergröße optimalen Wert für den Relaxationsparameter ω benutzt. Die jeweiligen Werte sind Abbildung 3.15 zu entnehmen.

Matrixgröße	Löser	Iterationen	Gesamtzeit in Sekunden	Zeit/Iteration in Sekunden
512×512	SOR 1.4	42	0.00929	2.969e-04
512×512	CG	178	0.02390	2.799e-04
512×512	BICGSTAB	76	0.01846	1.909e-04
512×512	PBICGSTAB	36	0.01153	2.430e-04
4096×4096	SOR 1.7	107	0.96636	9.026e-03
4096×4096	CG	607	2.93691	9.005e-03
4096×4096	BICGSTAB	233	2.33639	9.705e-03
4096×4096	PBICGSTAB	80	1.04195	1.197e-02
32768×32768	SOR 1.8	226	22.17362	9.889e-02
32768×32768	CG	divergiert	-	9.503e-02
32768×32768	BICGSTAB	338	44.64474	1.311e-01
32768×32768	PBICGSTAB	172	27.63710	1.581e-01
262144×262144	SOR 1.9	1166	1108.19386	9.459e-01
262144×262144	CG	divergiert	-	1.167e+00
262144×262144	BICGSTAB	1674	2156.48167	1.275e+00
262144×262144	PBICGSTAB	374	596.78711	1.555e+00

Tabelle 3.3: Vergleich der verschiedenen Löser bei unterschiedlicher Matrixgröße für das Laplace-System und komplizierter Geometrie.

Matrixgröße	Löser	Iterationen	Gesamtzeit in Sekunden	Zeit/Iteration in Sekunden
512×512	SOR 1.4	41	0.01145	4.329e-04
512×512	BICGSTAB	68	0.03368	2.139e-04
512×512	PBICGSTAB	33	0.01304	2.979e-04
4096×4096	SOR 1.7	107	0.93126	8.565e-03
4096×4096	BICGSTAB	192	1.86895	9.600e-03
4096×4096	PBICGSTAB	72	0.92323	1.207e-02
32768×32768	SOR 1.8	226	22.33319	9.837e-02
32768×32768	BICGSTAB	318	36.13266	1.178e-01
32768×32768	PBICGSTAB	170	27.62118	1.597e-01
262144×262144	SOR 1.9	469	442.38322	9.449e-01
262144×262144	BICGSTAB	837	971.89868	1.327e+00
262144×262144	PBICGSTAB	371	458.04469	1.223e+00

Tabelle 3.4: Vergleich der verschiedenen Löser bei unterschiedlicher Matrixgröße für das Poisson-System und komplizierter Geometrie.

Der Bi-CGSTAB-Löser konvergiert in allen Fällen erwartungsgemäß deutlich schneller als das CG-Verfahren. Seine Konvergenzkurve hat den üblichen zackigen Verlauf, siehe zum Beispiel [7] oder [22, Kap 5.9]. Die Diagonalskaliereung wirkt sich in allen Fällen positiv auf die Konvergenz aus. Am stärksten ist dieser Effekt bei dem Poisson-System mit einfacher Geometrie.
Der Vergleich der Berechnungen des Laplace-Systems mit dem CG-Verfahren zeigt die Abhängigkeit der Systemmatrix von der Geometrie. Durch die Ortsabhängigkeit der Koeffizienten g^{ij} der Differentialgleichung kann sich hier eine Asymmetrie ergeben. Bei der einfachen Geometrie variiert der kontravariante Metrische Tensor $(g^{ij})_{ij=1,\ldots,3}$ nur wenig, so dass die Asymmetrie der Matrix nur schwach ausfällt und das CG-Verfahren noch konvergiert. Bei der komplizierten Geometrie divergiert das Verfahren. Für das Poisson-System divergiert das CG-Verfahren in beiden Fällen, da die Matrix aufgrund der zentralen Differenzen zur Berechnung der Ableitungen ersten Grades asymmetrisch wird.
Das SOR-Verfahren mit optimal gewähltem Relaxationsparameter ω ist das schnellste der hier betrachteten Verfahren. Es benötigt zwar geringfügig mehr Iterationen als das Bi-CGSTAB-Verfahren, jedoch ist der einzelne Iterationsschritt aufgrund seiner Einfachheit deutlich schneller als ein Bi-CGSTAB-Schritt, so dass das Verfahren insgesamt noch schneller konvergiert.
In Abbildung 3.15 ist deutlich zu erkennen, dass der optimale Wert für den Relaxationsparameter maschenweitenabhängig ist. Mit abnehmender Maschenweite steigt der jeweils optimale Wert für ω. Es ergeben sich keine erkennbaren Unterschiede bezüglich unterschiedlicher Geometrie oder verwendetem Gleichungssystem.

3.3.4 Optimierung der Picard-Iteration

Die Picard-Iteration besteht, wie in Abschnitt 3.3.1 beschrieben, aus einer inneren und einer äußeren Iteration. Die Dauer der numerischen Lösung eines linearen Gleichungssystems hängt von der Konvergenzgeschwindigkeit des Lösers und von der gewünschten Genauigkeit ab. Die Konvergenzgeschwindigkeit der implementierten Löser für das lineare Gleichungssystem wurde in Abschnitt 3.3.3 behandelt.
In diesem Abschnit soll nun untersucht werden, wie genau das (innere) lineare Gleichungssystem gelöst werden muss, um mit der Picard-Iteration eine bestimmte Genauigkeit zu erreichen. Ziel dieses Abschnitts ist, den größten Wert des Abbruchkriteriums für das lineare Gleichungssystem zu finden, der nicht zu einem zusätztlichen Iterationsschritt in der äußeren Schleife führt. Dieser Wert ist dann optimal, bezüglich der Konvergenzgeschwindigkeit, bei einer bestimmten Genauigkeit.
Es wurden für die vier verschiedenen Fälle von Laplace- und Poisson-System bei einfacher und komplizierter Geometrie in vier verschiedenen Problemgrößen (mit $8^3, 16^3, 32^3$ und 64^3 Gitterpunkten) jeweils 15^2 Berechnungen durchgeführt. Die 15^2 Berechnungen setzten sich dabei aus den Kombinationen vom Abbruchkriterium für

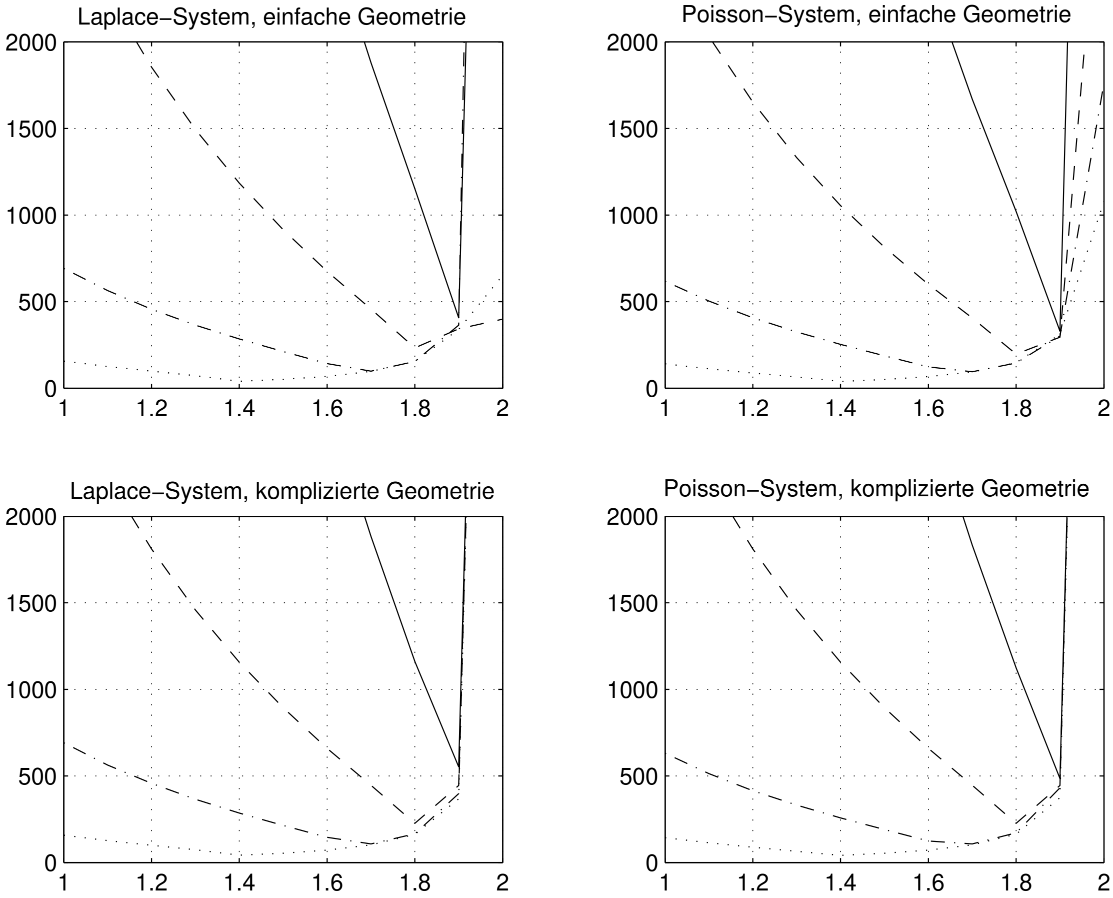

Abbildung 3.15: Anzahl der Iterationen bis zum Abbruch für unterschiedliche Werte des Relaxationsparameters ω.

das lineare Gleichungssystem und dem Abruchkriterium der äußeren Iteration mit Werten jeweils zwischen 10^{-1} und 10^{-15} zusammen. In allen Fällen hat sich die Wahl mit dem gleichen Wert für das innere und das äußere Abbruchkriterium als optimal herausgestellt.

3.3.5 Gedämpfte Picard-Iteration

Oft ist es sinnvoll, bei Picard-Iterationen eine Dämpfung einzuführen. Dabei wird wie beim SOR-Verfahren die nächste Iterierte durch

$$x^{(i+1)} = (1-\lambda)\tilde{x} + \lambda x^{(i)}$$

berechnet, wobei $\tilde{x} = \Phi_i(x^{(i)})$ ist. Dadurch wird in manchen Fällen eine Verbesserung der Konvergenzgeschwindigkeit erreicht. Für die vier verschiedenen Testsituationen

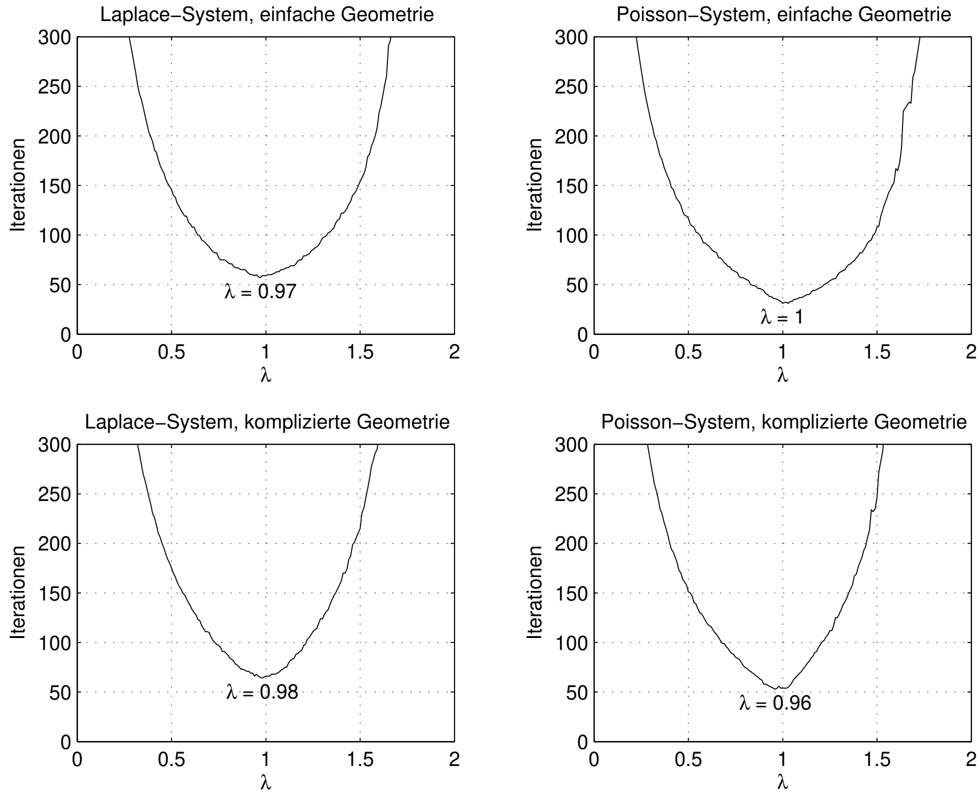

Abbildung 3.16: Benötigte Iterationen bis zum Abbruch für unterschiedliche Dämpfungsparameter λ bei 512 Gitterpunkten.

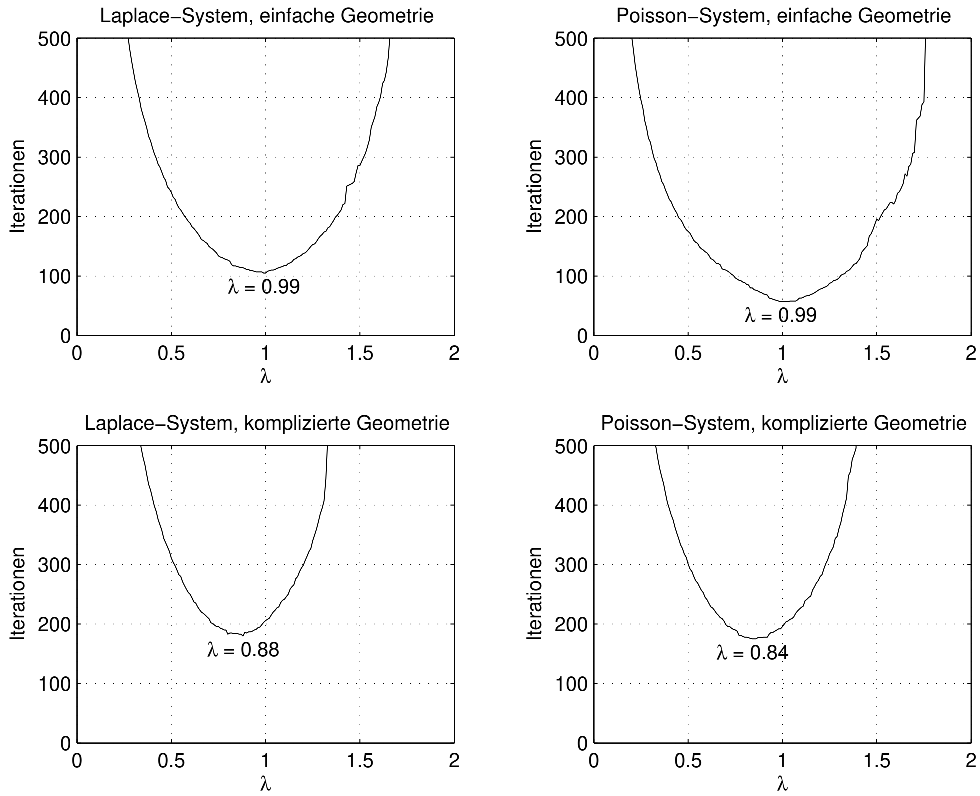

Abbildung 3.17: Benötigte Iterationen bis zum Abbruch für unterschiedliche Dämpfungsparameter λ bei 4096 Gitterpunkten.

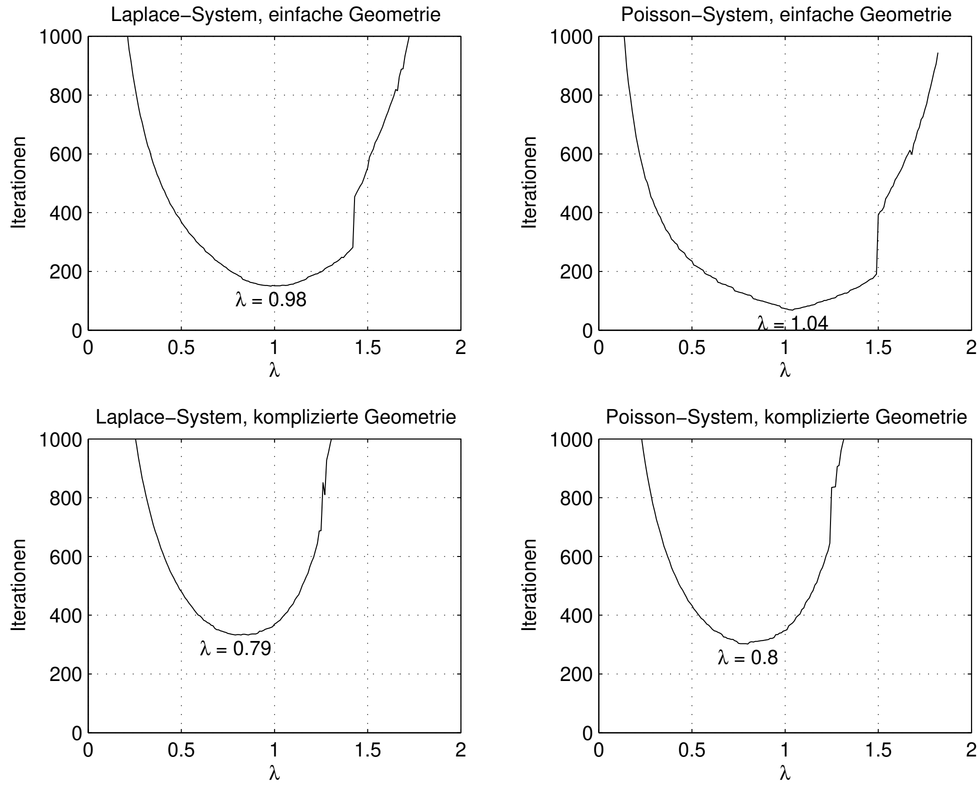

Abbildung 3.18: Benötigte Iterationen bis zum Abbruch für unterschiedliche Dämpfungsparameter λ bei 32768 Gitterpunkten.

