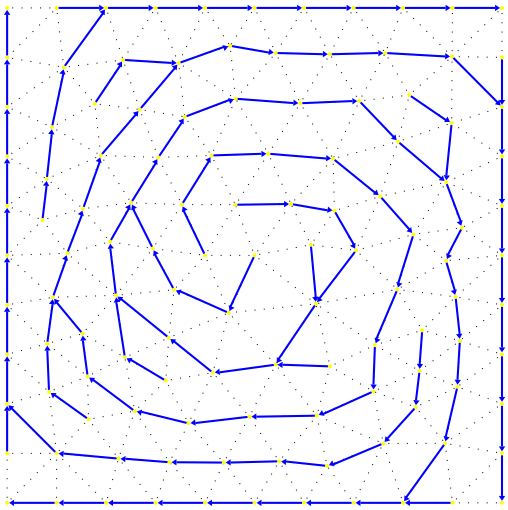
Flußorientierte Numerierungsstrategien zur Vorkonditionierung konvektionsdominanter Probleme



Diplomarbeit

vorgelegt von

Heiko Schilling

aus Leinefelde

angefertigt am

Institut für Numerische und Angewandte Mathematik

der Georg-August-Universität zu Göttingen

2000

Flußorientierte Numerierungsstrategien zur Vorkonditionierung konvektionsdominanter Probleme

Diplomarbeit

vorgelegt von

Heiko Schilling

aus Leinefelde

angefertigt am
Institut für Numerische und Angewandte Mathematik
der Georg-August-Universität zu Göttingen

2000

Inhaltsverzeichnis

Einleitung

Die Qualität und Leistungsfähigkeit numerischer Simulationen physikalischer Phänomene hat in den letzten drei Jahrzehnten stark zugenommen. Einerseits liegt dies natürlich an den rasanten Entwicklungen in der Computertechnologie. Andererseits könnten trotz immens gestiegener Rechnerleistung viele praxisrelevante Probleme heute noch nicht gelöst werden, wenn nicht auch auf mathematischer Seite bei der Entwicklung effizienter Algorithmen ähnlich große Fortschritte erzielt worden wären.

Eine wichtige Klasse untersuchter Probleme stammt aus der Strömungsmechanik. Als einfaches Beispiel seien hier stationäre Transportprobleme genannt, welche sich z.B. bei der Untersuchung von Temperatur– und Stoffverteilung in inkompressiblen Medien stellen.

Solche Transportprobleme werden mathematisch durch elliptische Differentialgleichungen beschrieben, die gewöhnlicherweise nicht analytisch lösbar sind. Bei der Bestimmung einer approximativen Lösung mittels der Finite–Elemente–Methode (FEM) diskretisiert man die Gleichung auf dem triangulierten Gebiet, wodurch schwach besetzte Gleichungssysteme endlicher, jedoch großer Dimension entstehen.

Die effektive Lösung dieser Systeme ist das Kernproblem bei der Anwendung der FEM. Das elementare Werkzeug dafür sind iterative Löser, die auch in komplexeren Lösungsmethoden (Gebietszerlegungs– oder Mehrgitterverfahren) zur Behandlung von Teilproblemen eingesetzt werden.

Während die Konvergenz bei Krylov–Unterraum–Methoden lediglich von der Kondition der Matrix abhängt, wird sie für einige stationäre Verfahren auch von der Struktur der Matrix bestimmt. So ist das Gauß–Seidel– bzw. SOR–Verfahren insbesondere dann effizient, wenn die betragsmäßig größten Einträge vorwiegend im unteren Teil der Matrix stehen. Dies erreicht man im konvektionsdominanten Fall durch eine Umnumerierung der Unbekannten in Flußrichtung.

Achi Brandt [Bra78] wies bereits 1978 darauf hin, daß Gauß–Seidel sich in diesem Fall mit einer entsprechend flußorientierten Numerierung als Löser eignet. Noch bis in den 80er Jahren wurden Numerierungstechniken in Lösungsverfahren bei Problemen mit konstanter Strömungsrichtung auf uniform unterteilten Rechteckgittern eingesetzt. Eine geeignete Numerierung erhält man hier leicht durch Abzählen der Gitterpunkte in lexikographischer Richtung.

Erst mit der Verbreitung adaptiver Lösungstechniken und dem Einsatz unstrukturierter Simplexgitter wurde eine Entwicklung neuer Numerierungsstrategien erforderlich.

Die ersten Algorithmen dazu stammen von P. Bastian & G. Wittum [BW94] und benutzen *Breadth First Search* (*Breitensuche*) aus der Graphentheorie. Sie führten den Begriff „*Downwind Numbering*“ für flußorientierte Numerierung ein. J. Bey erweiterte ihr Verfahren auf dreidimensionale Probleme [BW95].

Numerierungsverfahren machen allerdings nur unter der Voraussetzung Sinn, daß eine Numerierung in Richtung der Strömung überhaupt existiert. Dies ist genau dann der

Fall, wenn im betrachteten Strömungsfeld keine Wirbel auftreten, also der entsprechende Konvektionsgraph azyklisch ist. In der Praxis ist diese Bedingung aber eher selten erfüllt.

J. BEY [Bey98] und W. HACKBUSCH [Hac95] entwickelten zwei Verfahren für den praxisrelevanteren zyklischen Fall. Ersterer numeriert die auftretenden Zyklen (Zusammenhangskomponenten) in Flußrichtung, wodurch der Einsatz eines Block–Gauß–Seidel–Lösers möglich wird. Die Effizienz dieses Verfahrens sinkt mit der Größe der Zyklen.

Beim HACKBUSCH–Algorithmus wird versucht, die Zyklen an minimal vielen Knoten aufzutrennen, um so den zyklusfreien Anteil des Konvektionsgraphen abzuspalten. Die Menge der „Schnittknoten" bezeichnet man dabei als *Feedback–Back–Vertices*. Hier läßt sich dann die Schur–Komplement–Methode zur Lösung einsetzen. Das Verfahren ist allerdings auf planare Graphen beschränkt.

Die Suche minimaler *Feedback–Vertex–Sets* ist ein NP–vollständiges Problem, siehe GAREY [GJ79] (Problem GP 7), und somit nicht mit polynomialem Zeitaufwand lösbar. Der HACKBUSCH–Algorithmus findet ein bis auf den Faktor 2 minimales Feedback–Vertex–Set.

Zu den zahlreichen weiteren Anwendungen des Feedback–Vertex–Set–Problems zählen *Programmverifikation* [Sha79], *Auflösung von Datenabhängigkeiten* (*loop unrolling*) [PH96], *Vermeidung von Deadlocks* [WLS85] und *Bayesian Inference* [YGNR94].

Ein effektives Verfahren zum Downwind Numbering im dreidimensionalen Fall bei beliebigem Strömungsfeld ist nicht bekannt.

Gegenstand dieser Arbeit ist die Untersuchung der Numerierungstrategien von J. BEY und W. HACKBUSCH. Das beinhaltet ihre Implementation und numerische Analyse.

Bei den bisher in der Literatur vorhanden numerischen Analysen dieser beiden Verfahren wurde stets die Galerkin–FEM mit einer hybriden Upwind–Stabilisierung verwendet. Siehe J. BEY [Bey98], S. LE BORNE [Bor99], T. PROBST [Pro99] und J. WUNNER [Wun96].

Im Rahmen dieser Arbeit soll auch der Einsatz der Numerierungsstrategien bei SUPG– und Shock–Capturing–Stabilisierung der Galerkin–FEM untersucht werden.

Im Einzelnen ergibt sich damit folgender Aufbau dieser Arbeit:

Kapitel 1 enthält eine kurze Einführung in die Lösbarkeitstheorie der Konvektions–Diffusions–Reaktions–Probleme. Es wird auf die Galerkin–Finite–Elemente–Methode eingegangen und die SUPG– bzw. Shock–Capturing–Stabilisierung erläutert.

Kapitel 2 gibt eine Einführung in die Graphentheorie. Es werden Grundbegriffe erläutert und einfache Algorithmen zur Numerierung von Graphen vorgestellt, wie z.B. die Tiefensuche. Abschließend wird der von J. BEY verwendete TARJAN–Algorithmus erläutert.

Kapitel 3 stellt die Numerierungsstrategien für den zyklischen und den azyklischen Fall vor und erläutert die möglichen Kriterien zur Erzeugung eines Konvektionsgraphen aus den Problemdaten. Außerdem werden das SOR–Verfahren und die Schur–Komplement–Methode erläutert.

Kapitel 4 beschreibt die Implementation der Verfahren von J. BEY und W. HACKBUSCH. Die implementierten Kriterien werden erläutert, sowie die ebenfalls implementierte Visualisierung von Graphen und Matrizen.

Kapitel 5 stellt die numerischen Ergebnisse dar. Es werden die Testprobleme vorgestellt, die Durchführung der Versuche erläutert und die erzielten Ergebnisse diskutiert.

Kapitel 6 gibt eine Zusammenfassung und einen Ausblick.

1. Konvektions–Diffusions–Reaktions–Probleme

Dieses Kapitel gibt einen kurzen Überblick über die Lösbarkeitstheorie der Konvektions–Diffusions–Reaktions–Probleme. Ausgehend vom Begriff der klassischen Lösung wird eine Variationsformulierung hergeleitet, die zu einem verallgemeinerten Lösungsbegriff führt. Dazu wird die Definition angepaßter Funktionenräume erforderlich sein. Abschließend diskretisiert man die Differentialgleichung mittels einer Finite–Elemente–Methode (FEM), wodurch ein Gleichungssystem erzeugt wird, aus welchem durch iterative Lösungsverfahren eine approximierte Lösung berechnet werden kann. Die FEM muß zusätzlich noch durch geeignete Techniken stabilisiert werden.

1.1. Grundlagen

Die Darstellung beginnt mit der Einführung der Notation und setzt sich fort mit der Definition der betrachteten Funktionenräume.

Definition 1.1 (Gebiet) Unter einem *Gebiet* Ω versteht man eine offene, zusammenhängende Teilmenge des $\mathbb{R}^n$. Den Rand $\overline{\Omega} \setminus \Omega$ bezeichnet man mit $\partial\Omega$ bzw. Γ.

Im folgenden werden stets beschränkte Gebiete betrachtet.

Definition 1.2 (Träger) Sei u eine auf dem Gebiet Ω definierte Funktion. Dann nennt man

$$supp(u) := \overline{\{x \in \Omega \,|\, u(x) \neq 0\}}$$

den *Träger* dieser Funktion. Man sagt, u besitzt einen kompakten Träger in Ω, falls $supp(u)$ kompakt und in Ω enthalten ist. (Schreibweise: $supp(u) \subset\subset \Omega$)

Für die Definition der klassischen Räume stetiger und stetig differenzierbarer Funktionen werden sogenannte *Multiindizes* verwendet:

Definition 1.3 (Multiindex) Einen Vektor $\alpha := (\alpha_1, \ldots, \alpha_n)$ mit Einträgen $\alpha_i \in \mathbb{N}_0$, $i = 1, \ldots, n$ bezeichnet man als *Multiindex*. Desweiteren nennt man $|\alpha| := \sum_{i=1}^{n} \alpha_i$ den *Betrag* bzw. die *Länge* von α und definiert den folgenden Differentialoperator

$$D^\alpha := \frac{\partial^{|\alpha|}}{\partial x_1^{\alpha_1} \cdots \partial x_n^{\alpha_n}}.$$

Damit lassen sich Funktionenräume für stetige und stetig differenzierbare Funktionen definieren:

Definition 1.4 (stetige und stetig differenzierbare Funktionen) Sei $\Omega \subset \mathbb{R}^n$ ein offenes und beschränktes Gebiet und $m \in \mathbb{N}_0$ dann ist:

(i) die Menge der stetigen Funktionen auf Ω:

$$C(\Omega) := \{u : \Omega \to \mathbb{R} \mid u \text{ ist stetig}\},$$

(ii) die Menge der m-fach auf Ω stetig partiell differenzierbaren Funktionen:

$$C^m(\Omega) := \{u : \Omega \to \mathbb{R} \mid D^\alpha u \in C(\Omega) \text{ für alle } |\alpha| \leq m\},$$

(iii) die Menge der Funktionen aus $C^m(\Omega)$ mit stetiger Fortsetzung auf den Rand:

$$C^m(\overline{\Omega}) := \left\{u : \Omega \to \mathbb{R} \mid D^\alpha u \in C(\overline{\Omega}) \text{ für alle } |\alpha| \leq m\right\},$$

(iv) die Menge der beliebig oft auf Ω stetig partiell differenzierbaren Funktionen:

$$C^\infty(\Omega) := \bigcap_{m=0}^{\infty} C^m(\Omega),$$

(v) die Menge der beliebig oft auf Ω stetig partiell differenzierbaren Funktionen mit beschränktem Träger:

$$C_0^\infty(\Omega) := \{u \in C^\infty(\Omega) \mid supp(u) \subset\subset \Omega\}.$$

Die Funktionen aus $C^\infty(\Omega)$ bzw. $C_0^\infty(\Omega)$ bezeichnet man auch als *glatt*.

Lemma 1.1
Versieht man $C^m(\overline{\Omega})$ mit der Norm

$$\|u\|_{m,\infty} := \max_{0\leq|\alpha|\leq m} \sup_{x\in\Omega} |D^\alpha u(x)|, \quad u \in C^m(\overline{\Omega}),$$

so wird $C^m(\overline{\Omega})$ ein Banach-Raum.

Beweis. Vgl. H. W. Alt [Alt92] (Beweis zu Lemma 1.8, S.25). □

Eine solche grobe Unterscheidung von Funktionen nach ihrem Differenzierbarkeitsgrad wird im folgenden nicht ausreichen. Insbesondere für die Beschreibung der Randglätte eines Gebietes werden deshalb die sogenannten *Hölder-Räume* eingeführt.

Definition 1.5 (Hölder-Raum) Für $m \in \mathbb{N}_0$ und $\lambda \in [0,1]$ nennt man die Teilmenge von $C^m(\overline{\Omega})$, für deren Elemente

$$\|u\|_{C^{m;\lambda}(\overline{\Omega})} := \|u\|_{C^m(\overline{\Omega})} + \sum_{|\alpha|=m} \sup_{\substack{x,y\in\overline{\Omega}\\ x\neq y}} \frac{|D^\alpha u(x) - D^\alpha u(y)|}{|x-y|^\lambda} < \infty \tag{1.1}$$

gilt, den *Hölder Raum* $C^{m;\lambda}(\overline{\Omega})$.

Auch hier gilt:

Lemma 1.2
Mit der in (1.1) definierten Norm ist der Hölder-Raum $C^{m;\lambda}\left(\overline{\Omega}\right)$ *ein Banach-Raum.*

Beweis. Vgl. H. W. Alt [Alt92] (Beweis zu Lemma 1.8, S. 25). □

Definition 1.6 (Hölder-Stetigkeit) Die Funktionen aus $C^{0;\lambda}\left(\overline{\Omega}\right)$ bzw. $C^{0;\lambda}\left(\Omega\right)$ heißen *Hölder-stetig* bzw. *Hölder-stetig* in Ω zum Exponenten λ. Für $\lambda = 1$ sagt man auch, die Funktion sei *Lipschitz-stetig.*

Man vergleiche hierzu die Bemerkung 1.1.4, S.29 in [Bey98]: Hölder-stetige Funktionen sind gleichmäßig stetig, und der Grad der Hölder-Stetigkeit beschreibt, welche Art von Spitzen (Nullwinkeln) eine solche Funktion im ungünstigsten Fall besitzt. In Bezug auf die angesprochene notwendige Randglätte eines Gebietes läßt sich folgende Definition formulieren.

Definition 1.7 (Randglätte) Ein beschränktes Gebiet Ω gehört zur Klasse $C^{m;s}$ mit $m \in \mathbb{N}_0$ und $0 \leq s \leq 1$, falls es zu jedem Punkt $x_0 \in \partial\Omega$ eine Umgebung $U = U\left(x_0\right)$ um x_0 gibt, so daß eine bijektive Abbildung $\psi : U \to D \subset \mathbb{R}^n$ existiert mit den Eigenschaften:

(i) $\psi\left(U \cap \ \Omega\right) \subset \mathbb{R}^n_+$,
(ii) $\psi\left(U \cap \partial\Omega\right) \subset \partial\mathbb{R}^n_+$,
(iii) $\psi \ \ \in C^{m;s}\left(U\right)$,
(iv) $\psi^{-1} \in C^{m;s}\left(D\right)$.

Bemerkung 1.1 (zur Definition der Randglätte) Für Gebiete Ω der Klasse $C^{m;s}$ gilt: jeder Punkt $x_0 \in \partial\Omega$ besitzt eine Umgebung, in der $\partial\Omega$ als Graph einer Funktion aus $C^{m;s}$ von $n-1$ der Variablen $x_1, \ldots, x_n$ darstellbar ist.

Bemerkung 1.2 (zu Lipschitz-stetigen Gebieten) Auf dem Rand eines Lipschitz-stetigen Gebietes existiert fast überall der äußere Normalenvektor.

Klassische Differenzierbarkeitsvoraussetzungen sind in der Regel nicht realistisch, weshalb an dieser Stelle die *Sobolev-Räume* eingeführt werden, die auf den *Lebesgue-Räumen* aufbauen. Dazu versieht man zunächst den $\mathbb{R}^n$ mit dem *Lebesgue-Maß* μ. Falls nun $\Omega \subset \mathbb{R}^n$ eine meßbare Punktmenge ist, so heißen zwei meßbare Funktionen $v, u : \Omega \to \mathbb{R}$ als *äquivalent*, falls $v = u$ fast überall in Ω gilt, d.h. nur auf einer Lebesgue-Nullmenge ist $v \neq u$. Die äquivalenten Funktionen faßt man zu Äquivalenzklassen zusammen, die die Elemente der *Lebesgue-Räume* bilden.

Definition 1.8 (Lebesgue-Räume) Sei die Norm $\|\cdot\|_{L^p}$ für Funktionen $u : \Omega \to \mathbb{R}$ auf einer meßbaren Menge $\Omega \subset \mathbb{R}^n$ gegeben durch

$$\|u\|_{L^p} := \begin{cases} \left(\int_\Omega |u(x)|^p \, dx \right)^{\frac{1}{p}} & \text{für } 1 \leq p < \infty, \\ ess \sup\limits_{x \in \Omega} |u(x)| := \inf\limits_{\substack{N \text{meßbar} \\ \mu(N)=0}} \sup\limits_{x \in \Omega \setminus N} |u(x)| < \infty & \text{für } p = \infty. \end{cases}$$

In beiden Fällen werden die *Lebesgue-Räume* definiert durch

$$L^p(\Omega) := \{u : \Omega \to \mathbb{R} : u \text{ ist meßbar und } \|u\|_{L^p} < \infty\}.$$

Im Fall $p = 2$ wird für Funktionen $u, v \in L^2(\Omega)$ durch

$$(u, v) := (u, v)_{L^2(\Omega)} = \int_\Omega u(x) v(x) \, dx$$

ein Skalarprodukt erklärt. Der Raum $L^2(\Omega)$ ist damit ein Hilbert-Raum.

Für die Einführung der *Sobolev-Räume* benötigt man noch den Begriff der *verallgemeinerten Ableitung* über *Lebesgue-Räumen*.

Definition 1.9 (verallgemeinerte Ableitung) Sei

$$L^1_{loc}(\Omega) := \left\{ v : \Omega \to \mathbb{R} \text{ meßbar } \wedge\, v \in L^1(K) \text{ für jede kompakte Teilmenge } K \subset\subset \Omega \right\}$$

die Menge der *lokal integrierbaren Funktionen*. Dann heißt die Funktion $\omega_\alpha \in L^1_{loc}(\Omega)$ die *verallgemeinerte Ableitung* von $u \in L^1_{loc}(\Omega)$, falls

$$\int_\Omega \omega_\alpha v \, dx = (-1)^\alpha \int_\Omega u D^\alpha v \, dx, \qquad \forall v \in C_0^\infty(\Omega)$$

gilt. Man sagt dann auch, u ist *verallgemeinert differenzierbar*.

Für beliebiges α und $v \in C^{|\alpha|}(\Omega)$ entspricht die verallgemeinerte Ableitung ω_α der klassischen Ableitung $D^\alpha v$. Im folgenden wird stets $D^\alpha v$ statt ω_α für die verallgemeinerte Ableitung geschrieben. Damit läßt sich die eigentliche Definition der *Sobolev-Räume* formulieren.

Definition 1.10 (Sobolev-Räume) Seien $k \in \mathbb{N}$ und $v \in L^1_{loc}(\Omega)$. Die verallgemeinerten Ableitungen $D^\alpha v$ mögen für alle $|\alpha| \leq k$ existieren. Dann ist die *Sobolev-Norm* gegeben

durch

$$\|v\|_{W^{k,p}} := \begin{cases} \left(\sum\limits_{|\alpha| \le k} \|D^\alpha v\|_{L^p}^p \right)^{\frac{1}{p}} & \text{für } 1 \le p \le \infty \\ \max\limits_{|\alpha| \le k} \|D^\alpha v\|_{L^\infty} & \text{für } p = \infty. \end{cases}$$

Die *Sobolev-Räume* werden in beiden Fällen definiert durch

$$W^{k,p}(\Omega) := \left\{ v \in L^1_{loc}(\Omega) \,:\, \|v\|_{W^{k,p}} < \infty \right\}.$$

Satz 1.1
Der Sobolev-Raum $W^{k,p}(\Omega)$ wird zusammen mit der Norm $\|\cdot\|_{W^{k,p}(\Omega)}$ zu einem Banachraum.

Beweis. Vgl. H. W. Alt [Alt92] (Abschnitt 1.15, S. 31). □

Bemerkung 1.3 Für homogene Dirichlet-Randwerte ist noch der Raum $W_0^{k,p}(\Omega)$ von Bedeutung. Er ist definiert als Abschluß von $C_0^\infty(\Omega)$ in der $\|\cdot\|_{W^{k,p}(\Omega)}$-Norm.

Die Sobolev-Räume $W^{k,2}(\Omega)$ und $W_0^{k,2}(\Omega)$ sind Hilbert-Räume mit dem Skalarprodukt

$$(u,v)_{W^{k,2}(\Omega)} := \sum_{|\alpha| \le k} (D^\alpha u, D^\alpha v) = \sum_{|\alpha| \le k} \int_\Omega D^\alpha u D^\alpha v \, dx \tag{1.2}$$

und werden mit $H^k(\Omega)$ bzw. $H_0^k(\Omega)$ bezeichnet. Auf diesen Räumen werden für $k = 1$ die Bilinear- bzw. Linearformen der Variationsgleichnung formuliert.

1.2. Problemstellung

Betrachtet werden elliptische Randwertprobleme 2.Ordnung:

$$\begin{cases} \mathcal{L}u := -\varepsilon \Delta u + \vec{b} \cdot \nabla u + cu = f & \text{in } \Omega, \\ u = g & \text{auf } \partial\Omega, \end{cases} \tag{1.3}$$

auf einem offenen, beschränkten Gebiet $\Omega \subset \mathbb{R}^n$ und den folgenden Daten:

(i) dem Diffusionskoeffizienten $\varepsilon \in \mathbb{R}^+$,
(ii) dem Geschwindigkeitsfeld $\vec{b} \in \left(W^{1,\infty}(\Omega)\right)^n$,
(iii) dem Reaktionskoeffizienten $c \in L^\infty(\Omega)$,
(iv) und der Randfunktion $g \in L^2(\partial\Omega)$.

Gesucht ist eine Funktion $u : \Omega \to \mathbb{R}$, so daß (1.3) mit dem Quellterm $f : \Omega \to \mathbb{R}$, $f \in L^2(\Omega)$ erfüllt ist. Für $g \not\equiv 0$ bezeichnet man (1.3) auch als *inhomogenes Dirichlet-Problem*. Dieses wird zunächst auf ein *homogenes RWP* transformiert,

indem man voraussetzt, daß eine zulässige Funktion u_0 bekannt ist, welche auf dem Rand mit g übereinstimmt. Für $w := u - u_0$ ist dann

$$\begin{cases} \mathcal{L}w = f_1 & \text{in} \quad \Omega, \\ w = 0 & \text{auf} \quad \partial\Omega, \end{cases} \tag{1.4}$$

mit $f_1 := f - \mathcal{L}u_0$. Der Einfachheit halber geht man im folgenden immer von homogenen Randbedingungen aus. Für inhomogene Dirichlet-Probleme vgl. D. Gilbarg & N. S. Trudinger [GT98] (Kap. 6.3, S. 100ff).

Die physikalische Interpretation der Terme von (1.3) lassen sich z.B. bei J. Bey [Bey98] (Bemerkung 1.2.3, S. 46) nachlesen.

Definition 1.11 (Konvektionsdominantes Problem) Für $0 < \varepsilon \ll 1$ spricht man von einem *konvektionsdominanten Problem.*

Unter einer sogenannten *klassischen Lösung* elliptischer Randwertprobleme 2.Ordnung versteht man:

Definition 1.12 (Klassische Lösung) Jede Funktion $u \in C^{2,\lambda}(\Omega) \cap C(\overline{\Omega})$ mit $0 < \lambda < 1$ heißt klassische Lösung eines elliptischen Randwertproblems genau dann, wenn (1.3) bzw. (1.4) punktweise auf Ω bzw. $\partial\Omega$ erfüllt ist.

Aussagen zur Existenz und Eindeutigkeit einer solchen klassischen Lösung finden sich unter anderem in dem Buch von D. Gilbarg & N. S. Trudinger [GT98] (Kap. 6, S. 87ff). Allerdings erweist sich der klassische Lösungsbegriff für die Praxis als nicht geeignet, da die Glattheitsanforderung an den Rand des Gebietes und die Daten des Problems zu restriktiv sind und somit selten erfüllt werden. Aus diesem Grund muß der Lösungsbegriff im Hinblick auf seine Anwendbarkeit allgemeiner gefaßt werden. Dazu führt man die sogenannte *schwache Lösung* ein, die Gegenstand des folgenden Kapitels 1.3 ist.

Bemerkung 1.4 (nicht erfüllte Glattheitsanforderung) Die Beispiele 2.1.3 und 2.1.4 von W. Hackbusch [Hac86] (S. 22) zeigen, daß die starken Glattheitseigenschaften der klassischen Lösung von (1.3) bzw. (1.4) schon in den einfachsten Fällen nicht vorliegen.

1.3. Variationsformulierung und schwache Lösung

Interpretiert man das Randwertproblem (1.3) bzw. (1.4) im schwachen Sinne, so können die für eine klassische Lösung geforderten Voraussetzungen an $\mathcal{L}$ und f abgeschwächt werden. Als Folge erhält man dann Lösungen, die im allgemeinen nicht mehr in $C^2(\Omega)$ liegen, die aber physikalisch durchaus noch sinnvoll sind.

Eine Möglichkeit der schwachen Interpretation des Randwertproblems (1.4) besteht in der Umformulierung in ein sogenanntes *Variationsproblem.* Dazu multipliziert man (1.4) mit einer Testfunktion $v \in C_0^\infty(\Omega)$, integriert beide Seiten über Ω und führt auf der linken

Seite partielle Integration durch:

$$\int_\Omega \left(-\varepsilon\Delta u + \vec{b}\cdot\nabla u + cu\right) v\,dx = \int_\Omega f v\,dx$$

$$\overset{p.I.}{\Longleftrightarrow} \quad \int_\Omega \varepsilon\nabla u\nabla v\,dx - \int_{\partial\Omega} \underbrace{\varepsilon\left(\nabla u\cdot\vec{n}\right) v}_{\substack{=0,\text{ da}\\ v\in C_0^\infty(\Omega)}} ds + \int_\Omega \left(\vec{b}\cdot\nabla u + cu\right) v\,dx = \int_\Omega f v\,dx,$$

wobei $\vec{n}$ der äußere Normalenvektor ist. Nun setzt man

$$\begin{aligned} a(u,v) &:= \int_\Omega \varepsilon\nabla u\nabla v\,dx + \int_\Omega \left(\vec{b}\cdot\nabla u + cu\right) v\,dx \\ f(v) &:= \int_\Omega f v\,dx \end{aligned} \tag{1.5}$$

und vervollständigt schließlich unter Berücksichtigung der vorliegenden homogenen Dirichlet-Randbedingungen den $C_0^\infty(\Omega)$ zu $H_0^1(\Omega)$ (siehe Bemerkung 1.3). Dazu seien $(u_n)_{n\in N}$ und $(v_n)_{n\in N}$ Folgen in $C_0^\infty(\Omega)$ mit den Grenzwerten $u, v \in H_0^1(\Omega)$. Weiter sei $f \in L^2$ vorausgesetzt. Dann gilt nach Nullergänzung:

$$\begin{aligned} |f(v_n) - f(v)| &\le \|f\|_{L^2}\,\|v_n - v\|_{H_0^1(\Omega)} \to 0, \quad n\to\infty \\ |a(u_n,v_n) - a(u,v)| &= |a(u_n - u, v_n) + a(u, v_n - v)| \\ &\le \|u_n - u\|_{H_0^1(\Omega)}\,\|v_n\|_{H_0^1(\Omega)} + \|u\|_{H_0^1(\Omega)}\,\|v_n - v\|_{H_0^1(\Omega)} \\ &\to 0, \quad n\to\infty. \end{aligned}$$

Für die Wohldefiniertheit der Bilinear- bzw. Linearform von (1.5) bleibt nur noch die Linearität und Beschränktheit von a und f zu zeigen. Erstere folgt unmittelbar aus den Eigenschaften des Lebesgue-Integrals. Die Beschränktheit von f erhält man mittels Hölder-sche Ungleichung:

$$|f(v)| \le \|f\|_{L^2(\Omega)}\,\|v\|_{L^2(\Omega)} \le \|f\|_{L^2(\Omega)}\,\|v\|_{H^1(\Omega)}.$$

Die Beschränktheit von a ergibt sich ebenfalls über die Höldersche Ungleichung. Dies sei hier exemplarisch für den elliptischen Hauptteil ($\varepsilon = 1$, $b = 0$, $c = 0$) beschrieben,

$$\begin{aligned} |a(u,v)| &\le \left(\sum_{i=1}^n \int_\Omega \left|\frac{\partial u}{\partial x_i}\right|^2\right)^{\frac{1}{2}} \left(\sum_{i=1}^n \int_\Omega \left|\frac{\partial v}{\partial x_i}\right|^2\right)^{\frac{1}{2}} \\ &\le \|u\|_{H_0^1(\Omega)}\,\|v\|_{H_0^1(\Omega)}. \end{aligned}$$

Die Terme niederer Ordnung werden analog abgeschätzt.

Damit ist folgendes Lemma gezeigt:

Lemma 1.3

Seien $\Omega \subset \mathbb{R}^n$ ein beschränktes Gebiet sowie $f \in L^2(\Omega)$. Dann sind durch (1.5) eine stetige beschränkte Linearform bzw. stetige beschränkte Bilinearform auf $H_0^1(\Omega)$ bzw. $H_0^1(\Omega)\times H_0^1(\Omega)$ definiert.

Bemerkung 1.5 (zu Lemma 1.3) Man kann in der Formulierung (1.5) zu Gebieten mit lediglich Lipschitz-stetigem Rand übergehen, vgl. H. W. Alt [Alt92] (Satz 6.8, S. 208).

Diese Vorbetrachtungen motivieren die folgende Definition:

Definition 1.13 (schwache Formulierung bzw. Variationsgleichung) Die *schwache Formulierung* des homogenen Dirichlet-Randwertproblems lautet:

$$\text{Finde } u \in H_0^1(\Omega) : \; a(u,v) = f(v), \quad \forall v \in H_0^1(\Omega). \tag{1.6}$$

Die Lösung $u \in H_0^1(\Omega)$ heißt *schwache Lösung* von (1.4). Man nennt (1.6) auch die *Variationsgleichung* zu (1.4).

Gilt für die Bilinearform $a(\cdot,\cdot)$ von (1.5) neben der Beschränktheit und Stetigkeit noch die folgende Eigenschaft (für $V := H_0^1(\Omega)$)

$$\exists\, \alpha > 0 : \; a(v,v) \geq \alpha \|v\|_V^2, \quad \forall v \in V, \tag{1.7}$$

so nennt man $a(\cdot,\cdot)$ *V-elliptisch*. In diesem Fall liefert das Lax–Milgram–Lemma eine hinreichende Aussage zu Existenz und Eindeutigkeit einer schwachen Lösung.

Lemma 1.4 (Lax–Milgram)
Seien V ein Hilbert-Raum, $a(\cdot,\cdot) : V \times V \to \mathbb{R}$ eine stetige V-elliptische Bilinearform und $f : V \to \mathbb{R}$ eine stetige Linearform, d.h. es sei $f \in V^*$. Dann existiert ein eindeutig bestimmtes $u \in V$ mit

$$a(u,v) = f(v) \quad \textit{für alle } v \in V. \tag{1.8}$$

Beweis. Vgl. C. Grossmann & H.–G. Roos [GR94] (Beweis zu Lemma 3.6, S. 91ff). □

Sei für die Daten des RWP (1.3) zusätzlich noch

$$c - \frac{1}{2}\nabla \cdot \vec{b} \geq 0 \tag{1.9}$$

vorausgesetzt. Dann läßt sich zeigen, daß die Bilinearform $a(\cdot,\cdot)$ stetig ist. Die V-Elliptizität folgt aus

$$\begin{aligned} a(v,v) &= \varepsilon |v|_{H^1}^2 + \int_{\partial\Omega} \left(c - \frac{1}{2}\nabla\cdot\vec{b}v, v \right) dx + \int_{\partial\Omega} \underbrace{\left((\varepsilon \nabla v \cdot \vec{n})\, v + \frac{1}{2}\left(\vec{b}\cdot\vec{n}\right) v^2 \right)}_{=0,\ \text{da } v \in C_0^\infty(\Omega)} ds \\ &\geq \varepsilon |v|_{H^1}^2 \\ &\geq C_\varepsilon \|v\|_{H^1}^2. \end{aligned}$$

Da an v homogene Dirichlet–Randbedingungen gestellt wurden, verschwindet v auf dem Rand. Damit sind aber die Voraussetzungen von Lemma 1.4 erfüllt, und somit besitzt das RWP (1.3) eine eindeutig bestimmte verallgemeinerte Lösung $u \in H_0^1(\Omega)$, vgl. G. Lube [Lub00] (Kap. 6.2.1, S. 46ff).

1.4. Finite–Elemente–Methode

Ausgehend von der Variationsformulierung (1.6) in einem unendlich dimensionalen Sobolev-Raum $H_0^1(\Omega)$ möchte man zu einer diskreten Formulierung dieses Problems gelangen. Ziel dabei ist es, eine Näherung u_h an die Lösung u des kontinuierlichen Problems in einem endlich-dimensionalen Teilraum $V_h \subset H_0^1(\Omega)$ zu finden.

Definition 1.14 (Ritz-Galerkin–Verfahren) Die Aufgabe

$$\text{Finde } u_h \in V_h \ : \ a(u_h, v) = f(v), \quad \forall\, v \in V_h \tag{1.10}$$

heißt *Ritz-Galerkin–Verfahren* zur Variationsgleichung (1.6).

Aufgrund der Konformität $V_h \subset H_0^1(\Omega)$ lassen sich die auf $H_0^1(\Omega)$ gültigen Eigenschaften auf den Teilraum V_h übertragen. Damit folgt sofort die Existenz und Eindeutigkeit einer Lösung $u_h \in V_h$ aus der Existenz und Eindeutigkeit einer Lösung des zugehörigen Variationsproblems (1.6) .
Für eine Basis $\{\phi_1, \ldots, \phi_n\}$ von V_h kann eine approximative Lösung u_h auch wie folgt dargestellt werden:

$$u_h = \sum_{i=1}^{n} u_i \phi_i, \tag{1.11}$$

wobei $u = (u_1, \ldots, u_n)^T$ eine Lösung des linearen Gleichungssystems $Au = f$ ist mit

$$\begin{aligned} A &= (a_{ij})_{i,j=1}^n \quad \text{und} \quad a_{ij} = a(\phi_j, \phi_i), \\ f &= (f_i)_{i=1}^n \quad \text{und} \quad f_i \ = f(\phi_i). \end{aligned} \tag{1.12}$$

Satz 1.2
Das Ritz-Galerkin–Verfahren (1.10) und das lineare Gleichungssystem $Au = f$ sind äquivalent.

Beweis. Vgl. G. Lube [Lub00] (Beweis zu Satz 7.3, S. 54f). □

Die *Finite–Elemente–Methode* (FEM) kann nun als ein spezielles Ritz-Galerkin–Verfahren aufgefaßt werden, bei dem Ansatz- und Testfunktion stückweise definierte Funktionen mit kleinem Träger, meist Polynome, sind. Dadurch wird erreicht, daß das zum Ritz-Galerkin–Verfahren äquivalente Gleichungssystem dünn besetzt ist, was für das numerische Lösen erstrebenswert ist.
Die Güte der approximierten Lösung u_h hängt von der Wahl des Ansatzraumes V_h ab. Sei im folgenden $\Omega \subset \mathbb{R}^2$ vorausgesetzt. Weiterhin beschränkt man sich auf polygonal berandete Gebiete. Der endlich-dimensionale Teilraum V_h wird dann mittels Triangulierung des betrachteten Gebietes Ω konstruiert. Sei nun $\mathcal{T}_h$ $(h > 0)$ eine Familie zulässiger regulärer Triangulierungen von Ω, d.h. es gilt $\overline{\Omega} = \bigcup_{i=1}^{M} \overline{T}_i$ mit $M < \infty$ und den Eigenschaften

- (Zulässigkeit) Für paarweise verschiedene Elemente T_i und T_j $(i \neq j)$ der Triangulierung gilt: $\overline{T_i} \cap \overline{T_j}$ beschreibt entweder ein gemeinsames Randsimplex von T_i und T_j, d.h. eine vollständige Kante oder einen Eckpunkt, oder aber T_i und T_j sind durchschnittsfremd.

- (Regularität) Es existiert eine positive Konstante C mit

$$\sup_{T \in \mathcal{T}_h} \frac{h_T}{\rho_T} \leq C,$$

wobei h_T der Durchmesser und ρ_T der Inkreisradius von $T \in \mathcal{T}_h$ ist.

Auf $\mathcal{T}_h$ definiert man nun den Finite–Elemente–Raum

$$X_h^k := \left\{ v \in C\left(\overline{\Omega}\right) \mid v_{|T} \in \Pi_k\left(T\right), \quad \forall T \in \mathcal{T}_h \right\}, \quad k \geq 1 \tag{1.13}$$

der stetigen und stückweise polynomialen Funktionen vom Grad k und erhält dadurch den gesuchten endlich-dimensionalen Ansatzraum

$$V_h := X_h^k \cap H_0^1\left(\Omega\right). \tag{1.14}$$

Bemerkung 1.6 (P_1-Elemente) Die einfachste Wahl der Ansatz- und Testfunktionen sind stückweise lineare Funktionen, sogenannte P_1-*Elemente*. Damit läßt sich eine Basis des Ansatz- und Testraumes durch folgende Bedingung festlegen:

$$\phi_i \in C\left(\Omega\right) \ \wedge \ \phi_i\left(p_k\right) = \delta_{ik}, \quad i,k = 1, \ldots, n. \tag{1.15}$$

Hierbei sind p_k die Knoten der Triangulierung $\mathcal{T}_h$. Diese Funktionen nennt man auch *Hütchen-Funktionen* und die Basis eine *Knotenbasis*.

Bemerkung 1.7 (Stabilisierung konvektionsdominanter Probleme) Die Konvergenz der Galerkin–Diskretisierung wird für konvektionsdominierte Probleme numerisch instabil, da hier die diskrete Lösung u_h starke Oszillationen aufweist. In diesem Fall muß die Diskretisierung durch ein geeignetes *Upwindverfahren* stabilisiert werden. Siehe dazu unter anderem G. Lube [Lub94] bzw. A. Quarteroni & A. Valli [QV94]. In Kapitel 1.6 (S. 13) soll detaillierter auf Verfahren zur Stabilisierung eingegangen werden.

1.5. Eigenschaften der Steifigkeitsmatrix der Galerkin–FEM

In diesem Kapitel sollen Eigenschaften der mittels Galerkin–FEM erzeugten Steifigkeitsmatrix beschrieben werden. Vergleiche hierzu auch S. Le Borne [Bor99] (S. 13f) bzw. J. Bey [Bey98] (Kap. 1.4.3, S. 73ff).

Lemma 1.5 (Eigenschaften der Steifigkeitsmatrix)
Sei V ein reeller Hilbert-Raum und $V_h \subset V$ ein endlichdimensionaler Teilraum mit der Basis Φ_h. Weiter sei $a\left(\cdot,\cdot\right) : V \times V \to \mathbb{R}$ eine beschränkte Bilinearform und A_h die durch (1.12) definierte Steifigkeitsmatrix. Dann gilt:

(i) *Ist $a\left(\cdot,\cdot\right) : V \to \mathbb{R}$ symmetrisch, so ist auch A_h symmetrisch.*
(ii) *Ist $a\left(\cdot,\cdot\right) : V \to \mathbb{R}$ V-elliptisch, so ist A_h nichtsingulär und positiv definit.*

Bemerkung 1.8 (zu Lemma 1.5) Das Lemma 1.5 sichert zwar die eindeutige verallgemeinerte Lösbarkeit des RWP (1.3) in $V = H_0^1\left(\Omega\right)$ sowie der Galerkin–FEM (1.10),

genauer gilt jedoch zunächst nur

$$\begin{aligned} a(v,v) &= \varepsilon |v|_{H^1}^2 + \int_{\partial\Omega} \left(c - \frac{1}{2} \nabla \cdot \vec{b} \right) v^2 \, dx \\ &\geq \varepsilon |v|_{H^1}^2 + \int_{\Omega} \alpha v^2 \, dx \\ &= \varepsilon |v|_{H_1(\Omega)}^2 + \alpha \|v\|_{L^2(\Omega)}^2, \quad \forall v \in V := H_0^1(\Omega), \end{aligned}$$

falls

$$\left(c - \frac{1}{2} \nabla \cdot \vec{b} \right)(x) \geq \alpha \geq 0$$

gilt. Für $\varepsilon \to 0$ geht somit die Kontrolle über den Gradienten der Lösung, d.h. bzgl. der Halbnorm $|\cdot|_{H^1(\Omega)}$, verloren. Man spricht daher auch von einem *singulär gestörten Problem.*

Im Fall $\alpha > 0$ hat man also lediglich noch L^2-Stabilität. Hat das Strömungsfeld b keine Stagnationspunkte, d.h. $|b(x)| > 0$ für alle $x \in \overline{\Omega}$, und keine geschlossenen Stromlinien in $\overline{\Omega}$, kann man durch geeignete Transformation den Fall $\alpha > 0$ erzwingen, vgl. GOERING ET AL. [GFLR83] (Lemma 3.2, Theorem 3.2).

