Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Mathematik

Zur Druckrobustheit und Adaptivität einer Virtuellen-Elemente-Methode für das Stokes-Problem

Masterarbeit

zur Erlangung des akademischen Grades

Master of Science (M. Sc.)

von
Herrn Derk Frerichs
geboren am 02.08.1992 in Kiel

Gutachter:

1. *Prof. Dr. Carsten Carstensen*
2. *Dr. Christian Merdon*

Eingereicht am Institut für Mathematik der
Humboldt-Universität zu Berlin am:

20. August 2019

Abstract

This work tackles a virtual element method for an approximation of the two-dimensional Stokes problem.

In a first step, the virtual element method from da Veiga *et al.* is motivated and an a-priori erroranalysis is examined [dLV17].

Coming from that, a computable residual-based error estimator is constructed and its reliability is shown. In contrast to finite element methods, the main difficulty is to build the error estimator in such a way that it consists only of variables that can be computed from the virtual element solution. Local contributions of this error estimator are used as refinement indicators in an adaptive mesh refinement algorithm.

The focus of this work lies in the construction of a pressure-robust right-hand side discretization. Pressure-robustness means that changes in the continuous right-hand side, which influence only the pressure, also on the discrete level leave the velocity unchanged. Even though the original virtual element method is exactly divergence-free, it is not pressure-robust. It uses an L^2-projection to approximate the right-hand side which does not preserve the divergence in general. This leads to a velocity error scaling with the inverse of the viscosity constant ν and resulting in large errors in the case of small ν. To recover a divergence-free approximation an interpolation into Raviart-Thomas spaces of order $k-1$ is used. This interpolation is possible on both triangles and slightly more involved on general polygons.

Furthermore, a comparison between this virtual element method and a finite element method of the same order and with the same number of degrees of freedom is drawn. It points out similarities, differences and difficulties of virtual element methods.

Another focus lies on the implementation of the method that is theoretically discussed for an arbitrary order and practically implemented for the lowest order case $k = 2$. The computations presented in this work and the provided softwarepackage give bricks for an implementation of the virtual element method of higher order and of more cunning methods.

Numerical experiments at the end of the chapters confirm the theoretical statements.

Keywords:
numerical mathematics, virtual element method, pressure-robustness, adaptivity

Zusammenfassung

Diese Arbeit befasst sich mit einer Virtuellen-Elemente-Methode zur Approximation des zweidimensionalen Stokes-Problems.

Im ersten Schritt wird die Virtuelle-Elemente-Methode von da Veiga *et al.* motiviert und einer a-priori Fehleranalyse unterzogen [dLV17].

Ausgehend davon wird ein berechenbarer residuumbasierter Fehlerschätzer konstruiert und dessen Verlässlichkeit bewiesen. Im Unterschied zu den Finiten-Elemente-Methoden ist die Hauptschwierigkeit dabei, den Fehlerschätzer ausschließlich aus Größen bestehen zu lassen, die mit der Virtuellen-Elemente-Lösung berechenbar sind. Die lokalen Beiträge dieses Fehlerschätzers werden als Verfeinerungsindikatoren in einem adaptiven Gitterverfeinerungsalgorithmus genutzt.

Der Fokus der Arbeit liegt in der Konstruktion einer druckrobusten Diskretisierung der rechten Seite. Druckrobust meint, dass Änderungen der kontinuierlichen rechten Seite, die im Kontinuierlichen nur den Druck beeinflussen, auch im Diskreten die Geschwindigkeit unverändert lassen. Die ursprüngliche Virtuelle-Elemente-Methode ist zwar exakt divergenzfrei dennoch aber nicht druckrobust. Sie verwendet zur Approximation der rechten Seite eine L^2-Projektion, die im Allgemeinen die Divergenzfreiheit nicht erhält. Dies führt zu einem Geschwindigkeitsfehler, der mit dem Inversen der Viskosität ν skaliert, woraus große Fehler für kleine ν resultieren. Zur Wiederherstellung der Divergenzfreiheit wird eine Interpolation in Raviart-Thomas-Räume der Ordnung $k-1$ genutzt. Die Interpolation ist sowohl auf Dreiecken als auch etwas komplizierter auf allgemeinen Polygonen möglich.

Außerdem wird ein Vergleich zwischen einer Finiten-Elemente-Methode von gleicher Ordnung und mit der gleichen Anzahl an Freiheitsgraden und der Virtuellen-Elemente-Methode gezogen. Er zeigt Gemeinsamkeiten, Unterschiede und Schwierigkeiten bei den Virtuellen-Elemente-Methoden auf.

Ein weiterer Schwerpunkt der Arbeit liegt auf der Implementierung der Methode, die theoretisch für beliebige Ordnungen diskutiert und praktisch für die kleinste Ordnung $k = 2$ durchgeführt wird. Die hier vorgeführten Berechnungen und das Softwarepaket bieten damit Bausteine zur Implementierung der Virtuellen-Elemente-Methode höherer Ordnung oder zur Implementierung komplizierterer Methoden.

Numerische Experimente am Ende der Kapitel bestätigen die theoretischen Aussagen.

Schlagwörter:
Numerische Mathematik, Virtuelle-Elemente-Methode, Druckrobustheit, Adaptivität

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Viele Probleme in der Physik und den Ingenieurswissenschaften werden in Form von Differentialgleichungen formuliert, deren Lösungen ein beobachtbares Verhalten beschreiben. So stellen beispielsweise die Maxwell-Gleichungen die grundlegende Beschreibung des Elektromagnetismus dar, die Materialgesetze die Beschreibung der Verformungen von Festkörpern und die Navier-Stokes-Gleichungen die der Strömungsmechanik.

Die enorme Relevanz der Navier-Stokes-Gleichungen zeigt sich insbesondere auch darin, dass die Existenz eindeutiger Lösungen zu den sieben Milleniumsproblemen gehört. Die Navier-Stokes-Gleichungen beschreiben das Verhalten viskoser newtonscher Flüssigkeiten und Gase und sind daher unter anderem für den aerodynamischen Bau von Autos, Flugzeugen oder Zügen von großer Bedeutung [GDN98]. In Fällen, bei denen die Trägheitskräfte gegenüber den Reibungskräften vernachlässigbar sind, reicht es bereits aus, eine Vereinfachung dieser Gleichungen, die sogenannten Stokes-Gleichungen, zu lösen [Bes06], mit denen sich auch diese Arbeit beschäftigt. Das Verständnis der Stokes-Gleichungen ist daher eine wesentliche Voraussetzung für die Analyse der weitaus komplizierteren Navier-Stokes-Gleichungen.

Doch selbst für diese Vereinfachung lassen sich nur in wenigen Fällen explizite Lösungen angeben, weswegen auf Näherungslösungen zurückgegriffen werden muss. Seit den 1950er Jahren erfreuen sich beispielsweise die mittlerweile gut bekannten Finiten-Elemente-Methoden (FEM) großer Beliebtheit, die vor allen Dingen in den Ingenieurswissenschaften ein grundlegendes Hilfsmittel sind [Clo04]. Dafür wird das betrachtete Gebiet üblicherweise in Dreiecke, manchmal auch in Rechtecke (in 2D) oder in Tetraeder, manchmal Hexaeder (in 3D) zerlegt. Die Lösung wird dann in einem endlich dimensionalen Finiten-Elemente-Raum bestehend aus Polynomen eines gewissen Grades berechnet [BS02].

Es gibt allerdings auch einige Probleme, bei denen flexiblere Zerlegungen des Gebiets in Polygone beziehungsweise Polyeder von Vorteil sind. Dazu zählen zum Beispiel die Analyse von Rissausbreitungen in Materialien, Topologieoptimierungen, Fluid-Festkörperwechselwirkungen oder auch Zwei-Phasen-Flüsse, s. [Bre14] und die dortigen Referenzen. Polygone im Zweidimensionalen beziehungsweise Polyeder im Dreidimensionalen erlauben unter anderem einfachere Zerlegungen des Gebiets, eine effizientere Verfeinerung und sind robuster gegenüber Verdrehungen oder Verzerrungen des Gebiets. Neben anderen Methoden wie beispielsweise der hybriden dPG- oder Mimetischen-Finite-Differenzen-Methode eignen sich vor allen Dingen die Virtuellen-Elemente-Methoden (VEM) für flexible Zerlegungen, s. [dBMR14c] und die dortigen Referenzen.

Die erste Virtuelle-Elemente-Methode ist 2013 von da Veiga *et al.* als Entwicklung der Mimetischen-Finite-Differenzen-Methoden entstanden und kann als eine Erweiterungen von Finiten-Elemente-Methoden auf polygonale beziehungsweise polyhedrale Zerlegun-

gen und nicht-polynomielle Ansatzfunktionen gesehen werden [dBC+13]. Im Gegensatz zu den Finiten-Elemente-Methoden kann das Gebiet dabei in nahezu beliebige Polygone zerlegt werden. Die Lösungen werden zwar auch auf endlich dimensionalen Räumen errechnet, die bei den Virtuellen-Elemente-Methoden allerdings aus Polynomen und weiteren (virtuellen) nicht-Polynomen bestehen. Das hat zur Folge, dass die Basisfunktionen nicht mehr explizit sondern nur implizit zur Verfügung stehen, woher auch der Name herrührt [dBC+13].

Seit ihrer Entdeckung besteht großes Interesse an den Virtuellen-Elemente-Methoden. Diese werden auf die verschiedensten Probleme angewandt, darunter lineare Elastizitäts-Probleme, magnetostatische Probleme oder auch gemischte Probleme wie das oben angesprochene Stokes-Problem, s. [dLV17] und die dortigen Referenzen. Mit dem zuletzt genannten Problem beschäftigt sich diese Arbeit als Vorbereitung auf das umfangreichere Navier-Stokes-Problem.

Das Stokes-Problem modelliert den Fluss viskoser newtonscher Fluide unter Einwirkung äußerer Kräfte $\boldsymbol{f}$ in einem Gebiet Ω und sucht dabei ein genügend glattes Paar $(\boldsymbol{u}, p)$, so dass

$$\begin{aligned} -\nu\Delta\boldsymbol{u} + \nabla p &= \boldsymbol{f}, \\ \operatorname{div}\boldsymbol{u} &= 0, \\ \boldsymbol{u}_{|\partial\Omega} &= 0, \end{aligned}$$

wobei ν für die Viskosität des Mediums steht [Bar16]. Die Lösung der Stokes-Gleichungen ist ein Paar aus Geschwindigkeit $\boldsymbol{u}$ und Druck p des Fluids in dem bestimmten Gebiet. Eine fundamentale Eigenschaft dieser Lösung ist, dass die Geschwindigkeit unabhängig von beliebigen Gradientenfeldern ∇q als Komponente in den äußeren Kräften ist; sie bleibt unbeeinflusst [JLM+17]. Wegen der Divergenzfreiheit und den Nullranddaten folgt nämlich aus dem Gaußschen-Integralsatz, dass

$$\int_\Omega \boldsymbol{u} \cdot \nabla q \,\mathrm{d}\Omega = -\int_\Omega \operatorname{div}\boldsymbol{u}\, q \,\mathrm{d}\Omega + \int_{\partial\Omega} \boldsymbol{u} \cdot \boldsymbol{n}\, q \;\mathrm{d}s = 0,$$

wobei $\boldsymbol{n}$ der äußere Normalenvektor ist. Mit anderen Worten steht die Geschwindigkeit also insbesondere L^2-orthogonal auf Gradientenfelder.

Zur Sicherstellung der Konvergenz numerischer Methoden zur Näherung des Stokes-Problems muss die sogenannte inf-sup-Stabilität erfüllt werden, die überprüft, ob die diskreten Räume zusammen genutzt werden können. Um dies zu erreichen, wird oft die fundamentale Invarianz der Stokes-Gleichung verletzt, da die Divergenzfreiheit zum Erfüllen der inf-sup-Bedingung nur diskret gefordert wird [JLM+17]. Diese nur diskret divergenzfreien Funktionen stehen nicht mehr orthogonal auf beliebige Gradientenfelder, wodurch die diskrete Geschwindigkeit durch diese Gradientenfelder beeinflusst wird. Dieses Verhalten wird im Folgenden gemeint, wenn von fehlender *Druckrobustheit* gesprochen wird. Eine Methode wird druckrobust genannt, wenn Änderungen des Drucks, die im Kontinuierlichen die Geschwindigkeit nicht beeinflussen, auch im Diskreten keinen Einfluss auf die Geschwindigkeit haben. Fehlende Druckrobustheit macht insbesondere bei Fluiden mit kleinen Viskositäten ν Probleme, denn es lässt sich zeigen, dass der Geschwindigkeitsfehler nicht exakt divergenzfreier Methoden mit ν^{-1} skaliert [JLM+17].

Um für kleine Viskositäten dennoch zufriedenstellende Ergebnisse erzielen zu können, muss daher ein erheblicher Rechenaufwand betrieben werden.

Eine Möglichkeit, den Einfluss des Drucks auf die Geschwindigkeit zu verringen, ist die sogenannte grad-div-Stabilisierung. Durch Einfügen eines parameterabhängigen Terms, der im Kontinuierlichen Null und im Diskreten verschieden von Null ist, wird zwar der Einfluss vermindert, verhindert ihn aber nicht vollständig [JLM$^+$17]. Eine Möglichkeit zum vollständigen Herstellen der exakten Divergenzfreiheit auch im Diskreten und damit zur Herstellung der kompletten Druckrobustheit ist vor wenigen Jahren in [Lin14] vorgestellt worden. Während für Finite-Elemente-Methoden die fehlende Divergenzfreiheit mit Hilfe von Rekonstruktionsoperatoren repariert werden kann [JLM$^+$17], gibt es noch keine druckrobusten Virtuellen-Elemente-Methoden.

Selbst die vielversprechende exakt divergenzfreie Virtuelle-Elemente-Methode von da Veiga, Lovadina und Vacca wird diesem Problem nicht Herr [dLV17]. Dadurch, dass die virtuellen Basisfunktionen nicht explizit zur Verfügung stehen, muss zur Berechnung der rechten Seite eine Projektion benutzt werden. Dort wird die L^2-Projektion P_{k-2} benutzt, die im Allgemeinen die Divergenzfreiheit nicht erhält, denn es gilt für die diskrete Geschwindigkeit $\boldsymbol{u}_h$

$$\int_\Omega \nabla q \cdot P_{k-2}\boldsymbol{u}_h \,\mathrm{d}\Omega \neq 0.$$

Daraus resultiert ein Fehler zwischen der kontinuierlichen Geschwindigkeit $\boldsymbol{u}$ und der diskreten Geschwindigkeit $\boldsymbol{u}_h$, der sich unter Annahme genügender Regularität durch

$$\|\boldsymbol{u} - \boldsymbol{u}_h\|_{H^1(\Omega)} \lesssim |h_\mathcal{T}^k \boldsymbol{u}|_{H^{k+1}(\Omega)} + \nu^{-1} |h_\mathcal{T}^k \boldsymbol{f}|_{H^{k-1}(\Omega)}$$

beschränken lässt, wobei $h_\mathcal{T}$ die lokale Gitterweite, $2 \leq k \in \mathbb{N}$ die Ordnung der Methode und $\|\cdot\|_{H^1(\Omega)}$ beziehungsweise $|\cdot|_{H^{k-1}(\Omega)}$ die H^1-Norm beziehungsweise die H^{k-1}-Seminorm ist [dLV17]. Der zweite Term ist der Konsistenzfehler der rechten Seite, der auch vom exakten Druck abhängt. Die Virtuelle-Elemente-Methode ist also, auch wenn die Ansatzfunktionen exakt divergenzfrei sind, nicht druckrobust.

In dieser Arbeit wird aufbauend auf der oben genannten Virtuellen-Elemente-Methode eine druckrobuste Variante von beliebiger Ordnung entwickelt, analysiert und anhand numerischer Beispiele verifiziert. Dafür wird zunächst eine erweiterte Virtuelle-Elemente-Methode vorgestellt, die zwar die Konvergenzordnung verbessert, allerdings nicht vollständig druckrobust ist. Die fehlende Druckrobustheit wird daraufhin mit Hilfe einer $H(\mathrm{div})$-konformen Interpolation I_{RT} in Raviart-Thomas-Räume wieder hergestellt, welche die exakte Divergenzfreiheit der virtuellen Funktionen erhält. Für sie gilt dann nämlich für alle genügend glatten Gradienten ∇q, dass

$$\int_\Omega \nabla q \cdot I_{\mathrm{RT}}\boldsymbol{u}_h \,\mathrm{d}\Omega = 0.$$

Es wird sich herausstellen, dass sich der Fehler zwischen der kontinuierlichen Geschwindigkeit $\boldsymbol{u}$ und der diskreten Geschwindigkeit $\boldsymbol{u}_h$ unter Annahme genügender Regularität durch

$$\|\boldsymbol{u} - \boldsymbol{u}_h\|_{H^1(\Omega)} \lesssim |h_\mathcal{T}^k \boldsymbol{u}|_{H^{k+1}(\Omega)} + |h_\mathcal{T}^k \Delta\boldsymbol{u}|_{H^{k-1}(\Omega)}$$

abschätzen lässt. Die Konvergenz der Methode ist damit unabhängig von der Viskosität und insbesondere auch von Gradientenfeldern in den zu balancierenden Kräften.

Da die Basisfunktionen der Virtuellen-Elemente-Methoden nur implizit vorliegen, ist die Implementierung dieser Methode nicht trivial. Daher liegt ein weiterer Fokus dieser Arbeit auf der Implementierung dieser Virtuellen-Elemente-Methode. Es werden die grundlegenden Matrizen zur Berechnung der Virtuellen-Elemente-Lösung schrittweise aufgebaut, mit denen die Methode implementiert wird. Die Implementierung beruht im Wesentlichen auf Projektionen in Polynomräume, die nur mit Hilfe der Freiheitsgrade explizit berechnet werden können. Mit Hilfe dieser Bausteine lässt sich der Aufbau dieser Methode aber auch anderer Virtueller-Elemente-Methoden einfach nachvollziehen.

Lösungen, die Singularitäten enthalten, können im Allgemeinen nicht mit der optimalen Konvergenzordnung approximiert werden [BS02]. Deshalb beschäftigt sich ein weiterer Teil der Arbeit mit der Konstruktion eines verlässlichen Fehlerschätzers, mit dem adaptive Verfeinerungen möglich sind. Eine große Schwierigkeit dabei ist, die Terme des Fehlerschätzers nur aus berechenbaren Anteilen bestehen zu lassen. Dafür werden die Stabilisierungsbilinearform der Methode und erneut Projektionen in Polynomräume genutzt. Der Fehlerschätzer in Zusammenarbeit mit einem vorgestellten Verfeinerungsalgorithmus lassen dann eine adaptive Approximation zu, die auch auf nicht-konvexen Gebieten die optimale Ordnung wieder herstellt.

Die oben genannten Hauptpunkte werden dabei sowohl ausführlich theoretisch analysiert als auch implementiert. Es wird ein Softwarepaket mit dem Namen AVEM, angelehnt an das Paket AFEM aus [CGK$^+$10], zur Verfügung gestellt, mit dem alle in dieser Arbeit genannten numerischen Beispiele nachvollziehbar sind. Das Paket ist eine Realisierung der Virtuellen-Elemente-Methode der niedrigsten Ordnung $k = 2$ für das Stokes-Problem in MATLAB und bietet viele grundlegende Funktionen, die benötigt werden, um auch andere Virtuelle-Elemente-Methoden zu implementieren. Aufbauend auf diesem Paket lassen sich mit einigen Erweiterungen auch kompliziertere Probleme wie beispielsweise die Navier-Stokes-Gleichungen simulieren.

Die Arbeit gliedert sich dabei wie folgt: Nachdem im zweiten Kapitel die funktionalanalytischen Grundlagen zur Behandlung des Stokes-Problems gelegt wurden, beschreibt das dritte Kapitel den Aufbau der exakt divergenzfreien Virtuellen-Elemente-Methode von da Veiga, Lovadina und Vacca aus [dLV17]. Das dabei entstehende diskrete Problem wird auf eindeutige Lösbarkeit untersucht und a-priori Fehlerabschätzungen aufgestellt sowie bewiesen.

Darauf folgt Kapitel 4 über die Implementierung, in dem der Aufbau der grundlegenden Projektionen und Matrizen vorgestellt wird. Durch numerische Experimente am Ende des Abschnitts wird die Wirksamkeit der Methode aber auch die fehlende Druckrobustheit dokumentiert.

Das Kapitel 5 widmet sich der Konstruktion und dem Beweis der Verlässlichkeit des berechenbaren Fehlerschätzers. Darüber hinaus wird auch der Verfeinerungsalgorithmus besprochen, mit dem adaptive Verfeinerungen möglich sind. Numerische Experimente am Ende des Kapitels belegen das Potential des Fehlerschätzers.

Kapitel 6 befasst sich dann mit der Herstellung der Druckrobustheit. Dafür werden zuerst die erweiterten Virtuellen-Elemente-Räume vorgestellt und einige Bemerkungen zu

deren Implementierung gemacht. Anschließend behandelt das Kapitel der Rekonstruktion in Raviart-Thomas-Räume für Dreiecke und der Verallgemeinerung für Polygone, mit dessen Hilfe eine druckrobuste Methode geschaffen wird. Die numerischen Experimente belegen die theoretischen Überlegungen bezüglich der Druckrobustheit.

Das Kapitel 7 schlägt eine Brücke zu den Finiten-Elemente-Methoden und vergleicht die implementierte Virtuelle-Elemente- mit einer Finiten-Elemente-Methode mit gleicher Ordnung und mit der gleichen Anzahl an Freiheitsgraden. Es werden Vor- und Nachteile abgewägt und anhand numerischer Beispiele verifiziert.

Die Arbeit endet mit einem Ausblick noch offener Fragestellungen und zeigt Probleme auf, die es in Zukunft zu lösen gilt. Der Ausblick gibt darüber hinaus die Idee, wie die druckrobuste Form auch auf das kompliziertere Navier-Stokes-Problem anzuwenden ist.

2. Theoretische Grundlagen

Diese Arbeit beginnt damit, einige theoretische Grundlagen zu geben, die benötigt werden, um die Arbeit zu verstehen. Aus Platzgründen werden nur die für das Verständnis wichtigsten Resultate aus der Funktionalanalysis beziehungsweise der Theorie für die Numerik partieller Differentialgleichungen ohne Beweise aufgelistet. Die Inhalte sowie die Beweise lassen sich in klassischen Büchern über Funktionalanalysis beziehungsweise Numerik von Differentialgleichungen nachschlagen. Dazu zählen zum Beispiel [Eva10, Bar16, BS02], aus denen die Inhalte dieses Kapitels entnommen wurden.

Begonnen wird mit den funktionalanalytischen Grundlagen, die benötigt werden, um die Lösung partieller Differentialgleichungen zu beschreiben. Abschließend wird das Stokes-Problem eingeführt und auf seine Lösbarkeit untersucht.

2.1. Funktionalanalytische Grundlagen

Die Lösungen vieler Differentialgleichungen sind nicht mehr zwingend genügend oft stetig differenzierbar und können beispielsweise Knicke enthalten. Deshalb werden hier zunächst die benötigten Räume, die Sobolovräume, eingeführt, in denen später die Lösung des Stokes-Problems liegt. Anschließend folgen noch Rechenregeln für partielle Integration in diesen Räumen und einige wichtige Abschätzungen.

2.1.1. Schwache Ableitungen und Sobolevräume

Um nicht mehr nur stetig differenzierbare Lösungen zuzulassen, werden sogenannte schwache Ableitungen eingeführt. Die Definition der schwachen Ableitung benötigt unter anderem die Lebesgue-Räume.

Definition 2.1 (Lebesgue-Raum)**.** Sei $\Omega \subset \mathbb{R}^n$, $n \in \mathbb{N}$, und $p \in (0, \infty)$. Dann ist der Raum der p-fach integrierbaren Funktionen durch

$$L^p(\Omega) := \left\{ f : \Omega \to \mathbb{R} \,:\, f \text{ ist messbar und } \|f\|_{L^p(\Omega)} < \infty \right\}$$

definiert, wobei die L^p-Norm für eine messbare Funktion $f : \Omega \to \mathbb{R}$ durch

$$\|f\|^p_{L^p(\Omega)} := \int_\Omega |f|^p \, \mathrm{d}\Omega$$

gegeben ist.

Bemerkung 2.2. (i) Es kann mit Hilfe des wesentlichen Supremums auch der Raum L^∞ definiert werden.

(ii) Sei $f \in L^p(\Omega)$. Dann ist $f \in L^q(\Omega)$ für alle $p < q$.

(iii) Um genau zu sein, bestehen die L^p-Räume aus Äquivalenzklassen. Zwei Funktionen $f, g \in L^p(\Omega)$ sind äquivalent, wenn sie sich nur auf einer Menge mit Lebesgue-Maß Null unterscheiden. Daher sind die eigentlichen L^p-Räume die Faktorisierung der obigen Definition nach den Äquivalenzklassen.

Definition 2.3. Der Raum $L^1_{\text{loc}}(\Omega)$ besteht aus allen Funktionen, die in $L^1(\omega)$ sind für alle kompakten $\omega \subset \Omega$.

Nun können die schwachen Ableitungen definiert werden, deren Definition aus der Formel für partielle Integration motiviert ist.

Definition 2.4 (Schwache Ableitung)**.** Sei ein Multiindex $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{N}^n$ gegeben und $|\alpha| := \alpha_1 + \alpha_2 + \ldots, +\alpha_n$. Eine Funktion $g \in L^1_{\text{loc}}(\Omega)$ heißt α-te schwache Ableitung von $f \in L^1_{\text{loc}}(\Omega)$, bezeichnet mit $\mathrm{D}^\alpha f := g$, wenn

$$\int_\Omega f \frac{\partial^{|\alpha|}}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \ldots \partial x_n^{\alpha_n}} \phi \,\mathrm{d}\Omega = (-1)^{|\alpha|} \int_\Omega g\phi \,\mathrm{d}\Omega \qquad \text{für alle } \phi \in C_C^\infty(\Omega),$$

wobei C_C^∞ der Raum der unendlich oft stetig differenzierbaren Funktionen mit kompaktem Träger ist.

Lemma 2.5 (Eigenschaften schwacher Ableitungen)**.** *(i) Für alle stetig differenzierbaren Funktionen stimmen klassiche und schwache Ableitung überein.*

(ii) Hat eine Funktion eine schwache Ableitung, dann ist sie eindeutig.

Der Raum der Sobolevfunktionen besteht aus allen L^p-Funktionen, die schwache Ableitungen bis zu einem gewissen Grad haben.

Definition 2.6 (Sobolevraum)**.** Seien $k \in \mathbb{N}$ und $p \in [1, \infty]$ gegeben. Der Sobolevraum $W^{k,p}(\Omega)$ ist durch

$$W^{k,p}(\Omega) := \Big\{ f \in L^p(\Omega) \,:\, \text{Für alle } \alpha \in \mathbb{N}^n, |\alpha| \le k, \text{ existiert die schwache Ableitung } \mathrm{D}^\alpha f \text{ und } \mathrm{D}^\alpha f \in L^p(\Omega) \Big\}$$

definiert. Für $k = 0$ ist $W^{0,p}(\Omega) := L^p(\Omega)$.

Lemma 2.7 (Eigenschaften von $W^{k,p}(\Omega)$)**.** *Ausgestattet mit der Norm*

$$\|f\|^p_{W^{k,p}(\Omega)} := \sum_{|\alpha| \le k} \|\mathrm{D}^\alpha f\|^p_{L^p(\Omega)} \qquad \textit{für alle } f \in W^{k,p}(\Omega)$$

ist $(W^{k,p}(\Omega), \|\cdot\|_{W^{k,p}(\Omega)})$ *ein Banachraum, das heißt ein vollständig normierter Raum, für alle* $k \in \mathbb{N}$, $p \in [1, \infty]$.

Bemerkung 2.8. Die Seminorm $|\cdot|_{W^{k,p}(\Omega)}$ ist für alle $f \in W^{k,p}(\Omega)$ durch

$$|f|^p_{W^{k,p}(\Omega)} := \sum_{|\alpha|=k} \|\mathrm{D}^\alpha f\|^p_{L^p(\Omega)}$$

definiert.

Definition 2.9 (Sobolevraum mit Nullranddaten)**.** Der Sobolevraum mit Nullranddaten ist durch

$$W^{k,p}_0(\Omega) := \overline{C^\infty_C}^{\|\cdot\|_{W^{k,p}(\Omega)}}$$

gegeben, das heißt der Abschluss aller C^∞_c-Funktionen bezüglich der Sobolevnorm.

Bemerkung 2.10. (i) Die Funktionen $f \in W^{k,p}_0(\Omega)$ sind also Grenzwerte von Folgen unendlich oft stetig differenzierbarer Funktionen, die auf dem Rand $\partial\Omega$ identisch zur Null sind. Dafür wird auch $f_{|\partial\Omega} = 0$ geschrieben, obwohl im rigorosen Sinne die Funktionen nicht zwingend auf dem Rand definiert sind und der sogenannte Spur-Operator benötigt wird (s. [Eva10] Abschnitt 5.5).

(ii) Allgemein können mit Hilfe des Spur-Operators auch andere Randdaten zugelassen werden, in dem für eine vorgegebene Funktion $f_D : \partial\Omega \to \mathbb{R}$

$$H^1_D := \left\{ f \in H^1(\Omega) \,:\, f = f_D \text{ entlang } \partial\Omega \right\}$$

definiert wird.

Von besonderem Interesse sind außerdem die Sobolevräume $H^k(\Omega) := W^{k,2}(\Omega)$ und $H^k_0(\Omega) := W^{k,2}_0(\Omega)$ für $k \in \mathbb{N}$.

Lemma 2.11. *Die Sobolevräume* $(H^k(\Omega), \|\cdot\|_{H^k(\Omega)})$ *beziehungsweise* $(H^k_0(\Omega), \|\cdot\|_{H^k(\Omega)})$ *sind Hilberträume.*

Außerdem wird für die später eingeführte Helmholtz-Zerlegung der Raum der Funktionen mit schwacher Divergenz benötigt.

Definition 2.12. (i) Wenn für eine Funktion $\boldsymbol{f} \in [L^p(\Omega)]^n$ eine Funktion $g \in L^1_{\mathrm{loc}}(\Omega)$ existiert, so dass

$$-\int_\Omega \boldsymbol{f} \cdot \mathrm{D}\phi \,\mathrm{d}\Omega = \int_\Omega g\phi \,\mathrm{d}\Omega \qquad \text{für alle } \phi \in C^\infty_C$$

gilt, dann heißt g schwache Divergenz, geschrieben $\mathrm{div}\boldsymbol{f} = g$, von $\boldsymbol{f}$.

(ii) Der Raum der Vektorfelder mit schwacher Divergenz ist durch

$$H(\mathrm{div}, \Omega; \mathbb{R}^n) := \left\{ f \in [L^2(\Omega)]^n \,:\, \mathrm{div}\boldsymbol{f} \in L^2(\Omega) \right\}$$

definiert, wobei die Divergenz im schwachen Sinne zu verstehen ist.

2.1.2. Partielle Integration und wichtige Abschätzungen

In den klassischen Funktionenräumen mit stetig differenzierbaren Funktionen gelten die aus der Analysis bekannten Regeln für die partielle Integration. Dadurch, dass die unendlich oft stetig differenzierbaren Funktionen dicht in den Sobolevräumen liegen, übertragen sich die Regeln auch auf die Sobolevräume.

Satz 2.13 (Partielle Integration / Satz von Gauß)**.** *Für alle $\boldsymbol{f} \in H(\mathrm{div}, \Omega; \mathbb{R}^n)$ und $g \in H^1(\Omega)$ gelten die folgenden Regeln für partielle Integration:*

$$(i) \qquad \int_\Omega \mathrm{div}\boldsymbol{f}\,\mathrm{d}\Omega = \int_{\partial\Omega} \boldsymbol{f}\cdot\boldsymbol{n}\,\mathrm{d}s$$

$$(ii) \qquad \int_\Omega \mathrm{div}\boldsymbol{f}\,g\,\mathrm{d}\Omega + \int_\Omega \boldsymbol{f}\cdot\mathrm{D}g\,\mathrm{d}\Omega = \int_{\partial\Omega} \boldsymbol{f}\cdot\boldsymbol{n}\,g\,\mathrm{d}s$$

Hierbei ist $\boldsymbol{n}$ der äußere Einheits-Normalenvektor zu $\partial\Omega$.

Bemerkung 2.14. (i) Die obigen Randintegrale müssen im Spursinne verstanden werden.

(ii) Die obige Formel kann als Motivation für die Definition der schwachen Divergenz gesehen werden.

Abschließend folgen noch einige wichtige Ungleichungen.

Lemma 2.15 (Wichtige Ungleichungen)**.** *Sei $\Omega \subset \mathbb{R}^n$, $n \in \mathbb{N}$, ein beschränktes Lipschitz-Gebiet, das heißt der Rand $\partial\Omega$ ist lokal der Graph einer Lipschitz-stetigen Funktion. Dann gelten für alle $f \in W^{1,p}(\Omega)$ die folgenden Ungleichungen:*

(i) Poincaré-Ungleichung:

$$\|f - \frac{1}{|\Omega|}\int_\Omega f\,\mathrm{d}\Omega\|_{L^p(\Omega)} \le C_P \|\mathrm{D}f\|_{L^p(\Omega)},$$

wobei $0 < C_P < \infty$ vom Durchmesser von Ω und p abhängt.

(ii) Payne-Weinberger-Ungleichung:

$$\|f - \frac{1}{|\Omega|}\int_\Omega f\,\mathrm{d}\Omega\|_{L^p(\Omega)} \le \frac{h_\Omega}{\pi} \|\mathrm{D}f\|_{L^p(\Omega)},$$

wobei $h_\Omega := \mathrm{diam}\Omega := \sup_{\boldsymbol{x},\boldsymbol{y}\in\Omega} |\boldsymbol{x}-\boldsymbol{y}|$ ist.

(iii) Friedrichs-Ungleichung: Hat $\Gamma_D \subset \partial\Omega$ ein positives $n-1$ dimensionales Flächenmaß und ist $f \in W^{1,p}_{D,0} := \left\{ f \in W^{1,p}(\Omega) \,:\, f_{|\Gamma_D} = 0 \right\}$, dann gilt

$$\|f\|_{L^p(\Omega)} \le C_{\mathrm{Fr}} \|\mathrm{D}f\|_{L^p(\Omega)},$$

wobei $0 < C_{\mathrm{Fr}} < \infty$ von Ω, Γ_D und p abhängt.

Bemerkung 2.16. Wegen der Friedrichs-Ungleichung gilt, dass

$$\|f\|_{H^1(\Omega)} \approx |f|_{H^1(\Omega)} \qquad \text{für alle } f \in H^1_0(\Omega).$$

2.2. Kontinuierliches Stokes-Problem

Nach der Einführung der Sobolevräume kann mit der Vorstellung des Stokes-Problems und seiner Lösungstheorie begonnen werden.

Das Stokes-Problem ist eine Vereinfachung des Navier-Stokes-Problems und beschreibt das Verhalten beziehungsweise die Bewegung von inkompressiblen newtonschen Fluiden wie beispielsweise Wasser, Silikonöl oder auch Luftströmungen [GDN98]. Die gesuchten Größen sind dabei die Geschwindigkeit und der Druck des Fluids.

2.2.1. Beschreibung des Stokes-Problems

Sei $\Omega \subset \mathbb{R}^n$, $n \in \mathbb{N}$, ein polygonal umrandetes Lipschitz-Gebiet. Für eine vektorwertige Kraft $\boldsymbol{f} \in [L^2(\Omega)]^n$ und eine auf Ω konstante Viskosität $\nu \in \mathbb{P}_0(\Omega)$ liest sich das inhomogene *Stokes-Problem* wie folgt: Finde ein Paar $(\boldsymbol{u}, p)$ bestehend aus einer Geschwindigkeit $\boldsymbol{u}$ und einem skalaren Druck p, so dass

$$
\begin{aligned}
-\nu\Delta\boldsymbol{u} + \nabla p &= \boldsymbol{f} && \text{in } \Omega, && \text{(2.1a)}\\
\operatorname{div}\boldsymbol{u} &= 0 && \text{in } \Omega, && \text{(2.1b)}\\
\boldsymbol{u} &= 0 && \text{entlang } \partial\Omega && \text{(2.1c)}
\end{aligned}
$$

gelten. Der Laplaceoperator Δ in Gleichung (2.1a) muss dabei vektorwertig verstanden werden, das heißt angewandt auf jede Komponente von $\boldsymbol{u}$.

Im Allgemeinen ist es nicht möglich, eine starke Lösung zu finden, das heißt eine genügend oft stetig differenzierbares Paar, das die obigen Gleichungen erfüllt. Der benötigte Lösungsbegriff sind die schwachen Lösungen, die in Sobolevräumen liegen.

Um die schwache Formulierung zu erhalten, werden Gleichungen (2.1a) und (2.1b) mit einer Testfunktion

$$
\begin{aligned}
&v \in \boldsymbol{V} := [H_0^1(\Omega)]^n \text{ bzw.}\\
&q \in Q := L_0^2(\Omega) := \left\{ q \in L^2(\Omega) \,:\, \int_\Omega f \,\mathrm{d}\Omega = 0 \right\}
\end{aligned}
$$

multipliziert und über das Gebiet Ω integriert. Daraus ergibt sich

$$
\begin{aligned}
-\int_\Omega \nu\Delta\boldsymbol{u}\cdot\boldsymbol{v} + \nabla p\cdot\boldsymbol{v}\,\mathrm{d}\Omega &= \int_\Omega \boldsymbol{f}\cdot\boldsymbol{v}\,\mathrm{d}\Omega,\\
\int_\Omega \operatorname{div}\boldsymbol{u}\, q\,\mathrm{d}\Omega &= 0,
\end{aligned}
$$

was wegen $v \in \boldsymbol{V}$ nun in jeder Komponente nach Satz 2.13 partiell integriert werden kann. Unter Ausnutzung von Gleichung (2.1c) für den Randterm folgt

$$
\begin{aligned}
\int_\Omega \nu\mathrm{D}\boldsymbol{u} : \mathrm{D}\boldsymbol{v} - \operatorname{div}\boldsymbol{v}\, p\,\mathrm{d}\Omega &= \int_\Omega \boldsymbol{f}\cdot\boldsymbol{v}\,\mathrm{d}\Omega,\\
\int_\Omega \operatorname{div}\boldsymbol{u}\, q\,\mathrm{d}\Omega &= 0,
\end{aligned}
$$

wobei D die Jacobi-Matrix ist und der Doppelpunkt : das die Frobeniusnorm induzierende Skalarprodukt in $\mathbb{R}^{n\times n}$ bezeichnet. Das heißt

$$A : B := \sum_{j,k=1}^{n} A_{j,k}B_{j,k} \qquad \text{für zwei Matrizen } A, B \in \mathbb{R}^{n\times n}.$$

Damit ergibt sich die *schwache Form des Stokes-Problems*: Finde $(\boldsymbol{u}, p) \in \boldsymbol{V} \times Q$, so dass

$$a(\boldsymbol{u}, \boldsymbol{v}) + b(\boldsymbol{v}, p) = (\boldsymbol{f}, \boldsymbol{v})_{L^2(\Omega)} \qquad \text{für alle } \boldsymbol{v} \in \boldsymbol{V}, \tag{2.2a}$$

$$b(\boldsymbol{u}, q) = 0 \qquad \text{für alle } q \in Q \tag{2.2b}$$

gelten.

Hierbei ist $(\cdot,\cdot)_{L^2(\Omega)}$ das gewöhnliche L^2-Skalarprodukt. Außerdem sind die Bilinearformen $a(\cdot,\cdot)$ und $b(\cdot,\cdot)$ für alle $\boldsymbol{v}$, $\boldsymbol{w} \in \boldsymbol{V}$ und alle $q \in Q$ mittels

$$a(\boldsymbol{v}, \boldsymbol{w}) := \int_\Omega \nu \mathrm{D}\boldsymbol{v} : \mathrm{D}\boldsymbol{w} \,\mathrm{d}\Omega \tag{2.3a}$$

$$b(\boldsymbol{v}, q) := -\int_\Omega \mathrm{div}\boldsymbol{v}\, q \,\mathrm{d}\Omega \tag{2.3b}$$

definiert.

Bemerkung 2.17. Es können auch inhomogene Dirichletrandbedingungen zugelassen werden, das heißt

$$\boldsymbol{u} = \boldsymbol{u}_D \qquad \text{entlang } \partial\Omega$$

für eine Funktion $\boldsymbol{u}_D : \partial\Omega \to \mathbb{R}^n$. In diesem Fall wird ein $\boldsymbol{u} \in [H^1_D(\Omega)]^n$ gesucht, so dass die obige schwache Formulierung für alle $\boldsymbol{v} \in [H^1_D(\Omega)]^n$ gilt.

2.2.2. Wohldefiniertheit des kontinuierlichen Problems

Die obige schwache Formulierung ist ein Standardbeispiel für ein gemischtes Problem. Ein gemischtes Problem sucht ein Paar $(x_1, y_1) \in X \times Y$, so dass

$$a(x_1, x_2) + b(x_2, y_1) = F(x_2) \qquad \text{für alle } x_2 \in X,$$

$$b(x_1, y_2) = G(y_2) \qquad \text{für alle } y_2 \in Y,$$

wobei $F \in V^*$ und $G \in Q^*$ Elemente der jeweiligen Dualräume sind. Im Kontext des Stokes-Problems sind $X = \boldsymbol{V}$, $Y = Q$, $F := (\boldsymbol{f}, \cdot)$, $G = 0$ und $a(\cdot,\cdot)$ sowie $b(\cdot,\cdot)$ aus Gleichungen (2.3a) und (2.3b).

Die Eindeutigkeit schwacher Lösungen folgt aus dem folgenden Satz.

Satz 2.18 (Brezzis Zerlegungssatz)**.** *Seien* $(X, \|\cdot\|_X)$, $(Y, \|\cdot\|_Y)$ *zwei Hilberträume,* $a : X \times X \to \mathbb{R}$ *eine symmetrische, beschränkte und positiv semidefinite Bilinearform*

und $b : X \times Y \to \mathbb{R}$ eine beschränkte Bilinearform. Der Operator $L : X \times Y \to X^* \times Y^*$ definiert durch

$$(x, y) \mapsto (a(x, \cdot_X) + b(\cdot_X, y), b(x, \cdot_Y)),$$

wobei $\cdot_X$ beziehungsweise $\cdot_Y$ Platzhalter für Argumente aus X beziehungsweise Y sind, ist ein Isomorphismus genau dann, wenn die zwei folgenden Bedingungen erfüllt sind:

(i) Es existiert ein $0 < \alpha$, so dass

$$a(x, x) \geq \alpha \|x\|_X^2 \qquad \text{für alle } x \in Z := \{ x \in X : b(x, \cdot) = 0 \in Y^* \},$$

(ii) b erfüllt die inf-sup-Bedingung, das heißt es existiert ein $0 < \beta$, so dass

$$\beta \leq \inf_{y \in Y \setminus \{0\}} \sup_{x \in X \setminus \{0\}} \frac{b(x, y)}{\|x\|_X \|y\|_Y}.$$

Beweis. Siehe [BF91] in Kapitel II, Abschnitt 1.2 oder [Bar16] auf Seite 265. □

Um die eindeutige Existenz einer schwachen Lösung für das Stokes-Problem nachzuweisen, müssen somit die Voraussetzung aus dem vorherigen Satz überprüft werden. Die Herausforderung ist der Nachweis der inf-sup-Bedingung (ii). Zum Beweis der inf-sup-Bedingung wird der folgende Satz benötigt.

Satz 2.19 (Ladyzhenskaya)**.** *Für ein beschränktes Lipschitz-Gebiet Ω existiert eine Konstante $C(\Omega)$, so dass für alle $q \in L_0^2(\Omega)$ ein $\boldsymbol{v}_q \in [H_0^1(\Omega)]^n$ existiert, dass*

$$\operatorname{div} \boldsymbol{v}_q = q \qquad \text{und} \qquad \|\boldsymbol{v}_q\|_{H^1(\Omega)} \leq C(\Omega) \|q\|_{L^2(\Omega)}$$

erfüllt.

Beweis. Siehe zum Beispiel Korolar 2.4 2°) in [GR86] auf Seite 24. □

Der obige Satz kann nun auf das Stokes-Problem angewendet werden.

Satz 2.20. *Für die schwache Formulierung des Stokes-Problems existiert eine eindeutige Lösung.*

Beweis. Es werden die Voraussetzung aus Satz 2.18 geprüft. Dafür sind wie oben angesprochen $X = \boldsymbol{V}$, $Y = Q$, $F := (\boldsymbol{f}, \cdot)$, $G = 0$ und $a(\cdot, \cdot)$ sowie $b(\cdot, \cdot)$ aus Gleichungen (2.3a) und (2.3b).

Die Bilinearform $a(\cdot, \cdot)$ ist nach Definition symmetrisch und positiv semidefinit. Die Beschränktheit folgt aus der Cauchy-Schwarz-Ungleichung und $\boldsymbol{u}, \boldsymbol{v} \in \boldsymbol{V}$.

Wegen der Normäquivalenz von $\|\cdot\|_1$ und $\|\cdot\|_2$ in $\mathbb{R}^n$ folgt für alle $\boldsymbol{v} \in \boldsymbol{V}$, dass

$$\begin{aligned} \|\operatorname{div} \boldsymbol{v}\|_{L^2(\Omega)}^2 &= \int_\Omega \left(\sum_{j=1}^n \frac{\partial u_j}{\partial x_j} \right)^2 \mathrm{d}\Omega \lesssim \int_\Omega \sum_{j=1}^n \left(\frac{\partial u_j}{\partial x_j} \right)^2 \mathrm{d}\Omega \leq \int_\Omega \sum_{j,k=1}^n \left(\frac{\partial u_j}{\partial x_k} \right)^2 \mathrm{d}\Omega \\ &= \|\mathrm{D}\boldsymbol{v}\|_{L^2(\Omega)}^2 \leq \|\boldsymbol{v}\|_{H^1(\Omega)}^2, \end{aligned}$$

womit mit einer erneuten Cauchy-Schwarz-Ungleichung auch die Beschränktheit von $b(\cdot,\cdot)$ gefolgert werden kann.

Die Elliptizität (i) folgt aus der Friedrichs-Ungleichung.

Zum Nachweis der inf-sup-Bedingung (ii) sei ein $q \in Q$ fixiert. Aus Satz 2.19 folgt nun die Existenz einer Funktion $\boldsymbol{v}_q \in \boldsymbol{V}$, so dass $\operatorname{div}\boldsymbol{v}_q = -q$ und $\|\boldsymbol{v}_q\|_{H^1(\Omega)} \leq C(\Omega)\|q\|_{L^2(\Omega)}$. Damit lässt sich

$$\sup_{\boldsymbol{v}\in\boldsymbol{V}\setminus\{0\}} \frac{-\int_\Omega \operatorname{div}\boldsymbol{v}\, q \,\mathrm{d}\Omega}{\|\boldsymbol{v}\|_{H^1(\Omega)}\|q\|_{L^2(\Omega)}} \geq \frac{-\int_\Omega \operatorname{div}\boldsymbol{v}_q\, q \,\mathrm{d}\Omega}{\|\boldsymbol{v_q}\|_{H^1(\Omega)}\|q\|_{L^2(\Omega)}} \gtrsim \frac{\int_\Omega q^2 \,\mathrm{d}\Omega}{\|q\|^2_{L^2(\Omega)}} = 1$$

folgern. Da $q \in Q$ beliebig war, gilt die obige Überlegung auch für das Infimum.

Damit sind alle Voraussetzung aus Satz 2.19 erfüllt und das Stokes-Problem besitzt eine eindeutige Lösung $(\boldsymbol{u}, p) \in \boldsymbol{V} \times Q$.

□

Da nur in wenigen Fällen, das heißt für „einfache“ rechte Seiten die exakte schwache Lösung $\boldsymbol{u}, p$ analytisch bestimmt werden kann, gilt es, eine Näherungslösung zu finden, welche die Lösung möglichst gut approximiert. Mit einer möglichen Methode dafür und der Qualität der daraus resultierenden Approximation beschäftigt sich das folgende Kapitel.

3. Die Virtuelle-Elemente-Methode für das Stokes-Problem

In diesem Kapitel wird die in dieser Arbeit untersuchte Virtuelle-Elemente-Methode zur Approximation des Stokes-Problems in $\mathbb{R}^2$ beschrieben. Diese Methode sowie die a-priori Fehleranalyse beruhen auf der Veröffentlichung von da Veiga, Lovadina und Vacca [dLV17]. Daher basiert auch dieses Kapitel und ein Großteil der Beweise auf der vorher genannten Referenz. Dafür werden zuerst die Virtuellen-Elemente-Räume und das diskrete Problem eingeführt, welches dann auf eine eindeutige Lösung untersucht wird. Anschließend folgt ein Abschnitt über die Implementierung der Virtuellen-Elemente-Methode und einige numerische Beispiele.

3.1. Voraussetzungen zur Formulierung der Virtuellen-Elemente-Methode

Um zu einer diskreten Formulierung zu gelangen, muss eine endliche Teilmenge der Räume $\boldsymbol{V}$ und Q gewählt werden. Dafür wird zunächst das Gebiet $\Omega \subset \mathbb{R}^2$ selbst diskretisiert, das heißt in eine endliche Zahl kleinerer Bereiche unterteilt. Im Kontext der Finiten-Elemente-Methoden sind diese kleineren Strukturen in der Regel Dreiecke oder auch Rechtecke [BS02]. Ein Hauptvorteil der Virtuellen-Elemente-Methoden liegt darin, andere nahezu beliebige Polygone zuzulassen.