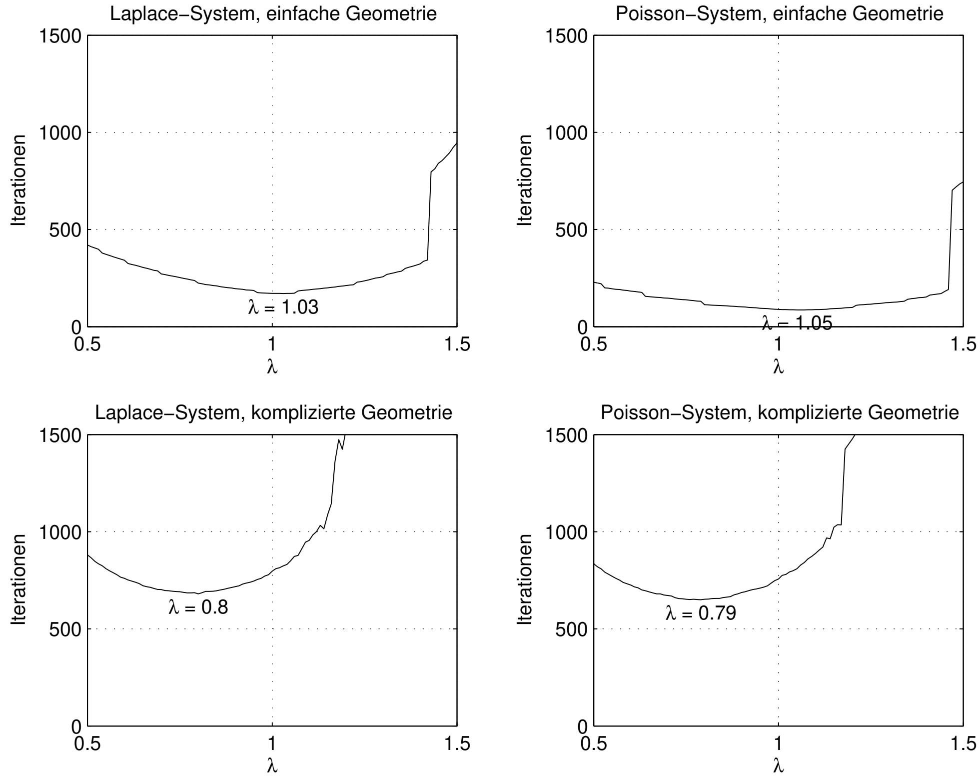

Abbildung 3.19: Benötigte Iterationen bis zum Abbruch für unterschiedliche Dämpfungsparameter λ bei 262144 Gitterpunkten.

mit einfacher und komplizierter Geometrie sowie zu lösenden Laplace- beziehungsweise Poisson-System wurden für die unterschiedlichen Gittergrößen die benötigten (inneren) Iterationen für Dämpfungsparameter λ zwischen 0 und 2 gemessen.
Es hat sich gezeigt, dass der optimale Dämpfungparameter geometrieabhängig ist. Bei der einfachen Geometrie führt eine Dämpfung der Picard-Iteration nicht zu besseren Ergebnissen. Für die komplizierte Geometrie hat sich eine Unterrelaxierung als konvergenzbeschleunigend erwiesen. Im Fall der groben Auflösung ($8 \times 8 \times 8$ Gitterpunkte) wurden die Unglattheiten des Gebietsrandes nicht aufgelöst, so dass hier ein ähnlichches Verhalten des Dämpfungsparameters wie bei der einfacheren Geometrie gemessen wurde.

3.3.6 Verifizierung der Lösung des quasilinearen Systems mittels Matlab

Durch Satz 5 ist dies die Äquivalenz des Laplace-Systems (2.9) und des transformierten Laplace-Systems (2.10) im kontinuierlichen Fall gegeben. Um zu überprüfen, ob die Umkehrabbildung ξ_h der diskreten Lösung x_h von (2.10) tatsächlich (2.9) erfüllt, wurde eine Matlab-Routine geschrieben. Diese berechnet den Laplace-Operator der Koordinatenfunktionen der Umkehrabbildung $\xi = (\xi_1, \xi_2, \xi_3) : X \longrightarrow \Xi$ auf dem Gitter in X. Die Funktionen $\xi_i(x_1, x_2, x_3)$ sind die (bekannten) Koordinatenfunktionen des kartesischen Gitters in Ξ. Die Koordinatenfunktionen x_i sind die berechneten Lösungen von (2.10). Sind diese Lösungen genau, so verschwindet $\Delta_X \xi_i$ auf X.
Die Berechnung des Laplace-Operators auf X erfolgt mit finiten Differenzen. Da das zugrunde liegende Gitter in X nicht äquidistant ist, muss der Differenzenstern für jeden Punkt neu berechnet werden. Aus diesen Differenzensternen setzt sich dann die Matrix für den diskreten Laplace-Operator zusammen. Die Matlab-Routine berechnet diese Sterne und anschließend deren Wirkung auf die diskreten Koordinatenfunktionen ξ_i.

Berechnung des Differenzensterns
Sei $f : X \longrightarrow \mathbb{R}$ eine Funktion auf X. Die Berechnung des Differenzensterns in einem Gitterpunkt x^0 erfolgt mittels Taylorentwicklung der Funktion $f(x)$ in den umliegenden Punkten $x^i, i = 1, ..., 26$. Die Punkte x^i sind durch die Gittertopologie eindeutig bestimmt. Zu den den Punkten $x^i, i = 0, \ldots, 26$ müssen nun Gewichte c_i so bestimmt werden, dass

$$\sum_{i=0}^{26} c_i f(x^i) \approx \Delta f(x^0) \tag{3.9}$$

gilt. Betrachte dazu die Taylorentwicklung von f um x^0 in den Punkten $x_i \quad i = 1, \ldots, 26$

$$f(x^i) = f(x^0) + hDf(x^0) + \frac{1}{2} h_i^T D^2(x^0) h_i + O(|h|^3),$$

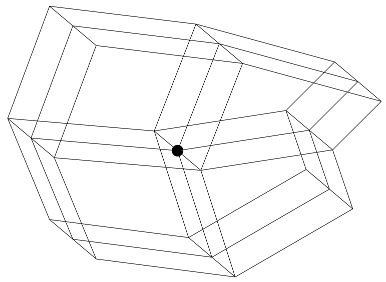

Abbildung 3.20: x_0 und die 26 umgebenden Punkte.

mit $h_i = x^i - x^0$ und $h := max_i(h_i)$.
Damit (3.9) erfüllt ist, müssen folgende Gleichungen gelten:

$$\sum_{i=0}^{26} c_i f(x^0) = 0$$
$$\sum_{i=0}^{26} c_i h_i Df(x^0) = 0$$
$$\sum_{i=0}^{26} \frac{1}{2} c_i h_i^T D^2 f(x^0) h_i = \sum_{i=1}^{3} \frac{\partial^2 f(x^0)}{\partial x_i^2}.$$

Daraus ergeben sich für die c_i die Bedingungen

$$\sum_{i=0}^{26} c_i = 0$$
$$\sum_{i=0}^{26} c_i h_i^j = 0 \quad \text{für} \quad j = 1, 2, 3$$
$$\sum_{i=0}^{26} c_i \frac{1}{2} (h_i^T h_i) : D^2 f(x^0) = Id : D^2 f(x^0).$$

Die letzte Bedingung ist äquivalent zu

$$\sum_{i=0}^{26} c_i \frac{1}{2} (h_i^T h_i) = Id.$$

Dabei ist $(.:.)$ das Salarprodukt zwischen Matrizen: $(a_{ij}) : (b_{ij}) := \sum_{i,j} a_{ij} b_{ij}$.
Um diese Bedingungen nun als lineares Gleichungssystem schreiben zu können, werden folgende Matrizen definiert:

$$H_0 = (\underbrace{1, \ldots, 1}_{27})$$

für die Bedingungen, die sich aus den Termen nullter Ordnung ergeben.

$$H_1 = (h_i^j)_{ji} = (h_0, \ldots, h_{26})$$

für die Bedingungen, die sich aus den Termen erster Ordnung ergeben.

$$H_2 = \begin{pmatrix} h_0^1 h_0^1 & h_1^1 h_1^1 & \ldots & h_{26}^1 h_{26}^1 \\ h_0^1 h_0^2 & h_1^1 h_1^2 & \ldots & h_{26}^1 h_{26}^2 \\ \vdots & \vdots & \ddots & \vdots \\ h_0^3 h_0^3 & h_1^3 h_1^3 & \ldots & h_{26}^3 h_{26}^3 \end{pmatrix} \tag{3.10}$$

für die Bedingungen, die sich aus den Termen zweiter Ordnung ergeben. Damit erhält man für das lineare Gleichungssystem die Matrix

$$H = \begin{pmatrix} H_0 \\ H_1 \\ H_2 \end{pmatrix}. \tag{3.11}$$

Das (überbestimmte) lineare Gleichungssystem für die Einträge des Differenzensternes lautet nun:

$$Hc = (0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1)^T. \tag{3.12}$$

Die Matlab-Routine berechnet für jeden Gitterpunkt den Stern c und berechnet damit punktweise den Laplace-Operator von den Koordinatenfunktionen ξ_i auf dem nicht äquidistanten Gitter in X. Der hier berechnete Stern ist konsistent von erster Ordnung, da man, wie bei der Shortley-Weller Approximation [13, Kap. 4.8.1], im Gegensatz zum äquidistanten, orthogonalen Fall nicht davon ausgehen kann, dass sich bei der Analyse mittels Taylorentwicklung die paarweise gegenüberliegenden (zum Beispiel rechts und links vom zentralen Punkt) Terme dritter Ordnung auslöschen.

Ergebnis

Mittels dieser Matlab-Routine wird nun überprüft, ob die Koordinatenfunktionen der Umkehrfunktionen der diskreten Lösungen $x_h = (x_h^1, x_h^2, x_h^3)$ des transformierten Lapalce-Systems (2.10) die Laplace-Gleichung erfüllen. Die Laplace-Gleichung wird dabei durch den oben berechneten Stern diskretisiert. Der zugehörige Differenzenoperator wird mit Δ_h bezeichnet. Es werden wieder die Fälle der einfachen und komplizierten Geometrie (in Abbildung 3.10 dargestellt) betrachtet. In Abbildung 3.21 sind die Normen der Fehler für ξ_h^1 und ξ_h^3 bei unterschiedlichen Maschenweiten dargestellt. Die Ergebnisse für die ξ_h^2-Koordinatenwerte werden nicht dargestellt. Hier sind die durch Interpolation erzeugten Werte von x_h^2, die die Funktion ξ_h^2 maßgeblich bestimmt, ausreichend, so dass die Betrachtung einer Glättung hier uninteressant

ist.
In keinem der beiden Geometriefällen zeichnet sich eine Konvergenz des Fehlers gegen Null ab, jedoch bleibt der Fehler beschränkt. Die verwendete Norm ist wieder $\|f\| = (\sum_{i=1}^{n} f_i^2/n)^{\frac{1}{2}}$ für $f \in \mathbb{R}^n$, wobei n wieder die Anzahl der Gitterpunkte beschreibt. Dieses Ergebnis steht im Einklang mit Satz 7. Dort wurde eine Abschätzung

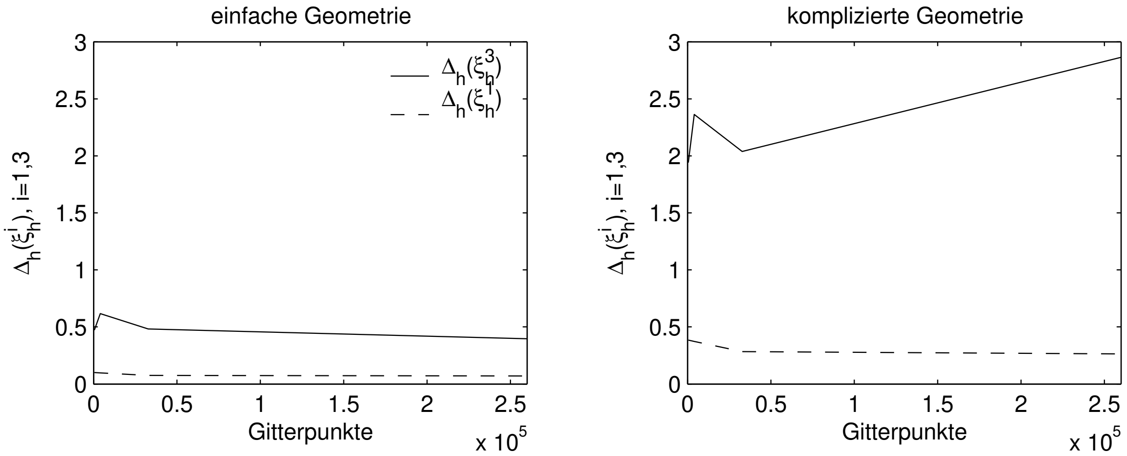

Abbildung 3.21: Norm des Fehlers von $\Delta_h(x_h)_i^{-1} = 0$ auf X für unterschiedliche Anzahlen von Gitterpunkten bei einfacher und komplizierter Geometrie.

für den Fehler der diskreten Lösungen von (2.9) beziehungsweise (2.18) in der H_2-Norm hergeleitet:

$$||R_h(x_l) - x_h^l||_{H_2} \leq c.$$

Für $l = 1, 2, 3$. Diese Abschätzung lässt sich benutzten, um den Fehler von $\Delta_h(x_h)_i^{-1} = 0$ in der H_0-Norm abzuschätzen.

Satz 8 *Für ξ_h, die inverse Abbildung der diskreten Lösung x_h von (2.9), gilt der Inversen der exakten Lösung ξ*

$$||\Delta_h \xi_h^l - \tilde{R}_h(\xi)||_{H_0} \leq c.$$

Dabei ist $\tilde{R}_h : \Xi^l \longrightarrow \Xi_h^l$ analog zu R_h in Satz 6, die Restriktion auf die Gitterfunktionen ξ_h^l.

Demnach konvergiert die Norm des Fehlers im Einklang mit den numerischen Ergebnissen für $h \to \infty$, nicht notwendigerweise gegen Null, bleibt aber beschränkt.
Beweis:
Die in Abschnitt 3.3.2 definierte H_0-Norm wird für vektor- und matrixwertige Abbildungen erweitert. Dies geschieht durch die Summierung der H_0-Normen der einzelnen Elemente. Im Fall der vektorwertigen Abbildung x_h lautet die verallgemeinerte

H_0-Norm dann

$$||x_h||_{\Sigma H_0} = \sum_{l=1}^{3} ||x_h^l||_{H_0}.$$

Laut Kettenregel gilt für $id_\xi = \xi \circ x$

$$\begin{aligned} \mathbb{1} &= \nabla(\xi \circ x) = \nabla\xi|_x \nabla x, \text{ also} \\ (\nabla x)^{-1} &= \nabla\xi|_x. \end{aligned} \tag{3.13}$$

Dabei bezeichnet ∇ das totale Differential, dessen entsprechender Differenzenoperator mit zentralen Differenzen im Folgenden mit D bezeichnet wird. Zweimalige Anwendung der Kettenregel ergibt

$$\begin{aligned} 0 &= \nabla^2(\xi \circ x) = \nabla(\nabla\xi|_x \nabla x) \\ &= \nabla^2\xi|_x \nabla x \nabla x + \nabla\xi|_x \nabla^2 x \text{ und damit} \\ \nabla^2\xi|_x &= -\nabla\xi|_x \nabla^2 x (\nabla x^{-1})^2. \end{aligned} \tag{3.14}$$

Mit der Restriktion auf den Raum der Gitterabbildungen nach Ξ, $\tilde{R}_h : \Xi \longrightarrow \Xi_h$ (vergleiche Definition von R_h in Satz 6) und der Inversen der kontinuierlichen Lösung von (2.9) ξ gilt

$$||\Delta_h \xi_h^l - \tilde{R}_h(\Delta\xi_l)||_{H_0} \leq ||D^2\xi_h - \tilde{R}_h(\nabla^2\xi)||_{\Sigma H_0}. \tag{3.15}$$

Um (3.15) abschätzen zu können, werden zuerst einige andere Abschätzungen durchgeführt. Zeige zuerst

$$||D\xi_h - \tilde{R}_h(\nabla\xi)||_{\Sigma H_0} \leq ch. \tag{3.16}$$

$$\begin{aligned} ||D\xi_h - \tilde{R}_h(\nabla\xi)||_{\Sigma H_0} &= ||Dx_h^{-1} - \tilde{R}_h(\nabla x^{-1})||_{\Sigma H_0} \\ &\leq ||Dx_h^{-1}||_{\Sigma H_0}\ ||\mathbb{1} - Dx_h R_h(\nabla x^{-1})||_{\Sigma H_0} \\ &\leq ||Dx_h^{-1}||_{\Sigma H_0} ||R_h(\nabla x^{-1})||_{\Sigma H_0}\ ||R_h(\nabla x) - Dx_h||_{\Sigma H_0} \\ &\overset{Satz 7}{\leq} ch\ ||Dx_h^{-1}||_{\Sigma H_0} ||R_h(\nabla x^{-1})||_{\Sigma H_0} \\ &\leq ch, \end{aligned}$$

denn x^{-1} nimmt auf dem Kompaktuum $\overline{\Xi}$ sein Maximum an, welches laut Satz 3 größer null ist.
Weiterhin gilt

$$\begin{aligned} \tilde{R}_h(D\xi)^{-1} D\xi_h &= \tilde{R}_h(\nabla\xi)^{-1}(D\xi_h - \tilde{R}_h(\nabla\xi) + \tilde{R}_h(\nabla(\xi)) \\ &= \tilde{R}_h(\nabla\xi)^{-1}(D\xi_h - \tilde{R}_h(\nabla\xi)) + \mathbb{1} \\ &\overset{(3.16)}{=} \mathbb{1} + O(h). \end{aligned} \tag{3.17}$$