Diese Überlegungen und numerischen Rechnungen für die Galerkin–FEM legen eine geeignete Stabilisierung der Galerkin–FEM nahe (vgl. dazu Kap. 1.6, S. 13).

Durch die Wahl der Basis Φ_h werden numerisch relevante Eigenschaften, wie z.B. das Besetzungsmuster und die Kondition von A_h, wesentlich beeinflußt. Durch die Verwendung von Basisfunktionen ϕ_j^h mit lokalem Träger erhält man so eine dünn besetzte Steifigkeitsmatrix A_h, wie das z.B. bei den P_1-Elemente der Fall ist (siehe Bemerkung 1.6).

Generell ist die Anzahl der nichtverschwindenden Einträge der mit Hilfe einer Knotenbasis Φ_h konstruierten Matrix A_h durch eine von der Triangulierung $\mathcal{T}_h$ unabhängigen Konstanten C beschränkt. Dies gilt für jede zulässige Triangulierung $\mathcal{T}_h$.

Für die im weiteren vorgestellten Numerierungsstrategien und die iterativen Löser ist die folgende Eigenschaft der Steifigkeitsmatrix von Bedeutung:

- **Zusammenhang zwischen der Triangulierung und der Struktur der Steifigkeitsmatrix**
  Bei finiten Elementen mit einer Knotenbasis korrespondieren die Unbekannten des linearen Gleichungssystems $Au = f$ mit den Knoten der Triangulierung, und die Kanten der Triangulierung entsprechen den Nicht-Nulleinträgen der Steifigkeitsmatrix A.

Weitere Eigenschaften der Steifigkeitsmatrix sollen im Kapitel 1.6.1 (S. 14) beschrieben werden.

1.6. Stabilisierung

Das bei der Ritz-Galerkin–Diskretisierung entstehende Gleichungssystem ist, wie in Bemerkung 1.8 (S. 12) angemerkt, im singulär gestörten Fall $0 < \varepsilon \ll 1$ eventuell nicht

stabil lösbar. Deswegen werden geeignete Stabilisierungstechniken benötigt. In diesem Abschnitt sollen solche Techniken zur Stabilisierung der Lösung singulär gestörter Probleme eingeführt werden. Zunächst wird ein vereinfachtes Upwind-Schema betrachtet, welches bisher in der Literatur im Zusammenhang mit flußorientierten Numerierungsstrategien untersucht wurde. Siehe u.a. J. Bey [Bey98] und S. Le Borne [Bor99]. Dann soll die im folgenden betrachtete Stabilisierungstechnik, eine Kombination einer verbesserten Upwind-Technik (SUPG-Stabilisierung) in Verbindung mit einer Shock-Capturing–Technik nach R. Codina [CS99] vorgestellt werden.

1.6.1. Stabilisierung bezüglich der diskreten Maximumnorm

Aus Bemerkung 1.8 (S. 12) folgt, daß für $\varepsilon \to 0$ die Stabilität der Galerkin–FEM–Lösung bezüglich der H^1-Halbnorm verlorengeht. Eine grundlegende Idee besteht darin, Stabilität bezüglich der diskreten Maximumnorm ggf. durch Abänderung von A_h zu erreichen.
Sei $v \in C_2(\Omega) \cap \overline{C_1(\Omega)}$, sei $\{\phi_i\}_{i=1}^N$ eine Knotenbasis und $v = \sum_{i=1}^N V_i \phi_i$. Dann bezeichne $R_h v \in \mathbb{R}^N$ die Einschränkung von v auf die Knoten, d.h. es gilt $\{R_{h,i} v\}_{i=1}^N = V_i$. Auf dem $\mathbb{R}^N$ sei folgende diskrete Maximumnorm eingeführt:

$$\| W \|_{\infty,\mathcal{T}_h} := \max_{i=1,\dots,N} |W_i|.$$

Dabei sind für $w \in C^2(\Omega)$ die W_i durch $W_i = R_{h,i} w$ gegeben.

Definition 1.15 (stabile Lösbarkeit) Eine FEM heißt *stabil in der Maximumnorm*, falls für den Vektor $W \in \mathbb{R}^N$ aus

$$AW = F$$

die Existenz einer von h unabhängigen Konstanten C_S folgt mit

$$\| W \|_{\infty,\mathcal{T}_h} = \left\| A^{-1} F \right\|_{\infty,\mathcal{T}_h} \leq C_S \, \| F \|_{\infty,\mathcal{T}_h}.$$

Wenn die Matrix A eine sogenannte M-Matrix ist, so ist das FEM–Schema stabil nach Definition 1.15. Vgl. G. Lube [Lub00] (Kap. 1.4, S. 14ff).

Definition 1.16 (M-Matrix) Eine nicht-singuläre Matrix A heißt *M-Matrix*, wenn sie folgende Eigenschaften besitzt:

(i) $A^{-1} \geq 0$ (komponentenweise),
(ii) $a_{ii} > 0$ für alle $1 \leq i \leq n$,
$a_{ij} \leq 0$ für alle $1 \leq i, j \leq n,\ i \neq j$.

Die Galerkin–FEM hat die M-Eigenschaft, wenn für die lokale Peclet–Zahl $Pe_T := \frac{|\vec{b}|_{T,\infty} h_T}{\varepsilon} > 2$ gilt. Man kann jedoch die M-Eigenschaft bei Zerlegungen vom schwach spitzen Typ (d.h. alle Innenwinkel von Dreicken sind $\leq \pi/2$) und stückweise linearen Elementen noch sichern, wenn man eine Upwind- Diskretisierung des konvektiven Anteils $(b \cdot \nabla u, v)$ vornimmt. Vergleiche hierzu H.–G. Roos, M. Stynes & L. Tobiska [RST96] (Kap. III.3.1), J. Bey [Bey98] (Kap. 5.1) bzw. S. Le Borne [Bor99] (Kap. 2.1.1).
Es sei vermerkt, daß mathematische Untersuchungen zu flußorientierten Numerierungsstrategien bislang nur auf derartigen Upwind-Techniken basieren. Unter Verwendung dieser Techniken ergeben sich folgende weitere Eigenschaften der Steifigkeitsmatrix:

- **Zusammenhang zwischen der Konvektionsrichtung und der Größe der Matrixeinträge**
  Ein Eintrag a_{ij} der Steifigkeitsmatrix A ist groß, wenn die korrespondierende Kante $\{j, i\}$ der Triangulierung in Konvektionsrichtung oder nahezu parallel dazu liegt. Vorausgesetzt ist dabei, daß der konvektions-dominante Fall betrachtet wird.

Die Beziehung zwischen der Triangulierung, der Konvektionsrichtung und der daraus resultierenden Struktur der Steifigkeitsmatrix mit der entsprechenden absoluten Größe der Matrixeinträge zeigt die Abbildung (1.1), vgl. S. Le Borne [Bor99] (Abb. 2.3, S. 15).

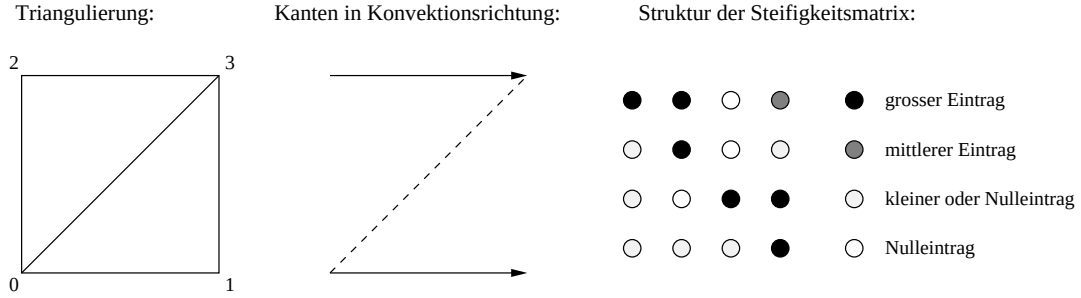


Abbildung 1.1.: Zusammenhang zwischen dem Gitter, der Konvektionsrichtung $\left(b = (1,0)^T\right)$ und der Struktur der Steifigkeitsmatrix

Bei einer Abschätzung der absoluten Größen der Einträge kann man im konvektionsdominanten Fall ($\varepsilon \ll 1$) den Diffusionsterm $\varepsilon \nabla\phi_j \nabla\phi_i$ vernachlässigen. Damit ergeben sich die Einträge des Konvektionsanteils $a_{i,j}^C$ in der Steifigkeitsmatrix mit der hybriden Upwind-Formulierung (vgl. S. Le Borne [Bor99]) folgendermaßen: $a_{i,j}^C$ ist

groß, wenn die Kante (i, j) parallel zur Konvektionsrichtung liegt,
d.h. $\angle\{(i,j)\,; b\,(i)\} = 0$,

mittel, wenn der Winkel $\angle\{(i,j)\,; b\,(i)\} \leq \frac{\pi}{2}$,

klein, wenn der Winkel $\angle\{(i,j)\,; b\,(i)\} > \frac{\pi}{2}$,

Null, wenn i und j nicht durch eine Kante der Triangulierung verbunden sind.

Dabei sind i, j Knoten der Triangulierung, und $b(i)$ bezeichnet den vom Knoten i ausgehenden Strömungsvektor.
Mit Hilfe dieser Überlegungen sieht man sehr deutlich, daß durch eine Numerierung der Gitterpunkte in Flußrichtung die betragsmäßig großen Einträge des Konvektionsanteils der Steifigkeitsmatrix Dreiecksmatrixgestalt annehmen. Für eine derart vorkonditionierte Matrix läßt sich aber z.B. ein einfaches Gauß–Seidel–Verfahren als Löser einsetzen, von dem man sehr gute numerische Resultate erwarten kann (siehe dazu auch Kap. 3, S. 31).
Dem grundlegenden Vorteil der Maximumnormstabilität derartiger Verfahren steht allerdings als gravierender Nachteil gegenüber, daß die Verfahren nicht vom residualen Typ sind, d.h. für Lösungen u des RWP (1.3) gilt nicht

$$\tilde{a}_h(u, v) = f(v), \quad \forall v \in V_h,$$

wobei $\tilde{a}_h(\cdot,\cdot)$ die abgeänderte Bilinearform ist. Es gilt lediglich

$$\lim_{h\to 0} \tilde{a}_h(u,v) = a(u,v).$$

Ein typisches Konvergenzresultat ist H.–G. Roos, M. Stynes & L. Tobiska [RST96] (Thm. 3.20)

$$\sqrt{\varepsilon}\,|u-u_h|_{H^1(\Omega)} + \sqrt{\alpha}\,\|u-u_h\|_{L^2(\Omega)} \leq \frac{Ch}{\sqrt{\varepsilon}}\left(\|u\|_{W^{2,2}(\Omega)} + \|f\|_{W^{1,p}(\Omega)}\right).$$

Abhilfe schaffen hier stabilisierte Galerkin–Verfahren vom residualen Typ.

1.6.2. SUPG-Stabilisierung

Mit Hilfe einer verbesserten Upwind-Technik (SUPG-Stabilisierung) wird stabilere Lösbarkeit des Gleichungssystems gesichert. Da die so erhaltene Matrix keine M-Matrix ist, können in inneren und parabolischen Grenzschichten noch kleine Oszillationen senkrecht zum Konvektionsfeld $\vec{b}$ auftreten. Um diese zu beseitigen, wird eine Shock-Capturing Technik nach R. Codina [CS99] verwendet, die Thema des folgenden Abschnittes 1.6.3 (S. 17) sein soll. Es sei jedoch betont, daß trotz SUPG- und Shock–Capturing–Stabilisierung die entstehende Matrix nur „approximativ“ eine M-Matrix ist.

Die Idee der SUPG-Stabilisierung ist es, in Bereichen großer Peclet–Zahlen ($Pe > 1$) künstliche Diffusivität in Richtung des Konvektionsfeldes zu addieren.

Insgesamt erhält man die stabilisierte Bi- bzw. Linearform

$$\begin{aligned} a_{SG}(u,v) &:= a(u,v) + a_S(u,v), \\ f_{SG}(v) &:= f(v) + f_S(v), \end{aligned}$$

wobei die SUPG-Stabilisierungsterme gegeben sind durch

$$\begin{aligned} a_S(u,v) &:= \sum_{T\in\mathcal{T}_h} \delta_T \left(-\nabla\cdot(\varepsilon\nabla u) + \vec{b}\cdot\nabla u + cu, \vec{b}\cdot\nabla v\right)_T, \\ f_S(v) &:= \sum_{T\in\mathcal{T}_h} \delta_T \left(f, \vec{b}\cdot\nabla v\right)_T. \end{aligned}$$

Durch die Wahl

$$\delta_T := \frac{h_T}{2\,|b|_{T,\infty}} \min\left(1, Pe_T\right),$$

wobei $Pe_T = \frac{h_T|\vec{b}|_T}{\varepsilon}$ die lokale Peclet–Zahl bezeichnet, kann die SUPG-Stabilisierung lokal gesteuert werden, d.h. nur auf Elementen mit $Pe_T > 1$ wird wesentlich künstliche Diffusivität addiert. Dann lautet die SUPG-stabilisierte diskrete Formulierung des elliptischen Randwertproblems 1.4

$$\begin{cases} \textit{Finde } u \in V_h \subset V \equiv H_0^1(\Omega), \text{ so daß} \\ \qquad a_{SG}(u,v) = f_{SG}(v) \text{ für alle } v \in V_h \text{ gilt.} \end{cases} \tag{1.16}$$

Bemerkung 1.9 Für die eindeutige und stabile Lösbarkeit des stabilisierten Problems sei verwiesen auf H.–G. Roos, M. Stynes & L. Tobiska [RST96] (S. 231).

1.6.3. Discontinuity Capturing

Um kleinere Oszillationen senkrecht zum Strömungsfeld zu beseitigen, ist es hilfreich eine Shock–Capturing–Technik zu verwenden. Im folgenden wird ein Vorschlag von R. Codina [CS99] vorgestellt.
Die Idee ist es, künstliche Viskosität senkrecht zum Konvektionsfeld zu addieren. Die Größe der zusätzlichen Viskosität beträgt

$$\varepsilon_{dc} := \frac{1}{2}\xi_c h \frac{|R(u)|_T}{|\nabla u|_T}.$$

Dabei wird der Parameter ξ_c elementweise berechnet nach

$$\xi_c := \max\left\{0, C_{dc} - \frac{2\varepsilon}{\left|\vec{b}^*\right|_T h}\right\} \quad \text{mit}$$
$$\vec{b}^* := \frac{1}{|\nabla u|_T^2}\left(\vec{b}\cdot\nabla u + cu - f\right)\nabla u.$$

Aufgrund numerischer Experimente schlägt R. Codina für lineare Elemente die Konstante $C_{dc} = 0.7$ vor [CS99].
Die Shock–Capturing–Methode besteht nun in der Addition des zusätzlichen Terms

$$\sum_{T\in\mathcal{T}_h}\left[\varepsilon_{dc}\left(\nabla u, \nabla v\right)_T + \left(\varepsilon_{sl} - \varepsilon_{dc}\right)\frac{1}{\left|\vec{b}\right|_T^2}\left(\vec{b}\cdot\nabla u, \vec{b}\cdot\nabla v\right)_T\right] \quad \text{mit}$$
$$\varepsilon_{sl} := \max\left\{0, \varepsilon_{dc} - \varepsilon_{supg}\right\}, \quad \text{wobei}$$
$$\varepsilon_{supg} := \frac{\delta_T}{\left|\vec{b}\right|_T^2} \quad \text{ist.}$$

Nun soll die Wirkung dieses Schemas kurz erläutert werden. Im Falle $\varepsilon_{dc} < \varepsilon_{supg}$ ist $\varepsilon_{sl} = 0$. Dann wird isotrop künstliche Diffusivität der Stärke ε_{dc} addiert. Ferner wird in Strömungsrichtung künstliche Diffusivität der Stärke ε_{dc} subtrahiert. Damit soll eine Überdämpfung in Strömungsrichtung vermieden werden. Die Gesamtwirkung von SUPG- und SC-Stabilisierung besteht darin, daß in Strömungsrichtung künstliche Diffusivität der Stärke ε_{supg} und in Crosswind-Richtung Diffusivität der Stärke ε_{dc} addiert wird.
Im Falle $\varepsilon_{dc} > \varepsilon_{supg}$ ist $\varepsilon_{sl} > 0$. Dann wird isotrop künstliche Diffusivität der Stärke ε_{dc} addiert und gleichzeitig in Strömungsrichtung künstliche Diffusivität der Stärke ε_{supg}, das ist gerade die Wirkung der SUPG-Stabilisierung, subtrahiert. Auch damit soll eine Überdämpfung in Strömungsrichtung vermieden werden. Als Gesamtwirkung von SUPG- und SC-Stabilisierung wird dann sowohl in Strömungsrichtung als auch in Crosswind-Richtung Diffusivität der Stärke ε_{dc} addiert. Vgl. T. Knopp [Kno99] (Kap. 4.3, S. 72).

Mit der zusätzlich eingeführten Form

$$a_{DC}(u,v) := \sum_{T\in\mathcal{T}_h} [\varepsilon_{dc}(\nabla u, \nabla v)_T + (\varepsilon_{sl} - \varepsilon_{dc}) \frac{1}{\left|\vec{b}\right|_T^2} (\vec{b}\cdot\nabla u, \vec{b}\cdot\nabla v)_T]$$

lautet die Shock–Capturing– und SUPG–stabilisierte diskrete Formulierung des elliptischen Randwertproblems (1.4)

$$\begin{cases} \quad Finde\ u \in V_h \subset V \equiv H_0^1(\Omega),\ \text{so daß} \\ a_{SG}(u,v) + a_{DC}(u,v) = f_{SG}(v)\ \text{für alle}\ v \in V_h\ \text{gilt.} \end{cases} \tag{1.17}$$

Für eine mathematische Analyse dieser Shock–Capturing–Technik sei verwiesen auf T. Knopp, G. Lube und G. Rapin [KLR].

2. Einführung in die Graphentheorie

In diesem Kapitel wird eine kurze Einführung in die Notation der Graphentheorie gegeben. Der Schwerpunkt liegt dabei auf den für diese Arbeit erforderlichen Definitionen und Algorithmen zum Durchnumerieren von zyklischen und azyklischen Graphen. Die verwendeten Standardalgorithmen wie Tiefensuchen und Tarjan–Algorithmus zum Auffinden starker Zusammenhangskomponenten werden am Ende des Kapitels erläutert. Der von W. HACKBUSCH entwickelte Algorithmus zur Numerierung azyklischer Graphen soll dagegen Thema des nächsten Kapitels sein.

2.1. Grundlagen

Definition 2.1 (Endliche gerichtete Graphen) Sei V eine endliche Menge und $E \subset V \times V$ eine Menge von geordneten Paaren $(u, v) \in V \times V$. Dann heißt $G = \{V, E\}$ ein *(endlicher) gerichteter Graph* oder auch *(endlicher) Digraph* (für directed graph). Die Elemente von V bezeichnet man als *Knoten* (engl. Vertex oder Vertices) und die Paare $(u, v) \in E$ als *Kanten* (engl. Edge oder Edges) des Graphen G.

Bemerkung 2.1 (ungerichtete Graphen) Existiert keine Ordnung der Paare $(u, v) \in V \times V$, so spricht man von einem *ungerichteten Graphen*.

Für die im weiteren vorgestellten Verfahren werden ausschließlich endliche gerichtete Graphen betrachtet.

Zur graphischen Darstellung eines gerichteten Graphen $G = \{V, E\}$ faßt man seine Knoten als Punkte einer Ebene bzw. des Raumes auf und verbindet für jede der Kanten $(u, v) \in E$ die entsprechenden Knoten u und v durch einen in Richtung v zeigenden Pfeil.

Die Graphen, die aus der Steifigkeitsmatrix generiert werden, lassen sich bereits insofern einschränken, als daß sie weder Kanten von einem Knoten auf sich selbst enthalten werden, sogenannte *Schlingen* oder *self-loops* $\{(2, 2)\}$, noch Mehrfachkanten $\{(4, 5); (5, 4)\}$. Allerdings können isolierte Knoten $\{7\}$ auftreten (siehe Abb. 2.1, S. 20).

Gerichtete Graphen können unter anderem zur Darstellung von Abhängigkeiten genutzt werden. Beeinflußt z.B. eine Operation u eine Operation v, so wird dies durch eine von u nach v gerichtete Kante im Graphen symbolisiert. D.h. jede Kante $(u, v) \in E$ läßt sich so interpretieren, daß v erst nach u abgearbeitet werden darf. Voraussetzung für eine solche Interpretation ist, daß der vorliegende Graph keine sogenannten *Zyklen* enthält.

Definition 2.2 (Wege und Zyklen in gerichteten Graphen) Sei $G = \{V, E\}$ ein gerichteter Graph.

(i) Ein Knoten $v \in V$ heißt *Nachfolger* eines Knotens $u \in V$, wenn $(u, v) \in E$ gilt. In diesem Fall heißt u *Vorgänger* von v.

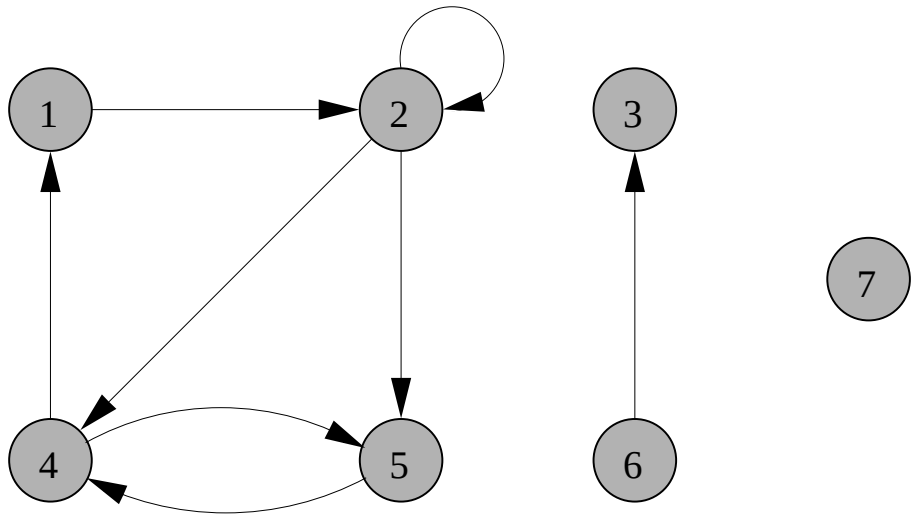


Abbildung 2.1.: Beispielgraph

(ii) Ein *Weg* in G ist eine endliche Folge $P = (u_0, \ldots, u_n)$ von Knoten $u_0, \ldots, u_n \in V$, wobei u_k Nachfolger von u_{k-1} für $1 \leq k \leq n$ ist. In diesem Fall bezeichnet man u_0 und u_n als *Anfangs-* bzw. *Endknoten* von P und sagt, u_0 wird durch P mit u_n verbunden. P heißt Weg von u_0 nach u_n.

(iii) Ein Weg P in G, dessen Anfangs- und Endknoten übereinstimmen, heißt *Zyklus* in G.

(iv) Existieren in G keinerlei Zyklen, so heißt G *azyklisch* oder *zyklusfrei*.

Die im Beispielgraph (siehe Abb. 2.1, S. 20) auftretenden Zyklen sind z.B.:
$\{(2,2); (1,2,4,1); (1,2,2,4,1)\}$.

Bemerkung 2.2 Jeder Knoten kann einen oder mehrere Nachfolger bzw. Vorgänger besitzen. Jeder Knoten u in einem Zyklus P hat mindestens einen Vorgänger und mindestens einen Nachfolger.

Definition 2.3 (Minimaler Zyklus) Ist $P = (u_0, \ldots, u_n)$ ein Zyklus in $G = \{V, E\}$, so heißt dieser *minimal* genau dann, wenn die u_i, $i = 1, \ldots, n$ paarweise voneinander verschieden sind, d.h. es gibt keine echte Teilmenge von P, die selbst wieder ein Zyklus ist.

Wie bereits erwähnt, ist für gerichtete, azyklische Graphen eine sequentielle Abarbeitung der einzelnen Knoten möglich. Das bedeutet, daß sich in diesem Fall für die N Knoten eines Graphen $G = \{V, E\}$ eine eineindeutige Zuordnung zu der Indexmenge $I = \{1, \ldots, N\}$ finden läßt.

Lemma 2.1 (Numerierung azyklischer, gerichteter Graphen)
Sei $G = \{V, E\}$ ein azyklischer, gerichteter Graph mit N Knoten. Dann können die Knoten von G so in einer Reihenfolge $u_1, u_2, \ldots, u_N$ angeordnet werden, daß $\forall (u_i, u_j) \in E$ gilt: $i < j$.

Bemerkung 2.3 (zur Umkehrung des Lemmas) Die Umkehrung dieses Lemmas gilt ebenfalls, d.h. G ist genau dann azyklisch, wenn eine solche Numerierung existiert.

Dies bedeutet, daß die Nummer jedes Knotens größer ist als die Nummern aller seiner Vorgänger und kleiner als die Nummern aller seiner Nachfolger. Ist eine Numerierung gemäß des Lemmas gegeben, so sagt man auch, *G ist in Richtung seiner Kanten* numeriert.
Die in dieser Arbeit betrachteten Konvektionsgraphen werden wegen der auftretenden Wirbel in der Regel nicht azyklisch sein.
Deshalb muß man sich für den zyklischen Fall eine neue Strategie überlegen und führt darum folgende Relation ein:
Die Existenz eines Weges P von u_0 nach u_n läßt sich in ungerichteten Graphen als Äquivalenzrelation auffassen, dagegen entfällt für gerichtete Graphen die Symmetrieeigenschaft. D.h. sei $G = \{V, E\}$ ein gerichteter Graph mit $u, v \in V$. Dann gilt: $u \sim v$ bzgl. $G \Leftrightarrow \exists P = (u, \ldots, v)$ mit $P \subset V$.
In gerichteten Graphen ist diese Relation nicht symmetrisch, d.h aus der Existenz eines Weges $P = (u, \ldots, v)$ folgt nicht notwendigerweise die Existenz eines Weges $P' = (v, \ldots, u)$. Allerdings gilt:
Sei $P \subset V$ ein minimaler Zyklus in $G = \{V, E\}$. Für alle $u, v \in P$ gilt $u \sim v$ und $v \sim u$ mit $P_1 = (u, \ldots, v)$ und $P_2 = (v, \ldots, u)$ sowie $P_1 \cup P_2 = P$.

Definition 2.4 (starke Zusammenhangskomponenten) Sei $G = \{V, E\}$ ein gerichteter Graph. Eine *starke Zusammenhangskomponente (Zshk)* von G ist eine nichtleere Teilmenge $C \subset V$, derart, daß für je zwei Knoten $u, v \in C$ mit $u \neq v$ ein Weg P_1 von u nach v und ein Weg P_2 von v nach u existiert, die ausschließlich aus Knoten in C bestehen, d.h. $u \sim v$ und $v \sim u$. Eine Zusammenhangskomponente $C \subset V$ heißt *maximal*, wenn nach Hinzufügen eines beliebigen Knotens $u \in V \setminus C$ die Menge $C \cup \{u\}$ keine Zusammenhangskomponente mehr ist.

In dem Beispielgraphen (siehe Abb. 2.1, S. 20) existieren folgende starken Zusammenhangskomponenten: $\{(1, 2, 4, 5); (3); (6); (7)\}$. Man beachte, daß es sich bei {(3,6)} nicht um eine starke Zusammenhangskomponente handelt, sondern lediglich um eine sogenannte *schwache Zusammenhangskomponente.*

Bemerkung 2.4 (schwache Zusammenhangskomponente) Bei schwachen Zusammenhangskomponenten gilt nur eine der Relationen $u \sim v$, $v \sim u$. In unserem Fall sind nur starke Zusammenhangskomponenten von Interesse.

Wie man leicht sieht, müssen die maximalen (starken) Zusammenhangskomponenten disjunkt sein. Es gilt:

Satz 2.1 (Eindeutige Zerlegung in maximale Zshk)
Jeder gerichtete Graph $G = \{V, E\}$ kann auf eindeutige Weise in maximale Zusammen-

hangskomponenten zerlegt werden, d.h. es existiert eine bis auf die Numerierung der Komponenten eindeutige Zerlegung der Form

$$V = C_1 \cup C_2 \cup \ldots \cup C_M, \quad M \geq 1,$$

wobei $C_1, C_2, \ldots, C_M$ maximale und daher paarweise disjunkte Zusammenhangskomponenten von G sind.

Lemma 2.2 (Maximale Zshk azyklischer Graphen)
Die maximalen Zusammenhangskomponenten eines azyklischen Graphen G bestehen aus jeweils einem einzigen Knoten. Umgekehrt gilt: Ein Graph G, der keine Zyklen der Form $P = (u, u)$ enthält und dessen maximale Zusammenhangskomponenten aus jeweils einem Knoten bestehen, ist azyklisch.

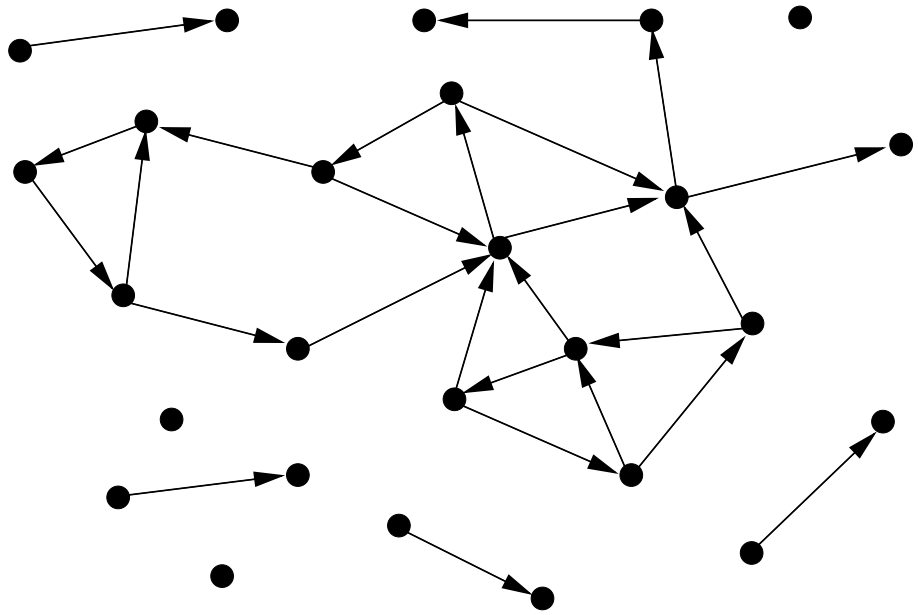


Abbildung 2.2.: Beispielgraph vor der Zerlegung in max. Zshk.

Sei $P = (u_1, \ldots, u_k)$ ein Zyklus in $G = \{V, E\}$. Dann lassen sich diese k Knoten zu einer Gruppe zusammenfassen, die ihrerseits einfach als ein weiterer Knoten $u_P \in G$ betrachtet wird. Die Kanten zwischen Knoten aus P werden aus dem Graphen herausgenommen und alle Kanten von Knoten aus $V \setminus P$ zu Knoten aus P werden nun mit u_P verbunden. Man zieht sozusagen den Zyklus P auf einen einzigen Knoten zusammen. Dieser neue Knoten u_P kann seinerseits durchaus wieder Teil eines Zyklus sein, den man eventuell in einem weiteren Schritt ebenfalls zusammenziehen könnte. Verfährt man so fort, erhält man nach endlich vielen Schritten einen sogenannten *Metagraphen* G', der azyklisch ist.

Wie man sieht, bestehen die weiter oben definierten maximalen Zusammenhangskomponenten entweder aus einem einzigen Knoten (*einfache Zusammenhangskomponente*) oder aber aus einem oder mehreren Zyklen. D.h. alle Elemente einer maximalen Zusammenhangskomponente sind Teil eines oder mehrerer Zyklen, wobei sämtliche Knoten des je-

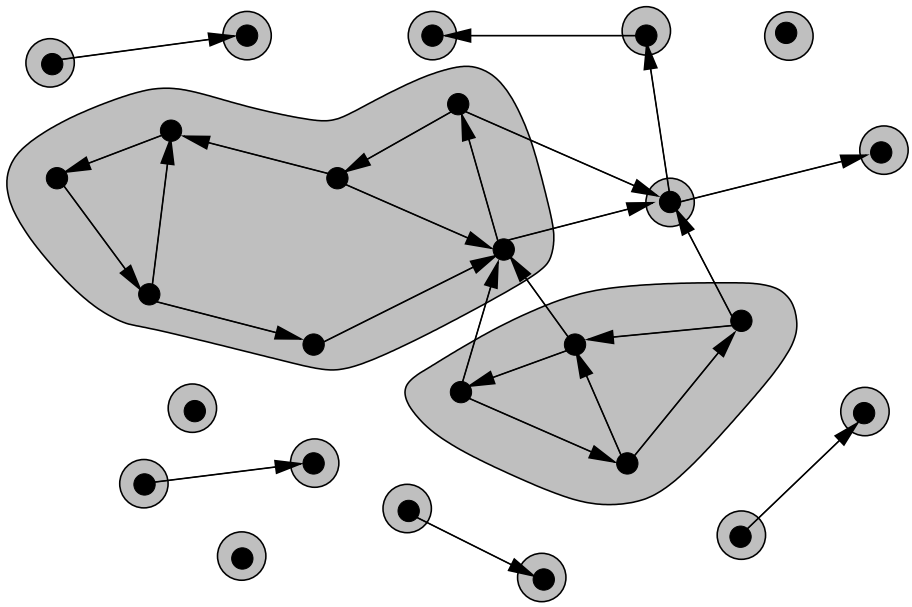


Abbildung 2.3.: Zerlegung des Graphen aus Abb. 2.2 in max. Zshk.

weiligen Zyklus oder der jeweiligen Zyklen ebenfalls in dieser maximalen Zusammenhangskomponente liegen. Damit läßt sich die oben eingeführte Strategie des Zusammenziehens von Zyklen verallgemeinern, indem man lediglich die Zusammenhangskomponenten zu einem Knoten zusammenzieht. Dieser dadurch entstehende Knoten kann nicht wieder Teil einer Zusamenhangskomponente sein, da sonst die ursprüngliche Zusammenhangskomponente nicht maximal gewesen wäre, d.h. man wäre bei dieser Stratgie bereits nach einem Durchlauf fertig. Der dabei entstehende *Metagraph* G läßt sich also wie folgt definieren:

Definition 2.5 (Metagraph) Sei $G = \{V, E\}$ ein gerichteter Graph. Die Menge der maximalen Zusammenhangskomponenten von G bezeichnet man mit V' und definiert $E' \subset V' \times V'$ durch

$$E' = \{(C_1, C_2) \in V' \times V' \mid C_1 \neq C_2;\ \text{es existieren } u \in C_1, v \in C_2 \text{ mit } (u, v) \in E.\}$$

Den gerichteten Graphen $G' = \{V', E'\}$ nennt man den *Metagraphen* von G.

Lemma 2.3 (Zyklusfreiheit des Metagraphen)
Der Metagraph G' eines gerichteten Graphen G is azyklisch.

2.2. Darstellung von Graphen

Für die konkrete Durchführung von Algorithmen auf Graphen ist eine geeignete Datenstruktur wünschenswert. Neben Kantenlisten, Inzidenz- und Adjazenzlisten sind für die im weiteren zu betrachtenden Algorithmen vor allem die sogenannten Adjazenzmatrizen zur internen Darstellung des Graphen wichtig.

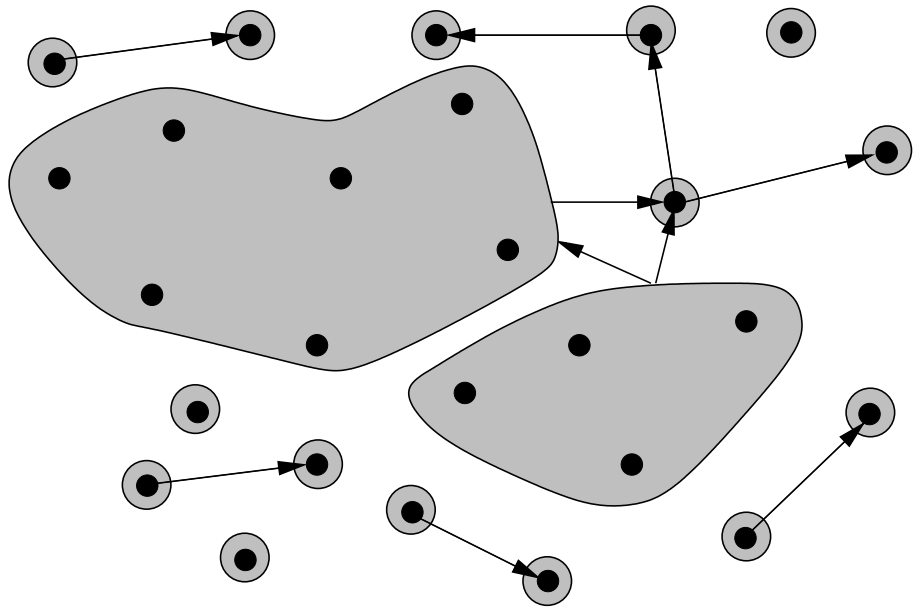


Abbildung 2.4.: Der zu Abb. 2.2 korrespondierende Metagraph

Definition 2.6 (Adjazenzmatrix) Sei $G = \{V, E\}$ ein Graph mit $|V| = n$ und A eine $(n \times n)$-Matrix mit Einträgen wie folgt:

$$a_{uv} := \begin{cases} 1 & \text{falls } (u, v) \in E, \\ 0 & \textit{sonst.} \end{cases}$$

Dann heißt A die zu G zugehörige *Adjazenzmatrix.*

Statt der 1-Einträge in der Matrix ließe sich auch eine Gewichtung der einzelnen Kanten des Graphen G definieren.

Definition 2.7 (Adjazenzmatrizen für gewichtete Graphen) Sei $G = \{V, E\}$ ein Graph mit Kantengewichtung $\omega : E \to \mathbb{R}$, dann hat die zugehörige Adjazenzmatrix $A = (a_{uv})_{u \in \{1,\ldots,|V|\}, v \in \{1,\ldots,|V|\}}$ folgende Gestalt:

$$a_{uv} := \begin{cases} \omega(u, v) & \text{falls } (u, v) \in E, \\ 0 & \textit{sonst.} \end{cases}$$

Hat man nun den umgekehrten Fall vorliegen, daß aus einer bereits vorhandenen Matrix A der entsprechende Graph konstruiert werden soll, muß dazu ein geeignetes Kriterium entwickelt werden. Dieses wird eine wichtige Rolle für die Effizienz der zu entwickelnden Verfahren spielen. In Kapitel 3.2 (S. 32) soll detaillierter auf mögliche Kriterien eingegangen werden.

2.3. Graphenalgorithmen

2.3.1. Tiefensuche

Durchsuchen, durchnumerieren oder durchmustern ist Grundlage zahlreicher Graphenalgorithmen. Zwei Standard-Suchstrategien dabei sind die Breiten- und die Tiefensuche (**B**readth **F**irst **S**earch und **D**epth **F**irst **S**earch). BFS durchsucht den Graphen $G = \{V, E\}$ „der Breite nach“: je weiter die Punkte vom Ausgangspunkt s entfernt sind, desto später werden sie betrachtet. Der Graph wird sozusagen wellenförmig duchsucht. DFS sucht dagegen so tief wie möglich in den Graphen hinein: Wenn man von einem gerade erreichten Punkt zu einem noch nicht markierten Punkt w gelangt, sucht man sofort von w aus weiter. So wird also vom Ausgangspunkt s zunächst ein maximaler Weg konstruiert.

Beide Algorithmen benötigen für die Durchnumerierung eines Graphen $G = \{V, E\}$ einen Zeitaufwand von $O(|E|)$. In der Regeln werden wir es allerdings mit dünn besetzten Graphen zu tun haben, d.h. es gilt $|E| = \Theta(|V|)$, und deshalb gilt in diesem Fall $O(|E|) \approx O(|V|)$.

Wie sich herausgestellt hat, eignet sich für unsere Zwecke die Tiefensuche eher als die Breitensuche, da bei letzterer ein zusätzliches Numerieren der Wellenfronten erforderlich wird, vgl. P. Bastian & G. Wittum [BW94] bzw. J. Bey & G. Wittum [BW95].

Gegeben sei ein Graph $G = \{V, E\}$ und gesucht ist eine Numerierung der Knoten des Graphen G.

Algorithmus: DFS

```
(1)  N ← 0
(2)  for each vertex v ∈ V do
(3)     nr(v) ← 0
(4)     pnr(v) ← 0
(5)     for each predecessor u from v do
(6)        pnr(v) ← pnr(v) + 1
(7)     od
(8)  od
(9)  for each vertex v ∈ V do
(10)    if (nr(v) = 0 and pnr(v) = 0) then
(11)       DFS − Visit(v)
(12)    fi
(13) od
```

Durch nr werden die Knoten des Graphen G in derjenigen Reihenfolge numeriert, in der sie das erste Mal erreicht wurden. Nach der Initialisierung gibt $pnr(v)$ die Anzahl der Vorgänger eines Knotens $v \in V$ an, während des eigentlichen Suchprozesses enthält $pnr(v)$ die Anzahl der noch nicht besuchten Vorgänger von v. Anhand der Bedingung $nr(v) = 0$ läßt sich feststellen, ob der jeweilige Knoten schon besucht wurde. Falls ein Knoten noch nicht besucht wurde und auch keine Vorgänger besitzt, wird für ihn die rekursive

Numerierungsfunktion $DFS - Visit$ aufgerufen, die die eigentliche Numerierung (mittels des globalen Zählers N) vornimmt und gegebenenfalls wiederum sich selbst aufruft, falls noch nicht numerierte Nachfolger von v existieren.

Algorithmus: $DFS - Visit$

```
(1) N ← N + 1
(2) nr(v) ← N
(3) for each successor w from v do
(4)    pnr(w) ← pnr(w) − 1;
(5)    if (pnr(w) = 0) then
(6)       DFS − Visit(w)
(7)    fi
(8) od
```

Satz 2.2 (Korrekheit von *DFS*)
Der Algorithmus DFS erzeugt für jeden azyklischen, gerichteten Graphen G eine Numerierung in Richtung seiner Kanten. Dabei wird jede Kante aus G genau einmal durchlaufen, und somit hat DFS die Komplexität $O(|V| + |E|)$.