3.1.1. Formregularität des Gebiets

Sei von nun an $(\mathcal{T}_h)_h$ eine Folge endlicher Zerlegungen von Ω in Polygone T_j mit positivem Flächeninhalt, so dass

$$\bigcup_{T_j \in \mathcal{T}_h} \overline{T_j} = \overline{\Omega}$$

gilt. Der Übersichtlichkeit wegen wird der Index j in Zukunft für die Polygone weggelassen, wenn keine Verwechslung auftreten kann. Für das Polygon $T \in \mathcal{T}_h$ wird mit $h_T \in \mathbb{R}$ der Durchmesser bezeichnet, also

$$h_T := \sup_{\boldsymbol{x},\boldsymbol{y} \in T} |\boldsymbol{x} - \boldsymbol{y}|.$$

Die Größen $h \in \mathbb{R}$ beziehungsweise $h_{\mathcal{T}} \in \mathbb{P}_0(\mathcal{T}_h)$ werden mittels

$$h := \sup_{T \in \mathcal{T}_h} h_T \qquad \text{bzw.} \qquad h_{\mathcal{T}|T} := h_T$$

definiert. Hierbei ist

$$\mathbb{P}_s(\mathcal{T}_h) := \{\, q_s \in L^2(\Omega) \,:\, q_{s|T} \text{ ist ein Polynom vom Grad maximal } s \text{ für alle } T \in \mathcal{T}_h \,\}$$

für ein $s \in \mathbb{N}$. Außerdem soll mit $\boldsymbol{x}_T$ und $|T|$ der Schwerpunkt beziehungsweise der Flächeninhalt, also das Maß des Polygons betitelt werden.

Leider kann nicht jegliche Art von Polygonen zur Diskretisierung genutzt werden. Für die Konvergenz der Methode muss vorausgesetzt werden, dass das Gitter formregulär (engl. "shape regular") ist, was im Folgenden definiert wird.

Definition 3.1 (Formregularität)**.** Eine Folge endlicher Zerlegungen $(\mathcal{T}_h)_h$ des Gebiets Ω heißt formregulär, wenn zwei positive Konstanten γ und c existieren, so dass für alle h und jedes $T \in \mathcal{T}_h$ die folgenden Bedingungen erfüllt sind:

(A1) T ist ein einfaches Polygon, das heißt eine offene, einfach zusammenhängende Menge, deren Rand eine nicht sich selbst schneidende Linie ist, die aus einer endlichen Anzahl von Strecken besteht,

(A2) T ist eine sternförmige Menge in Bezug auf einen Ball mit einem Radius, der größer oder gleich γh_T ist,

(A3) der Abstand zwischen zwei verschiedenen Knoten von T ist größer oder gleich ch_T.

Bemerkung 3.2. Im Gegensatz zu Finiten-Elemente-Methoden, in denen hängende Knoten, das heißt Knoten mit einem Innenwinkel von 180°, verboten sind, sind solche Knoten bei den Virtuellen-Elemente-Methoden gestattet. Ein Polygon in Form eines Dreiecks mit einem hängenden Knoten wird formal als Viereck betrachtet. Außerdem sind auch nichtkonvexe Polygone möglich. Beides zusammen erlaubt also eine viel flexiblere Zerlegung des Gebiets als es bei den meisten Finiten-Elemente-Methoden der Fall ist [dBC+13].

Eine Illustration solcher sternförmiger Mengen ist bspw. in [BS02] in Abschnitt 4.2 zu finden. Weiterhin werden die üblichen Bezeichnungen für die Bestandteile einer Triangulierung genutzt. Mit $\mathcal{E}$ wird die Menge aller Kanten und mit $\mathcal{N}$ die Menge aller Knoten der Zerlegung bezeichnet. Analog ist für ein Polygon $T \in \mathcal{T}_h$ die Menge seiner Kanten durch $\mathcal{E}(T)$ und die Menge seiner Knoten durch $\mathcal{N}(T)$ gegeben.

3.1.2. Benötigte Polynomräume und eine Zerlegung

Um die lokalen Räume zu definieren werden einige andere Räume benötigt. Sei dafür $T \in \mathcal{T}_h$ wieder ein beliebiges aber festes Polygon. Es werden mit

- $\mathbb{P}_s(T)$ der Raum aller Polynome vom Grad kleiner oder gleich $s \in \mathbb{N}$,
- $\mathbb{B}_s(T) := \{\, v \in C^0(\partial T) \,:\, v_{|E} \in \mathbb{P}_s(E) \text{ für alle Kanten } E \in \mathcal{E}(T) \,\}$ der Raum der Polynome auf dem Rand,
- $\mathcal{G}_s(T) := \nabla \mathbb{P}_{s+1}(T) \subset [\mathbb{P}_s(T)]^2$ der Gradientenraum der Polynome vom Grad kleiner oder gleich s und mit

- $\mathcal{K}_s(T) := \left\{ k \in \mathbb{P}_s(T)^2 \,:\, k = \begin{pmatrix} x_2 \\ -x_1 \end{pmatrix} p_{s-1} \text{ für ein } p_{s-1} \in \mathbb{P}_{s-1}(T) \right\} \subset [\mathbb{P}_s(T)]^2$ ein zu $\mathcal{G}_s(T)$ komplementärer Unterraum

bezeichnet. Für die beiden letzten Räume gilt, dass sie eine Zerlegung des ersten Raums sind, das heißt es gilt

$$[\mathbb{P}_s(T)]^2 = \mathcal{G}_s(T) \oplus \mathcal{K}_s(T), \tag{3.1}$$

wobei $\oplus$ die direkte Summe bezeichnet [DV19].

Es werden noch einige weitere Bezeichnungen beziehungsweise Definitionen benötigt. Sei mit $\pi_s := \dim \mathbb{P}_s(\mathbb{R}^2) = (s+1)(s+2)/2$, $s \in \mathbb{N}$, die Dimension des Vektorraums der Polynome vom Grad kleiner oder gleich s auf $\mathbb{R}^2$ bezeichnet. Für einen beliebigen Multiindex $\boldsymbol{\alpha} := (\alpha_1, \alpha_2) \in \mathbb{N}^2$ wird der Betrag $|\boldsymbol{\alpha}| := \alpha_1 + \alpha_2$ und für $\boldsymbol{x} \in \mathbb{R}^2$ die Potenz $\boldsymbol{x}^{\boldsymbol{\alpha}} := x_1^{\alpha_1} x_2^{\alpha_2}$ festgelegt.

Mit dieser Notation kann nun der Raum der skalierten Monome eingeführt werden.

Definition 3.3 (Raum der skalierten Monome)**.** Sei $\boldsymbol{x}_T$ das Baryzentrum von $T \in \mathcal{T}_h$ und h_T der Durchmesser. Mit

$$m_{\boldsymbol{\alpha}} := \left(\frac{\boldsymbol{x} - \boldsymbol{x}_T}{h_T} \right)^{\boldsymbol{\alpha}}$$

ist der Raum der skalierten Monome vom Grad kleiner oder gleich $s \in \mathbb{N}$ durch

$$\mathbb{M}_s(T) := \{\, m_{\boldsymbol{\alpha}} \in \mathbb{P}_s(T) \,:\, 0 \leq |\boldsymbol{\alpha}| \leq s \,\}$$

gegeben.

Bemerkung 3.4. Der Raum der skalierten Monome vom Grad kleiner oder gleich s ist eine Basis für $\mathbb{P}_s(T)$.

Beweis. Dies folgt genau wie für den Fall der Standardbasis des Polynomraums. Eine Linearkombination der skalierten Monome ist ein Polynom vom Grad kleiner oder gleich s und kann daher nur s Nullstellen haben.

Soll die Kombination aber mehr als s Nullstellen haben, dann muss das Polynom selbst schon das Null-Polynom gewesen sein und daher sind die Koeffizienten der Linearkombination gleich Null. Somit sind die skalierten Monome linear unabhängig.

Das Zählen der Anzahl der skalierten Monome und Vergleich mit der Dimension von $\mathbb{P}_s(T)$ beenden den Beweis. □

Für die später beschriebene Implementierung lohnt es sich, die Elemente anzuordnen. Dafür wird die folgende Korrespondenz zwischen eindimensionalen und zweidimensionalen Indizes benutzt:

$$1 \leftrightarrow (0,0), \quad 2 \leftrightarrow (1,0), \quad 3 \leftrightarrow (0,1), \quad 4 \leftrightarrow (2,0), \quad 5 \leftrightarrow (1,1), \quad 6 \leftrightarrow (0,2), \quad \ldots \tag{3.2}$$

Tabelle 3.1.: Zusammengefasst sind die verschiedenen Schreibweisen für die skalierten Monome.

$m_\emptyset$	-	die skalare, konstante 0
$m_{\boldsymbol{\alpha}}$	-	skalarwertiges skaliertes Monom mit Multiindex $\boldsymbol{\alpha}$
m_j	-	das j-te skalarwertige skalierte Monom
$\boldsymbol{m}_{\boldsymbol{\alpha},\boldsymbol{\beta}}$	-	vektorwertiges skaliertes Monom mit Multiindices $\boldsymbol{\alpha}, \boldsymbol{\beta}$
$\boldsymbol{m}_j$	-	das j-te vektorwertige skalierte Monom

Damit können die skalierten Monome auch als

$$\mathbb{M}_s(T) = \{\, m_j \in \mathbb{P}_s(T) \,:\, 1 \leq j \leq \pi_s \,\}$$

geschrieben werden. Darüber hinaus können außerdem vektorwertige Monome definiert werden. Seien dafür $\boldsymbol{\alpha} = (\alpha_1, \alpha_2)$ und $\boldsymbol{\beta} = (\beta_1, \beta_2)$ zwei Multiindices. Das daraus resultierende vektorwärtige skalierte Monom ergibt sich dann durch

$$\boldsymbol{m}_{\boldsymbol{\alpha},\boldsymbol{\beta}} := \begin{pmatrix} m_{\boldsymbol{\alpha}} \\ m_{\boldsymbol{\beta}} \end{pmatrix}.$$

Mit der erweiterten Definition $m_\emptyset := 0$ kann

$$[\mathbb{M}_s(T)]^2 := \{\, \boldsymbol{m}_{\boldsymbol{\alpha},\emptyset} \,:\, 0 \leq |\boldsymbol{\alpha}| \leq s \,\} \cup \{\, \boldsymbol{m}_{\emptyset,\boldsymbol{\beta}} \,:\, 0 \leq |\boldsymbol{\beta}| \leq s \,\}$$

eingeführt werden. Dies ist dann wegen Bemerkung 3.4 eine Basis für $[\mathbb{P}_s(T)]^2$. Auch die Elemente der Menge $[\mathbb{M}_s(T)]^2$ müssen später durchnummeriert werden. Dafür wird die obige Korrespondenz für vektorwertige Monome erweitert:

$$1 \leftrightarrow (1, \emptyset), \quad 2 \leftrightarrow (\emptyset, 1), \quad 3 \leftrightarrow (2, \emptyset), \quad 4 \leftrightarrow (\emptyset, 2), \quad \ldots \tag{3.3}$$

Damit ergibt sich

$$[\mathbb{M}_s(T)]^2 = \{\, \boldsymbol{m}_j \,:\, 1 \leq j \leq 2\pi_s \,\}.$$

Es wird zwar sowohl im skalaren als auch im vektorwertigen Fall der Index j benutzt, um die Monome anzuordnen, allerdings ist im zweiten Fall das vektorwertige Monom $\boldsymbol{m}_j$ fett geschrieben. Das geschieht im ersten Fall nicht. Ist der Index auch fett gedruckt, dann handelt es sich um einen Multiindex, was allerdings nur für skalare Monome auftreten kann. Die unterschiedlichen Schreibweisen sind in Tabelle 3.1 zusammengefasst.

Für die nun folgende Zerlegung wird außerdem noch

$$\boldsymbol{m}^\perp := \begin{pmatrix} m_{(0,1)} \\ -m_{(1,0)} \end{pmatrix}$$

benötigt. Wie in Gleichung (3.1) schon geschrieben, lässt sich nämlich jedes Polynom $\boldsymbol{p}_s \in [\mathbb{P}_s(T)]^2$ eindeutig in die Summe eines Elements aus $\mathcal{G}_s(T)$ und eines Elements aus

$\mathcal{K}_s(T)$ zerlegen, das heißt es existiert genau ein $\boldsymbol{g}_s \in \mathcal{G}_s(T)$ und genau ein $\boldsymbol{k}_s \in \mathcal{K}_s(T)$, so dass

$$\boldsymbol{p}_s = \boldsymbol{g}_s + \boldsymbol{k}_s$$

gilt. Äquivalent dazu ist, dass genau ein Polynom $p_{s+1} \in \mathbb{P}_{s+1}(T)/\mathbb{R}$ und genau ein $p_{s-1} \in \mathbb{P}_{s-1}(T)$ existieren, so dass

$$\boldsymbol{p}_s = \nabla p_{s+1} + \begin{pmatrix} x_2 \\ -x_1 \end{pmatrix} p_{s-1}.$$

Im Allgemeinen lässt sich die Zerlegung nicht so einfach bestimmen, aber mit Hilfe der skalierten Monome lässt sich wie in [DV19] eine einfach Formel angeben. Da sie eine Basis für $[\mathbb{P}_s(T)]^2$ bilden wird die folgende Zerlegung nur für die Basisfunktionen aufgeschrieben.

Lemma 3.5. *Die Basisfunktionen der vektorwertigen skalierten Monome $\boldsymbol{m}_{\boldsymbol{\alpha},\emptyset}$ und $\boldsymbol{m}_{\emptyset,\boldsymbol{\beta}}$ mit $\alpha = (\alpha_1, \alpha_2) \in \mathbb{N}^2$ und $\beta = (\beta_1, \beta_2) \in \mathbb{N}^2$ lassen sich eindeutig in*

$$\boldsymbol{m}_{\boldsymbol{\alpha},\emptyset} = \frac{h_T}{|\alpha|+1} \nabla m_{(\alpha_1+1,\alpha_2)} + \frac{\alpha_2}{|\alpha|+1} \boldsymbol{m}^{\perp} m_{(\alpha_1,\alpha_2-1)}, \tag{3.4}$$

$$\boldsymbol{m}_{\emptyset,\boldsymbol{\beta}} = \frac{h_T}{|\beta|+1} \nabla m_{(\beta_1,\beta_2+1)} - \frac{\beta_1}{|\beta|+1} \boldsymbol{m}^{\perp} m_{(\beta_1-1,\beta_2)} \tag{3.5}$$

zerlegen.

Beweis. Ausnutzen der Definition der skalierten Monome und einfaches Nachrechnen zeigen die Behauptung. □

Bemerkung 3.6. Das zeigt im Übrigen auch die Zerlegung aus Gleichung (3.1).

3.2. Virtuelle-Elemente-Räume und das diskrete Problem

In diesem Abschnitt sollen die Virtuellen-Elemente-Räume und die diskreten Bilinearformen definiert werden, um ein diskretes Problem festlegen zu können.

3.2.1. Formulierung des diskreten Problems in den Virtuellen-Elemente-Räumen

Im vorherigen Abschnitt wurden alle Räume eingeführt, die benötigt werden, um die Virtuellen-Elemente-Räume zu definieren. Dafür wird ein $k \in \mathbb{N}$ fixiert, das später die Ordnung der Konvergenz der Methode sein wird.

Definition 3.7 (Lokale Räume)**.** Die lokalen Räume für ein Polygon $T \in \mathcal{T}_h$ werden durch

$$\boldsymbol{V}_k^T := \Big\{ \boldsymbol{v}_h \in [H^1(T)]^2 \,:\, \boldsymbol{v}_{h|\partial T} \in [\mathbb{B}_k(T)]^2,\, -\nu\Delta\boldsymbol{v}_h + \nabla s \in \mathcal{K}_{k-2}(T) \text{ für ein } s \in L^2(T),\, \operatorname{div}\boldsymbol{v}_h \in \mathbb{P}_{k-1}(T) \Big\}$$

und

$$Q_k^T := \mathbb{P}_{k-1}(T)$$

definiert.

Es ist wichtig an dieser Stelle zu betonen, dass die in der obigen Definition auftauchenden Operatoren und Gleichungen in schwacher Form zu verstehen sind. Ebenfalls kann bemerkt werden, dass $[\mathbb{P}_k(T)]^2 \subset \boldsymbol{V}_k^T$ gilt, was später die richtige Konvergenzordung sichern wird.

Seien die Polynome $\boldsymbol{f}_b \in \mathbb{B}_s(T)$, $g \in \mathbb{P}_{k-1}$ und $\boldsymbol{k} \in \mathcal{K}_{k-2}(T)$ gegeben, so dass

$$\int_T g \,\mathrm{d}T = \int_{\partial T} \boldsymbol{f}_b \cdot \boldsymbol{n} \,\mathrm{d}s$$

mit dem äußeren Normalenvektor $\boldsymbol{n}$ gilt. In [dLV17] wird detailliert begründet, dass es eine injektive Abbildung von einem Datensatz $(\boldsymbol{f}_b, g, \boldsymbol{k}) \in [\mathbb{B}_s(T)]^2 \times \mathbb{P}_{k-1} \times \mathcal{K}_{k-2}(T)$ nach $\boldsymbol{v} \in \boldsymbol{V}_k^T$ gibt, wobei $(\boldsymbol{v}, s) \in \boldsymbol{V}_k^T \times L^2(T)/\mathbb{R}$ die eindeutige Lösung zu

$$-\nu\Delta\boldsymbol{v} + \nabla s = \boldsymbol{k} \qquad \text{in } T, \tag{3.6}$$

$$\mathrm{div}\boldsymbol{v} = g \qquad \text{in } T, \tag{3.7}$$

$$\boldsymbol{v} = \boldsymbol{f}_b \qquad \text{auf } \partial T \tag{3.8}$$

ist. Dies beruht im Wesentlichen darauf, dass der Rotationsoperator $\mathrm{rot} : \mathcal{K}_{k-2} \to \mathbb{P}_{k-3}$ ein Isomorphismus ist [dBMR14a].

Die Dimensionen sind daher

$$\begin{aligned}\dim \boldsymbol{V}_k^T =& \dim [\mathbb{B}_k(T)]^2 + \dim \mathcal{K}_{k-2}(T) + \dim \mathbb{P}_{k-1}(T) - 1\\ =& 2|\mathcal{N}(T)|k + \frac{(k-1)(k-2)}{2} + \frac{(k+1)k}{2} - 1\end{aligned}$$

und

$$\dim Q_k^T = \dim \mathbb{P}_{k-1}(T) = \frac{k(k+1)}{2},$$

wobei $|\mathcal{N}(T)|$ für die Anzahl der Knoten des Polygons T steht.

Nun können die folgenden lokalen Freiheitsgrade gewählt werden.

Definition 3.8 (Lokale Freiheitsgrade)**.** Für $\boldsymbol{v}_h \in \boldsymbol{V}_k^T$ werden

- $D_{\boldsymbol{V}}^{\mathcal{N}}$: die Werte von $\boldsymbol{v}_h$ an den Knoten,
- $D_{\boldsymbol{V}}^{\mathcal{E}}$: die Werte von $\boldsymbol{v}_h$ an $k-1$ paarweise verschiedenen Punkten auf jeder Kante $E \in \mathcal{E}(T)$, zum Beispiel die inneren Punkte der Gauß-Lobatto Integrationsformel mit $k+1$ Punkten,
- $D_{\boldsymbol{V}}^{\mathcal{M}_1}$: die Momente für $1 \le j \le \pi_{k-3}$

$$\frac{1}{|T|}\int_T \boldsymbol{v}_h \cdot \boldsymbol{m}^{\perp} m_j \,\mathrm{d}T,$$

- $D_{\boldsymbol{V}}^{\mathcal{M}_2}$: die Momente für $2 \leq j \leq \pi_{k-1}$

$$\frac{h_T}{|T|} \int_T \mathrm{div} \boldsymbol{v}_h \, m_j \, \mathrm{d}T$$

als lokale Freiheitsgrade definiert.

Als lokale Freiheitsgrade für $q_h \in Q_k^T$ werden

- $D_Q^{\mathcal{Y}}$: die Koeffizienten $y_j \in \mathbb{R}$ für $1 \leq j \leq \pi_{k-1}$ der Zerlegung in die skalierten Monome $\mathbb{M}_{k-1}(T)$

$$q_h = \sum_{j=1}^{\pi_{k-1}} y_j \, m_j$$

gewählt.

Alternativ hätten auch die Freiheitsgrade $D_{\boldsymbol{V}}^{\mathcal{M}_1}$ wie in [dLV17] durch die Momente

$$\int_T \boldsymbol{v}_h \cdot \boldsymbol{g}_{k-2}^{\perp} \mathrm{d}T$$

für alle $\boldsymbol{g}_{k-2}^{\perp} \in \mathcal{G}_{k-2}(T)^{\perp} := \{ \boldsymbol{h} \in [\mathbb{P}_{k-2}(T)]^2 \, : \, (\boldsymbol{g}, \boldsymbol{h})_{L^2(T)} = 0 \text{ für alle } \boldsymbol{g} \in \mathcal{G}_{k-2}(T) \}$ ersetzt werden können. Eine Visualisierung der Freiheitsgrade für die Geschwindigkeit für $k = 2, 3$ findet sich in Abbildung 3.1.

Satz 3.9. *Die lokalen Freiheitsgrade sind unisolvent für die jeweiligen Räume.*

Beweis. Der Beweis folgt der Idee aus [dLV17]. Da die Anzahl der Freiheitsgrade genau der Dimension von $\boldsymbol{V}_k^T$ entspricht, reicht es zu zeigen, dass wenn für eine Funktion $\boldsymbol{v}_h \in \boldsymbol{V}_k^T$ alle Freiheitsgrade gleich Null sind, dann $\boldsymbol{v}_h \equiv 0$ gelten muss.

Sei also $\boldsymbol{v}_h \in \boldsymbol{V}_k^T$ mit $D_{\boldsymbol{V}}^{\mathcal{N}}(\boldsymbol{v}_h) = 0$, $D_{\boldsymbol{V}}^{\mathcal{E}}(\boldsymbol{v}_h) = 0$, $D_{\boldsymbol{V}}^{\mathcal{M}_1}(\boldsymbol{v}_h) = 0$ und $D_{\boldsymbol{V}}^{\mathcal{M}_2}(\boldsymbol{v}_h) = 0$. Da die Funktion auf dem Rand ein Polynom vom Grade maximal k ist, folgt aus den Bedingungen $D_{\boldsymbol{V}}^{\mathcal{N}}(\boldsymbol{v}_h) = 0$, $D_{\boldsymbol{V}}^{\mathcal{E}}(\boldsymbol{v}_h) = 0$, dass $\boldsymbol{v}_{h|E} = 0$ für alle Kanten $E \in \mathcal{E}(T)$. Deshalb ist $\boldsymbol{v}_h \in [H_0^1(T)]^2$ und durch partielle Integration folgt

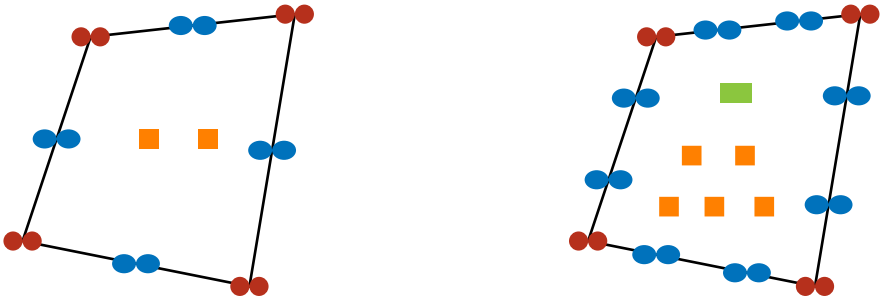
$$\nu \int_T \mathrm{D}\boldsymbol{v}_h : \mathrm{D}\boldsymbol{v}_h \, \mathrm{d}T = -\nu \int_T \Delta \boldsymbol{v}_h \cdot \boldsymbol{v}_h \, \mathrm{d}T.$$

Aus der Definition von $\boldsymbol{V}_k^T$ folgt, dass es ein $s \in L^2(T)$ und ein $p_{k-3} \in \mathbb{P}_{k-3}(T)$ gibt, so dass $-\nu \Delta \boldsymbol{v}_h = -\nabla s + \begin{pmatrix} x_2 \\ -x_1 \end{pmatrix} p_{k-3}$ gilt. Eingesetzt in die obige Gleichung folgt damit

$$\nu \int_T \mathrm{D}\boldsymbol{v}_h : \mathrm{D}\boldsymbol{v}_h \, \mathrm{d}T = \int_T \left(-\nabla s + \begin{pmatrix} x_2 \\ -x_1 \end{pmatrix} p_{k-3} \right) \cdot \boldsymbol{v}_h \, \mathrm{d}T = -\int_T \nabla s \cdot \boldsymbol{v}_h \, \mathrm{d}T,$$

wobei im letzten Schritt $D_{\boldsymbol{V}}^{\mathcal{M}_1}(\boldsymbol{v}_h) = 0$ ausgenutzt wurde. Erneute partielle Integration unter Berücksichtigung, dass $\boldsymbol{v}_h \in [H_0^1(T)]^2$, führt zu

$$\nu \int_T \mathrm{D}\boldsymbol{v}_h : \mathrm{D}\boldsymbol{v}_h \, \mathrm{d}T = -\int_T \nabla s \cdot \boldsymbol{v}_h \, \mathrm{d}T = \int_T \mathrm{div} \boldsymbol{v}_h \, s \, \mathrm{d}T.$$



(a) $k = 2$ (b) $k = 3$

Abbildung 3.1.: Visualisiert sind die Freiheitsgrade für den diskreten Geschwindigkeitsraum $\boldsymbol{V}_k^T$ für (a) $k = 2$ und (b) $k = 3$. Rote Kreise gehören zu $D_{\boldsymbol{V}}^{\mathcal{N}}$, blaue Ellipsen zu $D_{\boldsymbol{V}}^{\mathcal{E}}$, orange Quadrate zu $D_{\boldsymbol{V}}^{\mathcal{M}_1}$ und grüne Rechtecke zu $D_{\boldsymbol{V}}^{\mathcal{M}_2}$.

Da $\boldsymbol{v}_h \in \boldsymbol{V}_k^T$ ist, existiert ein $p_{k-1} \in \mathbb{P}_{k-1}(T)$ mit $\operatorname{div}\boldsymbol{v}_h = p_{k-1}$, so dass wegen $D_{\boldsymbol{V}}^{\mathcal{M}_2}(\boldsymbol{v}_h) = 0$

$$\int_T (\operatorname{div}\boldsymbol{v}_h)^2 \,\mathrm{d}T = \int_T \operatorname{div}\boldsymbol{v}_h p_{k-1} \,\mathrm{d}T = 0$$

folgt. Deshalb ist $\operatorname{div}\boldsymbol{v}_h = 0$, weshalb

$$\nu \int_T \mathrm{D}\boldsymbol{v}_h : \mathrm{D}\boldsymbol{v}_h \,\mathrm{d}T = 0$$

gilt. Daher muss $\boldsymbol{v}_h$ konstant sein und da die Funktion Null auf dem Rand ist, folgt sofort, dass $\boldsymbol{v}_h \equiv 0$.

Dass die Anzahl der Freiheitsgrade $D_Q^{\mathcal{Y}}$ mit der Dimension von Q_k^T übereinstimmt und dass $q_h = 0$ für eine Funktion $q_h \in Q_k^T$ mit $D_Q^{\mathcal{Y}}(q_h) = 0$ gilt, ist sofort ersichtlich, wodurch auch die Unisolvenz der Freiheitsgrade für Q_k^T folgt. □

Da die lokalen Räume nun definiert sind, können die globalen Räume definiert werden. Das sind lose gesprochen alle Funktionen, die auf jedem Polygon in den lokalen Räumen sind und darüberhinaus global noch weitere hinreichende Bedingungen wie schwache Differenzierbarkeit beziehungsweise Integrierbarkeit erfüllen. Insbesondere soll für die Geschwindigkeit eine $H_0^1(\Omega)$-konforme Methode benutzt werden.

Definition 3.10 (Globale Räume)**.** Die globalen Räume sind durch

$$\boldsymbol{V}_k := \{\, \boldsymbol{v}_h \in [H_0^1(\Omega)]^2 \,:\, \boldsymbol{v}_{h|T} \in \boldsymbol{V}_k^T \text{ für alle } T \in \mathcal{T}_h \,\} \tag{3.9a}$$

und

$$Q_k := \{\, q_h \in L_0^2(\Omega) \,:\, q_{h|T} \in Q_k^T \text{ für alle } T \in \mathcal{T}_h \,\} \tag{3.9b}$$

gegeben.

An dieser Stelle kann bereits beobachtet werden, dass

$$\mathrm{div}\boldsymbol{V}_k \subset Q_k \tag{3.10}$$

gilt, was später für die Druckrobustheit von Vorteil ist.

Ferner benötigen auch die globalen Räume Freiheitsgrade. Diese werden aus der Sammlung der lokalen Freiheitsgrade gebildet. Da die Funktionen für die Geschwindigkeit global in $H_0^1(\Omega)$ liegen, wird die Stetigkeit entlang der Polygongrenzen, das heißt der Kanten benötigt. Dafür reicht es aber die Stetigkeit in den zwei Endpunkten und den $k-1$ inneren Punkten jeder Kante zu fordern, da entlang der Kanten die Funktionen Polynome vom Grad kleiner oder gleich k sind. Die Druckfunktionen sind dahingegen nur stückweise Polynome, weshalb hier keine Stetigkeit entlang der Kanten gefordert werden muss.

Definition 3.11 (Globale Freiheitsgrade)**.** Für $\boldsymbol{v}_h \in \boldsymbol{V}_k$ werden als Freiheitsgrade

- $GD_{\boldsymbol{V}}^{\mathcal{N}}$: die Werte von $\boldsymbol{v}_h$ an den Knoten aller Polygone,
- $GD_{\boldsymbol{V}}^{\mathcal{E}}$: die Werte von $\boldsymbol{v}_h$ an $k-1$ paarweise verschiedenen Punkten auf jeder Kante $E \in \mathcal{E}(T)$ aller Polygone (zum Beispiel innere Punkte der Gauß-Lobatto Integrationsformel mit $k+1$ Punkten),
- $GD_{\boldsymbol{V}}^{\mathcal{M}_1}$: die Momente für alle Polygone und für $1 \leq j \leq \pi_{k-3}$

$$\frac{1}{|T|}\int_T \boldsymbol{v}_h \cdot m^{\perp} m_j \,\mathrm{d}T,$$

- $GD_{\boldsymbol{V}}^{\mathcal{M}_2}$: die Momente für alle Polygone und für $2 \leq j \leq \pi_{k-1}$

$$\frac{h_T}{|T|}\int_T \mathrm{div}\boldsymbol{v}_h \, m_j \,\mathrm{d}T$$

gewählt.

Für $q_h \in Q_k$ werden

- $GD_Q^{\mathcal{Y}}$: die Koeffizienten $y_j^{(T)}$ für $1 \leq j \leq \pi_{k-1}$ der Zerlegung in skalierte Monome $\mathbb{M}_{k-1}(T)$ für alle Polygone T

$$q_{h|T} = \sum_{j=1}^{\pi_{k-1}} y_j^{(T)} \, m_j$$

als Freiheitsgrade festgelegt.

Aus der Dimensionsüberlegung für die lokalen Räume und der Definition der globalen Räume folgt sofort die Dimension der globalen Räume. Es ist

$$\begin{aligned}
\dim \boldsymbol{V}_k &= 2 \cdot (|\mathcal{N}| + (k-1)|\mathcal{E}|) + |\mathcal{T}| \left(\frac{k \cdot (k+1)}{2} - 1 + \frac{(k-2)(k-1)}{2} \right), \\
\dim Q_k &= |\mathcal{T}| \frac{k \cdot (k+1)}{2} - 1,
\end{aligned}$$

wobei $|\mathcal{N}|$, $|\mathcal{E}|$ beziehungsweise $|\mathcal{T}|$ die Anzahl der Knoten, die Anzahl der Kanten beziehungsweise die Anzahl der Polygone der Zerlegung sind.

Satz 3.12. *Die globalen Freiheitsgrade sind unisolvent für die globalen Räume*

Beweis. Dies folgt direkt aus der Definition der Freiheitsgrade und Satz 3.9. Eine Funktion, deren globale Freiheitsgrade Null sind, ist wegen der lokalen Unisolvenz elementweise gleich Null und damit auf dem ganzen Gebiet Ω gleich Null. Erneutes Zählen der Freiheitsgrade und Vergleichen mit der Dimensionen vollendet den Beweis. □

Jetzt müssen noch die diskreten Bilinearformen definiert werden. Das Ganze ist durch die Zerlegung der kontinuierlichen Bilinearformen

$$a(\boldsymbol{v},\boldsymbol{w}) = \sum_{T\in\mathcal{T}} a^T(\boldsymbol{v},\boldsymbol{w}) \qquad \text{und} \qquad b(\boldsymbol{v},q) = \sum_{T\in\mathcal{T}} b^T(\boldsymbol{v},q) \tag{3.11}$$

inspiriert, was für alle $\boldsymbol{v},\boldsymbol{w} \in \boldsymbol{V}$ und alle $q \in Q$ gilt. Dabei ist

$$\begin{aligned} a^T(\boldsymbol{v},\boldsymbol{w}) &:= \int_T \nu D\boldsymbol{v} : D\boldsymbol{w}\,\mathrm{d}T, \\ b^T(\boldsymbol{v},q) &:= \int_T \mathrm{div}\boldsymbol{v}\, q\,\mathrm{d}T \end{aligned}$$

der jeweilige lokale Anteil an der globalen Bilinearform. Es werden daher zuerst lokale diskrete Bilinearformen definiert, die dann über alle Polygone $T \in \mathcal{T}_h$ aufsummiert werden, um die globale diskrete Bilinearform zu erhalten.

Wie für Virtuelle-Elemente-Methoden üblich (zum Beispiel in [dBC+13,dLV17]) muss die diskrete lokale Bilinearform $a_h^T : \boldsymbol{V}_k^T \times \boldsymbol{V}_k^T \to \mathbb{R}$ zwei Bedingungen erfüllen:

- **k-Konsistenz**: Für alle $\boldsymbol{q_k} \in [\mathbb{P}_k(T)]^2$ und $\boldsymbol{v}_h \in \boldsymbol{V}_k^T$ gilt

$$a_h^T(\boldsymbol{q}_k,\boldsymbol{v}_h) = a^T(\boldsymbol{q}_k,\boldsymbol{v}_h); \tag{3.12}$$

- **Stabilität:** Es existieren zwei positive Konstanten α_*, α^*, unabhänig von h und T, so dass für alle $\boldsymbol{v}_h \in \boldsymbol{V}_k^T$

$$\alpha_* a^T(\boldsymbol{v}_h,\boldsymbol{v}_h) \le a_h^T(\boldsymbol{v}_h,\boldsymbol{v}_h) \le \alpha^* a^T(\boldsymbol{v}_h,\boldsymbol{v}_h) \tag{3.13}$$

  gilt.

Diese Bedingungen werden später benötigt, um sowohl die Existenz eindeutiger Lösungen als auch eine hinreichenende Konvergenzgeschwindigkeit zu garantieren.

Bemerkung 3.13. Die Stabilitätsbedingung und die Symmetrie von a_h^T liefern zusammen mit der Stetigkeit von a^T die Stetigkeit von a_h^T, denn es gilt

$$\begin{aligned} a_h^T(\boldsymbol{v}_h,\boldsymbol{w}_h) &\le a_h^T(\boldsymbol{v}_h,\boldsymbol{v}_h)^{1/2} a_h^T(\boldsymbol{w}_h,\boldsymbol{w}_h)^{1/2} \le \alpha^* a^T(\boldsymbol{v}_h,\boldsymbol{v}_h)^{1/2} a^T(\boldsymbol{w}_h,\boldsymbol{w}_h)^{1/2} \\ &= \nu\,\alpha^* |\boldsymbol{v}_h|_{H^1(T)} |\boldsymbol{w}_h|_{H^1(T)} \end{aligned}$$

für alle $\boldsymbol{v}_h,\boldsymbol{w}_h \in \boldsymbol{V}_k^T$.

Im Kontext Virtueller-Elemente-Methoden wird für $\boldsymbol{v}_h \in \boldsymbol{V}_k^T$ die lokale Bestapproximation $\Pi_k^{\nabla,T} : \boldsymbol{V}_k^T \to [\mathbb{P}_k(T)]^2$ bezüglich des Energieskalarprodukts a benötigt, die als Lösung von

$$a^T(\boldsymbol{v}_h, \boldsymbol{q}_k) = a^T(\Pi_k^{\nabla,T}\boldsymbol{v}_h, \boldsymbol{q}_k) \qquad \text{für alle } \boldsymbol{q}_k \in [\mathbb{P}_k(T)]^2, \tag{3.14}$$

$$P_0^T \boldsymbol{v}_h = P_0^T \Pi_k^{\nabla,T} \boldsymbol{v}_h \tag{3.15}$$

definiert ist. Dabei ist für $\boldsymbol{v}_h \in \boldsymbol{V}_k^T$ die Projektion durch

$$P_0^T \boldsymbol{v}_h := \frac{1}{|T|} \int_T \boldsymbol{v}_h \mathrm{d}T$$

gegeben. Das Interessante an der Definition ist, dass die Projektion ausschließlich mittels der Freiheitsgrade berechnet werden kann.

Auf der rechten Seite stehen, da $\Pi_k^{\nabla,T}\boldsymbol{v}_h \in [\mathbb{P}_k(T)]^2$, Integrale von Polynomen über Polygone, deren Auswertung vorausgesetzt werden kann.

Die linke Seite von Gleichung (3.14) kann mittels partieller Integration zu

$$a^T(\boldsymbol{v}_h, \boldsymbol{q}_k) = \int_T \nu \mathrm{D}\boldsymbol{v}_h : \mathrm{D}\boldsymbol{q}_k \,\mathrm{d}T = -\int_T \nu \boldsymbol{v} \cdot \Delta \boldsymbol{q}_k \,\mathrm{d}T + \int_{\partial T} \nu\, \boldsymbol{v} \cdot \mathrm{D}\boldsymbol{q}_k \boldsymbol{n} \,\mathrm{d}s \tag{3.16}$$

umgeformt werden. Da $\Delta \boldsymbol{q}_k \in [\mathbb{P}_{k-2}(T)]^2$ ein Polynom ist, kann es nach Gleichung (3.1) eindeutig in einen Gradientenanteil ∇r_{k-1} und einen orthogonalen Anteil $\boldsymbol{m}^\perp r_{k-3}$ mit Polynomen $r_{k-1} \in \mathbb{P}_{k-1}(T)$, $r_{k-3} \in \mathbb{P}_{k-3}(T)$ zerlegt werden, so dass

$$\Delta \boldsymbol{q}_k = \nabla r_{k-1} + \boldsymbol{m}^\perp r_{k-3}$$

gilt. Eingesetzt in (3.16) ergibt sich damit mit erneuter partieller Integration

$$a^T(\boldsymbol{v}_h, \boldsymbol{q}_k) = -\int_T \nu \boldsymbol{v} \cdot \boldsymbol{m}^\perp r_{k-3} \,\mathrm{d}T + \int_T \nu \mathrm{div}\boldsymbol{v}\, r_{k-1} \,\mathrm{d}T + \int_{\partial T} \nu\, \boldsymbol{v} \cdot (\mathrm{D}\boldsymbol{q}_k \boldsymbol{n} - r_{k-1}\boldsymbol{n}) \,\mathrm{d}s.$$

Alle drei Terme können mit Hilfe der Freiheitsgrade berechnet werden. Für das Randintegral reichen die Freiheitsgrade $D_{\boldsymbol{V}}^{\mathcal{V}}$ und $D_{\boldsymbol{V}}^{\mathcal{E}}$, da der Integrand ein Polynom vom Grad maximal $2k-1$ ist. Die ersten beiden Integrale werden mit den Freiheitsgraden $D_{\boldsymbol{V}}^{\mathcal{M}_1}$ beziehungsweise $D_{\boldsymbol{V}}^{\mathcal{M}_2}$ berechnet.

Auch die linke Seite von Gleichung (3.15) kann mit Hilfe der Freiheitsgrade berechnet werden. Dafür wird die linke Seite durch

$$\begin{aligned}\frac{1}{|T|}\int_T \boldsymbol{v}_h \,\mathrm{d}T &= \frac{1}{|T|}\begin{pmatrix} \int_T \boldsymbol{v}_h \cdot \begin{pmatrix}1\\0\end{pmatrix} \mathrm{d}T \\ \int_T \boldsymbol{v}_h \cdot \begin{pmatrix}0\\1\end{pmatrix} \mathrm{d}T \end{pmatrix} = \frac{h_T}{|T|}\begin{pmatrix} \int_T \boldsymbol{v}_h \nabla m_2 \,\mathrm{d}T \\ \int_T \boldsymbol{v}_h \nabla m_3 \,\mathrm{d}T \end{pmatrix} \\ &= \frac{h_T}{|T|}\begin{pmatrix} -\int_T \mathrm{div}\boldsymbol{v}_h\, m_1 \,\mathrm{d}T + \int_{\partial T} \boldsymbol{v}_h \cdot \boldsymbol{n}\, m_2 \,\mathrm{d}T \\ -\int_T \mathrm{div}\boldsymbol{v}_h\, m_2 \,\mathrm{d}T + \int_{\partial T} \boldsymbol{v}_h \cdot \boldsymbol{n}\, m_3 \,\mathrm{d}T \end{pmatrix}\end{aligned} \tag{3.17}$$

umgeformt, wobei im letzten Schritt partielle Integration angewendet wird. Die Randintegrale können erneut mittels $D_{\boldsymbol{V}}^{\mathcal{N}}$ und $D_{\boldsymbol{V}}^{\mathcal{E}}$ und die Volumenintegrale mittels $D_{\boldsymbol{V}}^{\mathcal{M}_2}$ ausgewertet werden.

Dadurch könnte jetzt die diskrete lokale Bilinearform für alle $\boldsymbol{v}_h, \boldsymbol{w}_h \in \boldsymbol{V}_k^T$ als

$$a_h^T(\boldsymbol{v}_h, \boldsymbol{w}_h) = a^T(\Pi_k^{\nabla,T}\boldsymbol{v}_h, \Pi_k^{\nabla,T}\boldsymbol{w}_h)$$

gewählt werden. Allerdings ist damit nur die k-Konsistenz Bedingung (3.12) erfüllt und die Stabilitätsbedingung (3.13) würde im Allgemeinen nicht gelten. Daher wird noch eine sogenannte Stabilitätsbilinearform $S(\cdot,\cdot)$ addiert. Hier wird die gewöhnliche Stabilisierung, beispielsweise beschrieben in [DV19], genutzt die für alle $\boldsymbol{v}_h, \boldsymbol{w}_h \in \boldsymbol{V}_k^T$ durch

$$S^T(\boldsymbol{v}_h, \boldsymbol{w}_h) := \sum_{j=1}^{\dim \boldsymbol{V}_k^T} D_{V,j}(\boldsymbol{v}_h) D_{V,j}(\boldsymbol{w}_h) \tag{3.18}$$

definiert ist. Es werden nun die folgenden Bilinearformen gewählt.

Definition 3.14 (Lokale diskrete Bilinearformen)**.** Für alle $\boldsymbol{v}_h, \boldsymbol{w}_h \in \boldsymbol{V}_k^T$ und $q_h \in Q_k^T$ werden die lokalen diskreten Bilinearformen

$$a_h^T(\boldsymbol{v}_h, \boldsymbol{w}_h) := a^T\left(\Pi_k^{\nabla,T}\boldsymbol{v}_h, \Pi_k^{\nabla,T}\boldsymbol{w}_h\right) + \nu S^T\left((\boldsymbol{I} - \Pi_k^{\nabla,T})\boldsymbol{v}_h, (\boldsymbol{I} - \Pi_k^{\nabla,T})\boldsymbol{w}_h\right) \tag{3.19a}$$

und

$$b_h^T(\boldsymbol{v}_h, q_h) := b^T(\boldsymbol{v}_h, q_h) \tag{3.19b}$$

definiert.

Damit die Definitionen zielführend sind, muss noch geklärt werden, ob auch die Bilinearform $b(\boldsymbol{v}_h, q_h)$ für alle $\boldsymbol{v}_h \in \boldsymbol{V}_k^T$ und $q_h \in Q_k^T$ berechnet werden kann. Das folgt allerdings leicht aus der Definition der Freiheitsgrade. Es gilt nämlich

$$b^T(\boldsymbol{v}_h, m_\ell) = \int_T \operatorname{div}\boldsymbol{v}_h \, m_\ell \,\mathrm{d}T = \begin{cases} \int_{\partial T} \boldsymbol{v}_h \cdot \boldsymbol{n} \,\mathrm{d}s & \ell = 1, \\ \int_T \operatorname{div}\boldsymbol{v}_h \, m_\ell \,\mathrm{d}T & \ell = 2, 3, \ldots, \pi_{k-1} \end{cases} \tag{3.20}$$

für $m_\ell \in \mathbb{M}_{k-1}$. Im ersten Fall kann das Integral mittels der Randfreiheitsgrade $D_{\boldsymbol{V}}^{\mathcal{N}}$ und $D_{\boldsymbol{V}}^{\mathcal{E}}$ berechnet werden. Der andere Fall wird mit Hilfe von $D_{\boldsymbol{V}}^{\mathcal{M}_2}$ berechnet.

Die gewählten diskreten Bilinearformen sind also wohldefiniert und mittels der Freiheitsgrade berechenbar. Darüber hinaus gilt das folgende Lemma.

Lemma 3.15. *Die diskrete lokale Bilinearform $a_h^T : \boldsymbol{V}_k^T \times \boldsymbol{V}_k^T$ erfüllt die k-Konsistenz Bedingung (3.12) und die Stabilitätsbedingung (3.13).*

Beweis. Der Beweis skizziert die Idee aus [dBC+13, dBMR14b].

Sei für die k-Konsistenz Bedingung ein $\boldsymbol{q}_k \in [\mathbb{P}_k(T)]^2$ gegeben. Dann folgt sofort, dass $(\boldsymbol{I} - \Pi_k^{\nabla,T})\boldsymbol{q}_k = 0$, woraus $S^T(\boldsymbol{q}_k, \boldsymbol{v}_h) = 0$ für alle $\boldsymbol{v}_h \in \boldsymbol{V}_k^T$ folgt. Daraus folt instantan wegen der Definition der Projektion $\Pi_k^{\nabla,T}$

$$a_h^T(\boldsymbol{q}_k, \boldsymbol{v}_h) = a^T(\boldsymbol{q}_k, \boldsymbol{v}_h)$$

für alle $\boldsymbol{v}_h \in \boldsymbol{V}_k^T$.

Zum Nachweis der Stabilitätsbedingung reicht es aus, zu zeigen, dass $S^T(\cdot,\cdot)$ eine symmetrische, positiv definite Bilinearform ist, die

$$a^T(\boldsymbol{v}_h, \boldsymbol{v}_h) \approx \nu S^T(\boldsymbol{v}_h, \boldsymbol{v}_h) \qquad \text{für alle } \boldsymbol{v}_h \in \boldsymbol{V}_k^T \text{ mit } \Pi_k^{\nabla,T} \boldsymbol{v}_h = 0 \tag{3.21}$$

erfüllt. Einfaches Nachrechnen führt dann zur Stabilitätsbedingung.

Die Symmetrie und die positive Definitheit folgen sofort aus der Definition von $S^T(\cdot,\cdot)$ aus Gleichung (3.18). Zur Motivation von Gleichung (3.21) wird eine Funktion $\varphi \in \boldsymbol{V}_k^T$ betrachtet, die auf einem Gebiet $T \in \mathcal{T}_h$ entweder

$$D_{\boldsymbol{V}}^{\mathcal{M}_1}\varphi = \frac{1}{|T|}\int_T \varphi(\boldsymbol{x}) \cdot \begin{pmatrix} \left(\frac{\boldsymbol{x}-\boldsymbol{x}_T}{h_T}\right)^{(0,1)} \\ -\left(\frac{\boldsymbol{x}-\boldsymbol{x}_T}{h_T}\right)^{(1,0)} \end{pmatrix} \left(\frac{\boldsymbol{x}-\boldsymbol{x}_T}{h_T}\right)^{\boldsymbol{\alpha}} \mathrm{d}T = 1$$

für ein $\boldsymbol{\alpha} \in \mathbb{N}^2$ mit $0 \leq |\boldsymbol{\alpha}| \leq k-3$ oder

$$D_{\boldsymbol{V}}^{\mathcal{M}_2}\varphi = \frac{h_T}{|T|}\int_T \mathrm{div}\varphi(\boldsymbol{x}) \left(\frac{\boldsymbol{x}-\boldsymbol{x}_T}{h_T}\right)^{\beta} \mathrm{d}T = 1$$

für ein $1 \leq |\beta| \leq k-1$ erfüllt. Auch hier zeigt sich durch einfaches Nachrechnen, dass auf dem Gebiet $\widehat{T} := hT$ für ein positives $h \in \mathbb{R}$ durch $\varphi(\boldsymbol{x}) = \widehat{\varphi}(\widehat{\boldsymbol{x}}) = \widehat{\varphi}(h\boldsymbol{x})$ auch

$$\frac{1}{|\widehat{T}|}\int_{\widehat{T}} \widehat{\varphi}(\widehat{\boldsymbol{x}}) \cdot \begin{pmatrix} \left(\frac{\widehat{\boldsymbol{x}}-\boldsymbol{x}_{\widehat{T}}}{h_{\widehat{T}}}\right)^{(0,1)} \\ -\left(\frac{\widehat{\boldsymbol{x}}-\boldsymbol{x}_{\widehat{T}}}{h_{\widehat{T}}}\right)^{(1,0)} \end{pmatrix} \left(\frac{\widehat{\boldsymbol{x}}-\boldsymbol{x}_{\widehat{T}}}{h_{\widehat{T}}}\right)^{\boldsymbol{\alpha}} \mathrm{d}\widehat{T} = 1$$

und

$$\frac{h_{\widehat{T}}}{|\widehat{T}|}\int_{\widehat{T}} \widehat{\mathrm{div}\varphi}(\widehat{\boldsymbol{x}}) \left(\frac{\widehat{\boldsymbol{x}}-\boldsymbol{x}_{\widehat{T}}}{h_{\widehat{T}}}\right)^{\beta} \mathrm{d}\widehat{T} = 1$$

folgen. Die Basisfunktionen skalieren somit durch die Wahl der Freiheitsgrade und der skalierten Monome auf dem Gebiet wie 1. Dies tut aber auch $a^T(\cdot,\cdot)$, denn es gilt mit selbigen Überlegungen wie vorher

$$a^T(\varphi,\varphi) \approx \nu.$$

Die obige Wahl von $S^T(\cdot,\cdot)$ erfüllt also Gleichung (3.21), so dass die gewählte Bilinearform die Stabilitätsbedingung erfüllt. □

Bemerkung 3.16. (i) Ein rigoroser Stabilitätsbeweis findet sich auch in [dLR17] für die oben definierte Stabilitätsform als auch für eine Alternative.