Ebenso folgt

$$R_h(\nabla x))^2(Dx_h^{-1})^2 = \mathbb{1} + O(h). \tag{3.18}$$

Schließlich ergibt das Kompaktheitsargument von eben

$$||D(\xi)||_{\Sigma H_0} \leq c, \tag{3.19}$$

denn die stetige Funktion $||\nabla\xi||_\infty$ nimmt dem Kompaktuum $\overline{\Xi}$ sein Maximum an. Dasselbe Argument ergibt mit Satz 3

$$||(Dx^{-1})^2||_{\Sigma H_0} \leq c. \tag{3.20}$$

Damit kann nun (3.15) abgeschätzt werden:

$$\begin{aligned}
&||\tilde{R}_h(\nabla^2\xi) - D^2\xi_h||_{\Sigma H_0} = ||\tilde{R}_h(-\nabla\xi\nabla^2 x(\nabla x^{-1})^2) + D\xi_h D^2 x_h(Dx_h^{-1})^2||_{\Sigma H_0} \leq \\
&\leq || - \tilde{R}_h(\nabla\xi)\tilde{R}_h(\nabla^2 x)\tilde{R}_h(\nabla x^{-1})^2 + D\xi_h D^2 x_h (Dx_h^{-1})^2||_{\Sigma H_0} \\
&\leq ||D\xi_h||_{\Sigma H_0}||(Dx_h^{-1})^2||_{\Sigma H_0}||D\xi_h^{-1}\tilde{R}_h(\nabla\xi)\tilde{R}_h(\nabla^2 x)\tilde{R}_h(\nabla x^{-1})^2(Dx_h)^2 - D^2 x_h||_{\Sigma H_0} \\
&\overset{(3.19),(3.20)}{\leq} c||D\xi_h^{-1}\tilde{R}_h(\nabla\xi)\tilde{R}_h(\nabla^2 x)\tilde{R}_h(\nabla x^{-1})^2(Dx_h)^2 - D^2 x_h||_{\Sigma H_0} \\
&\overset{(3.17),(3.18)}{\leq} c||\tilde{R}_h(\nabla^2 x) - D^2 x_h||_{\Sigma H_0} \leq c \;\text{ wegen Satz (7).} \qquad \square
\end{aligned}$$

3.4 Parallelisierung

Mittels der Parallelisierung kann man die Gesamtrechenzeit für ein Problem dadurch verkürzen, dass man voneinander möglichst unabhängige Rechnungen gleichzeitig auf verschiedenen Prozessoren bearbeiten lässt. Außerdem ist es durch Parallelisierung möglich, wesentlich größere Probleme zu berechnen, falls die Daten auf mehrere Rechner verteilt werden. Für die Gittergenerierung bedeutet das, dass die Gittergröße des gesamten Gitters bei paralleler Berechnung nicht mehr durch die Größe des Arbeitsspeichers eines einzelnen Rechners beschränkt ist.
Es gibt verschiedene Strategien zur Parallelisierung. Diese werden nach Flynn [24] danach eingeteilt, ob der Befehlsfluss und der Datenfluss parallel abgearbeitet werden. Der hier verfolgte Ansatz ist ein MIMD-Typ (multiple instruction / multiple data stream) mit verteiltem Speicher. Es werden also sowohl der Befehlsstrom als auch der Datenstrom parallel abgearbeitet. Hierfür ist es nötig, Daten zwischen den beteiligten Prozessoren auszutauschen. Der Datenaustausch wird im Programm durch Funktionen der Bibliothek “MPI” (Message Passing Interface) gewährleistet. Siehe zur Benutzung von MPI in [37]. Einen Überblick über Parallelisierungsstrategien sowie eine Einführung in Parallelisierung findet man in [14].
Der Strömungslöser NSCL arbeitet blockweise parallel. Das heißt die kleinste Dateneinheit, die ein Prozessor bearbeiten kann, ist die zu einem Block gehörende

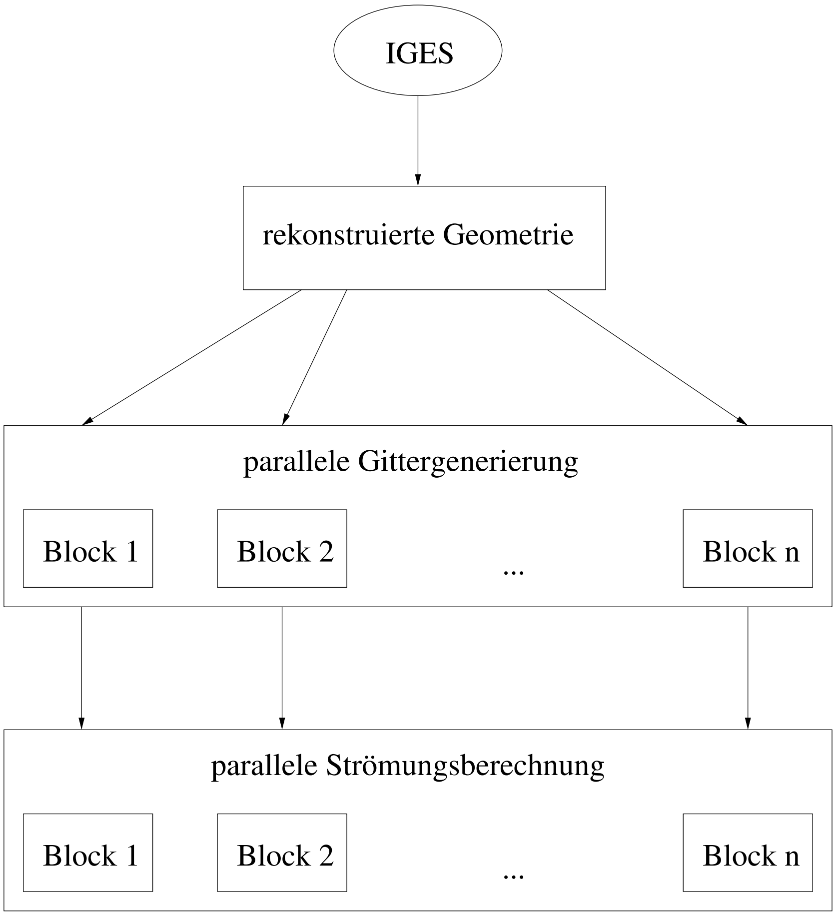

Abbildung 3.22: Flussdiagramm der parallelen Gittergenerierung und Stömungsberechnung. Nach der Rekonstruktion der Geometrie werden die Randdaten der einzelnen Gitter auf die verschiedenen Prozessoren verteilt, um dann parallel Gitter zu generieren und Strömung zu berechnen.

Datenmenge. Um effizient, das heißt vor allem ohne zusätzliche Kommunikation (Datenaustausch) mit NSCL zusammenarbeiten zu können, wurde die von NSCL vorgegebene Datenverteilung übernommen. Dadurch ist es möglich, in einem Block direkt nach Beendigung der Gittergenerierung mit der Strömungsberechnung zu beginnen, ohne dass die Gittergenerierung in anderen Blöcken abgeschlossen sein muss (vergleiche Abbildung 3.22). Der Datenfluss der parallelen Strömungssimulation ist demnach wie folgt: Die Rekonstruktion der Geometrie und die Generierung der Blöcke, das heißt die Generierung der Randpunkte der einzelnen Gitter, verläuft sequentiell. Anschließend werden die Randpunkte der Gitter blockweise auf mehre Prozessoren verteilt, wo parallel Gitter generiert werden. Im Anschluss an die Gittergenerierung wird ohne zusätzlichen Datenaustausch die parallele Strömungs-

berechnung gestartet.
Im Folgenden werden anhand eines Beispiels die zur Leistungsmessung der Parallelisierung üblichen Größen, Speedup

$$S(P) := \frac{T}{T(P)}$$

beziehungsweise parallele Effizienz,

$$E(P) := \frac{T}{P \cdot T(P)} = \frac{S(P)}{P}$$

bestimmt. Dabei bezeichnet P die Anzahl der eingesetzten Prozessoren und $T(P)$ die benötigte Gesamtrechenzeit bei Einsatz von P Prozessoren. T ist die benötigte Rechenzeit des sequentiellen Programms. Das zur Bestimmung von Speedup und paralleler Effizienz benutzte Beispiel ist in Kapitel 4.1 (Beipiel 1) näher beschrieben. Es wurden Gitter unterschiedlicher Größe auf 16 Blöcken berechnet. Dabei wurde zur Glättung der Gitter das Poisson-System (2.18) mit einem Abbruchkriterium von 10^{-8} gelöst. Die Ausführungszeiten für die Gittergenerierung auf einem Netzwerk von 16 Pentium III Xeon 933 Mhz Prozessoren sind in Tabelle 3.4 dargestellt. In Abbildung 3.23 sind die entsprechenden Kurven für Speedup und parallele Effizienz zu sehen.

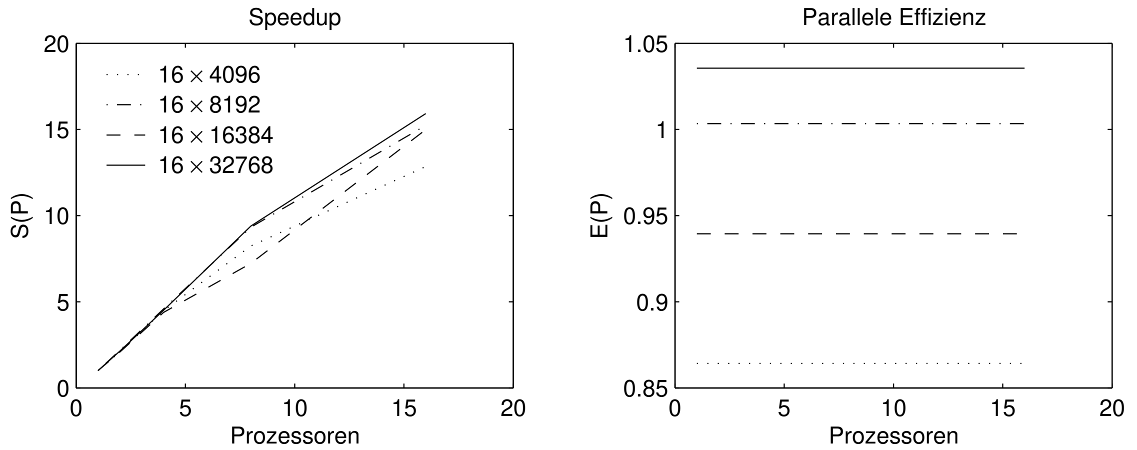

Abbildung 3.23: Speedup und parallele Effizienz für die Berechnung von 16 Gitterblöcken auf einem Netzwerk mit 16 Pentium III Xeon 933 Mhz Prozessoren.

Die parallele Effizienz liegt sehr nahe bei dem optimalen Wert von 1 und bei dem größten berechneten Problem sogar leicht darüber. Dieses Phänomen lässt sich auf Cache-Effekte bei der Berechnung größerer Datenmengen zurückführen. Die hohe

Zeit	Prozessoren				
Gitterpunkte	1	2	4	8	16
16×4096	220.801	105.518	49.212	26.798	17.179
16×8192	1125.494	517.1998	245.7728	120.691	73.834
16×16384	4765.730	2285.923	1088.578	659.725	317.885
16×32768	21444.434	10058.460	4758.583	2283.563	1346.996

Tabelle 3.4: Parallele Ausführungszeiten (in Sekunden) für die Berechnung von 16 Gitterblöcken auf einem Netzwerk mit 16 Pentium III Xeon 933 Mhz Prozessoren.

Effizienz der Parallelisierung insgesamt hängt damit zusammen, dass die Gittergenerierung in den einzelnen Blöcken unabhängig voneinander, das heißt ohne zusätzliche Kommunikation, stattfindet. Außerdem ist die zu Beginn zu versendende Datenmenge der Randdaten relativ gering im Verhältnis zu der Datenmenge eines gesamten Gitters, so dass sich Kommunikation praktisch nicht in einer Verlängerung der Rechenzeit niederschlägt.

Kapitel 4

Ergebnisse

Bis jetzt wurde die Gittergenerierung in dieser Arbeit als die Erfüllung "abstrakter Prinzipien" behandelt. Das heißt, es wurde die Generierung von Gittern besprochen, die möglichst glatt sind, keine sich schneidenden Gitterlinien haben und die Punkteverteilung, die auf dem Rand vorgegeben ist, im Inneren beibehalten. Die Erfüllung dieser Prinzipien durch Lösungen von Differentialgleichungen sowie die numerische Berechnung derselben wurde in den vorangehenden Kapiteln eingehend behandelt. In diesem Kapitel wird nun die Anwendbarkeit des implementierten Gittergenerierers beziehungsweise die Tauglichkeit der erzeugten Gitter im Zusammenspiel mit dem Strömungslöser NSCL gezeigt. Dazu werden erstens Gitter vorgestellt, die unter Vorgabe einer Blocktopologie aus gegebenen, CAD-erstellten, Geometrien generiert wurden und zweitens auf diesen Gittern berechnete Strömungen. Die hier vorgeführten Beispiele sowie weitere Gitter sind mit dem zu dieser Arbeit gehörendem Programm auf einfache Weise zu generieren. Eine Erklärung der dafür nötigen Eingabeparameter ist im Anhang B.1 zu finden.

4.1 Numerische Beispiele

Im Folgenden werden Beispiele für Gitter mit verschiedenen Blocktopologien gegeben. Die Topologie der zu generierenden Gitter ist auf die implementierten Möglichkeiten beschränkt. Das Programm ist auf weitere Topologiefälle erweiterbar. Hierbei kann auf bestehende Funktionen zurückgegriffen werden, so dass eine Erweiterung der implementierten Fälle in kurzer Zeit möglich ist. Die Flexibilität des Programms ist durch die Handhabbarkeit unterschiedlicher Geometrien für jeden Topologiefall gegeben. Alle hier verwendeten Geometrien wurden mit Pro/Engineer [31] erstellt und als IGES-Datei (siehe Kapitel 2.6) exportiert, um dann als Input für den Gittergenerierer zu dienen.

Beispiel 1 Durchströmung einer Landschaft

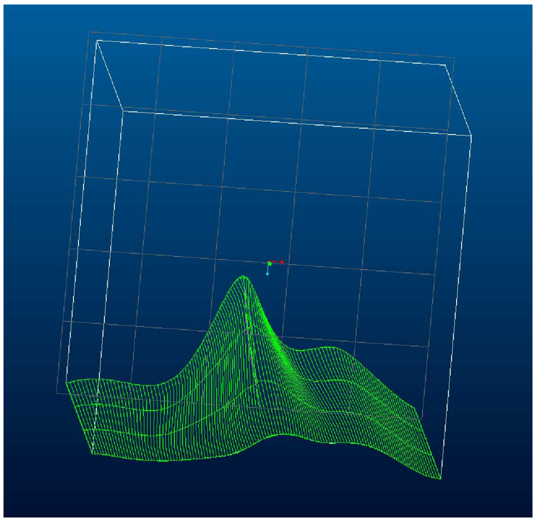

Abbildung 4.1: CAD-Modell einer Landschaft.