Beweis. Vgl. J. Bey [Bey98] (Beweis zu Lemma 5.3.13, S. 298f).
Es ist zu zeigen, daß folgende Bedingungen erfüllt sind:

(i) $1 \leq nr(v) \leq N$,
(ii) $nr(v) \neq nr(u)$ für $v \neq u$ und $v, u \in V$,
(iii) $nr(v) < nr(u)$ für alle Kanten $(v, u) \in E$.

Zunächst zeigt man, daß jeder Knoten $v \in V$ höchstens einmal numeriert wird. Dies sieht man leicht, indem man sich für die beiden Fälle, daß entweder v keine Vorgänger oder mindestens einen Vorgänger hat, überlegt, daß v nicht zweimal numeriert werden kann.

Weiter ist zu zeigen, daß jeder Knoten $v \in V$ mindestens einmal numeriert wird. Dies geschieht durch einen Widerspruchsbeweis, indem man annimmt, es gäbe einen Knoten $v' \in V$, der nicht numeriert würde. Dies ist aber nicht möglich, da G laut Voraussetzung zyklusfrei sein sollte.

Insgesamt ist damit (i) und (ii) gezeigt, da für die N Knoten des Graphen jeweils genau eine der Nummern 1 bis N vergeben wird.

Es fehlt der Nachweis, daß G durch DFS in Richtung seiner Kanten numeriert wird. Dies ist aber klar, da der aktuell bearbeitete Knoten immer gerade die höchste bisher zu vergebende Nummer bekommt und damit die Nummern seiner Vorgänger sämtlich kleiner sind. Damit ist auch (iii) gezeigt.

Da pro Aufruf der Funktion $DFS - Visit$ genau eine Nummer vergeben wird, wird also $DFS - Visit$ genau N-mal aufgerufen, was einer Komplexität $O(|V|)$ entspricht. Faßt man sämtliche Schleifen über die Nachfolger der einzelnen Knoten in $DFS - Visit$ zusammen, so ergibt das zusätzlich noch den Aufwand $O(|E|)$, also insgesamt $O(|V| + |E|)$. □

Bemerkung 2.5 Da wir es in der Regel mit dünn besetzten Graphen zu tun haben werden, hat somit DFS ebenfalls Komplexität $O(|V|)$, d.h. bezogen auf unser Diskretisierungsproblem ist der Zeitaufwand von DFS linear in der Anzahl der Unbekannten.

2.3.2. Bestimmung der starken Zusammenhangskomponenten

Im folgenden soll nun ein Algorithmus zur Bestimmung und Numerierung der maximalen Zusammenhangskomponenten eines gegebenen Graphen G vorgestellt werden. J. Bey nutzt diesen Algorithmus, um die Steifigkeitsmatrix einer gegebenen Diskretisierung auf untere Blockdreiecksgestalt zu transformieren, wobei die Diagonalblöcke gerade den starken Zusammenhangskomponenten entsprechen, vgl. J. Bey [Bey98] (Kap.5.3.2, S. 296ff).
Der vorgestellte Algorithmus wurde 1972 von R. E. Tarjan [Tar72] entwickelt. Gegeben sei ein Graph $G = \{V, E\}$ und gesucht sind seine maximalen starken Zusammenhangskomponenten.

Algorithmus: Tarjan

(1) $M \leftarrow 0$
(2) $N \leftarrow 0$
(3) $stack \leftarrow NIL$
(4) **for** *each vertex* $v \in V$ **do**
(5) $\quad nr(v) \leftarrow 0$
(6) **od**
(7) **for** *each vertex* $v \in V$ **do**
(8) $\quad$ **if** $(nr(v) = 0)$ **then**
(9) $\quad\quad StrongComp(v)$
(10) $\quad$ **fi**
(11) **od**

Analog zum DFS Algorithmus werden in der Basisroutine des Tarjan–Algorithmus alle benötigten Variablen initialisiert. Zusätzlich wird ein Stack *stack* benötigt sowie ein globaler Zähler M für die Zusammenhangskomponenten. Für jeden noch nicht besuchten Knoten v (d.h. $nr(v) = 0$) wird nun die rekursive Suchfunktion $StrongComp(v)$ aufgerufen:

Algorithmus: $StrongComp(v)$

(1) $N \leftarrow N + 1$
(2) $nr(v) \leftarrow N$
(3) $low\_nr(v) \leftarrow nr(v)$
(4) *put* v *on the stack*
(5) **for** *each successor* w *from* v **do**
(6) $\quad$ **if** $(nr(w) = 0)$ **then**
(7) $\quad\quad StrongComp(w)$

```
 (8)      low_nr(v) ← min{low_nr(v), low_nr(w)}
 (9)    else if (nr(w) < nr(v)) and (w ∈ Stack) then
(10)      low(v) ← min{low_nr(v), low_nr(w)}
(11)    fi
(12)  od
(13)  if (low_nr(v) = nr(v)) then
(14)    M ← M + 1
(15)    C_M ← NIL
(16)    repeat
(17)      get w from stack
(18)      C_M ← C_M ∪ {w}
(19)    until (w = v)
(20)  fi
```

Diese Unterroutine des Tarjan–Algorithmus hat folgende Aufgabe:

Analog zu $DFS-Visit$ wird dem übergebenen Knoten v die nächste freie Knotennummer N zugewiesen {(1),(2)}. Zusätzlich wird in dem Array low_nr der Eintrag für den Knoten v mit $nr(v)$ initialisiert {(3)}. Die Bedeutung dieses Arrays soll im weiteren verdeutlicht werden. Abschließend wird der Knoten v auf dem Stack *stack* abgelegt {(4)}.

In der Schleife {(5), ... ,(12)} wird für jeden der noch nicht numerierten Nachfolger des Knotens v gegebenenfalls wiederum *StrongComp* aufgerufen {(7)}, d.h. der Algorithmus arbeitet analog zur Tiefensuche alle von v aus erreichbaren Knoten ab. Ist dabei der Wert low_nr des aktuellen Knotens v größer als der Wert low_nr seines Nachfolgers w, so wird $low\_nr(v)$ entsprechend geändert {(8)}. Diese Anpassung erfolgt auch, falls ein Nachfolger von v schon besucht wurde und noch auf dem Stack liegt.

Während des Suchprozesses enthält $low\_nr(v)$ den kleinsten low_nr-Wert aller zur maximalen Zusammenhangskomponente von v gehörigen Knoten, die bereits besucht wurden. Ist also v der zuerst besuchte Knoten der maximalen Zusammenhangskomponente, sozusagen der Einstiegsknoten während des Suchprozesses, so erfolgt der Ausstieg wiederum über v, da v sonst nicht der erste besuchte Knoten der Komponente gewesen wäre. Damit ist das Kriterium $(low\_nr(v) = nr(v))$ {(13)} erfüllt. Alle Knoten derselben Komponente liegen noch oberhalb von v im Stack, da nur sie nach v besucht wurden und die Knoten von nachfolgenden Zusammenhangskomponenten bereits vom Stack genommen wurden. Deshalb liefern die Schritte {(13), ... ,(20)} alle Knoten der maximalen Zusammenhangskomponente. Insgesamt gilt also der Satz 2.3.

Satz 2.3 (Korrektheit des Tarjan–Algorithmus)

Für beliebige gerichtete Graphen $G = \{V, E\}$ stimmen die vom Tarjan–Algorithmus gefunden Knotenmengen $C_1, \ldots, C_M$ gerade mit den maximalen Zusammenhangskomponenten von G überein. Der Algorithmus hat Komplexität $0(|V|)$ für dünn besetzte Graphen.

Beweis: Vgl. D. Jungnickel [Jun94] (Beweis zu Satz 11.5.6, S. 428f). □

Bemerkung 2.6 Die Berechnung des Zeitaufwands ist die gleiche wie für DFS, da die Grundstruktur des Algorithmus analog ist. Der Aufwand zur Bearbeitung des Stacks ist

$0(|V|)$, da jeder Knoten genau einmal abgelegt und somit auch genau einmal heruntergenommen wird.

Da bei diesem Algorithmus analog zur Tiefensuche zunächst so tief wie möglich in den Graphen hineingesucht wird, um sich dann vom tiefsten Punkt wieder rekursiv nach oben zu arbeiten, wird die erste gefundene maximale Zusammenhangskomponente am weitesten vom Startknoten der Suche entfernt liegen, die nächste gefundene Komponente am zweitweitesten usw. Dabei ist diese Entfernung in Richtung der Kanten gemessen, d.h. die Zusammenhangskomponenten werden in umgekehrter Reihenfolge durchnumeriert. Zwischen zwei maximalen Zusammenhangskomponenten C_k und C_l kann eine Verbindung nur in einer Richtung bestehen, da sonst $C_k \cup C_l$ eine starke Zusammenhangskomponente wäre und somit C_k und C_l nicht maximal waren. Liegt C_k weiter vom Startknoten entfernt als C_l, so kann es nur eine Verbindung von C_l nach C_k geben, da C_k von C_l aus besucht wurde.

Dreht man die vom Tarjan–Algorithmus erzeugte Numerierung um, so erhält man eine Numerierung der maximalen starken Zusammenhangskomponenten von G in Richtung der Kanten von G. Der dazu notwendige Algorithmus sieht wie folgt aus:

Algorithmus: Tarjan2

```
(1)  M ← 0
(2)  N ← 0
(3)  stack ← NIL
(4)  for each vertex v ∈ V do
(5)     nr(v) ← 0
(6)  od
(7)  for each vertex v ∈ V do
(8)     if (nr(v) = 0) then
(9)        StrongComp(v)
(10)    fi
(11) od
(12) N ← 0
(13) for i ← 1,... ,M do
(14)    C'_k ← C_{M+1−k}
(15)    for each vertex v from C'_k do
(16)       N ← N + 1
(17)       nr(v) ← N
(18)    od
(19) od
```

Die Zeilen (12), ... ,(19) liefern eine entsprechende Numerierung der Zusammenhangskomponenten und zusätzlich eine Numerierung der einzelnen Knoten. Allerdings gibt die Knotennumerierung nur die Reihenfolge wieder, mit der Knoten gemäß der Tiefensuche im Tarjan–Algorithmus abgearbeitet wurden.

3. Numerierungsstrategien (Downwind Numbering)

3.1. Allgemeine Bemerkungen

Es sollen nun verschiedene Strategien vorgestellt werden, um die Unbekannten eines elliptischen Randwertproblems in Flußrichtung zu numerieren. Um dabei Algorithmen aus der Graphentheorie anwenden zu können, ist es zunächst erforderlich, mittels eines geeigneten Kriteriums einen sogenannten *Konvektionsgraphen* zu erzeugen, der den Fluß des Mediums in geeigneter Art und Weise auf dem diskreten Level repräsentiert. Siehe hierzu auch W. Hackbusch [Hac95] oder S. Le Borne [GP97]. Die Eckpunkte dieses Graphen korrespondieren dabei mit den Eckpunkten der Triangulierung $\mathcal{T}_h$. Hierbei wird zwischen azyklischem und zyklischem Konvektionsgraphen unterschieden. Ziel der Numerierungen ist es, die aus der Diskretisierung des entsprechenden Randwertproblems hervorgegangene Steifigkeitsmatrix approximativ in eine untere Dreiecksform umzuwandeln und so eine Vorkonditionierung des Problems zu erreichen. Dadurch lassen sich mit einfachen Lösern, wie z.B. dem Gauß–Seidel–Verfahren, sehr gute numerische Resultate erzielen.

Für die Numerierung des Konvektionsgraphen im einfacheren Fall ohne Zyklen benötigt man dafür lediglich Standardalgorithmen aus der Graphentheorie, wie z.B. die Tiefensuche. Diese ist mit einem Aufwand $0(n)$ sehr effizient.

Im Falle auftretender Zyklen hat man zwei Verfahren zur Verfügung. Zum einen lassen sich mittels des Tarjan-Algorithmus die Zyklen zu Blöcken zusammenfassen, um so zu der gewünschten unteren Blockdreiecksgestalt zu gelangen, wobei dann das Block–Gauß–Seidel–Verfahren als Löser angewendet wird. Dieses Verfahren wurde 1998 von J. Bey [Bey98] vorgestellt.

Eine andere Möglichkeit besteht darin, die auftretenden Zyklen an minimal vielen Stellen aufzutrennen, was der Suche nach einem *minimalen Feedback–Vertex–Set* entspricht. Der zyklusfreie Anteil des Konvektionsgraphen entspricht dann einer unteren Block-Dreiecksmatrix in der Steifigkeitsmatrix. Mit Hilfe einer Schur-Komplement-Methode erhält man auch hier einen effizienten Löser. Die Suche nach einem minimalen Feedback–Vertex–Set stellt dabei das eigentliche Problem dar. Da es sich hierbei um ein NP-vollständiges Problem handelt, läßt es sich lediglich approximativ lösen, vgl. M. R. Garey & D. S. Johnson [GJ79] (Problem GT 7, S. 191). Von W. Hackbusch wurde 1995 ein solcher Approximationsalgorithmus vorgestellt [Hac95]. Sowohl das Verfahren von J. Bey als auch jenes von W. Hackbusch benötigen lediglich linearen Aufwand $0(n)$ bzgl. der Problemgröße.

Für das Numerieren in Flußrichtung wurde 1994 von P. Bastian & G. Wittum [BW94] der Begriff *„Downwind Numbering“* eingeführt.

3.2. Konvektionsgraph

In diesem Kapitel soll der Konvektionsgraph beschrieben werden. Er stellt die Verbindung zwischen der Theorie elliptischer Randwertprobleme und der Graphentheorie her. Erst dadurch wird der Einsatz graphentheoretischer Mittel zur Vorkonditionierung von Lösern elliptischer Randwertaufgaben möglich.

Zunächst muß sichergestellt werden, daß mittels eines geeigneten Auswahlkriteriums aus den Daten des RWP ein Graph generiert wird, der die zu betrachtende Problematik adäquat wiederspiegelt.

Mit zunehmender Konvektionsdominanz verschlechter sich die Konvergenz iterativer Verfahren zur Lösung des RWPs (1.3) bzw. (1.4). Man sucht deshalb nach einer geeigneten Vorkonditionierung des diskretisierten Randwertproblems, um auch im Falle dominanter Konvektion gute numerische Resultate zu erreichen. Wie in Abschnitt 1.5 (S. 12) bzw. 1.6.1 (S. 14) dargelegt, ist es für diesen Fall sinnvoll, eine Numerierung der Unbekannten in Konvektionsrichtung zu ermitteln, da die diskretisierte Steifigkeitsmatrix mit einer solchen Numerierung nahezu untere Dreiecksgestalt annimmt. Für eine derart vorkonditionierte Matrix liefern einfache Löser wir Gauß–Seidel bzw. SOR gute numerische Resultate.

Der zu generierende Konvektionsgraph soll also die Konvektion im Gebiet Ω in geeigneter Art und Weise wiederspiegeln, um dann mit Hilfe von Numerierungsalgorithmen der Graphentheorie eine Numerierung der Unbekannten bzw. Gitterpunkte der Triangulierung $\mathcal{T}_h$ von Ω in Flußrichtung zu ermitteln. In jedem Fall wird es sich dabei um einen gerichteten Graphen handeln.

Für die verwendeten Numerierungsalgorithmen ist es absolut unerheblich, auf welche Weise der Konvektionsgraph generiert wurde. Für die Effizienz der durch die ermittelten Umnumerierung erreichten Vorkonditionierung gilt dies nicht. Mit anderen Worten, das verwendete Kriterium zur Bestimmung des Konvektionsgraphen spielt eine entscheidene Rolle für die Effizienz des Verfahrens.

Im folgenden sollen nun verschiedene Auswahlkriterien vorgestellt werden, wie sie in den Arbeiten von J. Bey [Bey98] und S. Le Borne [GP97] verwendet wurden. Beide arbeiten bei der Diskretisierung der Randwertprobleme mit der in Abschnitt 1.6.1 vorgestellten Upwind–Technik zur Stabilisierung der Lösung. Alle Aussagen zu den Kriterien beziehen sich deshalb auf diese Form der Stabilisierung.

Sei A eine $(n \times n)$-Matrix mit skalaren Einträgen, die aus der Diskretisierung eines Konvektions–Diffusions–Reaktions–Problems mittels Galerkin–FEM und Upwind–Stabilisierung hervorgeht. Identifiziert man die n Spalten (bzw. Zeilen) der Matrix mit den n Knoten eines Graphen G (also $|V| = n$), dann erhält man mittels

$$E_0 = \{(u, v) \in V \times V \,:\, a_{uv} \neq 0\}$$

auf kanonische Weise einen zu A gehörenden Graphen $G = \{V, E_0\}$.

Für die weiteren numerischen Verfahren sind aber nur diejenigen Matrixeinträge in A interessant, die mit den dominanten Konvektionsanteilen A^C der Diskretisierung korrespondieren. Der Diffusionsanteil entspricht dem symmetrischen Anteil A^S von A, während ein eventuell vorhandener Reaktionsanteil nur die Diagonale beeinflusst. Deshalb

läßt sich folgendes Kriterium zur Auswahl der Kantenmenge von G vorstellen [GP97]:

$$E_1 = \{(u,v) \in V \times V \,:\, |a_{uv}^C| > \kappa |a_{uv}^S|\}.$$

Dabei entspricht a_{uv}^C dem Konvektionsanteil von a_{uv} und a_{uv}^S dem symmetrischen Anteil (Diffusion und Reaktion). Der Parameter $\kappa \in [0,1]$ kann unter Umständen von der Gittergröße abhängen.

Eine weitere Möglichkeit ist es, für jede Spalte nur den betragsmäßig größten Wert zur Erzeugung einer Kante zu betrachten [GP97]:

$$E_2 = \{(u,v) \in V \times V \,:\, |a_{uv}^C| > \lambda \max_{u \in V} \{|a_{uw}^C| : |a_{uw}^C| > \kappa |a_{uw}^S|\}\}.$$

In den Fällen, in denen eine Aufteilung in den symmetrischen und den Konvektionsanteil nicht möglich ist, wäre auch eines der folgenden Kriterien denkbar [GP97]:

$$E_3 = \{(u,v) \in V \times V \,:\, |a_{uv}| > \kappa |a_{uu}|\},$$
$$E_4 = \{(u,v) \in V \times V \,:\, |a_{uv}| > \kappa |a_{vu}|\}.$$

Beim verwendeten Upwind-Verfahren gilt für den Konvektionsanteil $a_{uv}^C < 0$ genau dann, wenn die Eckpunkte u, v bzgl. der Triangulierung $\mathcal{T}_h$ benachbart sind und u von v aus gesehen stromabwärts liegt. Damit kann man auch folgendes Kriterium anwenden [Bey98]:

$$E_5 = \{(u,v) \in V \times V \,:\, a_{uv}^C < 0\}.$$

Die Effizienz der im weiteren vorgestellten Algorithmen wird entscheidend von der Größe der Kantenmenge E abhängen und damit auch von κ und λ (in den Fällen E_2, E_3 und E_4). Die Matrixeinträge $(a_{uv})_{u \in \{1,\ldots,|V|\}, v \in \{1,\ldots,|V|\}}$ hängen von der Gittergröße h, der zugrundeliegenden Triangulierung und der Diskretisierung des Problems ab.

Sämtliche vorgestellte Kriterien beziehen sich auf die Einträge in der Steifigkeitsmatrix, d.h. für die Generierung des Konvektionsgraphen muß die Diskretisierung des RWPs schon abgeschlossen sein. Die Steifigkeitsmatrix muß dann entsprechend der ermittelten neuen Numerierung permutiert werden.

Denkbar wäre es aber auch, den Konvektionsgraphen schon zu einem früheren Zeitpunkt zu erstellen, und zwar vor der Durchführung der Diskretisierung. Zum Beispiel läßt sich mit Hilfe des Gitters und des Strömungsfeldes $\vec{b}$ ebenfalls ein Konvektionsgraph erstellen. Der Vorteil wäre hierbei, daß die Permutation der Steifigkeitsmatrix entfällt, da die Diskretisierung auf dem umnummerierten Gitter erfolgt. Der Nachteil ist, daß für jeden Knoten der Konvektionsanteil doppelt berechnet wird, zum einen bei der Erzeugung des Graphen und zum anderen bei der Diskretisierung.

Im folgenden sollen Kriterien, die sich auf die Einträge in der Steifigkeitsmatrix beziehen, als *algebraische Kriterien* und jene, die sich auf das Gitter beziehen, als *geometrische Kriterien* bezeichnet werden.

Betrachtet man also den Patch aus Abbildung 3.1. Dann gilt

$$\alpha = \langle b(1), (1,2) \rangle$$

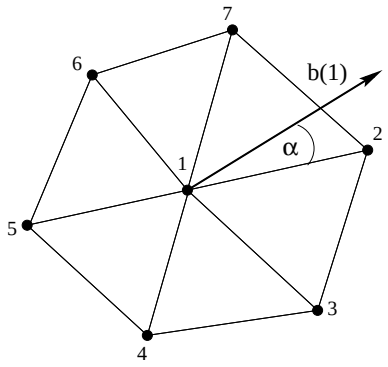


Abbildung 3.1.: Patch

mit dem euklidischen Skalarprodukt $\langle \cdot, \cdot \rangle$ und man erhält folgende Kantenmenge im Graphen:

$$E_6 := \{(u,v) \in V \times V \, : \langle b(u), (u,v) \rangle = \min\{\langle b(u), (u,w) \rangle : (u,w) \in E\}\}.$$

Im Fall E_0 entspricht der entstehende Graph gerade den Gitterknoten und -kanten der Triangulierung, während die Kantenmengen E_1 bis E_6 einen Graph erzeugen, der einer Untermenge dieses Gitters entspricht. In der vorliegenden Arbeit wurde mit der Kantenmenge E_6 gerechnet.

Bemerkung 3.1 (Zum verwendeten Kriterium E_6) Im Rahmen dieser Arbeit wurde an einem größeren Softwarepaket mitgearbeitet, mit dem es während dieser Arbeit noch nicht möglich war, eines der algebraischen Kriterien E_0 bis E_5 anzuwenden. Dazu aber mehr in Kapitel 4 (S. 57).

3.3. SOR–Verfahren

Der in dieser Arbeit verwendete Löser ist das *SOR–Verfahren* (**S**uccessive **O**ver**R**elaxation). Es basiert auf dem Gauß–Seidel–Verfahren, wobei ein zusätzlicher Faktor $\omega \in \mathbb{R}$ eingeführt wird, mit dem in jedem Iterationsschritt die neu berechneten Komponenten überrelaxiert, d.h. extrapoliert werden können.
Sei A eine nichtsinguläre $n \times n$-Matrix, wie sie z.B. aus der Diskretisierung eines RWPs mittels Galerkin–FEM hervorgeht, vgl. Gl. (1.12) (S. 11) und Lemma 1.5 (S. 12). Gegeben sei das lineare Gleichungssystem

$$Ax = b \tag{3.1}$$

mit $f \in \mathbb{R}^n$, vgl. Satz 1.2 (S. 11).
Seien für die Koeffizientenmatrix $A = (a_{i,j})_{1 \le i,j \le n}$ durch

$$L := (a_{ij})_{1 \le j < i \le n} \ , \ D := (a_{ii})_{1 \le i \le n} \quad \text{und} \quad U := (a_{ij})_{1 \le i < j \le n}$$

die *strikt untere Blockdreiecksmatrix*, die *Blockdiagonalmatrix* und die *strikt obere Blockdreiecksmatrix* von A definiert, so daß also $A = L + D + U$.

Damit läßt sich das Gauß–Seidel– oder auch Einzelschrittverfahren folgendermaßen angeben:

$$\begin{aligned} & x^{(k+1)} := x^{(k)} + (D+L)^{-1}\left(b - Ax^{(k)}\right) \\ \Longleftrightarrow \quad & x^{(k+1)} := -(D+L)^{-1} U x^{(k)} + (D+L)^{-1} b, \end{aligned} \tag{3.2}$$

mit dem Iterationsindex $k = 1, 2, \ldots$. Da A nichtsingulär ist und somit $a_{ii} \neq 0, 1 \leq i \leq n$, angenommen werden kann, ist die Matrix $(D+L)$ ebenfalls nichtsingulär, d.h. $(D+L)^{-1}$ existiert.

Beim SOR-Verfahren wird nun ein Faktor ω eingeführt, über den sich durch geeignete Wahl die Konvergenz des Verfahrens entscheidend beeinflussen läßt. Die abgeänderte Iterationsvorschrift lautet

$$\begin{aligned} & x^{(k+1)} := x^{(k)} + \omega\,(D+\omega L)^{-1}\left(b - Ax^{(k)}\right) \\ \Longleftrightarrow \quad & x^{(k+1)} := \left\{(D+\omega L)^{-1}\left((1-\omega)\,D - \omega U\right)\right\} x^{(k)} + \omega\,(D+\omega L)^{-1} b. \end{aligned} \tag{3.3}$$

Den Faktor ω nennt man auch *Relaxationsparameter*. Nach dem Satz von W. Kahan [Kah58] ist $0 < \omega < 2$ notwendig für die Konvergenz des Relaxationsverfahrens (3.3). Für $\omega = 1$ erhält man wieder das Gauß–Seidel–Verfahren.

Die komponentenweise Angabe des Verfahrens lautet

$$x_i^{(k+1)} := \omega \frac{b_i}{a_{ii}} - \omega \sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_j^{(k+1)} - \omega \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} x_j^{(k)} + (1-\omega)\, x_i^{(k)} \tag{3.4}$$

und führt zu folgendem Algorithmus

Algorithmus: SOR

(1) **for** $k = 1, 2, \ldots$ **do**
(2) **for** $i = 1, \ldots, n$ **do**
(3) $\tilde{x}_i^k := a_{ii}^{-1}\left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^k - \sum_{j=i+1}^{n} a_{ij} x_j^{k-1}\right)$
(4) $x_i^k := \omega \tilde{x}_i^k + (1-\omega) x_i^{k-1}$
(5) **od**
(6) **od**

SOR führt in jeder Iteration einen Informationstransport durch, da die bereits berechneten Komponenten der nächsten Näherungslösung zur Berechnung der folgenden Komponente verwendet werden. Die Menge der transportierten Information und daher auch die Konvergenzrate hängt sehr von der Struktur der Matrix ab und wird umso besser, je mehr Einträge unterhalb der Diagonalen liegen. Für den Fall unterer Dreiecksmatrizen ist das Gauß–Seidel– bzw. SOR–Verfahren dann sogar ein exakter Löser, da es hier dem Vorwärtseinsetzen entspricht.

Um diesen Zusammenhang auszunutzen, kann man die Bearbeitungsreihenfolge der Zeilen ändern. Man ersetzt dafür i in dem SOR–Algorithmus durch $\pi(i)$, wobei $\pi : \mathbb{N} \mapsto \mathbb{N}$ eine

Permutation beschreibt. Diese ist also optimal, wenn

$$\forall\, 1 \leq i,j \leq n: \quad a_{i,j} \neq 0 \;\Rightarrow\; \pi(i) \geq \pi(j).$$

Sind die Einträge in der Matrix wiederum Matrizen, so spricht man von einer sogenannten *Blockmatrix* und die vorgestellten Löser nennt man *Block–SOR–* bzw. *Block–Gauß–Seidel–Verfahren.*

Definition 3.1 (Blockmatrix) Seien $l, m \in \mathbb{N}$, so daß $lm = n$. $A \in \left(\mathbb{R}^{l\times l}\right)^{m\times m}$ (d.h. $A = (A_{i,j})_{1\leq i,j\leq m}$ mit $A_{ij} \in \mathbb{R}^{l\times l}$) heißt *Blockmatrix*, $b \in \left(\mathbb{R}^{l}\right)^{m}$ (d.h. $b = (b_i)_{1\leq i\leq m}$ mit $b_i \in \mathbb{R}^{l}$) *Blockvektor*. Entsprechend heißt A_{ij} *Matrixblock*, b_i *Vektorblock*.

Bei der Anwendung des SOR–Algorithmus bzw. des Gauß–Seidel–Verfahrens auf sogenannte *echte Blockmatrizen* ($l > 1$) erhebt sich die Frage nach der Berechnung des Produktes mit A_{ii}^{-1}. Für die Effizienz des Block–Lösers ist es wünschenswert, wenn die Diagonalblöcke kleine Dimension haben (siehe Abschnitt 3.5.1).

3.4. Azyklische Fall

Treten im Strömungsfeld $\vec{b}$ keine Verwirbelungen auf oder werden auftretende Verwirbelungen nicht durch die Triangulierung aufgelöst, so genügt eine einfache Tiefensuche (siehe Kap. 2.3.1, S. 25) zur Durchnumerierung des Konvektionsgraphen.

Dabei sichert der Satz 2.2 (S. 26), daß der Konvektionsgraph durch die Tiefensuche in Richtung seiner Kanten, d.h. in Flußrichtung, numeriert wird. Der hierfür benötigte Aufwand ist linear (siehe ebenfalls Satz 2.2).

Die Bemerkungen aus Kapitel 1.5 (S. 12) zu Eigenschaften der Steifigkeitsmatrix sichern dabei, daß die konvektionsdominanten Einträge der Steifigkeitsmatrix durch eine flußorientierte Umnumerierung der Unbekannten unterhalb der Diagonalen erscheinen. Dies wiederum ermöglicht den Einsatz des Gauß–Seidel–Verfahrens zur Lösung des Problems. Für den reinen Konvektionsfall wird dieses Verfahren sogar zum exakten Löser.

Damit ergibt sich folgender Algorithmus zur Lösung elliptischer Randwertprobleme im azyklischen Fall

Algorithmus: Downwind Numbering mittels Tiefensuche

(1) *Erzeuge den Konvektionsgraphen mit einer der Kantenmengen* $E_0, \ldots, E_6$.
(2) *Finde eine flußorientierte Numerierung mittels Tiefensuche.*
(3) *Löse das umnumerierte Problem mittels Gauß–Seidel–Verfahren.*

Bemerkung 3.2 (Breitensuche) Es ist auch möglich, anstelle der hier verwendeten Tiefensuche einen alternativen Numerierungsalgorithmus einzusetzten, nämlich die Breitensuche (siehe Kap. 2.3.1, S. 25). In den Arbeiten von P. Bastian & G. Wittum, [BW94] und J. Bey & G. Wittum, [BW95] werden solche Verfahren vorgestellt. Allerdings wird keine optimale Komplexität erreicht, denn bei beiden Verfahren wird der

Konvektionsgraph im Stile der Breitensuche wellenfrontartig durchsucht. Das macht eine zusätzliche Sortierung etwa mit Quicksort notwendig.

Der Vorteil der verwendeten Tiefensuche liegt zum einen in der optimalen Komplexität des Verfahrens, zum anderen in der Dimensionsunabhängigkeit, d.h. dieser Algorithmus kann sowohl für zwei- als auch für dreidimensionale Probleme eingesetzt werden. Die notwendige Voraussetzung der Zyklusfreiheit des Konvektionsgraphen ist aber unrealistisch, da auftretende Wirbel im Geschwindigkeitsfeld $\vec{b}$ und daraus resultierende zyklische Konvektionsgraphen in der Praxis die Regel sind. Deshalb sollen im folgenden Kapitel 3.5 zwei praxisorientierte Verfahren vorgestellt werden.

3.5. Zyklische Fall

3.5.1. Numerierung maximaler Zusammenhangskomponenten - Das Verfahren von J. Bey

Verallgemeinert man das Prinzip des Downwind Numbering im azyklischen Fall in kanonischer Weise auf den zyklischen Fall, so gelangt man zu dem von J. Bey [Bey98] eingeführten Verfahren. Die grundlegende Idee besteht darin, anstatt der Gitterpunkte die maximalen Zusammenhangskomponenten des Konvektionsgraphen in Richtung der Strömung zu numerieren, so daß der Konvektionsanteil der Steifigkeitsmatrix untere Dreiecksgestalt annimmt. Eine so vorkonditionierte Matrix läßt sich dann mittels des Block-Gauß–Seidel–Verfahrens effizient lösen. Für den reinen Konvektionsfall ist das Verfahren sogar ein exakter Löser.

Der von J. Bey vorgestellte Algorithmus ergibt sich nun wie folgt

Algorithmus: Downwind Numbering - das Verfahren von J. Bey

(1) *Erzeuge den Konvektionsgraphen mit einer der Kantenmengen $E_0, \ldots, E_6$.*
(2) *Finde eine flußorientierte Numerierung der starken Zusammenhangskomponenten des Konvektionsgraphen mittels Tarjan–Algorithmus.*
(3) *Löse das umnumerierte Problem mittels Block-Gauß–Seidel–Verfahren.*

Der hierbei verwendete Tarjan-Algorithmus ist von optimaler Komplexität und auch dimensionsunabhängig (Satz 2.3, S. 28). Damit läßt sich dieses Verfahren als effizienter Vorkonditionierer für den zwei - und dreidimensionalen Fall einsetzen. Die praktische Verwendbarkeit dieses Verfahren ist aus Komplexitätsgründen allerdings zunächst auf Probleme mit relativ kleinen Zyklen beschränkt, da in jedem Iterationsschritt für jeden der vorhandenen Diagonalblöcke ein Gleichungssystem entsprechender Dimension zu lösen ist.

Um auch Probleme mit größeren Zusammenhangskomponenten behandeln zu können, benötigt man ein effizientes Verfahren zur Lösung dieser Blocksysteme. Ein solches Verfahren soll im Kapitel 3.5.3 (S. 39) vorgestellt werden. Die entscheidene Idee ist, die maximalen Zusammenhangskomponenten an minimal vielen Stellen „aufzuschneiden“, und zwar solange, bis der restliche Teilgraph zyklusfrei ist. Die „Schnittknoten“, an denen die maximalen Zusammenhangskomponenten aufgetrennt werden, bezeichnet man auch als

Feedback–Vertices. Siehe dazu das folgende Kapitel 3.5.2.

3.5.2. Feedback–Vertex–Set–Problem

Definition 3.2 (Feedback–Vertex–Set (FVS)) Sei $G = \{V, E\}$ ein endlicher gerichteter Graph. Man nennt eine Teilmenge $F \subset V$ ein *Feedback–Vertex–Set* (FVS), wenn der von $V' := V \setminus F$ aufgespannte Teilgraph $G_{V'} := \{V', E_{V'}\}$ mit der Kantenmenge

$$E_{V'} = \left\{(u, v) \in E \mid u, v \in V'\right\}$$

zyklusfrei ist. Die Knoten $v \in F$ heißen *Feedback–Vertices* von G. Man nennt ein Feedback–Vertex–Set F *minimal*, wenn für jeden Knoten $v \in F$ die Menge $F \setminus \{v\}$ kein Feedback–Vertex–Set von G mehr ist. F heißt *optimal*, wenn für jedes andere Feedback–Vertex–Set F' von G die Ungleichung $|F'| \geq |F|$ gilt.

Das Feedback–Vertex–Set–Problem ist die Suche nach einem minimalem Feedback–Vertex–Set in einem gerichteten Graphen $G = \{V, E\}$. Formuliert man dieses Optimierungsproblem in ein Entscheidungsproblem um, so gelangt man zu der Definition 3.3.

Definition 3.3 (Feedback–Vertex–Set–Problem) Gegeben sei ein gerichteter Graph $G = \{V, E\}$ und eine natürliche Zahl k. Die Frage ist nun, ob es in G ein Feedback–Vertex–Set der Größe k gibt oder formal

$$\mathcal{FVS} = \{\langle G, k\rangle : \text{der Graph } G \text{ besitzt ein } FVS \text{ der Größe } k\}.$$

Zur Komplexität des Feedback–Vertex–Set–Problems existiert die nachfolgende Ausage. Vgl. M. R. Garey & D. S. Johnson [GJ79] (Problem GT 7, S. 191).

Satz 3.1

Das Vertex–Cover–Problem ist polynomial auf das Feedback–Vertex–Set–Problem für gerichtete Graphen reduzierbar. Das bedeutet, das Feedback–Vertex–Set Verfahren ist NP-vollständig.

Die Beweisidee ist, das Feedback–Vertex–Set–Problem in polynomialzeit auf das sogenannte *Vertex–Cover–Problem* zurückzuführen. Dazu aber zunächst die nachfolgenden Definitionen.

Definition 3.4 (Vertex–Cover) Sei $G = \{V, E\}$ ein ungerichteter Graph. Ein *Vertex–Cover* ist eine Teilmenge $V' \subset V$, so daß für alle Kanten $e = (v, w) \in E$ gilt: $v \in V'$ oder $w \in V'$ oder beide liegen in V'. Jeder Knoten des Graphen überdeckt die mit ihm inzidenten Kanten und ein *Vertex Cover* von G ist eine Menge von Knoten, die sämtliche Kanten in G überdecken. Die Größe eines *Vertex–Covers* die Anzahl der Knoten darin.

Definition 3.5 (Vertex–Cover–Problem) Gegeben sei ein Graph $G = \{V, E\}$ und eine natürliche Zahl $k \in \mathbb{N}$. Beim *Vertex–Cover–Problem* ist folgendes Entscheidungsproblem: Existiert in G ein Vertex–Cover der Größe k?

Das Vertex–Cover Problem ist bekannterweise NP–vollständig, vgl. [CLR90] (Thm. 36.12, S. 950). Damit kann man zum Beweis des Satzes 3.1 übergehen.

Beweis:

$\underline{\mathcal{FVS} \in NP}$

Sei $G = \{V, E\}$ und $k \in \mathbb{N}$. Als Zertifikat wählt man das Feedback–Vertex–Set $V' \in V$. Der Verifikationalgorithmus bestätigt, daß $|V'| = k$ ist. Entfernt man nun sämtliche Knoten aus V' inklusive der mit ihnen inzidenten Kanten aus dem Graphen G so müßte der verbleibende Graph $G' = \{V \setminus V', E'\}$ mit $E' = \{(v, w) \in E : v \notin V' \wedge w \notin V'\}$ azyklisch sein. Dies überprüft man leicht in polynomialzeit z.B. mittels TARJAN–Algorithmus, siehe Abschnitt 2.3.2 (S. 27). Die Reduzierung von G auf G' ist ebenfalls in polynomieller Zeit möglich.

$\underline{\mathcal{VC} \leq_P \mathcal{FVS}}$

Sei $G = \{V, E\}$ ein ungerichteter Graph und sei G' der gerichtete Graph, der aus G durch Kantenverdopplung hervorgeht, d.h. jede ungerichtete Kante wird durch zwei gerichtete ersetzt. Damit hat G' folgende Gestalt $G' = \{V, E'\}$ mit $E' = \{(v, w), (w, v) | (v, w) \in E\}$. G' entsteht also aus G, indem jede Kante aus E durch einen Kreis (w, v, w) ersetzt wird. Eine Menge $W \subset V$ ist nun ein Feedback–Vertex–Set für G' (jeder Kreis in G' enthält einen Knoten aus W) genau dann, wenn W ein Vertex Cover für G ist. Denn wäre W kein Vertex Cover für G, so gibt es eine Kante $(x, y) \in E$, so daß gilt $x, y \notin W$, aber (x, y, x) ist ein Zyklus in G', d.h. $x \in W$ oder $y \in W$. Dies ist ein Widerspruch.

Die Darstellung von G' kann in polynomialer Zeit aus G gewonnen werden. □

3.5.3. Approximationsalgorithmus von W. HACKBUSCH

Im folgenden wird ein Approximationsalgorithmus zur Berechnung eines quasioptimalen minimalen Feedback–Vertex–Set für planare Graphen mit Aufwand $O(|V|)$ betrachtet. Neben Zweidimensionalität wird eine weitere Eigenschaft des Graphen gefordert, die sich infolge der Herkunft des Graphen aus der Diskretisierung eines Konvektions–Diffusions–Reaktions–Problems ergibt, und zwar die sogenannte Einweg-Fluß-Bedingung oder auch one-flow-condition.

Definition 3.6 (one-flow-condition) Sei $G = \{V, E\}$ ein Digraph. Die *one-flow-condition* für einen Knoten $v \in V$ ist erfüllt genau dann, wenn sich die Liste $\{e_1, \ldots, e_k\}$ der mit v inzidenten Kanten so sortieren läßt, daß für ein $l \in \{0, \ldots, k\}$ die e_i $(1 \leq i \leq l)$ ausgehende Kanten und die e_i $(l < i \leq k)$ eingehende Kanten sind. Man sagt, $G = \{V, E\}$ erfüllt die *one-flow-condition*, wenn sie für jeden Knoten $v \in V$ erfüllt ist.

Bemerkung 3.3 Geometrisch gesehen bedeutet diese Bedingung, daß um den Knoten $v \in V$ zwei Sektoren existieren und die Konvektion durch den ersten Sektor in v hineinströmt und durch den zweiten Sektor aus v herausströmt. Da $l = 0$ und $l = k$ erlaubt ist, kann einer der beiden Sektoren leer sein, was dann gleichbedeutend ist mit einer Quelle bzw. einer Senke. Die Abbildung 3.3 zeigt eine verbotende Situation.

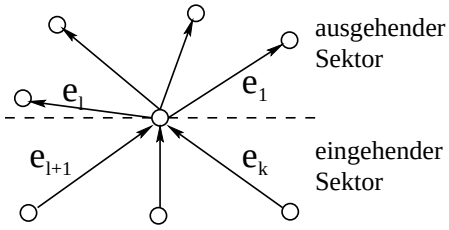


Abbildung 3.2.: one-flow-condition

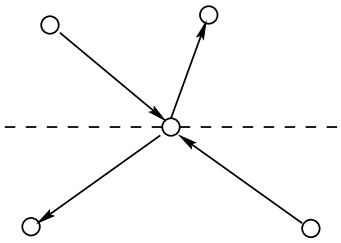


Abbildung 3.3.: Verbotene Situation

Filtern des Ausgangsgraphen

In einem ersten Schritt wird der Ausgangsgraph durchmustert und zwar in der Gestalt, daß alle Zyklen des Graphen herausgefiltert werden. Der erforderliche Aufwand ist linear. Der folgende Algorithmus liefert einen entsprechenden gefilterten Graphen, dabei werden den Knoten $v \in V$ Attribute F, L oder C zugeordnet. Hierbei bedeutet das $F(L)$-Attribut, daß der entsprechende Knoten als erstes (first) bzw. letztes (last) zu numerieren ist. Das Attribut C bekommen Knoten, die Teil eines Zyklus sind. Die Vergabe der Attribute erfolgt nach der folgenden (rekursiven) Vorschrift:

$v \in V$ erhält das Attribut F bzw. L genau dann, wenn alle Vorgänger (Nachfolger) das Attribut F bzw L bereits besitzen. Zu Beginn werden alle Knoten des Graphen mit dem C-Attribut initialisiert.