(ii) Der obige Beweis kann auch als Motivation für die Wahl der Freiheitsgrade und der skalierten Monome verstanden werden.

Nun, da die lokalen diskreten Bilinearformen definiert sind, können auch die globalen diskreten Bilinearformen festgelegt werden. Dies sind die Summen der lokalen Bilinearformen, um das Verhalten der kontinuierlichen Bilinearformen (3.11) zu spiegeln.

Definition 3.17 (Globale diskrete Bilinearformen)**.** Als globale diskrete Bilinearformen werden für alle $\boldsymbol{v}_h, \boldsymbol{w}_h \in \boldsymbol{V}_k$ und $q_h \in Q_h$

$$a_h(\boldsymbol{v}_h, \boldsymbol{w}_h) := \sum_{T \in \mathcal{T}_h} a_h^T(\boldsymbol{v}_h, \boldsymbol{w}_h) \tag{3.22a}$$

und

$$b_h(\boldsymbol{v}_h, q_h) := \sum_{T \in \mathcal{T}_h} b_h^T(\boldsymbol{v}_h, q_h) \tag{3.22b}$$

gewählt.

Das Einzige, was noch für das diskrete Problem fehlt ist eine Diskretisierung der rechten Seite. Wie in den meisten Fällen (zum Beispiel [dBC+13, dLV17]) wird als diskrete rechte Seite $\boldsymbol{f}_h$ die stückweise L^2-Projektion von $\boldsymbol{f}$ auf die Polynome $[\mathbb{P}_{k-2}(T)]^2$ gewählt. Diese Projektion muss durchgeführt werden, da die Basisfunktionen nur implizit zur Verfügung stehen, womit $(\boldsymbol{f}, \varphi_j)_{L^2(\Omega)}$, $j = 1, 2, \ldots, \dim \boldsymbol{V}_k$, nicht explizit berechnet werden kann.

Definition 3.18 (Standarddiskretisierung der rechten Seite)**.** Sei $\boldsymbol{f}_{h|T} := P_{k-2}^T \boldsymbol{f}$ die stückweise L^2-Projektion von $\boldsymbol{f}$ auf Polynome vom Grad kleiner oder gleich $k-2$. Die Standarddiskretisierung der rechten Seite ist für alle $\boldsymbol{v}_h \in \boldsymbol{V}_k$ durch

$$(\boldsymbol{f}_h, \boldsymbol{v}_h)_{L^2(\Omega)}$$

gegeben.

Schon hier darf vorweggegriffen werden, dass die Diskretisierung der rechten Seite große Auswirkungen auf die Druckrobustheit hat. Darauf wird in Kapitel 6 näher eingegangen. Hier wird nun zuerst diese Diskretisierung gewählt.

Es muss ebenfalls überprüft werden, ob die Diskretisierung der rechten Seite mit Hilfe der Freiheitsgrade berechnet werden kann. Sei dafür $\boldsymbol{v}_h \in \boldsymbol{V}_k$ gegeben. Aus der Definition der L^2-Projektion folgt

$$(\boldsymbol{f}_h, \boldsymbol{v}_h)_{L^2(\Omega)} = \sum_{T \in \mathcal{T}_h} (P_{k-2}^T \boldsymbol{f}, \boldsymbol{v}_h)_{L^2(T)} = \sum_{T \in \mathcal{T}_h} (\boldsymbol{f}, P_{k-2}^T \boldsymbol{v}_h)_{L^2(T)}.$$

Dies ist berechenbar, da $P_{k-2}^T \boldsymbol{v}_h$ mit Hilfe der Freiheitsgrade berechnet werden kann, wie nun gezeigt wird. Für alle $\boldsymbol{q}_{k-2} \in [\mathbb{P}_{k-2}(T)]$ gilt wegen Gleichung (3.1) und Lemma 3.5

$$\int_T P_{k-2}^T \boldsymbol{v}_h \cdot \boldsymbol{q}_{k-2} \, \mathrm{d}T = \int_T \boldsymbol{v}_h \cdot \boldsymbol{q}_{k-2} \, \mathrm{d}T = \int_T \boldsymbol{v}_h \cdot (\nabla q_{k-1} + \boldsymbol{m}^\perp q_{k-3}) \, \mathrm{d}T,$$

für geeignete $q_{k-1} \in \mathbb{P}_{k-1(T)}$ und $q_{k-3} \in \mathbb{P}_{k-3}(T)$. Deshalb folgt

$$\int_T P_{k-2}^T \boldsymbol{v}_h \cdot \boldsymbol{q}_{k-2} \,\mathrm{d}T = -\int_T \mathrm{div}\boldsymbol{v}_h \, q_{k-1} \,\mathrm{d}T + \int_{\partial T} \boldsymbol{v}_h \cdot \boldsymbol{n} \, q_{k-1} \,\mathrm{d}s + \int_T \boldsymbol{v}_h \cdot \boldsymbol{m}^\perp q_{k-3} \,\mathrm{d}T, \tag{3.23}$$

was mittels der Freiheitsgrade berechnet werden kann.

Bemerkung 3.19. Bereits an dieser Stelle kann festgestellt werden, dass die volle L^2-Projektion nur auf Polynome vom Grad kleiner oder gleich $k-2$ zur Verfügung steht. Dies ist der maximale Grad, der mit den Freiheitsgraden berechnet werden kann.

Nun sind alle Ingredenzien vorhanden, um das diskrete Problem zu definieren.

Definition 3.20 (Diskretes Problem)**.** Das diskrete Problem liest sich wie folgt: Finde $(\boldsymbol{u}_h, p_h) \in \boldsymbol{V}_k \times Q_k$, so dass

$$a_h(\boldsymbol{u}_h, \boldsymbol{v}_h) + b_h(\boldsymbol{v}_h, p_h) = (\boldsymbol{f}_h, \boldsymbol{v}_h)_{L^2(\Omega)} \qquad \text{für alle } \boldsymbol{v}_h \in \boldsymbol{V}_k, \tag{3.24a}$$

$$b(\boldsymbol{u}_h, q_h) = 0 \qquad \text{für alle } q_h \in Q_k \tag{3.24b}$$

gelten.

An dieser Stelle muss noch betont werden, dass durch Gleichung (3.24b) zusammen mit der Eigenschaft aus Gleichung (3.10) folgt, dass die diskrete Geschwindigkeit $\boldsymbol{u}_h \in \boldsymbol{V}_h$ *exakt divergenzfrei* ist. Das ist in vielen klassischen Finiten-Elemente-Methoden wie beispielsweise bei der Taylor-Hood Familie, dem MINI-Element oder dem Bernardi-Raugel-Element nicht der Fall [JLM$^+$17] und ist eine gute Voraussetzung für eine druckrobuste Diskretisierung. Mehr dazu in Kapitel 6.

3.2.2. Wohldefiniertheit des diskreten Problems

Die Theorie zur Existenz eindeutiger Lösungen für das diskrete Problem folgt der Sattelpunktstheorie für gemischte Probleme. Eine ausführliche Beschreibung der Theorie inklusive Beweisen findet sich beispielsweise in [Bar16]. Hier soll der Fokus aber auf der Konstruktion des Fortin-Operators liegen, weshalb im Folgenden nur kurz die wichtigsten Inhalte der Sattelpunktstheorie zusammengefasst werden.

Die Existenz eindeutiger Lösungen folgt im kontinuierlichen Fall aus Satz 2.18, der unter anderem die inf-sup Stabilität für $b : X \times Y \to \mathbb{R}$ voraussetzt. Er gilt ebenfalls für abgeschlossene lineare Unterräume $X_h \subset X$ und $Y_h \subset Y$ zweier Hilberträume X und Y. Allerdings folgt im Allgemeinen aus der kontinuierlichen inf-sup Stabilität der Bilinearform nicht die inf-sup Stabilität von $b : X_h \times Y_h \to \mathbb{R}$. Daher wird jetzt die diskrete inf-sup Stabilität oder auch *LBB*-Bedingung, benannt nach Ladyzhenskaya, Babuška und Brezzi, definiert.

Definition 3.21 (*LBB*-Bedingung)**.** Die Bilinearform erfüllt die diskrete inf-sup Stabilität oder auch *LBB*-Bedingung auf $X_h \times Y_h \subset X \times Y$, wenn es eine positive Konstante $0 < \beta_h \in \mathbb{R}$ gibt, so dass

$$0 < \beta_h \leq \inf_{y_h \in Y_h \setminus \{0\}} \sup_{x_h \in X_h \setminus \{0\}} \frac{b(x_h, y_h)}{\|x_h\|_X \|y_h\|_Y} \tag{3.25}$$

gilt.

Hieraus ergibt sich nun die folgende Proposition.

Proposition 3.22 (Babuška-Brezzi-Bedingung)**.** *Seien die Voraussetzungen aus Satz 2.18 für zwei Hilberträume X und Y und zwei Bilinearformen $a_h : X \times X \to \mathbb{R}$ und $b : X \times Y \to \mathbb{R}$ erfüllt. Seien ferner $X_h \subset X$ und $Y_h \subset Y$ zwei abgeschlossene lineare Unterräume. Außerdem erfülle b die LBB-Bedingung und für jedes $y_h \in Y \setminus \{0\}$ existiere ein $x_h \in X_h$, so dass $b(x_h, y_h) \neq 0$. Wenn darüber hinaus a_h elliptisch auf dem diskreten Kernraum*

$$Z_h := \{x_h \in X_h \,:\, b(x_h, y_h) = 0 \text{ für alle } y_h \in Y_h\}$$

ist, dann existiert ein eindeutiges Paar $(x_h, y_h) \in X_h \times Y_h$, so dass

$$\begin{aligned} a_h(x_h, v_h) + b(v_h, y_h) &= F(v_h) && \text{für alle } v_h \in X_h, \\ b(x_h, q_h) &= G(q_h) && \text{für alle } q_h \in Y_h \end{aligned}$$

für gegebene Funktionale $F \in X^*$ und $G \in Y^*$ gelten.

Generell kann es schwierig sein, die *LBB*-Bedingung direkt zu zeigen. Glücklicherweise gibt es aber eine äquivalente Bedingung, nämlich die Existenz eines Fortin-Operators.

Satz 3.23 (Fortin-Kriterium)**.** *Sei vorausgesetzt, dass b die kontinuierliche inf-sup Bedingung mit Konstante β erfüllt. Genau dann erfüllt b die LBB-Bedingung mit Konstante $\beta_h := C_F^{-1}\beta$, wenn ein linearer, nicht-trivialer Operator $I_F : X \to X_h$ existiert, so dass*

$$b(x - I_F x, y_h) = 0 \qquad \text{für alle } y_h \in Y_h, \tag{3.26a}$$

$$\|I_F x\|_X \leq C_F \|x\|_X \qquad \text{für alle } x \in X \tag{3.26b}$$

gelten. Dabei ist C_F die Operatornorm des sogenannten Fortin-Operators I_F.

Es kann nun gezeigt werden, dass solch ein Fortin-Operator auch für die diskreten Unterräume $\boldsymbol{V}_k \subset \boldsymbol{V}$ und $Q_k \subset Q$ der Hilberträume $\boldsymbol{V}$ und Q aus (3.9a) und (3.9b) sowie die Bilinearformen a_h und b_h aus (3.22a) und (3.22b) existiert. Ein wichtiger Bestandteil dafür ist die Existenz eines Quasi-Interpolators $J : \boldsymbol{V} \to V_k$ mit sogenannten Approximationseigenschaften erster Ordnung

$$\|\boldsymbol{v} - J\boldsymbol{v}\|_{L^2(T)} + h_T |\boldsymbol{v} - J\boldsymbol{v}|_{H^1(T)} \leq C\, h_T |\boldsymbol{v}|_{H^1(\omega_T)} \qquad \text{für alle } \boldsymbol{v} \in \boldsymbol{V} \text{ und } T \in \mathcal{T},$$

wobei mit ω_T der „Diamant“ von $T \in \mathcal{T}$ bezeichnet wird. Dies ist die Menge aller Polygone, deren Schnitt mit T nicht leer ist. Die Existenz solcher Operatoren auf Polygonen wurden beispielsweise in Lemma 5 und Proposition 1 aus [Wei17] oder in Proposition 4.1 aus [dLV17] auf verschiedene Weise nachgewiesen. Im ersten Fall werden die Koeffizienten der Basisfunktionen durch das Integralmittel der angrenzenden Polygone beziehungsweise Kanten gewählt. Im zweiten Fall wird im Wesentlichen eine Clément-Interpolation auf einer Subtriangulierung des Polygons durchgeführt, mit dessen Hilfe dann implizit eine Interpolation als Lösung eines Stokes-Problems festgelegt werden kann. Für die Zwecke dieser Arbeit kann also folgendes Lemma vorausgesetzt werden, welches inklusive des Beweises in [dLV17] zu finden ist.

Lemma 3.24 (Quasi-Interpolator nach [dLV17]). *Unter der Voraussetzung eines formregulären Gitters existiert für jedes* $0 \leq s \leq k$ *und jedes* $\boldsymbol{v} \in \boldsymbol{V} \cap [H^{s+1}(\Omega)]^2$ *ein* $J\boldsymbol{v} \in \boldsymbol{V}_k$*, so dass*

$$\|\boldsymbol{v} - J\boldsymbol{v}\|_{L^2(T)} + h_T|\boldsymbol{v} - J\boldsymbol{v}|_{H^1(T)} \leq C\, h_T^{s+1}|\boldsymbol{v}|_{H^{s+1}(\omega_T)}, \tag{3.27}$$

wobei die Konstante C *unabhänig von* h_T *ist.*

Diesen Interpolator vorausgesetzt, kann jetzt der folgende Satz bewiesen werden.

Satz 3.25 (Fortin-Operator). *Für die diskreten Unterräume* $\boldsymbol{V}_k \subset \boldsymbol{V}$ *und* $Q_k \subset Q$ *aus (3.9a) und (3.9b) sowie die Bilinearformen* a_h *und* b_h *aus (3.22a) und (3.22b) existiert ein Fortin-Interpolator, das heißt ein linearer Operator* $I_F : \boldsymbol{V} \to \boldsymbol{V}_k$*, der für alle* $\boldsymbol{v} \in \boldsymbol{V}$

$$b(\boldsymbol{v} - I_F\boldsymbol{v}, q_h) = 0 \qquad \text{für alle } q_h \in Q_k, \tag{3.28}$$

$$\|I_F\boldsymbol{v}\|_{H^1(\Omega)} \leq C_F\|\boldsymbol{v}\|_{H^1(\Omega)} \approx \|\mathrm{D}\boldsymbol{v}\|_{L^2(\Omega)} \tag{3.29}$$

erfüllt.

Beweis. Sei $\boldsymbol{v} \in \boldsymbol{V}$. Wegen der Unisolvenz der Freiheitsgrade (Satz 3.12) reicht es, die Freiheitsgrade von $\boldsymbol{v}_h := I_F\boldsymbol{v}$ festzulegen, um eine wohldefinierte Abbildung zu definieren.

Für ein Polygon $T \in \mathcal{T}$ werden die Freiheitsgrade der Divergenzmomente

$$D_{\boldsymbol{V}}^{\mathcal{M}_2}(\boldsymbol{v}_h) := D_{\boldsymbol{V}}^{\mathcal{M}_2}(\boldsymbol{v})$$

einfach als Freiheitsgrade der Momente von $\boldsymbol{v}$ gesetzt. Die Freiheitsgrade

$$D_{\boldsymbol{V}}^{\mathcal{M}_1}(\boldsymbol{v}_h) := D_{\boldsymbol{V}}^{\mathcal{M}_1}(J\boldsymbol{v}) \qquad \text{und} \qquad D_{\boldsymbol{V}}^{\mathcal{N}}(\boldsymbol{v}_h) := D_{\boldsymbol{V}}^{\mathcal{N}}(J\boldsymbol{v})$$

werden mit Hilfe des Quasi-Interpolators J aus Lemma 3.24 festgelegt. Auf jeder Kante $E \in \mathcal{E}(T)$ mit Endpunkten P_1^E und P_2^E werden als Werte von $\boldsymbol{v}_h$ an $k-2$ inneren Punkten P_j^E, $j = 3, 4, \ldots, k$ ebenfalls die Werte von $J\boldsymbol{v}$ an den Punkten festgesetzt, das heißt

$$\boldsymbol{v}_h\left(P_j^E\right) := J\boldsymbol{v}\left(P_j^E\right)$$

für alle $j = 3, 4, \ldots, k$. Bleibt noch die Punktauswertung am verbleibenden Punkt P_{k+1}^E. Diese wird durch

$$\begin{aligned}
\boldsymbol{v}_h\left(P_{k+1}^E\right) \cdot \boldsymbol{n}_E &:= \frac{1}{\alpha_{k+1}}\left(\int_E (\boldsymbol{v} - J\boldsymbol{v}) \cdot n_E \,\mathrm{d}E\right) - J\boldsymbol{v} \cdot n_E(P_{j+1}^E) \\
&= \frac{1}{\alpha_{k+1}}\left(\int_E \boldsymbol{v} \cdot \boldsymbol{n}_E \,\mathrm{d}E - \sum_{j=1}^{k} \alpha_j (\boldsymbol{v}_h \cdot \boldsymbol{n}_E)\left(P_j^E\right)\right), \\
\boldsymbol{v}_h(P_{k+1}^E) \cdot \boldsymbol{t}_E &:= \frac{1}{\alpha_{k+1}}\left(\int_E (\boldsymbol{v} - J\boldsymbol{v}) \cdot t_E \,\mathrm{d}E\right) - J\boldsymbol{v} \cdot t_E(P_{j+1}^E)
\end{aligned}$$

festgelegt, wobei $\boldsymbol{n}_E$ der äußere Normalenvektor von T entlang E, $\boldsymbol{t}_E$ ein Tangentialvektor im Punkt P_{k+1}^E ist und α_j das zum Punkt P_j^E, $j = 1, 2, \ldots, k+1$, gehörende Gewicht der Gauß-Lobatto Integrationsformel mit $k+1$ Punkten.

Dann gilt für alle $m_\ell \in \mathbb{M}_{k-1}(T) \setminus \{1\}$ wegen der Definition der Freiheitsgrade

$$\int_T \operatorname{div}\boldsymbol{v}_h \, m_\ell \, \mathrm{d}T = \frac{|T|}{h_T} D_{\boldsymbol{V}}^{\mathcal{M}_2}(\boldsymbol{v}_h) = \frac{|T|}{h_T} D_{\boldsymbol{V}}^{\mathcal{M}_2}(\boldsymbol{v}) = \int_T \operatorname{div}\boldsymbol{v} \, m_\ell \, \mathrm{d}T$$

und für $1 \in \mathbb{M}_{k-1}(T)$

$$\begin{aligned}
\int_T \operatorname{div}\boldsymbol{v}_h \, \mathrm{d}T &= \sum_{E \in \mathcal{E}(T)} \int_E \boldsymbol{v}_h \cdot \boldsymbol{n}_E \, \mathrm{d}E = \sum_{E \in \mathcal{E}(T)} \sum_{j=1}^{k+1} \alpha_j \left(\boldsymbol{v}_h \cdot \boldsymbol{n}_E\right)\left(P_j^E\right) \\
&= \sum_{E \in \mathcal{E}(T)} \int_E \boldsymbol{v} \cdot \boldsymbol{n}_E \, \mathrm{d}E \\
&= \int_T \operatorname{div}\boldsymbol{v} \, \mathrm{d}T,
\end{aligned}$$

wobei im ersten Schritt partielle Integration angewandt und im zweiten Schritt die Gauß-Lobatto Integrationsformel mit $k+1$ Punkten ausgenutzt wurde. Da $\mathbb{M}_{k-1}(T)$ eine Basis für $\mathbb{P}_{k-1}(T) = Q_k^T$ und $T \in \mathcal{T}_h$ beliebig war, folgt also

$$b(\boldsymbol{v} - I_F \boldsymbol{v}, q_h) = 0$$

für alle $q_h \in Q_k$.

Des Weiteren folgt aus der Eigenschaft des Quasi-Interpolators unter der Bedingung $h_T \leq 1$

$$\begin{aligned}
\|J\boldsymbol{v} - \boldsymbol{v}\|_{H^1(\Omega)} &= \sum_{T \in \mathcal{T}_h} \|J\boldsymbol{v} - \boldsymbol{v}\|_{H^1(T)} = \sum_{T \in \mathcal{T}_h} \left(\|J\boldsymbol{v} - \boldsymbol{v}\|_{L^2(T)}^2 + |J\boldsymbol{v} - \boldsymbol{v}|_{H^1(T)}^2\right)^{1/2} \\
&\leq \widetilde{C}_1 \sum_{T \in \mathcal{T}_h} |\boldsymbol{v}|_{H^1(\omega_T)} \leq \widetilde{C}_1 C_{\text{Überlapp}} \sum_{T \in \mathcal{T}_h} |\boldsymbol{v}|_{H^1(T)} \leq \widetilde{C}_1 C_{\text{Überlapp}} \|\boldsymbol{v}\|_{H^1(\Omega)} \\
&= C_1 \|\boldsymbol{v}\|_{H^1(\Omega)}
\end{aligned}$$

mit einer Konstante $0 \leq C_{\text{Überlapp}} \in \mathbb{R}$ und $C_1 := \widetilde{C}_1 C_{\text{Überlapp}}$. Durch die Regularität der Zerlegung des Gebiets gilt, dass die Überlapp-Konstante beschränkt ist, da nur endlich viele Polygone benachbart sein können. Sei für den Beweis eine Nummerierung der lokalen Freiheitsgrade in der Reihenfolge $D_{\boldsymbol{V}}^{\mathcal{N}}$, $D_{\boldsymbol{V}}^{\mathcal{E}}$, $D_{\boldsymbol{V}}^{\mathcal{M}_1}$, $D_{\boldsymbol{V}}^{\mathcal{M}_2}$ und eine lokale Basis φ_j, $j = 1, 2, \ldots, \dim \boldsymbol{V}_k^T$ dadurch gegeben, dass

$$D_{\boldsymbol{V},\ell}(\varphi_j) = \delta_{j,\ell}$$

für alle $j, \ell = 1, 2, \ldots, \dim \boldsymbol{V}_k^T$ gilt. Mit Hilfe dessen kann $\boldsymbol{v}_h - J\boldsymbol{v}$ in eine Summe aus Koeffizienten β_j multipliziert mit der Basisfunktion φ_j für $j = 1, 2, \ldots, \dim \boldsymbol{V}_k^T$ zerlegt werden. Die Koeffizienten sind genau die Freiheitsgrade von $\boldsymbol{v}_h - J\boldsymbol{v}$, weshalb $\beta_j = 0$ für

alle Basisfunktionen, die zu den Knoten beziehungsweise den ersten Momenten gehören. Seien φ_{j^ε}, $j^{\mathcal{E}} = 1, 2, \ldots, 2 \cdot |\mathcal{N}(T)|$ die Basisfunktionen, die zu den Kanten gehören und $\varphi_{j^{\mathcal{M}_1}}$, $j^{\mathcal{M}_1} = 1, 2, \ldots, (k+1)k/2 - 1$ die Basisfunktionen, die zu den Divergenzmomenten gehören. Aus Skalierungsargumenten (s. zum Beispiel [Wei17, dBMR14b]) und den Eigenschaften des Interpolators J folgt $|\beta_{j^\varepsilon}| = |D_{\boldsymbol{V},j^\varepsilon}(\boldsymbol{v}_h - J\boldsymbol{v})| \lesssim \|\boldsymbol{v}\|_{H^1(T)}$, $|\beta_{j^{\mathcal{M}_1}}| = |D_{j^{\mathcal{M}_1}}(\boldsymbol{v}_h - J\boldsymbol{v})| \lesssim \|\boldsymbol{v}\|_{H^1(T)}$ und $\|\varphi_j\|_{H^1(T)} \lesssim 1$, woraus

$$\begin{aligned}
\|I_F\boldsymbol{v} - J\boldsymbol{v}\|_{H^1(T)} &\leq \sum_{j^{\mathcal{E}}=1}^{2|\mathcal{N}(T)|} \left\|\beta_{j^\varepsilon}\varphi_{j^\varepsilon}\right\|_{H^1(T)} + \sum_{j^{\mathcal{M}_1}=1}^{(k+1)(k)/2-1} \left\|\beta_{j^{\mathcal{M}_1}}\varphi_{j^{\mathcal{M}_1}}\right\|_{H^1(T)} \\
&\leq \sum_{j^{\mathcal{E}}=1}^{2|\mathcal{N}(T)|} \left|\beta_{j^\varepsilon}\right| \left\|\varphi_{j^\varepsilon}\right\|_{H^1(T)} + \sum_{j^{\mathcal{M}_1}=1}^{(k+1)(k)/2-1} \left|\beta_{j^{\mathcal{M}_1}}\right| \left\|\varphi_{j^{\mathcal{M}_1}}\right\|_{H^1(T)} \\
&\leq C_2\|\boldsymbol{v}\|_{H^1(T)}
\end{aligned}$$

geschlussfolgert werden kann.

Zusammen folgt nun mit Hilfe der Dreieicksungleichung und $C_F := C_1 + C_2 + 1$

$$\|I_F\boldsymbol{v}\|_{H^1(\Omega)} \leq \|J\boldsymbol{v} - \boldsymbol{v}\|_{H^1(\Omega)} + \|I_F\boldsymbol{v} - J\boldsymbol{v}\|_{H^1(\Omega)} + \|\boldsymbol{v}\|_{H^1(\Omega)} \leq C_F\|\boldsymbol{v}\|_{H^1(\Omega)},$$

was den Beweis abschließt. □

Als Konsequenz des vorherigen Satzes folgt die eindeutige Existenz einer diskreten Lösung.

Proposition 3.26. *Für das diskrete Problem aus Gleichungen* (3.24a) *und* (3.24b) *existiert eine eindeutige Lösung.*

Beweis. Für $X = \boldsymbol{V}$, $Y = Q$, $X_h = \boldsymbol{V}_k$, $Y_h = Q_k$, a_h und b aus Gleichungen (3.22a) und (3.22b), $F = (\boldsymbol{f}_h, \cdot)$ und $G = 0$ sind alle Voraussetzungen aus Proposition 3.22 erfüllt.

Der vorherige Satz zeigt wegen des Fortin-Kriteriums, dass die *LBB*-Bedingung erfüllt ist. Außerdem folgt sofort aus der Stabilitätseigenschaft (3.13) der Bilinearform a_h, dass die Bilinearform elliptisch auf $Z_h = \{\, \boldsymbol{v}_h \in \boldsymbol{V}_k \,:\, b(\boldsymbol{v}_h, q_h) = 0 \text{ für alle } q_h \in Q_k \,\}$ ist. Zu guter Letzt existiert nach Satz 2.19 für jedes $q_h \in Q_k$ ein $\boldsymbol{v} \in \boldsymbol{V}$, so dass $\mathrm{div}\boldsymbol{v} = q_h$. Daher wird nun $\boldsymbol{v}_h := I_F\boldsymbol{v}$ mit dem Fortin-Operator aus vorherigem Satz gewählt, woraus wegen $q_h \neq 0$

$$b(\boldsymbol{v}_h, q_h) = b(\boldsymbol{v}, q_h) = \|q_h\|^2_{L^2(\Omega)} > 0$$

folgt. □

Es kann an dieser Stelle ebenfalls wegen der diskreten inf-sup Bedingung (3.25) und wegen Gleichung (3.10) gefolgert werden, dass

$$\mathrm{div}\boldsymbol{V}_k = Q_k.$$

3.2.3. A-priori Konvergenztheorie für die Virtuelle-Elemente-Methode

Nun, da Existenz und Eindeutigkeit der Lösung des diskreten Problems (3.24a), (3.24b) gezeigt sind, stellt sich die Frage, ob die diskrete Lösung gegen die kontinuierliche Lösung von (2.2a), (2.2b) für $h \to 0$ konvergiert und gegebenenfalls mit welcher Potenz von h.

Neben der Existenz des Quasi-Interpolators aus Lemma 3.24 wird ebenfalls eine Interpolation als Polynom vom Grade maximal k benötigt. Basierend auf [BS02, S. 106 f.] wird in [dLV17] ein Interpolator mit folgenden Eigenschaften angegeben.

Lemma 3.27 (Polynomielle Approximation nach Scott-Dupont)**.** *Sei $T \in \mathcal{T}_h$. Für alle $\boldsymbol{v} \in [H^{s+1}(T)]^2$ mit $0 \leq s \leq k$ existiert ein Polynom $I_\pi \boldsymbol{v} \in [\mathbb{P}_k(T)]^2$, so dass*

$$\|\boldsymbol{v} - I_\pi \boldsymbol{v}\|_{L^2(T)} + h_T |\boldsymbol{v} - I_\pi \boldsymbol{v}|_{H^1(T)} \leq C\, h_T^{s+1} |\boldsymbol{v}|_{H^{s+1}(T)}. \tag{3.30}$$

Mit Hilfe des Quasi-Interpolators und des polynomiellen Interpolators lassen sich einige kleinere Ungleichungen beweisen, die dann für die a-priori Fehlerabschätzung benötigt werden.

Lemma 3.28. *Sei $h_{\mathcal{T}} \in \mathbb{P}_0(\mathcal{T}_h)$ definiert durch $h_{\mathcal{T}|T} := h_T$ für alle $T \in \mathcal{T}_h$. Für alle $\boldsymbol{v} \in [H^{k+1}(\Omega)]^2$ und für alle $\boldsymbol{q} \in [H^k(\Omega)]^2$ gelten*

$$\inf_{\boldsymbol{v}_h \in \boldsymbol{V}_k} \|\boldsymbol{v} - \boldsymbol{v}_h\|_{H^1(\Omega)} \leq C |h_{\mathcal{T}}^k \boldsymbol{v}|_{H^{k+1}(\Omega)} \tag{3.31}$$

und

$$\inf_{\boldsymbol{q}_h \in [\mathbb{P}_k(\mathcal{T}_h)]^2} \|\boldsymbol{q} - \boldsymbol{q}_h\|_{L^2(\Omega)} \leq C |h_{\mathcal{T}}^k \boldsymbol{q}|_{H^k(\Omega)}. \tag{3.32}$$

Beweis. Aus der Eigenschaft des Infimums, dem Interpolator J und seiner Fehlerabschätzung aus Lemma 3.24 folgt mit $h_T \leq 1$ für alle $T \in \mathcal{T}_h$

$$\begin{aligned}
\inf_{\boldsymbol{v}_h \in \boldsymbol{V}_k} \|\boldsymbol{v} - \boldsymbol{v}_h\|_{H^1(\Omega)} \leq \|\boldsymbol{v} - J\boldsymbol{v}\|_{H^1(\Omega)} &= \sum_{T \in \mathcal{T}_h} \left(\frac{1}{h_T}^2 \|\boldsymbol{v} - J\boldsymbol{v}\|_{L^2(T)}^2 + |\boldsymbol{v} - J\boldsymbol{v}|_{H^1(T)}^2 \right)^{1/2} \\
&\leq \widetilde{C}_1 C_{\text{Überlapp}} |h_{\mathcal{T}}^k \boldsymbol{v}|_{H^{k+1}(\Omega)} \\
&\leq C_1 |h_{\mathcal{T}}^k \boldsymbol{v}|_{H^{k+1}(\Omega)},
\end{aligned}$$

wobei $C_1 := \widetilde{C}_1 C_{\text{Überlapp}}$ wegen der Regularität der Zerlegung eine beschränkte Konstante ist.

Analog folgt auch die zweite Aussage, wobei statt des Quasi-Interpolators der polynomielle Interpolator aus Lemma 3.27 benutzt wird. □

Es kann außerdem noch bemerkt werden, dass wenn $\boldsymbol{u} \in \boldsymbol{V}$ die Geschwindigkeitslösung des kontinuierlichen Problems (2.2a)-(2.2b) ist, dann ist $\boldsymbol{u}$ auch ein Element von

$$\boldsymbol{Z} := \{\, \boldsymbol{v} \in \boldsymbol{V} \,:\, b(\boldsymbol{v}, q) = 0 \text{ für alle } q \in Q \,\}$$

und löst daher auch

$$a(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{f}, \boldsymbol{v})_{L^2(\Omega)} \qquad \text{für alle } \boldsymbol{v} \in \boldsymbol{Z}. \tag{3.33}$$

Analog gilt für die diskrete Geschwindigkeitslösung $\boldsymbol{u}_h \in \boldsymbol{V}_k$, dass sie ein Element von

$$\boldsymbol{Z}_k := \{\, \boldsymbol{v}_h \in \boldsymbol{V}_k \,:\, b(\boldsymbol{v}_h, q_h) = 0 \text{ für alle } q_h \in Q_k \,\}$$

ist und daher

$$a_h(\boldsymbol{u}_h, \boldsymbol{v}_h) = (\boldsymbol{f}_h, \boldsymbol{v}_h)_{L^2(\Omega)} \qquad \text{für alle } \boldsymbol{v}_h \in \boldsymbol{Z}_k \tag{3.34}$$

löst.

Nun lässt sich zeigen, dass sich die Konvergenzordnungen für den diskreten Raum $\boldsymbol{V}_k$ auf den Raum $\boldsymbol{Z}_k$ übertragen.

Lemma 3.29 (Konvergenzordnung für Räume mit Nebenbedingungen)**.** *Für alle $\boldsymbol{z} \in \boldsymbol{Z}$ gilt*

$$\inf_{\boldsymbol{z}_h \in \boldsymbol{Z}_k} \|\boldsymbol{z} - \boldsymbol{z}_h\|_{H^1(\Omega)} \leq C \inf_{\boldsymbol{v}_h \in \boldsymbol{V}_k} \|\boldsymbol{z} - \boldsymbol{v}_h\|_{H^1(\Omega)}.$$

Beweis. Sei $\boldsymbol{w}_h \in \boldsymbol{V}_k$. Dann gilt mit dem Fortin-Operator I_F, dass $I_F \boldsymbol{w}_h = \boldsymbol{w}_h$ und aus der Beschränktheit des Fortin-Operators folgt mit der Dreiecksungleichung für alle $\boldsymbol{z} \in \boldsymbol{Z}$

$$\|\boldsymbol{z} - I_F \boldsymbol{z}\|_{H^1(\Omega)} \leq \|\boldsymbol{z} - \boldsymbol{w}_h\|_{H^1(\Omega)} + \|I_F(\boldsymbol{w}_h - \boldsymbol{z})\|_{H^1(\Omega)} \leq (1 + C_F)\|\boldsymbol{z} - \boldsymbol{w}_h\|_{H^1(\Omega)}.$$

Da $\boldsymbol{w}_h \in \boldsymbol{V}_k$ beliebig war, gilt obige Gleichung auch für das Infimum, woraus

$$\|\boldsymbol{z} - I_F \boldsymbol{z}\|_{H^1(\Omega)} \leq (1 + C_F) \inf_{\boldsymbol{v}_h \in \boldsymbol{V}_k} \|\boldsymbol{z} - \boldsymbol{v}_h\|_{H^1(\Omega)}$$

folgt. Darüber hinaus gilt für alle $\boldsymbol{z} \in \boldsymbol{Z}$ wegen der Definition des Fortin-Operators und $\boldsymbol{z} \in \boldsymbol{Z}$, dass für alle $q_h \in Q_h$

$$b(I_F \boldsymbol{z}, q_h) = b(I_F \boldsymbol{z} - \boldsymbol{z}, q_h) + b(\boldsymbol{z}, q_h) = 0 + 0 = 0$$

ist, woraus $I_F \boldsymbol{z} \in \boldsymbol{Z}_h$ gefolgert werden kann.

Zusammenfassend gilt also

$$\inf_{\boldsymbol{z}_h \in \boldsymbol{Z}_k} \|\boldsymbol{z} - \boldsymbol{z}_h\|_{H^1(\Omega)} \leq \|\boldsymbol{z} - I_F \boldsymbol{z}\|_{H^1(\Omega)} \leq (1 + C_F) \inf_{\boldsymbol{v}_h \in \boldsymbol{V}_k} \|\boldsymbol{z} - \boldsymbol{v}_h\|_{H^1(\Omega)},$$

was den Beweis beendet. □

Es bleibt für den Beweis der a-priori Fehlerabschätzung nur noch offen, wie sich der Fehler in der Diskretisierung der rechten Seite verhält. Dies beschreibt folgendes Lemma.

Lemma 3.30 (Diskretisierungsfehler der rechten Seite)**.** *Sei $\mathcal{F}_h$ die kleinste positive Zahl, so dass für die kontinuierliche rechte Seite $\boldsymbol{f} \in [L^2(\Omega)]^2$ des Problems (2.2a)-(2.2b) und die diskrete rechte Seite $\boldsymbol{f}_h \in [\mathbb{P}_{k-2}(\Omega)]^2$ des diskreten Problems (3.24a)-(3.24b)*

$$|(\boldsymbol{f}_h - \boldsymbol{f}, \boldsymbol{v}_h)_{L^2(\Omega)}| \leq \mathcal{F}_h |\boldsymbol{v}_h|_{H^1(\Omega)} \qquad \text{für alle } \boldsymbol{v}_h \in \boldsymbol{V}_k$$

gilt.

Ist $\boldsymbol{f} \in [H^{k-1}(\Omega)]^2$, dann gilt

$$\mathcal{F}_h \lesssim |h_{\mathcal{T}}^k \boldsymbol{f}|_{H^{k-1}(\Omega)}.$$

Beweis. Der Beweis folgt dem Beweis aus [dBC+13]. Durch Approximationsabschätzungen auf sternförmigen Gebieten und L^2-Orthogonalität folgt für alle $\boldsymbol{v}_h \in \boldsymbol{V}_k$

$$\begin{aligned} |(\boldsymbol{f}_h - \boldsymbol{f}, \boldsymbol{v}_h)_{L^2(\Omega)}| &= \sum_{T \in \mathcal{T}_h} \int_T (P_{k-2}^T \boldsymbol{f} - \boldsymbol{f}) \boldsymbol{v}_h \,\mathrm{d}T = \sum_{T \in \mathcal{T}_h} \int_T (P_{k-2}^T \boldsymbol{f} - \boldsymbol{f})(\boldsymbol{v}_h - P_0^T \boldsymbol{v}) \,\mathrm{d}T \\ &\lesssim \sum_{T \in \mathcal{T}_h} h_T^{k-1} |\boldsymbol{f}|_{H^{k-1}(T)} h_T |\boldsymbol{v}|_{H^1(T)} = |h_{\mathcal{T}} \boldsymbol{f}|_{H^{k-1}(\Omega)} |\boldsymbol{v}_h|_{H^1(\Omega)}, \end{aligned}$$

was die Behauptung zeigt. □

Nun sind alle Vorüberlegungen abgeschlossen, um die Fehlerabschätzung für die Geschwindigkeit zu beweisen.

Satz 3.31 (A-priori Geschwindigkeitsfehler)**.** *Seien $\boldsymbol{u} \in \boldsymbol{Z}$ die Lösung von Problem (3.33) und $\boldsymbol{u}_h \in \boldsymbol{Z}_k$ die Lösung des diskreten Problems (3.34). Dann lässt sich der Geschwindigkeitsfehler durch*

$$\|\boldsymbol{u} - \boldsymbol{u}_h\|_{H^1(\Omega)} \lesssim \inf_{\boldsymbol{v}_h \in \boldsymbol{V}_k} \|\boldsymbol{u} - \boldsymbol{v}_h\|_{H^1(\Omega)} + \inf_{\boldsymbol{v}_\pi \in [\mathbb{P}_k(\mathcal{T}_h)]^2} \|\boldsymbol{u} - \boldsymbol{v}_\pi\|_{h,H^1(\Omega)} + \frac{1}{\nu} \mathcal{F}_h \tag{3.35}$$

abschätzen, wobei die gebrochene H^1-Norm für alle stückweisen $H^1(\mathcal{T}_h)$ Funktionen $\boldsymbol{v}_h$ durch

$$\|\boldsymbol{v}_h\|_{h,H^1(\Omega)} := \sum_{T \in \mathcal{T}_h} \|\boldsymbol{v}_h\|_{H^1(T)}$$

definiert ist.

Beweis. Sei $\boldsymbol{w}_h \in \boldsymbol{Z}_k$ die Bestapproximation bezüglich der $\|\cdot\|_{H^1(\Omega)}$-Norm von $\boldsymbol{u}$ in $\boldsymbol{Z}_k$. Dann folgt wegen der Orthogonalität von $\boldsymbol{u} - \boldsymbol{w}_h$ auf $\boldsymbol{Z}_k$, dass

$$\|\boldsymbol{u} - \boldsymbol{u}_h\|_{H^1(\Omega)}^2 = \|\boldsymbol{u} - \boldsymbol{w}_h\|_{H^1(\Omega)}^2 + \|\boldsymbol{w}_h - \boldsymbol{u}_h\|_{H^1(\Omega)}^2, \tag{3.36}$$

was es jetzt einzeln abzuschätzen gilt.

Für den ersten Term ergibt sich

$$\|\boldsymbol{u} - \boldsymbol{w}_h\|_{H^1(\Omega)} = \inf_{\boldsymbol{z}_h \in \boldsymbol{Z}_k} \|\boldsymbol{u} - \boldsymbol{z}_h\|_{H^1(\Omega)} \lesssim \inf_{\boldsymbol{v}_h \in \boldsymbol{V}_k} \|\boldsymbol{u} - \boldsymbol{v}_h\|_{H^1(\Omega)}, \tag{3.37}$$

was aus der Definition der Bestapproximation und Lemma 3.29 folgt.

Für den zweiten Teil wird $\delta := \boldsymbol{w}_h - \boldsymbol{u}_h$ definiert. Dann gilt mit der Konstante α_* aus Gleichung (3.13) und der Äquivalenz der H^1-Norm und -seminorm, dass

$$\nu\,\alpha_*\|\delta\|^2_{H^1(\Omega)} \approx \nu\,\alpha_*|\delta|^2_{H^1(\Omega)} = \alpha_* a(\delta,\delta) \leq a_h(\delta,\delta) = a_h(\boldsymbol{w}_h,\delta) - (\boldsymbol{f}_h,\delta)_{L^2(\Omega)},$$

wobei im letzten Schritt Gleichung (3.34) ausgenutzt wird. Durch Aufspaltung der Bilinearform in die jeweiligen lokalen Beiträge, Addition und Subtraktion einer stückweisen $[\mathbb{P}_k(T)]^2$-Funktion $\widetilde{\boldsymbol{u}}_\pi \in [\mathbb{P}_k(\mathcal{T}_h)]^2$ und $\boldsymbol{u}$ folgt dann

$$\begin{aligned} a_h(\boldsymbol{w}_h,\delta) - (\boldsymbol{f}_h,\delta)_{L^2(\Omega)} &= \sum_{T\in\mathcal{T}_h} \Big(a_h^T(\boldsymbol{w}_h - \widetilde{\boldsymbol{u}}_\pi,\delta) + a_h^T(\widetilde{\boldsymbol{u}}_\pi,\delta)\Big) - (\boldsymbol{f}_h,\delta)_{L^2(\Omega)} \\ &= \sum_{T\in\mathcal{T}_h} \Big(a_h^T(\boldsymbol{w}_h - \widetilde{\boldsymbol{u}}_\pi,\delta) + a^T(\widetilde{\boldsymbol{u}}_\pi - \boldsymbol{u},\delta)\Big) + (\boldsymbol{f} - \boldsymbol{f}_h,\delta)_{L^2(\Omega)}, \end{aligned}$$

was sich durch die k-Konsistenz Bedingung (3.12) und Gleichung (3.33) sowie der Linearität der Bilinearformen wie auch des L^2-Skalarprodukts ergibt. Die Terme auf der rechten Seite können jetzt sowohl durch die Stetigkeit der kontinuierlichen als auch der diskreten Bilinearform (Bemerkung 3.13) abgeschätzt werden, woraus mit einer Dreiecksungleichung

$$\begin{aligned} \nu\,\alpha_*\|\delta\|^2_{H^1(\Omega)} &\lesssim \sum_{T\in\mathcal{T}_h} \Big(a_h^T(\boldsymbol{w}_h - \widetilde{\boldsymbol{u}}_\pi,\delta) + a^T(\widetilde{\boldsymbol{u}}_\pi - \boldsymbol{u},\delta)\Big) + (\boldsymbol{f} - \boldsymbol{f}_h,\delta)_{L^2(\Omega)} \\ &\leq \nu \left(\sum_{T\in\mathcal{T}_h} \Big(|\boldsymbol{w}_h - \widetilde{\boldsymbol{u}}_\pi|_{H^1(T)} + |\widetilde{\boldsymbol{u}}_\pi - \boldsymbol{u}|_{H^1(T)}\Big)\right) |\delta|_{H^1(\Omega)} + \mathcal{F}_h|\delta|_{H^1(\Omega)} \\ &\leq \nu \left(\sum_{T\in\mathcal{T}_h} \Big(|(\boldsymbol{w}_h - \boldsymbol{u}|_{H^1(T)} + 2|\widetilde{\boldsymbol{u}}_\pi - \boldsymbol{u}|_{H^1(T)}\Big)\right) |\delta|_{H^1(\Omega)} + \mathcal{F}_h|\delta|_{H^1(\Omega)} \\ &= \nu \Big(|(\boldsymbol{w}_h - \boldsymbol{u}|_{H^1(\Omega)} + 2|\widetilde{\boldsymbol{u}}_\pi - \boldsymbol{u}|_{h,H^1(\Omega)}\Big) |\delta|_{H^1(\Omega)} + \mathcal{F}_h|\delta|_{H^1(\Omega)} \end{aligned}$$

folgt. Da $\widetilde{\boldsymbol{u}}_\pi \in [\mathbb{P}_k(\mathcal{T}_h)]^2$ beliebig war, gilt obige Gleichung ebenfalls für das Infimum. Auf den ersten Summanden kann jetzt Gleichung (3.37) angewandt werden, womit nach Teilung durch $|\delta|_{H^1(\Omega)} \approx \|\delta\|_{H^1(\Omega)}$ und ν

$$\|\boldsymbol{w}_h - \boldsymbol{u}_h\|_{H^1(\Omega)} \lesssim \inf_{\boldsymbol{v}_h\in\boldsymbol{V}_k} \|\boldsymbol{u} - \boldsymbol{v}_h\|_{H^1(\Omega)} + \inf_{\boldsymbol{v}_\pi\in[\mathbb{P}_k(\mathcal{T}_h)]^2} \|\boldsymbol{u} - \boldsymbol{v}_\pi\|_{h,H^1(\Omega)} + \frac{1}{\nu}\,\mathcal{F}_h \quad (3.38)$$

folgt. Die Gleichungen (3.37) und (3.38) angewandt auf Gleichung (3.36) beenden damit den Beweis des Satzes. □

Die gleiche Konvergenzordnung gilt auch für den diskreten Druck, was im Folgenden gezeigt wird.

Satz 3.32 (A-priori Druckfehler)**.** *Seien $(\boldsymbol{u}, p) \in \boldsymbol{V} \times Q$ die Lösung des kontinuierlichen Problems (2.2a)-(2.2b) und $(\boldsymbol{u}_h, p_h) \in \boldsymbol{V}_k \times Q_k$ die Lösung des diskreten Problems (3.24a)-(3.24b). Dann lässt sich der Fehler des Drucks durch*

$$\begin{aligned}\|p - p_h\|_{L^2(\Omega)} \lesssim \nu \inf_{\boldsymbol{v}_h \in \boldsymbol{V}_k} \|\boldsymbol{u} - \boldsymbol{v}_h\|_{H^1(\Omega)} + \nu \inf_{\boldsymbol{v}_\pi \in [\mathbb{P}_k(\mathcal{T}_h)]^2} \|\boldsymbol{u} - \boldsymbol{v}_\pi\|_{h,H^1(\Omega)} \\ + \inf_{q_h \in Q_k} \|p - q_h\|_{L^2(\Omega)} + \mathcal{F}_h\end{aligned}$$

abschätzen.