In diesem ersten einfachen Beispiel ist das Simulationsgebiet das Innere der in Abbildung 4.1 dargestellten Geometrie. Das blockstrukturierte Gitter besteht aus 16 hintereinanderliegenden Gittern, die alle gleich orientiert sind. Diese Blocktopologie unterscheidet sich nicht wesentlich von einer Ein-Block-Topologie, da hier durch die Blockunterteilung kein Einfluss auf den Verlauf der Gitterlinien genommen wird. Die Motivation für die hier gewählte Blockaufteilung ist die Verteilung der Daten auf 16 Blöcke, um effizient parallel rechnen zu können. Eine zusätzliche Aufteilung in einer anderen Richtung wurde nicht vorgenommen, da hier durch die entstehenden Blockgrenzen die Glättung des Gitters stark eingeschränkt wäre. Für die Strömungsberechnung wurden 16 Gitter mit jeweils $42 \times 4 \times 28$ Gitterpunkten und einer leichten Gradierung in z-Richtung generiert (siehe Abbildung 4.2). Dabei wurde das Poisson-System (2.18) mit einem Abbruchkriterium von 10^{-8} zur Gittergenerierung benutzt. Wie in Abbildung 4.2 zu erkennen ist, besitzt das Gitter die in Kapitel 2.4 besprochenen Eigenschaften: Die Gradierung, also nicht äquidistante Punkteverteilung des Randes, wird im Innern beibehalten und es treten die bei "elliptischen" Gittern zu beobachtenden Effekte am Rand auf: Am konvexen Teil des Randes verlaufen die Gitterlinien weiter im Innern und am konkaven Rand näher am Rand.
Das Simulationsgebiet hat in dimensionslosen Größen eine Breite und eine Tiefe von jeweils 1. Am linken Gebietsrand sind Einströmbedingungen in Normalenrichtung mit einer Geschwindigkeit von 1 vorgegeben. Am rechten Rand sind Ausströmbedingungen gesetzt und am restlichen Rand sind Haftbedingungen vorgegeben. Die

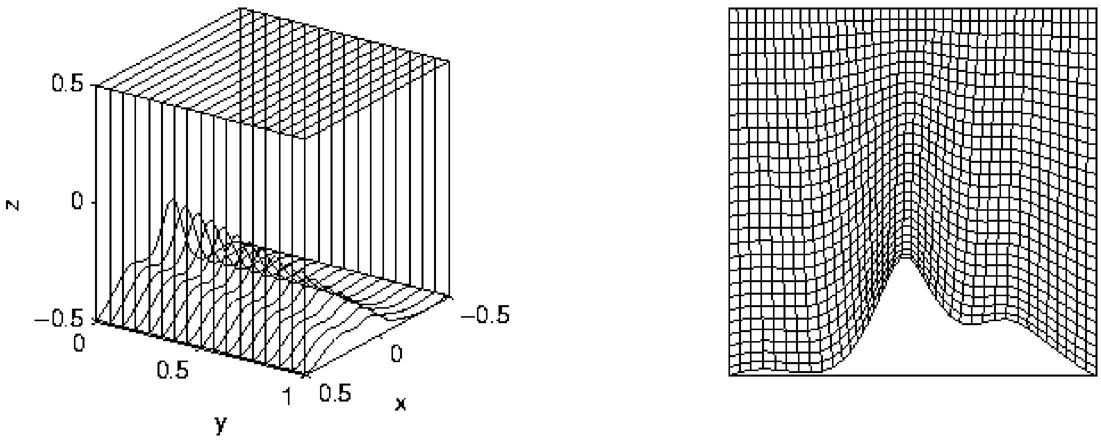

Abbildung 4.2: Die Blockstruktur des Gitters für die Landschaft und ein Querschnitt durch eines der 16 Teilgitter.

Reynoldszahl ist 1000.
In Abbildung 4.3 ist eine Visualisierung der von NSCL berechneten Strömung zum Zeitpunkt 0.5 zu sehen. Sie zeigt die Stromlinien im Gebiet eingestreuter Partikel und den Druck auf einer Schnittebene im hinteren Teil des Gebiets. Dabei steht Gelb für niedrigen und Blau für hohen Druck. Die Größe der Pfeilköpfe korrespondiert mit der Geschwindigkeit. Man erkennt deutlich eine Wirbelbildung hinter der Bergkuppe. Diese geht einher mit einem Sog, der sich hier herausbildet und als gelber Fleck zu erkennen ist.

Beispiel 2 Ein H-Gitter um eine symmetrische Tragfläche

In diesem Beispiel wird ein H-Gitter um eine symmetrische Tragfläche mit spitzem Profil (vergleiche Abbildung 4.4) generiert. Ausschlaggebend für die Wahl dieser Blocktopologie sind die scharfen Kanten der Geometrie. Sie ermöglicht einen glatteren Verlauf der Gitterlinien, als es zum Beispiel bei einem O-Gitter der Fall wäre.
Die einzelnen Gitter haben jeweils eine Größe von $32 \times 32 \times 32$ Gitterpunkten und wurden mit dem Poisson-System (2.18) und einem Abbruchkriterium von 10^{-8} berechnet. Das H-Gitter besteht aus acht einzelnen Gittern, aufgeteilt in jeweils eine Schicht à vier Blöcken oberhalb und eine Schicht unterhalb des Objekts (vergleiche Abbildungen 4.5 und 4.6). Alle Gitter sind in z-Richtung gradiert. Die oberen Blöcke in Abbildung 4.6, Blöcke 0 – 3, haben am unteren Rand dichter liegende Gitterlinien, und die unteren Blöcke 4 – 7 haben oben dichtere Gitterlinien. Des Weiteren sind die Gitter der Blöcke 0,2,4 und 6, also die vor und hinter dem zu umströmenden Objekt liegenden Gitter, in x-Richtung gradiert. Die links neben dem Objekt liegenden Blöcke 0 und 4 haben rechts dichtere Gitterlinien, und die rechts vom

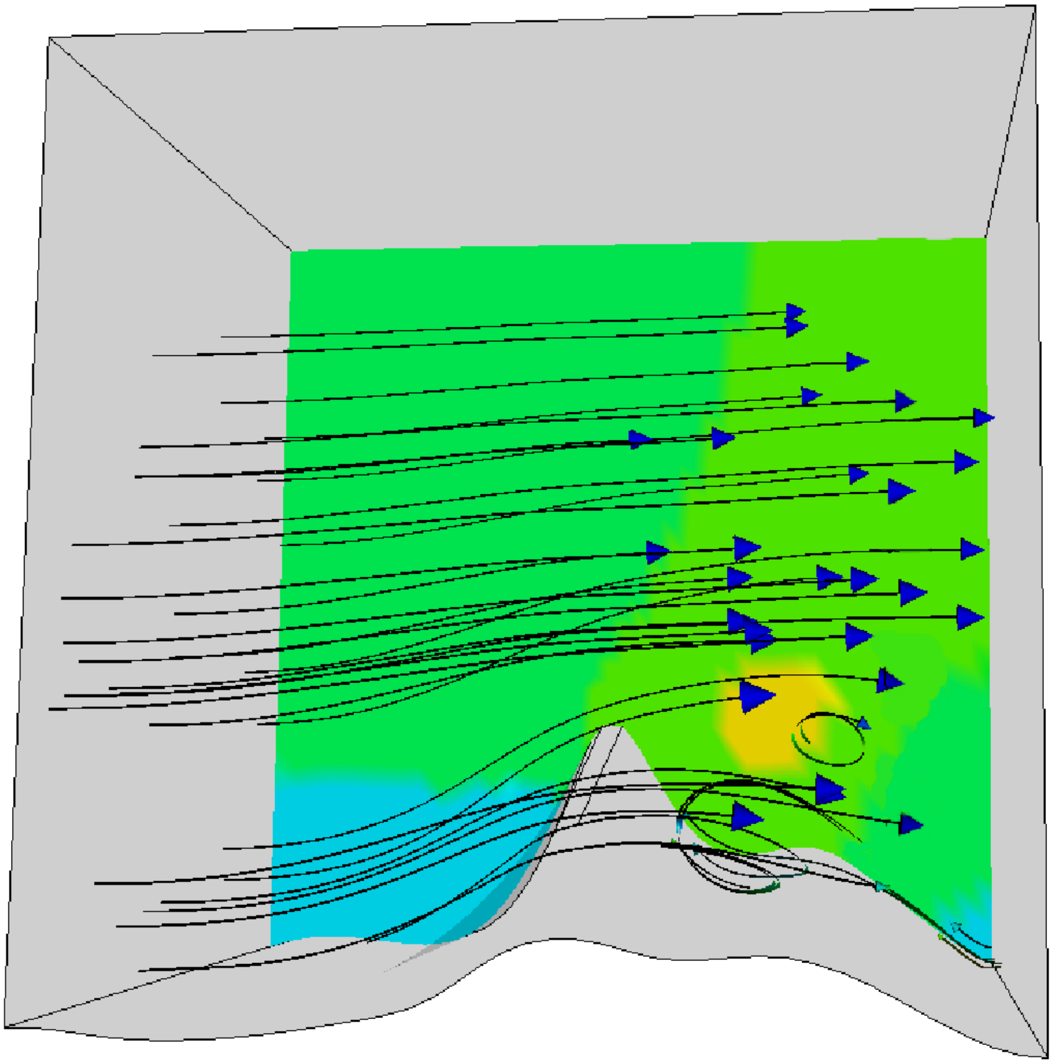

Abbildung 4.3: Stromlinien und Druck der auf dem Gitter berechneten Strömung durch die Landschaft.

Objekt liegenden Blöcke 2 und 6 haben auf der linken Seite dichtere Gitterlinien. Die Gradierung der Gitter dient der höheren Auflösung von Teilgebieten, in denen die Lösung, also die Strömung, stärker variiert als in anderen Gebietsteilen.
Das Simulationsgebiet hat die Abmessungen $3.2 \times 1.0 \times 1.3$ (Breite $\times$ Tiefe $\times$ Höhe). Die Tragfläche ist eine Längeneinheit tief, also genauso tief wie das Simulationsgebiet. Nach vorne hin wird sie schmaler und flacher. Die Breite beträgt 0.41 am hinteren Ende und 0.29 am vorderen Ende. Die maximale Höhe der Tragfläche variiert zwischen 0.16 (hinten) und 0.07 (vorne). Die Position der Tragfläche innerhalb

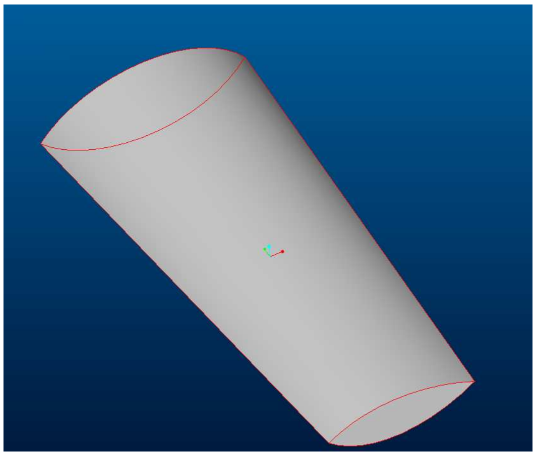

Abbildung 4.4: CAD-Modell der symmetrischen Tragfläche.

des Gebiets ist Abbildung 4.5 zu entnehmen. Am linken Rand sind Einströmbedingungen in Normalenrichtung mit Geschwindigkeit 1 gesetzt. Der rechte Rand ist Ausströmrand und am restlichen Rand gelten Haftbedingungen. Die Reynoldszahl ist 1000.
Die mit NSCL berechnete Strömung zum Zeitpunkt 3.5 ist in den Abbildungen 4.7 und 4.8 dargestellt. Zu sehen sind der Druck auf einer Schnittfläche und Stromlinien von eingestreuten Partikeln. Bei Druck und Geschwindigkeit steht Blau für großen Druck / hohe Geschwindigkeit und Rot für niedrigen Druck / geringe Geschwindigkeit. Die Partikel wurden in zwei Bereichen eingestreut, einmal vor der Tragfläche und einmal direkt hinter der Tragfläche. Die vor der Tragfläche beginnenden Stromlinien verlaufen alle laminar. Bei den Stromlinien hinter der Tragfläche sind Verwirbelungen sichtbar. Außerdem ist zu erkennen, dass der dickere Bereich des Flügels in der Nachströmung einen größeren Sog erzeugt, so dass sich die Strömung dorthin bewegt.

Beispiel 3 Ein O-Gitter um eine Tragfläche

Die in diesem Beispiel zu umströmende Tragfläche hat ein rundliches Profil ohne scharfe Kanten (siehe Abbildung 4.9). Für dieses Profil eignen sich O-Gitter, da hierbei die Gitterlinien O-förmig, entsprechend der rundlichen Gestalt der Geometrie, um das Objekt herumgeführt werden können.
Das O-Gitter besteht aus sechs einzelnen Gittern. Diese haben eine Auflösung von jeweils $32 \times 12 \times 32$ Gitterpunkten und wurden mit dem Poisson-System (2.18) und einem Abbruchkriterium von 10^{-8} berechnet. Die sechs Gitter sind angeordnet, wie es in Abbildung 4.10 zu erkennen ist: Vier Blöcke umgeben die Tragfläche als eigent-

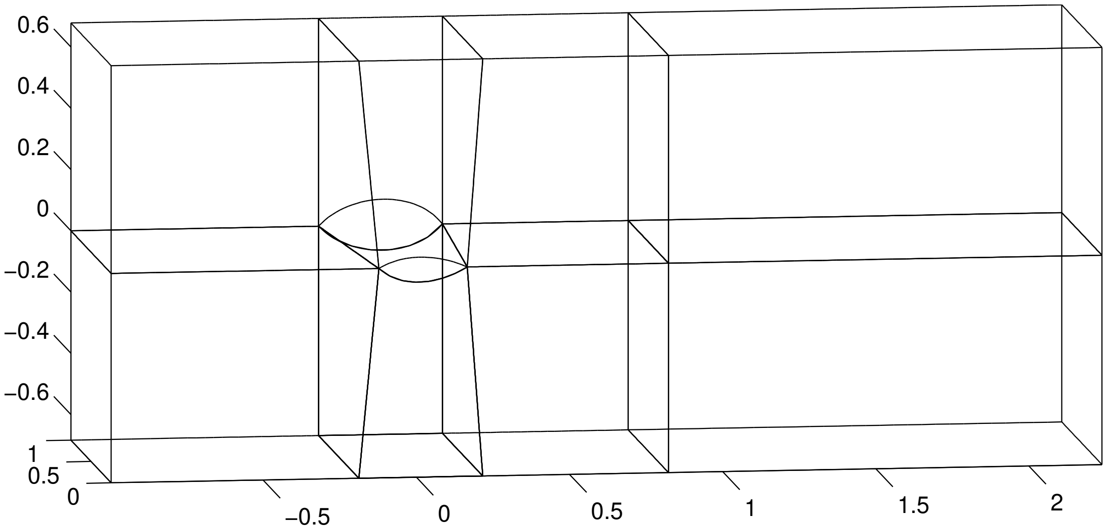

Abbildung 4.5: Die Blockstruktur des H-Gitters um die symmetrische Tragfläche.

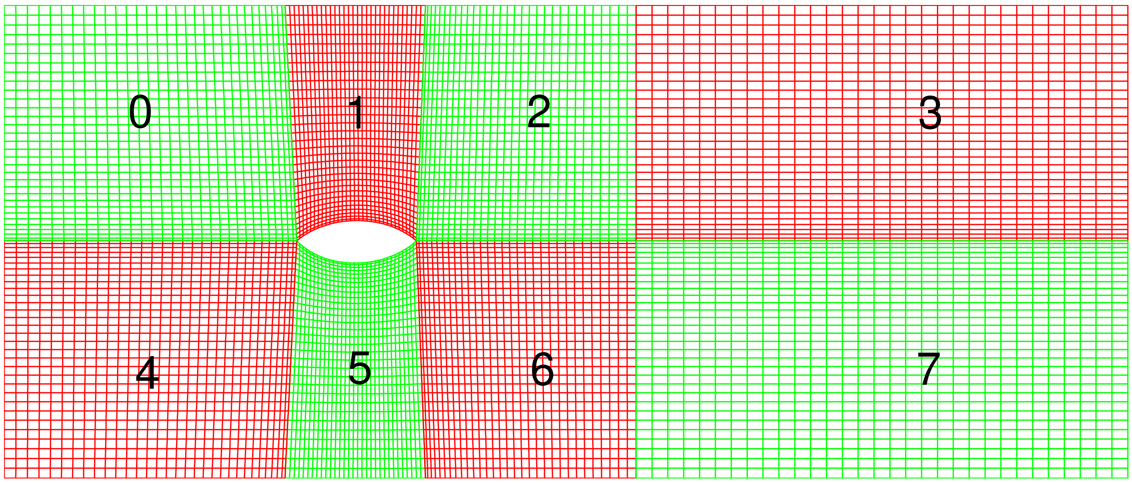

Abbildung 4.6: Querschnitt des gradierten H-Gitters um die symmetrische Tragfläche.

liches O-Gitter und rechts sowie links ist jeweils ein weiterer Block angefügt. Die Abmessungen des Simulationsgebietes sind $0.54 \times 0.2 \times 0.25$. Dabei stimmt die Tiefe des Simulationsgebiets wieder mit der Tiefe der Tragfläche überein. Die Tragfläche ist wie im vorherigen Beispiel vorne flacher und schmaler als hinten. Die maximale Höhe variiert zwischen 0.01 hinten und 0.015 vorne und die maximale Breite zwischen 0.07 und 0.05. Die Randbedingungen sind wie in den vorangegangenen Beispielen mit Einströmbedingungen der Geschwindigkeit 1 in Normalenrichtung auf

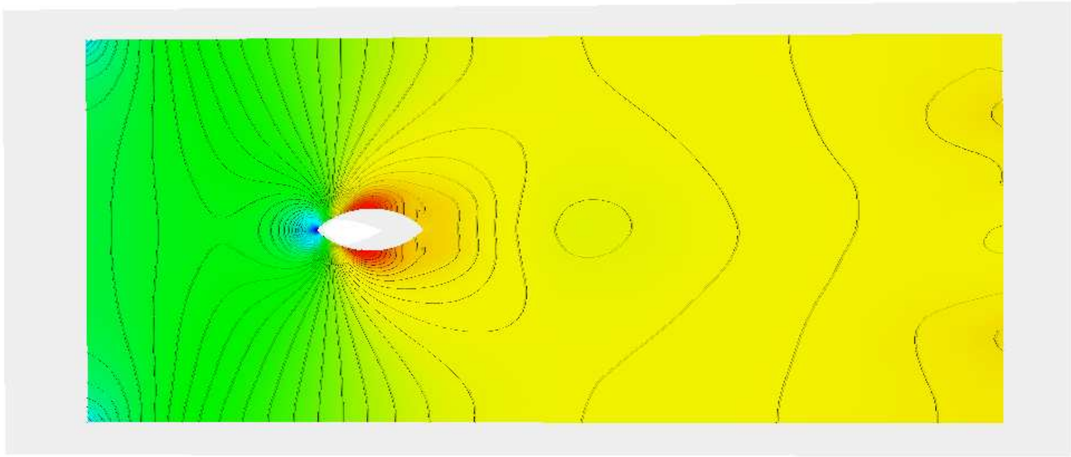

Abbildung 4.7: Druckverteilung und Isodrucklinen der berechneten Strömung um die symmetrische Tragfläche auf einer Schnittfläche aus der Seitenperspektive.