Algorithmus: Filter(G)

(1) **for** *each vertex* $v \in V$ **do**
(2) $attr[v] \leftarrow C$
(3) **od**
(4) **for** *each vertex* $v \in V$ **do**
(5) $SetAttr[v]$
(6) **od**

Algorithmus: SetAttr(v)

(1) **if** $attr[v] = C$ **then**
(2) $SetF[v]$
(3) **fi**

```
(4) if attr[v] = C then
(5)    SetL[v]
(6) fi
```

Algorithmus: SetF(v)

```
(1) if for all pred(v) attr[pred(v)] = F then
(2)    attr(v) ← F
(3)    for w ∈ {succ(v)} do
(4)       if attr[w] = C then
(5)          SetF(w)
(6)       fi
(7)    od
(8) fi
```

Algorithmus: SetL(v)

```
(1) if for all succ(v) attr[succ(v)] = L then
(2)    attr(v) ← L
(3)    for w ∈ {pred(v)} do
(4)       if attr[w] = C then
(5)          SetL(w)
(6)       fi
(7)    od
(8) fi
```

Bemerkung 3.4 Der Aufwand dieses Algorithmus ist $O(V + E)$, was für dünn besetzte Graphen einem Aufwand von $O(V)$ entspricht.

Beweis: Aus der globalen Warteschlange (Queue) wird jeder Knoten entfernt, sobald er besucht wird. Dabei ist es unerheblich, in welcher Unterroutine des Algorithmus der Knoten erstmalig bearbeitet wird. Dadurch entsteht der Aufwand $O(|V|)$.

Im weiteren kann jeder Knoten u nur noch so oft besucht werden, wie er eingehende und ausgehende Kanten hat. Im Fall der eingehenden Kanten, also beim Abarbeiten der *while*-Schleife in der Prozedur $SetF(v)$ für einen Vorgänger v von u, kann u nur so oft besucht werden wie er Vorgänger hat. Analog entspricht im Fall der ausgehenden Kanten beim Abarbeiten der Prozedur $SetL(w)$ für Nachfolger w von u die Anzahl der Besuche von u gerade der Anzahl der Nachfolger. Dadurch entsteht der Aufwand $O(|E|)$ des Algorithmus. □

Bemerkung 3.5 Für weitere Bemerkungen siehe W. Hackbusch [Hac95].

Entfernt man alle Knoten mit den Attributen F und L (inklusive der Kanten, die mit ihnen

inzident sind), dann besteht der sich ergebende Graph G_C nur aus Zyklen. In G_C besitzt jeder Knoten wenigstens einen Vorgänger und einen Nachfolger. Desweiteren besteht G_C nur aus starken Zusammenhangskomponenten.

Reduzierte Graph

Aufgrund der one-flow-direction Bedingung und der Planarität des Ausgangsgraphen und somit auch des Graphen G_C existiert für jeden Knoten v genau eine Anordnung (entgegen dem Uhrzeigersinn) des Tupels $\{e_1, \ldots, e_l\}$ von $l \geq 1$ ausgehenden Kanten. Sei dabei o.B.d.A. e_1 die am weitesten rechts liegende Kante und dementsprechend e_l die am weitesten links liegende Kante. Läuft man entgegen dem Uhrzeigersinn durch die mit v inzidenten Kanten, dann ist also e_1 die erste ausgehende Kante. Aus diesem Grund seien im folgenden die Kante $e_1(v)$ mit $e_{right}(v)$ und e_l mit $e_{left}(v)$ bezeichnet.

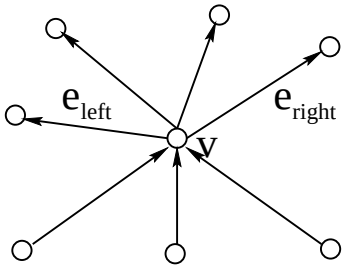


Abbildung 3.4.: e_{right} und e_{left}

Nun zur Definition des reduzierten Graphen.

Definition 3.7 (Reduzierter Graph) Der reduzierte Graph G^+ besteht aus den gleichen Knoten wie G_C, allerdings mit weniger Kanten,

$$V(G^+) := V(G),\ E(G^+) := \{e_{left}(v) \,|\, v \in V(G)\}.$$

Entsprechend gilt für den reduzierten Graphen G^-

$$V(G^-) := V(G),\ E(G^-) := \{e_{right}(v) \,|\, v \in V(G)\}.$$

Bemerkung 3.6

(1) Jeder Knoten in G^+ hat genau einen Nachfolger in G^+.
(2) Jeder Knoten in G^+ kann keinen oder mehrere Vorgänger in G^+ haben.
(3) Jeder Zyklus in G^+ ist auch ein Zyklus in G, aber nicht umgekehrt.
(4) Es gibt mindestens einen Zyklus in G^+, außer $G^+ = \varnothing$.
(5) Die Zyklen in G^+ sind paarweise disjunkt.

Minimale Zyklen

Sei C^+ die Menge der Zyklen in G^+. Im weiteren wird eine surjektive Abbildung von $V(G^+)$ auf C^+ definiert. Jeder Knoten $v \in V$ wird abgebildet auf einen Zyklus $\zeta \in C^+$, also $v \to \zeta(v)$. Da, wie bereits erwähnt, jeder Knoten genau einen Nachfolger besitzt, wird durch

$$v_0 := v \ , \ v_{i+1} := succ(v_i)$$

eindeutig eine unendliche Folge von Knoten in G^+ bestimmt. Da aber $\#V(G^+) < \infty$ ist, muß es ein $j \in I\!N$ geben, so daß $v_j = v_i$ für ein $i < j$ ist. Sei nun j minimal, dann wird durch $\{v_i, v_{i+1}, \ldots, v_j = v_i\}$ ein Zyklus aus C^+ gebildet. Dieser sei mit $\zeta(v)$ bezeichnet. Für beliebige $\zeta \in C^+$ und beliebige $v \in \zeta$ gilt $\zeta(v) = \zeta$. Denn wäre $v \in \zeta \cap \zeta'$ (mit $\zeta, \zeta' \in C^+$), dann wäre sowohl $\zeta(v) = \zeta$ als auch $\zeta'(v) = \zeta$ und damit $\zeta = \zeta'$. Das bedeutet, die Zuordnung

$$V(G^+) \ni v \mapsto \zeta(v) \in C^+$$

ist surjektiv.

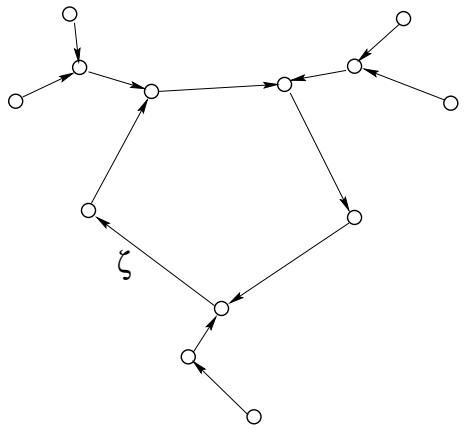


Abbildung 3.5.: G_ζ

Satz 3.2 (G^+)
G^+ zerfällt in disjunkte Untergraphen $G_\zeta, \zeta \in C^+$,

$$G^+ = \dot{\bigcup}\{G_\zeta \,|\, \zeta \in C^+\}.$$

Jeder Untergraph G_ζ ist zusammenhängend und enthält außer ζ keine weiteren Zyklen. Zusätzlich kann G_ζ noch Bäume enthalten, die mit ζ verbunden sind. (Abb.3.5)

Um nun den Begriff des *minimalen Zyklus* einführen zu können, ist zunächst die Definition des sogenannten *Inneren* und *Äußeren* eines Zyklus erforderlich.

Definition 3.8 (linke bzw. rechte Kanten von ζ und v) Seien u, v, w Knoten aus dem Zyklus ζ mit $u = pred(v)$ und $w = succ(v)$. Alle Kanten in dem Bereich zwischen

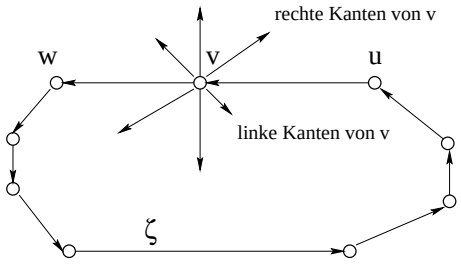


Abbildung 3.6.: linke ($E_{left}(v,\zeta)$) und rechte ($E_{right}(v,\zeta)$) Nachbarn von v

$(v,w) \in E(G)$ und $(u,w) \in E(G)$ (entgegen dem Uhrzeigersinn durchlaufen) werden *linke Kanten* genannt und zur Menge $E_{left}(v,\zeta)$ zusammengefaßt. Analog definiert man $E_{right}(v,\zeta)$ als die Menge aller *rechten Kanten* von v.

Sei G stark zusammenhängend und seien weiter G_i die schwachen Zusammenhangskomponenten von $G\backslash\zeta = \bigcup G_i$. Jeder Teilgraph G_i besitzt einen Knoten v_i, der selbst nicht in ζ liegt, aber damit verbunden ist:

$$\exists\, v_i \in V(G_i)\ \exists\, v \in \zeta : (v_i, v) \in E(G).$$

Ist $(v_i, v) \in E_{left}(v,\zeta)$, so gilt dies auch für jede andere Wahl von (v_i, v). Das bedeutet, daß der Graph G_i der linken Seite von (v,ζ) zugeordnet werden kann, welche man auch als das *Innere* des Zyklus ζ bezeichnet.

Definition 3.9 ($Int(v,\zeta)$ und $Ext(v,\zeta)$) Die Vereinigung aller Untergraphen G_i, die links von ζ liegen, bezeichnet man als das *Innere* oder auch *Interior* von ζ und schreibt $Int(\zeta)$. Analog definiert man das *Äußere* oder auch *Exterior* von ζ, geschrieben $Ext(\zeta)$.

Definition 3.10 ($\overline{Int}(v,\zeta)$ und $\overline{Ext}(v,\zeta)$) Der *Abschluß* (closure) des Inneren und des Äußeren von ζ wird wie folgt definiert:

$$\overline{Int}(v,\zeta) := Int(v,\zeta) \cup \zeta\ , \quad \overline{Ext}(v,\zeta) := Ext(v,\zeta) \cup \zeta.$$

Bemerkung 3.7 (Planarität) Für Zuordnungen 'links' und 'rechts' eines Zyklus ist die Planarität des Graphen G zwingend erforderlich.

Für die Zyklen in G^+ $(C+)$ läßt sich der folgende Satz formulieren.

Satz 3.3
Sei $G = \{V, E\}$ stark zusammenhängend. Dann gilt

$$\forall \zeta \in C^+ \ :\ Int(\zeta) = \varnothing.$$

Beweis. Sei $v' \in Int(\zeta)$ und $u' \in \zeta$. Da G stark zusammenhängend ist, gibt es einen Weg von u' nach v'. Die Folge von Kanten enthält eine Kante (u,v) mit $u \in \zeta$ und $v \in Int(\zeta)$. Da aber die Kante (u,v) links von (u,u'') mit $u'' = E_{left}(u) = succ_{G^+}(u)$ liegt, folgt daraus ein Widerspruch zur Definition von $E_{left}(u)$. □

Aufgrund der Aussage dieses Satzes lassen sich die Zyklen in C^+ wie folgt bezeichnen.

Definition 3.11 (Minimale Zyklen) Die Zyklen in C^+ nennt man minimale Zyklen, da sich in ihrem Inneren kein weiterer Knoten und damit auch kein weiterer Zyklus mehr befindet.

Konzentrische Kreise

Konstruktion

Algorithmus: $ConCyc(G)$

---

(1) $G_0 \leftarrow G\,, i \leftarrow 0$
(2) **do**
(3) $\quad choose\ v \in V(G_i)$
(4) $\quad \zeta_i \leftarrow \zeta(v)$
(5) $\quad G'_{i+1} \leftarrow G_i \backslash \zeta_i$
(6) $\quad Filter(G'_{i+1})$
(7) $\quad G_{i+1} \leftarrow G'_{i+1} \backslash \{F, L-edges\}$
(8) $\quad i \leftarrow i+1$
(9) **while** $G_i \neq \varnothing$

Bemerkung 3.8 Der Aufwand von $ConCyc(G)$ beträgt $O(|V|)$.

Bemerkung 3.9 Der Algorithmus $ConCyc(G)$ ermittelt eine Folge von Zyklen $\{\zeta_0, \zeta_1, \ldots, \zeta_n\}$, wobei G_n der letzte nichtleere Graph ist.

Der Metagraph Z^+

Die durch $ConCyc(G)$ ermittelten Zyklen werden als Knoten eines neuen Graphen Z^+ betrachtet, mit

$$V(Z^+) = \{\zeta_0, \zeta_1, \ldots, \zeta_n\}.$$

Innerhalb dieses so definierten Metagraphen läßt sich nun wie folgt eine partielle Anordnung definieren.

Definition 3.12 (partielle Anordnung in Z^+) Seien $\zeta, \zeta' \in V(Z^+)$. Dann läßt sich durch die Relation

$$\zeta' > \zeta \Longleftrightarrow \left\{\overline{Int}(\zeta) \subset Int(\zeta')\ ,\ \overline{Ext}(\zeta') \subset Ext(\zeta)\right\} \qquad \forall \zeta, \zeta' \in Z^+$$

eine Ordnung in Z^+ definieren.

Bemerkung 3.10 Diese Anordnung ist partiell, weil sie nur für ineinander liegende gleichgerichtete oder nicht ineinanderliegende ungleich gerichtete Zyklen gilt.

Die Kanten in Z^+ werden folgendermaßen gebildet:

$$E(Z^+) = \left\{(\zeta, \zeta') \in Z^+ \times Z^+ \mid \zeta' > \zeta \ \wedge \ \nexists\, \zeta'' \text{ s.d. } \zeta' > \zeta'' > \zeta\right\}.$$

Die Abbildungen 3.7 bis 3.9 zeigen mögliche Anordnungen verschiedener Zyklen und die korrespondierenden Kanten in Z^+.

- Konzentrische ineinanderliegende Zyklen bilden in Z^+ einen Pfad vom innersten zum äußersten Zyklus. (Abb.3.7)
- Besitzt ein Zyklus zwei oder mehrere gleichberechtigte Nachfolger, so führt dies zu einer Aufspaltung in Z^+. Zu beachten ist, daß die Nachfolgezyklen entgegengesetzten Drehsinn haben. (Abb.3.8)
- Ebenso kann ein Zyklus mehrere gleichberechtigte Vorgänger haben, wenn im Inneren des Zyklus mehrere nicht konzentrisch liegende Zyklen mit gleichem Drehsinn liegen. (Abb.3.9)

Bemerkung 3.11 Der Metagraph Z^+ muß aufgrund seiner Konstruktion nicht notwendig planar sein.

Für die Menge der Vorgänger und Nachfolger in Graphen Z^+ werden nun folgende Abkürzungen eingeführt:

$$\begin{aligned}
Pred(\zeta) &:= \left\{\zeta' \in V(Z^+) \mid (\zeta', \zeta) \in E(Z^+)\right\} \quad \text{für } \zeta \in V(Z^+),\\
Succ(\zeta) &:= \left\{\zeta' \in V(Z^+) \mid (\zeta, \zeta') \in E(Z^+)\right\} \quad \text{für } \zeta \in V(Z^+),\\
Bottom(Z^+) &:= \left\{\zeta \in V(Z^+) \mid Pred(\zeta) = \varnothing\right\},\\
Top(Z^+) &:= \left\{\zeta \in V(Z^+) \mid Succ(\zeta) = \varnothing\right\}.
\end{aligned}$$

Über Z^+ lassen sich die folgende Aussagen treffen.

Satz 3.4

(1) *Der Graph Z^+ ist azyklisch.*
(2) *Zwei Zyklen $\zeta, \zeta' \in V(Z^+)$ mit $\zeta' > \zeta$ sind durch genau einen Pfad in Z^+ miteinander verbunden.*
(3) $C^+ = Bottom(Z^+)$.

Beweis. Vgl. W. Hackbusch [Hac95]. □

Antizyklen und konzentrische Antizyklen

Die bisherigen Betrachtungen beziehen sich lediglich auf die E_{left}-Kanten und den dazugehörigen Graphen G^+. Analoge Schlüsse lassen sich aber auch für die E_{right}-Kanten und den entsprechenden Graphen G^- ziehen.

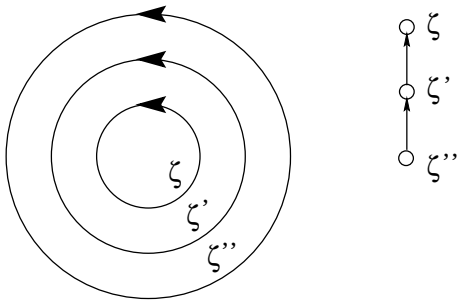


Abbildung 3.7.: $\zeta \to \zeta' \to \zeta''$

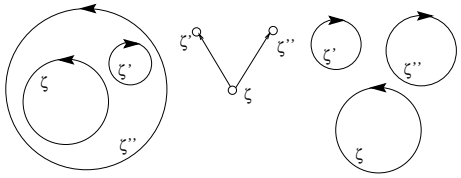


Abbildung 3.8.: $\zeta \to \zeta'$, $\zeta \to \zeta''$

Sei

$$C^- = \{\zeta \mid \zeta \text{ ist Zyklus in } G^-\}$$

die Menge der Zyklen in G^-. Um sie von den Zyklen in G^+ zu unterscheiden, werden sie *Antizyklen* genannt. Als Analogon zu den minimalen Zyklen aus G^+ (d.h. $Int(\zeta) = \varnothing$) kann man die Zyklen aus G^- auch als maximale Zyklen in G bezeichnen, da $\overline{Int}(\zeta) = G$. Zu der Folge von konzentrischen Kreisen aus G^+ (zur besseren Unterscheidung im weiteren $\{\zeta_0^+, \zeta_1^+, \ldots, \zeta_n^+\}$ genannt) läßt sich ein entsprechendes Gegenstück in G^- finden, nämlich $\{\zeta_0^-, \zeta_1^-, \ldots, \zeta_{n'}^-\}$. Der entsprechende Algorithmus zum Bestimmen dieser Folge sucht bei der Bestimmung der Antizyklen zunächst nach den äußersten Zyklen. Im übrigen arbeitet er analog zum Algorithmus $ConCyc(G)$ aus Kapitel 3.5.3 (S. 45).

Nun läßt sich auch hier ein Graph Z^- bilden, der aus den Knoten dieser Zyklusfolge besteht und für den sich wie bei Z^+ ebenfalls eine partielle Ordnung definieren läßt.

Relationen zwischen Z^+ und Z^-

Es soll nun diskutiert werden, in welcher Beziehung die Graphen Z^+ und Z^- zueinanderstehen, die aus den Zyklen bzw. Antizyklen in G gebildet sind. Sieht man von dem trivialen Fall (d.h. im Ausgangsgraphen G hat jeder Knoten genau einen Nachfolger, d.h. G besteht nur aus einem großen Zyklus) ab, dann unterscheiden sich zunächst die resultierenden Graphen G^+ und G^- und damit auch die gefundenen Folgen $\{\zeta_0^+, \zeta_1^+, \ldots, \zeta_n^+\}$ und $\{\zeta_0^-, \zeta_1^-, \ldots, \zeta_{n'}^-\}$. Das bedeutet aber nichts anderes, als daß sich auch die Graphen Z^+ und Z^- unterscheiden. Gesucht wird nun nach Übereinstimmungen der beiden Graphen. Dazu sei folgende Definition 3.13 gegeben.

Definition 3.13 Man sagt, ein Zyklus ζ^- und ein Antizyklus ζ^+ berühren einander, wenn

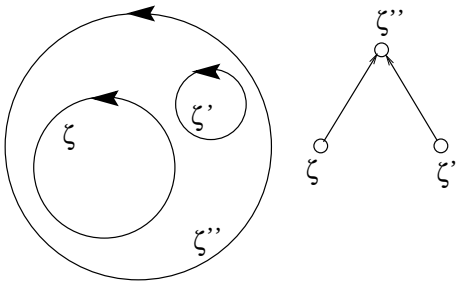


Abbildung 3.9.: $\zeta \to \zeta''$, $\zeta' \to \zeta''$

gilt, daß

$$\zeta^+ \cap \zeta^- \neq \varnothing\,, \quad Int(\zeta^+) \cap \zeta^- = \zeta^+ \cap Ext(\zeta^-) = \varnothing.$$

Diese Relation läßt sich auch schreiben als

$$\zeta^+ \circ \zeta^-.$$

Bemerkung 3.12 Ist die one-flow-condition beim Ausgangsgraphen erfüllt, können bei sich berührenden Zyklen genau zwei Fälle auftreten:
(1) die Zyklen liegen ineinander und sind gleichdrehend oder
(2) die Zyklen liegen nicht ineinander und haben entgegengesetzte Drehrichtung.

Das folgende Lemma 3.1 wird entscheidend in die Definition der Bewertungsfunktion des Approximationsalgorithmus für das Feedback–Vertex–Set eingehen.

Lemma 3.1
Sei $G = \{V, E\}$ ein beliebiger Graph. Dann gilt:
(1) *Für beliebige Zyklen ζ in G gibt es wenigstens einen Zyklus $\zeta^+ \in V(Z^+)$ mit $\zeta^+ \circ \zeta$ und $\zeta^+ \subset Int(\zeta)$.*
(2) *Für beliebige Zyklen ζ in G gibt es wenigstens einen Zyklus $\zeta^- \in V(Z^-)$ mit $\zeta \circ \zeta^-$ und $\zeta^- \subset Ext(\zeta)$.*
(3) *Für beliebige Zyklen $\zeta^+ \in V(Z^+)$ gibt es wenigstens einen Zyklus $\zeta^- \in V(Z^-)$ mit $\zeta^+ \circ \zeta^-$.*

Beweis: (3) folgt aus (1) oder (2), da $\zeta \in G_C$ aus (1) oder (2) insbesondere auch ein Zyklus aus $V(Z^-)$ sein kann. Weiterhin sind (1) und (2) äquivalent - es reicht also zu zeigen, daß (1) gilt.
Sei im folgenden $G_\zeta := \overline{Int}(\zeta)$. Man bestimmt die Folge der konzentrischen Zyklen $\{\zeta_0, \zeta_1, \ldots, \zeta_n\}$ mittels $ConCyc(G)$ (Kap. 3.5.3, S. 45) und erhält somit einen Graphen Z_ζ^+. Nun gilt offensichtlich, daß

$$Z_\zeta^+ \subset Z^+ \quad \text{mit} \quad V(Z_\zeta^+) = \left\{\zeta^+ \in V(Z^+) \,|\, \zeta^+ \subset G_\zeta\right\}.$$

Die äußersten Zyklen von Z_ζ^+ sind diejenigen $\zeta^+ \in V(Z_\zeta^+)$ ohne Nachfolger in Z_ζ^+. Diese Zyklen faßt man zur Menge M zusammen,

$$M := Top\,(Z_\zeta^+).$$

Der Teilgraph

$$G'_\zeta := \overline{Int}(\zeta) \setminus \bigcup \{\overline{Int}(\zeta^+) \mid \zeta^+ \in M\}$$

entspricht der schattierten Fläche in Abb. 3.10 (inklusive des Zyklus ζ, exklusive der Zyklen $\zeta^+ \in M$). Falls nun ein Element aus M den Zyklus ζ berührt, ist das Lemma bewiesen. Wäre das nicht der Fall, müßte ζ noch in G'_ζ enthalten sein, was bedeutet, daß G'_ζ zyklisch ist. Das heißt, es muß ein minimaler Zyklus ζ' in G'_ζ existieren (siehe Bem. 3.6, S. 42). Da aber $\zeta' \in V(Z^+_\zeta)$ gilt, ist dies ein Widerspruch zur Definition von M. □

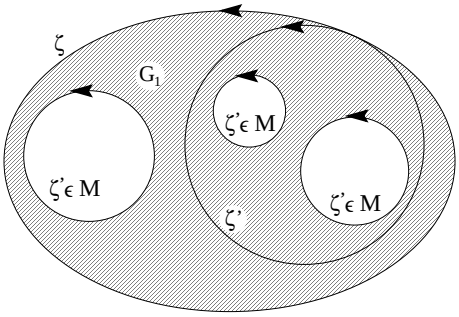


Abbildung 3.10.: Abbildung zum Beweis des Lemmas

Bewertungsfunktion Φ

In diesem Abschnitt soll nun die Bewertungsfunktion beschrieben werden, mit der es im Approximationsalgorithmus möglich ist, den nächsten Knoten für die Aufnahme in das Feedback–Vertex–Set zu bestimmen.

Sei o.B.d.A. $G = \{V, E\}$ stark zusammenhängend und sei weiter $v \in V(G)$. Dann betrachtet man folgende Funktionen:

$$\begin{aligned}
\varphi_1^+(v) &:= \varphi_1^+(v, Z^+) := \#\{\zeta \in V(Z^+) | v \in Ext(\zeta)\}, \\
\varphi_2^+(v) &:= \varphi_2^+(v, Z^+) := \#\{\zeta \in V(Z^+) | v \in \overline{Ext}(\zeta)\}, \\
\varphi^+(v) &:= \varphi^+(v, Z^+) := \frac{1}{2}[\varphi_1^+(v, Z^+) + \varphi_2^+(v, Z^+)].
\end{aligned}$$

Dabei unterscheiden sich $\varphi_1^+(v)$ und $\varphi_2^+(v)$ nur für jene v, die auf dem Rand des Zyklus $\zeta \in V(Z^+)$ liegen. $\varphi^+(v)$ bildet den Durchschnitt der beiden ersten Funktionen und nimmt somit Werte aus der Menge $\{0, \frac{1}{2}, 1, \frac{3}{2}, 2, \dots\}$ an. Analog setzt man diese Funktionen für Z^-:

$$\begin{aligned}
\varphi_1^-(v) &:= \varphi_1^-(v, Z^-) := \#\{\zeta \in V(Z^-) | v \in Int(\zeta)\}, \\
\varphi_2^-(v) &:= \varphi_2^-(v, Z^-) := \#\{\zeta \in V(Z^-) | v \in \overline{Int}(\zeta)\}, \\
\varphi^-(v) &:= \varphi^-(v, Z^-) := \frac{1}{2}[\varphi_1^-(v, Z^-) + \varphi_2^-(v, Z^-)].
\end{aligned}$$

Mit diesen Funktionen läßt sich nun die eigentliche Bewertungsfunktion Φ definieren.

Definition 3.14 (Bewertungsfunktion Φ) Sei $G = \{V, E\}$ stark zusammenhängend und $v \in V$ beliebig. Dann läßt sich für v folgende Bewertungsfunktion definieren:

$$\Phi(v) := \Phi(v, Z^+, Z^-) := \varphi^+(v, Z^+) + \varphi^-(v, Z^-).$$

Bemerkung 3.13 $\Phi(v)$ gibt also an, bei wievielen Zyklen sich der Knoten v durchschnittlich im Äußeren befindet plus die durchschnittliche Anzahl von Antizyklen, bei denen er sich im Inneren befindet.

In der obigen Definition läßt man nur Punkte (Knoten) aus $V(G)$ zu. Bettet man den Graphen G in den $\mathbb{R}^2$ ein, so lassen sich $Ext(\zeta), \overline{Ext}(\zeta), Int(\zeta), \overline{Int}(\zeta)$ als offene bzw. geschlossene Gebiete im $\mathbb{R}^2$ interpretieren. Dies würde eine Definition von Φ für beliebige Punkte $v \in \mathbb{R}^2$ zulassen. Insbesondere bliebe $\Phi(v)$ immer noch definiert, wenn man v aus dem Graphen G entfernt.

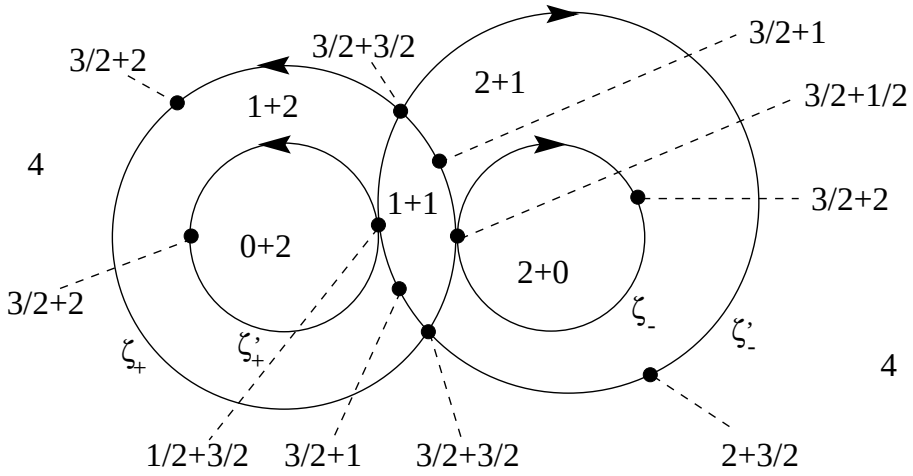


Abbildung 3.11.: Werte der Bewertungfunktion Φ

Die Funktion Φ ist nach oben beschränkt durch die maximale Anzahl von Zyklen im Graphen G. Nach Definition wird Φ maximal an denjenigen Stellen, die außerhalb aller Zyklen bzw. Antizyklen liegen.

Andererseits ist Φ genau dann 0, wenn der Graph keine Zyklen mehr besitzt. Dadurch ist im Algorithmus ein Abbruchkriterium gegeben, da der Graph für $\Phi = 0$ wie gewünscht azyklisch ist.

Lemma 3.2

Das Minimum von Φ wird im Inneren zwischen Zyklen und Antizyklen angenommen und ebenso in den Berührpunkten von Zyklen aus C^+ und C^-, d.h. auf der Knotenmenge T,

$$T := \left\{ v \in V(G) \,|\, v \in \zeta_+ \cap \zeta_- \,;\, \zeta_+ \circ \zeta_- \,,\, \zeta_+ \in V(Z^+) \,,\, \zeta_- \in V(Z^-) \right\}$$

Beweis: Vgl. W. Hackbusch [Hac95]. □

Als letztes sei nun noch ein Lemma angegeben, durch das sich die Laufzeit des Algorithmus verbessern läßt.

Lemma 3.3
Wenn sich ein minimaler Zyklus ζ^+ und ein minimaler Antizyklus ζ^- berühren und beide Nachfolger besitzen, berühren sich diese ebenfalls.

Beweis: Vgl. W. HACKBUSCH [Hac95]. □

Konstruktion des Feedback–Vertex–Set

Zunächst vereinfacht man die Notation von Φ zu

$$\Phi(v, G) := \Phi(v, Z^+, Z^-).$$

Dabei seien die $Z^{\pm}$ aus G entsprechend den Algorithmen aus den vorangegangenen Kapiteln konstruiert. Während der Laufzeit des Algorithmus ändert sich der Graph G (der jeweilige neue Untergraph sei G_i genannt) und deshalb auch die zugehörigen $Z^{\pm}$. Desweiteren sei die Definition

$$\Phi_{min}(G_i) := min\left\{\Phi(v, g_i) \,|\, v \in I\!R^2\right\}$$

gegeben. Die Startwerte für den Graphen sowie das Feedback–Vertex–Set sind

$$F_0 := \varnothing\,, \quad G_0 := G.$$

Die Menge $T = T(G_i)$ wurde in einem vorangegangenen Lemma definiert und ändert sich ebenfalls in jedem Schritt in Abhängigkeit von G_i.

Algorithmus: FVS

(1) $i \leftarrow 0\,;\ F_0 \leftarrow 0\,;\ G_0 \leftarrow 0$
(2) **while** $\Phi_{min}(G_i) > 0$ **do**
(3) *choose any* v_i *from* $\{v \in T(G_i) \,|\, \Phi(v, G_i) = \Phi_{min}(G_i)\}$
(4) $F_{i+1} \leftarrow F_i \cup \{v_{i+1}\}$
(5) $G_{i+1} \leftarrow G_i \backslash \{v_{i+1}\}$
(6) $i \leftarrow i + 1$
(7) **od**
(8) $Feedback--Vertex--Set \leftarrow F_i$

In diesem Algorithmus lassen sich noch Verbesserungen einbringen, z.B. läßt sich die Wahl irgendeines Knotens v aus Zeile (3) noch weiter präzisieren duch das folgende Lemma 3.4.

Lemma 3.4
Für die Berechnung von Φ_{min} reicht es aus, nur die $v \in T^+$ zu betrachten.

$$T^+(G_i) := T(G)_{|B^+(G_i)} = \left\{v \in V(G_i) \,|\, v \in \zeta_+ \cap \zeta_-\,,\ \zeta_+ \in B^+(G_i)\,,\ \zeta_- \in V(Z^-(G_i))\right\}$$

mit $B^+(G_i) := Bottom(Z^+(G_i))$. Wegen der Symmetrie von Z^+ und Z^- gilt diese Aussage genauso für minimale Antizyklen.

Beweis: Das Lemma wird durch Induktion über die Graphen $Z^{\pm}$ und G_i bewiesen. Vgl. W. HACKBUSCH [Hac95]. □

Quasioptimalität des Feedback–Vertex–Set

Um Aussagen über die Größe des vom Algorithmus gefundenen Feedback–Vertex–Set treffen zu können, muß diskutiert werden, wie sich die konzentrischen Kreise $Z^{\pm}$ verändern, wenn ein Knoten $v \in V(G)$ aus dem Graphen herausgenommen wird.

Wird ein Knoten gelöscht, der nicht auf einem Zyklus aus Z^+ oder Z^- liegt, ändern sich diese natürlich nicht. Sei daher nun

$$v \in \zeta \in V(Z^+) \quad \text{und} \quad \zeta_i^v \in Pred(\zeta)\,, \quad 1 \le i \le I.$$

Die Menge $Succ(\zeta_i^v)$ besteht dann aus den Nachfolgern dieser Vorgängerzyklen und ist für alle i gleich, außerdem enthält sie natürlich den Zyklus ζ selbst. Man setzt nun

$$G' := G \backslash \bigcup_i \overline{Int}(\zeta_i^v).$$

Der Graph G' wird gebildet aus dem Graphen G ohne die abgeschlossene Menge aller inneren, d.h. Vorgängerzyklen von ζ. Schränkt man nun G' auf die starken Zusammenhangskomponenten ein, ergibt sich G'_C. Falls G'_C nicht nur aus einer starken Zusammenhangskomponente besteht, kann ζ nicht das einzige Element von $Succ(\zeta_i^v)$ sein, und es ist

$$Succ(\zeta_i^v) \backslash \{\zeta\} \neq 0.$$

Man betrachtet nun den Graphen $\hat{G}$ mit

$$\hat{G} := G \backslash \left\{ \bigcup_i \overline{Int}\,(\zeta_i^v) \cup \bigcup \left\{ \overline{Ext}(\zeta') \,|\, \zeta' \in \{Succ(\zeta_i^v) \backslash \{\zeta\}\} \right\} \right\}.$$

Das ist derjenige Graph, der entsteht, wenn man aus G die abgeschlossene Menge aller innen liegenden Vorgängerzyklen von ζ herausnimmt und ebenso das abgeschlossene Äußere aller Nachfolgerzyklen, aber nicht ζ selbst. Per Definition ist dann ζ ein minimaler Zyklus in $\hat{G}$.

Seien $G' = G \backslash \{v\}$ und $\hat{G}' = \hat{G} \backslash \{v\}$ die Graphen ohne den Knoten v. Dann können zwei Fälle auftreten,

(1) es gibt minimale Zyklen ζ'_j $(1 \le j \le J)$ in $\hat{G}'$ mit $\zeta'_j \cup \zeta \neq 0$ oder

(2) alle minimalen Zyklen $\zeta'_j \in \hat{G}'$ sind mit ζ disjunkt.

Der erste Fall ist gleichbedeutend damit, daß der Zyklus ζ, auf dem v liegt, einen minimalen Zyklus in $\hat{G}'$ berührt. Berühren sie sich nicht, entspricht das dem zweiten Fall.

Der zweite Fall, bei dem die Zyklen $\zeta'_j \in \hat{G}'$ bereits konzentrische Zyklen in G sind, ist ein Sonderfall, für den

$$Z^+(G') = Z^+(G) \backslash \{\zeta\}$$

gilt. Dies bedeutet, daß ζ beim Entfernen von v nicht durch einen anderen Zyklus ersetzt wird. Der Standardfall ist jedoch die in Fall (1) auftretende Konstellation, bei der ζ durch den Zyklus ζ' ersetzt wird. Man kann also schreiben

$$v \in \overline{Int}(\zeta'_j) \supset Int(\zeta), \quad \zeta \circ \zeta'_j, \qquad (1 \leq j \leq J).$$

Da mit $v \in \zeta$ der Knoten v im abgeschlossenen Äußeren aller seiner kleineren Vorgängerzyklen liegt, führt dies zur folgenden Bemerkung 3.14.

Bemerkung 3.14 Sei $v \in \zeta V(Z^+(G))$. Die Vorgängerzyklen $\zeta^{pre} \in V(Z^+(G))$ sind wieder konzentrische Zyklen in der Menge $Z^+(G')$ mit $G' := G \setminus \{v\}$. Diese Zyklen sind dadurch charakterisiert, daß der Knoten v in ihrem Äußeren liegt. Daher gilt

$$\begin{aligned} \varphi_1^+(v, Z^+(G')) &= \varphi(v, Z^+(G)), \\ \varphi_2^+(v, Z^+(G')) &= \varphi(v, Z^+(G)) - 1. \end{aligned}$$

Insgesamt ergibt sich also, daß

$$\varphi^+(v) = \varphi^+(v, Z^+(G')) = \varphi^+(v, Z^+(G)) - \frac{1}{2}.$$

Analog gilt für $v \in \zeta V(Z^-(G))$ und $G' := G \setminus \{v\}$:

$$\varphi^-(v) = \varphi^-(v, Z^+(G')) = \varphi^-(v, Z^+(G)) - \frac{1}{2}.$$

Der im letzten Abschnitt vorgestellte Approximationsalgorithmus wählt Knoten v aus $T(G)$ aus, das heißt, $v \in \zeta_+ \cap \zeta_-$ mit $\zeta_+ \in V(Z^+)$ und $\zeta_- \in V(Z^-)$. Dann gelten die Gleichungen aus der obenstehenden Bemerkung und führen auf das nächste Lemma.

Lemma 3.5

(1) *Sei $v \in T(G)$ und $G' := G \setminus \{v\}$. Dann gilt*

$$\Phi(v, G') = \Phi(v, G) - 1.$$

(2) *Die Wahl von $v \in T$ als nächsten herauszunehmenden Knoten im zentralen Algorithmus führt auf*

$$\Phi_{min}(G_{i+1}) = \Phi_{min}(G_i) - 1.$$

(3) *Die WHILE-Schleife im zentralen Algorithmus terminiert nach höchstens $i = \Phi_{min}(G)$ Schritten. Insbesondere ist die Anzahl der Feedback–Vertices durch $\Phi_{min}(G)$ nach oben beschränkt mit*

$$\#FVS \leq \Phi_{min}(G).$$

Beweis:

(1) wurde mit der obigen Bemerkung 3.14 bereits gezeigt.

(2) Hier ersetzt man in der genannten Bemerkung G durch G_i und G' durch G_{i+1}. Wenn man $v = v_i$ setzt, gilt

$$\Phi(v, G_i) = \Phi_{min}(G_i)$$

und damit

$$\Phi_{min}(G_{i+1}) \leq \Phi(v, G_{i+1}) - 1 = \Phi_{min}(G_i) - 1.$$

(3) Nach höchstens $\Phi_{min}(G_i) - 1$ Schritten gilt wegen (2), daß $\Phi_{min}(G_i) = 0$ ist. Das resultierende Feedback–Vertex–Set $FVS := F_i$ hat höchstens $i \leq \Phi_{min}(G)$ Elemente.

□

Nun lassen sich Aussagen über die Quasi-Optimalität des Feedback–Vertex–Set treffen. Sei D eine beliebige Menge disjunkter Zyklen in einem Graphen G, dann enthält jedes Feedback–Vertex–Set F (mindestens) einen Knoten v_ζ für jeden Zyklus $\zeta \in D$. Daher ist die Kardinalität eines beliebigen Feedback–Vertex–Set trivialerweise größer oder gleich der Anzahl der disjunkten Zyklen von D. Formal schreibt man

$$\begin{aligned} \mu &:= max\,\{\#D \mid D \text{ ist eine Menge disjunkter Zyklen in } G\}, \\ \mu_{opt} &:= min\,\{\#F \mid F \text{ ist ein Feedback–Vertex–Set in } G\}, \end{aligned}$$

da bei μ_{opt} zusätzlich noch die Feedback–Vertices der nichtdisjunkten Zyklen hinzukommen. Ein Feedback–Vertex–Set F mit $\#F = \mu_{opt}$ nennt man dann *optimal*.
Die Mengen Z^+ und Z^- stellen spezielle Mengen disjunkter Zyklen dar. Es gilt deshalb trivialerweise die Ungleichung

$$\#V(Z^+) + \#V(Z^-) \leq 2\mu.$$

Aus dem Abschnitt 3.5.3 (S. 49) über die Bewertungsfunktion sind bereits die folgenden Ungleichungen bekannt

$$\text{'maximale Anzahl disjunkter Zyklen' in } G \leq \#V(Z^+) + \#V(Z^-) \leq 2\mu \leq 2\mu_{opt}.$$

wobei $\Phi = \Phi(v, G)\,\forall v \in G$. Diese Ungleichungen lassen sich umschreiben in

$$\mu \leq \Phi \leq \#V(Z^+) + \#V(Z^-) \leq 2\mu \leq \mu_{opt}.$$

Insbesondere gilt

$$\Phi_{min}(G) \leq 2\mu \leq 2\mu_{opt}.$$

In jedem Fall terminiert der Approximationsalgorithmus von W. Hackbusch nach höchstens $2\mu \leq \mu_{opt}$ Schritten. Wenn die Kardinalität eines Feedback Vertex Sets, wie im vorliegenden Algorithmus 2, optimal ist bis auf einen konstanten Faktor, so nennt man dieses Feedback–Vertex–Set *quasi-optimal*.