Beweis. Der Beweis kann wie folgt in [dLV17] gefunden werden. Sei $q_h \in Q_k$. Dann folgt aus der diskreten inf-sup Bedingung (3.25)

$$\beta_h \|p_h - q_h\|_{L^2(\Omega)} \leq \sup_{\boldsymbol{v}_h \in \boldsymbol{V}_k \setminus \{0\}} \frac{b(\boldsymbol{v}_h, p_h - q_h)}{\|\boldsymbol{v}_h\|_{H^1(\Omega)}} = \sup_{\boldsymbol{v}_h \in \boldsymbol{V}_k \setminus \{0\}} \frac{b(\boldsymbol{v}_h, p_h - p) + b(\boldsymbol{v}_h, p - q_h)}{\|\boldsymbol{v}_h\|_{H^1(\Omega)}},$$

was in zwei Teilen abgeschätzt werden soll.

Es wird mit dem ersten Summanden begonnen. Da $(\boldsymbol{u}, p)$ Gleichung (2.2a) für alle $\boldsymbol{v} \in \boldsymbol{V}$ löst, löst das Paar die Gleichung insbesondere auch für alle $\boldsymbol{v}_h \in \boldsymbol{V}_k \subset \boldsymbol{V}$. Daher folgt

$$\begin{aligned}|b(\boldsymbol{v}_h, p_h - p)| &= |(\boldsymbol{f}_h - \boldsymbol{f}, \boldsymbol{v}_h) + (a(\boldsymbol{u}, \boldsymbol{v}_h) - a_h(\boldsymbol{u}_h, \boldsymbol{v}_h))\,| \\ &\leq \mathcal{F}_h \|\boldsymbol{v}_h\|_{H^1(\Omega)} + |a(\boldsymbol{u}, \boldsymbol{v}_h) - a_h(\boldsymbol{u}_h, \boldsymbol{v}_h)|,\end{aligned}$$

wobei im letzten Schritt die Definition von $\mathcal{F}_h$ aus Lemma 3.30 benutzt wird. Für die beiden a-Terme, werden die k-Konsistenz Bedingung (3.12), die Stetigkeit der Bilinearform a_h und eine Dreiecksungleichung ausgenutzt, um

$$\begin{aligned}a(\boldsymbol{u}, \boldsymbol{v}_h) - a_h(\boldsymbol{u}_h, \boldsymbol{v}_h) &= \sum_{T \in \mathcal{T}_h} \left(a^T(\boldsymbol{u}, \boldsymbol{v}_h) - a_h^T(\boldsymbol{u}_h, \boldsymbol{v}_h) \right) \\ &= \sum_{T \in \mathcal{T}_h} \left(a^T(\boldsymbol{u} - \widetilde{\boldsymbol{u}}_\pi, \boldsymbol{v}_h) - a_h^T(\boldsymbol{u} - \boldsymbol{u} + \boldsymbol{u}_h - \widetilde{\boldsymbol{u}}_\pi, \boldsymbol{v}_h) \right) \\ &\leq \nu \sum_{T \in \mathcal{T}_h} \left(|\boldsymbol{u} - \widetilde{\boldsymbol{u}}_\pi|_{H^1(T)} + |\boldsymbol{u} - \widetilde{\boldsymbol{u}}_\pi + \boldsymbol{u}_h - \boldsymbol{u}|_{H^1(T)} \right) |\boldsymbol{v}_h|_{H^1(T)} \\ &\leq \nu \sum_{T \in \mathcal{T}_h} \left(2|\boldsymbol{u} - \widetilde{\boldsymbol{u}}_\pi|_{H^1(T)} + |\boldsymbol{u} - \boldsymbol{u}_h|_{H^1(T)} \right) |\boldsymbol{v}_h|_{H^1(T)} \\ &\lesssim \nu \left(|\boldsymbol{u} - \widetilde{\boldsymbol{u}}_\pi|_{h,H^1(\Omega)} + |\boldsymbol{u} - \boldsymbol{u}_h|_{H^1(\Omega)} \right) |\boldsymbol{v}_h|_{H^1(\Omega)}\end{aligned}$$

zu erhalten, wobei $\widetilde{\boldsymbol{u}}_\pi$ eine stückweise Approximation von $\boldsymbol{u}$ in $[\mathbb{P}_k(\mathcal{T}_h)]^2$ ist. Da $\widetilde{\boldsymbol{u}}_\pi$ beliebig war, gilt obige Abschätzung auch für das Infimum, wodurch mit Satz 3.31

$$\begin{aligned}|b(\boldsymbol{v}_h, p_h - p)| \lesssim \nu \left(\inf_{\boldsymbol{v}_h \in \boldsymbol{V}_k} \|\boldsymbol{u} - \boldsymbol{v}_h\|_{H^1(\Omega)} + \inf_{\boldsymbol{v}_\pi \in [\mathbb{P}_k(\mathcal{T}_h)]^2} \|\boldsymbol{u} - \boldsymbol{v}_\pi\|_{h,H^1(\Omega)} \right) |\boldsymbol{v}_h|_{H^1(\Omega)} \\ + \mathcal{F}_h |\boldsymbol{v}_h|_{H^1(\Omega)}\end{aligned}$$

folgt.

Für den zweiten Summanden gilt

$$|b(\boldsymbol{v}_h, p - q_h)| \lesssim \|p - q_h\|_{L^2(\Omega)} \|\boldsymbol{v}_h\|_{H^1(\Omega)},$$

wodurch mit vorheriger Abschätzung

$$\begin{aligned}\|p_h - q_h\|_{L^2(\Omega)} \lesssim \nu \inf_{\boldsymbol{v}_h \in \boldsymbol{V}_k} \|\boldsymbol{u} - \boldsymbol{v}_h\|_{H^1(\Omega)} + \nu \inf_{\boldsymbol{v}_\pi \in [\mathbb{P}_k(\mathcal{T}_h)]^2} \|\boldsymbol{u} - \boldsymbol{v}_\pi\|_{h,H^1(\Omega)} + \mathcal{F}_h \\ + \|p - q_h\|_{L^2(\Omega)}\end{aligned}$$

ergibt. Mit der Dreieicksungleichung lässt sich damit

$$\begin{aligned}\|p - p_h\|_{L^2(\Omega)} \leq& \|p - q_h\|_{L^2(\Omega)} + \|q_h - p_h\|_{L^2(\Omega)} \\ \lesssim& \nu \inf_{\boldsymbol{v}_h \in \boldsymbol{V}_k} \|\boldsymbol{u} - \boldsymbol{v}_h\|_{H^1(\Omega)} + \nu \inf_{\boldsymbol{v}_\pi \in [\mathbb{P}_k(\mathcal{T}_h)]^2} \|\boldsymbol{u} - \boldsymbol{v}_\pi\|_{h,H^1(\Omega)} + \mathcal{F}_h \\ &+ \|p - q_h\|_{L^2(\Omega)}\end{aligned}$$

schließen.

Da q_h beliebig war, gilt die Abschätzung auch für das Infimum, wodurch die Behauptung folgt. □

Ist die kontinuierliche Lösung $(\boldsymbol{u}, p)$ glatt genug, ergibt sich daher mit Hilfe der Interpolationsoperatoren eine a-priori Abschätzung für den Geschwindigkeits- und den Druckfehler.

Korollar 3.33. *Ist $\boldsymbol{f} \in [H^{k-1}(\Omega)]^2$, dann gelten die folgenden Aussagen.*

i. Wenn $\boldsymbol{u} \in [H^{k+1}(\Omega)]^2$, dann gilt

$$\|\boldsymbol{u} - \boldsymbol{u}_h\|_{H^1(\Omega)} \lesssim |h_{\mathcal{T}}^k \boldsymbol{u}|_{H^{k+1}(\Omega)} + \frac{1}{\nu} |h_{\mathcal{T}}^k \boldsymbol{f}|_{H^{k-1}(\Omega)}.$$

ii. Sind $(\boldsymbol{u}, p) \in [H^{k+1}(\Omega)]^2 \times H^k(\Omega)$, folgt

$$\|p - p_h\|_{L^2(\Omega)} \lesssim |h_{\mathcal{T}}^k \boldsymbol{f}|_{H^{k-1}(\Omega)} + \nu |h_{\mathcal{T}}^k \boldsymbol{u}|_{H^{k+1}(\Omega)} + |h_{\mathcal{T}}^k p|_{H^k(\Omega)}.$$

Beweis. Die erste Behauptung folgt direkt aus einer Kombination von Lemma 3.28 und Lemma 3.30 angewandt auf Satz 3.31. Werden auf Satz 3.32 die Argumente der ersten Behauptung und Gleichung (3.32) angewendet, folgt auch die zweite Behauptung. □

Bemerkung 3.34. Schon hier kann erkannt werden, dass diese Virtuelle-Elemente-Diskretisierung nicht druckrobust ist, da der Geschwindigkeitsfehler von $\boldsymbol{f}$ und damit auch von rotationsfreien Anteilen wie Gradienten in $\boldsymbol{f}$ abhängt.

4. Implementierung der Virtuellen-Elemente-Methode

Grundlegend werden die Virtuellen-Elemente-Methoden wie die Finite-Elemente-Methoden implementiert. Der große Unterschied besteht vor allen Dingen darin, dass die Basisfunktionen nur implizit zur Verfügung stehen. Das hat zur Folge, dass jegliche Berechnungen auf die Freiheitsgrade der Basisfunktionen zurückführbar sein müssen.

Im Folgenden werden zuerst die Datenstrukturen eingeführt, die zur Implementierung der Virtuellen-Elemente-Methode benötigt werden. Daraufhin wird gezeigt, wie die Steifigkeitsmatrix auszurechnen ist.

4.1. Benötigte Datenstrukturen für die Implementierung

Zunächst besteht die Herausforderung in der Darstellung diskrete Zerlegungen des Gebiets Ω im Computer. Die Darstellung in dieser Arbeit orientiert sich an den Datenstrukturen der Finiten-Elemente-Methode aus [ACF99,CGK$^+$10]. Dies geschieht in MATLAB über die folgenden Angaben:

c4n ist ein Vektor der Dimension $|\mathcal{N}|\times$ 2, wobei die j-te Zeile die x_1- und die x_2-Koordinate des j-ten Punkts enthalten (analog zu AFEM).

n4e ist ein Cell-array der Dimension $|\mathcal{T}_h| \times 1$. Einträge sind Vektoren der Dimension $1 \times \mathcal{N}(T)$ mit den Nummern der Knoten der jeweiligen Elemente. Dies ist ein Cell-array, da die Elemente im Allgemeinen unterschiedlich viele Eckpunkte haben. Die Auflistung der Punkte muss entgegen dem Uhrzeigersinn erfolgen, um einen positiven Flächeninhalt zu erhalten (für Dreiecke ähnlich zu AFEM).

n4sDb ist ein Vektor der Dimension Anzahl der Dirichletrandkanten $\times$ 2. Er enthält die Nummern der Knoten, die Eckpunkte einer Dirichletrandkante sind (analog zu AFEM).

n4sNb ist ein Vektor der Dimension Anzahl der Neumannrandkanten $\times$ 2. Dieser Vektor enthält die Nummern der Knoten, die Endpunkte einer Neumannrandkante sind (analog zu AFEM).

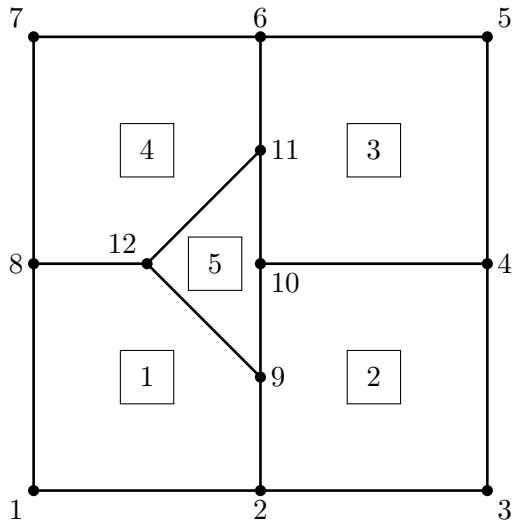


Abbildung 4.1.: Gezeigt ist eine mögliche Zerlegung des Einheitsquadrats $\Omega = (0,1)^2$ in fünf Polygone mit zwölf Knoten. Hängende Knoten wie die Nummern neun, zehn und elf sind genau wie ein Innenwinkel von 180° bei Knoten Nummer zehn explizit erlaubt. Eine Zahl in einem kleinen Quadrat zeigt die Nummer des Elements.

Exemplarisch wird der Vektor `c4n` und das Cell-array `n4e` für die Zerlegung des Einheitsquadrats aus Abbildung 4.1 gezeigt:

$$\texttt{c4n} = \begin{pmatrix} 0 & 0 \\ 0.5 & 0 \\ 1 & 0 \\ 1 & 0.5 \\ 1 & 1 \\ 0.5 & 1 \\ 0 & 1 \\ 0 & 0.5 \\ 0.5 & 0.25 \\ 0.5 & 0.5 \\ 0.5 & 0.75 \\ 0.25 & 0.5 \end{pmatrix} \quad \text{und} \quad \texttt{n4e} = \left\{ \begin{bmatrix} 1 & 2 & 9 & 12 & 8 \end{bmatrix} \begin{bmatrix} 2 & 3 & 4 & 10 & 9 \end{bmatrix} \begin{bmatrix} 4 & 5 & 6 & 11 & 10 \end{bmatrix} \begin{bmatrix} 6 & 7 & 8 & 12 & 11 \end{bmatrix} \begin{bmatrix} 9 & 10 & 11 & 12 \end{bmatrix} \right\}$$

Der j-te Knoten $z_j = (x_j, y_j)$ ist analog zur AFEM durch `c4n(j,:)`$= (x_j, y_j)$ erreichbar.
Das j-te Element $T_j = \operatorname{conv}\{z_k, z_\ell, \ldots, z_n\}$ ist durch die Indices $(k, \ell, \ldots, n)$ festgelegt und durch `n4e{j}`$=(k, \ell, \ldots, n)$ gegeben. Damit sind die Koordinaten der Knoten des j-ten Elements durch `c4n(n4e{j},:)` abrufbar.

Alle sich daraus ergebenden üblichen Datenstrukturen sind analog zu AFEM. Der

Unterschied besteht darin, dass solche Angaben, die von der Anzahl der Knoten beziehungsweise Kanten in einem Element abhängen, in Cell-arrays gespeichert werden, um die unterschiedlichen Längen zu handhaben.

4.2. Implementierung der Bilinearformen und Projektoren

Um eine diskrete Lösung berechnen zu können, müssen sowohl lokale als auch globale Basen gewählt werden. Diese können leider nicht explizit angegeben werden, was allerdings auch nicht notwendig ist. Es reicht für die Berechnung aus, sie implizit über ihre Freiheitsgrade zu definieren.

4.2.1. Implizite Basen und Berechnug der Bilinearformen

Zuerst soll eine Basis für die lokalen Räume angegeben werden. Das wird über die Freiheitsgrade geschehen, weswegen die lokalen Freiheitsgrade nummeriert werden müssen. Dafür sei $\texttt{nDof}^T := \dim \boldsymbol{V}_k^T$ die Anzahl der Freiheitsgrade auf einem Polygon $T \in \mathcal{T}_h$. Dann können die Freiheitsgrade mittels $(D_{V,j})_{j=1}^{\texttt{nDof}^T} \subset \left[\boldsymbol{V}_k^T\right]^*$ nummeriert werden. Dies geschieht in der Reihenfolge $D_{\boldsymbol{V}}^{\mathcal{N}}$, $D_{\boldsymbol{V}}^{\mathcal{E}}$, $D_{\boldsymbol{V}}^{\mathcal{M}_1}$, $D_{\boldsymbol{V}}^{\mathcal{M}_2}$. Das heißt die ersten $2 \cdot |\mathcal{N}(T)|$ Stück sind die Freiheitsgrade an den Knoten, danach die an den Kanten und analog geht es weiter. Die Reihenfolge der Knoten, der Kanten und der Momente orientiert sich dabei sinnvollerweise an der durch `n4e` beziehungsweise durch die Korrespondenz aus Gleichung (3.2) implizierte Ordnung. Die Knoten werden nach der Position in `n4e` nummeriert und die Kanten so, dass zuerst die lokalen Kanten des ersten Elements, dann die des zweiten Elements usw. aufgelistet werden. Lokal liegt dabei die j-te Kante zwischen dem j-ten und dem $(j+1)$-ten Knoten.

Mit Hilfe der lokalen Freiheitsgrade kann nun implizit eine Basis von $\boldsymbol{V}_k^T$ angegeben werden.

Definition 4.1 (Kanonische Basis von $\boldsymbol{V}_k^T$)**.** Implizit ist die kanonische Lagrange-Basis $(\varphi_j)_{j=1}^{\texttt{nDof}^T} \subset \boldsymbol{V}_k^T$ von $\boldsymbol{V}_k^T$ durch die Gleichungen

$$D_{V,j}(\varphi_\ell) = \delta_{j,\ell} \qquad \text{für } j, \ell = 1, 2, \ldots, \texttt{nDof}^T$$

definiert.

Diese Nummerierung der Freiheitsgrade und die Definition der Basen kann auch global durchgeführt werden. Seien dafür $\texttt{nDof} := \dim \boldsymbol{V}_k$ die Anzahl aller Freiheitsgrade des Geschwindigkeitraums. Die Freiheitsgrade werden dann durch

$$(GD_{V,j})_{j=1}^{\texttt{nDof}} \subset [\boldsymbol{V}_k]^*$$

nummeriert, wobei dies wieder der Reihenfolge $GD_{\boldsymbol{V}}^{\mathcal{N}}$, $GD_{\boldsymbol{V}}^{\mathcal{E}}$, $GD_{\boldsymbol{V}}^{\mathcal{M}_1}$, $GD_{\boldsymbol{V}}^{\mathcal{M}_2}$ folgt.

Auch hiermit kann jetzt eine implizite globale Basis angegeben werden.

Definition 4.2 (Kanonische Basis von $\boldsymbol{V}_k$)**.** Durch die Gleichungen

$$GD_{V,j}(\Phi_\ell) = \delta_{j,\ell} \qquad \text{für } j,\ell = 1,2,\ldots,\texttt{nDof}$$

ist implizit die kanonische Basis $(\Phi_j)_{j=1}^{\texttt{nDof}} \subset \boldsymbol{V}_k$ von $\boldsymbol{V}_k$ definiert.

Bemerkung 4.3. Im Gegensatz zu den Ansatzfunktionen bei Finiten-Elemente-Methoden sind die Basisfunktionen nicht explizit bekannt. Entlang des Randes sehen die Basisfunktionen, die zu den Randfreiheitsgraden $D_V^{\mathcal{V}}$ und $D_V^{\mathcal{E}}$ gehören, aus wie die Basisfunktionen der Finiten-Elemente-Methoden. Allerdings sind es im Allgemeinen auch auf Dreiecken nicht die gleichen Funktionen, da diese nicht zwingend divergenzfrei sind.

4.2.2. Berechnung der lokalen Matrizen

Um die Lösung von Gleichungen (3.24a) und (3.24b) berechnen zu können, müssen zuerst die lokalen Bilinearformen für die lokalen Basisfunktionen berechnet werden. Damit und mit der Verbindung von lokalen und globalen Basisfunktionen kann ein globales lineares Gleichungssystem gebildet werden, was die globale Lösung charakterisiert. Zuerst soll also Gleichung (3.19a) für alle Basisfunktionen $\varphi_j, \varphi_\ell \in \boldsymbol{V}_k^T$, $j,\ell = 1,2,\ldots,\texttt{nDof}^T$ berechnet werden, das heißt

$$a_h^T(\varphi_j,\varphi_\ell) = a^T\left(\Pi_k^{\nabla,T}\varphi_j, \Pi_k^{\nabla,T}\varphi_\ell\right) + \nu \sum_{n=1}^{\texttt{nDof}^T} D_{V,n}((\boldsymbol{I}-\Pi_k^{\nabla,T})\varphi_j) D_{V,n}((\boldsymbol{I}-\Pi_k^{\nabla,T})\varphi_\ell). \tag{4.1}$$

Um das zu berechnen, wird die Projektion $\Pi_k^{\nabla,T}\varphi_j$ aus Gleichungen (3.14) und (3.15) für alle $j = 1,2,\ldots,\texttt{nDof}^T$ benötigt. Die Berechnung der Projektion wird jetzt allgemein für alle $\boldsymbol{v}_h$ und damit insbesondere auch für die Basisfunktionen durchgeführt.

Zuerst kann dafür in Gleichung (3.14) $\boldsymbol{q}_k$ nur in $[\mathbb{M}_k(T)]^2$ variiert werden, da $[\mathbb{M}_k(T)]^2$ eine Basis für $[\mathbb{P}_k(T)]^2$ ist. Da $\Pi_k^{\nabla,T}$ nach $[\mathbb{P}_k(T)]^2$ abbildet, kann $\Pi_k^{\nabla,T}\boldsymbol{v}_h$ in Basiselemente $\boldsymbol{m}_j \in [\mathbb{M}_k(T)]^2$ zerlegt werden. Somit kann $\Pi_k^{\nabla,T}\boldsymbol{v}_h$ durch

$$\Pi_k^{\nabla,T}\boldsymbol{v}_h = \sum_{j=1}^{2\pi_k} s_j \boldsymbol{m}_j, \tag{4.2}$$

mit Koeffizienten $s_j \in \mathbb{R}$ für alle j dargestellt werden.

Wird die Darstellung aus Gleichung (4.2) nun in Gleichungen (3.14) und (3.15) eingesetzt ergibt sich das äquivalente Gleichungssystem

$$\sum_{j=1}^{2\pi_k} s_j\, a^T(\boldsymbol{m}_j,\boldsymbol{m}_\ell) = a^T(\boldsymbol{v}_h,\boldsymbol{m}_\ell) \qquad \text{für alle } \ell = 1,2,\ldots,2\pi_k \tag{4.3}$$

$$\sum_{j=1}^{2\pi_k} s_j P_0^T \boldsymbol{m}_j = P_0^T \boldsymbol{v}_h. \tag{4.4}$$

Für $\ell = 1, 2$ ist die erste Gleichung jeweils $0 = 0$, da in beiden Fällen die auftretende Jacobi-Matrix $D\boldsymbol{m}_\ell$ identisch zur Nullmatrix ist. Allerdings liefert die zweite Gleichung zwei linear unabhängige Gleichungen, so dass ein lineares Gleichungssystem in den Unbekannten s_j, $j = 1, 2, \ldots, 2\pi_k$, mit $2\pi_k$ linear unabhängigen Gleichungen entsteht. In Matrizenschreibweise ergibt sich dann

$$\underbrace{\begin{pmatrix} P_0^T\boldsymbol{m}_1 & P_0^T\boldsymbol{m}_2 & P_0^T\boldsymbol{m}_3 & \ldots & P_0^T\boldsymbol{m}_{2\pi_k} \\ 0 & 0 & a^T(\boldsymbol{m}_3, \boldsymbol{m}_3) & \ldots & a^T(\boldsymbol{m}_3, \boldsymbol{m}_{2\pi_k}) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & a^T(\boldsymbol{m}_{2\pi_k}, \boldsymbol{m}_3) & \ldots & a^T(\boldsymbol{m}_{2\pi_k}, \boldsymbol{m}_{2\pi_k}) \end{pmatrix}}_{=:G} \underbrace{\begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_{2\pi_k} \end{pmatrix}}_{=:\boldsymbol{s}} = \underbrace{\begin{pmatrix} P_0^T\boldsymbol{v}_h \\ a^T(\boldsymbol{v}_h, \boldsymbol{m}_3) \\ \vdots \\ a^T(\boldsymbol{v}_h, \boldsymbol{m}_{2\pi_k}) \end{pmatrix}}_{:=\boldsymbol{c}} \tag{4.5}$$

oder kompakter dargestellt

$$G\boldsymbol{s} = \boldsymbol{c}. \tag{4.6}$$

Dass dieses Gleichungssystem überhaupt mit den Freiheitsgraden berechenbar ist, ist in Abschnitt 3.2.1 begründet.

Zur Berechnung der Projektion $\Pi_k^{\nabla,T}\varphi_j$ als Element von $[\mathbb{P}_k(T)]^2$ für alle Basisfunktionen $\varphi_j \in \boldsymbol{V}_k^T$ für $j = 1, 2, \ldots, \texttt{nDof}^T$ wird dafür Gleichung (4.6) benutzt. Seien daher für $j = 1, 2, \ldots, \texttt{nDof}^T$

$$\boldsymbol{s}^{(j)} := \begin{pmatrix} s_1^{(j)} \\ s_2^{(j)} \\ \vdots \\ s_{2\pi_k}^{(j)} \end{pmatrix} \quad \text{und} \quad \boldsymbol{c}^{(j)} := \begin{pmatrix} P_0^T\varphi_j \\ a^T(\varphi_j, \boldsymbol{m}_3) \\ \vdots \\ a^T(\varphi_j, \boldsymbol{m}_{2\pi_k}) \end{pmatrix}$$

die Koeffizienten der j-ten Basisfunktion bezüglich der Zerlegung in die Monombasis $\Pi_k^{\nabla,T}\varphi_j = \sum_{\ell=1}^{\texttt{nDof}^T} c_\ell^{(j)}\boldsymbol{m}_\ell$ beziehungsweise die rechte Seite zur Berechnung eben dieser Koeffizienten. Werden

$$S^* := \begin{pmatrix} \boldsymbol{s}^{(1)} & \boldsymbol{s}^{(2)} & \ldots & \boldsymbol{s}^{(\texttt{nDof}^T)} \end{pmatrix} \in \mathbb{R}^{2\pi_k \times \texttt{nDof}^T}$$

und die $2\pi_k \times \texttt{nDof}^T$-dimensionale Matrix

$$\begin{aligned} C &:= \begin{pmatrix} \boldsymbol{c}^{(1)} & \boldsymbol{c}^{(2)} & \ldots & \boldsymbol{c}^{(\texttt{nDof}^T)} \end{pmatrix} \\ &= \begin{pmatrix} P_0^T\varphi_1 & P_0^T\varphi_2 & \ldots & P_0^T\varphi_{\texttt{nDof}^T} \\ a^T(\varphi_1, \boldsymbol{m}_3) & a^T(\varphi_2, \boldsymbol{m}_3) & \ldots & a^T(\varphi_{\texttt{nDof}^T}, \boldsymbol{m}_3) \\ \vdots & \vdots & \ddots & \vdots \\ a^T(\varphi_1, \boldsymbol{m}_{2\pi_k}) & a^T(\varphi_2, \boldsymbol{m}_{2\pi_k}) & \ldots & a^T(\varphi_{\texttt{nDof}^T}, \boldsymbol{m}_{2\pi_k}) \end{pmatrix} \end{aligned}$$

definiert, ergibt sich für die Berechnung der Matrixdarstellung S^* des Operators $\Pi_k^{\nabla,T}$ als Abbildung nach $[\mathbb{P}_k(T)]^2$

$$S^* = G^{-1}C, \tag{4.7}$$

wobei $G \in \mathbb{R}^{2\pi_k \times 2\pi_k}$ aus Gleichung (4.5) benutzt wird. In der j-ten Spalte von S^* stehen also die Koeffizienten von $\Pi_k^{\nabla,T}\varphi_j$ bezüglich der Monombasis.

Wie für Virtuelle-Elemente-Methoden üblich wird ebenfalls die (Matrix-)Darstellung des Operators $\Pi_k^{\nabla,T}$ als Abbildung auf $\boldsymbol{V}_k^T$ selbst benötigt [dBC+13, DV19]. Das ist möglich, da $[\Pi_k(T)]^2$ selbst eine Teilmenge von $\boldsymbol{V}_k^T$ ist. Dafür wird $\Pi_k^{\nabla,T}\varphi_j$ in die Basiselemente von $\boldsymbol{V}_k^T$ zerlegt. Es ergibt sich damit

$$\Pi_k^{\nabla,T}\varphi_j = \sum_{\ell=1}^{\texttt{nDof}^T} \pi_\ell^{(j)}\varphi_\ell,$$

wobei $\pi_\ell^{(j)} = D_{V,\ell}(\Pi_k^{\nabla,T}\varphi_j)$ der ℓ-te lokale Freiheitsgrad von $\Pi_k^{\nabla,T}\varphi_j$ ist. Es gilt weiterhin

$$\Pi_k^{\nabla,T}\varphi_j = \sum_{n=1}^{2\pi_k} s_n^{(j)}\boldsymbol{m}_n = \sum_{n=1}^{2\pi_k} s_n^{(j)} \sum_{\ell=1}^{\texttt{nDof}^T} D_{V,\ell}(\boldsymbol{m}_n)\varphi_\ell,$$

woraus folgt, dass

$$\pi_\ell^{(j)} = \sum_{n=1}^{2\pi_j} s_n^{(j)} D_{V,\ell}(\boldsymbol{m}_n). \tag{4.8}$$

Um diese Gleichung in Matrixform auszudrücken, wird jetzt die $\texttt{nDof}^T \times 2\pi_k$-dimensionale Matrix D als $D_{\ell,n} := D_{V,\ell}(\boldsymbol{m}_n)$, $\ell = 1, 2, \ldots \texttt{nDof}^T$, $n = 1, 2, \ldots, 2\pi_k$ definiert. Somit gilt

$$D := \begin{pmatrix} D_{V,1}(\boldsymbol{m}_1) & D_{V,1}(\boldsymbol{m}_2) & \ldots & D_{V,1}(\boldsymbol{m}_{2\pi_k}) \\ D_{V,2}(\boldsymbol{m}_1) & D_{V,2}(\boldsymbol{m}_2) & \ldots & D_{V,2}(\boldsymbol{m}_{2\pi_k}) \\ \vdots & \vdots & \ddots & \vdots \\ D_{V,\texttt{nDof}^T}(\boldsymbol{m}_1) & D_{V,\texttt{nDof}^T} & \ldots & D_{V,\texttt{nDof}^T}(\boldsymbol{m}_{2\pi_k}) \end{pmatrix}.$$

Damit lässt sich Gleichung (4.8) schreiben als

$$\pi_\ell^{(j)} = \sum_{n=1}^{2\pi_k} \left(G^{-1}C\right)_{n,j} D_{\ell,n} = \left(DG^{-1}C\right)_{\ell,j}.$$

Sei jetzt S die Matrixdarstellung des Operators $\Pi_k^{\nabla,T}$ als Abbildung auf $\boldsymbol{V}_k^T$. Dann ergibt sich also

$$S = DG^{-1}C = DS^*. \tag{4.9}$$

Nun sind alle Voraussetzungen vorhanden, um Gleichung (4.1) zu lösen. Für den ersten Teil ergibt sich wegen der Gleichung (4.7)

$$a^T\left(\Pi_k^{\nabla,T}\varphi_j, \Pi_k^{\nabla,T}\varphi_\ell\right) = \sum_{n,n_*=1}^{2\pi_k} s_n^{(j)} s_{n_*}^{(\ell)} a^T\left(\boldsymbol{m}_n, \boldsymbol{m}_{n_*}\right) = \left[S^{*t}\widetilde{G}S^*\right]_{j,\ell},$$

wobei S^{*t} die zu S^* transponierte Matrix bezeichnet und $\widetilde{G}$ identisch zu G ist mit Ausnahme der ersten beiden Zeilen, die überall zu 0 gesetzt werden. Für den zweiten Teil folgt mittels der Gleichung (4.9) für alle $j, \ell = 1, 2, \ldots, \texttt{nDof}^T$, dass

$$\begin{aligned}\sum_{n=1}^{\texttt{nDof}^T} D_{V,n}((\boldsymbol{I} - \Pi_k^{\nabla,T})\varphi_j) D_{V,n}((\boldsymbol{I} - \Pi_k^{\nabla,T})\varphi_\ell) &= \sum_{n=1}^{\texttt{nDof}^T} (\boldsymbol{I} - S)^t_{j,n} (\boldsymbol{I} - S)^t_{\ell,n} \\ &= \left((\boldsymbol{I} - S)^t (\boldsymbol{I} - S)\right)_{j,\ell}. \end{aligned} \tag{4.10}$$

Sei jetzt K_h^T die lokale Steifigkeitsmatrix des Laplaceoperators, das heißt

$$K_h^T = \left(a_h^T(\varphi_i, \varphi_j)\right)_{i,j=1}^{\texttt{nDof}^T}.$$

Dann ergibt sich zusammenfassend für die Steifigkeitsmatrix, dass

$$K_h^T = S^{*t} \widetilde{G} S^* + \nu (\boldsymbol{I} - S)^t (\boldsymbol{I} - S). \tag{4.11}$$

Ebenfalls muss noch Gleichung (3.19b) für alle $\varphi_j \in \boldsymbol{V}_k^T$, $j = 1, 2, \ldots, \texttt{nDof}^T$, und für alle m_ℓ, $\ell = 1, 2, \ldots, \pi_{k-1}$, berechnet werden. Es ergibt sich

$$\begin{aligned} b_h^T(\varphi_j, m_\ell) = b^T(\varphi_j, m_\ell) &= \int_T \mathrm{div}\varphi_j \, m_\ell \, \mathrm{d}T \\ &= \begin{cases} \int_{\partial T} \varphi_j \boldsymbol{n} \, \mathrm{d}s, & \text{für } \ell = 1 \\ D_{\boldsymbol{V}}^{\mathcal{M}_2}(\varphi_j), & \text{für } \ell = 2, 3, \ldots \pi_{k-1} \end{cases}, \end{aligned} \tag{4.12}$$

wobei für $\ell = 1$ die partielle Integration angewendet wird. Nun kann für $\ell = 1$ das letzte Integral mit Hilfe der Randfreiheitsgrade ausgewertet werden, womit also beide Fälle nur mit Hilfe der Freiheitsgrade berechenbar sind.

Für die Bestimmung der rechten Seite wird $P_{k-2}^T \varphi_j$ für alle $j = 1, 2, \ldots, \texttt{nDof}^T$ benötigt. Dafür wird $P_{k-2}^T \varphi_j$ in die skalierte Monombasis zerlegt und die Koeffizienten mittels Gleichung (3.23) berechnet, wobei für $\boldsymbol{q}_{k-2}$ die skalierten Monome $\boldsymbol{m}_\ell$, $\ell = 1, 2, \ldots, 2\pi_{k-2}$, durchlaufen werden.

4.2.3. Aufstellen der globalen Matrizen

Jetzt sind alle Berechnungen vorhanden, um das diskrete Problem (3.24a)-(3.24b) zu lösen. Dafür werden die Geschwindigkeitslösung $\boldsymbol{u}_h$ als Linearkombination der globalen Basiselemente und die Drucklösung p_h auf jedem Element als Linearkombination der skalierten Monome

$$\boldsymbol{u}_h = \sum_{j=1}^{\texttt{nDof}} x_j \, \Phi_j \qquad \text{und} \qquad p_{h|T} = \sum_{l=1}^{\pi_{k-1}} y_\ell^{(T)} \, m_\ell$$

dargestellt. Die Koeffizientenvektoren

$$\boldsymbol{x} := \begin{pmatrix} x_1 & x_2 & \cdots & x_{\texttt{nDof}} \end{pmatrix}^t \qquad \text{und} \qquad \boldsymbol{y} := \begin{pmatrix} \boldsymbol{y}^{(1)} & \boldsymbol{y}^{(2)} & \cdots & \boldsymbol{y}^{(|\mathcal{T}_h|)} \end{pmatrix}^t$$

mit $\boldsymbol{y}^{(j)} := \left(y_1^{(T_j)}, y_2^{(T_j)}, \ldots, y_{\pi_{k-1}}^{(T_j)}\right)^t$ für $j = 1, 2, \ldots, |\mathcal{T}_h|$ sind dabei die zu bestimmenden Unbekannten des zum diskreten Problem gehörenden linearen Gleichungssystems. Mit den Definitionen

$$K_h := (a_h(\Phi_i, \Phi_j))_{i,j=1}^{\texttt{nDof}}, \qquad B := (b_h(\Phi_j, m_\ell))_{j=1,\ldots,\texttt{nDof}}^{\ell=1,\ldots,|\mathcal{T}_h|\,\pi_k-1} \qquad \text{und}$$
$$\boldsymbol{F_h} := \left((\boldsymbol{f}_h, \Phi_j)_{L^2(\Omega)}\right)_{j=1}^{\texttt{nDof}}$$

ergibt sich das globale Gleichungssystem

$$\left(\begin{array}{c|c|c} K_h & B & \mathbf{0} \\ \hline B^t & \mathbf{0} & E \\ \hline 0 & E^t & 0 \end{array}\right) \left(\begin{array}{c} \boldsymbol{x} \\ \hline \boldsymbol{y} \\ \hline \lambda \end{array}\right) = \left(\begin{array}{c} \boldsymbol{F_h} \\ \hline \mathbf{0} \\ \hline 0 \end{array}\right), \tag{4.13}$$

wobei

$$E := \left(e^{(1)} \quad e^{(2)} \quad \ldots \quad e^{(|\mathcal{T}_h|)}\right)^t$$

mit $e^{(j)} := \left(\int_{T_j} m_1 \,\mathrm{d}T, \int_{T_j} m_2 \,\mathrm{d}T, \ldots, \int_{T_j} m_{\pi_{k-1}} \,\mathrm{d}T\right)^t$ für $j = 1, 2, \ldots, |\mathcal{T}_h|$. Die letzte Block-Zeile von Gleichung (4.13) ist zum Sicherstellen des globalen Integralmittels des Drucks $\int_\Omega p \,\mathrm{d}\Omega = 0$ mit dem zugehörigen Lagrange Multiplikator $\lambda \in \mathbb{R}$.

Um auch inhomogene Dirichletrandbedingungen zuzulassen, werden die Knotenwerte an den Endpunkten der Dirichletrandkanten durch Auswerten der Dirichletrandbedingung in diesen Punkten gesetzt. Analog geschieht dies auch für $k-2$ innere Punkte jeder Kante. Der letzte Punkt muss so gewählt werden, dass das Integralmittel entlang der Kante erhalten bleibt.

Der Vektor mit den inhomogenen Randdaten multipliziert an die Steifigkeitsmatrix K_h wird wie üblich (z. B. [CGK+10]) von der rechten Seite subtrahiert werden. Das Gleichungssystem kann anschließend an den inneren Freiheitsgraden gelöst werden, was den inneren Knoten, Kanten sowie allen Momenten entspricht.

4.3. Numerische Experimente mit der Virtuellen-Elemente-Methode

In diesem Abschnitt soll die Implementierung der in diesem Kapitel vorgestellten Virtuellen-Elemente-Methode für das Stokes-Problem demonstriert werden. Die Methode ist für die niedrigste Ordnung implementiert, also $k = 2$. Sei daher im Folgenden $\boldsymbol{V}_h := \boldsymbol{V}_2$ und $Q_h := Q_1$. Die Geschwindigkeit wird daher mit quadratischen Polynomen und der Druck mit stückweise linearen Polynomen approximiert, was unter genügend hoher Regularität der Lösung nach Korollar 3.33 eine Konvergenzrate von 2 für den Energiefehler der Geschwindigkeit und den L^2-Fehler des Drucks bewirken sollte.

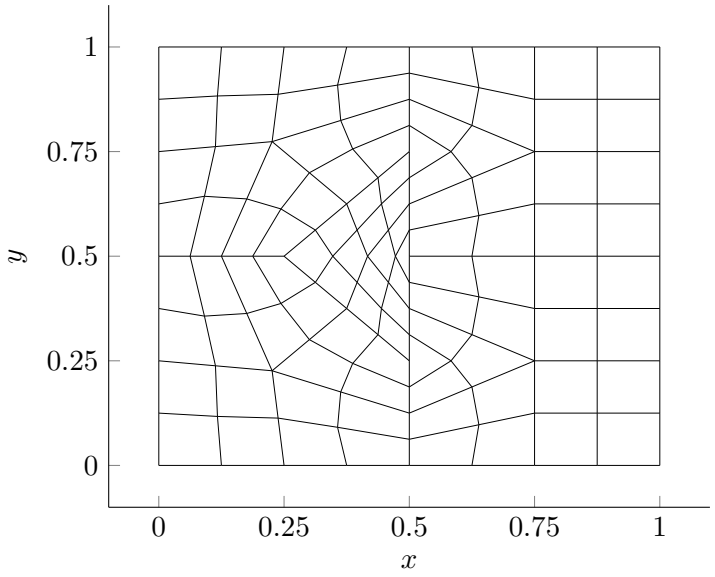


Abbildung 4.2.: Dargestellt ist die Zerlegung des Einheitsquadrats in Vierecke mit 995 Knoten für das Problem 1.

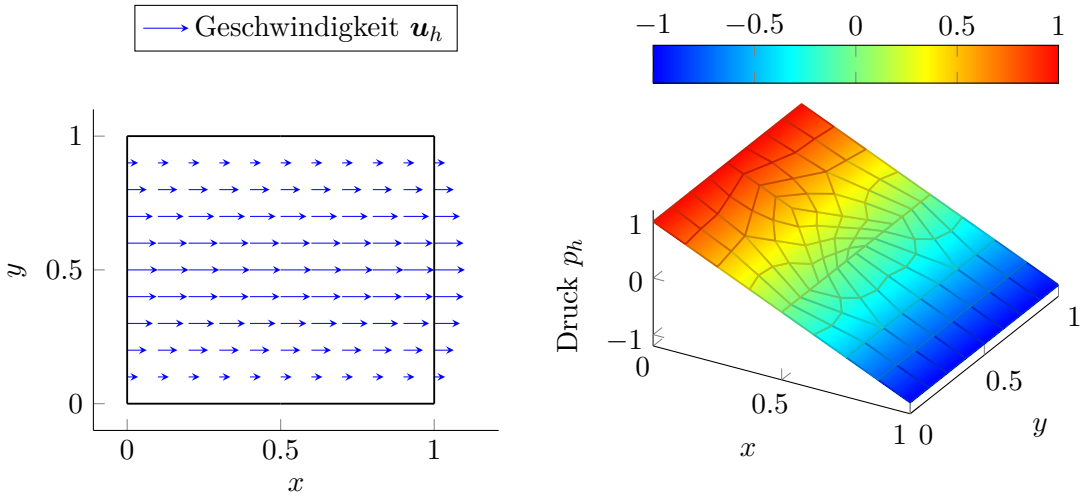
4.3.1. Problem 1: Hagen-Poiseuille

Das erste Problem beruht auf dem Gesetz von Hagen-Poiseuille, was zum Beispiel den laminaren stationären Strom eines homogenen Newtonschen Fluids durch ein Rohr beschreibt [GF05]. Das Strömungsprofil ist dabei eine quadratische Funktion, die ihr Maximum in der Mitte des Rohrs hat und entlang des Gradienten des linearen Drucks verläuft. Die Geschwindigkeit und der Druck sind dabei gerade so, dass die rechte Seite im Stokes-Problem genau Null ist. Hier wird der Einfachheit halber ein quadratischer Querschnitt eines Rohrs genommen, das heißt $\Omega = (0,1)^2$, und die Viskosität $\nu = 1$ gesetzt. Die exakte Lösung und die zugehörige rechte Seite des Stokes-Problems sind durch

$$\boldsymbol{u}(x,y) = \begin{pmatrix} y(1-y) \\ 0 \end{pmatrix} \in \boldsymbol{V}_h, \quad p(x,y) = -2x + 1 \in Q_h \quad \text{und} \quad \boldsymbol{f}(x,y) = 0 \in L^2(\Omega)$$

gegeben. Hier wird im Gegensatz zum homogenen Stokes-Problem die Dirichletrandbedingung entlang $\partial\Omega$ als exakte Lösung $\boldsymbol{u}$ vorgegeben. Einfaches Nachrechnen zeigt, dass die angegebene Geschwindigkeit und der Druck genau eine Lösung des Stokes-Problems (2.1a)-(2.1c) mit inhomogenen Dirichletrandbedingungen ist (s. Bemerkung 2.17). Als Zerlegung von Ω wird das Netz aus Abbildung 4.2 bestehend aus 96 Vierecken mit 995 Knoten zu Grunde gelegt.

Die approximierte Geschwindigkeit $\boldsymbol{u}_h$ und der approximierte Druck p_h sind in Abbildung 4.3 dargestellt. Zur besseren Darstellung der Geschwindigkeitslösung sind die Werte der Lösung auf einem äquidistanten Gitter aus den Knotenwerten der approximierten



(a) Diskretes Geschwindigkeitsfeld

(b) Diskreter Druck

Abbildung 4.3.: Vorgestellt wird die diskrete Lösung für Problem 1. In (a) ist die Geschwindigkeit aus den Knotenwerten der diskreten Lösung auf einem äquidistanten Gitter interpoliert und skaliert um den Faktor 0,4 dargestellt. In (b) ist der diskrete Druck gezeigt. Gut sichtbar ist das parabolische Strömungsprofil und der Fluß entlang des Gradienten des Drucks.

Lösung interpoliert. Es ist deutlich das parabolische Strömungsprofil zu erkennen, das entlang des Gradienten des Drucks fließt.

Da die exakte Lösung in den Ansatzräumen ist, sollte die Methode auch die exakte Lösung auf jedem beliebigen Gitter approximieren. Somit müssen der Geschwindigkeitsfehler und der Druckfehler Null sein. Dies ist auch bis auf Rundungsfehler der Fall. Die Fehler berechnen sich auf dem Gitter aus Abbildung 4.2 nämlich zu

$$\|\mathrm{D}(\boldsymbol{u} - \Pi_k^\nabla \boldsymbol{u}_h)\|_{L^2(\Omega)} \approx 4,4 \cdot 10^{-15} \text{ und}$$
$$\|p - p_h\|_{L^2(\Omega)} \approx 5,4 \cdot 10^{-16}.$$

Bemerkung 4.4. Achtung: Der vollständige Geschwindigkeitsfehler $\|\mathrm{D}(\boldsymbol{u} - \boldsymbol{u}_h)\|_{L^2(\Omega)}$ kann nicht berechnet werden, da $\boldsymbol{u}_h$ immer noch eine virtuelle Funktion ist, die nicht explizit zur Verfügung steht. Abhilfe schafft wieder die stückweise polynomielle Bestapproximation in der Energienorm Π_k^∇, die für $\boldsymbol{v}_h \in \boldsymbol{V}_k$ durch $\Pi_k^\nabla \boldsymbol{v}_{h|T} := \Pi_k^{\nabla,T} \boldsymbol{v}_h$ für alle $T \in \mathcal{T}_h$ definiert ist.

4.3.2. Problem 2: Hydrostatisches Problem für verschiedene Viskositäten

Für das Problem 2 wird das Gebiet $\Omega = (0,1)^2$ gewählt. Die rechte Seite und die Randbedingungen werden so gewählt, dass die exakte Lösung durch

$$\boldsymbol{u}(x,y) := \begin{pmatrix} 0 \\ 0 \end{pmatrix} \in \boldsymbol{V}_h \qquad \text{und} \qquad p(x,y) = \sin(2\pi x)\cos(2\pi y) \notin Q_h$$

Tabelle 4.1.: Aufgelistet sind die Fehler der Geschwindigkeit und des Drucks sowie zugehörige Konvergenzraten R_V und R_P in Bezug auf den mittleren Durchmesser h_m für das Problem 2 mit Viskosität $\nu = 1$. Beide Fehler konvergieren mit optimaler Ordnung 2 wie in Korollar 3.33 vorhergesagt.

`nDof`	h_m	$\|\mathrm{D}(\boldsymbol{u} - \Pi_k^\nabla \boldsymbol{u}_h)\|_{L^2(\Omega)}$	Rate R_V	$\|p - p_h\|_{L^2(\Omega)}$	Rate R_P
50	0,666	$5,1358 \cdot 10^{-2}$	-	$2,7239 \cdot 10^{-1}$	-
235	0,321	$4,0608 \cdot 10^{-2}$	0,32	$1,7611 \cdot 10^{-1}$	0,60
995	0,163	$8,2634 \cdot 10^{-3}$	2,34	$5,9090 \cdot 10^{-2}$	1,88
4099	0,081	$1,8088 \cdot 10^{-3}$	2,20	$1,2647 \cdot 10^{-2}$	1,96
16643	0,041	$4,2634 \cdot 10^{-4}$	2,09	$3,1857 \cdot 10^{-3}$	1,99
67075	0,020	$1,0445 \cdot 10^{-4}$	2,03	$8,9793 \cdot 10^{-4}$	2,00
269315	0,010	$2,5958 \cdot 10^{-5}$	2,01	$2,9958 \cdot 10^{-4}$	2,00

gegeben ist. Die rechte Seite ist somit nur der Gradient von p. Die Lösung wird auf einer Serie von Gittern ausgehend von der Zerlegung aus Abbildung 4.1 berechnet. Der zugrunde liegende Verfeinerungsalgorithmus wird in Kapitel 5 noch genauer erläutert.

Für den ersten Teil des Experiments wird die Viskosität $\nu = 1$ vorgegeben. In Tabelle 4.1 sind die Fehler der Geschwindigkeit, des Drucks und die Konvergenzraten der Geschwindigkeit R_V und des Drucks R_P in Bezug auf den mittleren Druchmesser

$$h_m := \frac{1}{|\mathcal{T}_h|} \sum_{T \in \mathcal{T}_h} h_T$$

aufgelistet. Sowohl der Geschwindigkeits- als auch der Druckfehler konvergieren mit der zu erwartenden Rate und bestätigen damit die Aussagen aus dem Korollar 3.33.