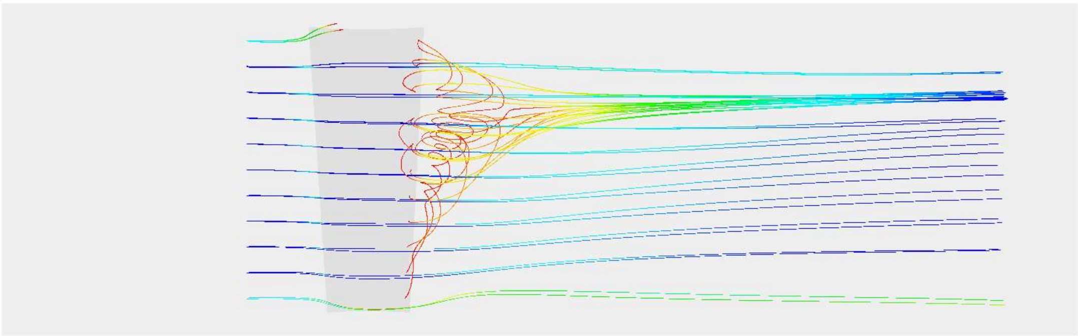

Abbildung 4.8: Stromlinien der berechneten Strömung um die symmetrische Tragfläche aus der Vogelperspektive.

der linken Seite, Ausströmbedingungen rechts und Haftbedingungen am restlichen Rand. Die Reynoldszahl ist wieder 1000.

Abbildung 4.12 zeigt die Druckverteilung der berechneten Strömung zum Zeitpunkt 0.5. Wieder ist der Druck vor der Tragfläche deutlich höher als dahinter. Außerdem ergibt sich bei dieser Tragfläche im Gegensatz zur Tragfläche mit symmetrischem Profil ein Auftrieb. Dieser ergibt sich durch den oberhalb der Tragfläche entstehenden Sog. Im Bereich der Blockgrenzen ergeben sich deutlich zu erkennende Artefakte, die auf Interpolationsfehler der von NSCL verwendeten Diskretisierung

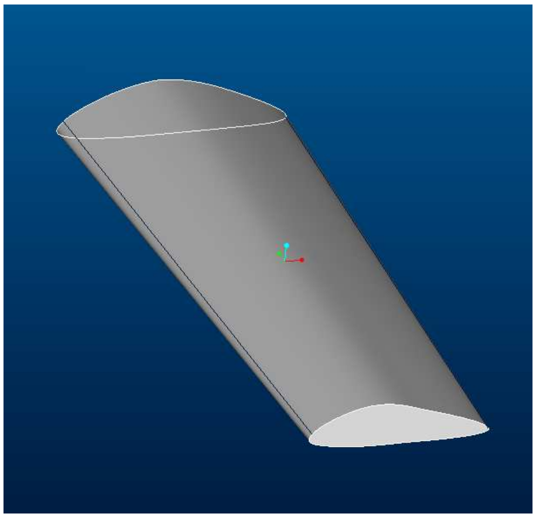

Abbildung 4.9: CAD-Modell der Tragfläche mit rundlichem Profil.

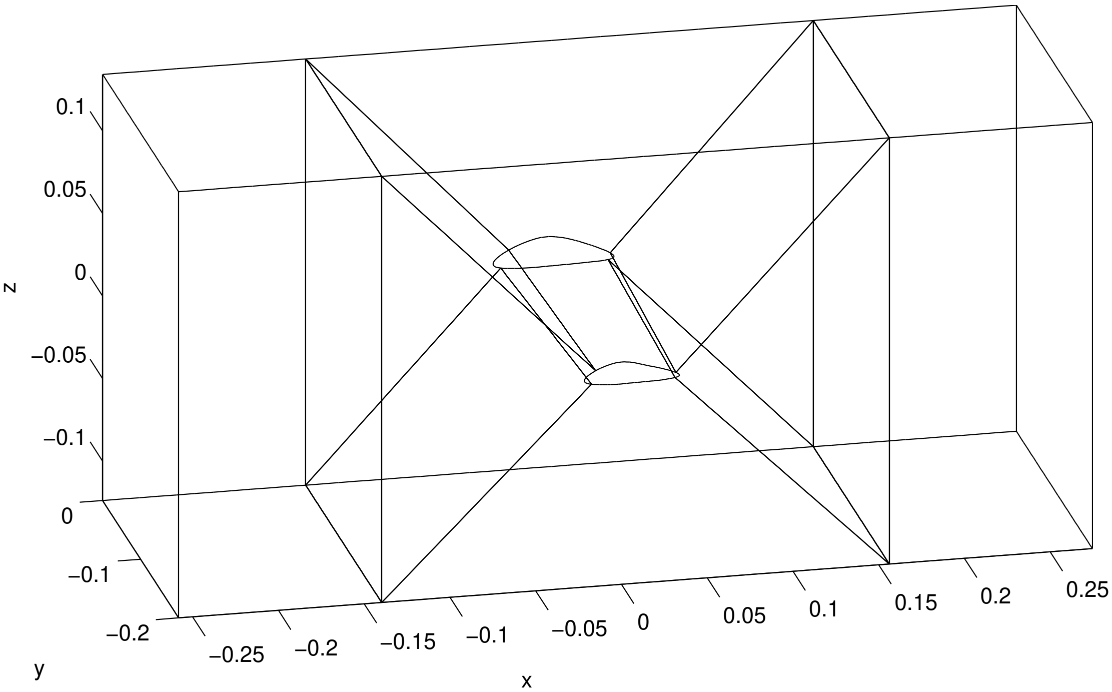

Abbildung 4.10: Blockstruktur des O-Gitters.

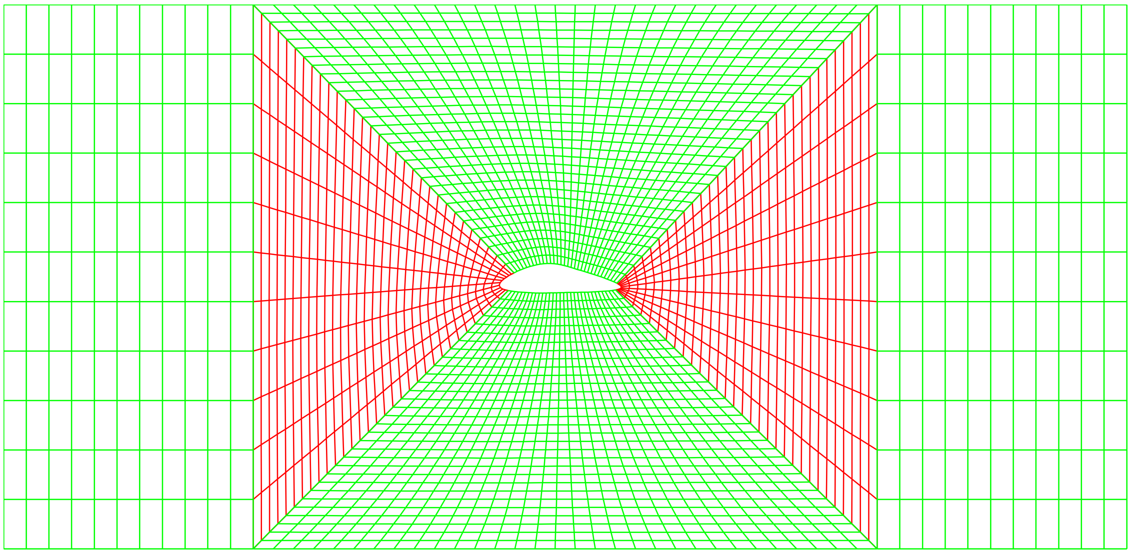

Abbildung 4.11: Querschnitt eines O-Gitters um eine Tragfläche.

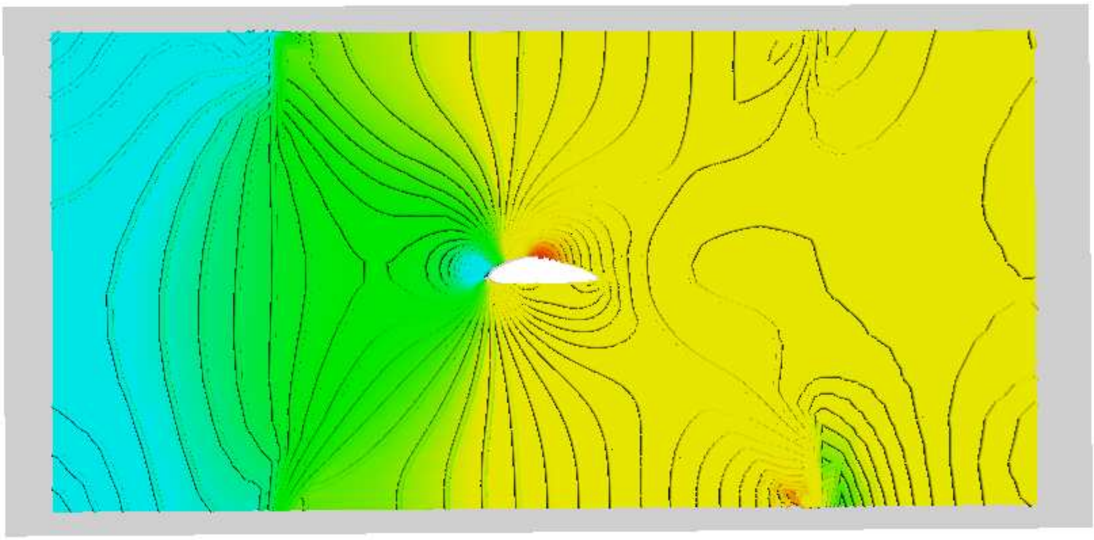

Abbildung 4.12: Druckverteilung und Isodrucklinien für die auf einem O-Gitter berechnete Strömung um eine Tragfläche aus der Seitenperspektive.

zurückzuführen sind. Hierauf wird in Beispiel 5 noch einmal eingegangen.

Beispiel 4 Eine 22-Block-Topologie zur Umströmung einer Tragflächenspitze

Mit der in diesem Beispiel vorgestellten Blocktopologie ist es möglich, auch mit den durch NSCL gegebenen Restriktionen an die Verbindungsmöglichkeiten unterschiedlicher Blöcke (siehe Kapitel 3.1.2) Geometrien zu umströmen, die in das Simulationsgebiet hineinragen. Hier wird als Beispiel ein Tragflächenende betrachtet. Andere mögliche Anwendungen sind die Umströmung von Antennen, Turmspitzen oder Ähnlichem. Als Tragfläche für dieses Beispiel wurde eine ähnliche Geometrie wie für das O-Gitter benutzt, siehe Abbildung 4.13. Die Blockstruktur ist Abbil-

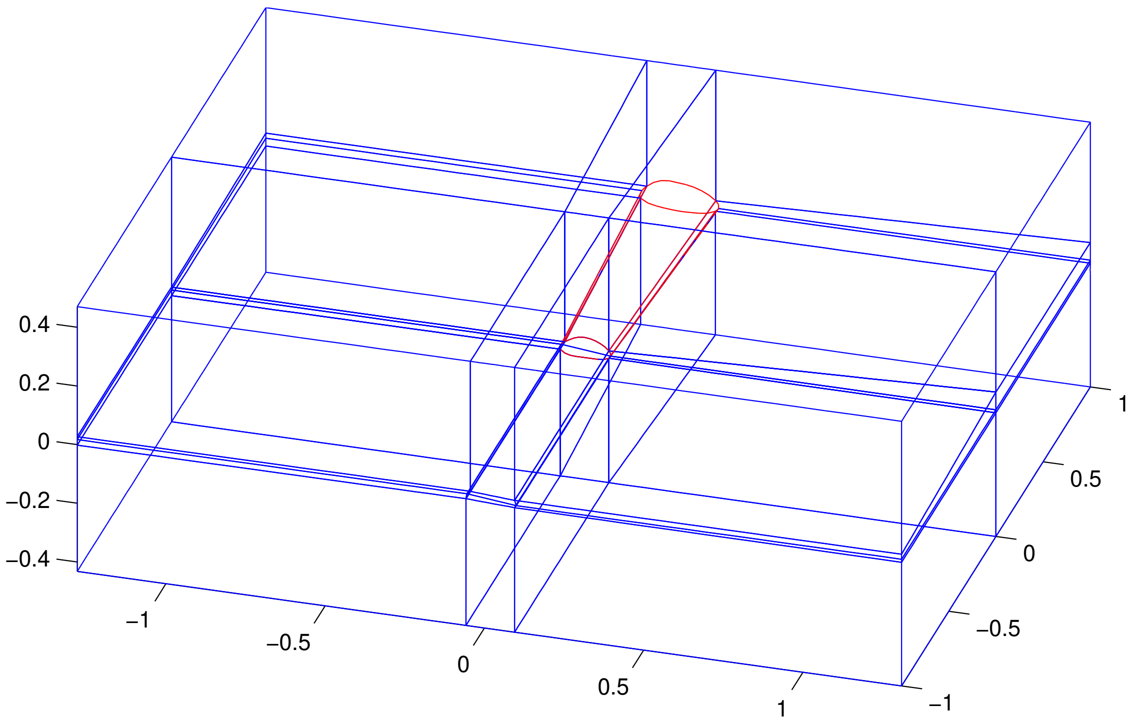

Abbildung 4.13: Die Blockstruktur des 22-Block-Gitters für ein Tragflächenende.

dung 4.13 zu entnehmen. Das Gitter besteht aus zwei Schichten von Gittern: Eine Schicht, die die Tragfläche umgibt, so wie vorher das H- beziehungsweise O-Gitter (siehe Abbildung 4.14), und eine Schicht davor, die den Rest des Gebietes diskretisiert (siehe Abbildung 4.15). Die 22 Blöcke sind alle gleich orientiert. Die Struktur lässt sich durch zwei Schichten kartesisch angeordneter Blöcke, mit jeweils zwölf Blöcken angeordnet in drei Spalten und vier Zeilen, beschreiben. Dabei sind in der hinteren Schicht, dort wo sich die Geometrie befindet, zwei Blöcke entfernt. (Dieser Bereich ist in Abbildung 4.13 rot gekennzeichnet). Die Gradierung der Gitter ist ähnlich wie bei dem H-Gitter aus Beispiel 2. Das heißt, die Gitter ober- und unter-

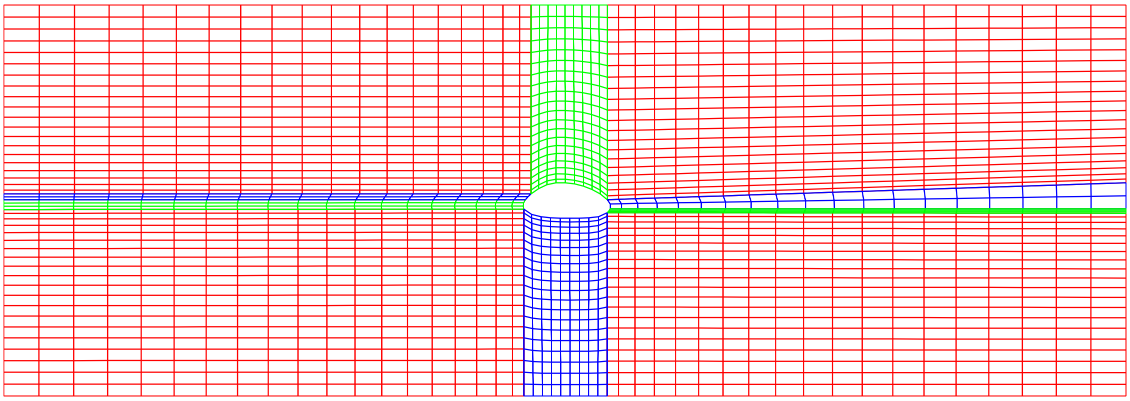

Abbildung 4.14: Querschnitt durch die hintere Schicht der 22 Blöcke (mit der Tragfläche in der Mitte).

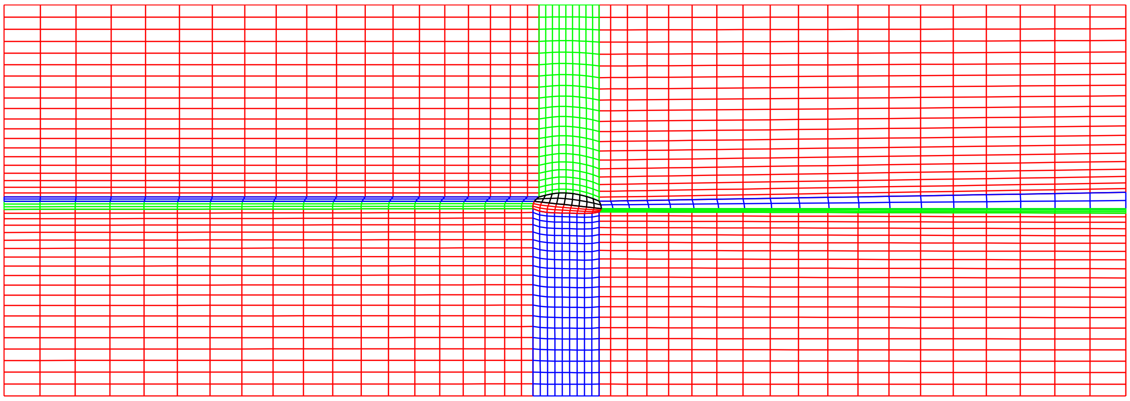

Abbildung 4.15: Querschnitt durch die vordere Schicht der 22 Blöcke (mit zwei Blöcken mehr als die hintere Schicht).

halb der Geometrie haben an den geometrienahen Rändern unten beziehungsweise oben dichtere Gitterlinien in z-Richtung. Die Blöcke in der mittleren Ebene (Zeilen 2 und 3) sind in z-Richtung nicht gradiert. Die Blöcke rechts und links von der Geometrie (Spalten 1 und 3) sind alle in x-Richtung gradiert, so dass auch hier der geometrienahe Bereich höher aufgelöst wird.
Die Abmessungen des Gebiets sind $1.7 \times 2 \times 0.87$. Die Berechnung der Strömung wurde mit einer Einströmgeschwindigkeit 0.5 und einer Reynoldszahl von 100 bei ansonsten gleichen Randbedingungen wie in den anderen Beispielen durchgeführt. In Abbildung 4.16 ist die berechnete Strömung zum Zeitpunkt 2.5 zu erkennen. Es sind sowohl Stromlinien vor und hinter der Tragfläche eingestreuter Partikel, als auch die Druckverteilung auf einer horizontalen Schnittfläche zu sehen. Bedingt durch die geringere Einströmgeschwindigkeit und die deutlich kleinere Reynoldszahl als in den

anderen Beispielen, bilden sich hier keine Wirbel. Die Strömung wird erwartungsgemäß durch die Tragfläche abgebremst beziehungsweise umgeleitet, so dass die Strömung direkt hinter dem Objekt am langsamsten verläuft. Am Tragflächenende konnte das Gebiet aufgrund sich ergebender unakzeptabel kleiner Zeitschrittweiten des Lösers nicht hoch genug aufgelöst werden, so dass in diesem Bereich Artefakte erkennbar sind.