3.5.4. Lösungsverfahren mittels Feedback–Vertex–Set - Das Verfahren von W. Hackbusch

In diesem Abschnitt soll das Lösungsverfahren von W. Hackbusch [Hac95] für den zyklischen Fall mittels des Approximationsalgorithmus aus Abschnitt 3.5.3 (S. 39) beschrieben werden.
Das Verfahren basiert auf der Schur–Komplement–Methode. Die eventuell vorhandenen Zyklen im Konvektionsgraphen werden mittels des *FVS*–Algotihmus auf Seite 51 aufgeschnitten und der verbleibende (azyklische) Graph wird entlang der Flußrichtung numeriert, z.B. durch Tiefensuche. So erhält man eine Einteilung der Steifigkeitsmatrix in

Blöcke, auf die sich die Schur–Komplement–Methode zur Lösung anwenden läßt. Der Algorithmus ergibt sich also wie folgt:

Algorithmus: Downwind Numbering - das Verfahren von W. HACKBUSCH

---

(1) *Erzeuge den Konvektionsgraphen mit einer der Kantenmengen* $E_0, \ldots, E_6$.
(2) *Berechne ein quasioptimales Feedback–Vertex–Set mittels Approximationsalgorithmus aus Abschnitt 3.5.3.*
(3) *Entferne die Feedback–Vertices aus dem Konvektionsgraphen.*
(4) *Finde eine flußorientierte Numerierung des restlichen (azyklischen) Graphen mittels Tiefensuche.*
(5) *Löse das umnumerierte Problem mittels Schur–Komplement–Methode.*

Mit der folgenden Schur–Komplement–Methode:

Sei C eine maximale Zusammenhangskomponente im Konvektionsgraphen und $A := A_k^{ii}$ der zugehörige Diagonalblock. Weiter sei $F \subset C$ eine beliebiges Feedback–Vertex–Set von C. Dann induziert die Zerlegung $C = C' \cup F$ bei geeigneter Numerierung der Eckpunkte die folgende Blockstruktur von A:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}. \tag{3.5}$$

Hierbei sei A_{22} der Diagonalblock zu den Feedback–Vertices und A_{11} der entsprechende Block zu den Eckpunkten in C'. Die Dimension von A_{22} stimmt folglich mit der Anzahl der Feedback–Vertices überein.

Man setzt nun voraus, daß sowohl A als auch A_{11} invertierbar sind, was für hinreichend kleine h der Fall ist. Die entsprechende Block–LU–Zerlegung von A führt dann auf die Darstellung

$$A = \begin{pmatrix} A_{11} & 0 \\ A_{21} & I \end{pmatrix} \begin{pmatrix} 0 & A_{11}^{-1}A_{12} \\ I & S \end{pmatrix} \tag{3.6}$$

mit dem sogenannten Schur–Komplement

$$S = A_{22} - A_{21}A_{11}^{-1}A_{12}. \tag{3.7}$$

Nun ist mit A und A_{11} auch das Schurkomplement S nichtsingulär. Durch Invertierung der beiden Blockdreiecksmatrizen in (3.6) erhält man daher für A^{-1} die Darstellung

$$A^{-1} = \begin{pmatrix} I & A_{11}^{-1}A_{12}S^{-1} \\ 0 & S^{-1} \end{pmatrix} \begin{pmatrix} A_{11}^{-1} & 0 \\ -A_{21}A_{11}^{-1} & I \end{pmatrix} \tag{3.8}$$

Jedes Gleichungssystem $Ax = b$ mit der Blockstruktur

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \tag{3.9}$$

kann mit folgendem Algorithmus direkt gelöst werden.

Algorithmus: Schur–Komplement–Löser

(1) $S = A_{22} - A_{21}A_{11}^{-1}A_{12}$
(2) $y_1 = A_{11}^{-1}b_1$
(3) $y_2 = b_2 - A_{21}y_1$
(4) $x_2 = S^{-1}y_2$
(5) $x_1 = y_1 - A_{11}^{-1}A_{12}x_2$

Damit läßt sich also die Lösung des linearen Gleichungssystems $Ax = b$ im wesentlichen auf die Berechnung des Schur–Komplements S sowie auf die Lösung von insgesamt drei Gleichungssystemen geringerer Dimension zurückführen, davon eins mit Matrix S und je zwei mit Matrix A_{11}. Allerdings ist für die Berechnung von S für jeden Feedback–Vertex ein weiteres Gleichungssystem mit der Matrix A_{11} zu lösen. Die Anwendung dieses Verfahrens macht also nur Sinn, wenn das Feedback–Vertex–Set im Vergleich zur Größe der maximalen Zusammenhangskomponente relativ klein ist und sich Gleichungssysteme mit der Matrix A_{11} effizient lösen lassen. Letzteres ist der Fall bei dominanter Konvektion und einer Umnumerierung der Unbekannten in Flußrichtung, da A_{11} gerade dem zyklusfreien Teil des Konvektionsgraphen entspricht.

4. Implementation der Verfahren

In diesem Kapitel wird ein wesentlicher Teil der vorliegenden Diplomarbeit beschrieben. Es soll die Implementation der Numerierungsalgorithmen aus Kapitel 3 (S. 31) erläutert werden. Die Darstellung beginnt mit einigen allgemeinen Bemerkungen, in denen auch auf die zeitliche Entwicklung der Arbeit eingegangen wird. Daran schließen die Darstellungen der Datenvisualisierung und Generierung des Konvektionsgraphen an. Unter den Beschreibungen der Numerierungsalgorithmen nimmt die des umfangreiche HACKBUSCH–Algorithmus den größten Platz ein, vor der der Tiefensuche und des BEY–Verfahrens.

4.1. Allgemeine Bemerkungen

Im Rahmen dieser Diplomarbeit wurde an einem Softwarepaket namens *ADR* mitentwickelt. Der Name steht für **A**dvection–**D**iffusion–**R**eaction (zu deutsch Konvektion–Diffusion–Reaktion) und bezeichnet die Art der betrachteten Probleme. Es handelt sich hierbei um einen Finite–Elemente–Code, der in der objektorientierten Programmiersprache *C++* entwickelt wird. Diese macht einen konsequent modularen Aufbau der Software möglich. Weitere Gründe sprechen ebenfalls für *C++* als die Programmiersprache der Wahl. So verfügt sie über eine umfangreiche Standardbibliothek — die STL [Jos96]; sie ist als Weiterentwicklung der weitverbreiteten Sprache *C* dementsprechend populär und portabel und hat im Vergleich zu anderen objektorientierten Sprachen wie z.B. *Java* einen deutlichen Geschwindigkeitsvorteil bei rechenintensiven numerischen Anwendungen.
Der objektorientierte *ADR* –Code beinhaltet sämtliche Schritte, die zur Lösung eines elliptischen Randwertproblems mittels der FEM nötig sind. Im einzelnen sind das

- ein **Funktionenparser**, mit dem sich die Beschreibung des Problems und des Gebietes komfortabel in Eingabedateien angeben läßt. Dadurch entfällt eine Neucompilation bei veränderter Problemgleichung.

- ein **Gittergenerator**, mit dem sich 2D–Gitter mittels Advancing–Front– bzw. Delaunay–Methode generieren lassen. Siehe dazu J. KOHLAMMER [Koh00].

- ein **Netz**, das mit einer adaptiven Struktur Rot–Grün–Verfeinerungen ermöglicht. Siehe dazu J. KOHLAMMER [Koh00].

- ein **Diskretisierer** zur Galerkin–Diskretisierung des kontinuierlichen Problems auf $P1$–Elementen unter Anwendung von SUPG– und Shock–Capturing–Stabilisierung.

- eine **Matrix– und Löserbibliothek** zur iterativen Lösung des diskreten Problems. Derzeit implementiert sind CG–, SOR–, SSOR– und GMRES–Verfahren sowie ILU– und Downwind–Numbering–Vorkonditionierer.

Derzeit umfaßt *ADR* ca. 40000 Zeilen Quellcode, wovon etwa ein Viertel im Rahmen dieser Arbeit implementiert wurde. *ADR* wird seit ca. einem Jahr entwickelt.

Da mit der Entwicklung des Codes erst in der Entstehungszeit dieser Arbeit begonnen wurde, mußten die beschriebenen Algorithmen zunächst „stand alone“ bzw. auf Basis der bereits vorhandenen Software *PNS* (**P**arallelized solution of **N**avier–**S**tokes–equations) implementiert werden [ea].

Einige der getroffenen Entscheidungen zur Implementation lassen sich deshalb heute als „historisch bedingt“ erklären. So wurde zunächst mit der Implementation des geometrischen Downwind Numbering begonnen, da sich der algebraische Ansatz mit *PNS* nicht verwirklichen ließ. Hier war es lediglich möglich, die Numerierung der Gitterpunkte zu permutieren, um auf diesem Wege eine Umstrukturierung der diskretisierten Matrix im gewünschte Sinne zu erreichen. Die Gitter selbst werden in *PNS* extern mit *GeoMesh* erzeugt und liegen im *netCDF*–Format vor [net]. Das Format ermöglicht Zugriffe auf die Knoten-ID's sowie auf Nachbarknoten. Durchläufe über sämtliche benachbarten Knoten eines Knoten v der Triangulierung sind nur beschränkt möglich. Dies ist aber zur Erzeugung eines Konvektionsgraphen erforderlich, wie in Abschnitt 3.2 (S. 32) beschrieben.

Aufgrund dieser Erkenntnisse wurde nicht weiter an einer Implementation innerhalb von *PNS* gearbeitet. Zu diesem Zeitpunkt existierten erste Ansätze zu *ADR*, die sehr vielversprechend waren. Sehr frühzeitig besaß *ADR* eine eigenständige Gittergenerierung mit wesentlich komfortableren Zugriffsmöglichkeiten als vorher mit *PNS* . Deshalb wurde auch der geometrische Ansatz zur Generierung eines Konvektionsgraphen weiter verfolgt.

An dieser Stelle muß erwähnt werden, daß es letztlich keine Rolle spielt, wie der Konvektionsgraph zu Stande kommt. Die implementierten Graphenalgorithmen haben ein klar definiertes API (**A**pplication **P**rogramming **I**nterface), das durch den modularen Aufbau des objektorientierten *ADR* –Codes unterstützt wird. Mit anderen Worten, es ist jederzeit ohne großen Aufwand möglich, weitere Verfahren zur Generierung eines Konvektionsgraphen zu implementieren, um so andere Aspekte bei der Auswahl von Kanten der Triangulierung bzw. Einträgen in der Steifigkeitsmatrix zu berücksichtigen. Bei der Implementation wurde stets Wert auf die Erweiterbarkeit und Flexibilität des Codes gelegt.

Nach einem Jahr *ADR* –Entwicklung ist dies die zweite Diplomarbeit, die daraus hervorgeht und die erste Arbeit, die den kompletten Code benutzt, was entsprechenden größeren Aufwand für Testläufe und Debugging nach sich zog.

4.2. Datenvisualisierung

Wenn es um die Implementation komplexer Algorithmen wie der des Approximationsalgorithmus von Hackbusch geht, ist es unerläßlich, die Funktionsweise der Algorithmen in geeigneter Weise zu visualisieren. Da es sich hierbei ausschließlich um Graphenalgorithmen handelt, ist eine bildliche Darstellung von Graphen erforderlich. Die Sprache *C++* verfügt leider nur über beschränkte graphische Darstellungsmöglichkeiten. Den Ausweg bildet hier *PostScript* [ps]. Dies ist, wie der Name schon sagt, eine Script-Sprache. In dieser Arbeit wurde PostScript Level 2 der Firma Adobe verwendet, das sehr mächtige Tools für die Darstellung von Texten, Grafiken und Bildern zu Verfügung stellt.

4.2.1. Postscript–Darstellung von Graphen

Die Darstellung reduziert sich zunächst auf Knoten und Kanten. Will man allerdings weitere Informationen wie die Gewichtung von Knoten oder Einfärbungen von Kanten unterbringen, wird die Aufgabe schon komplizierter. Da die Ausgabe für beliebige Graphen mit beliebiger geometrischer Lage der Knoten und Kanten möglich sein mußte, entstand die umfangreiche PostScript–Makro–Bibliothek *graph_makros.eps*.

Dazu muß zunächst gesagt werden, daß PostScript auf einem sehr niedrigen Level arbeitet, was einerseits eine sehr hohe Effizienz zur Folge hat, aber andererseits nicht sehr anwenderfreundlich in der Programmierung ist.

Ein Kernstück der Sprache sind Polygonzüge. Das bedeutet, daß sämtliche geometrischen Formen durch diese Polygonzüge beschrieben werden müssen. PostScript bietet für das Zeichnen von Kreisen, Pfeilen usw., also für die Darstellung von Knoten und Kanten eines Graphen erforderlichen Elementen, von sich aus keine Tools.

Die Routine zum Zeichnen eines ausgefüllten Kreises sieht z.B. wie folgt aus:

```
/circle {
3 dict begin
/r exch def /y exch def /x exch def
x r add y moveto
x y r 0 360 arc
fill stroke
end
} def
```

Dabei werden die PostScript–Befehle immer in postfix–Notation gegeben. Damit läßt sich die Funktion zur Darstellung eines Knoten mit den Koordinaten (x, y) folgendermaßen formulieren:

```
/xyNode {
3 dict begin
% Auslesen der Funktionsparameter
/x exch def /y exch def /subnumber exch def /number exch def
% Aufruf der Funktion circle mit den Koordinaten (x, y) und Radius radius
x y radius circle
normalFont
1 defaultRot sub 1 defaultGelb sub 1 defaultBlau sub setrgbcolor
newpath
x number stringwidth pop 2 div sub
y scaleMin 1.7 mul sub
moveto
number show
smallFont
newpath
x subnumber stringwidth pop 2 div sub
y scaleMin 3.8 mul sub
```

```
moveto
subnumber show
defaultRot defaultGelb defaultBlau setrgbcolor
end
} def
```

Damit können Knoten mit der Bezeichnung *number* an den Koordinaten (x, y) gezeichnet werden. Ähnliche Funktionen existieren für gerichtete und ungerichtete Kanten, wobei hier die verschiedene Lage der Knoten zueinander berücksichtigt werden muß, damit z.B. Pfeile mittels Polygonzügen richtig darstellen werden können.

Für die Darstellung des Beispielgraphen aus Abbildung 2.1 (S. 20) ist das folgende Inputfile erforderlich:

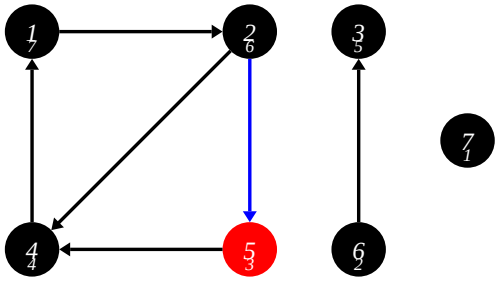


Abbildung 4.1.: Beispielgraph

```
%!PS-Adobe-3.0 EPSF-3.0
// ...
(graph_makros.eps) run
// ...
(1) (7) 2 0 n
(2) (6) 2 2 n
(3) (5) 2 3 n
(4) (4) 0 0 n
(5) (3) 0 2 1.0 0.0 1.0 nrgb
(6) (2) 0 3 n
(7) (1) 1 4 n
2 0 2 2 e
2 2 0 2 0.0 0.0 1.0 ergb
0 2 0 0 e
0 0 2 0 e
2 2 0 0 e
```

```
0 3 2 3 e
// ...
```

Nachdem die Makro–Bibliothek *graph_makros.eps* geladen ist, werden zunächst die Knoten mit ihren Koordinaten angegeben. Dabei ist es möglich, eine zweite Subnummer an den Knoten zu vergeben, um zum Beispiel im TARJAN–Algorithmus gefundene Zusammenhangskomponenten markieren zu können. Nach der Knotenbezeichnung und den Koordinaten erfolgt der Funktionsaufruf n aus der Makro–Bibliothek, um den Knoten in der Standardfarbe zu zeichnen. Entsprechend dazu bietet *nrgb* die Möglichkeit, den Knoten gemäß *RGB*–Farbskalierung einzufärben. Bei den Kanten ist es erforderlich, die Koordinaten des Start- bzw. Endpunktes anzugeben. Der Aufruf e zeichnet analog zu den Knoten eine gerichtete Kante in der Standardfarbe und *ergb* ermöglicht zusätzliche Farbangaben. Schleifen und Doppelkanten wurden nicht implementiert, da sie nicht benötigt wurden.

Sämtliche Ausgaben von Graphen in diesem Abschnitt wurden mit dieser Makro–Bibliothek erzeugt.

4.2.2. Postscript–Darstellung von Matrizen

Ziel der Numerierungsalgorithmen soll es sein, eine bestimmte Matrixstruktur zu erreichen, z.B. die untere Blockdreiecksgestalt. Dazu ist es wünschenswert, die Einträge der Matrix gerade für großere Matrizen grafisch darzustellen, da eine Textausgabe der Matrix sehr schnell an Übersichtlichkeit verliert.

Auch zu diesem Zweck wurde auf *PostScript* zur Visualisierung zurückgegriffen. Das Problem bei den Matrizen war dabei im Gegensatz zu den Graphen weniger die Darstellung verschiedener geometrischer Formen, sondern die kompakte Speicherung der Einträge. Denn pro Matrixeintrag müssen drei Farbwerte abgespeichert werden, man erhält also allein dadurch die dreifache Größe der Ausgangsmatrix, was schon für kleine Matrizen sehr schnell in den Megabyte–Bereich an Speicherplatzbedarf geht.

PostScript bietet auch bei diesem Problem bereits eine Lösung, nämlich die Funktion *readhexstring*, die in der Lage ist, Strings der Länge 256 mit hexadezimalen Einträgen zu interpretieren.

Zusätzlich dazu wurde in der Makro–Bibliothek zur Matrixausgabe ein eigener, sehr effizienter Komprimierungsalgorithmus implementiert, der die Größe der Postscriptdatei auf ein Zehntel reduziert. Die Idee ist, wie bei herkömmlichen Packprogrammen üblich, Sequenzen von aufeinanderfolgenden Einträgen gleichen Wertes geeignet zusammenzufassen.

Die Abbildung 4.2 zeigt die PostScript–Ausgabe einer Matrix.

Es handelt sich hierbei um eine 20×20 Blockmatrix mit quadratischen Blöcken der Dimension 10 und diagonaldominanten Eintragen sowie einem Störeintrag in der rechten oberen Ecke. Zusätzlich dazu kann man eine Skala anzeigen lassen, die den Farbverlauf vom minimalem zum maximalem Eintrag verdeutlicht. Der Farbverlauf ist natürlich beliebig wählbar.

Diese PostScript–Ausgabe der Matrix wurde sowohl in *BLANC*, der Löserbibliothek von *PNS* , als auch in *ADR* implementiert. Der Aufruf für *ADR* lautet:

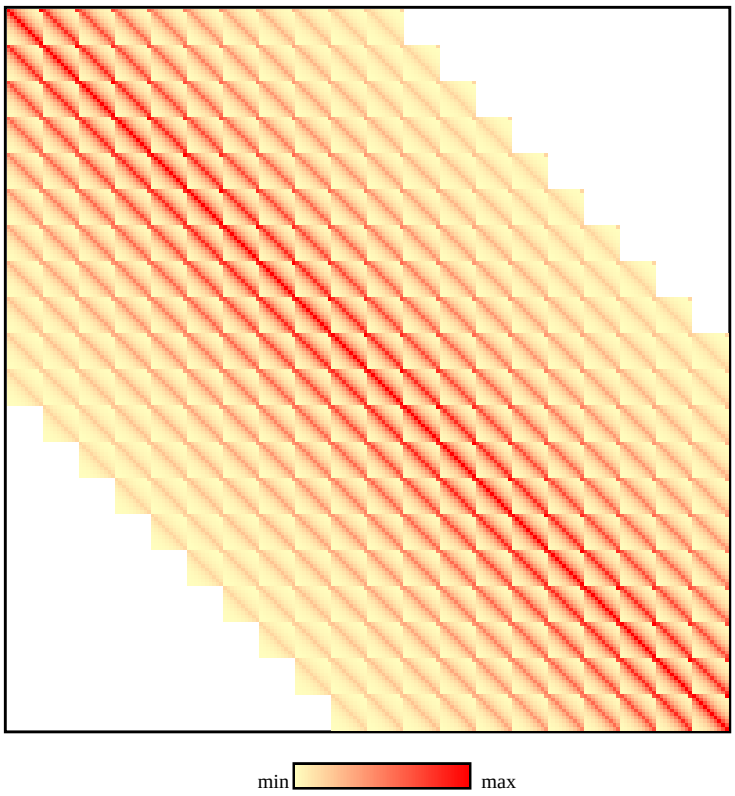


Abbildung 4.2.: Beispielmatrix

```
writeNewSparseMatrixPS(NewSparseMatrix<floatT>* A,ofstream &os);
```

wobei A die auszugebende Matrix ist und *os* ein Ausgabestream.

4.3. Implementation der Numerierungsverfahren

In diesem Abschnitt wird die *C++*–Implementation der in Abschnitt 3 (S. 31) und Abschnitt 2 (S. 19) beschriebenen Numerierungsalgorithmen erläutert. Sämtliche Code–Auszüge sind *ADR* entnommen.

4.3.1. Allgemeiner Aufbau der Implementation

Die Abbildung 4.3 gibt einen vollständigen Überblick über die implementierten Methoden zum Downwind Numbering. Die Grafik ist von links nach rechts zu lesen. Als Input steht entweder das Geschwindigkeitsfeld mit dem Gitter oder die diskretisierte Matrix zur Verfügung — je nachdem, ob geometrisches oder algebraisches Downwind Numbering durchgeführt wird.

Die Eingabedaten werden durch ein Kriterium gefiltert, um so einen der Konvektionsgraphen G_1 bis G_n zu erzeugen. Verschiedene Kriterien liefern dabei verschiedene Graphen

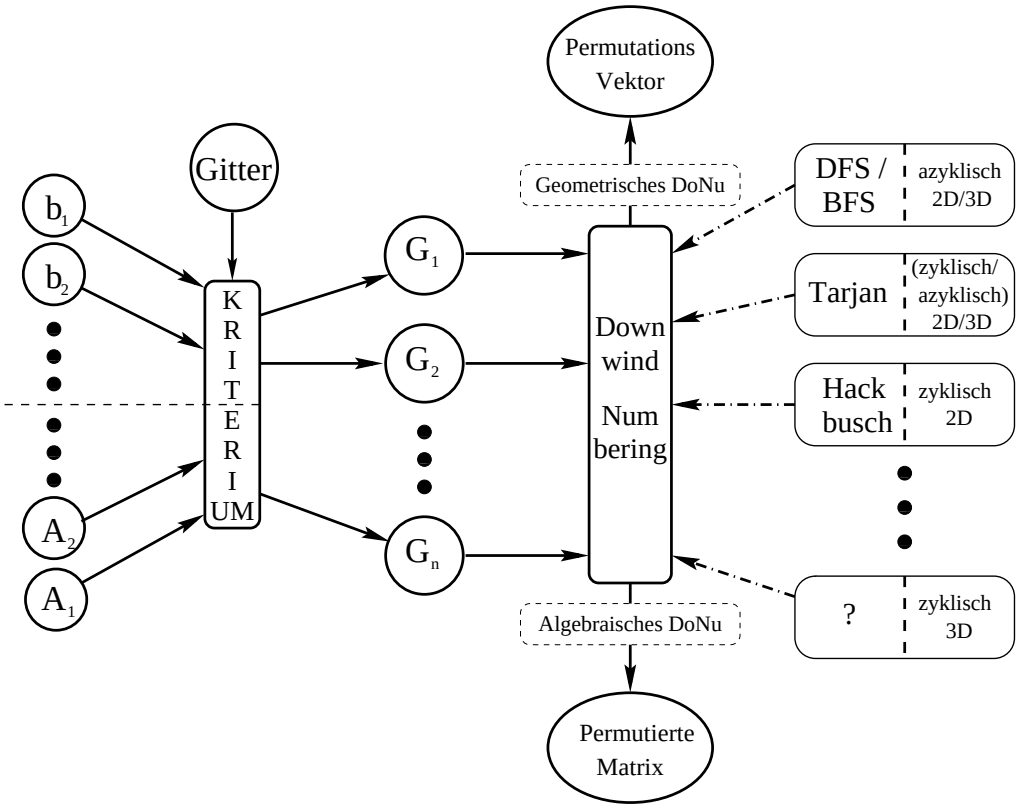


Abbildung 4.3.: Datenstruktur

als Output. Ab dieser Stelle laufen sämtliche Downwind Numbering Algorithmen absolut unabhängig vom gewählten Verfahren. Es ist nun unerheblich, ob das gewählte Kriterium algebraisch oder geometrisch ist, ob man also die Steifigkeitsmatrix permutieren möchte oder die Knotennummern des Gitters.

In Abhängigkeit von der Dimension des Verfahrens und eventuell vorhandenen Zyklen kann nun ein entsprechender Numerierungsalgorithmus zur Durchnumerierung des Konvektionsgraphen eingesetzt werden. Bisher implementiert sind die Tiefensuche (für den azyklischen 2D– und 3D–Fall), der TARJAN–Algorithmus (für beliebige 2D– und 3D–Graphen) und der Approximationsalgorithmus von HACKBUSCH für den zyklischen 2D-Fall. Nicht bekannt ist bisher ein Algorithmus für den zyklischen 3D Fall zum Auflösen der Zyklen.

Das Ergebnis des Downwind Numbering ist im geometrischen Fall ein Permutationsvektor, mit dessen Hilfe man die Knotennumern der Triangulierung permutieren kann. Im algebraischen Fall liefert das Verfahren eine permutierte Steifigkeitsmatrix, wobei nur die Nicht–Nulleinträge permutiert werden.

4.3.2. Datenstrukturen

Graph–Objekt

Die zentrale Datenstruktur in der Implementation ist das Graph–Objekt. Sowohl der TARJAN– als auch der HACKBUSCH–Algorithmus operieren auf diesem Objekt.

Im wesentlichen beinhaltet es eine Liste von Adjazenzlisten, und zwar genau eine Adjazenliste für jeden Knoten. Zusätzlich dazu die Anzahl der Knoten und Kanten im Graphen.

Jeder Graph enthält eine Liste seiner starken Zusammenhangskomponenten, die ihrerseits wiederum eigene Graph–Objekte sind.

Bei dem Konstruktor ist es möglich, eine Liste von Startknoten anzugeben. Numerierungsalgorithmen haben so ein Anfangsset für den Beginn der Numerierung zur Verfügung.

In *C++* sieht das Objekt wie folgt aus:

```
class graphC {
     //adjacency list
     typedef vector< list< base::indexT >* > AdjList;
     //number of vertices in graph
     base::indexT V;
     //number of edges in graph
     base::indexT E;
     //adjacency list
     AdjList a;
     //a list of strong components within this graph
     vector< graphC* > strc;
// ...
 public:
     //constructor
     graphC( AdjList *a,
          vector<base::indexT >* startNodes );
// ...
}
```

Der Graph stellt Iteratoren für die Knoten und Zusammenhangskomponenten zur Verfügung. Zusätzlich dazu enthält er weitere Funktionen für die Numerierungsalgorithmen und PostScript–Ausgaberoutinen.

Kriterium–Objekt

Das Kriterium–Objekt ermöglicht die Wahl unterschiedlichster Kriterien zur Erzeugung des Konvektionsgraphen. Derzeit implementiert sind ein geometrisches und ein algebraisches Kriterium:

```
class criteriaC {
// ...
 public:
```

```
    //geometrisches Kriterium
    graph::graphC* getMaxConvectionPerNodeGraph(
                   net::hierarchC* hierarchy,
                   adr::AdrC* adr );
    //algebraisches Kriterium
    graph::graphC* getMaxValuePerRow(NewSparseMatrix<base::floatT>* A);
}
```

Weitere Objekte

Weitere implementierte Objekte sind das Downwind–Numbering–Objekt `donuC` zum Aufruf der verschiedenen Numerierungsalgorithmen. Das TARJAN–Objekt `tarjanC` für den TARJAN–Algorithmus, sowie das HACKBUSCH–Objekt `hackbuschC` für den HACKBUSCH–Algorithmus.

4.3.3. Generierung des Konvektionsgraphen

Beim geometrischen Downwind Numbering wird der Konvektionsgraph aus dem Gitter und dem Geschwindkeitsfeld $\vec{b}$ erzeugt. Der Nachteil hierbei ist, daß für das Auswahlkriterium bereits vor der Berechnung der Einträge in der Steifigkeitsmatrix die Konvektionsanteile in irgendeiner Form berechnet werden müssen, um eine Auswahl der Kanten der Triangulierung für den Konvektionsgraphen treffen zu können. Die Kanten der Triangulierung entsprechen ja gerade den Einträgen in der Steifigkeitsmatrix, siehe Abschnitt 1.5 (S. 12). Ein Vorteil des Verfahrens ist, daß die Steifigkeitsmatrix gleich mit der richtigen Permutation generiert wird, was ein aufwendiges Umsortieren der Matrix erspart.

Wendet man das Kriterium E_6 aus Abschnitt 3.2 (S. 32) an, so ergibt sich folgende Schleife:

```
// ...
vector<indexT> neighbourNodes;
floatT minAngle=360.0,angle=0.0;
indexT maxConvNode=indexTmax;

neighbourNodes = patchNodes( this->hierarchy->getNod(nodeIdx),
                             this->hierarchy->curNet() );

for(vector<indexT>::const_iterator nn = neighbourNodes.begin();
  nn != neighbourNodes.end();
  nn++)
{
  angle = scalarProduct( &(this->hierarchy->getNod(nodeIdx)),
                         // current node
                         &(this->hierarchy->getNod(*nn)),
                         // neighbour node
                         &(hierarchy->getNod(nodeIdx) + *bNode) );
                         // velocity vector
```

```
    if ( angle < minAngle )
    {
      minAngle = angle;
      maxConvNode = *nn;
    }
  }
  // ...
```

Dabei liefert die Funktion *patchNodes* des Gitters gerade die Nachbarknoten des betrachteten Knotens im Gitter. In der Schleife wird dann für jeden benachbarten Knoten das Skalarprodukt $\langle b(u), (u, w)\rangle$ berechnet. Das Minimum sämtlicher Skalarprodukte erhält dabei den Zuschlag, weil hier der Winkel $\angle\{b\,(u)\,;(u, w_{min})\}$ am kleinsten ist. Das heißt, daß der Konvektionsvektor liegt am dichtesten an dieser Kante der Triangulierung.

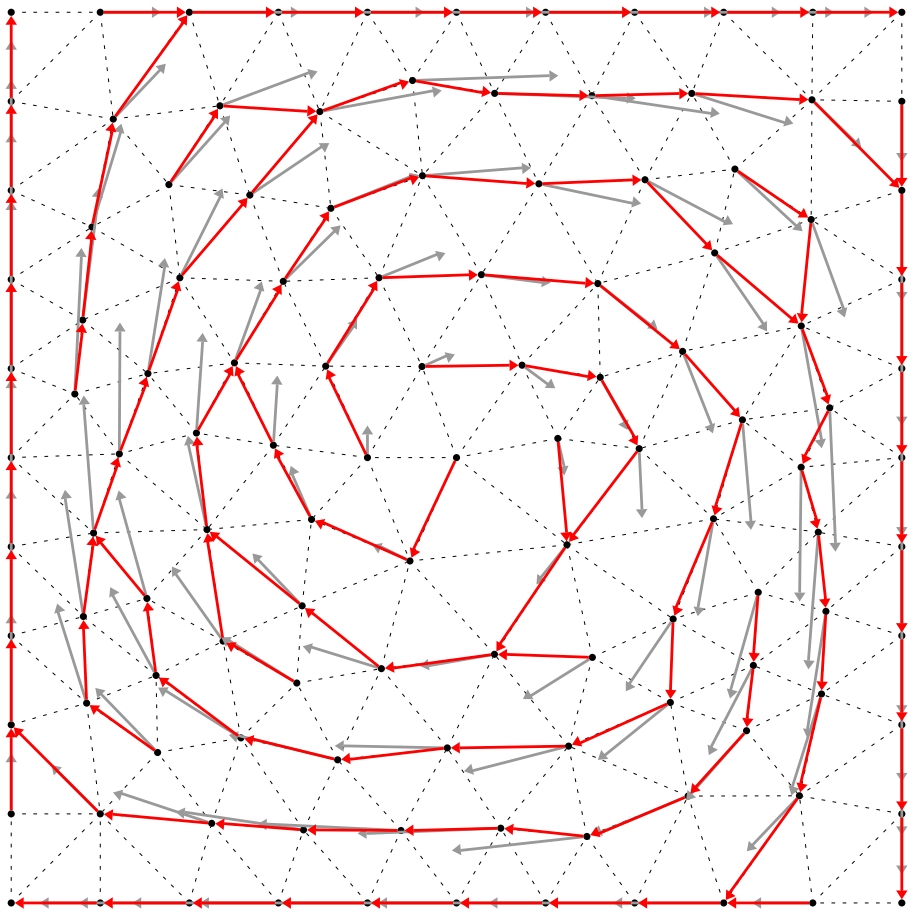


Abbildung 4.4.: Kriterium E_6 für Rotationsströmung auf unstrukt. 17×17 Gitter

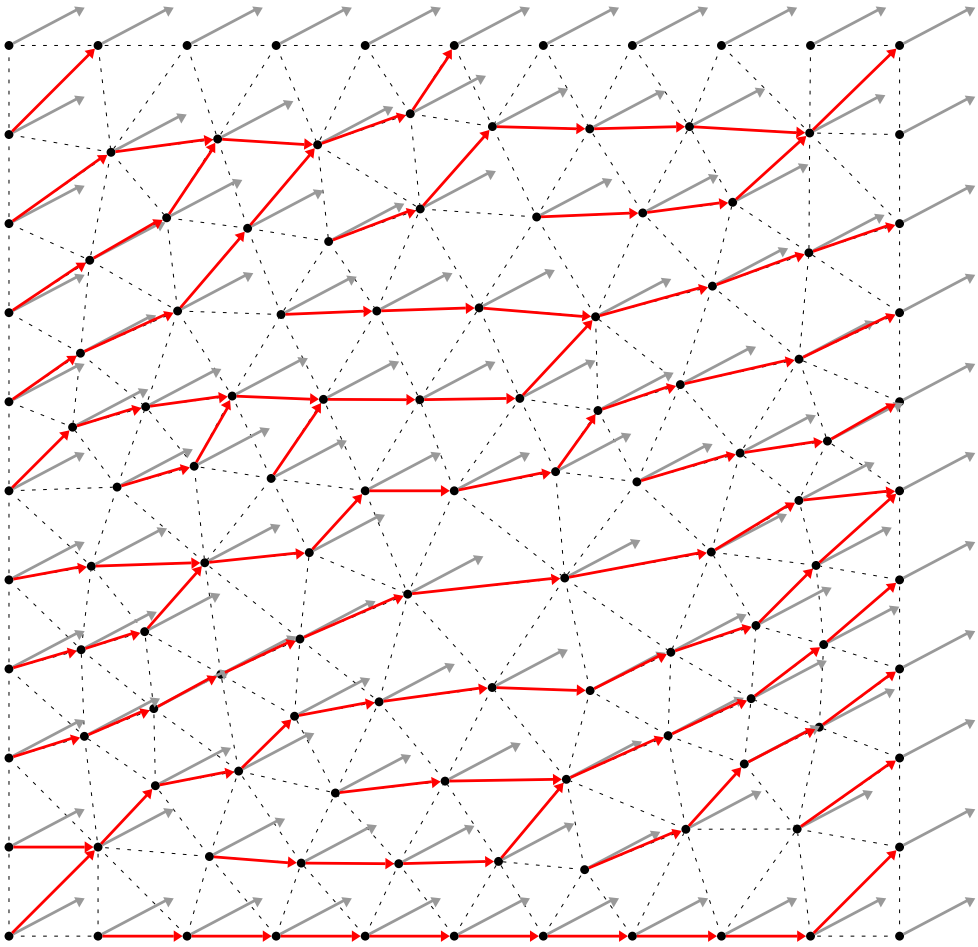


Abbildung 4.5.: Kriterium E_6 für Schrägströmung auf unstrukt. 17×17 Gitter

Bemerkung 4.1 (zu *PNS*) Die Implementation dieser Verfahren in *PNS* wurde unter anderem aus dem Grund nicht weiter verfolgt, daß eine solche Funktion *patchNodes* bei *PNS* nicht zu Verfügung steht. Siehe Abschnitt 4.1 (S. 57).

Durch den modularen Aufbau der Implementation ist es ohne weiteres möglich, beliebige weitere Auswahlkriterien zu implementieren.

Die Abbildungen 4.4 und 4.5 zeigen den entstandenen Konvektionsgraphen (rote Kanten) auf einem unstrukturiertem 17×17 Gitter (gestrichelte Kanten) bei Kreis- und Schrägströmung (graue Kanten).

Bemerkung 4.2 (Konvektionsgraph und Strömungsverlauf) An den beiden Abbildungen 4.4 und 4.5 erkennt man sehr gut, daß durch das gewählte geometrische Kriterium der Strömungsverlauf optimal verfolgt wird. Unter anderen sieht man dies bei der Rotationsströmung, wo durch den leichten Außendrall des Stromungsfeldes in der Re-

gel keine geschlossenen Zyklen entstehen. Vielmehr nimmt der Konvektionsgraph eine nach außen gerichtete Spiralstruktur an.

4.3.4. TARJAN–Algorithmus

Wie im Abschnitt 2.3.2 (S. 27) beschrieben, ist der TARJAN–Algorithmus der Tiefensuche sehr ähnlich. Dabei beobachtet man den angenehmen Effekt, daß der Algorithmus im azyklischen Fall gerade der Tiefensuche entspricht.

Dadurch entfällt eine vorherige Überprüfung des Graphen auf eventuell enthaltene Zyklen. Die Vorgehensweise ist die, daß man in jedem Fall das TARJAN–Verfahren startet. Nach Abarbeitung des Graphen hat man dann nur zu überprüfen, ob die Anzahl der gefundenen starken Zusammmenhangskomponenten gleich oder kleiner der Knotenanzahl im Graphen ist. Im ersten Fall ist der Graph azyklisch, da hier die Knoten im Graphen gerade den starken Zusammenhangskomponenten entsprechen. Siehe Lemma 2.2 (S. 22). Ansonsten befinden sich im betrachteten Graphen tatsächlich Zyklen, die gerade den starken Zusammenhangskomponenten entsprechen, siehe Abschnitt 2.1 (S. 19). In diesem Fall kann man mit dem Verfahren von BEY fortfahren, d.h. die Steifigkeitsmatrix so zu einer Blockmatrix permutieren, daß die gefundenen Diagonalblöcke gerade starken Zusammenhangskomponenten entsprechen.

Eine andere Möglichkeit besteht darin, daß man für jede Zusammenshangskomponente den HACKBUSCH–Algorithmus startet, um die Zyklen in jeder Komponente aufzuschneiden.

In jedem Fall ist der TARJAN–Algorithmus zentraler Bestandteil der vorgestellten Verfahren.

Für jeden Knoten wird zunächst die folgende Struktur angelegt:

```
struct TJNode {
  /** numbering of vertices used in tarjan algo */
  int nr;
  /** lower numbers of vertices used in tarjan algo */
  int low_nr;
  /** number of components */
  int comp_nr;
  /** should be obvious */
  bool is_on_stack;
};
```

Nach entsprechender Initialisierung mit 0 bzw. *false* startet die Hauptschleife des Algorithmus

```
while(!mainQueue->empty())
{
  if (tjnd[mainQueue->front()].nr==0)
    StrongComp( graph, mainQueue->front(), tMap );
  mainQueue->pop();
}
```

— wobei nach Abarbeitung des Graphen die neue Numerierung in der Map *tMap* gespeichert wird. Die gefundenen Zusammenhangskomponenten werden in dem Graph–Objekt in einer eigenen Liste gespeichert.

4.3.5. HACKBUSCH–Algorithmus

Die Grafik 4.6 gibt einen Überblick über die einzelnen Schritte des HACKBUSCH–Algorithmus. Das Ziel ist es, solange Kanten und Knoten aus dem Graphen zu entfernen, bis mittels einer Bewertungsfunktion entschieden werden kann, welche Knoten in das Feedback–Vertex–Set aufgenommen werden sollen.

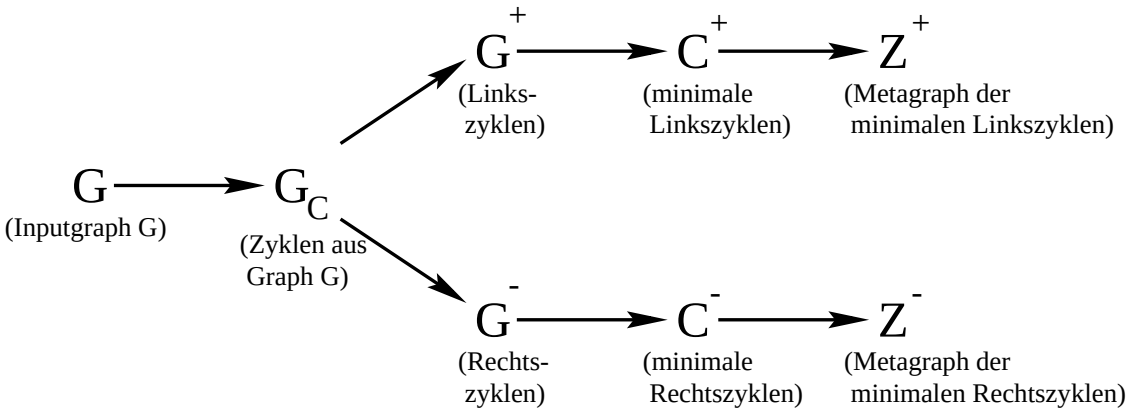


Abbildung 4.6.: Überblick über das Hackbuschverfahren

Zunächst wird der Ausgangsgraph mittels des TARJAN–Algorithmus in seine starken Zusammenhangskomponenten zerlegt. Das heißt, hierbei wird nicht der von HACKBUSCH vorgeschlagene Algorithmus `Filter` aus Abschnitt 3.5.3 (S. sec:hackbusch) verwendet, da er die gleiche Funktionsweise hat. Deshalb wurde auf eine Implementation verzichtet. Hieran erkennt man auch die Möglichkeiten eines modularisierten Codes im Hinblick auf Wiederverwendbarkeit und Flexibilität. Man erhält durch den TARJAN–Algorithmus den Graph G_C.
Nun werden die linkeste und rechteste ausströmende Kante durch die Funktion

```
void hackbuschC::markLeftRightOutEdges(
                    graphC *graph,
                    pointPoolC *pp );
```

markiert. Das bedeutet, sie werden in einer Liste abgespeichert. Die Abbildung 4.7 zeigt eine Graphenkomponente mit entsprechend markierten Kanten als Output der Funktion `markLeftRightOutEdges`. Dabei ist die linkeste Kante *rot* markiert und rechteste Kante *schwarz*. Handelt es sich um dieselbe Kante ist sie *rot–schwarz* markiert.