Im zweiten Teil des Experiments wird die gleiche exakte Lösung und damit die gleiche rechte Seite vorgegeben. Allerdings wird jetzt zwischen verschiedenen Viskositäten ν unterschieden, um die Abhängigkeit des Geschwindigkeitsfehler von der Viskosität zu untersuchen. Dafür werden $\nu = 1$, $\nu = 0,01$ und $\nu = 0,0001$ gewählt. Der Fehler der Geschwindigkeit ist gegen die Anzahl der Freiheitsgrade `nDof` in Abbildung 4.4 dargestellt. Für die jeweilige Viskosität konvergiert der Geschwindigkeitsfehler mit der optimalen Rate. Allerdings skaliert er mit dem Inversen von ν und bestätigt damit die Aussagen aus Korollar 3.33. Bei der herkömmlichen Virtuellen-Elemente-Methode tritt daher ein sogenannter “locking“-Effekt auf, der bei einer druckrobusten Methode nicht möglich wäre. In Kapitel 6 ist gezeigt, dass mit einer druckrobusten Diskretisierung der rechten Seite der Geschwindigskeitsfehler unabhängig von ν ist. Dadurch wird ein “locking“ in jedem Fall verhindert. Druckrobust meint hier, dass Änderungen in der rechten Seite, die nur den kontinuierlichen Druck ändern, auch im Diskreten die Geschwindigkeit nicht beeinflussen würden. Auf dieses Beispiel angewandt, müsste die Methode die Geschwindigkeit exakt approximieren, da sie im Ansatzraum enthalten ist. Da dies nicht der Fall ist, ist die Methode also nicht druckrobust. Dies wird in 6 detailierter diskutiert.

Der Druckfehler hingegen skaliert nicht mit ν und konvergiert in allen drei Fällen mit der optimaten Rate, wie es die zweite Aussage aus Korollar 3.33 mit $\boldsymbol{u} = 0$ auch

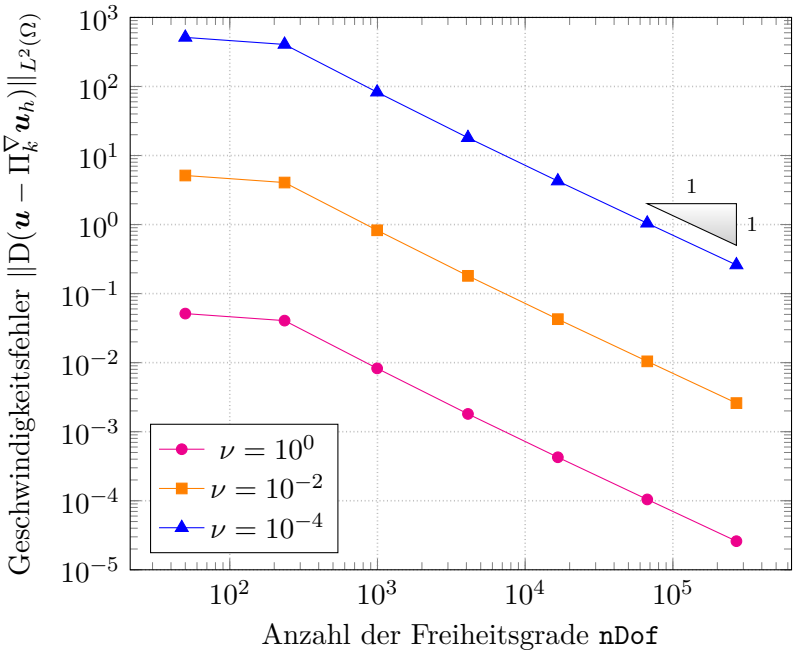


Abbildung 4.4.: Gezeigt ist die Konvergenzhistorie der Geschwindigkeit gegenüber der Anzahl der Freiheitsgrade für Problem 2 mit verschiedenen Viskositäten ν. Der Geschwindigkeitsfehler für die verschiedenen Viskositäten konvergiert jeweils mit der optimalen Rate, skaliert aber mit dem Inversen von ν. Die normale Virtuelle-Elemente-Methode ist daher nicht druckrobust.

vorhersagt.

4.3.3. Problem L: Das L-Gebiet zerlegt in Quadrate

In den vorherigen Beispielen waren die Lösungen $(\boldsymbol{u}, p)$ immer glatt genug, um die Voraussetzungen aus Korollar 3.33 zu erfüllen. Hier wird nun untersucht, was geschieht, wenn dies nicht der Fall ist.

Dafür wird das bekannte L-Gebiet $\Omega = (-1,1)^2 \setminus (0,1) \times (0,-1)$ (engl. "l-shape") zur Hilfe genommen, das inklusiver einer möglichen Zerlegung in Abbildung 4.5 dargestellt ist. Die Viskosität wird auf $\nu = 1$ gesetzt.

Als exakte Lösung wird in Polarkoordinaten (r, φ)

$$\boldsymbol{u}(r,\varphi) = r^{\alpha} \begin{pmatrix} (\alpha+1)\sin(\varphi)\psi(\varphi) + \cos(\varphi)\psi'(\varphi) \\ -(\alpha-1)\cos(\varphi)\psi(\varphi) + \sin(\varphi)\psi'(\varphi) \end{pmatrix},$$
$$p(r,\varphi) = \frac{r^{\alpha-1}}{\nu(1-\alpha)}((1+\alpha)^2\psi'(\varphi) + \psi'''(\varphi))$$

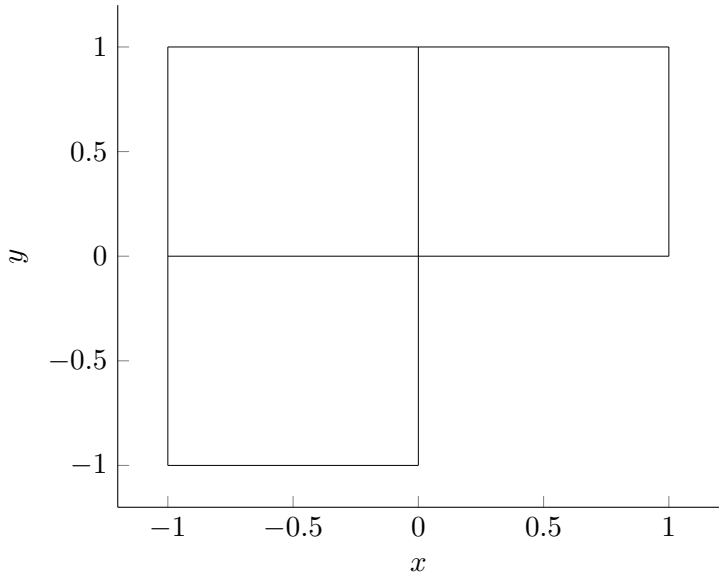


Abbildung 4.5.: Dargestellt ist eine Zerlegung des L-Gebiets in drei Quadrate mit acht Knoten.

mit

$$\begin{aligned}\psi(\varphi) =&(\alpha+1)^{-1}\sin((\alpha+1)\varphi)\cos(\alpha\beta)\\ &-\cos((\alpha+1)\varphi)-(\alpha-1)^{-1}\sin((\alpha-1)\varphi)\cos(\alpha\beta)-\cos((\alpha-1)\varphi)\end{aligned}$$

vorgegeben, wobei $\alpha = 856399/1572864 \approx 0,54$ und $\beta = 3\pi/2$ ist [Ver89]. Diese Wahl führt zu der rechten Seite $\boldsymbol{f} = 0$, wie einfach nachgerechnet werden kann. Die Fehler für die Geschwindigkeit und den Druck sind in Tabelle 4.2 zusammen mit den jeweiligen Raten R_V und R_P in Bezug auf den mittleren Durchmesser h_m aufgelistet. Sowohl für den Druck als auch für die Geschwindigkeit wird eine Konvergenzrate von etwa 0,54 erreicht, was deutlich unter der Erwartung von 2 liegt. Die suboptimalen Konvergenzraten hängen mit der Nicht-Konvexität des Gebiets zusammen. Die einspringende Ecke führt

Tabelle 4.2.: Aufgelistete sind die Fehler der Geschwindigkeit und des Drucks sowie zugehörige Konvergenzraten R_V und R_P in Bezug auf den mittleren Durchmesser h_m für Problem L.

`nDof`	h_m	$\|\mathrm{D}(\boldsymbol{u}-\Pi_k^\nabla \boldsymbol{u}_h)\|_{L^2(\Omega)}$	Rate R_V	$\|p-p_h\|_{L^2(\Omega)}$	Rate R_P
8195	0,088	$4,0856\cdot10^{-1}$	-	$3,0260\cdot10^{-1}$	-
33283	0,044	$2,8010\cdot10^{-1}$	0,54	$2,0705\cdot10^{-1}$	0,55
134147	0,022	$1,9204\cdot10^{-1}$	0,54	$1,4183\cdot10^{-1}$	0,55
538627	0,011	$1,3166\cdot10^{-1}$	0,54	$9,7198\cdot10^{-2}$	0,55
2158595	0,006	$9,0273\cdot10^{-2}$	0,54	$6,6628\cdot10^{-2}$	0,54

zu einer Singularität, die das Konvergenzverhalten der Methode stört. Dies kann zum Beispiel mit Hilfe von lokaler Verfeinerung um die Singularität wieder repariert werden, womit sich das Kapitel 5 ausführlicher beschäftigt.

5. A-Posteriori Fehleranalyse

Problem L aus Kapitel 4 zeigt, dass die Virtuelle-Elemente-Methode mit suboptimaler Rate konvergieren kann, wenn das Gebiet nicht konvex ist. Die Nicht-Konvexität führt für elliptische Probleme auf eine Singularität in der Lösung an den Eckpunkten mit einem größeren Innenwinkel als π [Bar16]. Uniform verfeinerte Gitter können solch eine Singularität nicht ausreichend auflösen, weshalb es sinnvoll ist, das Gitter lokal um die Singularitäten besser aufzulösen. Eine Möglichkeit dafür sind a-priori gebaute gestaffelte Gitter (engl. "graded meshes"), die lokal fein um die Singularität sind. Alternativ kann auch a-posteriori versucht werden, mit Hilfe der diskreten Lösung abzuschätzen, an welchen Orten der Fehler groß ist und dementsprechend adaptiv an diesen Stellen lokal zu verfeinern [BS02, Bar16].

Die Idee der adaptiver Verfeinerungen soll in diesem Kapitel für die Virtuelle-Elemente-Methode vorgestellt werden. Dafür wird zunächst ein möglicher Verfeinerungsalgorithmus vorgestellt, mit dem automatisch Gitter verfeinert werden können. Daraufhin wird ein residuumbasierter Fehlerschätzer eingeführt und auf dessen Verlässlichkeit untersucht. Numerische Experimente zur Demonstation des Fehlerschätzers schließen das Kapitel ab.

5.1. A-Posteriori Fehlerschätzer und adaptive Verfeinerungen

Ein adaptiver Verfeinerungsalgorithmus berechnet sukzessive eine Reihe formregulärer Gitter $\mathcal{T}_1$, $\mathcal{T}_2, \ldots$ und endliche Unterräume

$$(\boldsymbol{V}_{k,1} \times Q_{k,1}) \subset (\boldsymbol{V}_{k,2} \times Q_{k,2}) \subset \ldots \subset (\boldsymbol{V} \times Q)$$

und läuft dabei standardmäßig in den Schritten

Löse $\rightarrow$ **Schätze** $\rightarrow$ **Markiere** $\rightarrow$ **Verfeinere** $\rightarrow$ **Löse** $\ldots$

ab. Die vier verschiedenen Schritte haben dabei die folgenden Aufgaben:

Löse das diskrete Problem mit der Virtuellen-Elemente-Methoden,

schätze den Fehler lokal auf jedem Polygon,

markiere die Polygone mit den größten Beiträgen zum Fehler und

verfeinere die markierten Polygone.

Der Lösungsalgorithmus ist bereits in Kapitel 3 vorgestellt.

Ein schwieriger Schritt des adaptiven Algorithmus, das Schätzen, beruht auf einem lokalen Indikator η_T für $T \in \mathcal{T}_h$, der den lokalen Beitrag zum Fehler quantifiziert. Dieser Indikator wird in Abschnitt 5.1.2 vorgestellt.

Das Markieren erfolgt mit Hilfe des sogenannten Dörfler-Markierens [Dö96]. Mit vorgegebenem Volumenparameter $\theta \in (0, 1]$, wird die kleinste Teilmenge $\mathcal{M} \subset \mathcal{T}_h$ gewählt, so dass

$$\theta \sum_{T \in \mathcal{T}_h} \eta_T^2 \leq \sum_{T \in \mathcal{M}} \eta_T^2 \tag{5.1}$$

gilt. Je kleiner θ gewählt wird, desto weniger Polygone werden pro Verfeinerungsschritt markiert beziehungsweise desto lokaler wird verfeinert. Für gewöhnlich wird dabei ein Wert zwischen 0 und 0,5 gewählt [CFPP14]. Die Wahl von $\theta = 1$ führt zu uniformer Verfeinerung aller Elemente.

Der letzte Schritt des adaptiven Algorithmus ist das Verfeinern. Dafür notwendig ist ein Verfeinerungsalgorithmus, der automatisch aus einem gegebenen Polygon kleinere Polygone erzeugt. Im Folgenden wird zuerst der Verfeinerungsalgorithmus vorgestellt, woraufhin der Fehlerschätzer η_T beschrieben wird.

5.1.1. Verfeinerungsalgorithmus für Polygone

Beim Verfeinern ist es wichtig, dass die erzeugten Polygone nicht degenerieren, das heißt die Voraussetzungen der Formregularität verletzen. Sonst können keine optimalen Konvergenzraten erwartet werden.

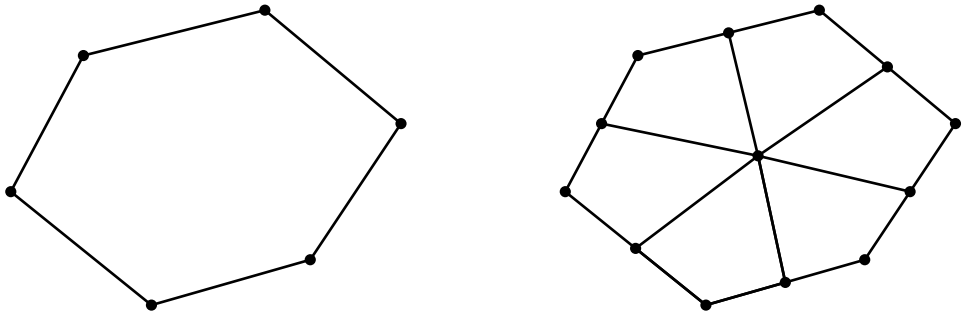
Der hier vorgestellte Verfeinerungsalgorithmus ist eine Vereinfachung der von Dr. O. Sutton in [Sut17] vorgestellten Idee. Ein Polygon $T \in \mathcal{T}_h$ wird dabei wie folgt verfeinert: Jede Kante $E \in \mathcal{E}(T)$ wird im Mittelpunkt geteilt und mit dem Baryzentrum $\boldsymbol{x}_T$ des Polygons verbunden. Dies stellt eine Vereinfachung des von Dr. O. Sutton vorgestellten Vorgehens dar, da dort nur planare Seiten geteilt werden. Anders ausgedrückt werden hägende Knoten bei der Verfeinerung ignoriert, was bei dem in dieser Arbeit benutzten Algorithmus nicht der Fall ist. Eine Illustration dieses Verfeinerungsalgorithmus ist in Abbildung 5.1 dargestellt.

Durch diese Verfeinerungsstrategie wird ein Polygon mit $|\mathcal{E}(T)|$ Kanten in $|\mathcal{E}(T)|$ Polygone mit jeweils vier Ecken geteilt. Es werden die Kantenmittelpunkte verwendet und nicht die Eckpunkte selbst, um ein Degenerieren der Winkel zu verhindern. Die Methode ist eine Verallgemeinerung der Zerteilung von Rechtecken in vier gleich große Rechtecke.

Hierbei können durchaus hängende Knoten entsehen. Dies stellt im Gegensatz zur Finiten-Elemente-Methode allerdings kein Problem dar, da in der Virtuellen-Elemente-Methode Polygone mit Innenwinkeln von 180° zugelassen werden können (s. Bemerkung 3.2).

5.1.2. Der Fehlerschätzer und seine Analyse

Nun kann mit dem residuumbasierten Fehlerschätzer des adaptiven Algorithmus begonnen werden. Gesucht ist eine lokale Größe η_T, die nur mit Hilfe der Eingangsdaten und



(a) Ursprüngliches Polygon (b) Verfeinerung

Abbildung 5.1.: Illustriert ist die Verfeinerungsstrategie für ein Polygon mit sechs Ecken und ohne hängende Knoten. Das ursprüngliche Element ist in (a) dargestellt, während die neuen Elemente in (b) gezeigt sind. Der Mittelpunkt jeder Kante wird für die Verfeinerung mit dem Baryzentrum verbunden.

der berechneten diskreten Lösung berechnet werden kann und für die

$$\eta_T \approx \|\mathrm{D}(\boldsymbol{u} - \boldsymbol{u}_h)\|_{L^2(\omega_T)}$$

gilt.

Die folgenden Aussagen gelten für homogene Randdaten beziehungsweise genau für den Fall, dass die inhomogenen Randdaten exakt aufgelöst werden. Abschließend folgt eine Bemerkung für den inhomogenen Fall.

Lemma 5.1 (Residuumgleichung)**.** *Für den Fehler zwischen der exakten Lösung $\boldsymbol{u} \in \boldsymbol{Z}$ und der diskreten Lösung $\boldsymbol{u}_h \in \boldsymbol{Z}_k$ des Stokes-Problems gilt*

$$\nu\|\mathrm{D}(\boldsymbol{u} - \boldsymbol{u}_h)\|_{L^2(\Omega)} = \|\mathrm{Res}(\boldsymbol{u}_h)\|_{\boldsymbol{Z}^*} := \sup_{\boldsymbol{v}\in\boldsymbol{Z}} \frac{|\mathrm{Res}(\boldsymbol{u}_\mathrm{h})(\boldsymbol{v})|}{\|\mathrm{D}\boldsymbol{v}\|_{L^2(\Omega)}},$$

wobei $\mathrm{Res}(\boldsymbol{v}_h) \in \boldsymbol{Z}^*$ *für* $\boldsymbol{v}_h \in \boldsymbol{Z}_k$ *mittels*

$$\mathrm{Res}(\boldsymbol{v}_h)(\boldsymbol{v}) = a(\boldsymbol{u} - \boldsymbol{v}_h, \boldsymbol{v}) \qquad \text{für alle } \boldsymbol{v} \in \boldsymbol{Z}$$

definiert ist.

Beweis. Da für exakt aufgelöste Randdaten $\boldsymbol{u} - \boldsymbol{u}_h \in \boldsymbol{Z}$ gilt, folgt

$$\|\mathrm{Res}(\boldsymbol{u}_h)\|_{\boldsymbol{Z}^*} \geq \nu\|\mathrm{D}(\boldsymbol{u} - \boldsymbol{u}_h)\|_{L^2(\Omega)}$$

aus der Definition des Supremums. Die andere Richtung folgt aus der Cauchy-Schwarz-Ungleichung. □

Um den Fehlerschätzer für die Geschwindigkeit zu definieren, wird der Sprung entlang einer Kante $E = \partial T_+ \cap \partial T_-$ einer elementweise stetigen, matrixwertigen Funktion A benötigt. Dieser ist durch

$$[A\boldsymbol{n}_E] := \left(A_{|T_+} - A_{|T_-}\right)\boldsymbol{n}_E$$

definiert.

Definition 5.2 (Residuumbasierter Fehlerschätzer für die Geschwindigkeit)**.** Sei für $(\boldsymbol{v}_h, q_h) \in \boldsymbol{V}_k \times Q_k$ und für alle $T \in \mathcal{T}_h$

$$\begin{aligned}\eta_T(\boldsymbol{v}_h, q_h)^2 :=& \frac{h_T^2}{\nu^2} \|\boldsymbol{f} - \boldsymbol{f}_h\|_{L^2(T)}^2 + \frac{h_T^2}{\nu^2} \left\|\boldsymbol{f}_h - \nabla q_h + \nu\Delta(\Pi_k^{\nabla,T}\boldsymbol{v}_h)\right\|_{L^2(T)}^2 \\ &+ \frac{1}{\nu^2} \sum_{E\in\mathcal{E}(T)} h_E \left\|\left[(\nu \mathrm{D}(\Pi_k^{\nabla,T}\boldsymbol{v}_h) + q_h)\boldsymbol{n}_E\right]\right\|_{L^2(E)}^2 \\ &+ S^T\left((\boldsymbol{I} - \Pi_k^{\nabla,T})\boldsymbol{v}_h, (\boldsymbol{I} - \Pi_k^{\nabla,T})\boldsymbol{v}_h\right)\end{aligned}$$

definiert, wobei $h_E := |E|$ die Länge von E ist. Dann ist

$$\eta(\boldsymbol{v}_h, q_h) := \left(\sum_{T\in\mathcal{T}_h} \eta_T(\boldsymbol{v}_h, q_h)^2\right)^{1/2}$$

der residuumbasierte Fehlerschätzer für die Geschwindigkeit.

Bemerkung 5.3. Der Stabilisierungsterm in diesem Fehlerschätzer kommt daher, dass die diskrete Bilinearform im Gegensatz zu den Finite-Elemente-Methoden aus der kontinuierlichen Bilinearform plus der Stabilisierungsbilinearform besteht. Die Projektion hat ihren Ursprung darin, dass die diskrete Lösung nur implizit zur Verfügung steht. Um eine berechenbare Größe zur Hand zu haben, muss diese Projektion gewählt werden. Der Fehlerschätzer hat ansonsten die üblichen Komponenten, das heißt Volumenterm, Sprungterm der Normalenableitung der Geschwindigkeit und Oszillationsterme, eines Fehlerschätzers aus dem Kontext der Finiten-Elemente [Ver89, HSV12].

Die lokalen Beiträge η_T werden für gewöhnlich im Schritt **Schätze** als Verfeinerungsindikatoren genutzt.

Um zu zeigen, dass mit diesem Fehlerschätzer die optimale Rate erreicht wird, müssten die Axiome der Adaptivität aus [CFPP14] bewiesen werden. Da dies den Rahmen dieser Arbeit sprengen würde, wird hier nur die Verlässlichkeit des Fehlerschätzers bewiesen.

Satz 5.4 (Verlässlichkeit des Fehlerschätzers)**.** *Der Fehlerschätzer η ist zuverlässig, das heißt für den Geschwindigkeitsfehler zwischen der exakten Lösung $(\boldsymbol{u}, p) \in \boldsymbol{Z} \times Q$ und der diskreten Lösung $(\boldsymbol{u}_h, p_h) \in \boldsymbol{Z}_k \times Q_k$ gilt*

$$\|\mathrm{D}(\boldsymbol{u} - \boldsymbol{u}_h)\|_{L^2(\Omega)} \leq C_{\mathrm{R}}\eta(\boldsymbol{u}_h, p_h)$$

mit einer Verlässlichkeitskonstante $0 < C_{\mathrm{R}} \in \mathbb{R}$.

Beweis. Aus Lemma 5.1 folgt, dass

$$\nu\|\mathrm{D}(\boldsymbol{u} - \boldsymbol{u}_h)\|_{L^2(\Omega)} = \sup_{\boldsymbol{v}\in\boldsymbol{Z}} \frac{|\mathrm{Res}(\boldsymbol{u}_\mathrm{h})(\boldsymbol{v})|}{\|\mathrm{D}\boldsymbol{v}\|_{L^2(\Omega)}},$$

weshalb im Folgenden das Residuum betrachtet wird.

Für jedes $\boldsymbol{v} \in \boldsymbol{Z}$ sei $\boldsymbol{v}_h := I_F \boldsymbol{v} \in \boldsymbol{V}_k$ die Fortin-Interpolation aus dem Satz 3.25. Dann gilt wegen der Gleichung (3.28), dass $\boldsymbol{v}_h$ divergenzfrei ist und somit $\boldsymbol{v}_h \in \boldsymbol{Z}_k$. Durch Additon von Nullen und Ausnutzen der Lösungseigenschaft von $\boldsymbol{u}$ und $\boldsymbol{u}_h$ folgt

$$\begin{aligned}
\operatorname{Res}(\boldsymbol{u}_h)(\boldsymbol{v}) =& a(\boldsymbol{u} - \boldsymbol{u}_h, \boldsymbol{v}) = (\boldsymbol{f}, \boldsymbol{v})_{L^2(\Omega)} - a(\boldsymbol{u_h}, \boldsymbol{v_h}) - a(\boldsymbol{u_h}, \boldsymbol{v} - \boldsymbol{v_h}) \\
=& \underbrace{(\boldsymbol{f}, \boldsymbol{v} - \boldsymbol{v}_h)_{L^2(\Omega)} - a(\boldsymbol{u_h}, \boldsymbol{v} - \boldsymbol{v_h})}_{=:(A)} \underbrace{- a(\boldsymbol{u_h}, \boldsymbol{v_h}) + a_h(\boldsymbol{u}_h, \boldsymbol{v}_h)}_{=:(B)} \\
&+ \underbrace{(\boldsymbol{f} - \boldsymbol{f}_h, \boldsymbol{v}_h)_{L^2(\Omega)}}_{=:(C)} \\
=& (A) + (B) + (C).
\end{aligned}$$

Die Terme werden nun separat abgeschätzt.

Behauptung 1: Der erste Term lässt sich durch

$$\begin{aligned}
(A) \leq & \left(\sum_{T \in \mathcal{T}_h} h_T^2 \left\| \boldsymbol{f}_h - \nabla p_h + \nu \Delta \Pi_k^{\nabla, T} \boldsymbol{u}_h \right\|_{L^2(T)}^2 \right)^{1/2} \|\mathrm{D}\boldsymbol{v}\|_{L^2(\Omega)} \\
&+ \left(\sum_{E \in \mathcal{E}(T)} h_E \left\| \left[(\nu \mathrm{D}(\Pi_k^{\nabla, T} \boldsymbol{u}_h) + p_h) \boldsymbol{n}_E \right] \right\|_{L^2(E)}^2 \right)^{1/2} \|\mathrm{D}\boldsymbol{v}\|_{L^2(\Omega)} \\
&+ \nu \left(\sum_{T \in \mathcal{T}_h} S^T \left(\left(\boldsymbol{I} - \Pi_k^{\nabla, T} \right) \boldsymbol{u}_h, \left(\boldsymbol{I} - \Pi_k^{\nabla, T} \right) \boldsymbol{u}_h \right) \right)^{1/2} \|\mathrm{D}\boldsymbol{v}\|_{L^2(\Omega)}
\end{aligned}$$

abschätzen.

Beweis von Behauptung 1. Durch Addition und Subtraktion von $\Pi_k^{\nabla, T} \boldsymbol{u}_h$ und p_h sowie mit Hilfe partieller Integration lässt sich

$$\begin{aligned}
(A) &= \sum_{T \in \mathcal{T}_h} \left((\boldsymbol{f}, \boldsymbol{v} - \boldsymbol{v}_h)_{L^2(T)} - a^T(\Pi_k^{\nabla, T} \boldsymbol{u}_h, \boldsymbol{v} - \boldsymbol{v}_h) + a^T(\Pi_k^{\nabla, T} \boldsymbol{u}_h - \boldsymbol{u}_h, \boldsymbol{v} - \boldsymbol{v}_h) \right) \\
&= \sum_{T \in \mathcal{T}_h} (\boldsymbol{f} - \nabla p_h + \nu \Delta (\Pi_k^{\nabla, T} \boldsymbol{u}_h), \boldsymbol{v} - \boldsymbol{v}_h)_{L^2(T)} \\
&\quad + \sum_{E \in \mathcal{E}} \int_E \left[(\nu \mathrm{D}(\Pi_k^{\nabla, T} \boldsymbol{u}_h) + p_h) \boldsymbol{n}_E \right] \cdot (\boldsymbol{v} - \boldsymbol{v}_h) \, \mathrm{d}E + \sum_{T \in \mathcal{T}_h} a^T(\Pi_k^{\nabla, T} \boldsymbol{u}_h - \boldsymbol{u}_h, \boldsymbol{v} - \boldsymbol{v}_h)
\end{aligned}$$

schlussfolgern. Die ersten beiden Summen können mit der Cauchy-Schwarz-Ungleichung,

der Spurgleichung sowie den Approximationseigenschaften der Interpolation durch

$$\begin{aligned}
&\sum_{T\in\mathcal{T}_h} (\boldsymbol{f} - \nabla p_h + \nu\Delta(\Pi_k^{\nabla,T}\boldsymbol{u}_h), \boldsymbol{v} - \boldsymbol{v}_h)_{L^2(T)} \\
&\qquad + \sum_{E\in\mathcal{E}} \int_E \left[(\nu \mathrm{D}(\Pi_k^{\nabla,T}\boldsymbol{u}_h) + p_h)\boldsymbol{n}_E\right] \cdot (\boldsymbol{v} - \boldsymbol{v}_h)\,\mathrm{d}E \\
&\leq \left(\sum_{T\in\mathcal{T}_h} h_T^2 \|\boldsymbol{f} - \nabla p_h + \nu\Delta(\Pi_k^{\nabla,T}\boldsymbol{u}_h)\|_{L^2(T)}^2\right)^{1/2} \|\mathrm{D}\boldsymbol{v}\|_{L^2(\Omega)} \\
&\qquad + \left(\sum_{E\in\mathcal{E}} h_E \| \left[(\nu \mathrm{D}(\Pi_k^{\nabla,T}\boldsymbol{u}_h) + p_h)\boldsymbol{n}_E\right] \|_{L^2(E)}^2\right)^{1/2} \|\mathrm{D}\boldsymbol{v}\|_{L^2(\Omega)}
\end{aligned}$$

abgeschätzt werden. Die letzte Summe lässt sich mittels der Cauchy-Schwarz-Ungleichung, der Stabilität der Bilinearform und der Beschränktheit des Fortin-Operators (3.29) durch

$$\begin{aligned}
&\sum_{T\in\mathcal{T}_h} a^T\left(\Pi_k^{\nabla,T}\boldsymbol{u}_h - \boldsymbol{u}_h, \boldsymbol{v} - \boldsymbol{v}_h\right) \\
&\qquad\qquad \leq \nu^{1/2} \sum_{T\in\mathcal{T}_h} \left(a^T\left(\left(\boldsymbol{I} - \Pi_k^{\nabla,T}\right)\boldsymbol{u}_h, \left(\boldsymbol{I} - \Pi_k^{\nabla,T}\right)\boldsymbol{u}_h\right)\right)^{1/2} \|\mathrm{D}\boldsymbol{v}\|_{L^2(\Omega)} \\
&\qquad\qquad \lesssim \nu \left(\sum_{T\in\mathcal{T}_h} S^T\left(\left(\boldsymbol{I} - \Pi_k^{\nabla,T}\right)\boldsymbol{u}_h, \left(\boldsymbol{I} - \Pi_k^{\nabla,T}\right)\boldsymbol{u}_h\right)\right)^{1/2} \|\mathrm{D}\boldsymbol{v}\|_{L^2(\Omega)}
\end{aligned}$$

abschätzen, womit Behauptung 1 folgt. □

Behauptung 2: Für den zweiten Term gilt

$$(B) \lesssim \nu \left(\sum_{T\in\mathcal{T}_h} S^T\left(\left(\boldsymbol{I} - \Pi_k^{\nabla,T}\right)\boldsymbol{u}_h, \left(\boldsymbol{I} - \Pi_k^{\nabla,T}\right)\boldsymbol{u}_h\right)\right)^{1/2} \|\mathrm{D}\boldsymbol{v}\|_{L^2(\Omega)}.$$

Beweis von Behauptung 2. Aus der Definition von a_h und der Definition von $\Pi_k^{\nabla,T}$ folgt

$$\begin{aligned}
(B) &= \sum_{T\in\mathcal{T}_h} \Big(a^T\left(\Pi_k^{\nabla,T}\boldsymbol{u}_h, \Pi_k^{\nabla,T}\boldsymbol{v}_h\right) - a^T(\boldsymbol{u}_h, \boldsymbol{v}_h) \\
&\qquad\qquad + \nu S^T\left(\left(\boldsymbol{I} - \Pi_k^{\nabla,T}\right)\boldsymbol{u}_h, \left(\boldsymbol{I} - \Pi_k^{\nabla,T}\right)\boldsymbol{v}_h\right)\Big) \\
&= \sum_{T\in\mathcal{T}_h} \Big(- a^T\left(\left(\boldsymbol{I} - \Pi_k^{\nabla,T}\right)\boldsymbol{u}_h, \left(\boldsymbol{I} - \Pi_k^{\nabla,T}\right)\boldsymbol{v}_h\right) \\
&\qquad\qquad + \nu S^T\left(\left(\boldsymbol{I} - \Pi_k^{\nabla,T}\right)\boldsymbol{u}_h, \left(\boldsymbol{I} - \Pi_k^{\nabla,T}\right)\boldsymbol{v}_h\right)\Big) \\
&\lesssim 2 \sum_{T\in\mathcal{T}_h} a^T\left(\left(\boldsymbol{I} - \Pi_k^{\nabla,T}\right)\boldsymbol{u}_h, \left(\boldsymbol{I} - \Pi_k^{\nabla,T}\right)\boldsymbol{v}_h\right),
\end{aligned}$$

wobei im letzten Schritt die Stabilitätseigenschaft von S^T ausgenutzt wird. Die Cauchy-Schwarz-Ungleichung, erneut die Stabilitätseigenschaft und Approximationseigenschaften der Interpolation führen zu

$$\begin{aligned}(B) &\lesssim \sum_{T\in\mathcal{T}_h} \left(\nu\|\mathrm{D}(\boldsymbol{u}-\Pi_k^{\nabla,T}\boldsymbol{u}_h)\|_{L^2(T)}\|\mathrm{D}(\boldsymbol{v}_h-\Pi_k^{\nabla,T}(\boldsymbol{v}_h))\|_{L^2(T)}\right)\\ &\lesssim \nu\left(\sum_{T\in\mathcal{T}_h} S^T\left(\left(\boldsymbol{I}-\Pi_k^{\nabla,T}\right)\boldsymbol{u}_h,\left(\boldsymbol{I}-\Pi_k^{\nabla,T}\right)\boldsymbol{u}_h\right)\right)^{1/2}\|\mathrm{D}\boldsymbol{v}\|_{L^2(\Omega)},\end{aligned}$$

was die Behauptung 2 ist. □

Behauptung 3: Der dritte Term kann durch

$$(C)\lesssim\left(\sum_{T\in\mathcal{T}_h} h_T^2\|\boldsymbol{f}-\boldsymbol{f}_h\|_{L^2(T)}^2\right)^{1/2}\|\mathrm{D}\boldsymbol{v}\|_{L^2(\Omega)}$$

beschränkt werden.

Beweis von Behauptung 3. Aus der Definition von $\boldsymbol{f}_h$ folgt, dass $\boldsymbol{f}-\boldsymbol{f}_h$ orthogonal auf Polynomen vom Grad $k-2$ stehen. Daher gilt wegen der Cauchy-Schwarz-Ungleichung und den Approximationseigenschaften der Interpolation, dass

$$\begin{aligned}(C)&=\sum_{T\in\mathcal{T}_h}(\boldsymbol{f}-\boldsymbol{f}_h,\boldsymbol{v}_h-\Pi_0\boldsymbol{v}_h)_{L^2(T)}\le\sum_{T\in\mathcal{T}_h}\|\boldsymbol{f}-\boldsymbol{f}_h\|_{L^2(T)}\|\boldsymbol{v}_h-\Pi_0\boldsymbol{v}_h\|_{L^2(T)}\\ &\lesssim\sum_{T\in\mathcal{T}_h} h_T\|\boldsymbol{f}-\boldsymbol{f}_h\|_{L^2(T)}\|\mathrm{D}\boldsymbol{v}_h\|_{L^2(T)}\\ &\lesssim\left(\sum_{T\in\mathcal{T}_h} h_T^2\|\boldsymbol{f}-\boldsymbol{f}_h\|_{L^2(T)}^2\right)^{1/2}\|\mathrm{D}\boldsymbol{v}\|_{L^2(\Omega)},\end{aligned}$$

was der Behauptung entspricht. □

Die Kombination von Behauptungen 1, 2 und 3 zusammen mit einer Division durch ν beenden den Beweis. □

Wie weiter oben bemerkt, gelten im inhomogenen Fall die obigen Aussagen nur für exakt aufgelöste Randdaten. Die Schwierigkeit dabei liegt in dem Lemma 5.1, wo im Beweis $\boldsymbol{u}-\boldsymbol{u}_h\in\boldsymbol{Z}_k$ nur für homogene Randdaten gilt. Eine Verallgemeinerung für inhomogene Randdaten $\boldsymbol{u}_D$ lässt sich unter anderem in [LMS19] finden. Dort wird die Korrekturfunktion für die Randdaten eingeführt, die zu einem zusätzlichen Term im Fehlerschätzer führt, der sich wie in [BCD04] abschätzen lässt. Der Fehlerschätzer für inhomogene Randdaten $\widehat{\eta}_T^2(\boldsymbol{u}_h,p_h,\boldsymbol{u}_D)$ wird damit zu

$$\widehat{\eta}_T^2(\boldsymbol{u}_h,p_h,\boldsymbol{u}_D):=\eta_T^2(\boldsymbol{u}_h,p_h)+\sum_{E\in\mathcal{E}(T)\cap\mathcal{E}(\partial\Omega)} h_E^3\left\|\partial^2/\partial s^2(\boldsymbol{u}_D-\boldsymbol{u}_h)\right\|_{L^2(E)}^2,\qquad(5.2)$$

wobei $\partial^2/\partial s^2$ die zweite partielle Ableitung in tangentialer Richtung bezeichnet.

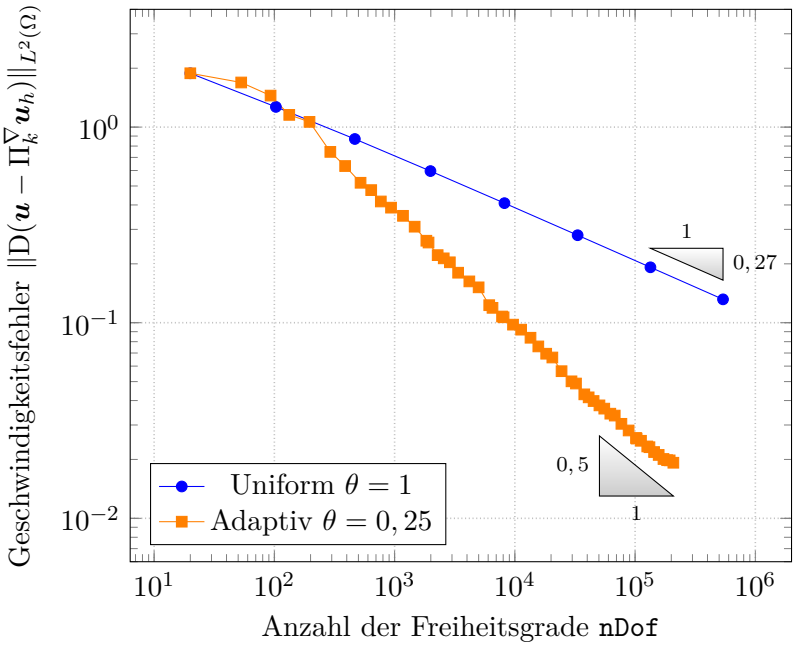


Abbildung 5.2.: Dargestellt ist der Vergleich von uniformer und adaptiver Konvergenzhistorie für Problem L auf einer Zerlegung des L-Gebiets in Quadrate. Der adaptive Durchlauf hat eine Konvergenzrate von etwa 1 bezogen auf den mittleren Druchmesser h_m. Die suboptimale Rate hat ihren Ursprung in der Verfeinerung, die die Formregularität nicht erhält.

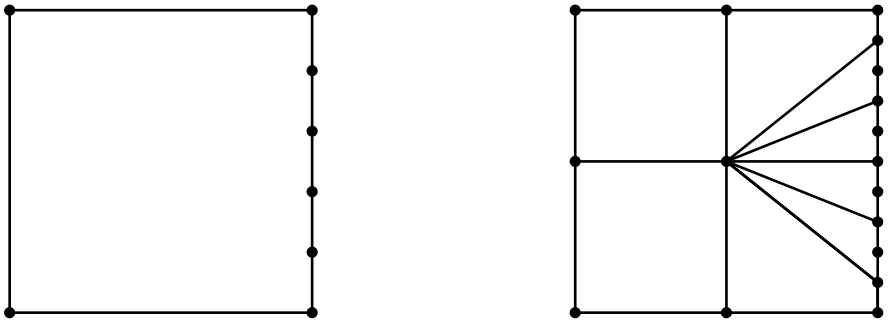
5.2. Numerische Experimente zur adaptiven Verfeinerung

Der im vorherigen Kapitel eingeführte Fehlerschätzer wird nun numerisch untersucht. Dafür wird erneut das Problem L aus Abschnitt 5.1.2 betrachtet. Es wird die gleiche exakte Lösung $(\boldsymbol{u}, p)$ genutzt und die Viskosität wieder auf $\nu = 1$ gesetzt.

5.2.1. Problem L: Erneut das L-Gebiet zerlegt in Quadrate

Zuerst wird wieder die Zerlegung des L-Gebiets in drei gleiche Quadrate aus Abbildung 4.5 als Ausgangszerlegung benutzt. Dafür wird mit uniformer Verfeinerung sowohl für den Geschwindigkeits- als auch den Druckfehler eine suboptimale Konvergenzrate von etwa 0,54 festgestellt (s. Tab 4.2, S. 53). Ausgehend vom gleichen Gitter wird nun mit Hilfe des Fehlerschätzers eine adaptive Verfeinerung durchgeführt und mit der uniformen Verfeinerung verglichen. Der Geschwindigkeitsfehler $\|\mathrm{D}(\boldsymbol{u} - \Pi_k^{\nabla} \boldsymbol{u}_h)\|_{L^2(\Omega)}$ ist gegen die Anzahl der Freiheitsgrade `nDof` in Abbildung 5.2 dargestellt.

Erkennbar ist, dass die adaptive Verfeinerung eine deutliche Steigerung der Konvergenzrate von etwa 0,54 auf etwa 1 bezogen auf den mittleren Durchmesser h_m bewirkt. Allerdings ist eine Rate von 1 noch weit entfernt von der optimalen Rate von 2. Eine



(a) Ursprüngliches Polygon (b) Verfeinerung

Abbildung 5.3.: Illustriert ist die Verfeinerungsstrategie für ein Polygon mit hängenden Knoten. Das urspüngliche Polygon ist in (a) gezeigt und die neuen Polygone sind in (b) dargestellt. Wird ein Polygon mit hängenden Knoten verfeinert, kann das Verhältnis von Seitenlänge zu Durchmesser sehr klein werden, was die Formregularität zerstört.

Ursache dafür ist, dass die benutzte Verfeinerungsstrategie nicht die Formregularität erhält. Durch das Verfeinern von Kanten mit hängenden Knoten wird Voraussetzung (A3) der Formregularität, die das Verhältnis von Kante zu Durchmesser eines Polygons beschränkt, verletzt. Eine Illustration der Verfeinerung für Polygone mit hängenden Knoten ist in Abbildung 5.3 gezeigt.

Hat eine Kante einen hängenden Knoten, betrachten die Virtuelle-Elemente-Methode und der Verfeinerungsalgorithmus die Kante als zwei Kanten, weshalb die Verfeinerungsroutine jeweils die Strecke zwischen dem hängenden Knoten und dem Start- beziehungsweise Endpunkt der Kante halbiert. Dadurch werden die neu eingefügten Kanten sehr kurz im Verhältnis zum Abstand zum Baryzentrum, welches ebenfalls ein neuer Punkt in den neuen Polygonen ist.

Dies tritt vor allen Dingen an Polygonen auf, die nahe um den Ursprung im L-Gebiet liegen. Numerisch kann dies auch an Hand sehr kleiner Konditionszahlen der Systemmatrix und eines großen Residuums des Gleichungssystem beobachtet werden. In den letzten beiden Verfeinerungsschritten liegt die Konditionszahl des zu lösenden linearen Gleichungssystems in der Größenordnung von etwa 10^{-24} und das Residuum des Gleichungssystems bei etwa 10^{-7}.

Dies würde mit der ursprünglichen Verfeinerung von Dr. O. Sutton nicht passieren, da diese in der Verfeinerung nur planare Kanten verfeinert und dafür hängende Knoten ignoriert [Sut17]. Daher lässt sich schlussfolgern, dass die hier genutzte Verfeinerungsroutine in dieser Form der Virtuellen-Elemente-Methode nicht benutzt werden sollte und die ursprüngliche Form von Dr. O. Sutton zu bevorzugen ist.

Eine Möglichkeit, diese Verfeinerungsroutine dennoch nutzen zu können, könnte sein, eine andere Stabilisierung zu wählen. In [dLR17] wird ein alternativer Stabilisierungsterm vorgeschlagen, der für die Konvergenz weniger Formregularität des Gitters fordert. Dort wird nämlich gezeigt, dass die Bedingung (A3) der Formregularität weggelassen

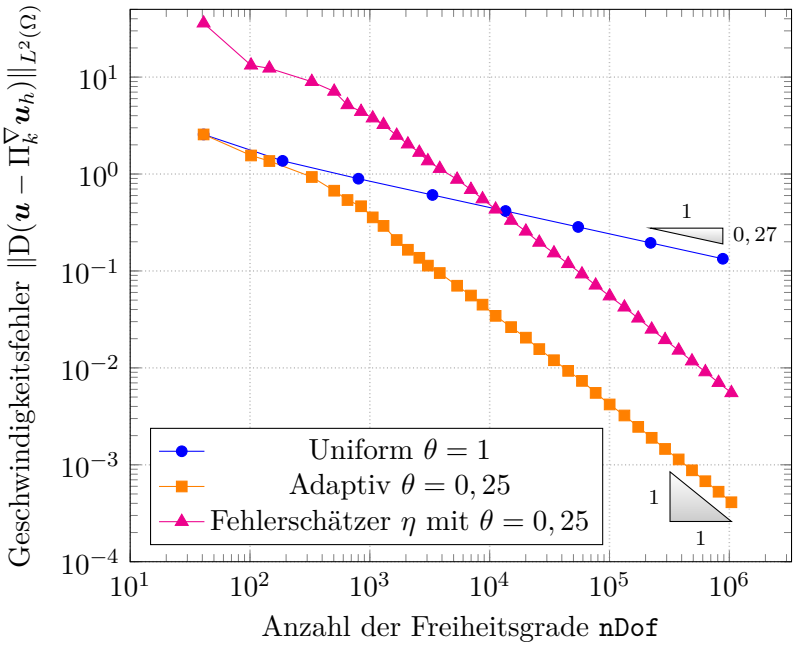


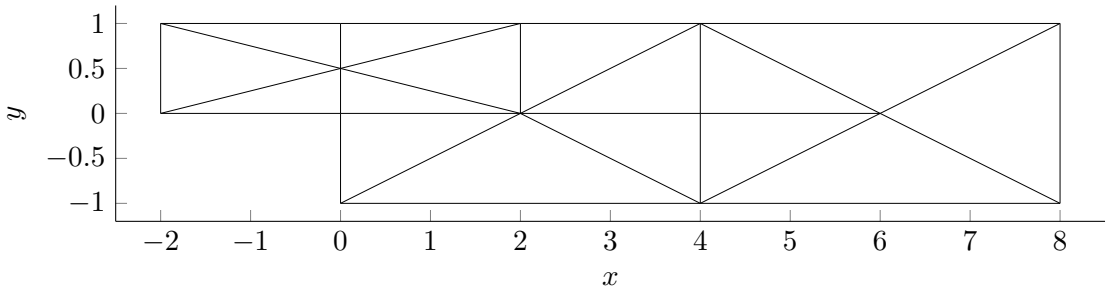
Abbildung 5.4.: Gezeigt ist ein Vergleich von adaptiver und uniformer Konvergenzhistorie für Problem L auf einer Zerlegung des L-Gebiets in Dreiecke. Zur Verfeinerung wird der Rot-Grün-Blau-Verfeinerungsalgorithmus aus [CGK+10] genutzt. Im adaptiven Durchlauf verläuft der Fehlerschätzer η parallel zum Fehler des adaptiven Durchlaufs. Die adaptive Verfeinerungsstrategie führt zu einer optimalen Rate von 2 in Bezug auf den mittleren Durchmesser, was die Korrektheit des Fehlerschätzers belegt.

werden kann und immer noch optimale Konvergenzraten zu erwarten sind, wenn die Lösung $\boldsymbol{u}$ in $H^s(\Omega)$ für $s \geq 3/2$ liegt [dLR17].

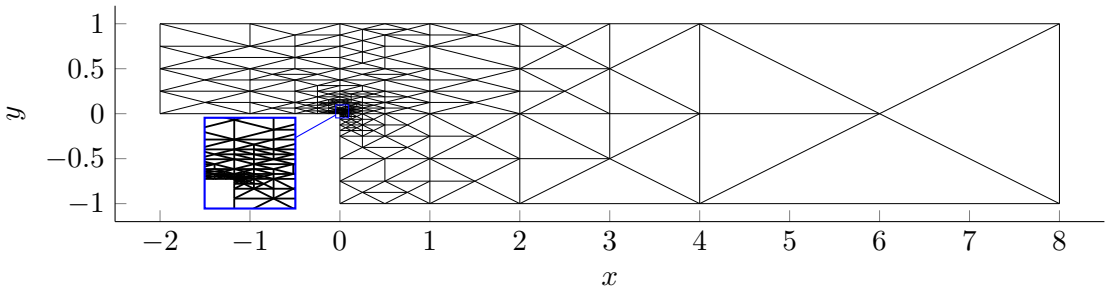
5.2.2. Problem L: Das L-Gebiet zerlegt in Dreiecke

Um die Verletzung der Formregularität auszuschließen und den Fehlerschätzer trotzdem zu prüfen, muss also das Problem L auf einer Reihe von Netzen berechnet werden, die formregulär sind. Dafür wird ausgehend von einer Zerlegung in Dreiecke der Verfeinerungsalgorithmus aus AFEM genutzt, der hängende Knoten ausschließt und dafür sorgt, dass die Formregularität erhalten bleibt [CGK+10]. Die Dreiecke werden dabei entweder rot, grün oder blau verfeinert und der closure-Algorithmus verhindert, dass hängende Knoten entstehen.