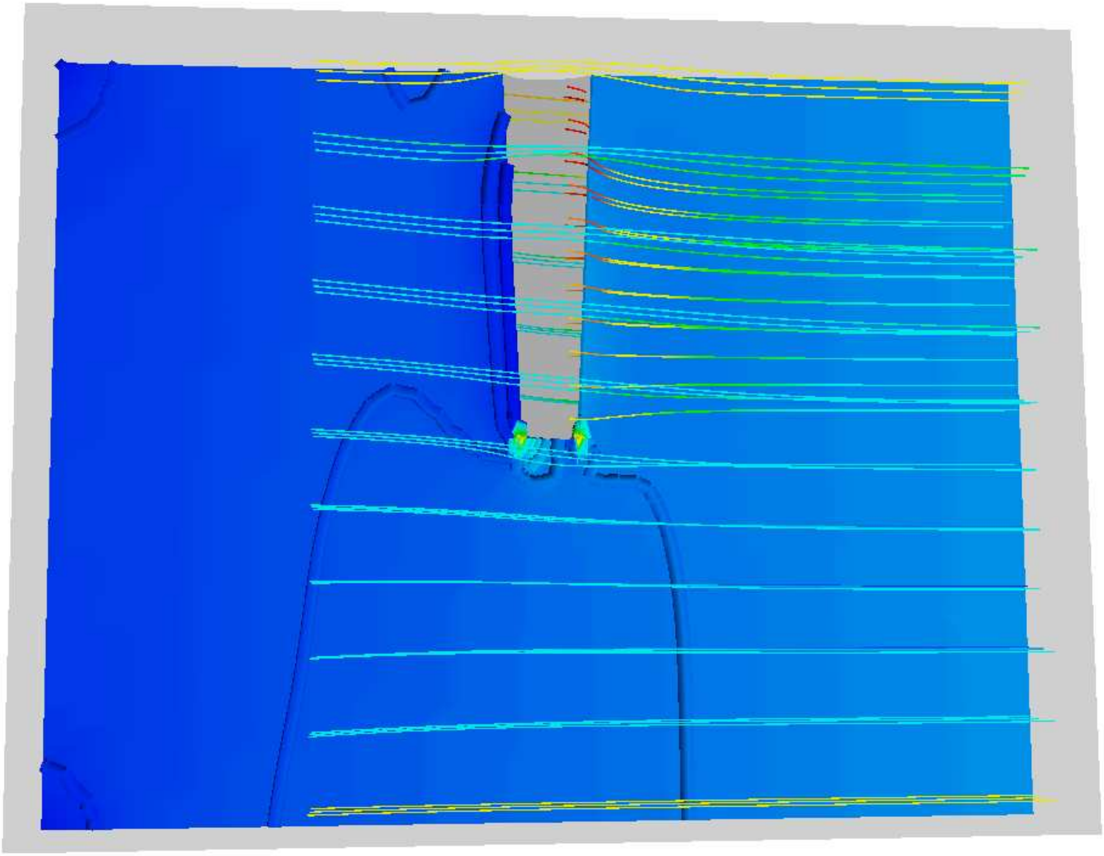

Abbildung 4.16: Druckverteilung und Stromlinien der berechneten Strömung um ein Tragflächenende aus der Vogelperspektive.

Beispiel 5 Variation der Geometrie und Glättung der Blockgrenzen

Dieses Beispiel zeigt dieselbe Blocktopologie wie das Beispiel 3 mit einer anderen Geometrie. Damit wird gezeigt, dass diese Blocktopologie nicht nur für Tragflächen geeignet ist, sondern auch für andere, ähnliche Geometrien, wie Baumstümpfe, Türme, etc. Der Phantasie sind hier keine Grenzen gesetzt. Außerdem wurden für

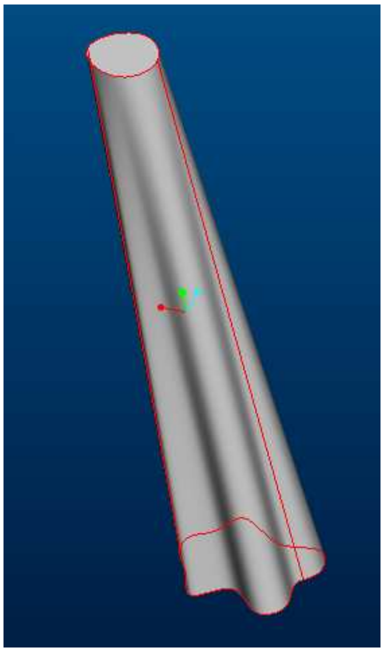

Abbildung 4.17: CAD-Modell eines Baumstamms.

diese Geometrie Strömungen auf unterschiedlichen Gittern berechnet. Erstens wurden geglättete und ungeglättete, also nur durch Transfinite Interpolation erzeugte Gitter, benutzt. Zweitens wurde für beide Gitterarten jeweils ein Gitter mit und ohne der in Kapitel 3.1.4 besprochenen Glättung der Blockgrenzen generiert. Insgesamt wurden für dieses Beispiel also Stömungen auf vier unterschiedlichen Gittern berechnet. Die veschiedenen Gitter sind in den Abbildungen 4.18 bis 4.21 zu sehen. Die einzelnen Teilgitter haben jeweils die Auflösung $16 \times 16 \times 16$. Die Gitter des O-Gitters sind in Richtung der inneren Gebietsgrenze gradiert. Für die Glättung der Gitter wurde ein Abbruchkriterium von 10^{-8} benutzt.
Das Simulationsgebiet hat, wie gesagt, dieselbe Blockstruktur wie in Beispiel 3 und dabei die Abmessungen $2 \times 1 \times 1$. Die Randbedingungen mit einer Einströmgeschwindigkeit von 1 und einer Reynoldszahl von 100 sind ebenfalls dieselben wie in Beispiel 3. In den Abbildungen 4.22 bis 4.25 ist der Druck mit Isodrucklinien zum Zeitpunkt 2.5 auf Schnittebenen in der Seitenansicht dargestellt. Die Struktur der Druckverteilungen ist für alle berechneten Strömungen ähnlich: Am Einströmrand sowie direkt vor dem Objekt ist der Druck besonders hoch und in Strömungsrichtung hinter dem Objekt ist der Druck niedriger. Soweit stimmen alle Ergebnisse mit der intuitiven Erwartung überein. Bei genauerer Betrachtung entdeckt man in allen vier Strömungen Stellen mit nichtintuitiver Druckverteilung. Diese Stellen treten in vier Bereichen, jeweils am oberen beziehungsweise unteren Gebietsrand, eine

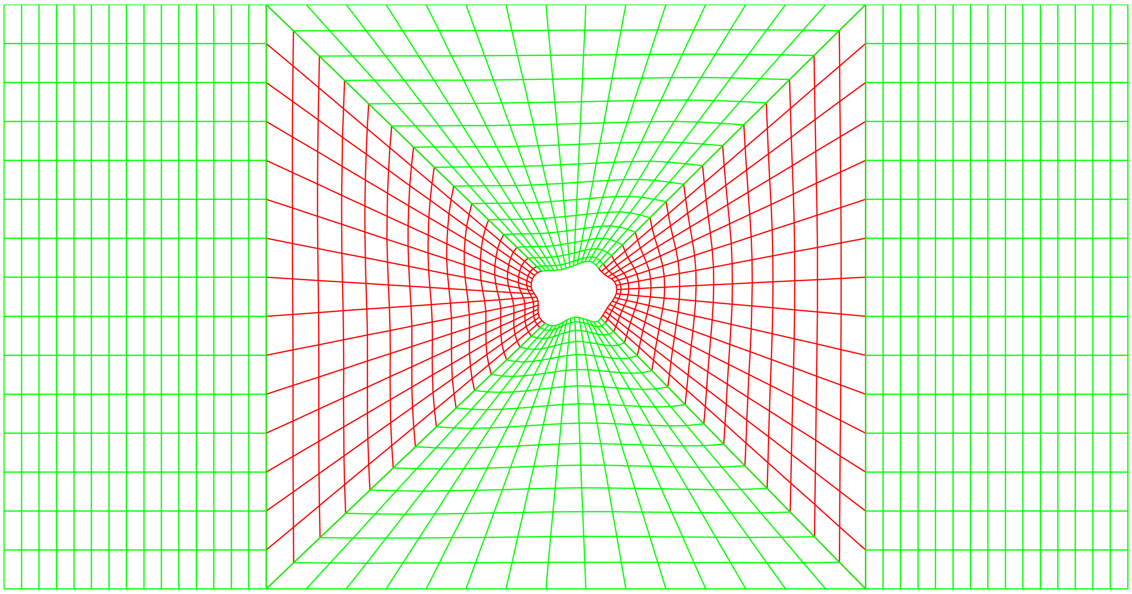

Abbildung 4.18: Nicht geglättetes O-Gitter um einen Baumstamm ohne Glättung der Blockgrenzen.

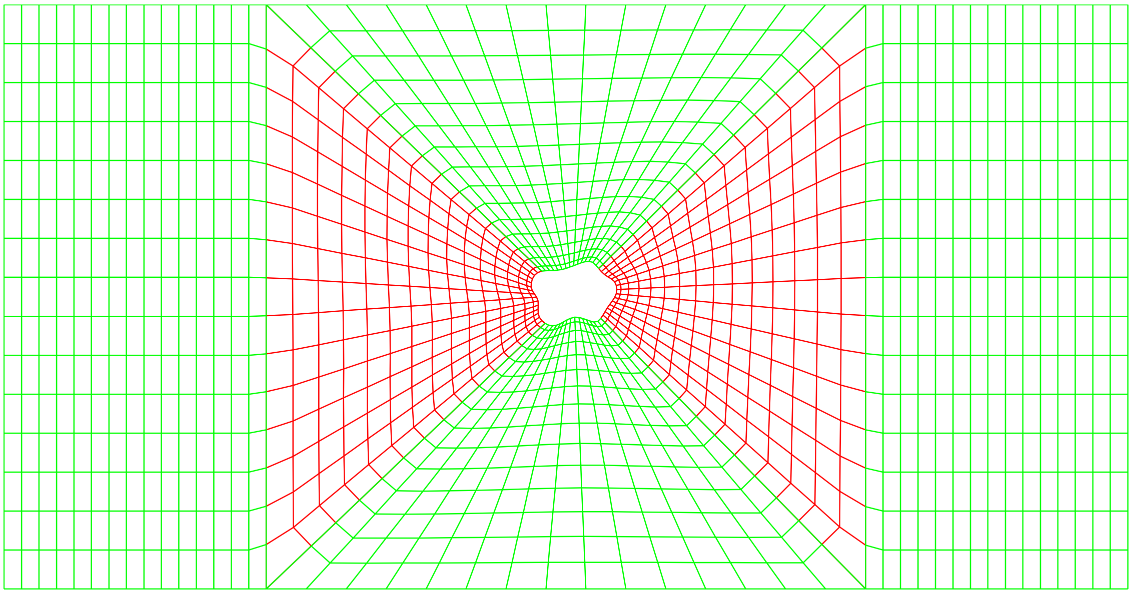

Abbildung 4.19: Nicht geglättetes O-Gitter um einen Baumstamm mit Glättung der Blockgrenzen.

halbe Längeneinheit vor und hinter dem Objekt, auf. An diesen Stellen befinden sich auch Blockgrenzen. Wie schon in Beispiel 3 erwähnt, sind diese Effekte auf In-

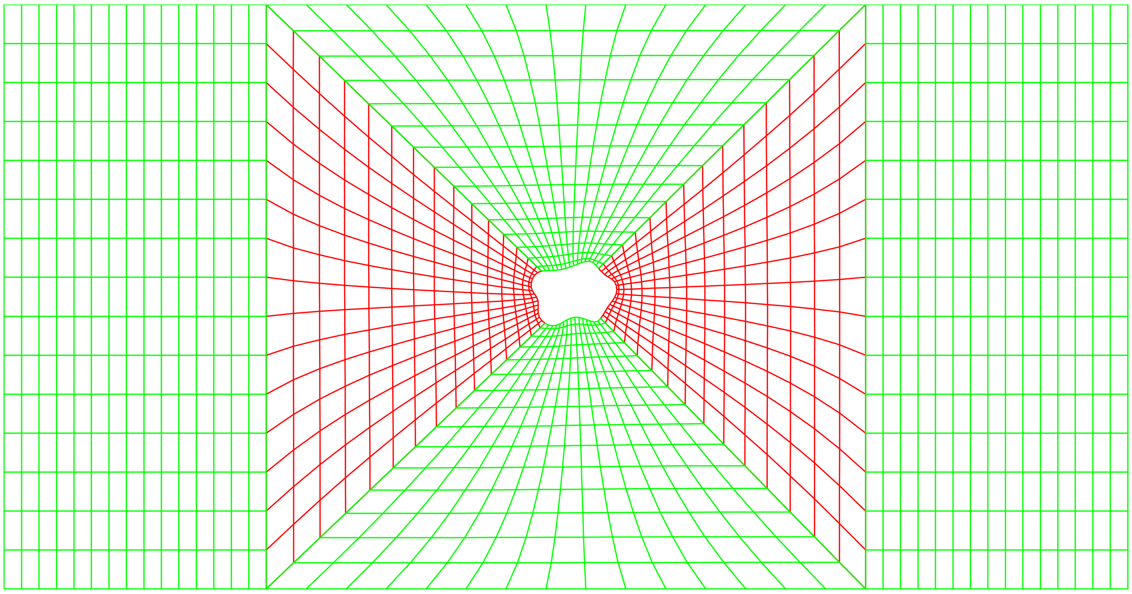

Abbildung 4.20: Geglättetes O-Gitter um einen Baumstamm ohne Glättung der Blockgrenzen.

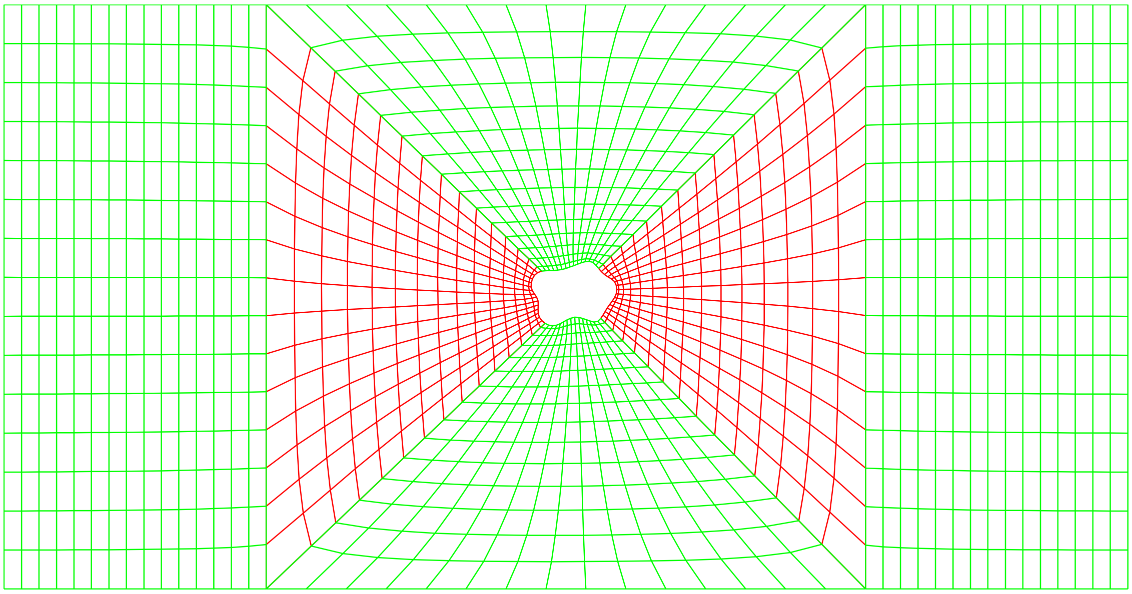

Abbildung 4.21: Geglättetes O-Gitter um einen Baumstamm mit Glättung der Blockgrenzen.

terpolationsfehler der von NSCL verwendeten Diskretisierung zurückzuführen. An den Gebietsrändern sind die sich durch die Blocktopologie ergebenden Winkel von

Randzellen besonders klein (vergleiche Abbildungen 4.18 und 4.20), so dass der entstehende Interpolationsfehler hier besonders groß ist. Sowohl bei dem geglätteten als auch bei dem ungeglätteten Gitter tritt das beschriebene Phänomen abgeschwächt auf, wenn die Blockgrenzen geglättet sind. Das heißt, die in Kapitel 3.1.4 Glättung der Blockgrenzen wirkt sich positiv auf die berechnete Lösung aus. Insgesamt intuitivere Ergebnisse liefern hier die mit den nur durch Transfinite Interpolation erzeugten, also ungeglätteten Gittern, berechneten Strömungen.