Damit erhält man die Graphen G^+ und G^-, auf den nun Zyklen gesucht werden müssen. Dies geschieht durch die nachfolgende Funktion:

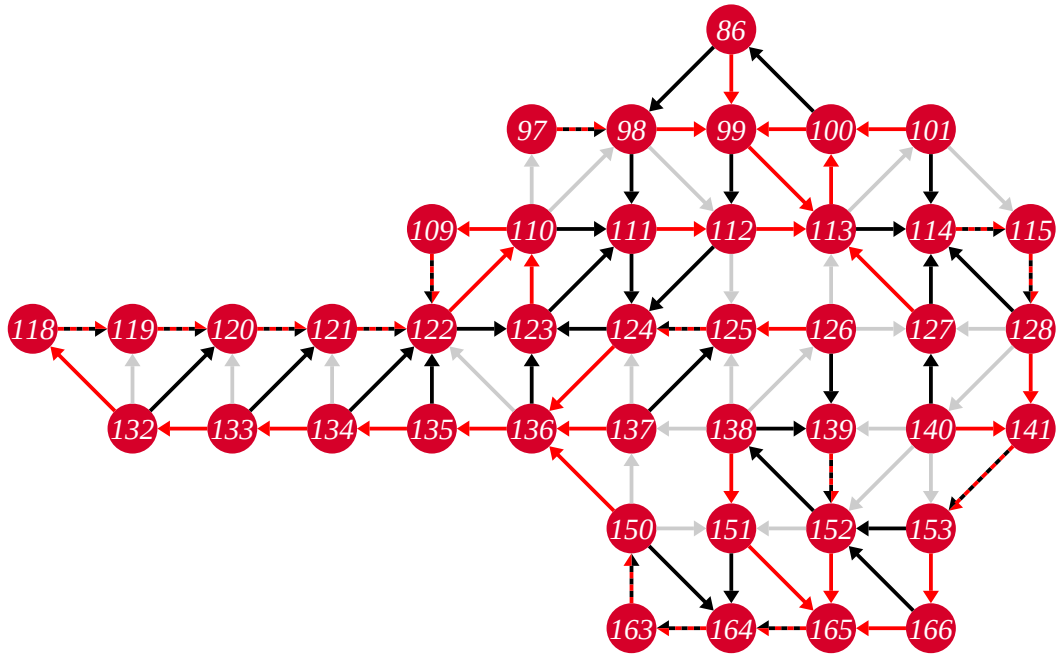


Abbildung 4.7.: Links– und Rechtskanten

```
void hackbuschC::findCycles( graphC *graph );
```

Dabei werden in den beiden Graphen sämtliche Kanten abgelaufen und die Anfangs- und Endknoten mit dem gleichen Stempel markiert. Stößt man dabei wieder auf diesen Stempel, ist ein Zyklus gefunden und der Stempel wird um Eins erhöht. Man fährt fort mit der Suche auf einem noch nicht besuchten Knoten. Die gefundenen Zyklen werden als Knotenliste in einer weiteren Liste gespeichert. Damit erhält man die Graphen C^- und C^+.

Jetzt muß für jeden Zyklus seine Orientierung ermittelt werden. Dabei läuft man von einem Startknoten v_0 des Zyklus aus los und addiert die Winkel

$$\angle\{(v_0, v_{i-1}) \ , \ (v_0, v_i)\} \, ,$$

wobei v_i der Nachfolger von v_{i-1} ist in Durchlaufrichtung. Ist die Summe negativ, so wurde der Kreis im mathematisch positiven Sinn durchlaufen. Zusatzlich dazu hat man noch die Information, ob es sich um einen Zklus der Links- oder der Rechtskanten handelt. Damit läßt sich nun die Bewertungsfunktion ermitteln. Diese funktioniert ebenfalls über ein solches Winkelargument. Sei v der zu untersuchende Knoten und ζ ein Zyklus im Graph, dann gilt es zu ermitteln, ob v geometrisch innerhalb oder außerhalb von ζ liegt. Man addiert nun einfach die Winkel

$$\angle\{(v, v_{i-1}) \ , \ (v, v_i)\} \, ,$$

indem man über die Knoten $v_i \in \zeta$ läuft. Ist die Summe 360° so liegt der Knoten im Innern, sonst außerhalb. Damit hat die geometrische Lage sämtlicher Knoten bzgl. aller Zyklen, womit sich nun die Bewertungsfunktion gemäß Abschnitt 3.5.3 (S. sec:hackbusch) ermitteln läßt. Im *ADR* –Code geschieht dies durch folgende Funktion:

```
void hackbuschC::weightVertices(
                    graphC *graph,
                    pointPoolC* pp );
```

Bemerkung 4.3 Zumindest die Anfangsinvestition zur Berechnung der Bewertungsfunktion hat quadratischen Aufwand.

Insgesamt ergibt sich damit der HACKBUSCH–Algorithmus folgendermaßen in der Implementierung:

```
// initialization
// ...
     hackbuschC::markLeftRightOutEdges( graph, pp );
     hackbuschC::findCycles( graph );
     graph->initCycleOrientation( pp );
     graph->initInsideOutside( pp );
     graph->printCycleOrientation( pp );

     indexT i=0;
     indexT fvsVertex;
     floatT weight=1;
     while( weight != 0 )
     {
          weight=hackbuschC::weightVertices( graph, pp );
          fvsVertex=graph->getNM( graph->getMinWeightVertex() );
          hackbuschC::addFVSVertex( fvsVertex );

          graph->deleteVertexCircle( fvsVertex );
     }
// ...
```

Das Abbruchkriterium ist dann erreicht, wenn die Bewertungfunktion 0 ist, d.h. `weight=0`. Die Indizes der Knoten des Feedback–Vertex–Set werden in einer Liste gespeichert, die das HACKBUSCH–Objekt nach Abarbeitung des Graphen zurückliefert.

Die Abbildung 4.8 zeigt das Ergebnis eines Durchlaufs dieses Algorithmus. Die gefundenen Feedback–Vertices sind grün eingezeichnet. Die vom TARJAN–Algorithmus bestimmten Zusammenhangskomponenten sind unterschiedlich farbig eingezeichnet, und fortlaufend durchnummeriert (kleine Nummer).

Die Abbildung 4.8 einen beliebig erzeugten Graphen, der die one–flow–condition erfüllt, nach einem Durchlauf des Algorithmus.

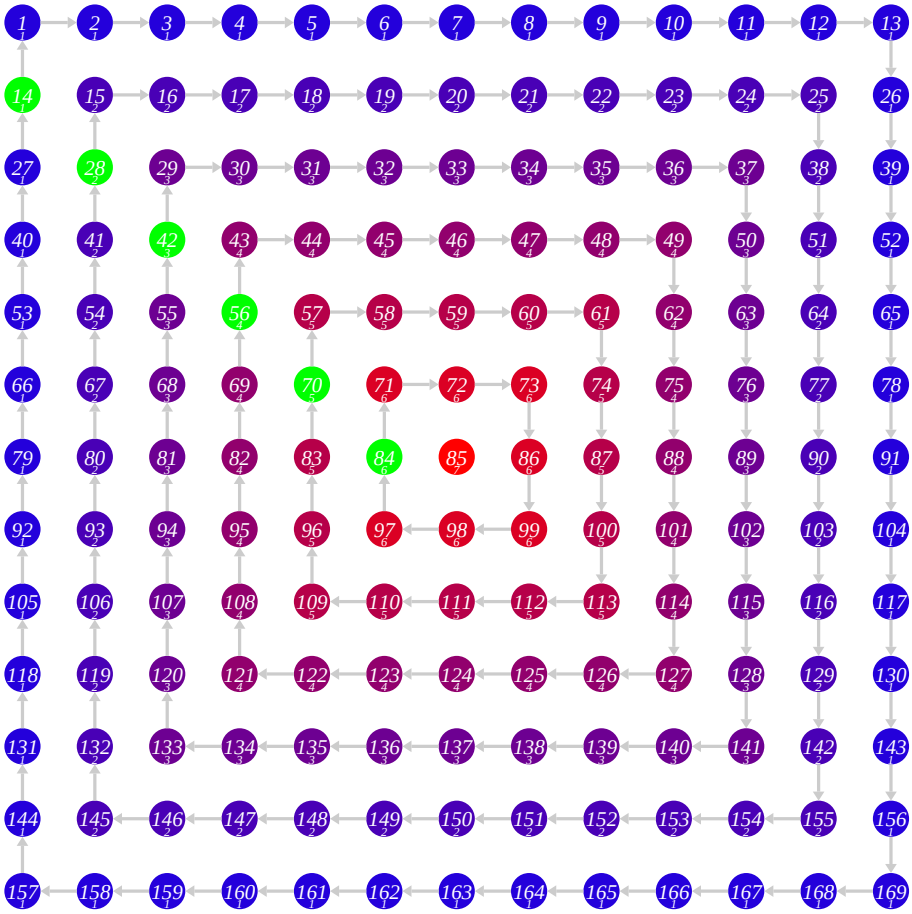


Abbildung 4.8.: FVS bei konzentrischen Kreisen

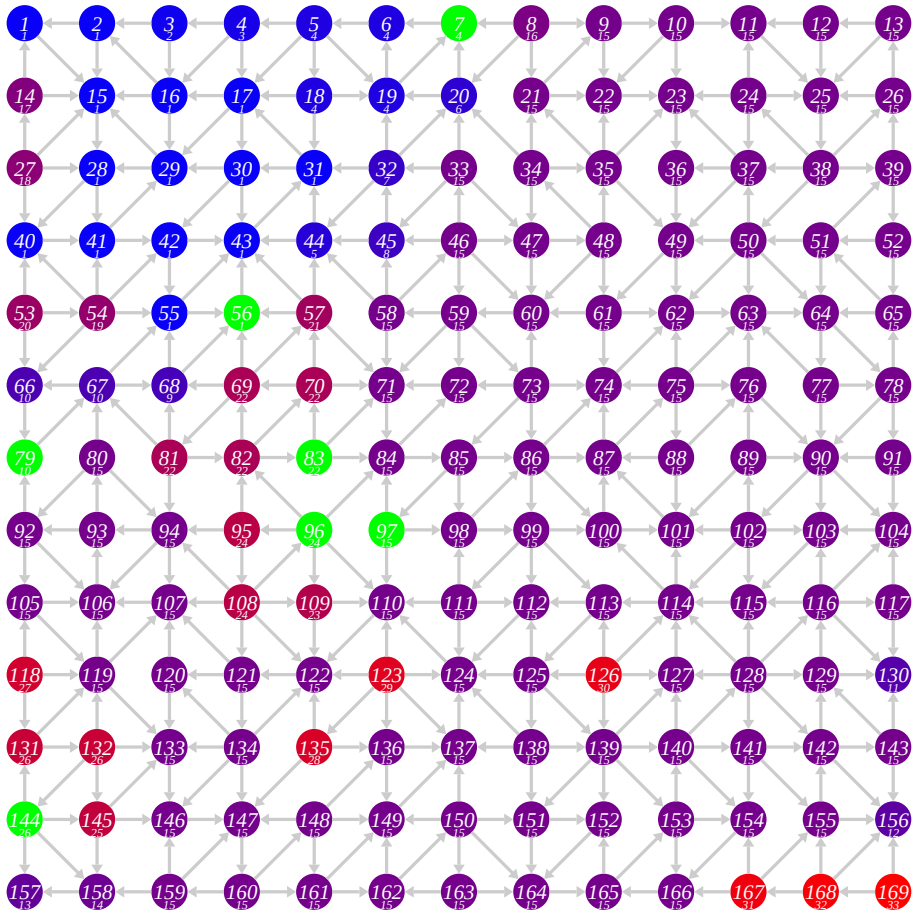


Abbildung 4.9.: FVS bei beliebigem Graph mit one–flow–condition

5. Numerische Experimente zum Downwind Numbering

In diesem Kapitel werden die numerischen Ergebnisse der beschriebenen Verfahren dargestellt und diskutiert. Begonnen wird mit der Beschreibung der gerechneten Testprobleme und der Durchführung der Experimente, damit Aussagen über die Laufzeiten der implementierten Algorithmen und Kriterien gemacht werden können.

5.1. Beschreibung der Testprobleme

Für die numerischen Tests der vorgestellten Algorithmen wurde je ein Problem für Schrägströmung und Rotationsströmung gerechnet. Diese beiden Strömungen sind Spezialfälle eines zweidimensionalen stationären Konvektions-Diffusions-Problems der Form

$$-\nabla \cdot (\varepsilon \nabla u) + (\boldsymbol{b} \cdot \nabla) u = f \text{ in } \Omega.$$

Alle Testbeispiele wurden auf dem Einheitsquadrat gerechnet. In diesem Kapitel soll also stets $\Omega = (0,1) \times (0,1)$ mit dem Rand $\partial\Omega = \Gamma$ gelten. Alle Probleme wurden mit $c = 0$ gerechnet.

5.1.1. Schrägströmung

Das Geschwindigkeitsfeld b sei mit $b = \frac{1}{\sqrt{5}}(2,1)^T$ konstant gewählt (damit ist $\|b\|_2 = 1$). Die rechte Seite sei $f \equiv 0$. Der Gebietsrand $\partial\Omega$ läßt sich somit aufspalten in

$$\begin{aligned}
\partial\Omega^+ &= \{(\, x\,,\, 1\,)^T, (\, 1\,,\, y\,)^T \,:\, 0 \le x, y \le 1\}, \\
\partial\Omega^0 &= \emptyset, \\
\partial\Omega^- &= \{(\, x\,,\, 0\,)^T, (\, 0\,,\, y\,)^T \,:\, 0 \le x, y \le 1\}.
\end{aligned}$$

Dabei sei mit $\partial\Omega^+$ der Einströmrand und mit $\partial\Omega^-$ der Ausströmrand bezeichnet. Weiter sei $\partial\Omega^0$ der Teil des Randes, bei dem die Strömung parallel dazu verläuft.
Auf dem Rand seien folgende Dirichlet–Bedingungen gegeben:

$$\begin{aligned}
u = 1 \quad &\text{auf} \quad \left\{(\, x\,,\, 0\,)^T,\ (\, 0\,,\, y\,)^T \,:\, 0 \le x \le 1,\ 0 \le y \le \frac{1}{3}\right\}, \\
u = 0 \quad &\text{sonst.}
\end{aligned}$$

Das Strömungsfeld $\vec{b}$ und die Randwerte werden in der Abbildung 5.1 dargestellt.
Das Problem besitzt eine innere parabolische Grenzschicht, und zwar entlang der Charakteristik ζ_{x_1} mit $x_1 = (0, \frac{1}{3})^T$. Diese ist bedingt durch die Unstetigkeit an x_1 in den Dirichlet–Randbedingungen am Einströmrand. Eine zweite Unstetigkeitsstelle befindet sich am Ausströmrand in $x_2 = (1, \frac{2}{3})^T$, die so gewählt wurde, daß sie dem Austrittspunkt der Charakteristik ζ_{x_1} aus dem Gebiet entspricht.

Im konvektionsdominanten Fall oszilliert die Lösung der Galerkin–FEM global sehr stark. Aus diesem Grund müssen noch Stabilisierungsterme hinzugefügt werden, siehe Abschnitt 1.6 (S. 13). Die in den Abbildungen 5.2 bis 5.10 dargestellten Lösungen wurden mit SUPG–Stabilisierung gerechnet, siehe Abschnitt 1.6.2 (S. 16). Zusätzlich dazu wurde noch eine Shock–Capturing–Technik verwendet, um lokale Oszillationen an Grenzschichten zu reduzieren, siehe Abschnitt 1.6.3 (S. 17).

Da im benutzten FEM–Code *ADR* Randbedingungen segmentweise vergeben werden, liegen beim betrachteten Beispiel echte Unstetigkeitsstellen am Ein- und Ausströmrand vor, die die beobachteten verbliebenen Oszillationen erzeugen. Im Vergleich dazu werden im FEM–Code *PNS* Randdaten knotenweise vergeben, weshalb in den Arbeiten von T. Knopp [Kno99] und A. P. Priesnitz [Pri96] geringere Überschwinger bei diesem Testbeispiel auftreten.

Die auftretenden Überschwinger können allerdings vernachlässigt werden, da Dirichlet–Randbedingungen auf $\partial\Omega^+$ untypisch sind in praktisch relevanten physikalischen Fragestellungen, hier würde man eher homogene Neumann-Daten (do–nothing–Bedingung) am Ausströmrand setzen.

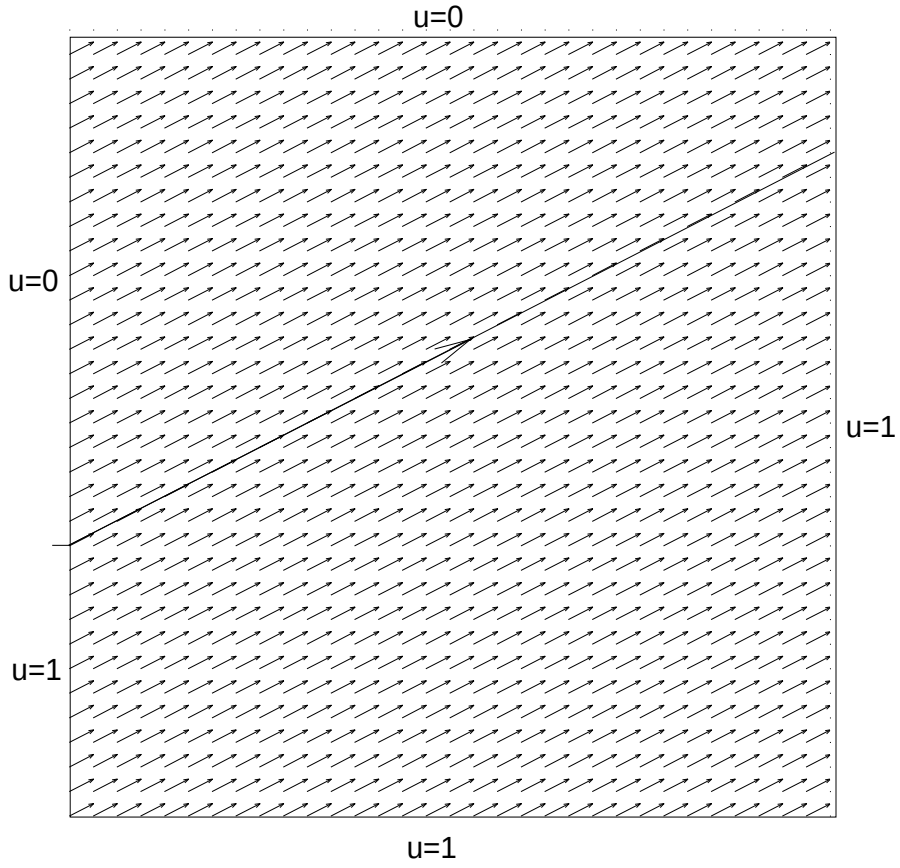


Abbildung 5.1.: Schrägströmung

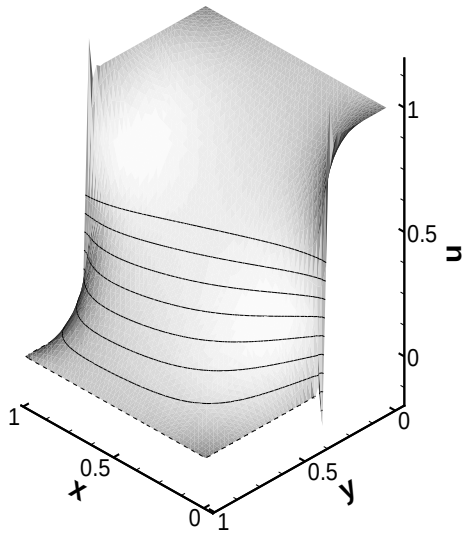


Abbildung 5.2.: Schräg, $\varepsilon = 1, c = 0$

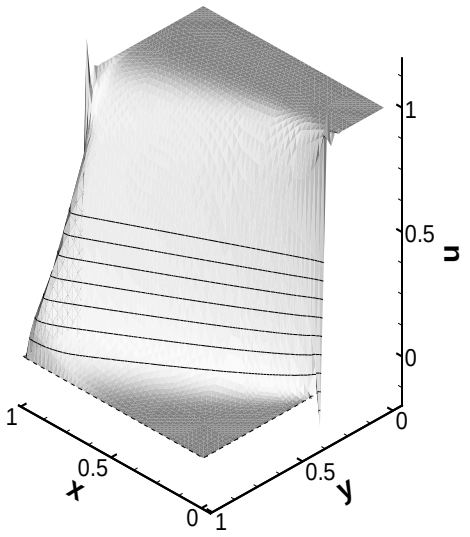


Abbildung 5.3.: Schräg, $\varepsilon = 10^{-2}, c = 0$

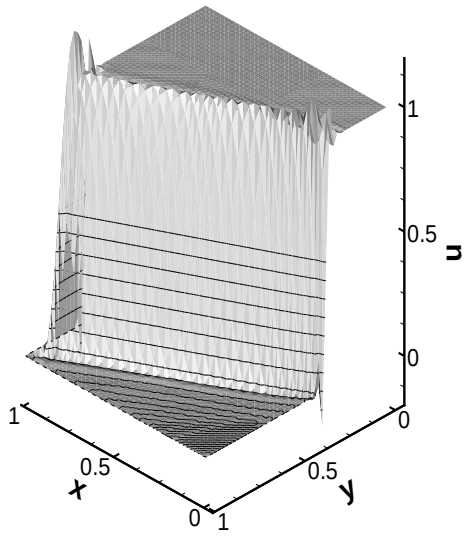


Abbildung 5.4.: Schräg, $\varepsilon = 10^{-4}, c = 0$

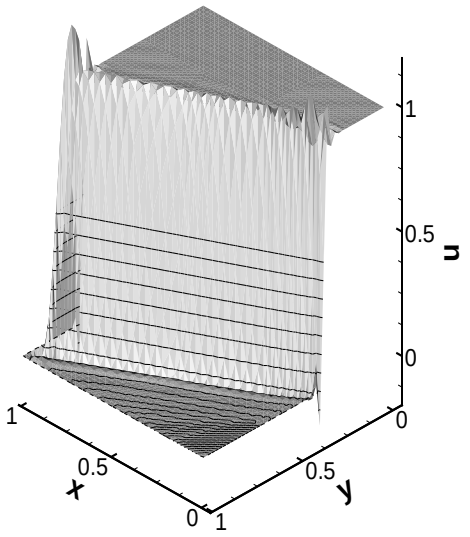


Abbildung 5.5.: Schräg, $\varepsilon = 10^{-6}, c = 0$

5.1.2. Rotationsströmung

In diesem Beispiel sei das Geschwindigkeitsfeld $\vec{b}$ bestimmt durch

$$\vec{b}(x,y) := \begin{pmatrix} (2y-1)(1-(2x-1)^2) \\ 4y(2x-1)(y-1) \end{pmatrix}.$$

Dadurch ist eine Rotations- oder Rotationsströmung um den Mittelpunkt von Ω gegeben, die am Rand $\partial\Omega$ parallel zu diesem verläuft. Damit ist $\partial\Omega = \partial\Omega^0$, und alle Charakteristiken sind geschlossen.
Auf dem Rand seien folgende Dirichlet–Bedingungen gegeben:

$$\begin{aligned} u &= -0.5 \quad \text{auf} \quad \left\{ (\,0 \quad y\,)^T \,:\, 0 \le y \le 1 \right\}, \\ u &= \quad 0.5 \quad \text{auf} \quad \left\{ (\,1 \quad y\,)^T \,:\, 0 \le y \le 1 \right\}, \\ u &= \quad 0 \quad \text{sonst.} \end{aligned}$$

Abbildung 5.6 illustriert das Strömungsfeld $\vec{b}$ und die Randwerte. Sei wieder $f \equiv 0$ und $c = 0$.
Es sei erwähnt, daß $c = 0$ in diesem Testbeispiel im Gegensatz zur Schrägströmung ein kritischer Fall ist. Hier muß für $\varepsilon \to 0$ die richtige globale Konstante u_C, d.h.

$$\lim_{\varepsilon \to 0} u_\varepsilon(x) = u_C \quad \forall x \in G,\ \forall G \subset\subset \Omega$$

gefunden werden. Mit analytischen Überlegungen kann man zeigen, daß

$$u_C = \frac{\int_{\partial\Omega} g\, ds}{\int_{\partial\Omega} ds} = 0$$

gilt. Dies stellt ein schwieriges Problem für den diskreten Löser dar, was sich auch an den ermittelten Ergebnissen zeigt.

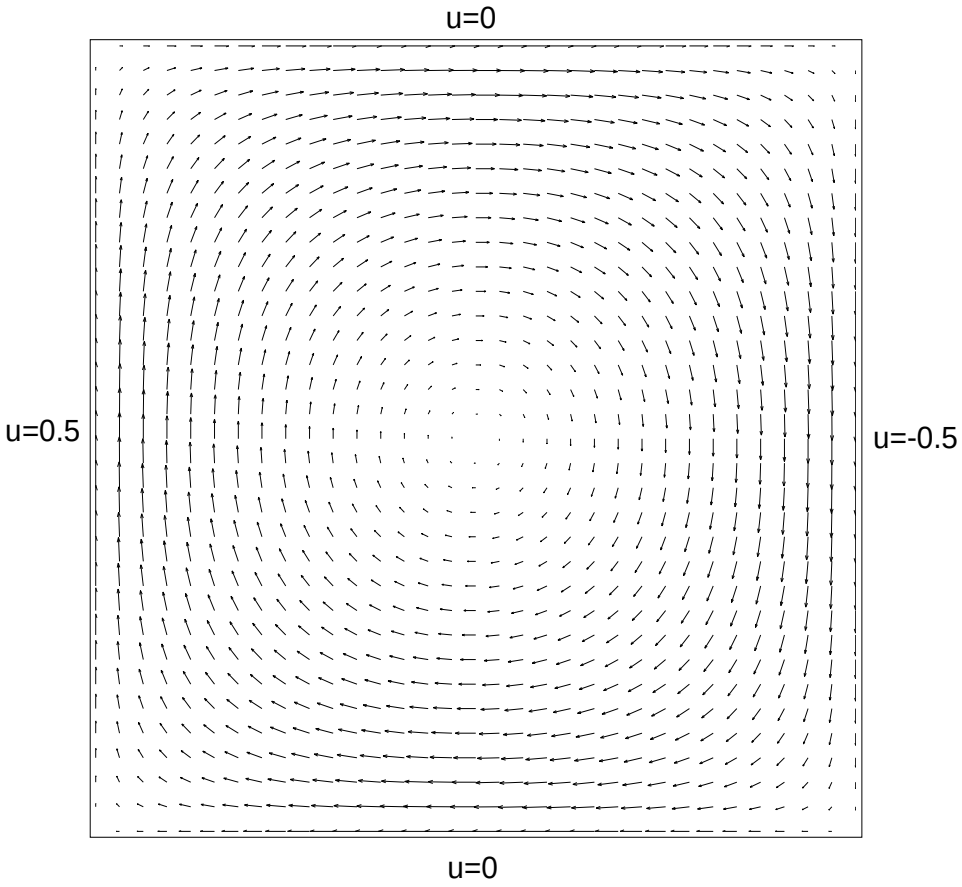


Abbildung 5.6.: Rotationsströmung

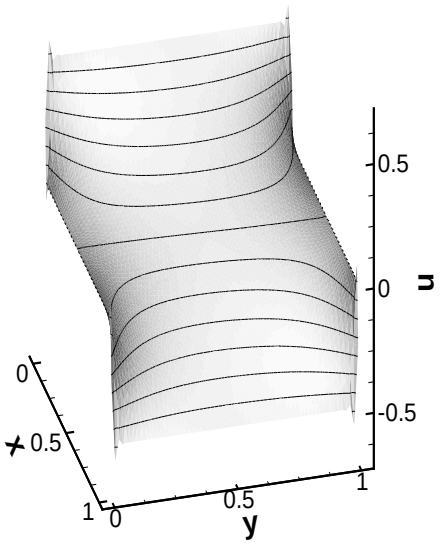


Abbildung 5.7.: Kreis, $\varepsilon = 1, c = 0$

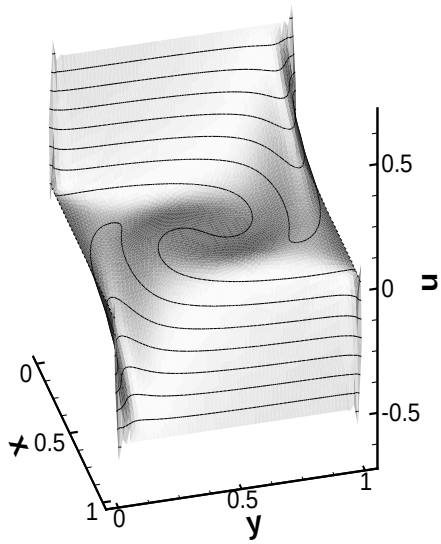


Abbildung 5.8.: Kreis, $\varepsilon = 10^{-2}, c = 0$

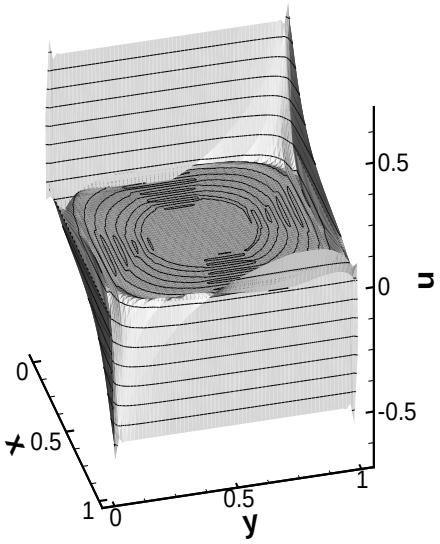


Abbildung 5.9.: Kreis, $\varepsilon = 10^{-4}, c = 0$

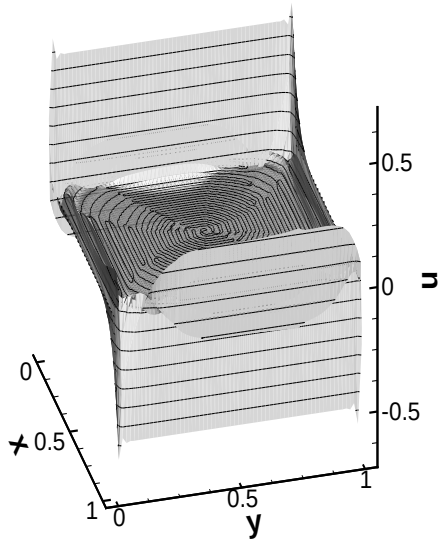


Abbildung 5.10.: Kreis, $\varepsilon = 10^{-6}, c = 0$

$h^{-1}+1$	strukturiert	unstrukturiert
33	1089	1121
65	4225	4353
129	16641	17153

Abbildung 5.11.: Anzahl der Eckpunkte in der Triangulierung

5.2. Durchführung der Versuche

Bei den numerischen Tests wurde jeweils die Rechenzeit verglichen, die das Verfahren benötigt, bis die Residuennorm $\|Au_k - f\|_2/\|Au_0 - f\|_2$ unter 10^{-10} liegt. Dabei sei k der Iterationsindex und u_0 die Startlösung. Zusätzlich dazu wird auch noch die Anzahl der benötigten Iterationen mit angegeben. In vielen Darstellungen, die Konvergenzresultate für iterative Verfahren formulieren, beschränkt sich die Angabe auf die Anzahl der Iterationen. Dies läßt jedoch keinen sinnvollen Vergleich verschiedener Verfahren zu, da sich diese in dem pro Iteration erforderlichen Rechenaufwand eventuell signifikant unterscheiden.

Um weitere Aussagen über die Qualität des Vorkonditionierers treffen zu können, wurde bei den numerischen Experimenten der zeitliche Konvergenzverlauf betrachtet, also die Entwicklung der Konvergenz $\|Au_k - f\|_2/\|Au_0 - f\|_2$ des Residuums in der Euklidischen Norm bezüglich der Rechenzeit. Dabei wurde für ausgewählte Beispiele der Konvergenzverlauf der Lösung des nicht vorkonditionierten Problems gegenüber dem vorkonditionierten dargestellt, siehe Anhang A.2 (S. 99).

Als Startlösung wurde immer $u_0 \equiv 0$ gesetzt. Numerische Tests haben gezeigt, daß sich für $-1 \leq u_0 \leq 1$ die Anzahl der benötigten Iterationen nicht signifikant unterscheidet. Dieser Sachverhalt muß allerdings für die Rotationsströmung noch intensiver als bisher untersucht werden, siehe dazu auch die Aussagen zum Reaktionskoeffizienten c in Abschnitt 5.1.2.

Die Downwind Numbering Verfahren zeigen ihre Wirkung erst im konvektionsdominanten Fall. Um diesen Effekt zu verdeutlichen, wurde bei den Tests der Diffusionskoeffizient ε über $\{10^{-2}, 10^{-4}, 10^{-6}\}$ variiert. Für den Fall $\varepsilon = 1$ ergeben sich sowohl bei Schräg- als auch bei Rotationsströmung identische Iterationszahlen für das vorkonditionierte und das nicht vorkonditionierte Problem.

Die Probleme wurden auf strukturierten und unstrukturierten 33×33, 65×65 und 129×129 Gittern gerechnet, die durch *ADR* mittels einer *Delaunay–Triangulation* generiert wurden. Die Tabelle 5.11 zeigt die Anzahl der Knotenpunkte der verschiedenen Gitter.

Rechenumgebung

Die vorgestellten Ergebnisse wurden gerechnet auf einer *Compaq XP 1000 Personal Workstation* mit 667 MHz mit folgenden Daten:

- ALPHA-Prozessor EV6, 4MB Cache, DECchip 21264, 64 bit,
- 1280 MB ECC Hauptspeicher und
- onBoard Ultra-SCSI Controller.

Gitter	Rotationsströmung	Schrägströmung
33×33	0.07	0.07
65×65	1.03	1.03
129×129	17.02	17.07

Abbildung 5.12.: Rechenzeit des TARJAN–Algorithmus in Sekunden

Das Betriebssystem ist *True 64 Unix 4.0f*, ehemals *Digital Unix*. Der *C++*–Code *ADR* wurde mit dem *GNU C++ Compiler gcc, version 2.95.2 19991024 (release)*, compiliert.

5.3. Numerische Ergebnisse zum Konvektionsgraph

5.3.1. Aufwand der Downwind Numbering Algorithmen

Als Downwind Numbering Verfahren wurde der TARJAN–Algorithmus eingesetzt, da in den gerechneten Beispielen nur im Fall der Rotationsströmung auf dem unstrukturierten 33×33 Gitter ein Zyklus auftrat, der durch die verfeinerte Triangulierung (ab 65×65 Gitter) verschwand. Außerdem ist momentan in *ADR* noch keine Schur–Komplement–Methode implementiert, so daß derzeit kein Löser für das mit dem HACKBUSCH–Algorithmus vorkonditierte Problem zur Verfügung steht.

Der TARJAN–Algorithmus benötigt die in Tabelle 5.12 angegebenen Zeiten für die verschiedenen Gitter.

In den gemessenen Zeiten bei strukturierten und unstrukturierten Gittern gibt es leichte Unterschiede, die durch die unterschiedliche Anzahl der Gitterknoten entstehen und deshalb vernachlässigt werden können. Weiter läßt sich beobachten, daß der Zeitaufwand pro Verfeinerungsschritt, das entspricht einer Vervierfachung der Anzahl der Unbekannten, um den Faktor 16 steigt. Das bedeutet quadratischer Aufwand $O(n^2)$ in der Problemgröße.

Die derzeit implementierte Version des TARJAN–Algorithmus ist noch nicht optimal in der Laufzeit. Dies entsteht durch die Verwaltung der Warteschlange im Algorithmus, wodurch quadratische Laufzeit entsteht, allerdings mit kleinen Konstanten, wie die Tabelle 5.12 zeigt.

An dieser Stelle läßt sich das Verfahren noch weiter optimieren.

Bei den Tabellen mit den numerischen Ergebnissen wurde beim vorkonditionierten Fall der Zeitaufwand für den TARJAN–Algorithmus weggelassen, da er eine fixe Größe ist und das unterschiedliche Verhalten des Lösers im nicht vorkonditionierten und im vorkonditionierten Fall untersucht werden sollte.

Bemerkung 5.1 (zum HACKBUSCH–Algorithmus) Da bei den vorliegenden Problemfällen bis auf eine Ausnahme keine Zyklen auftreten, kam der HACKBUSCH–Algorithmus nicht zum Einsatz. Ihm ist ein TARJAN–Algorithmus zur Bestimmung der starken Zusammenhangskomponenten im Konvektionsgraphen vorgeschaltet. Wird dabei festgestellt, daß der Graph azyklisch ist, erfolgt kein Aufruf des HACKBUSCH–Algorithmus.

Wegen der noch fehlenden Schur–Komplement–Methode in *ADR* wurde auch in dem einen Ausnahmefall auf einen Aufruf dieses Algorithmus verzichtet.

Gitter	Rotationsströmung	Schrägströmung
33×33	0.17	0.15
65×65	3.65	3.47
129×129	61.18	61.78

Abbildung 5.13.: Rechenzeit des geometrischen Kriteriums in Sekunden

5.3.2. Aufwand der verwendeten Kriterien

Geometrisches Kriterium

Für die Erzeugung des Konvektiongraphen wurde das geometrische Kriterium E_6 verwendet, siehe Abschnitt 3.2 (S. 32). Dabei wird für jeden Knoten v der Triangulierung in einer Schleife über seine Nachbarknoten bestimmt, welcher Knoten in Richtung der größten Konvektion liegt.

Faßt man die Triangulierung als ungerichtetes Gitter auf, wobei $|V|$ die Anzahl der Gitterpunkte und $|E|$ die Anzahl der Kanten im Gitter ist, dann ist der Aufwand dieses Verfahrens linear mit $O(|V| + |E|)$. Die Tabelle 5.13 zeigt die benötigten Zeiten für die verschiedenen Gitter.

In der praktischen Realisierung zeigt sich auch hier quadratischer Aufwand, wobei dies dem Iterator über die Nachbarknoten des Gitters geschuldet ist. Eine Laufzeitoptimierung dieses Iterators würde sich auch positiv auf die Laufzeit der Berechnung des geometrischen Kriteriums auswirken.

Das geometrische Kriterium operiert ausschließlich auf der Triangulierung $\mathcal{T}_h$ und dem Geschwindigkeitsfeld $\vec{b}$. Hat man einen Konvektionsgraphen erzeugt, so lassen sich darauf die implementierten Graphenalgorithmen anwenden, um eine flußorientierte Numerierung für diese Triangulierung zu berechnen. Mit einer einmal berechneten Numerierung lassen sich beliebige Problemstellungen auf dem so umnumerierten Gitter lösen, solange sich $\vec{b}$ nicht ändert. Man hat also lediglich eine Anfangsinvestition in Form einer Berechnung des Konvektionsgraphen mittels des geometrischen Kriteriums sowie der Bestimmung der Permutation zu tätigen.

Algebraisches Kriterium

Um zu demonstrieren, daß es problemlos möglich ist, weitere Kriterien in die implementierte Datenstruktur einzufügen, wurde noch ein algebraisches Kriterium implementiert. Die ermittelten Werte sind allerdings mit Vorbehalt zu bewerten, da *ADR* noch nicht sämtliche Möglichkeiten für geometrisches Downwind Numbering bietet. Insbesondere ist eine Permutation der Steifigkeitsmatrix momentan nicht ohne weiteres möglich. Deshalb wurde zur vorhandenen Steifigkeitsmatrix $A = (a_{uv})$ eine neue Matrix erzeugt, mit entsprechend der neuen Numerierung π permutierten Werten $A' = (a_{\pi(u)\pi(v)})$. Das Erzeugen dieser neuen Matrix ist natürlich recht zeitaufwendig, weshalb die gemessenen Zeiten noch nicht aussagefähig genug sind. Hiermit sollte lediglich die Flexibilität der vorhandenen Downwind Numerbing Algorithmen demonstriert werden.

Zur Erzeugung des Konvektionsgraphen wurde das Kriterium E_4 aus Abschnitt 3.2 (S. 32)

Gitter	Rotationsströmung	Schrägströmung
33×33	0.01	0.01
65×65	0.02	0.02
129×129	0.09	0.08

Abbildung 5.14.: Rechenzeit des algebraischen Kriteriums in Sekunden

Gitter	geometrisches K.	algebraisches K.
33×33	0.15	0.01
65×65	3.47	0.02
129×129	61.78	0.08

Abbildung 5.15.: Vergleich des Zeitaufwands für das geometrische und das algebraische Kriterium

verwendet, wobei allerdings pro Zeile nur ein maximaler Wert ausgewählt wurde. Damit ergibt sich also das neue Kriterium:

$$E_4' = \{(u,v) \in V \times V \,:\, |a_{uv}| = \max_{w \in V,\, w>u} \{|a_{uw}| : |a_{uw}| > |a_{wu}|\}\}.$$

Für dieses Kriterium läuft man über sämtliche Einträge im oberen Dreiecksblock der Steifigkeitsmatrix und vergleicht mit dem entsprechenden Eintrag im unteren Dreiecksblock. Das sind zusammen $|V|^2/2$ Vergleiche, was einem Aufwand von $O(|V|^2)$ entspricht. Die Zeiten für dieses Kriterium sind in Tabelle 5.14 angegeben.

Auf eine Darstellung des Zeitaufwandes für die Permutation der Steifigkeitsmatrix wurde verzichtet, da dieser noch nicht optimal ist.

Vergleich der Kriterien

Die Zeiten für die Berechnung der beiden Kriterien seien in der Tabelle 5.15 exemplarisch für den Fall der Schrägströmung verglichen.

Der immense Zeitunterschied entsteht durch die zusätzlich anfallende Berechnung des Konvektionsanteils beim geometrischen Kriterium für die Berechnung der Kante mit maximalem Fluß, siehe Abschnitt 5.3.2.

Das geometrische Kriterium würde sich dann eignen, wenn man die Zeit zu seiner Berechnung als Startinvestition verbuchen kann, um verschiedene Problemfälle auf demselben Gitter bei gleichem Strömungsfeld zu berechnen, wie z.B. bei Mehrgitter– oder Gebietszerlegungsverfahren.

Der Vorteil des algebraischen Kriteriums besteht darin, daß der Konvektionsanteil nicht doppelt berechnet werden muß. Der Zeitgewinn dabei ist immens, wie die Tabelle 5.15 verdeutlicht. Allerdings gilt es, für jeden zu rechnenden Problemfall einen neuen Konvektionsgraphen zu erzeugen und zusätzlich dazu, für jeden Fall die diskretisierte Matrix zu permutieren.

Generell ist zu sagen, daß die algebraischen Kriterien in der Zukunft fr *ADR* wohl das Mittel der Wahl sein werden, da sich hierbei wesentlich mehr Aspekte für die Erzeugung eines Konvektionsgraphen berücksichtigen lassen, wie z.B. die Art der Stabilisierung. Allerdings müssen dazu noch die Zugriffe auf die Matrixstruktur optimiert werden, sowie eine effiziente Permutation der Matrix ermöglicht werden.