Mit den gleichen Parametern wie vorher wird das Problem L ausgehend von einem Dreiecksgitter berechnet. Die Konvergenzhistorie des Geschwindigkeitsfehler ist zusammen mit dem Fehlerschätzer η in Abbildung 5.4 gegen die Anzahl an Freiheitsgrade `nDof` abgebildet.



(a) Ausgangstriangulierung



(b) 10 adaptive Verfeinerungen

Abbildung 5.5.: Dargestellt ist die Ausgangstriangulierung (a) und adaptiv erzeugte Triangulierung (b) für das Gebiet $\Omega = (-2, 8) \times (-1, 1) \setminus (-2, 0) \times (0, -1)$ für Problem BFS. Das adaptive Gitter wurde in zehn Verfeinerungsschritten mit der Verfeinerungsroutine aus AFEM [CGK$^+$10] erzeugt und besteht aus 500 Elementen. Deutlich zu erkennen ist die Verfeinerung hin zur einspringenden Ecke, an der die Lösung eine Singularität aufweist.

Die uniforme Verfeinerung führt analog zum vorherigen Beispiel zu einer Konvergenzrate von etwa 0,54. Mit der Verfeinerungsstrategie von AFEM werden mittels adaptiver Verfeinerung allerdings signifikant bessere Ergebnisse als mit der vorher genutzten Verfeinerungsroutine erzielt, was mit der Erhaltung der Formregularität zusammenhängt. Es wird asymptotisch wieder die optimale Rate von 2 erreicht. Darüber hinaus verläuft der Fehlerschätzer parallel zum Geschwindigskeitsfehler.

Zusammenfassend spricht alles dafür, dass ein verlässlicher und effizienter Fehlerschätzer konstruiert worden ist, mit dem auch auf nicht-konvexen Gebieten die optimale Rate wieder hergestellt werden kann - vorausgesetzt der Verfeinerungsalgorithmus erhält die Formregularität.

5.2.3. Problem BFS: Die rückwärtsgewandte Stufe

Der Fehlerschätzer η soll abschließend auf ein für die Ingenieurswissenschaften relevantes Problem angewendet werden. Dafür wird die rückwärtsgewandte Stufe (engl. “backward-facing step“) gewählt. Bei diesem Problem fließt ein Fluid mit einem parabolischen

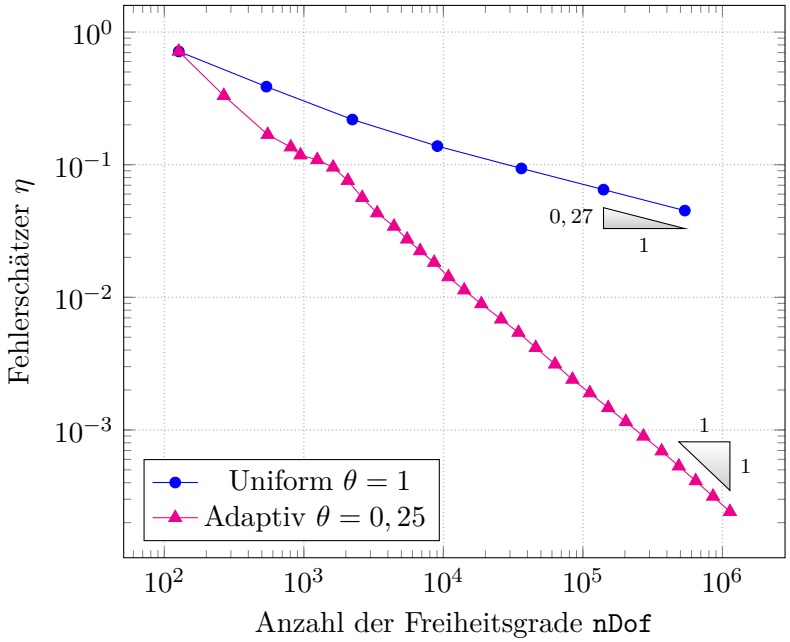


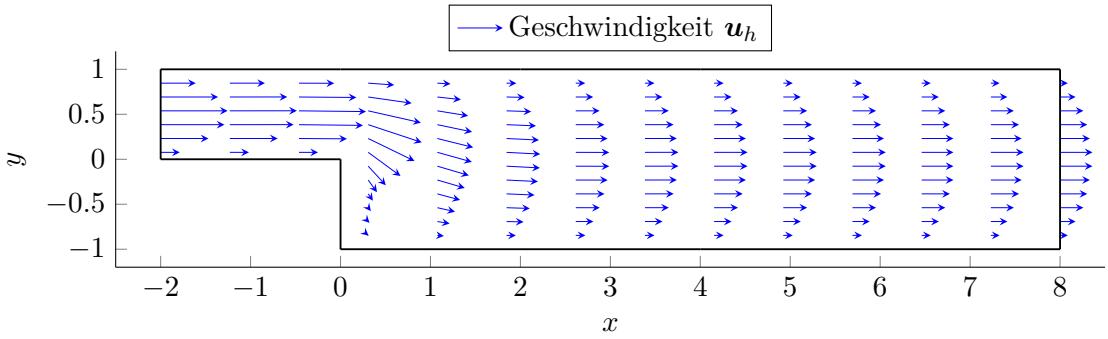
Abbildung 5.6.: Die Konvergenzhistorie für Problem BFS mit $\nu = 1$ ausgehend von der Triangulierung in Abbildung 5.5a ist hier gezeigt. Für den adaptiven Durchgang wird der Verfeinerungsalgorithmus aus AFEM benutzt [CGK+10]. Mit uniformen Verfeinerungen treten suboptimale Konvergenzraten auf, die durch Singularitäten im Ursprung des Gebiets bedingt sind. Die optimale Konvergenzrate von 2 in Bezug auf den mittleren Durchmesser kann durch adaptives Verfeinern wieder hergestellt werden.

Strömungsprofil in ein Gebiet, das sich nach einem Stück des Weges in Form einer Stufe verbreitert. Nach der Stufe bleibt der Durchmesser wieder konstant und das Fluid fließt am Ende mit einem parabolischen Profil aus dem Gebiet wieder heraus.

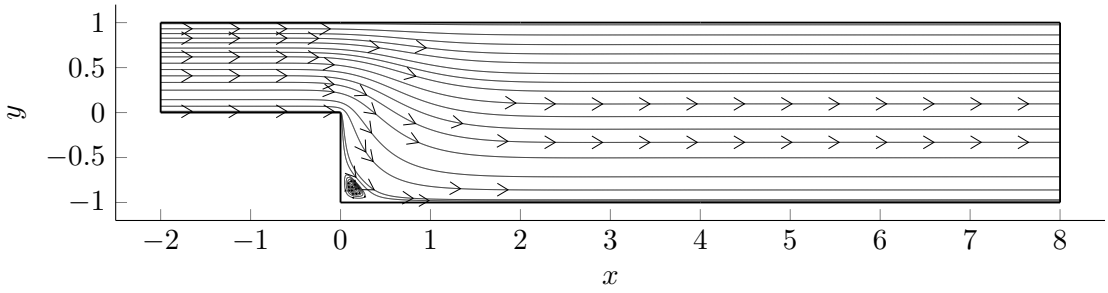
Das Gebiet $\Omega := (-2, 8) \times (-1, 1) \setminus (-2, 0) \times (-1, 0)$ ist mit der Anfangstriangulierung in Abbildung 5.5a dargestellt. Im Bild fließt das Medium von links in das Gebiet und tritt auf der rechten Seite wieder aus. Es wird $\nu = 1$ und als Dirichletrandbedingung

$$\boldsymbol{u}_D(x, y) = \begin{cases} (y(1-y)/10, 0)^t, & \text{wenn } x = -2, \\ ((1-y^2)/80, 0)^t, & \text{wenn } x = 8, \\ (0, 0)^t, & \text{sonst} \end{cases}$$

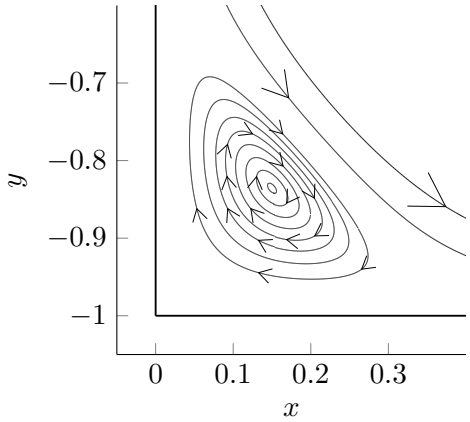
gewählt. Für dieses Problem ist für die rechte Seite $\boldsymbol{f} = 0$ keine exakte Lösung bekannt. Zur Verfeinerung wird erneut der Algorithmus aus AFEM genommen, um die Formregularität zu garantieren [CGK+10]. Die Konvergenzhistorie ist für eine Serie uniformer Gitter und eine Serie adaptiv erzeugter Gitter mit Parameter $\theta = 0, 25$ in Abbildung 5.6 dargestellt.



(a) Geschwindigkeitsfeld



(b) Stromlinien



(c) Vergrößerung um den Wirbel

Abbildung 5.7.: Abgebildet sind das Geschwindigkeitsfeld (a), die Stromlinien (b) sowie eine Vergrößerung um den Wirbel (c) für Problem BFS. Zur besseren Darstellung ist die Geschwindigkeitslösung auf einem uniformen Gitter interpoliert und skaliert um den Faktor 40 dargestellt. In (c) sind die Stromlinien zur Verdeutlichung der Richtung des Stroms mit Pfeilspitzen versehen. Gut zu erkennen ist das parabolische Strömungsprofil auf der linken und auf der rechten Seite, sowie eine Beschleunigung im Bereich der Stufe. Außerdem ist der Moffatt-Wirbel gut aufgelöst.

Der uniforme Durchlauf führt zu einer Konvergenzrate von etwa $0,54$ bezogen auf den mittleren Durchmesser h_m, was ähnlich zu der üblichen Konvergenzrate für das Problem ist [CP18]. Bedingt wird diese suboptimale Rate durch die einspringende Ecke im Ursprung des Gebiets, die eine Singularität in der Lösung hervorruft.

Mit Hilfe des adaptiven Fehlerschätzers kann die optimale Konvergenzrate von 2 wieder hergestellt werden, was wie in dem vorherigen Problem die Annahme unterstützt, dass der Fehlerschätzer verlässlich und effizient ist.

Das Gitter nach zehn adaptiven Verfeinerungen bestehend aus 500 Elementen mit 4427 Freiheitsgraden ist in Abbildung 5.5b dargestellt. Deutlich zu erkennen ist eine Verfeinerung in Richtung der einspringenden Ecke. Dies kann auch so erwartet werden, da dort die Lösung eine Singularität aufweist.

Die zur besseren Darstellung auf einem äquidistanten Gitter interpolierte und um den Faktor 40 skalierte Geschwindigkeitslösung wird zusammen mit den Stromlinien in Abbildung 5.7 gezeigt. Das Medium fließt mit dem vorgegebenen parabolischen Strömungsprofil von links in das Gebiet und dann am rechten Rand wieder heraus. Auf dem Weg in der Region der einspringenden Ecke wird es dabei beschleunigt. In der Ecke bei $(0, -1)$ kann ein gut aufgelöster Moffatt-Strudel gesehen werden, was für die numerische Qualität der Lösung spricht.

6. Eine druckrobuste Form der Virtuellen-Elemente-Methode

Im Kontinuierlichen hat das Stokes-Problem eine fundamentale Invarianz. Ist $(\boldsymbol{u}, p)$ eine Lösung des Stokes-Problems mit der rechten Seite $\boldsymbol{f}$, löst $(\boldsymbol{u}, p+q)$ das Stokes-Problem mit der rechten Seite $\boldsymbol{f}+\nabla q$ für alle $q \in H^1(\Omega) \cap Q$. Genau diese Invarianz bleibt bei den meisten herkömmlichen Finiten-Elemente-Methoden im Diskreten nicht mehr erhalten. Dies führt dazu, dass eine Änderung der rechten Seite, die im Kontinuierlichen nur den Druck beeinflusst, auch die diskrete Geschwindigkeit verändert [JLM$^+$17]. Dieses Verhalten wird gemeint, wenn im Folgenden von Druckrobustheit gesprochen wird. Für eine ausführlichere Einführung kann beispielsweise auf [JLM$^+$17] verwiesen werden.

Ein nicht-druckrobustes Verhalten wird auch bei Problem 2 für die bisher vorgestellte Virtuelle-Elemente-Methode beobachtet. Dies soll im Folgenden behoben werden. In Satz 3.31 ist der Geschwindigkeitsfehler zwischen der exakten Geschwindigkeit $\boldsymbol{u}$ und der diskreten $\boldsymbol{u}_h$ durch

$$\|\boldsymbol{u}-\boldsymbol{u}_h\|_{H^1(\Omega)} \lesssim \inf_{\boldsymbol{v}_h \in \boldsymbol{V}_k} \|\boldsymbol{u}-\boldsymbol{v}_h\|_{H^1(\Omega)} + \inf_{\boldsymbol{v}_\pi \in [\mathbb{P}_k(\mathcal{T}_h)]^2} \|\boldsymbol{u}-\boldsymbol{v}_\pi\|_{h,H^1(\Omega)} + \frac{1}{\nu}\mathcal{F}_h$$

abgeschätzt, wobei $\mathcal{F}_h$ die Operatornorm von $\boldsymbol{f}-\boldsymbol{f}_h = \boldsymbol{f} - P_{k-2}\boldsymbol{f}$ im Dualraum $\boldsymbol{V}_k^*$ ist. Die Operatornorm lässt sich für genügend glatte $\boldsymbol{f}$ durch $\mathcal{F}_h \leq |h_{\mathcal{T}}^k \boldsymbol{f}|_{H^{k-1}(\Omega)}$ abschätzen (s. Lem. 3.30).

Eine Möglichkeit die Virtuelle-Elemente-Methode robuster gegenüber Gradienten in der rechten Seite zu machen, ist es daher, den Polynomgrad der L^2-Projektion von $\boldsymbol{f}$ zu erhöhen. Diese Methode basiert auf den bereits bekannten sogenannten erweiterten Virtuellen-Elemente-Räumen und wird im ersten Abschnitt dieses Kapitels vorgestellt.

Eine komplett andere Idee für Finite-Elemente-Methoden verfolgt der Ansatz aus [Lin14]. Dort werden Testfunktionen mittels eines $H(\mathrm{div})$-konformen Rekonstruktionsoperators auf exakt divergenzfreie Funktionen abgebildet, um die L^2-Orthogonalität von Gradientenfeldern und Testfunktionen wieder herzustellen. Auch wenn die Testfunktionen in diser Virtuellen-Elemente-Methode divergenzfrei sind, ist es die Projektion P_{k-2} im Allgemeinen nicht, was die Orthogonalität zerstört. Diese Methode des Rekonstruktionsoperators ist allerdings bisher nur für die gängigen Finiten-Elemente-Methoden wie beispielsweise Taylor-Hood, MINI oder Bernardi-Raugel bekannt [LLMS17, ALM17]. Im zweiten Abschnitt des Kapitels wird die Rekonstruktion auf die Virtuelle-Elemente-Methode für Polygone übertragen.

Numerische Experimente am Ende des Kapitels bestätigen die theoretischen Aussagen.

6.1. Verbesserte Virtuelle-Elemente-Räume

Erweiterte Virtuelle-Elemente-Räume wurden das erste Mal in [AAB+13] vorgestellt. Sie werden schon in leicht komplizierteren Problemen wie bspw. in den Navier-Stokes-Gleichungen oder auch bei Eigenwertproblemen benötigt. Genauer werden sie immer dann benötigt, wenn die gesuchte Funktion ohne Ableitung in der kontinuierlichen Differentialgleichung auftaucht beziehungsweise aus diskreter Sicht dann, wenn die Massematrix $\int_T \varphi_j \varphi_\ell \, \mathrm{d}T$ berechnet werden soll. Dies ist mit den normalen Virtuellen-Elemente-Räumen nicht exakt möglich, denn zur Berechnung der Massematrix wäre die volle L^2-Projektion auf Polynome vom Grad kleiner oder gleich k nötig [AAB+13].

In Abschnitt 3.1 wird erläutert, dass nur die Bestapproximation bezüglich der Energienorm Π_k^∇ auf Polynome vom Grad kleiner oder gleich k bzw. die volle L^2-Projektion P_{k-2} nur auf Polynome vom Grad maximal $k-2$ explizit vorhanden sind. Mit den erweiterten Virtuellen-Elemente-Räumen ist die Projektion auf Polynome vom Grad kleiner oder gleich k möglich und das sogar mit den gleichen Freiheitsgraden wie vorher [AAB+13].

Daher werden im Folgenden zuerst die erweiterten Virtuellen-Elemente-Räume und anschließen deren Implementierung vorgestellt. Dieser Abschnitt 6.1 und die Beweise sind sinngemäß aus [DV19, Vac17, AAB+13] entnommen.

6.1.1. Einführung der erweiterten Virtuellen-Elemente-Räume

Die Hauptidee der erweiterten Virtuellen-Elemente-Räume liegt darin, einen geringfügig geänderten Raum $\boldsymbol{W}_k^T$ einzuführen,

- in dem die gleichen Freiheitsgrade wie von $\boldsymbol{V}_k^T$ benutzt werden können,
- der die Polynome vom Grad kleiner oder gleich k immer noch enthält,
- in dem weiterhin die Bestapproximation bezüglich der Energienorm $\Pi_k^{\nabla,T}$ exakt berechnet werden kann und
- in dem zusätzlich die volle L^2-Projektion auf Polynome vom Grad kleiner oder gleich k zur Verfügung stehen soll [AAB+13].

Solch ein Raum steht tatsächlich zur Verfügung und wurde in [Vac17] eingeführt sowie auf seine Eigenschaften untersucht.

Der vergrößerte lokale Virtuelle-Elemente-Raum sei durch

$$\begin{aligned}\boldsymbol{U}_k^T := \{\boldsymbol{v}_h \in [H^1(T)]^2 \,:\, &\boldsymbol{v}_{h|\partial T} \in [\mathbb{B}_k(T)]^2,\, -\nu\boldsymbol{\Delta}\boldsymbol{v}_h + \nabla s \in \mathcal{K}_k(T) \text{ für ein } s \in L^2(T) \\ &\mathrm{div}\boldsymbol{v}_h \in \mathbb{P}_{k-1}(T)\}\end{aligned}$$

definiert. Der Unterschied von $\boldsymbol{U}_k^T$ und $\boldsymbol{V}_k^T$ liegt in der Eigenschaft, dass für die Elemente $\boldsymbol{v}_h \in \boldsymbol{U}_k^T$ gilt, dass $-\nu\boldsymbol{\Delta}\boldsymbol{v}_h + \nabla s$ im Komplementärraum vom Grad k im Gegensatz zum Grad $k-2$ für $\boldsymbol{V}_k^T$ liegt. Dies erhöht daher auch die Anzahl der Freiheitsgrade. Die Dimension und die Freiheitsgrade von $\boldsymbol{U}_k^T$ sind in folgendem Lemma zusammengefasst

Lemma 6.1. *Es gelten die folgenden Aussagen.*

(i) Für die Dimension von $\boldsymbol{U}_k^T$ gilt, dass

$$\dim \boldsymbol{U}_k^T = 2|\mathcal{N}(T)|k + \frac{(k+1)k}{2} + \frac{(k+1)k}{2} - 1.$$

(ii) Als Freiheitsgrade von $\boldsymbol{v}_h \in \boldsymbol{U}_k^T$ können die Freiheitsgrade von $\boldsymbol{V}_k^T$ und zusätzlich noch

$$D_{\boldsymbol{U}} : \quad \frac{1}{|T|}\int_T \boldsymbol{v}_h \cdot \boldsymbol{m}^\perp m_j \,\mathrm{d}T$$

für $m_j \in \mathbb{M}_{k-1}(T) \setminus \mathbb{M}_{k-3}(T)$ genommen werden.

Beweis. Beide Aussagen folgen eins zu eins der Argumentation für $\boldsymbol{V}_k^T$ aus Abschnitt 3.1. □

Damit kann nun der erweiterte Virtuelle-Elemente-Raum $\boldsymbol{W}_k^T$ eingeführt werden, der im Wesentlichen eine geschickte Einschränkung von $\boldsymbol{U}_k^T$ ist.

Definition 6.2 (Erweiterter lokaler Virtueller-Elemente-Raum)**.** Der erweiterte lokale Virtuelle-Elemente-Raum ist durch

$$\boldsymbol{W}_k^T := \Big\{ \boldsymbol{v}_h \in \boldsymbol{U}_k^T \,:\, (\boldsymbol{v}_h - \Pi_k^{\nabla,T}\boldsymbol{v}_h, \boldsymbol{m}^\perp p_{k-1})_{L^2(T)} = 0 \text{ für alle } p_{k-1} \in \mathbb{P}_{k-1}(T) \setminus \mathbb{P}_{k-3}(T) \Big\}$$

gegeben.

Die Dimension und die Freiheitsgrade sind erneut in dem folgendem Lemma zusammengefasst.

Lemma 6.3 (Eigenschaften von $\boldsymbol{W}_k^T$)**.** *Es gelten die folgenden Aussagen.*

(i) Die Dimension von $\boldsymbol{W}_k^T$ ist gleich der Dimension von $\boldsymbol{V}_k^T$.

(ii) Es können die gleichen Freiheitsgrade für $\boldsymbol{W}_k^T$ wie für $\boldsymbol{V}_k^T$ benutzt werden.

Beweis. Einfaches Nachrechnen zeigt, dass

$$\dim\left(\mathbb{P}_{k-1}(T) \setminus \mathbb{P}_{k-3}(T)\right) = \dim \mathbb{P}_{k-1}(T) - \dim \mathbb{P}_{k-3}(T) = 2k - 1.$$

Deshalb folgt unter Vernachlässigung etwaiiger linearer Unabhängigkeit der Bedingungen in der Definition von $\boldsymbol{W}_k^T$, dass

$$\dim \boldsymbol{W}_k^T \geq \dim \boldsymbol{U}_k^T - (2k-1) = \dim \boldsymbol{V}_k^T.$$

Sei jetzt eine Funktion $w_h \in \boldsymbol{W}_k^T$ mit $D_{\boldsymbol{V}}^{\mathcal{N}}(\boldsymbol{w}_h) = D_{\boldsymbol{V}}^{\mathcal{E}}(\boldsymbol{w}_h) = D_{\boldsymbol{V}}^{\mathcal{M}_1}(\boldsymbol{w}_h) = D_{\boldsymbol{V}}^{\mathcal{M}_2} = 0$ vorgegeben. Aus der Definition von $\Pi_k^{\nabla,T}$ folgt sofort, dass auch $\Pi_k^{\nabla,T}\boldsymbol{w}_h = 0$, weshalb auch $D_{\boldsymbol{U}}(\boldsymbol{w}_h) = 0$ gilt. Durch Lemma 6.1 folgt, dass $\boldsymbol{w}_h = 0$ ist, woraus sich ergibt, dass die Dimension von $\boldsymbol{W}_k^T$ und $\boldsymbol{V}_k^T$ gleich sind und die gleichen Freiheitsgrade verwendet werden können. □

Der globale Raum $\boldsymbol{W}_k$ ergibt sich analog zu $\boldsymbol{V}_k$ mittels

$$\boldsymbol{W}_k := \left\{ \boldsymbol{v}_h \in H_0^1(\Omega) \,:\, \boldsymbol{v}_{h|T} \in \boldsymbol{W}_k^T \text{ für alle } T \in \mathcal{T}_h \right\}.$$

Es gibt nun also zwei Räume $\boldsymbol{V}_k$ und $\boldsymbol{W}_k$, für die die gleichen Freiheitsgrade benutzt werden können. Haben darüber hinaus eine Funktion aus dem ersten und eine Funktion aus dem zweiten Raum die gleichen Werte an den Freiheitsgraden, sind es zwar unterschiedliche Funktionen, aber die polynomielle Approximation $\Pi_k^{\nabla,T}$ ist für alle $T \in \mathcal{T}_h$ die gleiche. Sind die Funktionen selbst sogar Polynome, dann sind auch die Funktionen identisch. Der Unterschied zwischen Funktionen aus beiden Räumen ist, dass sich die volle L^2-Projektion auf Polynome vom Grad maximal k nur im letzten Fall berechnen lassen, wie im Folgenden gezeigt wird.

Zur Berechnung der vollen Projektion $P_k^T \boldsymbol{v}_h \in \mathbb{P}_k(T)$ einer Funktion $\boldsymbol{v}_h \in \boldsymbol{W}_k$ auf Polynome vom Grad kleiner oder gleich k muss

$$(P_k^T \boldsymbol{v}_h - \boldsymbol{v}_h, \boldsymbol{q}_k)_{L^2(T)} = 0 \qquad \text{für alle } \boldsymbol{q}_k \in [\mathbb{P}_k(T)]^2$$

berechnet werden. Dafür wird jedes $\boldsymbol{q}_k$ nach Gleichung (3.1) in einen Gradientenanteil ∇q_{k+1} für ein $q_{k+1} \in \mathbb{P}_{k+1}(T)$ und einen dazu komplementären Anteil $\boldsymbol{m}^\perp q_{k-1}$ für ein $q_{k-1} \in \mathbb{P}_{k-1}(T)$ zerlegt, woraus sich mit partieller Integration

$$\begin{aligned}(P_k^T \boldsymbol{v}_h, \boldsymbol{q}_k)_{L^2(T)} &= (\boldsymbol{v}_h, \boldsymbol{q}_k)_{L^2(T)} = (\boldsymbol{v}_h, \nabla q_{k+1} + \boldsymbol{m}^\perp q_{k-1})_{L^2(T)} \\ &= -(\mathrm{div}\boldsymbol{v}_h, q_{k+1})_{L^2(T)} + (\boldsymbol{v}_h, q_{k+1}\boldsymbol{n})_{L^2(\partial T)} + (\boldsymbol{v}_h, \boldsymbol{m}^\perp q_{k-1})_{L^2(T)} \quad (6.1)\end{aligned}$$

ergibt. Die ersten beiden Summanden können mit den Freiheitsgraden berechnet werden, da sowohl die Divergenz von $\boldsymbol{v}_h$ in T als auch die Funktion selbst auf ∂T Polynome vom Grad $k-1$ bzw. k und damit explizit berechenbar sind. Für die Berechnung des Randintegrals muss die Funktion auf dem Rand mit Hilfe der Freiheitsgrade rekonstruiert werden und dann mit einer Integrationsformel vom Grad $2k+1$ berechnet werden. Für das letzte Integral aus Gleichung (6.1) kann im Falle $q_{k-1} \in \mathbb{P}_{k-1}(T) \setminus \mathbb{P}_{k-3}(T)$ nun die Definition von $\boldsymbol{W}_k^T$ ausgenutzt werden, womit

$$(\boldsymbol{v}_h, \boldsymbol{m}^\perp q_{k-1})_{L^2(T)} = (\Pi_k^{\nabla,T} \boldsymbol{v}_h, \boldsymbol{m}^\perp q_{k-1})_{L^2(T)}$$

folgt, was ebenfalls berechenbar ist. Falls $q_{k-1} \in \mathbb{P}_{k-3}(T)$, dann lässt sich das letzte Integral einfach mit $D_{\boldsymbol{V}}^{\mathcal{M}_1}$ berechnen.

Damit lässt sich jetzt die erweiterte rechte Seite $\boldsymbol{f}_h^e$ definieren.

Definition 6.4 (Erweiterte Diskretisierung der rechten Seite)**.** Sei $\boldsymbol{f}_{h|T}^e := P_k^T \boldsymbol{f}$ die stückweise L^2-Projektion auf Polynome vom Grad kleiner oder gleich k. Die erweiterte Diskretisierung der rechten Seite ist für alle $\boldsymbol{v}_h \in \boldsymbol{W}_k$ durch

$$(\boldsymbol{f}_h^e, \boldsymbol{v}_h)_{L^2(\Omega)}$$

gegeben.

Das erweiterte diskrete Problem ist dann analog zu Definition 3.20 durch Austausch von $\boldsymbol{f}_h$ und $\boldsymbol{f}_h^e$ gegeben:

Definition 6.5 (Erweitertes diskretes Problem)**.** Das erweiterte diskrete Problem liest sich wie folgt: Finde $(\boldsymbol{u}_h, p_h) \in \boldsymbol{W}_k \times Q_k$, so dass

$$a_h(\boldsymbol{u}_h, \boldsymbol{v}_h) + b_h(\boldsymbol{v}_h, p_h) = (\boldsymbol{f}_h^e, \boldsymbol{v}_h)_{L^2(\Omega)} \qquad \text{für alle } \boldsymbol{v}_h \in \boldsymbol{W}_k, \tag{6.2a}$$

$$b(\boldsymbol{u}_h, q_h) = 0 \qquad \text{für alle } q_h \in Q_k \tag{6.2b}$$

gilt.

Die eindeutige Lösung und die Theorie der a-priori Fehlerabschätzung bis einschließlich Satz 3.32 übertragen sich analog, da Proposition 3.22 beliebige rechte Seiten $F \in \boldsymbol{V}^*$ zulässt. Allerdings wird die Abschätzung für den Konsistenzfehler der rechten Seite besser.

Lemma 6.6 (Diskretisierungsfehler der erweiterten rechten Seite)**.** *Sei $\mathcal{F}_h^e$ die Dualnorm von $\boldsymbol{f}_h^e - \boldsymbol{f}$ in $\boldsymbol{V}^*$, das heißt*

$$\mathcal{F}_h^e := \sup_{\boldsymbol{v}_h \in \boldsymbol{V} \setminus \{0\}} \frac{|(\boldsymbol{f}_h^e - \boldsymbol{f}, \boldsymbol{v}_h)_{L^2(\Omega)}|}{|\boldsymbol{v}_h|_{H^1(\Omega)}}.$$

Ist $\boldsymbol{f} \in H^{k+1}(\Omega)$, dann gilt

$$\mathcal{F}_h^e \leq |h_{\mathcal{T}}^{k+2} \boldsymbol{f}|_{H^{k+1}(\Omega)}.$$

Beweis. Der Beweis ist analog dem Beweis von Lemma 3.30 wobei $\boldsymbol{f}_h$ durch $\boldsymbol{f}_h^e$ inklusive der vollen Projektion sowie deren Abschätzung ausgetauscht werden muss. □

Damit ergibt sind in Analogie zu Korollar 3.33 die folgende a-priori Fehlerabschätzung.

Korollar 6.7. *Ist $\boldsymbol{f} \in [H^{k+1}(\Omega)]^2$ und ist $(\boldsymbol{u}_h, p_h) \in \boldsymbol{W}_k \times Q_k$ die Lösung des erweiterten diskreten Problems (6.2a)-(6.2b), dann gelten die folgenden Aussagen.*

(i) Wenn $\boldsymbol{u} \in [H^{k+1}(\Omega)]^2$, dann gilt

$$\|\boldsymbol{u} - \boldsymbol{u}_h\|_{H^1(\Omega)} \lesssim |h_{\mathcal{T}}^k \boldsymbol{u}|_{H^{k+1}(\Omega)} + \frac{1}{\nu} |h_{\mathcal{T}}^{k+2} \boldsymbol{f}|_{H^{k+1}(\Omega)}.$$

(ii) Sind $(\boldsymbol{u}, p) \in [H^{k+1}(\Omega)]^2 \times H^k(\Omega)$, folgt

$$\|p - p_h\|_{L^2(\Omega)} \lesssim |h_{\mathcal{T}}^{k+2} \boldsymbol{f}|_{H^{k+1}(\Omega)} + \nu |h_{\mathcal{T}}^k \boldsymbol{u}|_{H^{k+1}(\Omega)} + |h_{\mathcal{T}}^k p|_{H^k(\Omega)}.$$

Die Abschätzung für den Geschwindigkeitsfehler ist damit robuster im Vergleich zur gewöhnlichen Virtuellen-Elemente-Methode für kompliziertere rechte Seiten $\boldsymbol{f}$. Im Gegensatz zur herkömmlichen Diskretisierung der rechten Seite, bei der die rechte Seite ein Polynom vom Grade maximal $k-2$ sein darf, um die Geschwindigkeit exakt approximieren zu können, werden jetzt auch Polynome vom Grad k zugelassen.

Bemerkung 6.8. Der Fehlerschätzer aus Abschnitt 5.1.2 überträgt sich analog, wenn $\boldsymbol{f}_h$ durch $\boldsymbol{f}_h^e$ ausgetauscht wird.

6.1.2. Implementierung der erweiterten Virtuellen-Elemente-Räume

Ein großer Vorteil an der Implementierung der erweiterten Virtuellen-Elemente-Räume ist, dass sich kaum etwas ändert, da die gleichen Freiheitsgrade und die gleiche Projektion $\Pi_k^{\nabla,T}$ benutzt werden können. Das heißt insbesondere, dass es bei der herkömmlichen Virtuellen-Elemente-Methode nicht klar ist, ob die berechnete Lösung ein Element aus $\boldsymbol{V}_k$ oder aus $\boldsymbol{W}_k$ ist, da sie die gleichen Freiheitsgrade teilen. Das Einzige, was in der Implementierung geändert werden muss, ist die rechte Seite, bei der die Projektion P_{k-2} gegen P_k ausgetauscht werden muss. Dafür muss Gleichung (6.1) für alle Basisfunktionen φ_j, $j = 1, 2, \ldots, \texttt{nDof}^T$ und alle $\boldsymbol{m}_n$, $n = 1, 2, \ldots, 2\pi_k$ berechnet werden. Die Koeffizienten für alle $n = 1, 2, \ldots, 2\pi_{k-2}$ sind bereits aus der P_{k-2} Projektion bekannt. Es verbleiben also noch die Koeffizienten für die höheren Monome.

Um das erste Integral zu berechnen, wird die Divergenz für alle φ_j benötigt. Analog zu obiger Zerlegung wird $\mathrm{div}\varphi_j = \sum_{\ell=1}^{\pi_{k-1}} t_\ell^{(j)} m_\ell$ in Basisfunktionen zerlegt und dann

$$\sum_{\ell=1}^{\pi_{k-1}} t_\ell^{(j)} \int_T m_\ell m_n \,\mathrm{d}T = \int_T \mathrm{div}\varphi_j m_n \,\mathrm{d}T = b_h(\varphi_j, m_n)$$

für alle $n = 1, 2, \ldots, \pi_{k-1}$ gelöst, wobei die rechte Seite bereits aus der herkömmlichen Methode vorhanden ist. Mit der explizitien Darstellung der Divergenz reduziert sich das erste Integral in Gleichung (6.1) auf die Integration von Polynomen über Polygone.

Zur Berechnung des mittleren Integrals in Gleichung (6.1) müssen zuerst die Basisfunktionen explizit auf dem Rand rekonstruiert werden. Dafür kann ausgenutzt werden, dass die Basisfunktionen, die zu einer Kante $E \in \mathcal{E}(T)$ gehören, entlang dieser Kante den Basisfunktionen aus herkömmlichen P_k-Lagrange-Finite-Elemente-Methoden entsprechen. Für die Integration muss anschließend eine Integrationsformel vom Grad $2k + 1$ angewendet werden.

Da $\Pi_k^{\nabla,T}\varphi_j$ explizit vorliegt, kann das letzte Integral in Gleichung (6.1) für alle Polynome vom Grad mindestens $k - 1$ und maximal $k - 3$ einfach als Integration von Polynomen über Polygone berechnet werden.

6.2. Herstellung der Druckrobustheit mittels eines Rekonstruktionsoperators

Wie in der Einleitung des Kapitels erwähnt, ist das Problem der herkömmlichen und auch der erweiterten Virtuellen-Elemente-Methode, dass zwar die Testfunktionen $\boldsymbol{v}_h$ exakt divergenzfrei sind und damit L^2-orthogonal auf Gradientenfelder stehen, aber die Projektion $P_{k-2}\boldsymbol{v}_h$ bzw. $P_k\boldsymbol{v}_h$ ist es im Allgemeinen nicht. Die Projektion zerstört die gewünschte Orthogonalität und macht die Methode damit nicht druckrobust.

Im Kontext der Finiten-Elemente-Methoden können nicht druckrobuste Methoden mit Hilfe von geeigneten Rekonstruktionsoperatoren druckrobust gemacht werden, in dem genau die gewünschte L^2-Orthogonalität wieder hergestellt wird. Dies wurde das erste mal in [Lin14] eingeführt und als ein Variationsverbrechen (engl. “variational crime“) bezeichnet.

Diese auf Dreiecken bekannte Methode soll im Folgenden auf die Virtuelle-Elemente-Methode angewendet werden. Da Dreiecke nur besondere Polygone sind, wird zuerst die Rekonstruktion auf Dreiecken für die Virtuelle-Elemente-Methode vorgestellt und danach auf beliebige der Formregularität entsprechenden Polygone verallgemeinert. Abschließend folgen erneut Bemerkungen zur Implementierung.

6.2.1. Druckrobuste Rekonstruktion für die Virtuelle-Elemente-Methode

Für die Rekonstruktion wird ein Operator benötigt, der eine divergenzfreie virtuelle Funktion auf ein divergenzfreies Polynom abbildet. Dafür kann beispielsweise auf Dreiecken die standardmäßige Interpolation in Raviart-Thomas Finite-Elemente-Räume der Ordnung $k-1$ benutzt werden.

Zuerst werden daher Dreiecke behandelt, mit deren Hilfe die Rekonstruktion auch auf Polygonen funktioniert. Sei also jetzt $T \in \mathcal{T}$ ein Dreieck.

Definition 6.9 (Raviart-Thomas Finite-Elemente-Raum)**.** Der Raviart-Thomas Finite-Elemente-Raum der Ordnung $s \in \mathbb{N}$ auf einem Dreieck $T \in \mathcal{T}_h$ ist durch

$$\mathrm{RT}_s(T) := \left\{ \boldsymbol{q} \in [\mathbb{P}_{s+1}(T)]^2 \,:\, q(\boldsymbol{x}) = \begin{pmatrix} a \\ b \end{pmatrix} + c\boldsymbol{x} \text{ für } a,b,c \in \mathbb{P}_s(T) \right\}$$

gegeben.

Die standardmäßige lokale Raviart-Thomas Interpolation ist dann wie folgt definiert [BBF13].

Definition 6.10 (Raviart-Thomas Interpolation auf Dreiecken)**.** Für ein $\boldsymbol{v}_h \in \boldsymbol{V}_k^T$ ist die Raviart-Thomas Interpolation $\Pi_{\mathrm{RT}}^T \boldsymbol{v}_h \in \mathrm{RT}_{k-1}(T) \subset \mathbb{P}_k(T)$ durch

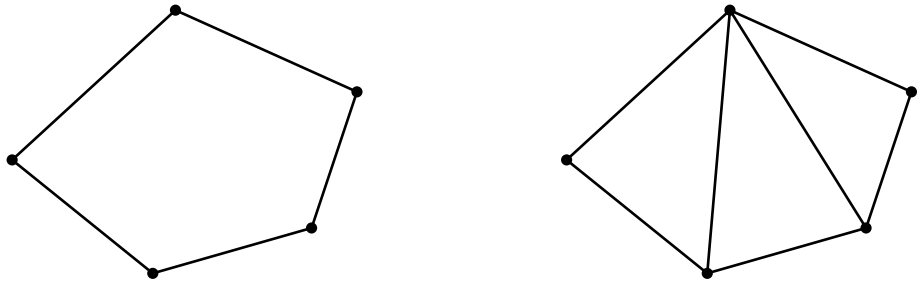
$$\int_E (\boldsymbol{v}_h - \Pi_{\mathrm{RT}}^T \boldsymbol{v}_h) \cdot \boldsymbol{n}_E \, q_{k-1} \,\mathrm{d}E = 0 \quad \text{für alle } q_{k-1} \in \mathbb{P}_{k-1}(E), \text{ für alle } E \in \mathcal{E}(T), \tag{6.3a}$$

$$\int_T (\boldsymbol{v}_h - \Pi_{\mathrm{RT}}^T \boldsymbol{v}_h) \cdot \boldsymbol{q}_{k-2} \,\mathrm{d}T = 0 \quad \text{für alle } \boldsymbol{q}_{k-2} \in [\mathbb{P}_{k-2}(T)]^2 \tag{6.3b}$$

bestimmt.

Wie üblich ist zu prüfen, ob die Interpolation mit Hilfe der Freiheitsgrade berechnet werden kann. Dies ist aber in der Tat der Fall. Die Gleichung (6.3a) kann mit Hilfe der Randfreiheitsgrade berechnet werden, denn sowohl $\boldsymbol{v}_h$ als auch die Interpolation sind auf $E \in \mathcal{E}(T)$ Polynome vom Grad maximal k, so dass der Integrand maximal ein Polynom vom Grad $2k-1$ ist. Damit reichen die Randfreiheitsgrade für die Berechnung aus. Für Gleichung (6.3b) wird abermals die Zerlegung von $\boldsymbol{q}_{k-2}$ in einen Gradientenanteil ∇q_{k-1} für ein $\boldsymbol{q}_{k-1} \in \mathbb{P}_{k-1}(T)$ und einen dazu orthogonalen Anteil $\boldsymbol{m}^\perp q_{k-3}$ für ein $q_{k-3} \in \mathbb{P}_{k-3}(T)$ ausgenutzt. Damit folgt unter Zuhilfenahme partieller Integration

$$\begin{aligned}\int_T \Pi_{\mathrm{RT}}^T \boldsymbol{v}_h \cdot \boldsymbol{q}_{k-2} \,\mathrm{d}T &= \int_T \boldsymbol{v}_h \cdot \boldsymbol{q}_{k-2} \,\mathrm{d}T = \int_T \boldsymbol{v}_h \cdot (\nabla q_{k-1} + \boldsymbol{m}^\perp q_{k-3}) \,\mathrm{d}T \\ &= -\int_T \mathrm{div}\boldsymbol{v}_h \, q_{k-1} \,\mathrm{d}T + \int_{\partial T} \boldsymbol{v}_h \cdot \boldsymbol{n} \, q_{k-1} \,\mathrm{d}s + \int_T \boldsymbol{v}_h \cdot \boldsymbol{m}^\perp q_{k-3} \,\mathrm{d}T.\end{aligned}$$



(a) Ursprüngliches Polygon (b) Mögliche Subtriangulierung

Abbildung 6.1.: Hier ist ein konvexes Polygon (a) und eine mögliche Subtriangulierung für das Polygon (b) gezeigt. Für konvexe Polygone kann einfach jeder Knoten mit einem beliebigen aber festen anderen Knoten verbunden werden.

Die Integrale lassen sich dann durch $D_{\boldsymbol{V}}^{\mathcal{M}_2}$, $D_{\boldsymbol{V}}^{\mathcal{N}}$ und $D_{\boldsymbol{V}}^{\mathcal{E}}$ bzw. $D_{\boldsymbol{V}}^{\mathcal{M}_1}$ berechnen.

Um später eine Abschätzung für den Konsistenzfehler der rechten Seite zu erhalten wird das folgende Lemma benötigt.

Lemma 6.11 (Eigenschaften von I_{RT})**.** *Für alle $T \in \mathcal{T}$ und für alle $\boldsymbol{v}_h \in \boldsymbol{V}_k^T$ gilt, dass*

$$(i) \qquad \mathrm{div}\Pi_{\mathrm{RT}}^T \boldsymbol{v}_h = \mathrm{div}\boldsymbol{v}_h,$$

$$(ii) \qquad \|\Pi_{\mathrm{RT}}^T \boldsymbol{v}_h - \boldsymbol{v}_h\|_{L^2(T} \lesssim |h_{\mathcal{T}} \boldsymbol{v}_h|_{H^1(T)}.$$

Beweis. Da $\mathrm{div}\Pi_{\mathrm{RT}}^T \boldsymbol{v}_h - \mathrm{div}\boldsymbol{v}_h \in \mathbb{P}_{k-1}(T)$, reicht es zum Beweis von (i)

$$\int_T \mathrm{div}(\Pi_{\mathrm{RT}}^T \boldsymbol{v}_h - \boldsymbol{v}_h)\, q_{k-1}\, \mathrm{d}T = 0 \qquad \text{für alle } q_{k-1} \in \mathbb{P}_{k-1}(T)$$

zu zeigen. Dies folgt aus der Definition von Π_{RT}^T. Partielle Integration und Ausnutzen von Gleichungen (6.3a) und (6.3b) zeigen, dass

$$\begin{aligned}\int_T \mathrm{div}(\Pi_{\mathrm{RT}}^T \boldsymbol{v}_h - \boldsymbol{v}_h)\, q_{k-1}\, \mathrm{d}T &= -\int_T (\Pi_{\mathrm{RT}}^T \boldsymbol{v}_h - \boldsymbol{v}_h) \cdot \nabla q_{k-1}\, \mathrm{d}T \\ &\quad + \int_{\partial T} (\Pi_{\mathrm{RT}}^T \boldsymbol{v}_h - \boldsymbol{v}_h) \cdot \boldsymbol{n}\, q_{k-1}\, \mathrm{d}s \\ &= 0,\end{aligned}$$

da $\nabla q_{k-1} \in [\mathbb{P}_{k-2}(T)]^2$ und $q_{k-1} \in \mathbb{P}_{k-1}(E)$ für alle $E \in \mathcal{E}(T)$. Dies beendet den Beweis von (i).

Der zweite Teil (ii) folgt aus dem Bramble-Hilbert Lemma auf dem Referenzelement zusammen mit einer Transformation auf allgemeine Elemente und kann beispielsweise in [Bar16] auf Seite 294 nachgelesen werden. □

Dies sind alle notwendigen Informationen, um später eine druckrobuste Version für Dreiecke zu beweisen. Zusätzlich ist einer der großen Vorteile von Virtuellen-Elemente-Methoden, dass sie auch auf allgemeineren Polygonen funktionieren, weshalb obiges Verfahren auf Polygone erweitert werden muss.

Sei also jetzt $T \in \mathcal{T}_h$ wieder ein Polygon. Um obige Ideen auch auf einem Polygon anwenden zu können, wird jedes Polygon in Dreiecke zerlegt. Dies ist in jedem Fall möglich, da jedes Polygon sternförmig in Bezug auf einen Kreis ist. Dadurch kann einfach jeder Knoten aus $\mathcal{N}(T)$ mit einem beliebigen aber festen Punkt aus diesem Kreis verbunden werden. Für ein konvexes Polygon können auch alle Knoten mit einem bestimmten anderen verbunden werden wie es auch in Abbildung 6.1 gezeigt ist.

Auf jeder Subtriangulierung $\mathcal{T}(T)$ kann jetzt eine passende stückweise Raviart-Thomas Interpolation durchgeführt werden.

Definition 6.12 (Stückweise Raviart-Thomas Interpolation auf Polygonen)**.** Sei $\mathcal{T}(T)$ eine Subtriangulierung eines Polygons $T \in \mathcal{T}$. Für alle $\boldsymbol{v}_h \in \boldsymbol{V}_k^T$ ist die lokale lokale Raviart-Thomas Interpolation $I_{\mathrm{RT}}^T \boldsymbol{v}_h$ durch

$$I_{\mathrm{RT}}^T \boldsymbol{v}_h := \operatorname*{argmin}_{\boldsymbol{w}_h \in \widehat{RT}_k(T,\boldsymbol{v}_h)} \|\Pi_k^{\nabla,T} \boldsymbol{v}_h - \boldsymbol{w}_h\|_{L^2(T)}$$

definiert, wobei

$$\begin{aligned}\widehat{RT}_k(T,\boldsymbol{v}_h) := \Big\{ & \boldsymbol{w}_h \in H(\mathrm{div},T;\mathbb{R}^2) \,:\, \boldsymbol{w}_{h|T'} \in \mathrm{RT}_{k-1}(T') \text{ für alle } T' \in \mathcal{T}(T) \\ & \int_T (\boldsymbol{w}_h - \boldsymbol{v}_h)\cdot \boldsymbol{q}_{k-2}\,\mathrm{d}T = 0 \text{ für alle } \boldsymbol{q}_{k-2} \in [\mathbb{P}_{k-2}(T)]^2 \subset [\mathbb{P}_{k-2}(\mathcal{T}(T))]^2 \text{ und} \\ & \int_E (\boldsymbol{w}_h - \boldsymbol{v}_h)\cdot \boldsymbol{n}_E\, q_{k-1}\,\mathrm{d}E = 0 \text{ für alle } q_{k-1} \in \mathbb{P}_{k-1}(E) \text{ und alle } E \in \mathcal{E}(T) \Big\}\end{aligned}$$

der Ansatzraum der Raviart-Thomas Funktionen auf der Subtriangulierung ist.

Diese stückweise Raviart-Thomas Interpolation für Polygone ist eindeutig definiert, da der Ansatzraum $\widehat{RT}_k(T,\boldsymbol{v}_h)$ mindestens die Funktion enthält, die auf jedem Dreieck der Subtriangulierung der Raviart-Thomas Interpolation von $\boldsymbol{v}_h$ entspricht. Dies ist gerade die Funktion, die approximiert werden soll, da sie nicht explizit bekannt ist. Darüber hinaus besteht der Raum $\widehat{RT}_k(T,\boldsymbol{v}_h)$ für Dreiecke aus genau einer Funktion, die der vorherigen Interpolation entspricht.

In Analogie zu Lemma 6.11 können jetzt folgende Eigenschaften für die Interpolation bewiesen werden.