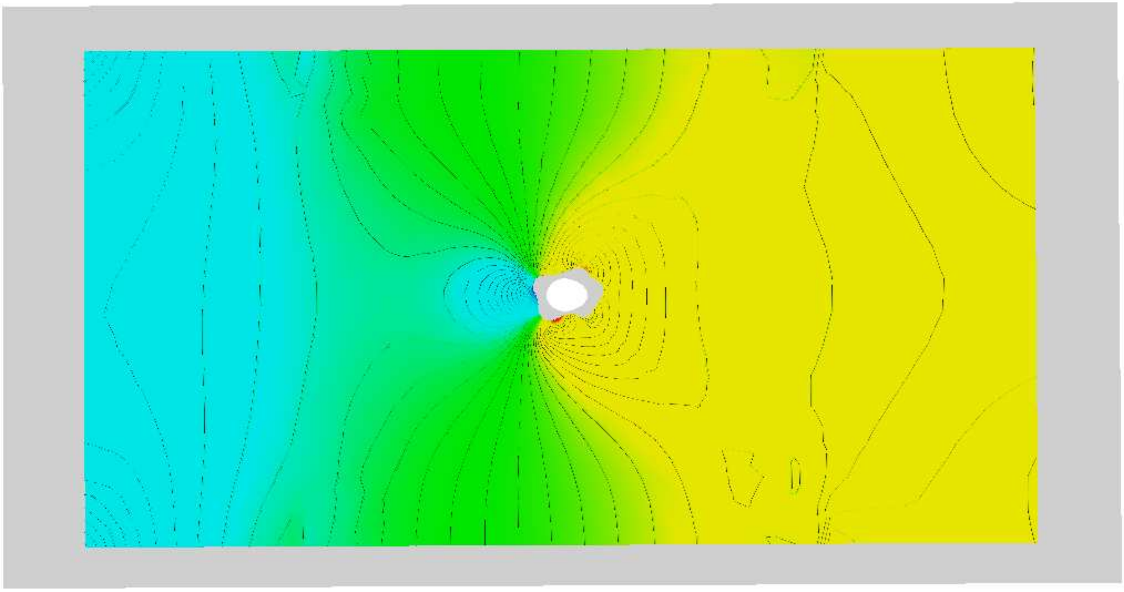

Abbildung 4.22: Druck mit Isodrucklinie auf einer Schnittfläche für die berechnete Strömung auf dem mit TFI generierten Gitter ohne Glättung der Blockgrenzen.

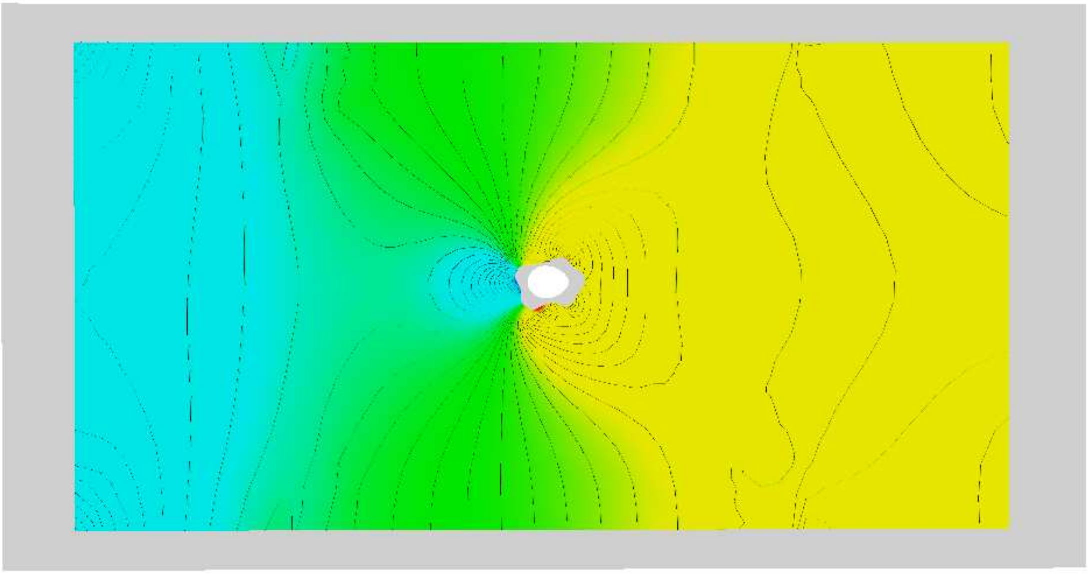

Abbildung 4.23: Druck mit Isodrucklinie auf einer Schnittfläche für die berechnete Strömung auf dem mit TFI generierten Gitter mit Glättung der Blockgrenzen.

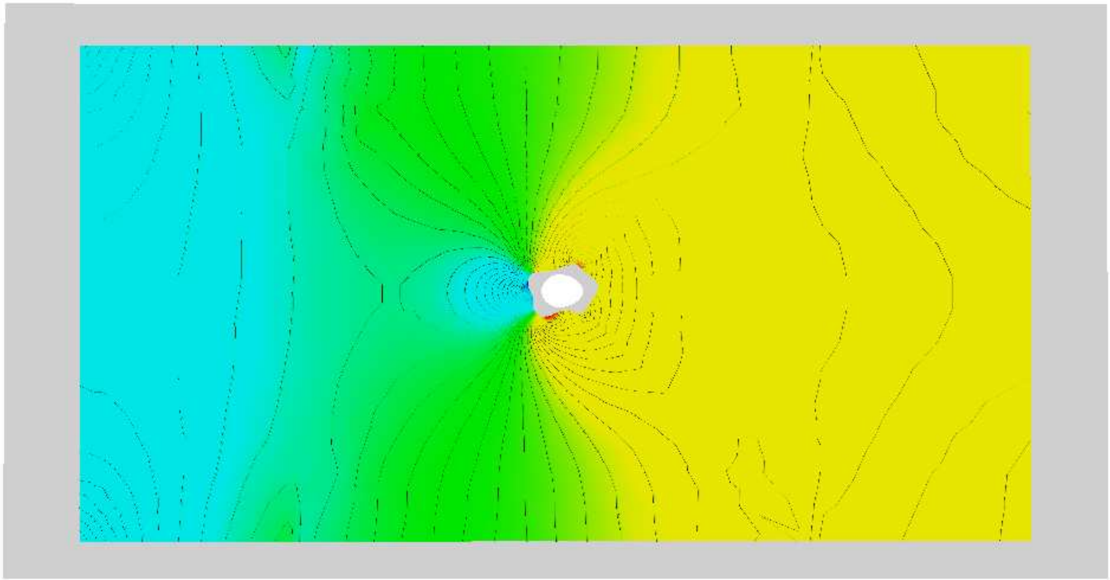

Abbildung 4.24: Druck mit Isodrucklinie auf einer Schnittfläche für die berechnete Strömung auf dem geglätteten Gitter ohne Glättung der Blockgrenzen.

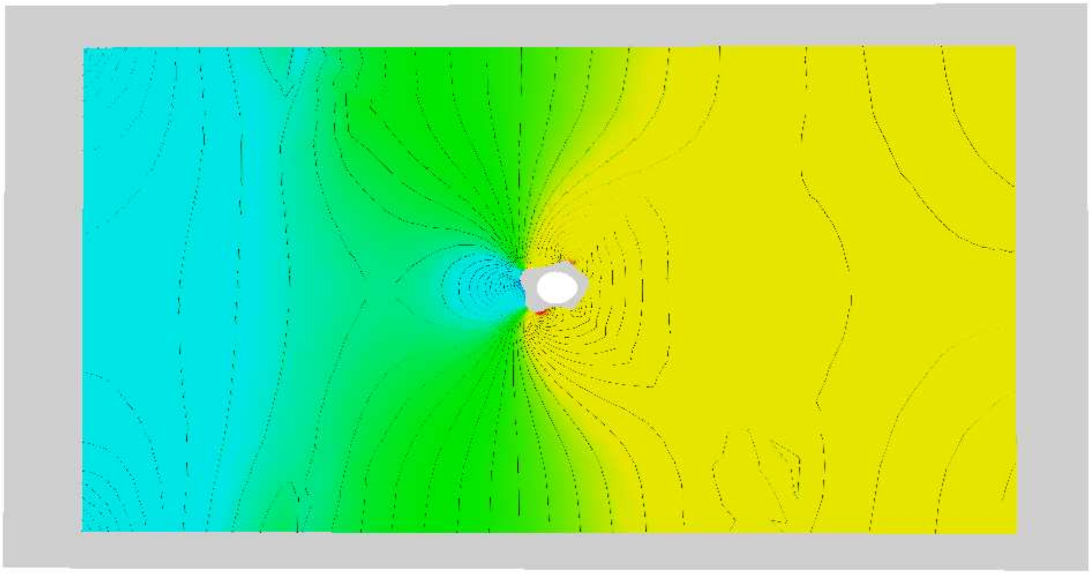

Abbildung 4.25: Druck mit Isodrucklinie auf einer Schnittfläche für die berechnete Strömung auf dem geglätteten Gitter mit Glättung der Blockgrenzen.

Kapitel 5

Zusammenfassung und Ausblick

5.1 Zusammenfassung

Zunächst wurde ein paralleler Gittergenerierer für blockstrukturierte Gitter entwickelt, mit dem für eine große Klasse von CAD-erstellten Geometrien automatisch Gitter für den Navier-Stokes-Gleichungs-Löser NSCL berechnet werden. Der Gittergenerierer liest die Geometriedaten im weit verbreiteten IGES-Format ein und benutzt zur internen Darstellung der Geometrie NURBS-Flächen. Anhand einer vorher auszuwählenden Blocktopologie wird die Umgebung der Geometrie beziehungsweise das Innere der Geometrie, in Blöcke zerteilt, in denen dann blockweise parallel Gitter generiert werden.
Die Gitter in den einzelnen Blöcken werden mittels Transfiniter Interpolation erzeugt und anschließend mit elliptischen Differentialgleichungsmethoden geglättet. Durch die Glättung der Gitter wird die Bijektivität der Gitterabbildung garantiert, die eine Voraussetzung für die Benutzbarkeit der Gitter ist. Außerdem werden durch die Transfinite Interpolation vom Gebietsrand her übernommene Unglattheiten geglättet.
Es wurden zwei verschiedene elliptische Differentialgleichungssysteme zur Gittergenerierung implementiert. Erstens ein transformiertes Laplace-System, anhand dessen die grundlegenden Eigenschaften "elliptischer" Gitter aufgezeigt wurden. Und zweitens ein auf Spekreijse [33] zurückgehendes transformiertes Poisson-System, das unerwünschte, bei elliptischen Gittern auftretende Effekte der Krümmung des Gebietsrandes auf den Verlauf der Gitterlinien, vermeidet. Die zur Bestimmung der Lösungen der transformierten Differentialgleichungen zu lösenden nichtlinearen Gleichungssysteme wurden mittels der Picard-Iteration gelöst. Für das hierbei entstehende lineare Teilproblem wurden verschiedene Methoden zur Lösung linearer Geichungssysteme verglichen. Des Weiteren wurde bei der Picard-Iteration der Einfluss einer Dämpfung auf die Konvergenzgeschwindigkeit untersucht.

Um eine starke Krümmung der Gitterlinien an den Blockgrenzen zu vermeiden, wurde eine einfache Methode entwickelt, die die Gitterpunkte an den Blockgrenzen so verschiebt, dass die Gitterlinien hier glatter verlaufen.
Durch Beispiele wurde die Einsetzbarkeit des Gittergenerierers für unterschiedliche Geometrien in mehreren Topologiefällen, sowie die Effizienz der Parallelisierung gezeigt. Außerdem konnte ein positiver Effekt der Glättung an den Blockgrenzen auf die berechnete Lösung gezeigt werden. Schließlich ist NSCL durch den hier vorgestellten Gittergenerierer zu einem flexiblen, leicht zu benutzenden Simulationswerkzeug geworden.

5.2 Ausblick

Zur Erhöhung der Flexibilität, bezüglich der handhabbaren Geometrien, können noch weitere Teile des IGES-Standards in die Implementierung mitaufgenommen werden. Gundsätzlich ist es möglich, den gesamten NASA-IGES-Standard (die Teilmenge von IGES, welche die komplette NURBS-Oberflächenbeschreibung beinhaltet) zu implementieren und so beliebige CAD-erzeugte Geometrien zumindest einlesen zu können.
Da an eine automatische (Block-) Topologiegenerierung nicht zu denken ist, wird die Erstellung neuer Blocktopologien wohl immer Handarbeit bleiben. Ein Ziel für die Zukunft und Thema aktueller Forschung ist, die dafür nötigen Schritte zu vereinfachen. Die einzelnen Schritte bei der Blockgenerierung sind immer die selben. So ist es naheliegend die vorhandenen Funktionen weiterzubenutzen. Dies könnte in Form einer erweiterbaren Bibliothek, die auf das hier entwickelte Programm aufbaut, geschehen.
Ein weiterer Schritt in Richtung Erweiterung der handhabbaren Geometrien wäre, wie in Kapitel 3.1.3 erwähnt, die Implementierung einer Glättung für zweidimensionale Untermannigfaltigkeiten im $\mathbb{R}^3$. Solche Glättungen sind zum Beispiel von Khamayseh und Mastin [17] entwickelt worden.
Im Bereich der Gittergenerierung gibt es zahlreiche Methoden, die durch Verwendung anderer Differentialgleichungen Gitter mit speziellen Eigenschaften erzeugen. Um die Bandbreite der Eigenschaften der generierten Gitter zu erweitern, könnte man zusätzlich zum Laplace- und Poisson-System noch weitere Differentialgleichungssysteme implementieren, um zum Beispiel Orthogonalität der Gitterzellen am Rand zu erreichen. Die Orthogonalität der Gitter am Blockrand stellt eine vielversprechende Alternative zur hier entwickelten Glättung der Blockgrenzen dar. Eine solche Methode wird zum Beispiel in [16] vorgestellt.
Das CAD-Programm Pro/Engineer bietet mit Pro/Toolkit die Möglichkeit, über eine Bibliothek in der Programmiersprache C Funktionen von Pro/Engineer aus anderen Programmen heraus zu bedienen. Diese Möglichkeit lässt sich mit der von

NSCL bereitgestellten Optimierung kombinieren, um einen geschlossenen Optimierungskreislauf von Gittergenerierung, Strömungssimulation mit Optimierung, automatischer Geometrieänderung und erneuter Gittergenerierung zu erhalten. Durch die Definierung geometriebestimmender Größen im CAD-Programm, nach denen optimiert würde, ließe sich so ein flexibles Optimierungswerkzeug bereitstellen.

An dieser Stelle möchte ich mich bei Herrn Prof. Dr. Michael Griebel und Marcel Arndt für die Überlassung des Themas und die intensive Betreuung bedanken.