Es sei hierbei noch einmal erwähnt, daß das implementierte algebraische Kriterium lediglich zur Demonstration der Möglichkeiten in *ADR* implementiert wurde. Dieses Kriterium wurde weniger wegen der Optimalität seines Wirkungsgrades sondern vielmehr wegen der einfach zu realisierenden Implementation gewählt.

Solche Kriterien, die neben den Zugriffen auf die Einträge in der Steifigkeitsmatrix auch eine Unterscheidung etwa des Konvektions- und Diffusionsanteils in der Matrix benötigen sind etwas aufwendiger zu implementieren, da sie in die Diskretisierungsschleife eingefügt werden müssen. Allerdings bleibt auch dieser Aufwand überschaubar.

Die von J. Bey [Bey98] und S. Le Borne [Bor99] vorgestellten Auswahlkriterien E_1 bis E_5 beziehen sich sämtlich auf eine durch Galerkin–FEM zusammen mit Upwind–Stabilisierung hervorgegangene Steifigkeitsmatrix, die M–Matrix–Eigenschaften besitzt, siehe Abschnitt 1.6.1 (S. 14). In *ADR* werden die verbesserten SUPG– und Shock–Capturing–Stabilisierungen verwendet, die stabilere Lösbarkeit des Problems sichern. Die entstehende Matrix besitzt hierbei approximativ die M–Matrix–Eigenschaft. Es bleibt zu untersuchen, inwieweit die erwähnten Kriterien auf diese Form der Stabilisierung anwendbar sind, oder ob neue angepaßte Kriterien gefunden werden müssen.

5.4. Numerische Ergebnisse zum Downwind Numbering

Dieser Abschnitt erläutert die Ergebnisse der Rechnungen zum Downwind Numbering vorstellen. Zunächst werden Konvergenzzeiten und –verläufe diskutiert. Dadurch erhält man einen Überblick über die Wirkungsweise des Vorkonditionierers auf die Struktur der Matrix, und abschließend werden Sequenzen von Zwischenresultaten bei Lösung vorkonditionierter Probleme gezeigt. Die Notationen der Grafiken und Tabellen im Anhang werden jeweils in den entsprechenden Unterabschnitten erläutert.

5.4.1. Konvergenzzeiten und –verläufe

In diesem Abschnitt sind die numerischen Ergebnisse der Rechnungen zusammenfassend dargestellt. Eine ausführliche Auflistung der erreichten Konvergenzzeiten und Iterationszahlen findet sich im Anhang A.1 (S. 93).

Es wird zwischen Schräg- und Rotationsströmung unterschieden. Der Diffusionskoeffizient ε läuft jeweils über $\{10^{-2}, 10^{-4}, 10^{-6}\}$. Die verschiedenen Gitter sind 33×33, 65×65 und 129×129, jeweils strukturiert (s) und unstrukturiert (u).

Für jeden gerechneten Problemfall wurden das nicht vorkonditionierte, das mit geometrischem Downwind Numbering und das mit algebraischem Downwind Numbering vorkonditionierte Problem betrachtet. Als Löser wurde das SOR–Verfahren eingesetzt. Der Relaxationsparamter ω wurde in 0.1–Schritten zwischen 0.6 und 1.8 variiert, wobei in den Tabellen immer gerade das Intervall dargestellt ist, in dem der optimale Paramter liegt. Da dieser leicht variiert von Gittergröße zu Gittergröße, aber für alle Gittergrößen der opti-

male Wert in einer Tabelle dargestellt wird, kann es vorkommen, daß das Optimum gerade am Rand des dargestellten Intervalls liegt. In jedem Fall sind die optimalen Werte fett gezeichnet. Die Verwendung von Downwind Numbering beeinflußt den optimalen Wert für den Relaxationsparamter nicht wesentlich. Abweichungen liegen im Bereich ± 0.1. Weitere Untersuchungen zum Relaxationsparameter finden sich bei A. P. Priesnitz [Pri96].

Die Einträge in den Tabellen sind: Anzahl der Iterationen / benötigte Zeit in Sekunden.

Die Rechnungen wurden jeweils nach 15000 Iterationen abgebrochen, falls das Abbruchkriterium $\|Au_k - f\|_2 / \|Au_0 - f\|_2 < 10^{-10}$ nicht vorher erreicht wurde. Diese Fälle sind mit $-/-$ gekennzeichnet.

In den Konvergenzdiagrammen soll folgende Nomenklatur gelten:

- ohne Downwind Numbering : durchgehende Linie,
- geometrisches Downwind Numbering : grob gepunktete Linie,
- algebraisches Downwind Numbering : fein gepunktete Linie.

Dabei ist auf der y-Achse die relative L_2-Norm des Residuums abgetragen mit logarithmischer Skalierung von 10^{-10} bis 100. Auf der der x-Achse ist die Iterationszeit abgetragen.

Zunächst beobachtet man, wie erwartet, daß die Downwind Numbering Vorkonditionerer mit kleiner werdendem ε größere Wirkung zeigen, da der Konvektionsfluß das Problem zunehmend bestimmt und die daraus abgeleitete Numerierung das Lösungsverhalten entsprechend stärker beeinflußt.

Bei der Schrägströmung steigt zum konvektionsdominanten Fall hin die Verbesserung der Lösungszeit von etwa 20% ($\varepsilon = 10^{-2}$) auf fast 50% ($\varepsilon = 10^{-6}$).

Beim komplizierteren Fall der Rotationsströmung ist der Effekt der Vorkonditionierung wesentlich deutlicher. Schon bei schwachem Konvektionsanteil ($\varepsilon = 10^{-2}$) halbiert bis viertelt sich der zeitliche Aufwand für die Lösung des Problems. Bei konvektionsdominanten Problemen ermöglicht erst das Downwind Numbering die Konvergenz des Verfahrens.

Die Wirkung des Downwind Numbering auf Gittern unterschiedlicher Struktur ist stark problemabhängig. Bei der Schrägströmung werden die Charakteristiken nur schlecht auf das verwendete strukturierte Gitter abgebildet. Entsprechend beobachtet man hier besonders schlechte Konvergenz durch geometrisches Downwind Numbering im Fall schwacher Konvektion. Das algebraische Downwind Numbering ist dann stabiler, da es nicht nur die Konvektion berücksichtigt. Allgemein zeigt sich ein relativ geringer Effekt beider Verfahren auf strukturierten und unstrukturierten Gittern, die dort in den übrigen Fällen etwa gleiches leisten.

Bei unstrukturierten Gittern dagegen zeigen beide Methoden bessere Wirkung: in diesem Fall wird der Strömungsverlauf eher durch den Kantenverlauf wiedergegeben.

Der beobachtete Effekt, daß ohne Downwind Numbering bessere Konvergenz des SOR auf strukturierten Gittern erzielt wird, läßt sich auf eine zufällig günstigere Ausgangsnumerierung des verwendeten Gitters zurückführen, siehe dazu auch die Untersuchungen von A. P. Priesnitz [Pri96].

Die Rotationsströmung dagegen wird auf strukturierten Gittern durch alle Verfahren besser als auf unstrukturierten gelöst, da hier die Strömungsvektoren in weiten Teilen nahezu

parallel zum Rand und damit zu den Kanten der Triangulierung verlaufen.

Insbesondere bringt das geometrische Downwind Numbering auf strukturierten Gittern in allen Fällen Konvergenz, während SOR ohne oder mit algebraischer Vorkonditionierung bei Konvektionsdominanz divergiert. Dies ist umso bemerkenswerter, als selbst vorkonditionierte Krylov–Unterraum–Verfahren an diesem Problem scheitern. Siehe A. P. Priesnitz [Pri96].

In diesem Zusammenhang sei ausdrücklich auf die Möglichkeit der Verwendung des SOR mit Downwind Numbering als Vorkonditionierer für solche Verfahren hingewiesen. Hier entartet SOR zum Lösen eines unteren Dreieckssystems durch Vorwärtseinsetzen. Da Downwind Numbering die betragsmäßig größten Eintrage der Matrix in ihre untere Hälfte verschiebt, sollte es die Wirkung dieses Vorkonditionierers verstärken. Eine detailliertere Untersuchung dieser Möglichkeit sollte durchgeführt werden, da sie nach den hier dargestellten Ergebnissen sehr vielversprechend erscheint.

Die geschilderten Beobachtungen hinsichtlich der Abhängigkeit der Verfahren von der Struktur des Gitters legen nahe, bereits diese entsprechend auszulegen. Beispielsweise könnte bei der Generierung und Glättung des Gitters die Richtung der Konvektion als Kriterium zur Wahl neu anzulegender Kanten berücksichtigt werden. Besonders einfach ließe sich dies bereits durch einen *Edge–Swap* in annähernd quadratischen Patches aus zwei Dreiecken umsetzen . Dabei wählt man die innere Kante eines solchen Patch so, daß sie parallel zur Konvektion liegt. Siehe dazu auch [Koh00].

5.4.2. Struktur der Steifigkeitsmatrix

Die Abbildungen im Anhang B (S. 105) zeigen die Struktur der Steifigkeitsmatrix. Dabei sind die absoluten Größen der Werte entlang einer Farbskala von Gelb (minimaler Wert) nach Rot (maximaler Wert) dargestellt.

Im azyklischen Fall werden durch das geometrische Downwind Numbering die großen Einträge der Steifigkeitsmatrix in der Regel auf die untere Subdiagonale geschoben. Außerdem werden Matrixeinträge fern der Hauptdiagonalen zu mehr oder weniger diagonalen Ketten gruppiert. Diese entsprechen gerade den in Flußrichtung numerierten Strängen des Konvektionsgraphen.

Wenn es gelänge, diese diagonalen Ketten in die Nähe der Hauptdiagonalen zu verschieben und dadurch die Matrix möglichst auf Bandstruktur zu bringen, so könnten eventuell effektivere (direkte) Lösungsverfahren als die allgemeinen iterativen Verfahren auf das System angewendet werden.

Solche direkten Lösungsverfahren basieren auf der LU–Zerlegung, die für eine Bandmatrix selbst wieder Banddreiecksmatrizen produziert und deshalb nur geringen Fill–in im zerlegten System gegenüber der umnumerierten Problemmatrix verursacht.

5.4.3. Sequenzen

Im Abschnitt C (S. 110) werden Sequenzen von Zwischenresultaten des SOR bei Lösung des Problems ohne bzw. mit geometrischem Downwind Numbering gezeigt. Dabei wurde der Fall Schrägströmung mit $\varepsilon = 10^{-4}$ auf einem unstrukturierten 65×65 Gitter mit Relaxationsparameter $\omega = 0.8$ gerechnet.

Der Vergleich zwischen vorkonditionierter und nicht vorvorkonditionierter Rechnung zeigt sehr gut den Effekt, der durch das Downwind Numbering erzielt werden sollte:

Bereits nach einem Itationsschritt tritt ein Informationstransport weit in das Gebiet hinein ein und schon nach 15 Iterationsschritten ist bei geometrischem Downwind Numbering die Information durch das gesamte Gebiet transportiert worden. Im nicht vorkonditionierten Problem werden dazu wenigstens 25 Schritte benötigt.

Dies zeigt, daß das Downwind Numbering einen schnelleren Informationstransport ermöglicht, der natürlich mit höherem Konvektionsanteil zunimmt.

Ziel sollte es also sein, durch die Wahl des Kriteriums den Weg des Transports von Informationen durch das Gebiet optimal nachzuvollziehen, was entsprechend geringere CPU-Zeit der Rechnungen bedeuten würde.

6. Zusammenfassung und Ausblick

In dieser Arbeit wurden Methoden zum Downwind Numbering untersucht. Dazu wurde der FEM-Code *ADR* mitentwickelt und um drei graph-basierte Numerierungsstrategien erweitert.

Die Anwendung graphentheoretischer Methoden im FEM–Kontext erfordert entsprechende Verfahren zur Überführung der Problemstellung in ein Graphenmodell. Dazu wurde in der Arbeit ein geometrischer und ein algebraischer Ansatz verwirklicht, je nachdem, ob das Gitter oder die diskretisierte Matrix zur Erzeugung eines Konvektionsgraphen benutzt werden sollte.

Die numerischen Resultate haben gezeigt, daß das Downwind Numbering im konvektionsdominanten Fall Verbesserungen in der Löserzeit um bis zu 50% und mehr erreicht. Darüber hinaus wird in dem extrem schlecht konditionierten Fall der konvektionsdominanten Rotationsströmung eine Konvergenz der Lösung erst durch Downwind Numbering ermöglicht. An diesem Problem scheitern selbst vorkonditionierte Krylov–Unterraum–Verfahren.

Dies sind die ersten Untersuchungen zu Numerierungsstrategien bei SUPG– und Shock–Capturing–Stabilisierung der Galerkin–Diskretisierung.

Der FEM–Code *ADR* wurde mit dieser Arbeit erstmalig für umfangreiche numerische Experimente genutzt, was mit umfangreichem Debugging und Testen einherging.

Um die Korrektheit der implementierten Algorithmen zu überprüfen, wurde eine leistungsfähige Datenvisualisierung für Matrizen und Graphen entwickelt, die auf Postscript Level 2 basiert.

Für eine weitere Verbesserung der numerischen Ergebnisse wird die Suche nach angepaßteren Kriterien für die SUPG– und Shock–Capturing–Stabilisierung empfohlen. In der Literatur wurden bisher lediglich Kriterien für hybride Upwind–Techniken vorgestellt.

Weiterhin sei ausdrücklich auf die Möglichkeit der Verwendung des SOR mit Downwind Numbering als Vorkonditionierer für Krylov–Unterraum–Verfahren hingewiesen.

Es steht noch die Erweiterung von *ADR* um Schur–Komplement–Methoden für das Hackbusch–Verfahren sowie um den Block–Gauß–Seidel– bzw. Block–SOR–Löser für den Tarjan–Algorithmus an, um so auch für den zyklischen Fall numerische Untersuchungen durchführen zu können.

Der Tarjan–Algorithmus ist noch nicht optimal. Hier ließe sich ein verbessertes Verfahren von Aho, Hopcroft und Ullman einsetzen [AHU83]. Dabei handelt es sich um eine modifizierte Tiefensuche, die sich zusätzlich die Bearbeitungszeit für jedem Knoten merkt.

Die beobachteten Abhängigkeiten des Downwind Numbering von der Struktur des Gitters legen es nahe, schon bei der Gittergenerierung eine Anpassung des Gitters an die betrachtete Strömung zu berücksichtigen. Dies ließe sich bereits auch relativ unkompliziert durch Edge–Swap realisieren.

A. Numerische Ergebnisse

A.1. Konvergenzzeiten

A.1.1. Schrägströmung

Diffusionskoeffizient $\epsilon = 0.01$

Ohne Downwind Numbering

Relax.	1.2	1.3	1.4	1.5	1.6
33x33 s	55/0.39	**51/0.34**	70/0.51	104/0.73	172/1.23
33x33 u	54/0.39	**51/0.38**	68/0.49	92/0.68	134/0.96
65x65 s	171/4.33	131/3.34	**90/2.33**	99/2.53	147/3.78
65x65 u	172/4.58	133/3.54	**95/2.54**	102/2.66	144/3.76
129x129 s	584/57.93	466/46.31	363/36.18	268/27.36	**168/17.04**
129x129 u	592/61.31	474/48.96	371/38.26	277/28.68	**186/19.21**

Mit geometrischem Downwind Numbering

Relax.	1.2	1.3	1.4	1.5	1.6
33x33 s	**51/0.36**	63/0.44	98/0.68	174/1.24	535/3.88
33x33 u	49/0.36	**40/0.28**	59/0.43	86/0.63	137/0.99
65x65 s	154/3.89	114/2.88	**94/2.39**	142/3.78	248/6.53
65x65 u	153/3.96	114/2.96	**78/2.03**	86/2.23	137/3.53
129x129 s	539/53.44	421/41.61	317/31.43	223/22.01	**193/19.13**
129x129 u	549/56.21	431/43.96	328/33.63	235/23.86	**145/14.74**

Mit algebraischem Downwind Numbering

Relax.	1.3	1.4	1.5	1.6	1.7
33x33 s	**37/0.26**	57/0.38	101/0.71	247/1.73	-/-
33x33 u	**37/0.28**	55/0.39	83/0.61	137/0.96	-/-
65x65 s	126/3.34	90/2.39	**70/1.84**	130/3.46	-/-
65x65 u	116/2.99	81/2.08	**74/1.91**	127/3.28	-/-
129x129 s	450/44.79	349/34.74	258/25.69	172/17.16	**164/16.28**
129x129 u	432/44.29	330/33.78	239/24.48	**152/15.59**	173/17.49

Diffusionskoeffizient $\epsilon = 0.0001$

Ohne Downwind Numbering

Relax.	0.6	0.7	0.8	0.9	1.0
33x33 s	121/0.83	95/0.66	**75/0.54**	136/0.93	-/-
33x33 u	129/0.93	104/0.74	**84/0.59**	93/0.68	-/-
65x65 s	189/4.78	149/3.69	**118/2.94**	207/5.18	-/-
65x65 u	174/4.44	140/3.59	**115/2.93**	-/-	-/-
129x129 s	311/30.69	245/24.09	**196/19.28**	-/-	-/-
129x129 u	259/26.66	206/21.21	**166/17.14**	-/-	-/-

Mit geometrischem Downwind Numbering

Relax.	0.6	0.7	0.8	0.9	1.0
33x33 s	112/0.78	87/0.61	**69/0.49**	129/0.89	-/-
33x33 u	102/0.73	77/0.51	58/0.41	**43/0.29**	139/0.98
65x65 s	169/4.23	130/3.23	**103/2.56**	-/-	-/-
65x65 u	143/3.64	107/2.73	**80/2.06**	83/2.11	-/-
129x129 s	268/26.14	205/20.29	**161/15.83**	-/-	-/-
129x129 u	208/21.13	155/15.79	**115/11.73**	163/16.58	-/-

Mit algebraischem Downwind Numbering

Relax.	0.6	0.7	0.8	0.9	1.0
33x33 s	111/0.79	86/0.61	**68/0.46**	137/0.99	-/-
33x33 u	101/0.73	76/0.58	58/0.43	**47/0.34**	132/0.98
65x65 s	167/4.38	129/3.38	**101/2.63**	-/-	-/-
65x65 u	142/3.74	106/2.78	**79/2.09**	80/2.08	-/-
129x129 s	267/26.99	**204/20.56**	-/-	-/-	-/-
129x129 u	206/20.99	153/15.61	**112/11.44**	145/14.83	-/-

Diffusionskoeffizient $\epsilon = 0.000001$

Ohne Downwind Numbering

Relax.	0.6	0.7	0.8	0.9	1.0
33x33 s	121/0.84	95/0.63	**75/0.53**	169/1.18	-/-
33x33 u	134/0.99	107/0.76	**87/0.64**	118/0.88	-/-
65x65 s	189/4.78	149/3.78	**118/2.98**	-/-	-/-
65x65 u	183/4.71	**147/3.76**	394/10.06	-/-	-/-
129x129 s	312/30.71	246/24.23	**197/19.48**	-/-	-/-
129x129 u	277/28.61	**222/22.98**	-/-	-/-	-/-

Mit geometrischem Downwind Numbering

Relax.	0.6	0.7	0.8	0.9	1.0
33x33 s	112/0.78	87/0.59	**69/0.48**	140/0.98	-/-
33x33 u	104/0.76	78/0.56	59/0.43	**46/0.34**	163/1.18
65x65 s	170/4.21	131/3.21	**104/2.56**	-/-	-/-
65x65 u	148/3.79	111/2.84	**83/2.13**	116/2.96	-/-
129x129 s	271/26.43	**209/20.48**	-/-	-/-	-/-
129x129 u	220/22.31	164/16.74	**122/12.51**	-/-	-/-

Mit algebraischem Downwind Numbering

Relax.	0.6	0.7	0.8	0.9	1.0
33x33 s	111/0.79	86/0.61	**68/0.48**	-/-	-/-
33x33 u	103/0.76	78/0.56	58/0.43	**48/0.34**	151/1.13
65x65 s	169/4.33	130/3.31	**103/2.64**	-/-	-/-
65x65 u	147/3.89	110/2.91	**82/2.14**	98/2.61	-/-
129x129 s	270/27.51	**208/21.19**	-/-	-/-	-/-
129x129 u	217/22.06	161/16.38	**119/12.16**	-/-	-/-

A.1.2. Rotationsströmung

Diffusionskoeffizient $\epsilon = 0.01$

Ohne Downwind Numbering

Relax.	1.4	1.5	1.6	1.7	1.8
33x33 s	**520/3.63**	-/-	-/-	-/-	-/-
33x33 u	**632/4.94**	-/-	-/-	-/-	-/-
65x65 s	1759/43.59	**1432/36.31**	1781/45.43	-/-	-/-
65x65 u	2210/71.99	**1760/85.17**	-/-	-/-	-/-
129x129 s	5933/592.11	4864/484.16	3872/384.58	**2930/290.85**	-/-
129x129 u	-/-	6430/1230.42	4949/945.24	**3612/691.15**	-/-

Mit geometrischem Downwind Numbering

Relax.	1.4	1.5	1.6	1.7	1.8
33x33 s	**439/3.16**	-/-	-/-	-/-	-/-
33x33 u	**264/1.94**	-/-	-/-	-/-	-/-
65x65 s	1821/46.83	1370/34.31	**955/23.93**	-/-	-/-
65x65 u	1607/41.31	1370/34.53	**955/24.21**	-/-	-/-
129x129 s	6112/607.77	4828/478.58	3636/360.95	**2502/248.35**	-/-
129x129 u	-/-	5187/529.62	3720/379.91	**2393/244.42**	-/-

Mit algebraischem Downwind Numbering

Relax.	1.4	1.5	1.6	1.7	1.8
33x33 s	**476/3.41**	-/-	-/-	-/-	-/-
33x33 u	**385/2.86**	-/-	-/-	-/-	-/-
65x65 s	1919/49.29	1444/36.19	**1009/25.31**	-/-	-/-
65x65 u	1604/64.14	1161/29.66	**762/19.58**	-/-	-/-
129x129 s	5625/567.19	4459/438.43	3381/332.48	**2356/231.52**	-/-
129x129 u	-/-	5417/551.16	3929/400.66	**2571/262.52**	-/-

Diffusionskoeffizient $\epsilon = 0.0001$

Ohne Downwind Numbering

Relax.	0.7	0.8	0.9	1.0	1.1
33x33 s	-/-	-/-	-/-	-/-	-/-
33x33 u	-/-	-/-	-/-	-/-	-/-
65x65 s	-/-	-/-	-/-	-/-	-/-
65x65 u	-/-	-/-	-/-	-/-	-/-
129x129 s	-/-	-/-	-/-	-/-	-/-
129x129 u	-/-	-/-	-/-	-/-	-/-

Mit geometrischem Downwind Numbering

Relax.	0.7	0.8	0.9	1.0	1.1
33x33 s	8416/58.61	**6164/42.68**	-/-	-/-	-/-
33x33 u	-/-	**11877/81.11**	-/-	-/-	-/-
65x65 s	2271/56.79	1473/36.99	**1008/25.46**	-/-	-/-
65x65 u	-/-	-/-	**13949/361.46**	-/-	-/-
129x129 s	6357/647.35	4319/439.69	2750/279.92	**1389/141.71**	-/-
129x129 u	-/-	-/-	-/-	-/-	-/-

Mit algebraischem Downwind Numbering

Relax.	0.7	0.8	0.9	1.0	1.1
33x33 s	-/-	10606/69.24	**6739/45.64**	-/-	-/-
33x33 u	-/-	12157/81.41	**7105/48.53**	-/-	-/-
65x65 s	-/-	-/-	**11916/290.42**	-/-	-/-
65x65 u	-/-	-/-	**13908/348.81**	-/-	-/-
129x129 s	-/-	-/-	-/-	-/-	-/-
129x129 u	-/-	-/-	-/-	-/-	-/-

Diffusionskoeffizient $\epsilon = 0.000001$

Ohne Downwind Numbering

Relax.	0.6	0.7	0.8	0.9	1.0
33x33 s	-/-	-/-	-/-	-/-	-/-
33x33 u	-/-	-/-	-/-	-/-	-/-
65x65 s	-/-	-/-	-/-	-/-	-/-
65x65 u	-/-	-/-	-/-	-/-	-/-
129x129 s	-/-	-/-	-/-	-/-	-/-
129x129 u	-/-	-/-	-/-	-/-	-/-

Mit geometrischem Downwind Numbering

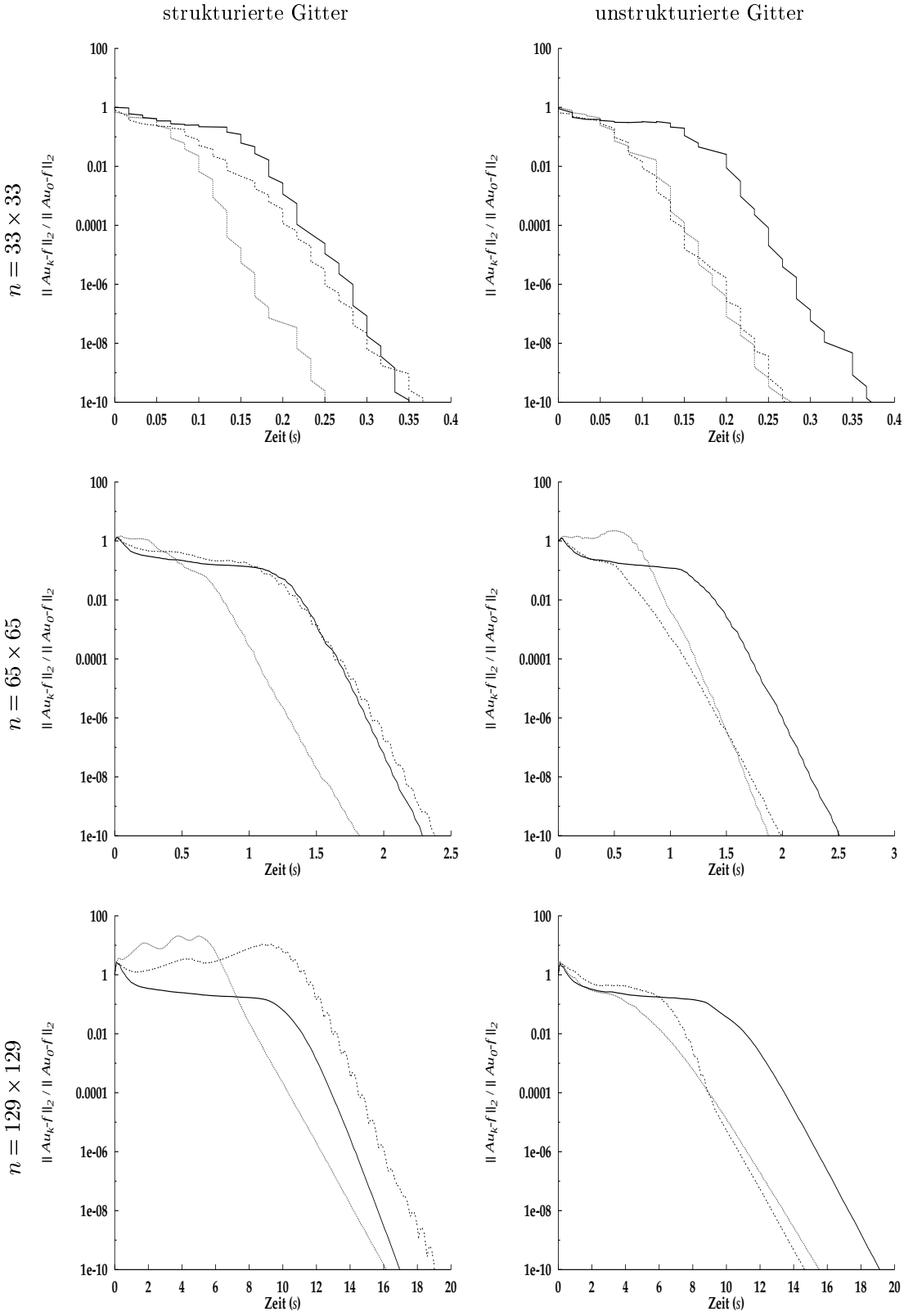
Relax.	0.6	0.7	0.8	0.9	1.0
33x33 s	12897/84.71	**9589/63.39**	-/-	-/-	-/-
33x33 u	-/-	-/-	-/-	-/-	-/-
65x65 s	4315/106.29	**3152/77.99**	-/-	-/-	-/-
65x65 u	-/-	-/-	-/-	-/-	-/-
129x129 s	-/-	**12324/1247.3**	-/-	-/-	-/-
129x129 u	-/-	-/-	-/-	-/-	-/-

Mit algebraischem Downwind Numbering

Relax.	0.6	0.7	0.8	0.9	1.0
33x33 s	-/-	-/-	-/-	-/-	-/-
33x33 u	-/-	-/-	-/-	-/-	-/-
65x65 s	-/-	-/-	-/-	-/-	-/-
65x65 u	-/-	-/-	-/-	-/-	-/-
129x129 s	-/-	-/-	-/-	-/-	-/-
129x129 u	-/-	-/-	-/-	-/-	-/-

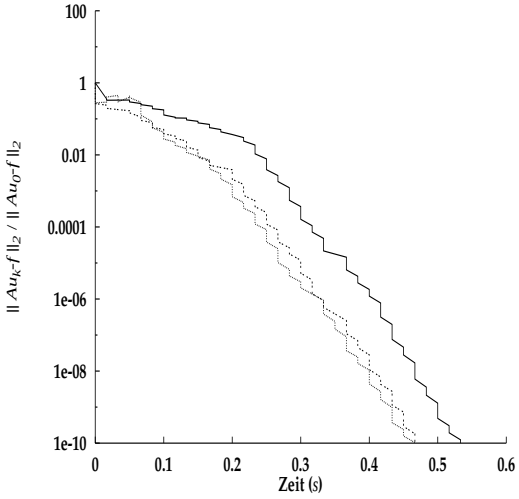
A.2. Konvergenzverläufe

A.2.1. Schrägströmung, $\varepsilon = 10^{-2}$

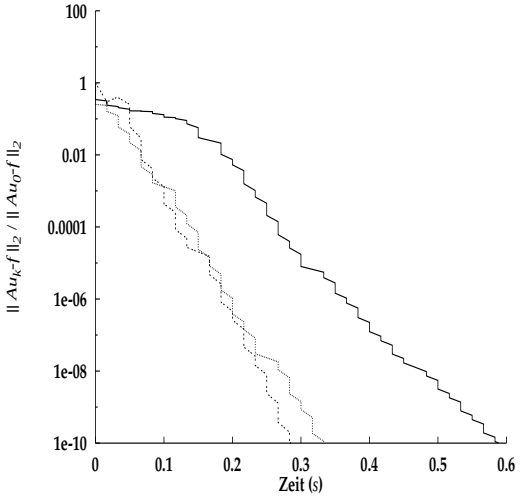


A.2.2. Schrägströmung, $\varepsilon = 10^{-4}$

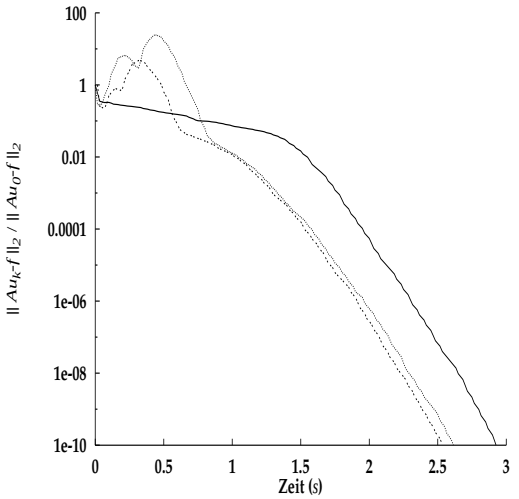
strukturierte Gitter

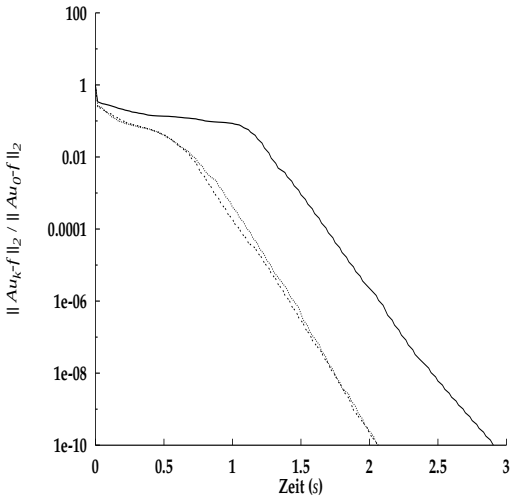


unstrukturierte Gitter

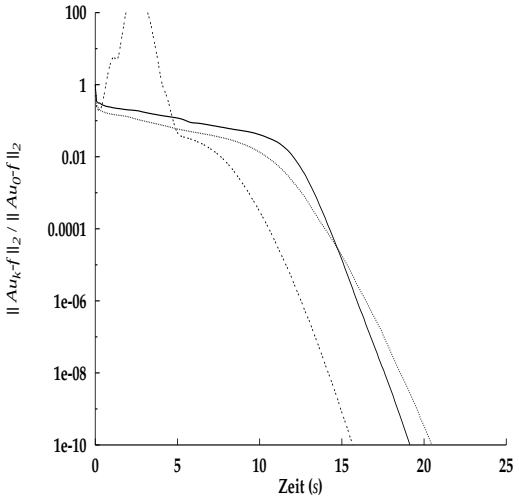


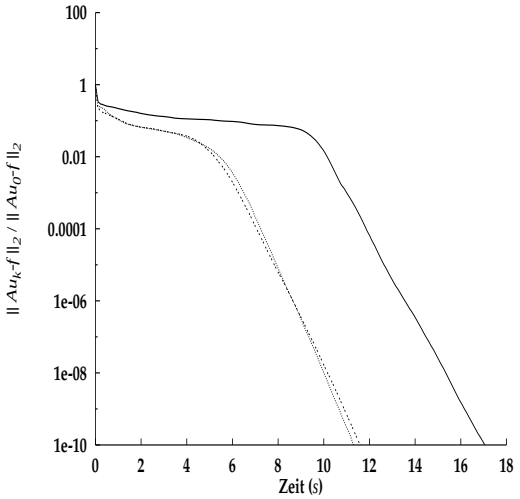
$n = 33 \times 33$





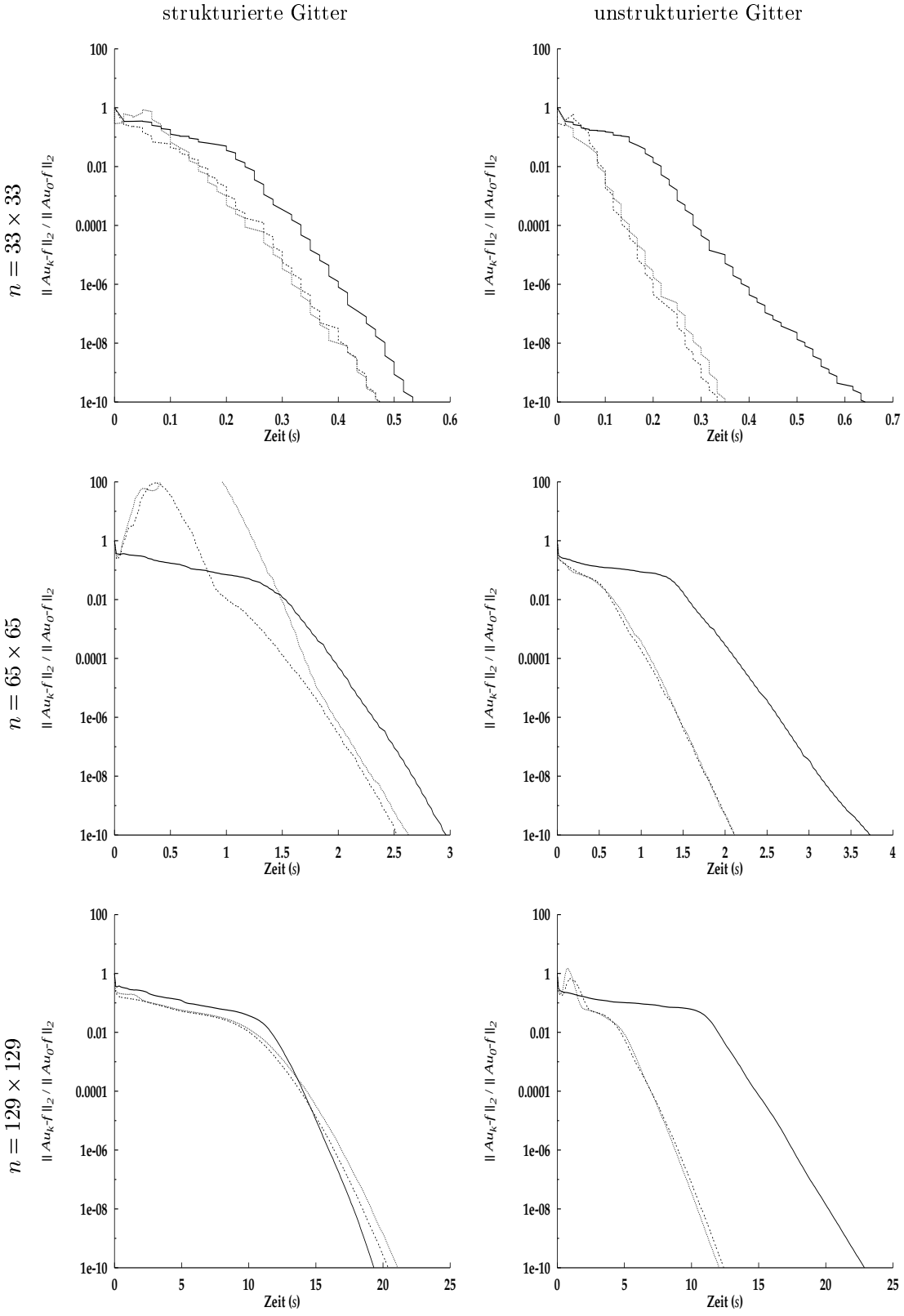
$n = 65 \times 65$



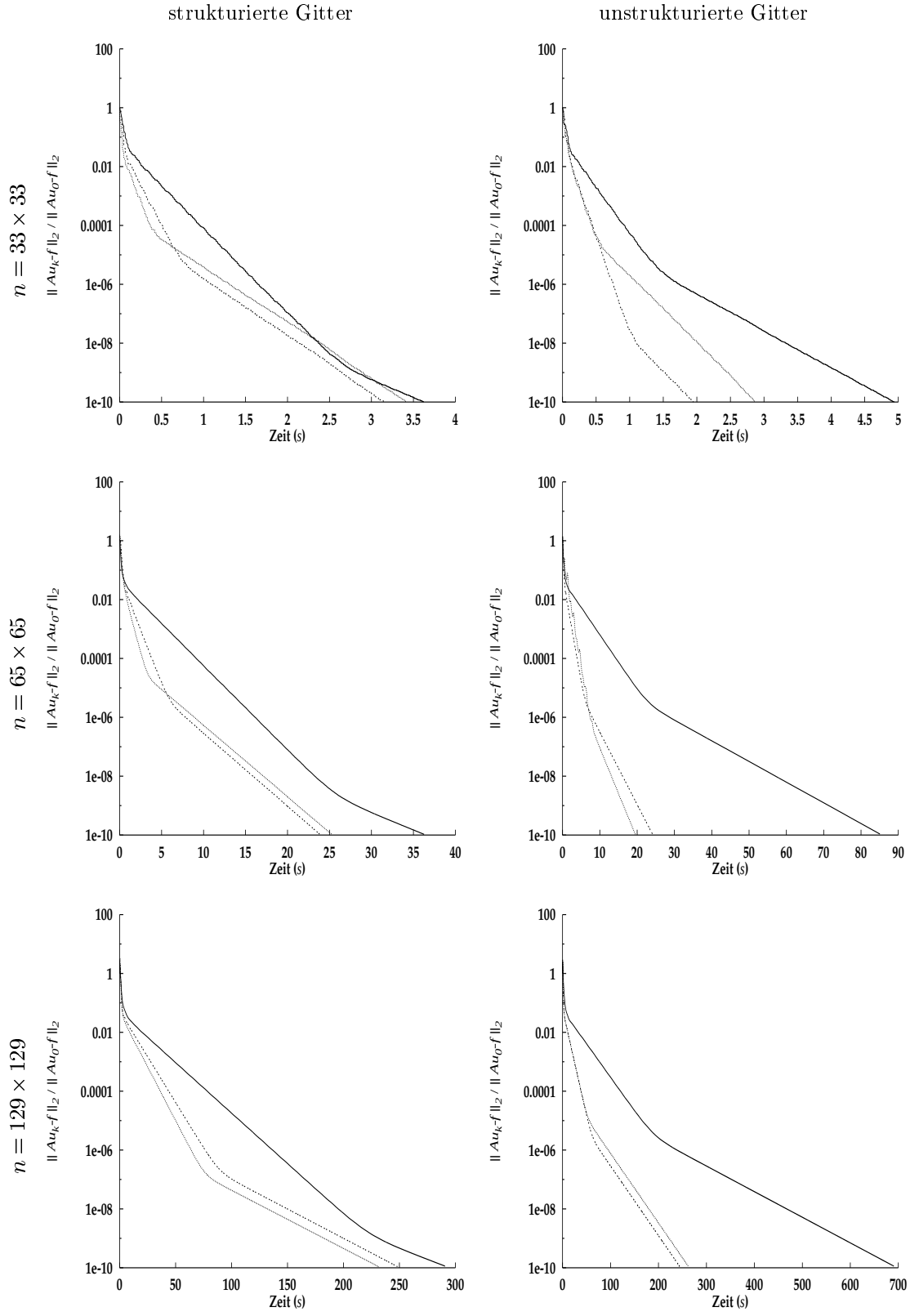


$n = 129 \times 129$

A.2.3. Schrägströmung, $\varepsilon = 10^{-6}$

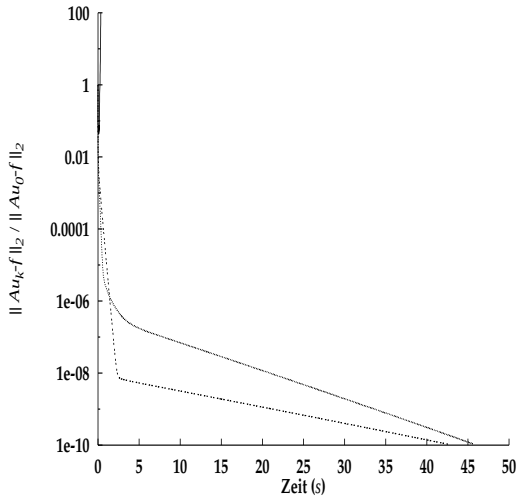


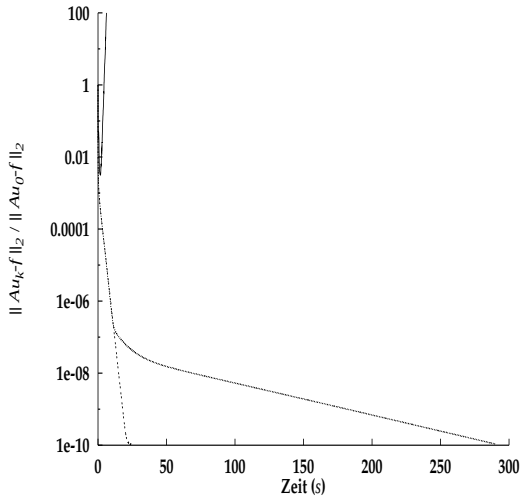
A.2.4. Rotationsströmung, $\varepsilon = 10^{-2}$