Lemma 6.13 (Eigenschaften von I_{RT}^T)**.** *Für alle $T \in \mathcal{T}$ und für alle $\boldsymbol{v}_h \in \boldsymbol{V}_k^T$ gilt*

$$\begin{aligned} &(i) && \mathrm{div} I_{\mathrm{RT}}^T \boldsymbol{v}_h = \mathrm{div}\boldsymbol{v}_h, \\ &(ii) && \|I_{\mathrm{RT}}^T \boldsymbol{v}_h - \boldsymbol{v}_h\|_{L^2(T)} \lesssim |h_{\mathcal{T}} \boldsymbol{v}_h|_{H^1(T}. \end{aligned}$$

Beweis. Der Beweis der Behauptung (i) kann wegen der Definition von $\widehat{RT}_k(T,\boldsymbol{v}_h)$ eins zu eins aus dem Beweis von Lemma 6.11 übernommen werden.

Für (ii) sei $\Pi_{\mathrm{RT}}\boldsymbol{v}_h$ die obige stückweise Raviart-Thomas Interpolation für Dreiecke auf der Subtriangulierung $\mathcal{T}(T)$. Dann gilt wegen der Definition von $\widehat{RT}_k(T,\boldsymbol{v}_h)$, dass $\Pi_{\mathrm{RT}}\boldsymbol{v}_h \in \widehat{RT}_k(T,\boldsymbol{v}_h)$. Aus der Definition von I_{RT}^T folgt, dass

$$(I_{\mathrm{RT}}^T \boldsymbol{v}_h - \Pi_k^{\nabla,T}\boldsymbol{v}_h, \boldsymbol{w}_h)_{L^2(T)} = 0 \qquad \text{für alle } \boldsymbol{w}_h \in \widehat{RT}_k(T,\boldsymbol{v}_h),$$

weshalb mit $\boldsymbol{w}_h = I_{\mathrm{RT}}^T \boldsymbol{v}_h - \Pi_{\mathrm{RT}} \boldsymbol{v}_h$

$$\begin{aligned}\|I_{\mathrm{RT}}^T \boldsymbol{v}_h - \Pi_{\mathrm{RT}} \boldsymbol{v}_h\|_{L^2(T)}^2 &= (I_{\mathrm{RT}}^T \boldsymbol{v}_h - \Pi_{\mathrm{RT}} \boldsymbol{v}_h, I_{\mathrm{RT}}^T \boldsymbol{v}_h - \Pi_{\mathrm{RT}} \boldsymbol{v}_h)_{L^2(T)} \\ &= (\Pi_k^{\nabla,T} \boldsymbol{v}_h - \Pi_{\mathrm{RT}} \boldsymbol{v}_h, I_{\mathrm{RT}}^T \boldsymbol{v}_h - \Pi_{\mathrm{RT}} \boldsymbol{v}_h)_{L^2(T)} \\ &\leq \|\Pi_k^{\nabla,T} \boldsymbol{v}_h - \Pi_{\mathrm{RT}} \boldsymbol{v}_h\|_{L^2(T)} \|I_{\mathrm{RT}}^T \boldsymbol{v}_h - \Pi_{\mathrm{RT}} \boldsymbol{v}_h\|_{L^2(T)}\end{aligned}$$

folgt.

Nach der Definition der Projektion $\Pi_k^{\nabla,T}$ hat $\boldsymbol{v}_h - \Pi_k^{\nabla,T} \boldsymbol{v}_h$ das Integralmittel Null, wodurch mit Hilfe der Poincaré-Ungleichung auch die Approximationseigenschaft erster Ordnung $\|\Pi_k^{\nabla,T} \boldsymbol{v}_h - \boldsymbol{v}_h\|_{L^2(T)} \lesssim h_T |\boldsymbol{v}_h|_{H^1(T)}$ folgt.

Die Approximationseigenschaft von $\Pi_k^{\nabla,T}$ und Π_{RT} auf jedem Dreieck $T' \in \mathcal{T}(T)$ zeigen dann mit Hilfe einer Dreiecksungleichung, dass

$$\|I_{\mathrm{RT}}^T \boldsymbol{v}_h - \Pi_{\mathrm{RT}} \boldsymbol{v}_h\|_{L^2(T)} \leq \|\Pi_k^{\nabla,T} \boldsymbol{v}_h - \Pi_{\mathrm{RT}} \boldsymbol{v}_h\|_{L^2(T)} \lesssim h_T |\boldsymbol{v}_h|_{H^1(T)}.$$

Aus der Dreiecksungleichung folgt, dass

$$\|I_{\mathrm{RT}}^T \boldsymbol{v}_h - \boldsymbol{v}_h\|_{L^2(T)} \leq \|I_{\mathrm{RT}}^T \boldsymbol{v}_h - \Pi_{\mathrm{RT}} \boldsymbol{v}_h\|_{L^2(T)} + \|\Pi_{\mathrm{RT}} \boldsymbol{v}_h - \boldsymbol{v}_h\|_{L^2(T)},$$

woraus mit obiger Überlegung und der Approximationseigenschaft von Π_{RT} die Behauptung (ii) folgt. □

Mit Hilfe der Interpolation wird nun eine druckrobuste rechte Seite konstruiert.

Definition 6.14 (Druckrobuste Diskretisierung der rechten Seite)**.** Die druckrobuste Diskretisierung $\boldsymbol{f}_h^+$ ist durch

$$(\boldsymbol{f}_h^+, \boldsymbol{v}_h)_{L^2(\Omega)} := (\boldsymbol{f}, I_{\mathrm{RT}} \boldsymbol{v}_h)_{L^2(\Omega)}$$

definiert, wobei $I_{\mathrm{RT}} \boldsymbol{v}_{h|T} := I_{\mathrm{RT}}^T \boldsymbol{v}_h$ für alle $T \in \mathcal{T}_h$.

Damit kann jetzt das druckrobuste Problem definiert werden.

Definition 6.15 (Druckrobustes diskretes Problem)**.** Das druckrobuste diskrete Problem liest sich wie folgt: Finde $(\boldsymbol{u}_h, p_h) \in \boldsymbol{V}_k \times Q_k$, so dass

$$a_h(\boldsymbol{u}_h, \boldsymbol{v}_h) + b_h(\boldsymbol{v}_h, p_h) = (\boldsymbol{f}_h^+, \boldsymbol{v}_h)_{L^2(\Omega)} \qquad \text{für alle } \boldsymbol{v}_h \in \boldsymbol{V}_k, \tag{6.4a}$$

$$b(\boldsymbol{u}_h, q_h) = 0 \qquad \text{für alle } q_h \in Q_k \tag{6.4b}$$

gelten.

Natürlich muss zunächst geklärt werden, ob das obige Problem eine eindeutige Lösung hat. Glücklicherweise überträgt sich wie auch für das erweiterte Problem die Lösungstheorie und auch die Konvergenztheorie ganz analog zum gewöhnlichen diskreten Problem, da Proposition 3.22 allgemeine rechte Seiten $F \in \boldsymbol{V}^*$ zulässt.

Allerdings ändert sich das Korollar 3.33, da der Konsistenzfehler der rechten Seite besser wird. Für den Konsistenzfehler wird die Helmholtz-Hodge-Zerlegung der rechten Seite $\boldsymbol{f} \in [L^2(\Omega)]^2$ benötigt.

Lemma 6.16 (Helmholtz-Hodge-Zerlegung)**.** *Wenn Ω ein zusammenhängendes Gebiet ist, dann lässt sich jedes $\boldsymbol{f} \in [L^2(\Omega)]^2$ eindeutig in ein Vektorfeld $\boldsymbol{f}_0 \in H(\mathrm{div}, \Omega; \mathbb{R}^2)$ und ein $\phi \in H^1(\Omega)/\mathbb{R}$ zerlegen, so dass*

$$
\begin{aligned}
&(i) && \boldsymbol{f} = \boldsymbol{f}_0 + \nabla\phi, \\
&(ii) && \mathrm{div}\boldsymbol{f}_0 = 0, \\
&(iii) && (\boldsymbol{f}_0, \nabla\psi)_{L^2(\Omega)} = 0 \text{ für alle } \psi \in H^1(\Omega)
\end{aligned}
$$

gelten.

Beweis. Der Beweis findet sich beispielsweise in [JLM+17] und basiert auf der eindeutigen Lösung des folgenden Neumannproblems: Finde $\phi \in H^1(\Omega)/\mathbb{R})$, so dass

$$
(\nabla\phi, \nabla\psi)_{L^2(\Omega)} = (\boldsymbol{f}, \nabla\psi)_{L^2(\Omega)} \qquad \text{für alle } \psi \in H^1(\Omega)/\mathbb{R}
$$

gilt. Die Wahl von $\boldsymbol{f}_0 := \boldsymbol{f} - \nabla\phi$ führt zum gewünschten Ergebnis. □

Definition 6.17 (Helmholtz-Projektion)**.** Die Abbildung $\mathcal{H} : [L^2(\Omega)]^2 \to H(\mathrm{div}, \Omega; \mathbb{R}^2)$ für ein $\boldsymbol{f} \in [L^2(\Omega)]^2$ definiert durch $\mathcal{H}(\boldsymbol{f}) := \boldsymbol{f}_0$ für $\boldsymbol{f}_0$ aus vorherigem Lemma wird auch Helmholtz-Hodge- oder kurz Helmholtz-Projektion genannt.

Damit kann jetzt der Diskretisierungsfehler der druckrobusten rechten Seite bewiesen werden.

Lemma 6.18 (Diskretisierungsfehler der druckrobusten rechten Seite)**.** *Sei $\mathcal{F}_h^+$ die Dualnorm von $\boldsymbol{f}_h^+ - \boldsymbol{f}$ in $\boldsymbol{V}^*$, das heißt*

$$
\mathcal{F}_h^+ := \sup_{\boldsymbol{v}_h \in \boldsymbol{V} \setminus \{0\}} \frac{|(\boldsymbol{f}, I_{\mathrm{RT}}\boldsymbol{v}_h - \boldsymbol{v}_h)_{L^2(\Omega)}|}{|\boldsymbol{v}_h|_{H^1(\Omega)}}.
$$

(i) Sind $\boldsymbol{f}, \mathcal{H}(\boldsymbol{f}) \in H^{k-1}(\Omega)$, dann gilt

$$
\mathcal{F}_h^+ \leq |h_{\mathcal{T}}^k \, \mathcal{H}(\boldsymbol{f})|_{H^{k-1}(\Omega)},
$$

wobei $\mathcal{H}(\boldsymbol{f})$ die Helmholtz-Projektion von $\boldsymbol{f}$ ist.

(ii) Ist $\boldsymbol{u} \in H^{k+1}(\Omega)$ und $p \in H^1(\Omega)$, dann ist

$$
\mathcal{F}_h^+ \leq \nu |h_{\mathcal{T}}^k \, \Delta\boldsymbol{u}|_{H^{k-1}(\Omega)}.
$$

Beweis. Aus (i) von Lemma 6.13 folgt, dass $\mathrm{div}(I_{\mathrm{RT}}\boldsymbol{v}_h - \boldsymbol{v}_h) = 0$ und $I_{\mathrm{RT}}\boldsymbol{v}_h - \boldsymbol{v}_h \in H_0^1(\Omega)$ für alle $\boldsymbol{v}_h \in \boldsymbol{V}_k$ gilt. Deshalb folgt mit der Helmholtz-Hodge-Zerlegung aus Lemma 6.16 und partieller Integration, dass

$$
\begin{aligned}
\int_\Omega \boldsymbol{f} \cdot (I_{\mathrm{RT}}\boldsymbol{v}_h - \boldsymbol{v}_h) \, \mathrm{d}\Omega &= \int_\Omega (\mathcal{H}(\boldsymbol{f}) + \nabla\phi) \cdot (I_{\mathrm{RT}}\boldsymbol{v}_h - \boldsymbol{v}_h) \, \mathrm{d}\Omega \\
&= \int_\Omega \mathcal{H}(\boldsymbol{f}) \cdot (I_{\mathrm{RT}}\boldsymbol{v}_h - \boldsymbol{v}_h) \, \mathrm{d}\Omega.
\end{aligned}
$$

Wegen der Definition von I_{RT} folgt, dass

$$\begin{aligned}\int_\Omega \mathcal{H}(\boldsymbol{f}) \cdot (I_{\mathrm{RT}}\boldsymbol{v}_h - \boldsymbol{v}_h)\,\mathrm{d}\Omega &= \int_\Omega (\mathcal{H}(\boldsymbol{f}) - P_{k-2}\mathcal{H}(\boldsymbol{f})) \cdot (I_{\mathrm{RT}}\boldsymbol{v}_h - \boldsymbol{v}_h)\,\mathrm{d}\Omega \\ &\leq \|\mathcal{H}(\boldsymbol{f}) - P_{k-2}\mathcal{H}(\boldsymbol{f})\|_{L^2(\Omega)} \|I_{\mathrm{RT}}\boldsymbol{v}_h - \boldsymbol{v}_h\|_{L^2(\Omega)} \\ &\lesssim |h_{\mathcal{T}}^k\,\mathcal{H}(\boldsymbol{f})|_{H^{k-1}(\Omega)} |\boldsymbol{v}_h|_{H^1(\Omega)},\end{aligned}$$

wobei im letzten Schritt (ii) aus Lemma 6.13 und die Approximationstheorie auf sternförmigen Gebieten ausgenutzt wurde. Einsetzen in das Supremum beendet den Beweis von (i).

Für (ii) kann ganz analog argumentiert werden: Unter der Ausnutzung der starken Form von $\boldsymbol{f} = -\nu\Delta\boldsymbol{u} + \nabla p$ folgt zuerst, dass

$$\begin{aligned}\left|\int_\Omega \boldsymbol{f} \cdot (I_{\mathrm{RT}}\boldsymbol{v}_h - \boldsymbol{v}_h)\,\mathrm{d}\Omega\right| &= \int_\Omega (\nu\Delta\boldsymbol{u} - \nabla p) \cdot (I_{\mathrm{RT}}\boldsymbol{v}_h - \boldsymbol{v}_h)\,\mathrm{d}\Omega \\ &= \nu \int_\Omega \Delta\boldsymbol{u} \cdot (I_{\mathrm{RT}}\boldsymbol{v}_h - \boldsymbol{v}_h)\,\mathrm{d}\Omega.\end{aligned}$$

Der Rest folgt analog zu Fall (i), wobei $P_{k-2}\Delta\boldsymbol{u}$ anstelle von $P_{k-2}\mathcal{H}(\boldsymbol{f})$ eingefügt wird. □

Damit ergibt sich nun folgendes Korollar für die a-priori Fehlerabschätzung der druckrobusten Methode.

Korollar 6.19. *Ist $\boldsymbol{f} \in [H^{k-1}(\Omega)]^2$ und ist $(\boldsymbol{u}_h, p_h) \in \boldsymbol{V}_k \times Q_k$ die Lösung der druckrobusten Diskretisierung (6.4a)-(6.4b), dann gelten die folgenden Aussagen.*

(i) Wenn $\boldsymbol{u} \in [H^{k+1}(\Omega)]^2$, dann gilt

$$\|\boldsymbol{u} - \boldsymbol{u}_h\|_{H^1(\Omega)} \lesssim |h_{\mathcal{T}}^k \boldsymbol{u}|_{H^{k+1}(\Omega)} + |h_{\mathcal{T}}^k \Delta\boldsymbol{u}|_{H^{k-1}(\Omega)}.$$

(ii) Sind $(\boldsymbol{u}, p) \in [H^{k+1}(\Omega)]^2 \times H^k(\Omega)$, folgt

$$\|p - p_h\|_{L^2(\Omega)} \lesssim \nu\,|h_{\mathcal{T}}^k\,\Delta\boldsymbol{u}|_{H^{k-1}(\Omega)} + \nu\,|h_{\mathcal{T}}^k\,\boldsymbol{u}|_{H^{k+1}(\Omega)} + |h_{\mathcal{T}}^k\,p|_{H^k(\Omega)}.$$

Beweis. Die Behauptung folgt genau wie in dem Korollar 3.33, wobei (ii) aus Lemma 6.18 für den Diskretisierungsfehler benutzt wird. □

Bemerkung 6.20. Alternativ kann natürlich auch unter genügend glatter Regularität mit (i) aus Lemma 6.18 für den Diskretisierungsfehler

$$\begin{aligned}&(i) \qquad \|\boldsymbol{u} - \boldsymbol{u}_h\|_{H^1(\Omega)} \lesssim |h_{\mathcal{T}}^k \boldsymbol{u}|_{H^{k+1}(\Omega)} + \frac{1}{\nu}|h_{\mathcal{T}}^k \mathcal{H}(f)|_{H^{k-1}(\Omega)}, \\ &(ii) \qquad \|p - p_h\|_{L^2(\Omega)} \lesssim |h_{\mathcal{T}}^k\,\mathcal{H}\boldsymbol{f}|_{H^{k-1}(\Omega)} + \nu\,|h_{\mathcal{T}}^k\,\boldsymbol{u}|_{H^{k+1}(\Omega)} + |h_{\mathcal{T}}^k\,p|_{H^k(\Omega)}.\end{aligned}$$

gezeigt werden.

Die obige Abschätzung macht die Methode tatsächlich druckrobust. Jegliche Änderung von $\boldsymbol{f}$ zu $\boldsymbol{f} + \nabla q$ für alle $q \in H^1(\Omega)$ beeinflussen die diskrete Geschwindigkeitsabschätzung nicht, da $\mathcal{H}(\nabla q) = 0$ für alle $q \in H^1(\Omega)$.

6.2.2. Implementierung der druckrobusten Virtuellen-Elemente-Methode

Allgemein wird sowohl für Polygone als auch für Dreiecke für die Implementierung der druckrobusten Methode auf jedem Element die Raviart-Thomas Interpolation $I_{\mathrm{RT}}\varphi_j$ für alle Basisfunktionen $\varphi_j \in \boldsymbol{V}_k^T$, $j = 1, 2, \ldots, \mathtt{nDof}^T$ benötigt. Für allgemeine Polygone muss auf jedem Polygon das Minimierungsproblem aus Definition 6.12 gelöst werden. Da dies jedoch den Rahmen dieser Arbeit gesprengt hätte, wird nur die Implementierung für Dreiecke besprochen.

Zunächst muss eine Basis $\psi_\ell \in \mathrm{RT}_{k-1}(T)$, $\ell = 1, 2, \ldots, k(k+2) =: \pi_{\mathrm{RT}_{k-1}}$ gewählt werden. Beispielsweise ist in [Erv12] eine berechenbare Basis für Raviart-Thomas Räume beliebiger Ordnung angegeben. Dann kann die Projektion $\Pi_{\mathrm{RT}}^T\varphi_j$ mit Hilfe der Definition von Π_{RT}^T ausgerechnet werden. Analog zu vorherigen Projektionen wird dafür die Projektion jeder Basisfunktion

$$\Pi_{\mathrm{RT}}^T\varphi_j = \sum_{\ell=1}^{\pi_{\mathrm{RT}_{k-1}}} c_\ell^{(j)}\psi_\ell \tag{6.5}$$

zerlegt und damit die Gleichungen (6.3a) bzw. (6.3b) für alle skalierten Monome m_n, $n = 1, 2, \ldots, \pi_{k-1}$ bzw. $\boldsymbol{m}_n$, $n = 1, 2, \ldots, 2\pi_{k-2}$ gelöst. Im letzten Fall muss dabei wieder auf die Zerlegung der skalierten Monome in einen Gradienten- und einen dazu orthogonalen Anteil zurückgegriffen werden.

Da in [Erv12] die Basisfunktionen auf dem Referenzelement angegeben werden und dann mit Hilfe einer Piola-Transformation auf allgemeine Elemente übertragen werden, soll hier eine andere Möglichkeit vorgestellt werden. Diese beschränkt sich auf den Fall $k = 2$, lässt sich jedoch auch für höhere Polynomordnungen verallgemeinern. Für ein Dreieck $T := \mathrm{conv}\{V_1, V_2, V_3\}$ sind Basisfunktionen für den Raum $\mathrm{RT}_0(T)$ durch

$$\psi_j(\boldsymbol{v}) := \frac{1}{2|T|}(\boldsymbol{x} - V_{j-1}) \qquad \text{für alle } j = 1, 2, 3$$

definiert, wobei $V_0 := V_3$, und erfüllen für alle $j, k = 1, 2, 3$

$$D_k(\psi_j) := \int_{E_k} \psi_j \cdot \boldsymbol{n}_{E_k}\,\mathrm{d}E_k = \delta_{j,k} = \begin{cases} 1, & \text{falls } j = k \\ 0, & \text{sonst} \end{cases}$$

mit dem Kronecker-Delta $\delta_{j,k}$ [Bar16].

Da die Dimension von $\mathrm{RT}_1(T)$ acht ist, werden fünf weitere Basisfunktionen benötigt. Dafür werden die baryzentrischen Koordinaten $\lambda_1, \lambda_2, \lambda_3$ mit der Eigenschaft

$$\boldsymbol{x} = \lambda_1(\boldsymbol{x})V_1 + \lambda_2(\boldsymbol{x})V_2 + \lambda_3(\boldsymbol{x})V_3 \qquad \text{für } \boldsymbol{x} \in T$$

benötigt. Die fünf weiteren Basisfunktionen sind dann durch

$$\begin{aligned} \psi_{3+j} &:= 12(\lambda_j - 0{,}5)\psi_j && \text{für } j = 1, 2, 3 \\ \psi_{6+j} &:= 12\lambda_{j-1}\psi_j && \text{für } j = 1, 2 \end{aligned}$$

definiert, wobei $\lambda_0 := \lambda_3$. Da $\mathrm{RT}_1(T) = \mathbb{P}_1(T) \times \mathrm{RT}_0(T)$ ist, gilt, dass $\psi_j \in \mathrm{RT}_1(T)$ für alle $j = 1, 2, \ldots, 8$. Die zugehörigen Momente sind

$$D_{3+k}(\psi) := \int_{E_k} \psi \cdot \boldsymbol{n}_{E_k}(\lambda_k - 0,5)\,\mathrm{d}E_k \qquad \text{für } k = 1,2,3$$
$$D_{7,8}(\psi) := \int_T \psi\,\mathrm{d}T,$$

womit das folgende Lemma bewiesen werden kann.

Lemma 6.21. *Es gilt für alle $k = 1, 2, \ldots 6$ und alle $j = 1, 2, \ldots, 8$, dass*

$$D_k(\psi_j) = \delta_{j,k}.$$

Beweis. Für die ersten $j, k = 1, 2, 3$ ist das Resultat einfach, da $\psi_k \cdot \boldsymbol{n}_{E_k} = |E_k|^{-1}$ entlang der Kante E_k und entlang $\partial T \setminus E_k$ gleich Null ist. Daher wird jetzt $k = 1, 2, 3$ und $j = 4, 5, 6$ betrachtet. Dann gilt, dass

$$D_k(\psi_j) = 12 \int_{E_k} (\lambda_{j-3} - 0,5)\psi_{j-3} \cdot \boldsymbol{n}_{E_k}\,\mathrm{d}E_k = 12\frac{\delta_{j-3,k}}{|E_k|}\int_{E_k} \lambda_{j-3} - 0,5\,\mathrm{d}E_k = 0,$$

da entweder $\delta_{j-3,k} = 0$ oder $\int_{E_k} \lambda_{j-3} - 0,5\,\mathrm{d}E_k = 0$.

Für $j, k = 4, 5, 6$ ergibt sich

$$\begin{aligned} D_k(\psi_j) &= 12 \int_{E_k} (\lambda_{j-3} - 0,5)\psi_{j-3} \cdot \boldsymbol{n}_{E_k}(\lambda_{j-3} - 0,5)\,\mathrm{d}T = 12\frac{\delta_{j,k}}{|E_k|}\int_{E_k} (\lambda_{j-3} - 0,5)^2\,\mathrm{d}E_k \\ &= \delta_{j,k}. \end{aligned}$$

Abschließend kann bemerkt werden, dass $\lambda_{j-1}\psi_j = 0$ entlang ∂T, weshalb

$$D_k(\psi_j) = 0 \qquad \text{für alle } k = 1, 2, \ldots, 6, j = 7, 8,$$

was den Beweis beendet. □

Bemerkung 6.22. Die hier angegebene Basis ist keine vollständige Orthonomalbasis, denn es gilt mit $\boldsymbol{x}_{E_j} := (V_j + V_{j+1})/2$ dem Mittelpunkt von E_j, dass

$$\begin{aligned} D_{7,8}(\psi_j) &= (\boldsymbol{x}_T - V_{j-1})/2 && \text{für alle } j = 1,2,3, \\ D_{7,8}(\psi_j) &= V_{j-1} - \boldsymbol{x}_{E_{j-1}} && \text{für alle } j = 4,5,6, \\ D_{7,8}(\psi_j) &= \boldsymbol{x}_{E_j} - V_j && \text{für alle } j = 7,8, \end{aligned}$$

wie durch einfaches Nachrechnen gezeigt werden kann.

Die Bemerkung zusammen mit dem vorherigen Lemma zeigen die lineare Unabhängigkeit der Basisfunktionen, da die Matrix $A := (A_{j,k})$, $j = 1, 2, \ldots, 8, k \in \{1, 2, \ldots, 6, \{7, 8\}\}$ mit

$$A_{j,k} := (D_k(\psi_j)) \qquad \text{für } j = 1, 2, \ldots, 8, k \in \{1, 2, \ldots, 6, \{7, 8\}\}$$

regulär ist.

Damit ergeben sich die Koeffizienten $c_\ell^{(j)}$ aus Gleichung (6.5) für alle $\ell = 1, 2, \ldots, 6$ durch

$$c_\ell^{(j)} = D_\ell(\varphi_j)$$

und die verbleibenden beiden Koefizienten durch

$$D_{7,8}(c_7^{(j)}\psi_7 + c_8^{(j)}\psi_8) = D_{7,8}(\varphi_j) - \sum_{\ell=1}^{6} c_\ell^{(j)} D_{7,8}(\psi_j),$$

was ein lineares 2×2 Gleichungssystem in den Unbekannten $c_7^{(j)}$ und $c_8^{(j)}$ ist.

6.3. Numerische Experimente zur Druckrobustheit

In diesem Abschnitt werden abschließend die normale, die erweiterte sowie die druckrobuste Methode numerisch auf Dreiecksgittern verglichen werden. Dafür wird zunächste erneut das Problem 2 betrachtet.

6.3.1. Problem 2: Hydrostatisches Problem mit kleiner Viskosität

Es sei wie in Abschnitt 4.3.2 wieder das Gebiet $\Omega = (0,1)^2$. Die rechte Seite und die Randbedingungen werden so gewählt, dass die exakte Lösung durch

$$\boldsymbol{u}(x,y) := \begin{pmatrix} 0 \\ 0 \end{pmatrix} \in \boldsymbol{V}_h \qquad \text{und} \qquad p(x,y) = \sin(2\pi x)\cos(2\pi y) \notin Q_h$$

gegeben ist. Dies führt dazu, dass die rechte Seite nur der Gradient von p ist.

In Abschnitt 4.3.2 ist festgestellt worden, dass die herkömmliche Virtuelle-Elemente-Methode nicht druckrobust sein kann, da der Fehler für die Geschwindigkeit mit dem Inversen der Viskosität ν skaliert (s. Abbildung 4.4). Insbesondere $\nu = 10^{-4}$ sorgt für eine schlechte Approximation der Geschwindigkeit. Deshalb wird hier wieder die Viskosität auf 10^{-4} gesetzt, um die erweiterte Virtuelle-Elemente-Methode und die druckrobuste Form mit der herkömmlichen Methode zu vergleichen. Da die druckrobuste Form nur für Dreiecke implementiert ist, wird die Lösung auf einer Serie von uniform mit AFEM verfeinerten Dreiecksgittern ausgehend vom Dreiecksgitter aus Abbildung 6.2 berechnet. Es wird ein unstrukturiertes Gitter genommen, da mit dieser Methode auf strukturierten Gittern bessere Konvergenzraten beobachtet werden können als auf unstrukturierten Gittern (s. Abschnitt 7.2.2). Der Geschwindigkeitsfehler $\|\mathrm{D}(\boldsymbol{u} - \Pi_k^\nabla \boldsymbol{u}_h)\|_{L^2(\Omega)}$ ist für alle drei Methoden in Abbildung 6.3 gegen die Anzahl der Freiheitsgrade `nDof` dargestellt.

Direkt zu sehen ist, dass sowohl die gewöhnliche als auch die erweiterte Virtuelle-Elemente-Methode mit einem großen Fehler starten und dann mit ihrer jeweiligen optimalen Rate von 2 beziehungsweise 4 in Bezug auf den mittleren Durchmesser konvergieren. Die erweiterte Virtuelle-Elemente-Methode ist somit ebenfalls nicht druckrobust, da sie nicht mit ν skalieren dürfte, konvergiert dann aber viel schneller als die gewöhnliche

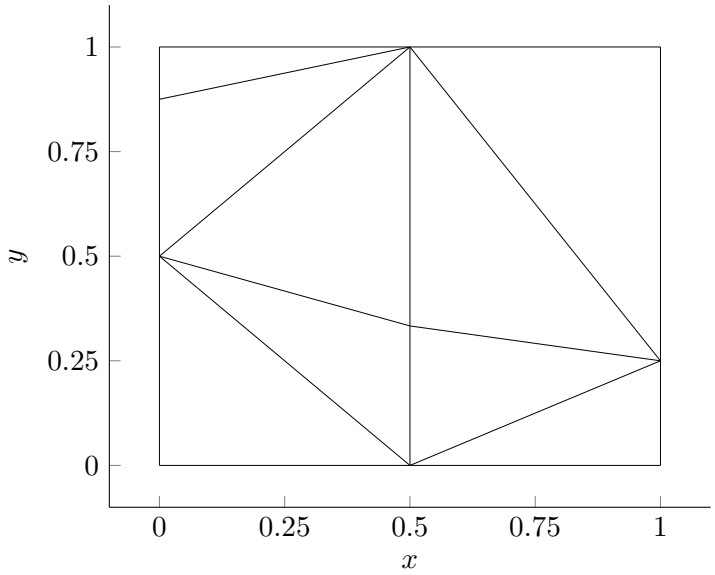


Abbildung 6.2.: Gezeigt ist das Ausgangsgitter für Problem 2 zum Vergleich von herkömmlicher, erweiterter und druckrobuster Virtueller-Elemente-Methode. Es wird ein unstrukturiertes Gitter gewählt, um Superkonvergenz auf strukturierten Gittern zu vermeiden.

Virtuelle-Elemente-Methode und stellt also eine Verbesserung für kleine Viskositäten ν dar. Damit bestätigen sie die jeweiligen Aussagen aus Korollar 3.33 bzw. Korollar 6.7.

Die druckrobuste Methode hingegen beginnt mit einem Fehler von etwa $7 \cdot 10^{-7}$, fällt dann sehr schnell bis etwa $1 \cdot 10^{-12}$ und steigt dann langsam wieder bis zu $2 \cdot 10^{-11}$ an. Der anfänglich vorhandene Fehler liegt möglicherweise daran, dass der Exaxtkheitsgrad für die numerische Integration von $\boldsymbol{f} = \nabla p$ mit 15 zu niedrig gewählt ist. Damit hat der Integrationsfehler bei groben Netzen einen zu großen Einfluss, der zur Erhöhung des Fehlers führt. Der Anstieg im weiteren Verlauf lässt sich vermutlich durch nicht exaktes Lösen des linearen Gleichungssystems erklären. Insgesamt bestätigen diese Ergebnise bis auf numerische Ungenauigkeiten die theoretischen Ergebnise aus Korollar 6.19, dass eine druckrobuste Virtuelle-Elemente-Methode konstruiert ist.

6.3.2. Problem 3: Ein Vortex-Problem

Abschließend wird hier noch ein Vortex-Problem betrachtet, bei dem im Gegensatz zu vorher weder die Geschwindigkeit $\boldsymbol{u}$ noch der Druck p in den Ansatzräumen vorhanden sind. Dafür wird erneut das Gebiet $\Omega = (0,1)^2$ gewählt und die rechte Seite so, dass die

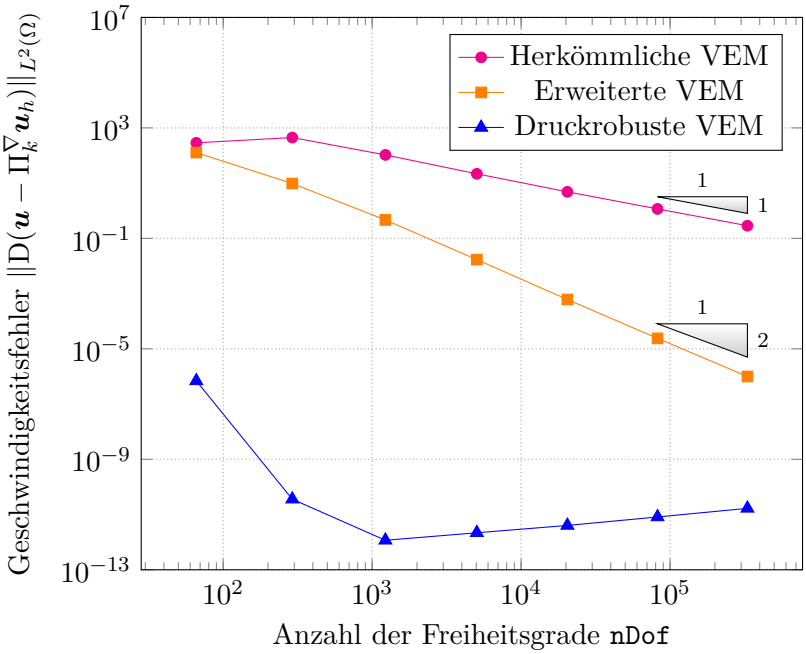


Abbildung 6.3.: Dargestellt ist die Konvergenzhistorie für Problem 2 mit fester Viskosität $\nu = 10^{-4}$ für verschiedene Methoden. Sowohl die herkömmliche als auch die erweiterte Virtuelle-Elemente-Methode (VEM) konvergieren mit ihrer jeweiligen optimalen Rate aus Korollar 3.33 bzw. Korollar 6.7, skalieren aber mit dem Inversen der Viskosität. Die druckrobuste Methode ist bis auf numerische Ungenauigkeiten exakt und damit tatsächlich druckrobust wie es das Korollar 6.19 vorhersagt.

exakte Lösung

$$\boldsymbol{u}(x,y) = \begin{pmatrix} -\partial/\partial y \\ \partial/\partial x \end{pmatrix} \left(x^2(x-1)^2 y^2 (y-1)^2 \right) \notin \boldsymbol{V}_h \text{ und}$$
$$p(x,y) = \sin(2\pi x)\cos(2\pi y) \notin Q_h$$

ist. Die Viskosität wird erneut auf $\nu = 10^{-4}$ gesetzt und die Lösung auf einer Serie von rotverfeinerten Gittern ausgehend vom Gitter aus Abbildung 6.2 berechnet. Die Konvergenzhistorien für die drei Methoden sind in Abbildung 6.4 aufgetragen.

Die herkömmliche Virtuelle-Elemente-Methode startet wie bereits zuvor mit einem großen Fehler und konvergiert dann mit einer optimalen Ordnung von 2 in Bezug auf den mittleren Durchmesser h_m. Einen anfänglich großen Fehler hat auch die erweiterte Virtuelle-Elemente-Methode, allerdings konvergiert sie dann schneller als die herkömmliche Methode. Solange der Diskretisierungsfehler der rechten Seite dominant ist, konvergiert die Methode mit leicht größerer Ordnung als 4. Danach, wenn der Diskretisierungsfehler nicht mehr dominant ist, konvergiert die Methode mit Ordnung 2, was

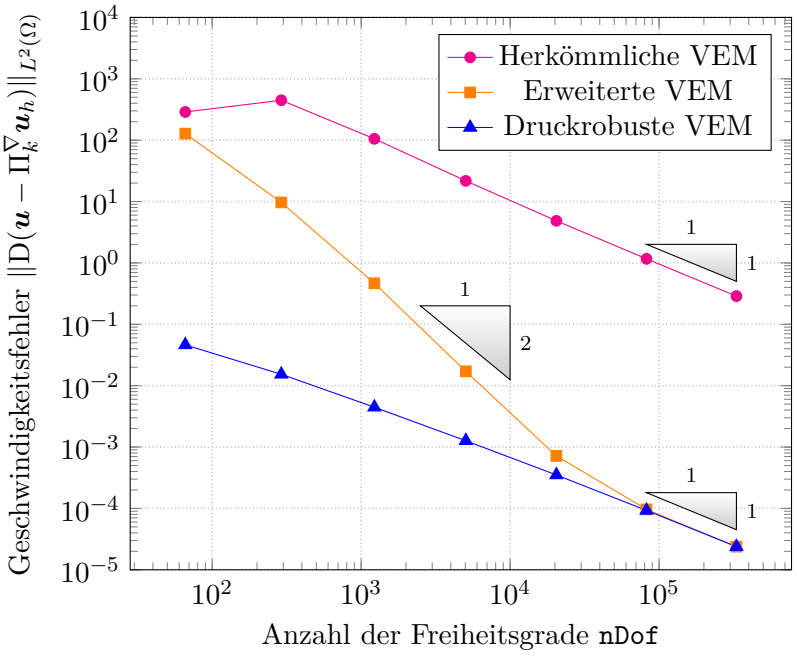


Abbildung 6.4.: Abgebildet ist die Konvergenzhistorie für Problem 3 mit Viskosität $\nu = 10^{-4}$ für verschiedene Methoden. Die herkömmliche und die erweiterte Virtuelle-Elemente-Methode sind nicht druckrobust und konvergieren jeweils mit der optimalen Ordnung. Solange der Diskretisierungsfehler der rechten Seite dominant ist, konvergiert die erweiterte Virtuelle-Elemente-Methode mit einer Ordnung von etwas besser als 4 in Bezug auf den mittleren Durchmesser, danach mit Ordnung 2. Die druckrobuste Virtuelle-Elemente-Methode startet mit viel kleinerem Fehler und konvergiert permanent mit der optimalen Ordnung von 2.

erneut die Korrektheit des Korollars 6.7 bestätigt. Die druckrobuste Methode startet mit einem Fehler, der vier Größenordnungen kleiner als die der gewöhnlichen bzw. erweiterten Methode ist. Sie konvergiert anschließend mit einer optimalen Rate von 2 in Bezug auf den mittleren Durchmesser, was im Einklang mit Korollar 6.19 steht.

Auch wenn die erweiterte Virtuelle-Elemente-Methode eine beträchtliche Verbesserung gegenüber der herkömmlichen Methode ist, sind beide nicht druckrobust. Ausschließlich die druckrobuste Methode ist wirklich druckrobust und damit resistent gegen Änderungen der rechten Seite, die im Kontinuierlichen nur den Druck beeinflussen.

7. Vergleich von Finiter-Elemente- und Virtueller-Elemente-Methode

Diese Arbeit hat sich bisher ausschließlich mit einer Virtuellen-Elemente-Methode und deren Erweiterungen beschäftigt. In diesem Kapitel soll ein direkter Vergleich zu einer Finiten-Elemente-Methode gezogen werden.

Dafür wird eine gängige inf-sup stabile Finite-Elemente-Methode gewählt, die die gleiche Ordnung wie die Virtuelle-Elemente-Methode hat. Betrachtet wird die Methode $\boldsymbol{\mathcal{P}}_2^b/\mathcal{P}_1$ aus [CR73], die zur Approximation der Geschwindigkeit stetige Polynome vom Grad kleiner oder gleich zwei angereichert mit sogenannten Zellblasen (engl. "cell bubbles") und für den Druck stückweise stetige Polynome der Ordnung eins benutzt.

Im Folgenden werden zuerst einige Vor- und Nachteile der Virtuellen-Elemente-Methoden allgemein im Vergleich zu Finiten-Elemente-Methoden aufgelistet. Anschließend folgen numerische Vergleiche zwischen der Virtuellen-Elemente-Methode und der oben genannten Finiten-Elemente-Methode.

7.1. Vor- und Nachteile Virtueller-Elemente-Methoden

Die Finiten-Elemente-Methoden charakterisieren sich dadurch, dass sie ausschließlich Polynome als Ansatzfunktionen zulassen. Das macht die Implementierung im Vergleich zu den Virtuellen-Elemente-Methoden leichter, da die Konstruktion der benötigten Matrizen durch die explizite Form der Basisfunktionen vergleichsweise leicht ist. Die Virtuellen-Elemente-Methoden benutzen hingegen neben diesen Polynomen auch gewisse andere nicht-Polynome. Dies macht das Aufstellen der benötigten Matrizen im Verhältnis zu den Finiten-Elemente-Methoden schwieriger, da jegliche Berechnungen auf die Auswertung der implizit definierten Basisfunktionen zurückgeführt werden muss. Stehen die benötigten Matrizen dann zur Verfügung, ist die Implementierung nicht weiter schwierig.

Die implizite Definition der Basisfunktionen führt dazu, dass bestimmte gewünschte Eigenschaften schon in der Konstruktion der Virtuellen-Elemente-Methoden bedacht werden können. So ist die in dieser Arbeit betrachtete Virtuelle-Elemente-Methode so konstruiert, dass sie exakt divergenzfrei ist.

Die Virtuellen-Elemente-Methoden benötigen zur Definition der diskreten Bilinearformen einen Stabilisierungsterm, der zur Behandlung der nicht-Polynome gebraucht wird und daher bei Finiten-Elemente-Methoden entfällt. Er ist für die Konvergenz der Methode notwendig [dBC$^+$13], kann aber verschieden gewählt und sogar mit verschiedenen positiven reelen Zahlen skaliert werden [dLR17]. Abhängig von der gewählten Stabilisierung können sogar schwächere Voraussetzungen an das Gitter formuliert werden [dLR17],

was in ausgewählten Fällen von Vorteil sein könnte.

Durch die implizit definierten Basisfunktionen sind auch die errechneten Lösungen der Virtuellen-Elemente-Methoden schlussendlich immer noch virtuell, das heißt sie stehen im Gegensatz zu den Lösungen Finiter-Elemente-Methoden nicht explizit zur Verfügung. Das scheint auf den ersten Blick unüblich, da zur Visualisierung und zur weiteren Verarbeitung eine explizite Form benötigt wird. Dem kann aber mit Hilfe der Projektion Π_k^∇ in Polynomräume begegnet werden, so dass anschließend eine explizite Form vorhanden ist.

Die Projektion führt jedoch dazu, dass einige Eigenschaften der Lösung nicht erhalten bleiben. Beispielsweise ist die Projektion $\Pi_k^\nabla \boldsymbol{v}_h$ für ein Element $\boldsymbol{v}_h \in \boldsymbol{V}_k$ im Gegensatz zum Element selbst nicht mehr global H^1-konform und nicht mehr global divergenzfrei. Werden also für die weitere Verarbeitung global H^1-konforme und global divergenzfreie Lösungen benötigt, ist eine Finite-Elemente-Methode zu wählen, die das erfüllt. Beispiele dafür sind das Scott-Vogelius-Element [SV85, GS17] oder druckrobuste Formen Finiter-Elemente-Methoden [JLM$^+$17]. Die zuletzt genannten Methoden benötigen allerdings ein Nachbearbeiten der Lösung, das auch für Virtuelle-Elemente-Methoden möglich ist.

Der Hauptvorteil der Virtuellen-Elemente-Methoden gegenüber den Finiten-Elemente-Methoden liegt vor allen Dingen in der Flexibilität des zugrunde liegenden Gitters. Im Gegensatz zu Finiten-Elemente-Methoden, wo hauptsächlich Dreiecke und Rechtecke (2D) beziehungsweise Tetraeder und Hexaeder (3D) genutzt werden, lassen die Virtuellen-Elemente-Methoden viel allgemeinere Polygone zu. Dadurch kann das Gitterdesign flexibler gestaltet werden. Das macht das Gitter robuster gegen Verformungen und kann bei Problemen helfen, die auf komplizierten, vielleicht bereits aus Polygonen bestehenden, Gebieten zu lösen sind. Dazu zählt unter anderem die Oberflächenoptimierung, bei der Dreiecksgitter zu numerischen Instabilitäten führen können [Lan07].

Diese Gitterflexibilität macht darüber hinaus adaptive Verfeinerungen auf der einen Seite leichter als bei Finiten-Elemente-Methoden, andererseits auch schwieriger. Insbesondere hängende Knoten können bei der Verfeinerung von polygonalen Gittern bei den Virtuellen-Elemente-Methoden zugelassen werden, so dass der Verfeinerungsalgorithmus hängende Knoten nicht vermeiden muss. Allerdings muss dafür der Verfeinerungsalgorithmus zumindest für die hier gewählte Stabilisierung sicherstellen, dass die kleinste Kante im Verhältnis zur größten Kante nicht zu klein wird, um die Formregularität zu erhalten [dLR17]. Tut er das nicht, kann die optimale Konvergenzrate im Allgemeinen nicht erwartet werden (s. Abschnitt 5.2.1). Andere Stabilisierungen könnten daher auch für anisotrope Gitterverfeinerungen von Interesse sein.

Abschließend lässt die Gitterflexibilität auch eine Kopplung an andere Methoden wie beispielsweise Finite-Volumen-Methoden zu. Die dort auftretenden Kontrollvolumen müssen keine Dreiecke oder Vierecke sein und lassen ebenfalls allgemeinere Formen zu [JK18]. Dadurch, dass die Virtuellen-Elemente-Methoden polygonale Gitter zulassen, muss das Gitter nicht weiter verändert werden, um die Lösung für weitere Berechnungen mit Finiten-Volumen-Methoden nutzen zu können.

7.2. Vergleichende numerische Experimente

Hier wird ein numerischer Vergleich auf Dreiecksgittern zwischen einer Finiten-Elemente-Methode der gleichen Ordnung und der Virtuellen-Elemente-Methode, die in dieser Arbeit betrachtet wird, durchgeführt.

Dafür wird die in der Einleitung angesprochene Finite-Elemente-Methode $\boldsymbol{\mathcal{P}}_2^b/\mathcal{P}_1$ aus [CR73] genutzt. Dabei ist der Ansatzraum für die Geschwindigkeit

$$\boldsymbol{\mathcal{P}}_2^b := \left[\left(\mathbb{P}_2(\mathcal{T}_h) \cap C(\overline{\Omega}) \cap H_0^1(\Omega)\right) \cup \mathcal{B}(\mathcal{T}_h)\right]^2,$$

wobei

$$\mathcal{B}(\mathcal{T}_h) := \left\{ b \,:b_{|T} \in \operatorname{span}\{\lambda_1\lambda_2\lambda_3\} \text{ für alle } T \in \mathcal{T}_h \right\}$$

mit den baryzentrischen Koordinaten $\lambda_1, \lambda_2, \lambda_3$ für jedes Dreieck $T \in \mathcal{T}_h$ der Raum der Zellblasen ist. Der Ansatzraum besteht somit aus stetigen Polynomen vom Grad maximal zwei angereichert mit Zellblasen. Der Ansatzraum für den Druck ist durch

$$\mathcal{P}_1(\mathcal{T}_h) := \mathbb{P}_1(\mathcal{T}_h) \cap L_0^2(\Omega)$$

gegeben, das heißt stückweise stetige Polynomen der Ordnung eins mit Integralmittel Null.

Diese Methode wird gewählt, da sie auf Dreiecksgittern die gleiche Anzahl an Freiheitsgraden pro Element hat und die Virtuelle-Elemente-Methode vom Aufbau her als divergenzfreies Analogon zu dieser Finiten-Elemente-Methode gesehen werden kann. Die bei der Virtuellen-Elemente-Methode auftretenden Freiheitsgrade $D_{\boldsymbol{V}}^{\mathcal{M}_1}$ und $D_{\boldsymbol{V}}^{\mathcal{M}_2}$ definieren Funktionen, die als virtuelle Zellblasen angesehen werden können.

Der Fehler zwischen der kontinuierlichen Geschwindigkeit und der diskreten Geschwindigkeit mit der $\boldsymbol{\mathcal{P}}_2^b/\mathcal{P}_1$-Finiten-Elemente-Methode lässt sich durch

$$\begin{aligned}\|\boldsymbol{u} - \boldsymbol{u}_h\|_{H^1(\Omega)} &\lesssim \inf_{\boldsymbol{v}_h \in \boldsymbol{\mathcal{P}}_2^b} |\boldsymbol{u} - \boldsymbol{v}_h|_{H^1(\Omega)} + \inf_{q_h \in \mathcal{P}_1(\mathcal{T}_h)} \frac{1}{\nu}\|p - q_h\|_{L^2(\Omega)} \\ &\lesssim |h_{\mathcal{T}}^k \boldsymbol{u}|_{H^{k+1}(\Omega)} + \frac{1}{\nu}|h_{\mathcal{T}}^k p|_{H^k(\Omega)}\end{aligned}$$

abschätzen [CR73, JLM+17]. Die Methode hat daher die gleiche Konvergenzrate wie die hier betrachtete Virtuelle-Elemente-Methode.

Für die numerischen Experimente mit der $\boldsymbol{\mathcal{P}}_2^b/\mathcal{P}_1$-Methode wird freundlicherweise ein von Dr. C. Merdon implementierter AFEM-kompatibler Code verwendet.

7.2.1. Problem 3: Das Vortex-Problem mit moderater Viskosität

Als erstes Problem wird das Vortex-Problem aus Abschnitt 6.3.2 mit moderater Viskosität $\nu = 1$ betrachtet.