Anhang A

A.1 Herleitung der Formeln aus Abschnitt 3.3

Es wird die Herleitung der Formeln

$$a^i = \nabla \xi_i = \frac{1}{\sqrt{det(g_{ij})}} \; (a_j \times a_k)$$

und

$$\begin{aligned} a^i \cdot a_l &= \frac{1}{\sqrt{det(g_{ij})}} \; (a_j \times a_k) \cdot a_l \\ &= \delta_{il} \end{aligned}$$

gezeigt. Dabei sind $a_i = \left(\frac{\partial x_1}{\partial \xi_i}, \frac{\partial x_2}{\partial \xi_i}, \frac{\partial x_3}{\partial \xi_i} \right)^T$, sowie $a^j = \left(\frac{\partial \xi_j}{\partial x_1}, \frac{\partial \xi_j}{\partial x_2}, \frac{\partial \xi_j}{\partial x_3} \right)^T$.
Dafür benötigen wir das Divergenztheorem. Laut diesem gilt

$$\int_\Omega \nabla A \; d\Omega = \int_{\partial\Omega} A \cdot n \; ds, \tag{A.1}$$

für ein stetiges Vektorfeld A, das auf dem offenen zusammenhängenden Gebiet $\Omega \subset \mathbb{R}^3$ und $\partial\Omega$ integrierbar ist.
Wir wenden das Theorem auf einen infinitesimal kleinen Würfel und dessen begrenzende Flächen an. Diese Flächen sollen Koordinatenflächen sein.
Um das Flächenintegral in gekrümmten Koordinaten zu berechnen, benötigt man also die Größe eines infinitesimal kleinen Flächenelements mit konstanter Koordinate ξ_i:

$$dS_i = |a_j \times a_i| \, d\xi_j d\xi_k, \tag{A.2}$$

wobei i, j und k zyklisch vertauschen. Das Produkt mit der äußeren Normalen lautet dann

$$n \; dS_i = \pm \; a_j \times a_k \; d\xi_j d\xi_k.$$

Das Vorzeichen hängt hier von der Lage der Fläche zum Volumen ab.
Nun braucht man noch die Größe eines Volumenelements in gekrümmten Koordinaten:

$$\begin{aligned} dV &= a_i \cdot (a_j \times a_k)\, d\xi_i d\xi_j d\xi_k, \text{ also auch} \\ &= a_1 \cdot (a_2 \times a_3)\, d\xi_1 d\xi_2 d\xi_3. \end{aligned}$$

Sei g die 3×3 Matrix mit den Einträgen $g_{ij} = a_i \cdot a_j$. Dann erhält man folgenden Ausdruck für die Deteminante $det(g)$ von g:

$$\begin{aligned} [a_1 \cdot (a_2 \times a_3)]^2 &= g_{11}\left(g_{22}g_{33} - g_{23}^2\right) - g_{13}^2 g_{22} \\ &\quad -g_{12}^2 g_{33} + 2g_{13}g_{12}g_{23} \\ &= g_{11}\left(g_{22}g_{33} - g_{23}^2\right) \\ &\quad -g_{12}\left(g_{12}g_{33} - g_{13}g_{23}\right) \\ &\quad +g_{13}\left(g_{12}g_{23} - g_{13}g_{22}\right) \\ &= det(g). \end{aligned}$$

Damit ist $dV = \sqrt{det(g_{ij})}\; d\xi_1\; d\xi_2\; d\xi_3$.
Das Divergenztheorem A.1 für ein Volumenelement V kann nun geschrieben werden als

$$\begin{aligned} \int_V (\nabla \cdot A)\; \sqrt{det(g_{ij})}\; d\xi_1\; d\xi_2\; d\xi_3 &= \sum_{i=1}^{3} \int_{S_+^i} A \cdot (a_j \times a_k)\; d\xi_j d\xi_k \\ &- \sum_{i=1}^{3} \int_{S_-^i} A \cdot (a_j \times a_k)\; d\xi_j d\xi_k, \end{aligned}$$

wobei i, j, k zyklisch vertauschen.
Dabei sind S_+^i und S_-^i jeweils die gegenüberliegenden Seiten des Volumenelements, auf denen die Koordinate ξ_i konstant ist.
Die Betrachtung des Limes $vol(V) \to 0$ liefert nach Division durch $\sqrt{det(g_{ij})}\, d\xi_1\; d\xi_2\; d\xi_3$ den Ausdruck

$$\nabla \cdot A = \frac{1}{\sqrt{det(g_{ij})}} \sum_{i=1}^{3} \left[(a_j \times a_k) \cdot A\right]_{\xi_i}.$$

Nun gilt

$$
\begin{aligned}
\sum_{i=1}^{3} (a_j \times a_k)_{\xi_i} &= \sum_{i=1}^{3} \left(x_{\xi_j} \times x_{\xi_k}\right)_{\xi_i} \\
&= \sum_{i=1}^{3} \left(\left(x_{\xi_j \xi_i} \times x_{\xi_k}\right) + \left(x_{\xi_j} \times x_{\xi_k \xi_i}\right)\right) \\
&= \sum_{i=1}^{3} \left(\left(x_{\xi_j \xi_i} \times x_{\xi_k}\right) + \left(x_{\xi_k} \times x_{\xi_i \xi_j}\right)\right) \\
&= 0
\end{aligned}
$$

und man erhält:

$$
\nabla \cdot A = \frac{1}{\sqrt{det(g_{ij})}} \sum_{i=1}^{3} (a_j \times a_k) \cdot A_{\xi_i}. \tag{A.3}
$$

Dabei stehen die Indizes $_{\xi_i}$ für Differenziation nach ξ_i.
Ersetzt man in A.1 A durch einen Skalar, so wird A.3 zu:

$$
\begin{aligned}
\nabla A &= \frac{1}{\sqrt{det(g_{ij})}} \sum_{i=1}^{3} (a_j \times a_k) \, A_{\xi_i}, \text{ also für } \xi_i\text{:} \\
\nabla \xi_k &= \frac{1}{\sqrt{det(g_{ij})}} \sum_{i=1}^{3} (a_j \times a_k) \, \xi_{k \xi_i} \\
&= \frac{1}{\sqrt{det(g_{ij})}} \sum_{i=1}^{3} (a_j \times a_k) \, \delta_{ki}.
\end{aligned}
$$

Dadurch erhält man eine Formel für die a^i abhängig von den a_j:

$$
a^i = \nabla \xi_i = \frac{1}{\sqrt{det(g_{ij})}} \, (a_j \times a_k),
$$

und man erhält

$$
\begin{aligned}
a^i \cdot a_l &= \frac{1}{\sqrt{det(g_{ij})}} \, (a_j \times a_k) \cdot a_l \\
&= \delta_{il}.
\end{aligned}
$$

Damit ist auch die Formel

$$\begin{pmatrix} a^{1T} \\ a^{2T} \\ a^{3T} \end{pmatrix} (a_1, a_2, a_3) = (id)$$

gezeigt.

Anhang B

B.1 Handhabung des Programms

Zur Generierung blockstrukturierter Gitter mit dem in dieser Arbeit entwickelten Gittergenerierers werden erstens eine IGES-Datei mit der Geometriebeschreibung und zweitens eine Datei mit den für die Gittergenerierung nötigen Daten sowie den Daten für die Strömungsberechnung benötigt. Außerdem werden in einigen Fällen noch zwei weitere Dateien benötigt, die die Feinsteuerung der Blockunterteilung regeln. Im Folgenden wird kurz auf die Erzeugung einer vom Gittergenerierer benutzbaren Geometrie eingegangen. Hierfür wird von der Nutzung des CAD-Programms Pro/Engineer ausgegangen. Anschließend werden die Eingabeparameter für die Gittergenerierung erklärt.
Es wird nur grob auf die Konstruktion der Geometrie eingegangen. Für Details wird auf Handbücher, zum Beispiel [35], verwiesen. In der vorliegenden Implementierung ist die Rekonstruktion von Objekten, die in Pro/Engineer mittels "Protrusion" erzeugt wurden, vorgesehen. Von den verschieden Möglichkeiten, Protrusion-Objekte zu erzeugen, sind "Blend" und "Extrude" berücksichtigt. Mittels Extrude wird ein vorzugebender zweidimensionaler Querschnitt in die dritte Dimension "herausgezogen". Hiermit ist es möglich, unterschiedliche "2 1/2D"-Objekte zu erstellen. Die Geometrien für die Gitter zur Konvergenzuntersuchung in Kapitel 3.3.3 mit einfacher und komplizierter Geometrie wurden auf diese Weise erstellt. Blend bietet die Möglichkeit, zwei verschiedene Querschnitte vorzugeben. Das dreidimensionale Objekt ist dann das Ergebnis einer Interpolation zwischen den beiden Querschnitten. Mit dieser Funktion wurden alle anderen betrachteten Beispiele in dieser Arbeit erstellt. Hiermit ist es also möglich "echte" 3D-Objekte zu erstellen. Beim Export der Geometrie in eine IGES-Datei sollte sichergestellt sein, dass die Oberflächendarstellung eingeschaltet ist.
Die Eingabedatei mit den Gitter- und Strömungsparametern enthält die entsprechenden Parameter in der Reihenfolge, wie sie nun erklärt werden. Dabei sind die einzelnen Werte jeweils durch Leerzeichen zu trennen.

- Die IGES-Datei mit absolutem Pfad
- Die Blocktopologie, d.h. einer der folgenden Strings: 22BLOCK, O_GITTER, H_GITTER, C_GITTER (keine NSCL-Ausgabe, aber implementiert), SPLIT-_INSIDE (Topologie für Beispiel 5)
- Die Anzahl der Blöcke (für SPLIT_INSIDE)
- Anzahl der Gitterpunkte in x-Richtung [1]
- Anzahl der Gitterpunkte in y-Richtung
- Anzahl der Gitterpunkte in z-Richtung
- 0 für Laplace-System oder 1 für Poisson-System
- Abbruchkriterium für Glättung
- Gradierung in x-Richtung [2]
- Gradierung in y-Richtung
- Gradierung in z-Richtung
- Maximale Anzahl der Picard-Iterationen
- Dämpfungsparameter für die Picard-Iteration

Die folgenden Daten sind NSCL-Parameter, die auch miteingegeben werden müssen.

- ϵ, absolutes Abbruchkriterium für das Residuum der Druckgleichung
- δ, relatives Abbruchkriterium für das Residuum der Druckgleichung, d. h. $\frac{||res||}{||res_{begin}||} < \delta$
- maxiter, maximale Anzahl der Iterationen für die Druckberechnung
- re, die Reynoldszahl
- $\nu = re^{-1}$
- α, Upwind-Blending-Parameter zur Steuerung der Disketisierung

[1] Bei der 22-Block-Topologie werden diese Werte für einige Blöcke im Programm geändert. Gezählt werden alle Gitterpunkte inklusive Randpunkte

[2] Für 0 oder 1 keine Gradierung. Werte > 1 konzentrieren Punkte mit kleinen Parameterwerten, Werte < 1 konzentrieren Punkte mit größeren Parameterwerten (in der entsprechenden Richtung)

- tMax, die Zeit des Berechnungsendes
- deltLimit, die maximale Zeitschrittweite[3]
- diffDeltFactor, Sicherheitsfaktor die maximale Zeitschritweite aus dem diffusiven Term [4]
- convDeltFactor, Sicherheitsfaktor die maximale Zeitschritweite aus dem konvektiven Term
- writeInterval, das Zeitintervall für die Datenausgabe

Diese Daten müssen in der angegebenen Reihenfolge in einer Datei stehen deren Name erstes Argument des Gittergenerierers sein muss. In den Topologiefällen 22BLOCK,O_GITTER, H_GITTER und C_GITTER werden zur Feinsteuerung der Blockaufteilung die Dateien "splitpoints.in" und "umgebung.in" benötigt. In der Datei "splitpoints.in" wird bestimmt, wo die Seiten des Geometrieblocks zerschnitten werden, um neue Blöcke zu generieren. Diese Datei wird eingelesen, falls der Geometrieblock nur aus vier Seiten besteht, was bei Geometrien, die für die genannten Blocktopologien ausgewählt werden, häufig der Fall ist. Es wird davon ausgegangen, dass der Block aus einer oberen und unteren sowie aus einer vorderen und einer hinteren Seite besteht. Um einen sechsseitigen Block zu erhalten, werden nun die Ober- und Unterseite in y-Richtung jeweils in drei Teile zerschnitten. Der mittlere Teil bleibt jeweils Ober- beziehungsweise Unterseite. Die restlichen vier Seitenstücke werden zu einer linken und einer rechten Seite kombiniert, um schließlich einen sechsseitigen Block zu erhalten. Dabei wird das linke Teilstück der Oberseite mit dem linken Teilstück der Unterseite zur linken Seite des Geometrieblocks verschmolzen. Analog wird die rechte Seite erstellt. Die Seiten werden entlang zweier Kurven mit konstanten Parameterwerten zerschnitten. Die Datei "splitpoints.in" enthält die vier Parameterwerte entlang derer die Seiten zerschnitten werden. Die ersten beiden sind für die Oberseite, hier erst der Wert für die linke Kurve, dann der Wert für die linke Kurve. Dann folgen in derselben Reihenfolge die Werte für die untere Seite.
Die Datei "umgebung.in" steuert im Falle der Umströmung des Objekts die Größe des Simulationsgebiets. Hier stehen in der Reihenfolge links, rechts, oben, unten, vorne, hinten die jeweiligen Steuerparameter. Sie geben an, wie groß der Abstand des Objekts zur Gebietsgrenze nach rechts, links, etc. im Verhältnis zur maximalen Breite/Höhe/Tiefe ist.

[3]Die deltLimits sind Parameter zur Feinsteuerung der adaptiven Zeitschrittwahl zur Erfülleung der CFL-Bedingung. Siehe zur erklähreng [10, Kap 3.2.4]

[4]Da die in [10, Kap3.2.4] gegebene Abschätzung relativ scharf ist, wird hier i.A. ein Sicherheitsfaktor < 1 benutzt.

Literaturverzeichnis

[1] *Special Issue on Meshless Methods*, volume 139. Comput. Meths. Appl. Mech. Engrg, 1996.

[2] Satish Balay, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matt Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc home page. http://www.mcs.anl.gov/petsc, 2001.

[3] S.N. Bernstein. Démonstration du thèoreme de Weierstrass fondeè sur le calcul dès probabilitès. *Commun. Soc. Math. Khrakow*, 12:1–2, 1912.

[4] M.P. Do Carmo. *Riemannian Geometry.* Birkhäuser, Boston, Basel, Berlin, 1992.

[5] J.F. Dannenhoffer. A technique for optimizing grid blocks. In *Proceedings of the Surface Modeling, Grid Generation and Related Issues in Computational Fluid Dynamics Workshop, NASA Conference Publication 3291*, Cleveland, OH, May 1995.

[6] P. Deufelhard and A. Hohmann. *Numerische Mathematik I.* de Gryter, Berlin, 2002.

[7] H.A. Van Der Forst. A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13:631 – 644, 1992.

[8] D. Gilbarg and N.S. Trudiner. *Elliptic Partial Differential Equations of Second Order.* Springer, Berlin, Heidelberg, New York, 1977.

[9] W.N. Gordon and C.A. Hall. Construction of curvilinear coordinate systems and application to mesh generation. *International Journal of Numerical Method in Engineering*, 7:461 – 477, 1973.

[10] M. Griebel, T Dornseifer, and Tilman Neunhoeffer. *Numerische Simulation in der Strömungsmechanik.* Vieweg, Wiesbaden, 1995.

[11] M. Griebel and M.A. Schweitzer, editors. *Meshfree methods for partial differential equations. International workshop, Univ. Bonn, Germany, September 11–14, 2001, Lecture Notes in Computational Science and Engineering 26*, Berlin, 2002. Springer.

[12] Grossmann and Roß. *Numerik elliptischer Differentialgleichungen.* B. G. Teubner, Stuttgart, 1992.

[13] W. Hackbusch. *Theorie und Numerik elliptischer Differentialgleichungen.* B. G. Teubner, Stuttgart, 1986.

[14] W. Huber. *Paralleles Rechnen.* Oldenburg Verlag, München, 1997.

[15] O.D. Kellog. *Foundations of Potential Theory.* Springer, 1967.

[16] A. Khamayseh, A. Kuprat, and C.W. Mastin. Boundary orthogonality in elliptic grid generation. In J. F. Thompson, B. K. Soni, and N. P. Weatherill, editors, *Handbook of Grid Generation.* CRC Press, Boca Raron, London, New York, Washington, D.C, 1999.

[17] A. Khamayseh and C.W. Mastin. Computational conformal mapping for surface grid generation. *Journal of Computational Physics*, 123:394 – 401, 1996.

[18] K. Königsberger. *Analysis 2.* Springer, Berlin, Heidelberg, 1993.

[19] Philipe Lavoie. nurbs++ homepage. http://www.libnurbs.sourceforge.net, 2001.

[20] V.D. Liseikin. *Grid Generation Methods.* Springer, Berlin, Heidelberg, 1999.

[21] C.W. Mastin and J.F. Thompson. Tansformation of three-dimensional regions onto rectangular regions by elliptic systems. *Numerische Mathematik*, 29:397 – 407, 1978.

[22] A. Meister. *Numerik linearer Gleichungssysteme.* Vieweg, Braunschweig, Wiesbaden, 1999.

[23] M. Meyer. Optimierung und parallele Berechnung von Strömungen in gekrümmten dreidimensionalen Koordinaten. Diplomarbeit, Institut für Angewandte Mathematik, Universität Bonn, 2001.

[24] M.Flynn. Very high-speed computing systems. *Proc. IEEE*, 54:1901 – 1909, 1966.

[25] S. Nakamura. Marching grid generation using parabolic partial differential equations. *Numerical Grid Generation*, 1982.

[26] Ralph W. Noack and Dale A. Anderson. Solution-adaptive grid generation using parabolic partial differential equations. *AIAA Journal. American Inst. of Aeronautics and Astronautics 28, No.6, 1016-1023*, 1990.

[27] Ogawa, Satoru, Ishiguro, Tomiko, and Yoko Takakura. Hyperbolic grid generation scheme for simulating flow about three- dimensional complex configuration. *Computational techniques and applications, Proc. Int. Conf., CTAC-3, Sydney/Aust. 1987, 577-587 (1988)*, 1988.

[28] J.M. Ortega and W.C. Rheinboldt. *Methods for Solving Systems of Nonlinear Equations in Several Variables.* Academic Press, New York, 1970.

[29] L. Piegl and W. Tiller. *The NURBS Book.* Springer, 1995.

[30] M.H. Protter and H.F Weinberger. *Maximum Principles in Differential Equations.* Springer, 1984.

[31] ptc. ptc homepage. //http://www.ptc.com/go/wildfire/index.htm, 2001.

[32] K. Reed. *The Initial Graphics Exchange Specification Version 5.1.* National Institut of Standards and Technologie, Gaithersburg, USA, 1991.

[33] S. Spekreijse. Elliptic grid generation based on Laplace equations and algebraic transformations. *Journal of Computational Physics*, 118:38 – 61, 1994.

[34] J.F. Thompson, Z.U.A. Warsi, and C.W. Mastin. *Numerical Grid Generation Foundations and Applications.* North-Holland, 1985.

[35] R. Toogood. *Pro/Engineer Tutorial Release* $200i^2$. Schroff Development Corporation, 2000.

[36] V.N. Vatsa, Sanetrik, M.D., and E.B. Parlette. Block-structured grids for complex aerodynamic configurations. In *Proceedings of the Surface Modeling, Grid Generation and Related Issues in Computational Fluid Dynamics Workshop, NASA Conference Publication 3291*, Cleveland, OH, May 1995.

[37] W.Cropp, E. Lusk, and A. Skjellum. *Using MPI.* MIT Press, Cambridge, Ma, 1994.