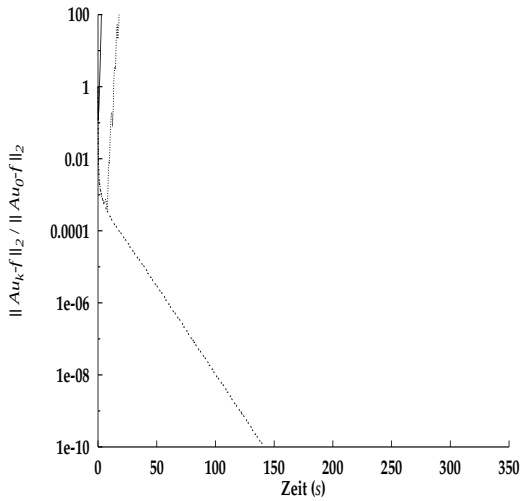


A.2.5. Rotationsströmung, $\varepsilon = 10^{-4}$

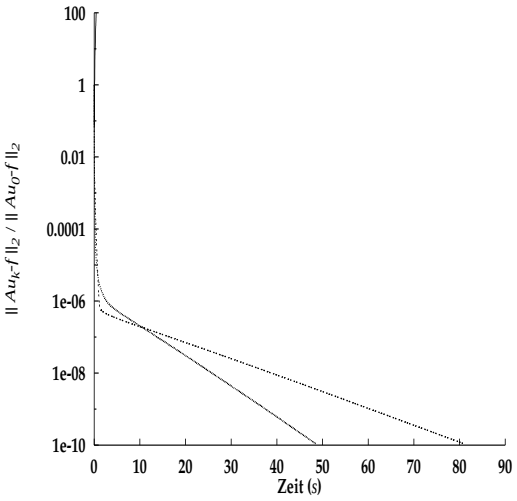
strukturierte Gitter

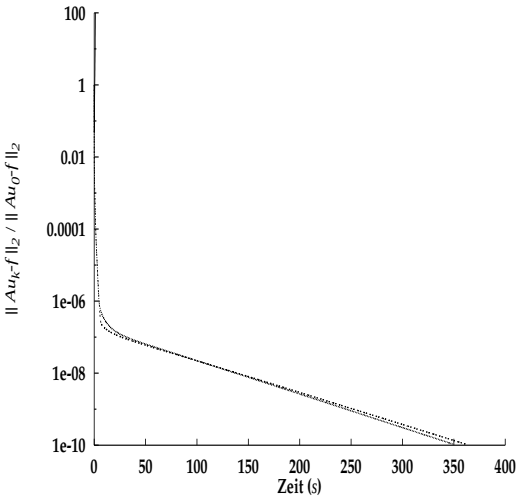


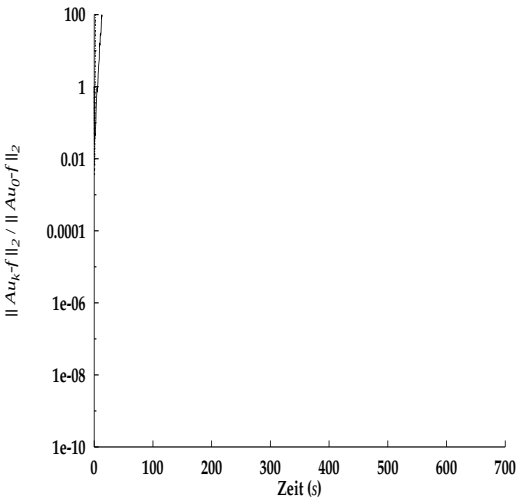




unstrukturierte Gitter





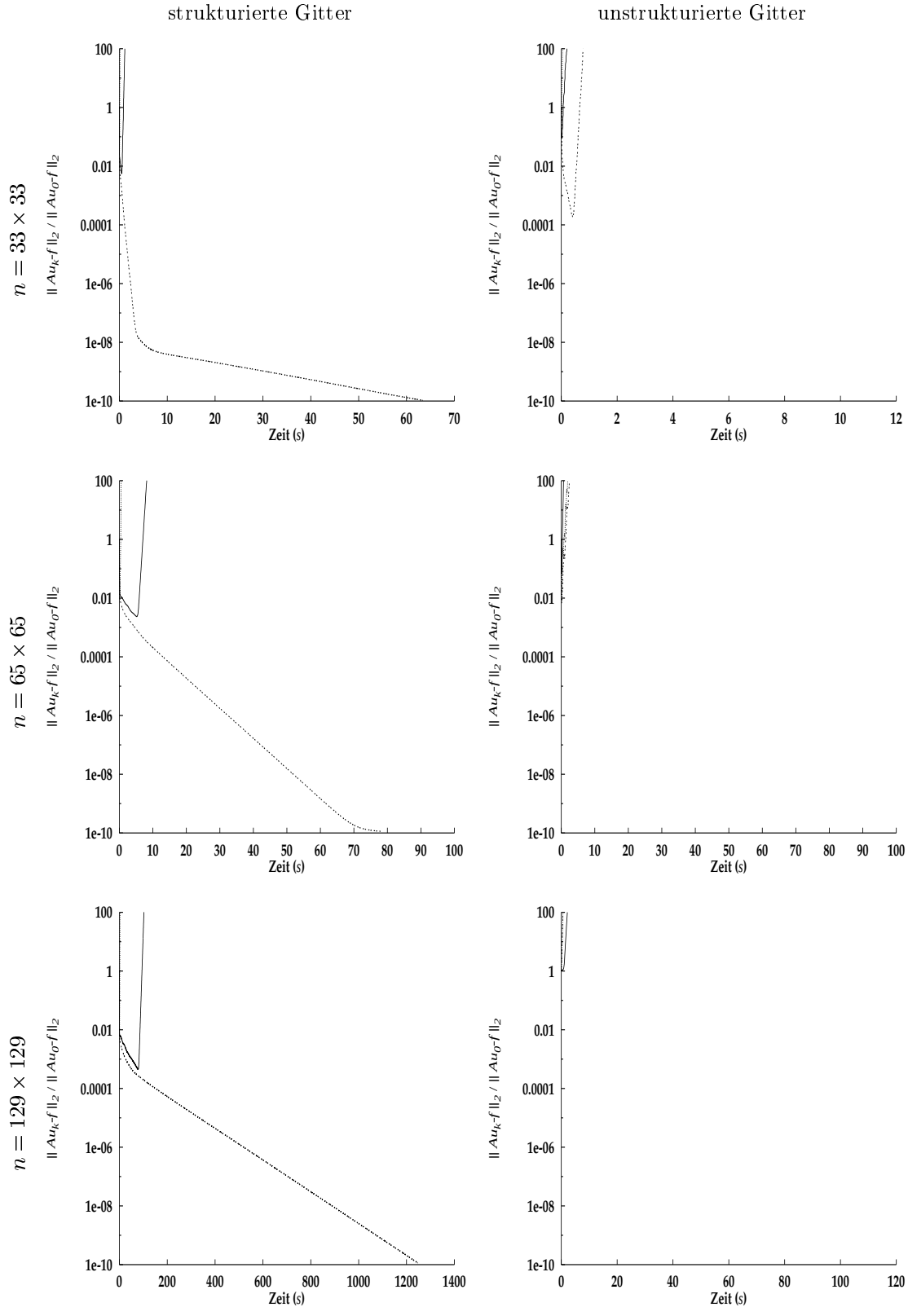


$n = 33 \times 33$

$n = 65 \times 65$

$n = 129 \times 129$

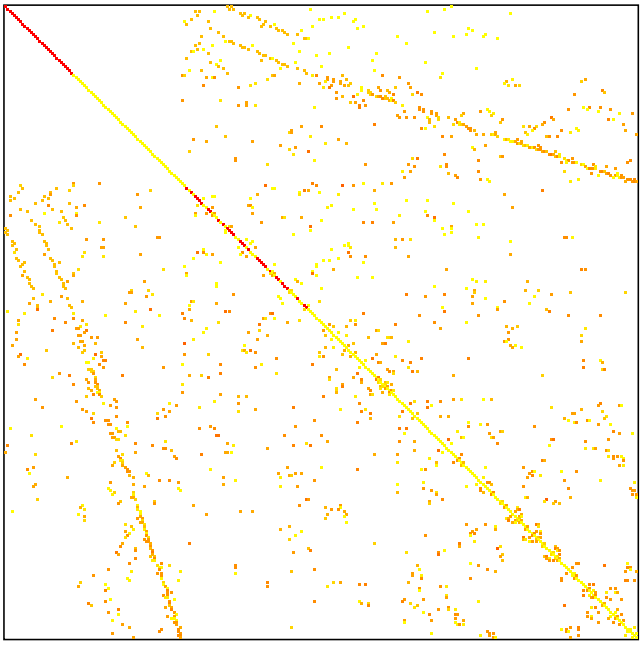
A.2.6. Rotationsströmung, $\varepsilon = 10^{-6}$



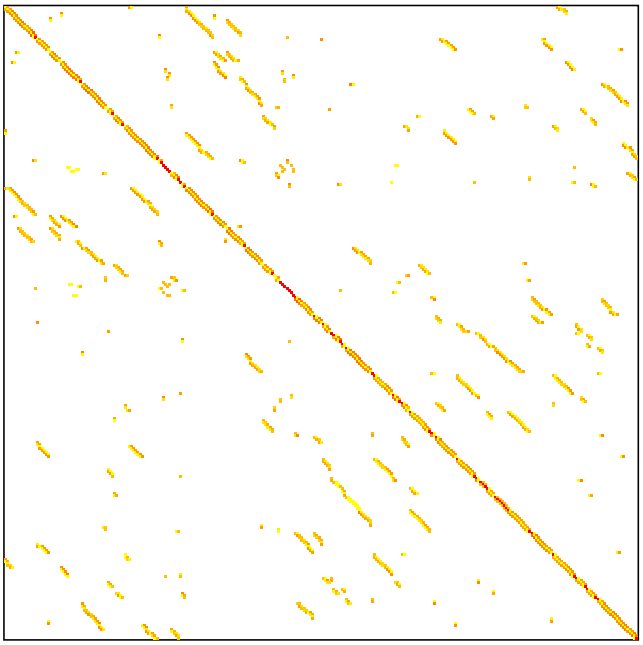
B. Struktur der Steifigkeitsmatrix bei Schrägströmung mit $\varepsilon = 10^{-6}$

Galerkin-FEM

ohne downwind numbering

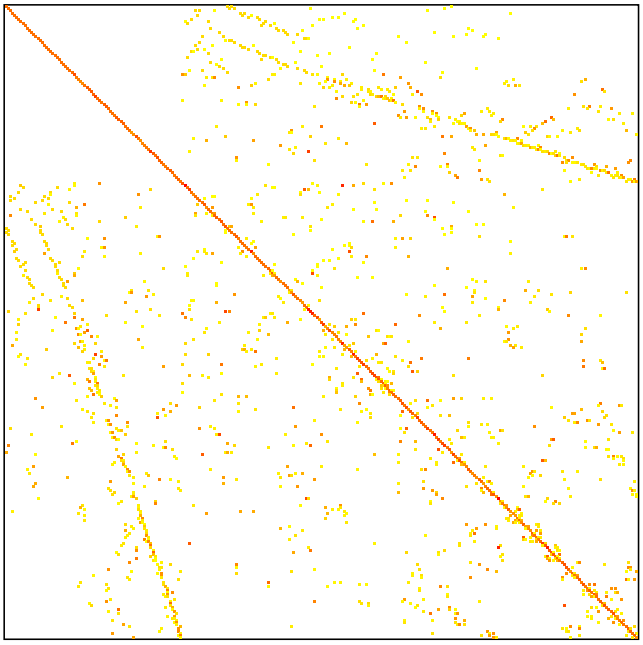


mit downwind numbering

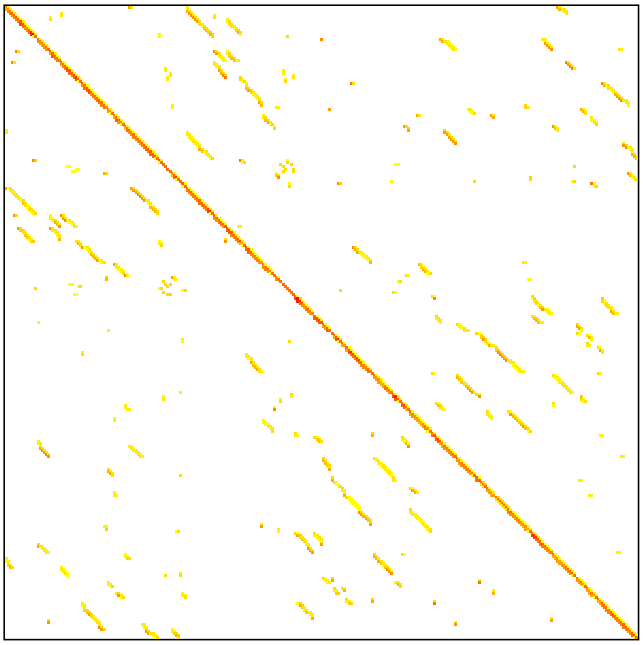


Galerkin-FEM mit SUPG

ohne downwind numbering

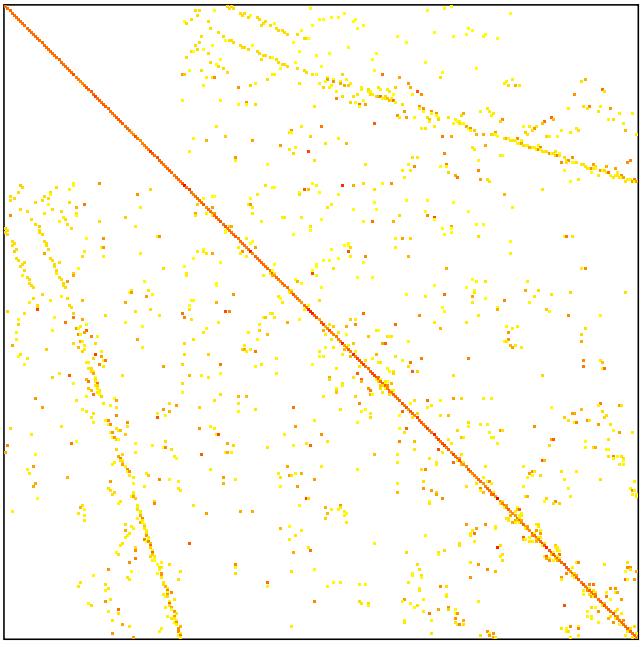


mit downwind numbering

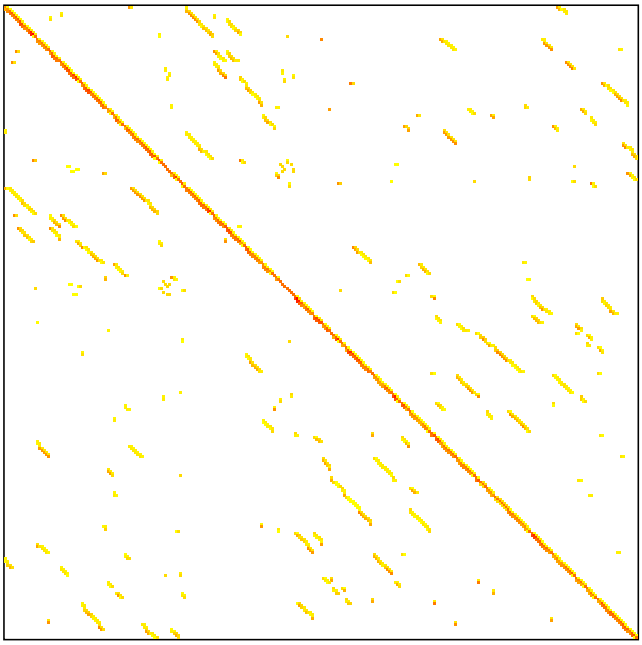


Galerkin-FEM mit SUPG und SC

ohne downwind numbering

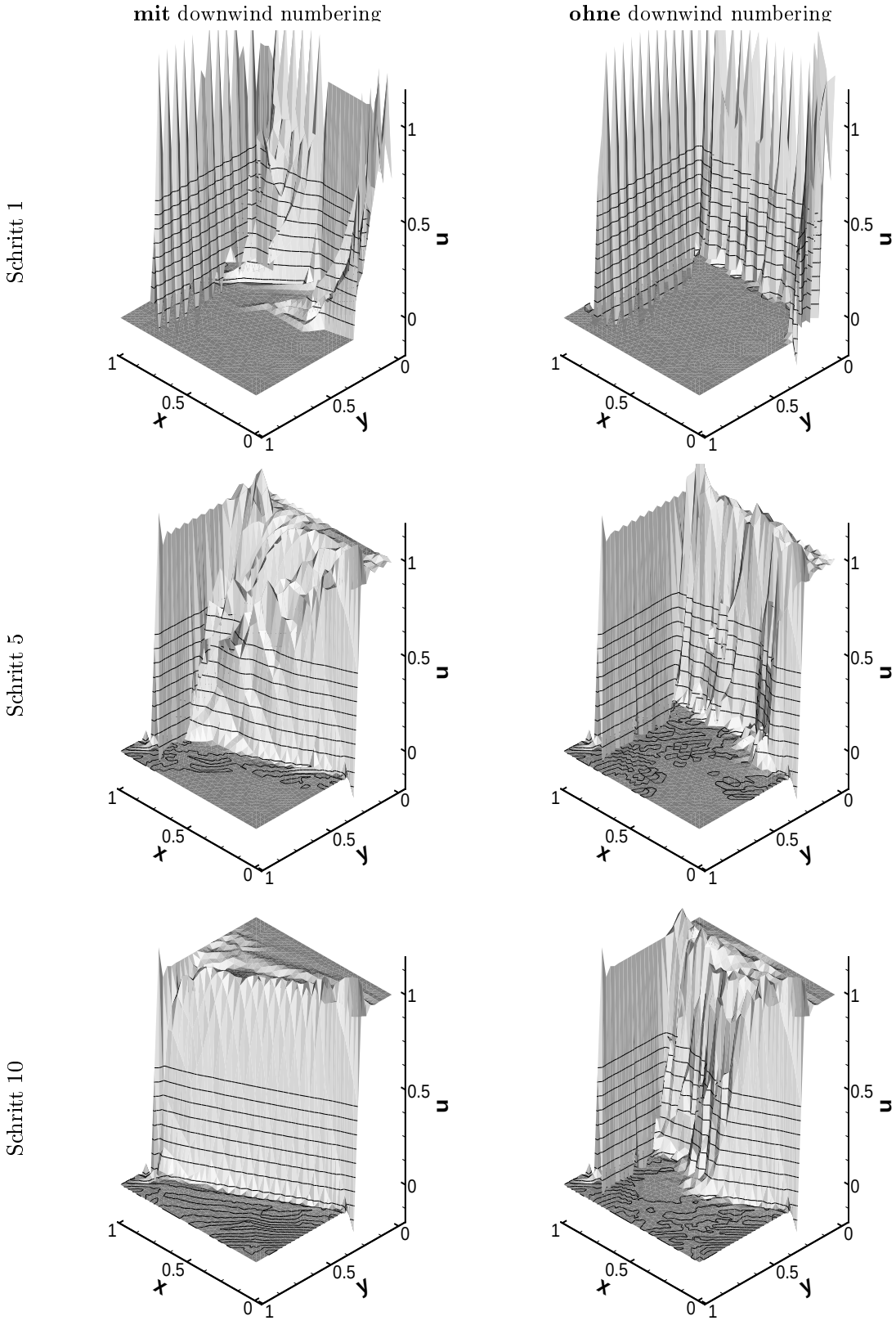


mit downwind numbering



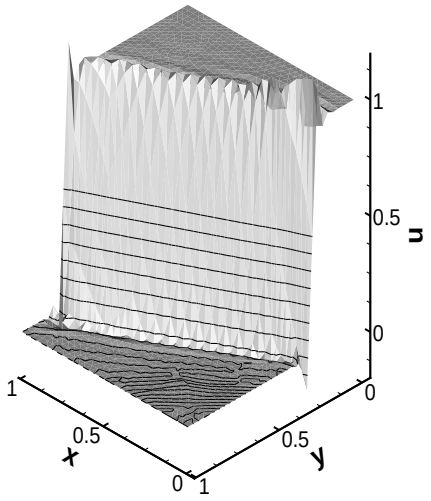
C. Sequenzen

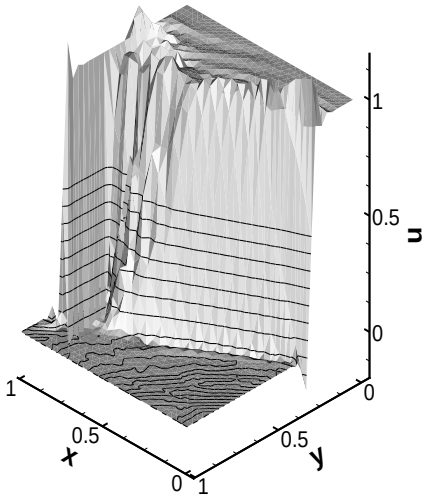
C.1. Schrägströmung ($\varepsilon = 10^{-4}$, $\omega = 0.8$, unstrukturiertes 65×65 Gitter)



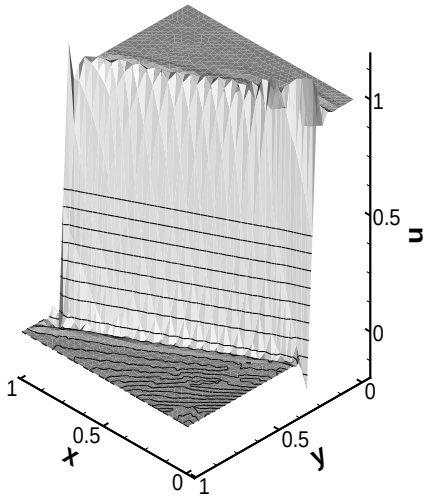
mit downwind numbering

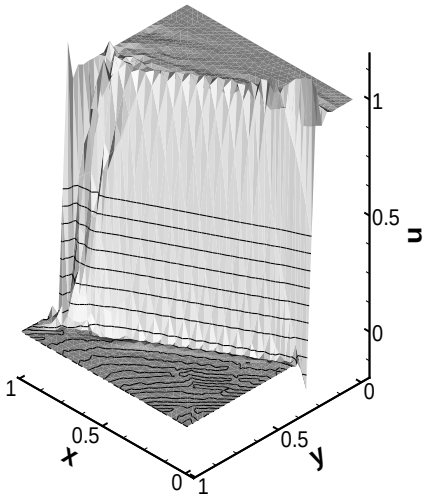
ohne downwind numbering



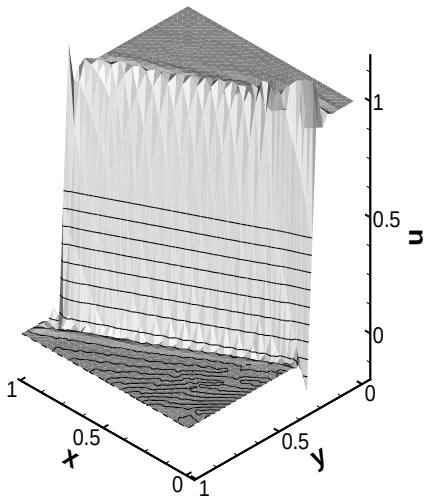


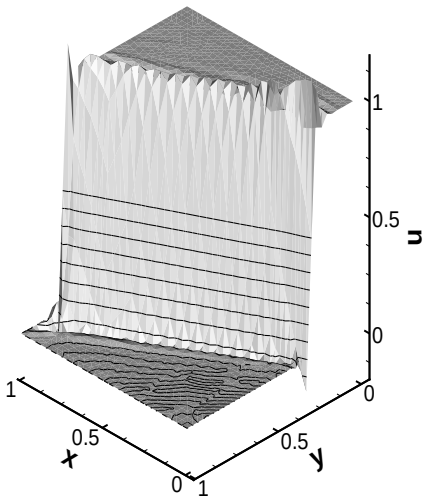
Schritt 15





Schritt 20



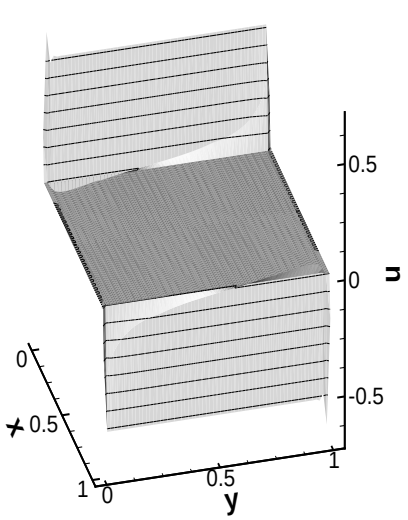


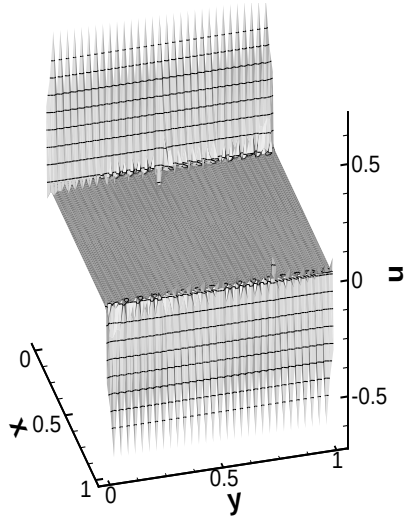
Schritt 25

C.2. Rotationsströmung ($\varepsilon = 10^{-4}$, $\omega = 0.8$, strukturiertes 65×65 Gitter)

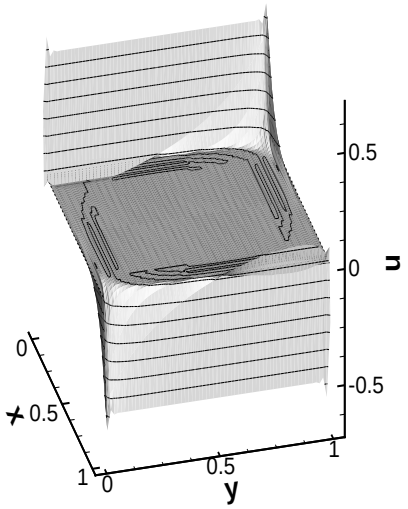
mit downwind numbering | **ohne** downwind numbering

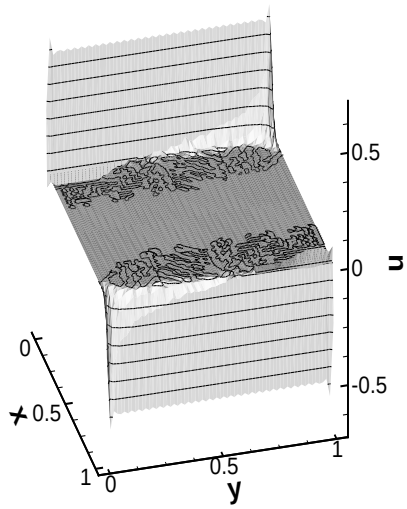
Schritt 1



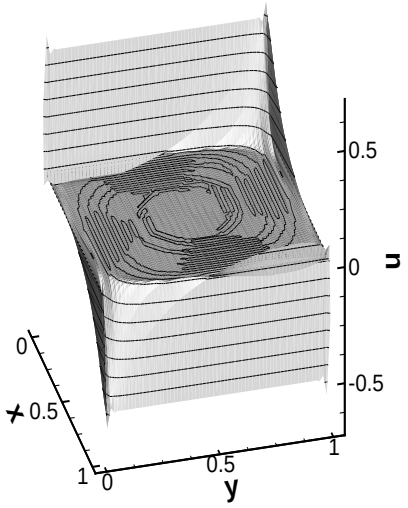


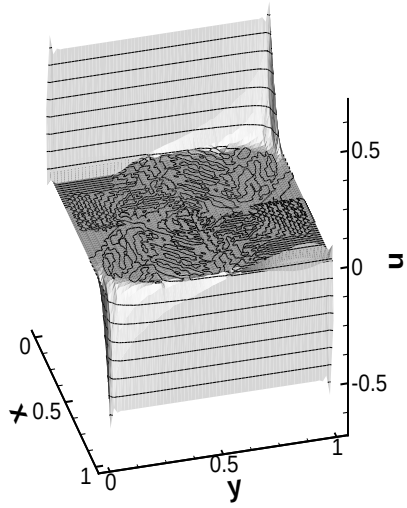
Schritt 5





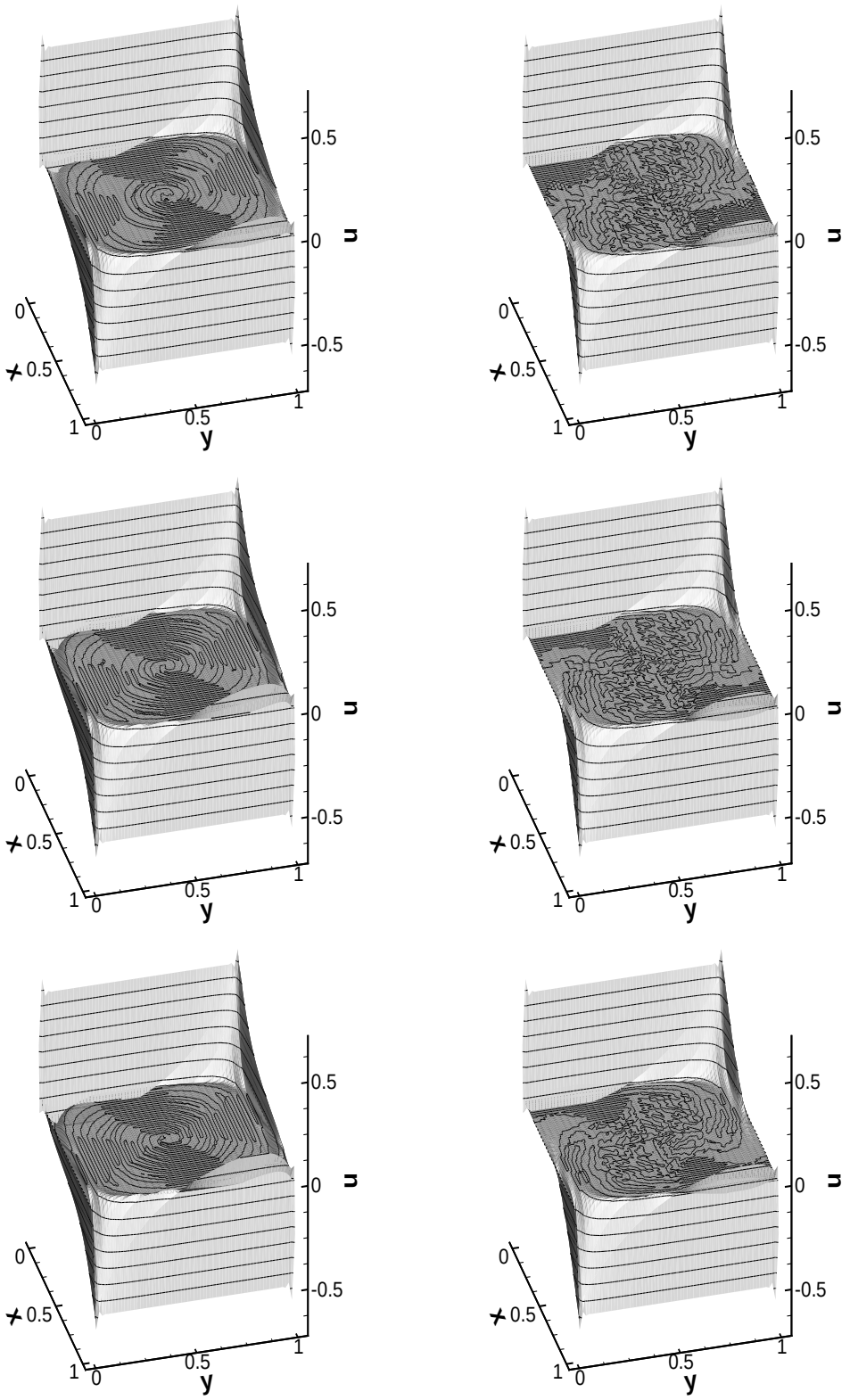
Schritt 10





mit downwind numbering

ohne downwind numbering



Abbildungsverzeichnis

Literaturverzeichnis

[AHU76] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley Series in Computer Science and Information Processing. Addison-Wesley Publishing Company, 1976.

[AHU83] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *Data Structures and Algorithms.* Addison-Wesley Publishing Company, 1983.

[Alt92] Hans Wilhelm Alt. *Lineare Funktionalanalysis.* Springer–Lehrbuch. Springer–Verlag, Berlin, Heidelberg, New York, 1992.

[BBC+94] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition.* SIAM, Philadelphia, PA, 1994.

[Bey98] Jürgen Bey. *Finite-Volumen- und Mehrgitter-Verfahren für elliptische Randwertprobleme.* Advances in Numerical Mathematics. B.G.Teubner Stuttgart, Leipzig, 1998.

[Bor99] Sabine Le Borne. *Multigrid methods for convection-dominated problems.* PhD thesis, Christian-Albrecht-Universität Kiel, 1999.

[Bra78] Achi Brandt. Multi-level adaptive techniques (mlat) for singular-perturbation problems. *NASA Langley Research Center, Hampton, Virgina*, 78(18):1–90, October 1978.

[Bra87] Andreas Brandstädt. The computational complexity of feedback vertex set, hamiltonian circuit, dominating set, steiner tree, and bandwidth on special perfect graphs. *Journal of information processing and cybernetics EIK*, 23:471–477, 1987.

[Bra94] Andreas Brandstädt. *Graphen und Algorithmen.* Leitfäden und Monographien der Informatik. B.G. Teubner Stuttgart, 1994.

[Bra97] Dietrich Braess. *Finite Elemente: Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie.* Springer–Lehrbuch. Springer–Verlag, Berlin, Heidelberg, New York, 1997.

[BW94] P. Bastian and G. Wittum. Adaptive multigrid methods: The UG concept. *Adaptive Methods: Algorithms, Theory and Applications, NNFM*, pages 17–37, 1994.

[BW95] Jürgen Bey and G. Wittum. Downwind numbering: A robust multigrid method for convection-diffusion problems on unstructured grids. In *Fast Solvers for*

Flow Problems. Proceedings of the 10th GAMM-Seminar Kiel, January 14 to 16, 1994, NNFM, volume 49, Braunschweig, 1995. Vieweg.

[CLR90] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to algorithms.* The MIT Press, Cambridge, Massachusets, 1990.

[CS99] R. Codina and O. Soto. Finite element implementation of two-equation and algebraic stress turbulence models for steady incompressible flows. *International Journal for Numerical Methods in Fluids*, 30(3):309–334, 1999.

[ea] Andreas Auge et al. *ParallelNS, User's Guide.*

[ENSS95] Guy Even, Joseph (Seffi) Naor, Baruch Schieber, and Madhu Sudan. Approximating minimum feedback sets and multi-cuts in directed graphs. *4th International IPCO Conference, Integer Programming and Combinatorial Optimization*, pages 14–28, 1995.

[GFLR83] H. Goering, A. Felgenhauer, G. Lube, and H.-G. Roos. *Singularly Perturbed Differential Equations.* Akademie–Verlag, 1983.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability.* A Series of Books in the Mathematical Sciences. W.H.Freeman and Company, 1979.

[GP97] Sabine Gutsch and Thomas Probst. Cyclic and feedback vertex set ordering for the 2d convection-diffusion equation. *Berichtsreihe des Mathematischen Seminars Kiel*, 22:1–14, Dezember 1997.

[GR94] Christian Großmann and Hans-Görg Roos. *Numerik partieller Differentialgleichungen.* Teubner-Studienbücher : Mathematik. Teubner, Stuttgart, 1994.

[GT98] David Gilbarg and Neil S. Trudinger. *Elliptic Partial Differential Equations of Second Order*, volume 224 of *Springer Series in Computational Mathematics.* Springer, Berlin, Heidelberg, New York, 1998.

[Hac80] Wolfgang Hackbusch. *Multi-Grid Methods and Applications*, volume 4 of *Springer Series in Computational Mathematics.* Springer–Verlag, Berlin, Heidelberg, New York, 1980.

[Hac84] Wolfgang Hackbusch. Multigrid convergence for a singular pertubation problem. *Linear Algebra and its Applications*, 58:125–145, April 1984.

[Hac86] Wolfgang Hackbusch. *Theorie und Numerik elliptischer Differentialgleichungen.* Teubner-Studienbücher : Mathematik. Teubner, Stuttgart, 1986.

[Hac95] Wolfgang Hackbusch. On the feedback vertex set problem for a planar graph. Technical Report 9503, Institut für Informatik und Praktische Mathematik, Christian-Albrecht-Universität Kiel, 1995. (submitted to Computing, 58 No 2:129–155, 1997).

[Jos96] N. Josuttis. *Die C++-Standardbibliothek.* Addison-Wesley, 1996.

[Jun94] Dieter Jungnickel. *Graphen, Netzwerke und Algorithmen.* BI-Wissenschaftsverlag Mannheim,Leipzig,Wien,Zürich, 1994.

[Kah58] W. Kahan. *Gauß-Seidel Method of Solving Large Systems of Linear Equations.* PhD thesis, University of Toronto, 1958.

[KLR] Tobias Knopp, Gert Lube, and Gerd Rapin. Stabilized fem with shock-capturing for advection-diffusion problems. In *Proc. GAMM Jahrestagung 2000, Göttingen.* (eingereicht bei ZAMM 2000).

[Kno99] Tobias Knopp. *Eine stabilisierte Finite-Elemente-Methode für das k/ε-Turbulenzmodell der inkompressiblen und nichtisothermen Navier-Stokes-Gleichungen.* Georg-August-Universität Göttingen, 1999. Diplomarbeit.

[Koh00] Joachim Kohlhammer. *Erzeugung und Verfeinerung von Finite–Elemente–Triangulierungen.* Georg–August–Universität Göttingen, 2000. Diplomarbeit.

[Lub94] Gert Lube. Stabilized galerkin finite element methods for convection dominated and incompressible flow problems. *Numerical Analysis and Mathematical Modelling*, pages 85–104, 1994.

[Lub99] Gert Lube. *Lineare Funktionalanalysis und Anwendungen auf partielle Differentialgleichungen.* Georg-August-Universität Göttingen, 1999. Skript Wintersemester 1999/2000.

[Lub00] Gert Lube. *Analysis und Numerik elliptischer Differentialgleichungen.* Georg-August-Universität Göttingen, 2000. Skript Sommersemester 2000.

[net] *netCDF Homepage.* http://www.unidata.ucar.edu/packages/netcdf.

[PH96] David D. Patterson and John L. Hennessy. *Computer architecture a quantitative approach.* Morgan Kaufmann Publishers, INC., San Francisco, California, 1996.

[Pri96] Andreas P. Priesnitz. *Untersuchung iterativer Lösungsverfahren am Beispiel diskretisierter Konvektions-Diffusions-Reaktions-Gleichungen.* Georg-August-Universität Göttingen, 1996. Diplomarbeit.

[Pro99] Thomas Probst. *Mehrgitterverfahren für Konvektionsdiffusionsgleichungen.* PhD thesis, Christian-Albrecht-Universität Kiel, 1999.

[ps] *Adobe PostScript.* http://www.adobe.com/print/postscript/main.html.

[QV94] A. Quarteroni and A. Valli. *Numerical Approximation of Partial Differential Equations*, volume 23 of *Springer Series in Computational Mathematics.* Springer, Berlin, Heidelberg, New York, 1994.

[RST96] Hans-Görg Roos, M. Stynes, and L. Tobiska. *Numerical Methods for Singularly Perturbed Differential Equations. Convection-Diffusion and Flow Problems.* Springer, Berlin, Heidelberg, New York, 1996.

[Sha79] A. Shamir. A linear time algorithm for finding minimum cutsets in reduced graphs. *SIAM Journal On Computing*, 8:654–655, 1979.

[Tar72] Robert Endre Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1:146–160, March 1972.

[Tar83] Robert Endre Tarjan. *Data Structures and Network Algorithms.* CBMS-NSF Regional Conference Series in Applied Mathematics. Society for industrial and applied mathematics, Philadelphia, Pennsylvania, 1983.

[Wes82] P. Wessling. A robust and efficient multigrid method. *Algorithms, Theory and Applications*, 41, 1982.

[WLS85] C. Wang, E. Lloyd, and M. Soffa. Feedback vertex sets and cyclically reducible graphs. *Journal of the ACM*, 32:296–313, 1985.

[Wun96] Julia Wunner. *Ein Feedback Vertex Set–Verfahren zur Lösung von Konvektions-Diffusionsgleichungen.* Eberhard–Karls–Universität Tübingen, 1996. Diplomarbeit.

[YGNR94] B. Yehuda, D. Geiger, J. Naor, and R. M. Roth. Approximation algorithms for the vertex feedback set problem with applications to constraint satisfaction and bayesian inference, 1994.

Danksagung

Ich möchte allen danken, die zum Gelingen dieser Arbeit beigetragen haben.

Herrn Prof. Dr. G. Lube danke ich für die interessante Themenstellung, sowie für seine hilfreiche und motivierende Betreuung.

Herrn Wassmann und Herrn Siebrasse danke ich für Ihre Hilfe bei den kleinen und großen Problemen mit Unix. Herrn Könnecke gilt mein Dank für seine Unterstützung bei *gnuplot*.

Dem *ADR* –Team im Alpha–Raum danke ich für die gute Stimmung während dieser Zeit. Vielen Dank an Knoppstar Tobias für unsere tägliche Lektüre, Gerd für seine wohlformulierten Bemerkungen, Priesel Priesi für sein aufmunterndes „Wasn“ und Viktor für die netten Geschichten aus der Heimat.

Für das Korrekturlesen meiner Arbeit möchte ich der Arbeitsgruppe sowie Bämchen, Marcus, Henrik, Johns, Angela und Rüdiger danken.