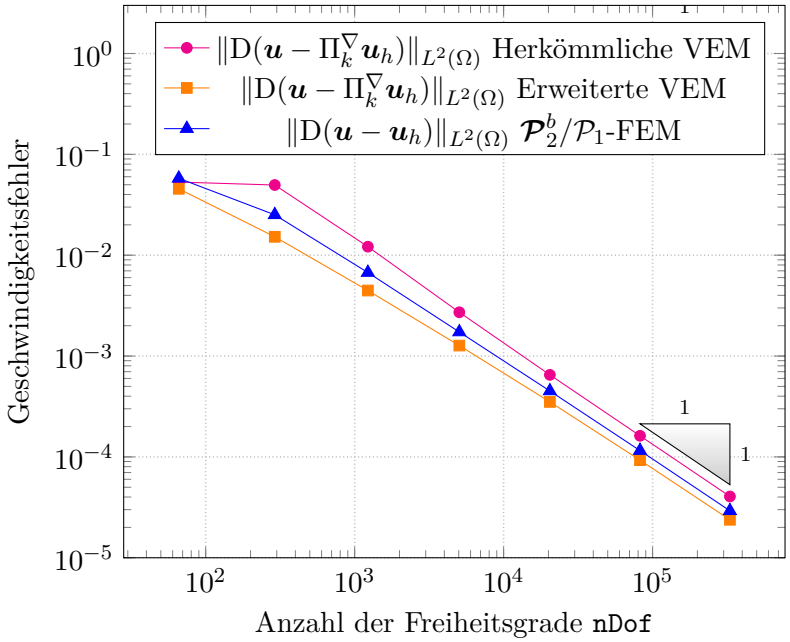


Abbildung 7.1.: Gezeigt ist die Konvergenzhistorie für Problem 3 mit Viskosität $\nu = 1$ für verschiedene Methoden. Sowohl die herkömmliche als auch die erweiterte Virtuelle-Elemente-Methode (VEM) und die $\boldsymbol{\mathcal{P}}_2^b/\mathcal{P}_1$-Finite-Elemente-Methode (FEM) konvergieren mit der erwarteten optimalen Ordnung. Die Fehler sind alle in der gleichen Größenordnung.

Das Gebiet $\Omega = (0,1)^2$ wird dabei in Dreiecke zerlegt, um die gleiche Anzahl an Freiheitsgraden pro Element zu garantieren. Die rechte Seite $\boldsymbol{f}$ wird so vorgegeben, dass die exakte Lösung

$$\boldsymbol{u}(x,y) = \begin{pmatrix} -\partial/\partial y \\ \partial/\partial x \end{pmatrix} \left(x^2(x-1)^2 y^2 (y-1)^2 \right) \notin \boldsymbol{V}_h \text{ und}$$
$$p(x,y) = \sin(2\pi x)\cos(2\pi y) \notin Q_h$$

ist.

Das Problem wird auf einer Serie uniform rotverfeinerter Gitter ausgehend vom Gitter aus Abbildung 6.2 für die gewöhnliche und die druckrobuste Virtuelle-Elemente-Methode sowie die $\boldsymbol{\mathcal{P}}_2^b/\mathcal{P}_1$-Methode gelöst. Die Konvergenzhistorie des Fehlers gegenüber der Anzahl der Freiheitsgrade ist in Abbildung 7.1 dargestellt.

Alle drei Methoden konvergieren mit der optimalen zu erwarteten Rate von 2 in Bezug auf den mittleren Durchmesser. Darüber hinaus sind die Fehler der drei Methoden in der gleichen Größenordnung.

Daher lässt sich schlussfolgern, dass für moderate Viskositäten ν jede der Methoden wählbar ist und keine besonders zu bevorzugen ist.

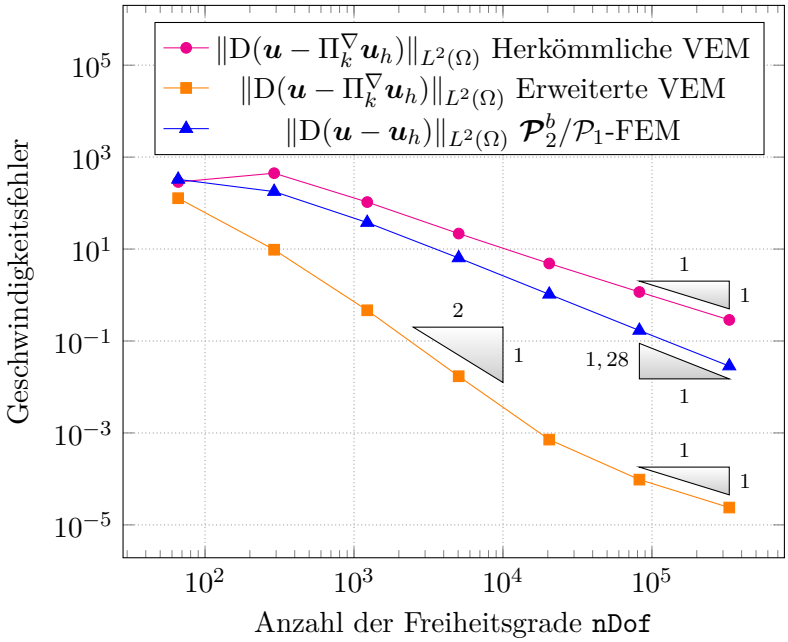


Abbildung 7.2.: Die Konvergenzhistorie für Problem 3 mit Viskosität $\nu = 10^{-4}$ mit uniform verfeinerten Gitter ausgehend von Abbildung 6.2 wird hier gezeigt. Die Fehler der herkömmlichen Virtuellen-Elemente-Methode und der $\boldsymbol{\mathcal{P}}_2^b/\mathcal{P}_1$-Methode sind anfangs in der gleichen Größenordnung. Die Finite-Elemente-Methode konvergiert dann aber etwas schneller als erwartet. Ein Grund dafür könnten die rechtwinkligen Dreiecke in der Ausgangstriangulierung sein.

7.2.2. Problem 3: Das Vortex-Problem mit kleiner Viskosität

Abschließend wird erneut das Vortex-Problem mit kleinerer Viskosität als vorher betrachtet.

Das Gebiet, die exakte Lösung und auch die Ausgangstriangulierung werden wie im vorherigen Abschnitt 7.2.1 gewählt. Im Gegensatz zu dem obigen Abschnitt wird allerdings die Viskosität auf $\nu = 10^{-4}$ gesetzt.

Der Geschwindigkeitsfehler für die herkömmliche und die druckrobuste Virtuelle-Elemente-Methode sowie die $\boldsymbol{\mathcal{P}}_2^b/\mathcal{P}_1$-Methode ist auf einer Serie uniform rotverfeinerter Gitter in Abbildung 7.2 gegen die Anzahl der Freiheitsgrade `nDof` dargestellt. Im Gegensatz zu vorherigem Experiment gibt es hier deutliche Unterschiede zwischen den Methoden.

Die herkömmliche Virtuelle-Elemente-Methode und die $\boldsymbol{\mathcal{P}}_2^b/\mathcal{P}_1$-Methode haben anfangs Geschwindigkeitsfehler von der gleichen großen Größenordnung 10^2. Im weiteren Verlauf konvergiert die $\boldsymbol{\mathcal{P}}_2^b/\mathcal{P}_1$-Methode dann sogar mit einer Konvergenzrate von etwa $2,56$ in Bezug auf den mittleren Durchmesser und damit etwas schneller als erwartet und als die herkömmliche Virtuelle-Elemente-Methode. Diese konvergiert nämlich mit

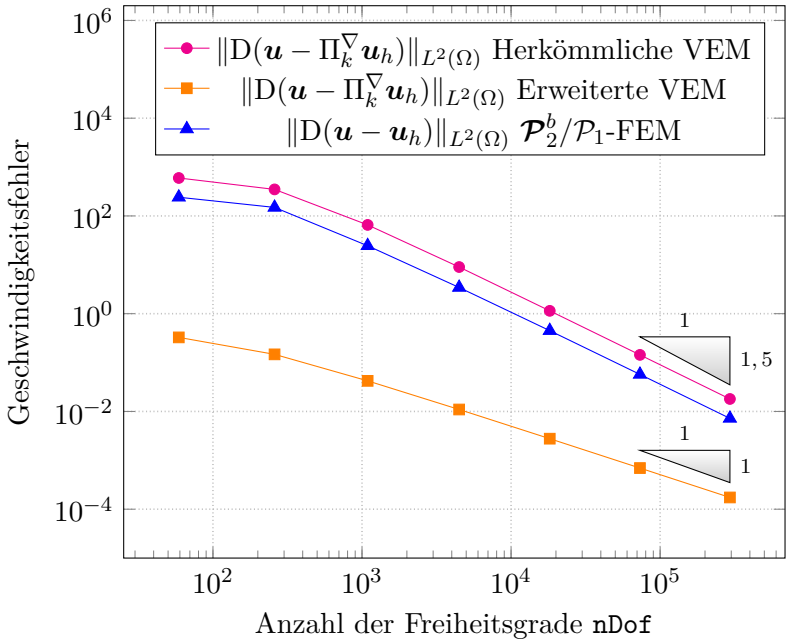


Abbildung 7.3.: Dargestellt ist die Konvergenzhistorie für Problem 3 mit Viskosität $\nu = 10^{-4}$ ausgehend von einem strukturierten Dreiecksgitter. Für die herkömmliche Virtuelle-Elemente-Methode und die Finite-Elemente-Methode treten Superkonvergenzeffekte auf. Die erweiterte Virtuelle-Elemente-Methode beginnt mit deutlich kleinerem Fehler als auf dem unstrukturierten Gitter und konvergiert dann mit optimaer Rate.

der optimalen Rate von 2 in Bezug auf den mittleren Durchmesser. Das hat zur Folge, dass der Geschwindigkeitsfehler auf dem feinsten Netz mit 10^{-1} eine Größenordnung größer ist als der der Finiten-Elemente-Methode mit einem Fehler der Größenordnung 10^{-2}. Dieser Effekt der Superkonvergenz für die Finite-Elemente-Methode kann an den teilweisen strukturierten Dreiecken im Gebiet liegen (s. [CB18] und die dortigen Referenzen für eine kurze Einführung).

Die erweiterte Virtuelle-Elemente-Methode startet ebenfalls mit einem großen Fehler, konvergiert dann aber mit einer Rate bezogen auf den mittleren Durchmesser, die sogar etwas besser als 4 ist. Dies liegt daran, dass anfangs der Fehler in der Diskretisierung der rechten Seite überwiegt. Auf den feineren Netzen, wo der Diskretisierungsfehler der rechten Seite nicht mehr dominant ist, verlangsamt sich dann die Konvergenzrate zu der optimalen Rate von 2 bezogen auf den mittleren Durchmesser. Damit ist die auf dem feinsten Netz etwa drei Größenordnungen kleiner als die Finite-Elemente-Methode.

Um die Superkonvergenz etwas genauer zu untersuchen, wird daher das gleiche Problem auf einer Serie uniform verfeinerter Gitter ausgehend vom einem einmal crossverfeinerten Einheitsquadrat erneut berechnet. Die Konvergenzhistorie ist in Abbil-

dung 7.3 dargestellt.

Deutlich zu erkennen ist, dass die herkömmliche Virtuelle-Elemente-Methode und die Finite-Elemente-Methode beide mit einer Rate von 3 in Bezug auf den mittleren Durchmesser konvergieren. Dies ist deutlich besser als erwartet und stützt die These, dass die vorher beobachtete Superkonvergenz durch die strukturierten Anteile im Gitter hervorgerufen werden. Bedingt durch das strukturierte Gitter startet die erweiterte Virtuelle-Elemente-Methode mit einem viel kleineren Fehler als mit dem unstrukturierten Gitter und konvergiert dann sofort mit der zu erwartenden Rate von 2.

Zusammenfassend lässt sich schlussfolgern, dass für kleine Viskositäten die erweiterte Virtuelle-Elemente-Methode die zu bevorzugende Methode ist, da sie am schnellsten konvergiert. Die Möglichkeit, eine erweiterte Methode mit verbesserter Konvergenzrate zu bilden und ohne die ursprüngliche Methode zu ändern, ist eine Besonderheit der Virtuellen-Elemente-Methoden im Vergleich zu den Finiten-Elemente-Methoden, was wie oben gesehen in bestimmten Fällen von Vorteil sein kann. Eine druckrobuste Form der Finiten-Elemente-Methode [JLM$^+$17] oder die druckrobuste Form der Virtuellen-Elemente-Methode aus Kapitel 6 würden allerdings noch bessere Ergebnisse ohne die vorasymptotische Phase liefern.

8. Ausblick

Das letzte Kapitel dieser Arbeit gibt einen Ausblick darüber, welche Probleme noch ungelöst sind und wie die Virtuelle-Elemente-Methode weiter verwendet werden könnte.

Insbesondere im Bereich der Adaptivität und der damit verknüpften Frage nach einer geeigneten Stabilisierung liegt noch Forschungspotential. Für die Adaptivität wird wie in Kapitel 5 bemerkt eine Verfeinerungsroutine benötigt, die die Formregularität des Gitters erhält. Damit die Verfeinerungsroutine von Dr. O. Sutton aus [Sut17] anwendbar ist, muss das Baryzentrum der Polygone innerhalb des Balls liegen, in dessen Bezug die Polygone sternförmig sind. Sonst würden die neuen Kanten außerhalb des Gebiets liegen. Um das volle Potential Virtueller-Elemente-Methoden in Bezug auf die Zerlegung des Gebiets nutzen zu können, werden daher für Polygone, auf die die obige Bedingung nicht zutrifft, kompliziertere Verfeinerungsalgorithmen benötigt, die die Formregularität erhalten.

Darüber hinaus kann der Einfluss der Stabilisierung noch über die Arbeit von [dLR17] hinausgehend untersucht werden. Insgesamt scheint die Stabilisierung für die Konvergenz benötigt zu werden [AAB+13], aber verschiedene Stabilisierungen führen zu unterschiedlich genauen Approximationen [dLR17]. Hier ist insbesondere interessant, ob mit dem anderen Stabilisierungsterm aus [dLR17] die in dieser Arbeit genutzte Verfeinerungsroutine zu besseren Konvergenzraten führt.

Eine weitere interessante Fragestellung ist die Anwendbarkeit der in dieser Arbeit präsentierten druckrobusten Form der Virtuellen-Elemente-Methode auf die eingangs erwähnten komplizierteren Navier-Stokes-Gleichungen. Die Navier-Stokes-Gleichungen suchen in variationeller Form auf einem Gebiet Ω während eines Zeitintervals $(0, T)$, $T < \infty$, eine Geschwindigkeit $\boldsymbol{u}$ und einen Druck p, so dass

$$\int_\Omega \frac{\partial}{\partial t}\boldsymbol{u}\,\boldsymbol{v}\,\mathrm{d}\Omega + \int_\Omega \nu \mathrm{D}\boldsymbol{u} : \mathrm{D}\boldsymbol{v}\,\mathrm{d}\Omega + \int_\Omega \mathrm{D}\boldsymbol{u}\,\boldsymbol{u}\cdot\boldsymbol{v}\,\mathrm{d}T - \int_\Omega \mathrm{div}\boldsymbol{v}p\,\mathrm{d}\Omega = \int_\Omega \boldsymbol{f}\cdot\boldsymbol{v}\,\mathrm{d}\Omega$$
$$\int_\Omega \mathrm{div}\boldsymbol{u}\,q\,\mathrm{d}\Omega = 0$$

für alle Testfunktionen $\boldsymbol{v}, q$ gelten. In [dLV18] wird gezeigt, dass die selbe Virtuelle-Elemente-Methode, die auch in dieser Arbeit benutzt wird, auch auf die stationären Navier-Stokes-Gleichungen angewendet werden kann und ebenfalls zu einer exakt divergenzfreien Lösung führt. In diskreter Form werden $(\boldsymbol{u}_h, p_h)$ gesucht, so dass

$$a_h(\boldsymbol{u}_h, \boldsymbol{v}_h) + c_h(\boldsymbol{u}_h; \boldsymbol{u}_h, \boldsymbol{v}_h) + b(\boldsymbol{v}_h, p_h) = (\boldsymbol{f}_h^e, \boldsymbol{v}_h)_{L^2(\Omega)}$$
$$b(\boldsymbol{u}_h, q_h) = 0$$

für alle diskreten Testfunktionen $\boldsymbol{v}_h, q_h$ gelten, wobei $a_h(\cdot,\cdot)$, $b(\cdot,\cdot)$ beziehungsweise $\boldsymbol{f}_h^e$ genau so wie in Kapitel 3 beziehungsweise Kapitel 6 definiert sind. Die Trilinearform

$c_h(\cdot;\cdot,\cdot)$ ist für alle $\boldsymbol{w}_h, \boldsymbol{u}_h, \boldsymbol{v}_h$ durch

$$c_h(\boldsymbol{w}_h; \boldsymbol{u}_h, \boldsymbol{v}_h) := \sum_{T\in\mathcal{T}_h} \int_T [P_{k-1}(\mathrm{D}\boldsymbol{u}_h)P_k\boldsymbol{w}_h] \cdot P_k\boldsymbol{v}_h \,\mathrm{d}T$$

definiert.

Allerdings ist diese Methode ebenfalls nicht druckrobust, da in der Diskretisierung erneut L^2-Projektonen genutzt werden, die die Divergenzfreiheit im Allgemeinen nicht erhalten. Damit führt diese Methode insbesondere für kleine Viskositäten ν zu großen Fehlern in der Approximation der Geschwindigkeit.

Auch diese Methode könnte durch einige Änderungen wie schon in [LM16] für Finite-Elemente-Methoden geschehen wahrscheinlich druckrobust gemacht werden. Dafür müsste der Konvektionsterm durch

$$c_h^+(\boldsymbol{w}_h; \boldsymbol{u}_h, \boldsymbol{v}_h) := \sum_{T\in\mathcal{T}_h} \int_T [P_{k-1}(\mathrm{D}\boldsymbol{u}_h)I_{\mathrm{RT}}\boldsymbol{w}_h] \cdot I_{\mathrm{RT}}\boldsymbol{v}_h \,\mathrm{d}T$$

ersetzt werden, wobei I_{RT} die Projektion aus Kapitel 6 ist. Außerdem müsste die diskrete rechte Seite genau wie in dieser Arbeit durch $(\boldsymbol{f}_h^+, \boldsymbol{v}_h)_{L^2(\Omega)}$ ausgetauscht werden.

Darüber hinaus kann auch der nicht-stationäre Fall wie schon in [ALM18] für Finite-Elemente-Methoden vorgeschlagen betrachtet werden, in dem der zeitabhängige Term $d_h(\boldsymbol{u}_h, \boldsymbol{v}_h) := \int_\Omega \partial/\partial t P_k\boldsymbol{u}_h \cdot P_k\boldsymbol{v}_h \,\mathrm{d}\Omega$ durch

$$d_h^+(\boldsymbol{u}_h, \boldsymbol{v}_h) := \int_\Omega \frac{\partial}{\partial t} I_{\mathrm{RT}}\boldsymbol{u}_h \cdot I_{\mathrm{RT}}\boldsymbol{v}_h \,\mathrm{d}\Omega$$

abgelöst wird. Dies könnte in Fällen helfen, wo der Zeitableitungs- und der Konvektionsterm, mit anderen Worten die Materialableitung

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u},$$

nahe an einem Gradienten eines eventuell komplizierten Drucks sind.

Ist eine druckrobuste Diskretisierung der Navier-Stokes-Gleichungen vorhanden, dann kann die daraus resultierende Lösung auch auf gekoppelte Probleme angewendet werden. Dafür eignen sich Virtuelle-Elemente-Methoden besonders gut wegen ihrer hohen Flexibilität in Bezug auf das Gitter. Ein Beispiel dafür sind die Nernst-Planck-Poisson-Navier-Stokes-Gleichungen aus [FLLB09,DGM13,FGL$^+$18], die den Fluss von Elektrolyten beschreiben. Dort wird eine Finite-Elemente-Methode zum Lösen der Navier-Stokes-Gleichungen an eine Finite-Volumen-Methode zum Lösen der Nernst-Planck-Poisson-Gleichungen gekoppelt. Sie garantiert das Maximumsprinzip für die Konzentration der beteiligten Spezies, falls das Geschwindigkeitsfeld divergenzfrei ist [FGL$^+$18]. Mit Hilfe einer druckrobusten Virtuellen-Elemente-Methode könnte die Diskretisierung des Gebiets anstelle von Dreiecken mit allgemeineren Polygonen durchgeführt werden.

Dafür kann darüber nachgedacht werden, die errechnete Lösung mit der Interpolation I_{RT} aus Abschnitt 6 nachzubehandeln. Die so nachbehandelte Lösung kann dann immer noch gut mit anderen Problemen gekoppelt werden.

A. Dokumentation AVEM

Dieser Abschnitt beschäftigt sich mit der Dokumentation des während dieser Arbeit erstellten Softwarepakets, dass sich auf der beiligenden CD beziehungsweise in der beiliegenden .zip-Datei befindet. Die Software ist in MATLAB R2018a implementiert. Kompatibilität zu anderen Versionen oder zu GNU Octave können daher nicht garantiert werden.

Das Paket ist sowohl strukturell als auch syntaktisch an das Paket AFEM angelehnt und nutzt auch einige dessen Funktionen. Daher ist es dringend empfohlen, die Dokumentation von AFEM, wenn nicht bereits geschehen, zuerst zu studieren [CGK+10]. Wegen der großen Ähnlichkeit werden im Folgenden auch nur die wichtigsten Funktionen von AVEM erlärt und exemplarisch Unterschiede zu AFEM aufgeführt. Die Bedienung von AVEM ergibt sich dann analog wie bei AFEM.

Das Softwarepaket löst adaptiv das Stokes-Problem in dem Gebiet $\Omega \subset \mathbb{R}^2$ mit Dirichletrand $\Gamma_D \subset \partial\Omega$ und Neumannrand $\Gamma_N \subset \partial\Omega$, $\overline{\Gamma}_D \cup \overline{\Gamma}_N = \partial\Omega$, $\Gamma_D \cap \Gamma_N = \emptyset$. Zur Erinnerung: Es findet ein Paar $(\boldsymbol{u}, p)$, so dass

$$-\nu\Delta\boldsymbol{u} + \nabla p = \boldsymbol{f} \qquad \text{in } \Omega, \tag{A.1a}$$
$$\operatorname{div}\boldsymbol{u} = 0 \qquad \text{in } \Omega, \tag{A.1b}$$
$$\boldsymbol{u} = \boldsymbol{u}_D \qquad \text{entlang } \Gamma_D, \tag{A.1c}$$
$$\mathrm{D}\boldsymbol{u}\,\boldsymbol{n} = 0 \qquad \text{entlang } \Gamma_N \tag{A.1d}$$

mit einer positiven Viskosität $\nu \in \mathbb{R}$ für eine gegebene rechte Seite $\boldsymbol{f}$ und vorgegebene Dirichletrandbedingungen $\boldsymbol{u}_D$ gilt.

Zum Lösen werden die Daten für die Geometrie und die rechte Seite beziehungsweise die Dirichletrandbedingung benötigt. Begonnen wird daher im Folgenden zuerst mit dem Vorgeben der Geometrie und der Problemdaten.

Daraufhin folgt exemplarisch die Berechnung der Seiten pro Element, um den Unterschied zwischen Funktionen aus AFEM und Funktionen aus AVEM zu beschreiben.

Der Hauptfokus wird dann darauf liegen, den Löser und seine Funktionen zu erläutern.

Darüber hinaus wird auch die grafische Darstellung der Geschwindigkeitslösung besprochen.

Abschließend folgen dann die beiden ausführenden Skripte, mit denen adaptiv das Stokes-Problem gelöst werden kann.

A.1. Einlesen der Daten durch `loadGeometryVEM.m`, `getProblemdata.m`

Das Stokes-Problem wird in einem Gebiet Ω gelöst. Dafür muss die Zerlegung des Gebiets in Polygone für den Computer verarbeitbar sein. Es werden die Datenstrukturen aus Abschnitt 4.1 benötigt, das heißt `c4n` (coordinates for nodes), `n4e` (nodes for elements), `n4sDb` (nodes for Dirichletboundary sides), `n4sNb` (nodes for Neumannboundary sides).

Diese müssen in `loadGeometryVEM.m` bereitgestellt werden. Folgende Ein- und Ausgaben hat die Funktion.

INPUT:

`geometry`, *string*, spezifiziert Geometrie

INPUT optional:

`OPTlvl4Ref`, *natürliche Zahl*, Anzahl an Verfeinerungen
`OPTflag4AFEMref`, *Boolsche Variable*, gibt Verfeinerungsroutine an

OUTPUT:

`c4n`; `n4e`; `n4sDb`; `n4sNb` (s. Abschnitt 4.1)

Es werden jetzt im Folgenden einige Beispielgeometrien erläutert mit deren Verständnis dann auch eigene Geometrien erschaffen werden können.

Durch die Wahl von `'unitSasP'` als `geometry` kann die Zerlegung des Einheitsquadrats aus Abbildung 4.1 aufgerufen werden. Sie ist in Form einer `.mat`-Datei im Unterordner `geometries` gespeichert. Dabei muss die Datei die Strukturen `c4n`, `n4e`, `n4sDb` und `n4sNb` enthalten. Die Strukturen müssen dabei die Form aus Abschnitt 4.1 haben. Das Laden der Datei passiert dabei durch das folgende Listing.

```
58 switch geometry
59     case 'unitSasP'  % square (0,1)^2 as polygons
60         mesh_filepath = 'squareExampleAsPolygons.mat';
61         mesh = load(mesh_filepath);
62         c4n = mesh.c4n;
63         n4e = mesh.n4e;
64         n4sDb = mesh.n4sDb;
65         n4sNb = mesh.n4sNb;
```

Alternativ können die Strukturen auch direkt in die Funktion eingegeben werden. Das Listing, das die Geometrie aus Abbildung 6.2 erzeugt, sieht wie folgt aus.

```
106 case 'unitSasTus'   % square (0,1)^2 as unstructured triangles
107     c4n = [0,0; .5,0; 1,0; 1,1/4; 1,1; 0.5,1; 0,1; 0,7/8; 0,0.5; 0.5,1/3];
108     n4e = [2 9 1; 9 2 10; 2 4 10; 4 2 3; 8 6 7; 9 6 8; 6 9 10; 4 6 10; 6 4 5];
109     n4sDb = [1,2;2,3;3,4;4,5;5,6;6,7;7,8;8,9;9,1];
110     n4sNb = [];
111     rowDist = ones(1,size(n4e,1));
112     n4e = mat2cell(n4e,rowDist);
```

Die Daten werden dort einfach in Form von Matrizen geschrieben. Das ist nur möglich, da in diesem Fall jedes Element aus `n4e` gleich viele Knoten hat. Zur Konvertierung in ein Cell-array werden die Zeile 111 und die darauffolgende benötigt. Damit können auch andere Dreiecksgitter aus AFEM übertragen werden.

Zuletzt wird noch die Geometrie aus Abbildung 4.5 erläutert. Diese ist durch

```
163     case 'l-shapeAsS'   % (-1,1)^2\(0,1)x(0,-1) as squares
164         c4n = [-1,1; -1,0; -1,-1; 0,-1; 0,0; 1,0; 1,1; 0,1];
165         n4e = cell(3,1);
166         n4e{1} = [3, 4, 5, 2];
167         n4e{2} = [8, 1, 2, 5];
168         n4e{3} = [5, 6, 7, 8];
169         n4sDb = [1,2; 2,3; 3,4; 4,5; 5,6; 6,7; 7,8; 8,1];
170         n4sNb = [];
```

implementiert. Dabei wird in den Zeilen 165-168 das Cell-array per Hand angelegt und dann jeweils befüllt.

Weitere bereits implementierte Geometrien werden auch im Hilfetext der Funktion angegeben. Um eigene Geometrien hinzuzufügen, muss ein neuer `case` zusammen mit den Daten eingefügt werden.

Die Geometrie wird nach dem Einlesen dann `OPTlvl4Ref` mal uniform verfeinert, wenn dieser optionale Parameter angegeben ist. Als Verfeinerungsroutine wird entweder der Verfeinerungsalgorithmus aus AFEM [CGK+10] (`OPTflag4AFEMref=true`) oder der aus Abschnitt 5.1.1 (`OPTflag4AFEMref=false`) genutzt.

Um die Problemdaten vorzugeben, kann `getProblemdata.m` benutzt werden, dessen Struktur wie folgt aussieht:

INPUT:

`problem`, *string*, gibt Problemdaten an
`nu`, *float*, Viskositätskonstante $0 \leq \nu \in \mathbb{R}$

OUTPUT:

`rhs(x,y)`, *function handle*, vektorwertige rechte Seite $\boldsymbol{f} \in [L^2(\Omega)]^2$
`u_Db(x,y)`, *function handle*, vektorwertige Dirichletrandbedingung $\boldsymbol{u}_D$
`uExact(x,y)`, *function handle*, vektorwertige exakte Geschwindigkeit $u \in [H^1_D(\Omega)]^2$ wenn bekannt, sonst konstant 0
`pExact(x,y)`, *function handle*, skalarwertige exakte Drucklösung $p \in L^2_0(\Omega)$ wenn bekannt, sonst konstant 0
`dxuExact1(x,y)`, *function handle*, skalarwertige Ableitung der ersten Komponente von $\boldsymbol{u}$ nach x wenn bekannt, sonst konstant 0
`dyuExact1(x,y)`, *function handle*, skalarwertige Ableitung der ersten Komponente von $\boldsymbol{u}$ nach y wenn bekannt, sonst konstant 0
`dxuExact2(x,y)`, *function handle*, skalarwertige Ableitung der zweiten Komponente von $\boldsymbol{u}$ nach x wenn bekannt, sonst konstant 0
`dyuExact2(x,y)`, *function handle*, skalarwertige Ableitung der zweiten Komponente von $\boldsymbol{u}$ nach y wenn bekannt, sonst konstant 0
`TD2u4Db(pts, tangents)`, *function handle*, vektorwertige zweite tangentiale Ableitung von $\boldsymbol{u}_D$ wenn bekannt, sonst konstant `NaN`

Damit die anderen Programme mit den Funktionen umgehen können, müssen die Dimensionen beachtet werden. Die Ausgabe der rechten Seite `rhs(x,y)` muss, wenn n Punkte jeweils der Dimension $n \times 1$ eingegeben werden, die Dimension $n \times 2$ zurückgegeben werden. Analog verhält es sich auch mit `u_Db(x,y)` und `uExact(x,y)`.

Die Funktionen `pExact(x,y)`, `dxuExact1(x,y)`, `dyuExact1(x,y)`, `dxuExact2(x,y)` und `dyuExact2(x,y)` müssen ähnlich dazu eine Rückgabe der Dimension $n \times 1$ haben, wenn die Eingabe jeweils von der Dimension $n \times 1$ war.

Mit der Funktion für die zweite tangentiale Ableitung verhält es sich etwas anders. Sie erwartet drei Eingaben: Die zwei Koordinaten von n Punkten `x,y` jeweils mit der Dimension $n \times 1$ und Tangenten der Dimension $n \times 2$. Unter diesen Eingaben gibt sie die Werte der zweiten tangentialen Ableitung der Dirichletrandbedingung an diesen Punkten zurück und hat daher die Dimension $n \times 2$.

Beispielhaft wird jetzt die Implementierung des Problems `problem='p1'` gezeigt, dass das Hagen-Poiseuille Problem aus Abschnitt 4.3.1 kodiert.

```
58 uExact = @(x,y) zeros(size(x,1),2);
59 pExact = @(x,y) zeros(1, size(x,1));
60 dxuExact1 = @(x,y) zeros(size(x));
61 dyuExact1 = @(x,y) -2.*y+1;
62 dxuExact2 = @(x,y) zeros(size(x));
63 dyuExact2 = @(x,y) zeros(size(x));
64 TD2u4Db = @(x,y, tangents) NaN;
66 switch problem
67     case 'p1'   % Hagen-Poiseuille
68         u_Db = @(x, y) u1Exact(x,y);
69         uExact = @(x,y) u1Exact(x, y);
70         pExact = @(x,y) 1-2.*x;
71         rhs=@(x,y)-nu*laplacianU1(x,y)-2*[ones(length(y),1),zeros(length(x),1)];
72         dxuExact1 = @(x,y) zeros(size(x));
73         dyuExact1 = @(x,y) -2.*y+1;
74         dxuExact2 = @(x,y) zeros(size(x));
75         dyuExact2 = @(x,y) zeros(size(x));
135 function val = u1Exact(x,y)
136 val = zeros(length(y),2);
137 val(:,1) = y.*(1-y);
138 end
140 function val = laplacianU1(x,y)
141 val = zeros(length(x),2);
142 val(:, 1) = -2;
143 end
```

Dabei sind die Zeilen 58-64 für den Fall, dass keine exakte Lösung bekannt ist. Da für Hagen-Poiseuille eine exakte Lösung bekannt ist, werden sie nachfolgend überschrieben.

Neue Probleme können dann analog definiert werden, in dem ein neuer `case` geschrieben wird.

A.2. Hilfsfunktionen am Beispiel `computeS4eVEM.m`

Es gibt in dem Unterorder `/common/` viele Hilfsfunktionen, die für den Löser benutzt werden und geometrische Informationen berechnen. Sie berechnen in der Regel ähnliche Informationen, wie die gleichnamigen in AFEM, erwarten allerdings als Input ein Cell-array oder geben diese als Ausgabe zurück, wenn sich die berechnete Größe in der Anzahl pro Element unterscheiden.

Beispielsweise erwartet `computeArea4eVEM.m` den Vektor `c4n` und das Cell-array `n4e` und gibt einen Vektor `a4e` mit der Größe jedes Elements zurück. Andererseits benötigt `computeNormal4eVEM.m` die gleichen Eingaben und gibt dann ein Cell-array `normal4e`

mit den äußeren Normalenvektoren für jedes Element zurück. Die Rückgabe muss ein Cell-array sein, um der Tatsache Rechnung zu tragen, dass die Elemente im Allgemeinen nicht gleich viele Kanten haben. Werden die Geometrien aus AFEM eingegeben, kann nicht garantiert werden, dass die Ausgabe abgesehen vom Format identisch ist.

Exemplarisch wird zur Verdeutlichung jetzt `computeS4e.m` erläutert. Die Struktur sieht wie folgt aus:

INPUT:

`n4e`, *cell array*, gibt die Knoten der Elemente an (s. vorheriger Abschnitt)

OUTPUT:

`s4e`, *cell array*, sides for elements, gibt die Seitennummern der Seiten in jedem Element an, wobei die n-te lokale Seite zwischen dem n-ten und n+1-ten Knoten liegt.

Die Implementierung ist ähnlich zu der gleichnamigen Funktion bei AFEM, muss dabei aber mit Cell-arrays umgehen. Die Implementierung ist durch

```
37    allSides = getAllSides(n4e);    % all sides
39    [~,ind,back] = unique(sort(allSides,2),'rows','first');
40    [~, sortInd] = sort(ind);
41    sideNr(sortInd) = 1:length(ind); % sideNr(back)' = numbers for allSides
42    s4e=cell(size(n4e));    % j-th row contains number of sides for j-th element
43    for elemNr = 1:nrElems       % loop over all elements
44        len = length(n4e{elemNr});   % number of vertices of j-th element
45        s4e{elemNr} = reshape(sideNr(back(1:len)),size(n4e{elemNr}));
46        back = back((len+1):end);
47    end
```

realisiert.

Dabei wird in Zeile 37 die Funktion `getAllSides.m` aufgerufen, die alle Seiten in der Zerlegung berechnet. Sie gibt die Knotennummern aller Seiten aller Elemente zurück und enthält daher Mehrfachnennungen.

Die Mehrfachnennungen werden in Zeile 39 entfernt. Jede Seite erhält in Zeile 41 eine einzigartige Nummer. Die Seiten sind jetzt global so durchnummeriert, als hätte der Algorithmus jedes Element betrachtet und die lokalen Seiten, sofern sie das erste mal auftauchen, der Reihe nach durchnummeriert.

In der Schleife in den Zeilen 43-47, die über jedes Element geht, werden für jedes Element die zugehörigen Seitennummern rausgesucht und in den entsprechenenden Eintrag in `s4e` geschrieben. Die Zahlen sind in Reihenfolge der lokalen Seiten aufgelistet, das heißt die erste Zahl in `s4e{elemNr}` ist die erste lokale Seite im Element mit der Nummer `elemNr`. Lokal liegt dabei die n-te Seite zwischen dem n-ten und n+1-ten Knoten.

A.3. Der Löser `StokesVemSolver.m`

In diesem Abschnitt wird der Löser des Pakets betrachtet, der später in der adaptiven Schleife aufgerufen wird. Er führt dabei die komplette Berechnung der Virtuellen-Elemente-Lösung durch und benutzt dafür je nach Eingabe die herkömmliche, die erweiterte oder die druckrobuste Virtuelle-Elemente-Methode. Die Funktion hat die folgenden Ein- und Ausgabeparameter.

Algorithm 1 StokesVemSolver

1:	initialization	▷ lines 69-155
2:	**for** elemNr = 1,2,..., # elements **do**	
3:	compute local matrices	▷ lines 156-254
4:	compute local right-hand side	▷ lines 255-377
5:	save local degrees of freedom and local matrices	▷ lines 378-410
6:	**end for**	
7:	build global matrices from local ones and degrees of freedom	▷ lines 411-430
8:	set inhomogeneous Dirichlet boundary data	▷ lines 431-444
9:	solve system of equations for free degrees of freedom	▷ lines 445-447
10:	extract velocity and pressure solution	▷ lines 448-457

INPUT:

`c4n`, `n4e`, `n4sDb`, `n4sNb`, siehe `loadGeometryVEM.m` aus Abschnitt A.1
`nu`, `rhs`, `u_Db`, siehe `getProblemdata.m` aus Abschnitt A.1
`OPTmethod`, *integer*, 1 = herkömmliche VEM, 2 = erweiterte VEM, 3 = druckrobuste VEM

OUTPUT:

`u`, *vector*, enthält die Freiheitsgrade der diskreten Lösung $\boldsymbol{u}_h \in \boldsymbol{V}_2$
`p`, *vector*, enthält die Freiheitsgrade der diskreten Lösung $p_h \in Q_2$
`nrFreeDof`, *integer*, Anzahl an freien Freiheitsgraden
`meanDiam`, *float*, mittlerer Durchmesser h_m
`SstarCol`, *cell array*, enthält elementweise die Matrix S^* (s. Gleichung (4.7))
`stabilizationCol`, *cell array*, enthält elementweise die Stabilisierungsmatrix (s. Gleichungen (3.18) und (4.10))
`intFScalMonCol`, *cell array*, enthält elementweise die Integrale $\int_T \boldsymbol{f} \cdot \boldsymbol{m}_j \, \mathrm{d}T$, für die skalierten Monome $\boldsymbol{m}_j$, $j = 1, 2, \ldots, J$, $J = 2$ für herkömmliche VEM, $J = 12$ für die erweiterte und druckrobuste VEM
`MaMaScalMon4fhCol`, *cell array*, enthält elementweise die Massematrix der skalierten Monome $\int_T \boldsymbol{m}_j \cdot \boldsymbol{m}_\ell \, \mathrm{d}T$, $j, \ell = 1, 2, \ldots, J$, $J = 2$ für herkömmliche VEM, $J = 12$ für die erweiterte und druckrobuste VEM

Eine Zusammenfassung des logischen Aufbaus des Funktion `StokesVemSolver.m` ist in Algorithmus 1 dargestellt. Die Details der einzelnen Schritte werden im Folgenden erläutert.

Während der Initialisierung werden unter anderem geometrische Informationen berechnet, die benötigten Felder für die zu berechnenden Matrizen, die rechte Seite und die Ausgaben angelegt. Der Kürze halber wird dies nicht explizit erläutert.

Darüber hinaus werden aber auch die skalierten Monome aus Definition 3.3 und die Freiheitsgrade durchnummeriert. Das Nummerieren der skalierten Monome erfolgt wie in Gleichung (3.2) beschrieben in

```
k = 2;  % order of method, up to now just k=2 possible
indices4velo = getIndices(k);   %  indices of scalar scaled monomials M_k
```

Dabei gibt `getIndices` für `k=2` den Vektor

$$\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 2 & 0 \\ 1 & 1 \\ 0 & 2 \end{bmatrix}$$

zurück. Die aufgerufe Funktion funktioniert für alle $k \in \mathbb{N}$, allerdings ist der Löser selbst nur für den Fall $k = 2$ geschrieben.

Im weiteren Verlauf werden dann die vektorwertigen skalierten Monome wie in Gleichung (3.3) beschrieben nummeriert. Zur Erinnerung: Dafür taucht jedes skalare skalierte Monom zweimal hintereinander auf, einmal in der ersten und einmal in der zweiten Komponente.

Die globalen Freiheitsgrade werden in

```
off4s = 2 * nrNodes; % offset for DoF wrt. edges
off4m = off4s + 2* (k-1) * nrSides;   % offset for DoF wrt. moments
DbNodes = unique(n4sDb);    % nodes at Db
bdryDof = reshape([2*DbNodes-1,2*DbNodes; off4s+(2*s4Db)-1,off4s+(2*s4Db)]',2*
    length(DbNodes)+2*length(2*s4Db),1);  % nodes at Db and Db sides
nrDof4Vel=nrElems*((k+1)*k/2-1+(k-1)*(k-2)/2)+2*(nrNodes+(k-1)*nrSides);%#DoFVelo
nrDof4Pres = nrElems *  (k+1) * k / 2;  %#DoF pressure
freeDof=setdiff(1:(nrDof4Vel+nrDof4Pres+1), bdryDof);
```

nummeriert.

Für die Geschwindigkeit geschieht dies wie in Definition 4.2 geschrieben in der Reihenfolge $GD_{\boldsymbol{V}}^{\mathcal{N}}$, $GD_{\boldsymbol{V}}^{\mathcal{E}}$, $GD_{\boldsymbol{V}}^{\mathcal{M}_2}$. Die ersten $2|\mathcal{N}|$ Freiheitsgrade gehören zu den Basisfunktionen in Bezug auf die Knoten, die darauffolgenden $2|\mathcal{E}|$ zu den Basisfunktionen in Bezug auf die Kanten und die nächsten $2|\mathcal{T}_h|$ zu den Basisfunktionen, die durch $GD_{\boldsymbol{V}}^{\mathcal{M}_2}$ definiert sind.

Da es sich um vektorwertige Basisfunktionen handelt, gehören zu jedem Knoten und jeder Kante zwei Basisfunktionen: eine für die erste und eine für die zweite Komponente. Sie werden hintereinander folgend aufgelistet, so dass die ersten beiden Freiheitsgrade zu Basisfunktionen vom Knoten 1 gehören, Nummer drei und vier zu Basisfunktionen zum Knoten 2 und so weiter. Genau so verhält es sich auch mit den Basisfunktionen für die Kanten. Die Basisfunktionen für die Divergenzmomente werden dann elementweise aufgeführt.

Da die Ansatzfunktionen für den Druck nur stückweise stetig sind, werden sie elementweise in der Reihenfolge der skalierten Monome aufgelistet.

Nach der Initialisierung werden die lokalen Matrizen K_h^T und $B^T = (b(\varphi_j, m_\ell))$, $j = 1, 2, \ldots, \texttt{nDof}^T, \ell = 1, 2, 3$ in einer Schleife über alle Elemente aufgebaut.

Um dabei über Polygone integrieren zu können, wird im ersten Schritt eine Delaunay-Triangulierung unter Nebenbedingungen durchgeführt. Dafür wird die MATLAB-interne Funktion `delaunayTriangulation.m` genutzt, die unter Angabe des Randes des Polygons, den Rand erhält. Sie ist wie folgt implementiert:

```
160 nrVertC=size(n4e{elemNr},2);    % number of vertices of current element
162 c4nC = c4n(n4e{elemNr}, :);    % current c4n
176 n4sLoc = [(1:(nrVertC-1))',(2:nrVertC)'; nrVertC,1];   % collect all edges
178 DT = delaunayTriangulation(c4nC,n4sLoc);   % constrained delaunayTriangulation
179 n4eDT = DT(isInterior(DT),:);   % select triangles which are inside of polygon
```

Zur Konstruktion der Steifigkeitsmatrix für den Laplaceoperator wird Gleichung (4.11) ausgenutzt, wofür die Matrizen G, C, D, S^* und S aus Abschnitt 4.2.2 benötigt werden.

Die Berechnung von D benötigt die Auswertung der skalierten Monome an den Knoten und an den Kantenmittelpunkten sowie das Integral der Divergenz der skalierten Monome multipliziert mit skalierten Monomen über das Gebiet. Die Integration wird dabei mittels `integrate.m` aus AFEM durchgeführt, wobei

$$\operatorname{div} \boldsymbol{m}_{(\boldsymbol{\alpha},\emptyset)} = \frac{\partial m_{\boldsymbol{\alpha}}}{\partial x} = \frac{1}{h_T} \left(\frac{\boldsymbol{x} - \boldsymbol{x}_T}{h_T} \right)^{(\alpha_1 - 1, \alpha_2)}$$

für einen Multiindex $\boldsymbol{\alpha} = (\alpha_1, \alpha_2) \in \mathbb{N}^2$ mit $|\boldsymbol{\alpha}| \leq 2$ ist. Der zugehörige Quellcode der Berechnung ist in Listing A.1 dargestellt.

Die Implementierung der Matrix C nutzt für die Projektion Gleichung (3.17) und für die Berechnung der Bilinearform Gleichung (3.16) aus. Für den implementierten Fall $k = 2$ errechnet sich $\Delta \boldsymbol{m}_j$ zu $2/h_T^2$ für $j = 7, 8, 11, 12$ und konstant Null sonst, so dass $\Delta \boldsymbol{m}_j$ nicht weiter zerlegt werden muss.

Für das Randintegral der Basisfunktionen kann die Gauß-Lobatto Integrationsformel mit drei Punkten beziehungsweise für den hier betrachteten Fall $k = 2$ die Simpsonregel ausgenutzt werden. Für eine auf einer Kante $E = \operatorname{conv}\{V_\ell, V_{\ell+1}\} \in \mathcal{E}(T)$ integrierbare Funktion $f \in L^1(E)$ ergibt sich das Integral durch

$$\int_E f \, \mathrm{d}E = \frac{|E|}{6} \left(f(V_\ell) + f(V_{\ell+1}) + 4 f((V_\ell + V_{\ell+1})/2) \right) + R, \tag{A.2}$$

wobei der Rest $R \in \mathbb{R}$ gleich Null ist für alle $f \in \mathbb{P}_3(E)$ [Sto05].

Der Quellcode zur Berechnung von C ist in Listing A.2 dargestellt. Dabei enthält `ExN4eC` die Normalenvektoren der Seiten, wobei der Normalenvektor der letzten Seite aus implementatorischen Gründen erneut an der ersten Stelle eingefügt wurde.

Nun kann die Matrix G durch die Formel

$$G = C\,D$$

```
185 D = zeros(nrDof,nrScalMon4velo);
186 scalMonAtVert=scalMonom(c4nC(:,1)',c4nC(:,2)');
187 D(1:2:off4sC,1:2:end)=scalMonAtVert;
188 D(2:2:off4sC,2:2:end)=scalMonAtVert;
189 scalMonAtMidE=scalMonom(c4mid4sC(:,1)',c4mid4sC(:,2)');
190 D(off4sC+1:2:off4mC,1:2:end)=scalMonAtMidE;
191 D(off4sC+2:2:off4mC,2:2:end)=scalMonAtMidE;
193 intDivScalMonom=h4e(elemNr)*sum(integrate(c4nC,n4eDT,@(n4p,Gpts4p,Gpts4ref)
        integrand4divMon(n4p,Gpts4p,Gpts4ref, elemNr), 2),1)/a4e(elemNr);
195 D(end-1:end,:)=intDivScalMonom(1,:,:);
```

Listing A.1: Berechnung der lokalen Matrix D

```
C=zeros(nrScalMon4velo, nrDof);    % Matrix C
% Compute projection
C(1:2,end-1:end)=-eye(2); % DoF wrt. divergence moments
bdryI4n=(h4e(elemNr)*(length4s(ExS4eC(1:end-1)).*ExN4eC(1:end-1,:)+length4s(
    ExS4eC(2:end)).*ExN4eC(2:end,:))/(6*a4e(elemNr)))';   % boundary integral wrt
    . nodes
C(1,1:off4sC)=reshape(bdryI4n.*scalMonAtVert(:,2)',[1,off4sC]);
C(2,1:off4sC)=reshape(bdryI4n.*scalMonAtVert(:,3)',[1,off4sC]);
bdryI4s=(2*h4e(elemNr)*length4s(ExS4eC(2:end)).*ExN4eC(2:end,:)/(3*a4e(elemNr)))
    ';    % boundary integral wrt. midpoints
C(1,off4sC+1:off4mC)=reshape(bdryI4s.*scalMonAtMidE(:,2)',[1,off4mC-off4sC]);
C(2,off4sC+1:off4mC)=reshape(bdryI4s.*scalMonAtMidE(:,3)',[1,off4mC-off4sC]);
% Compute bilinearform
gradScalMonAtVert=gradScalMonom(c4nC(:,1)',c4nC(:,2)');  % grad scaled monomials
gradScalMonAtMidE=gradScalMonom(c4mid4sC(:,1)',c4mid4sC(:,2)');
for i = 3:2:nrScalMon4velo   % loop over vectorvalued scaled monomials
  gradScalMonAtVertC=permute(gradScalMonAtVert(:,(i+1)/2,:),[1,3,2]);
  gradScalMonAtMidEC=permute(gradScalMonAtMidE(:,(i+1)/2,:),[1,3,2]);
  bdryI4n=(length4s(ExS4eC(1:end-1)).*sum(gradScalMonAtVertC.*ExN4eC(1:end-1,:)
      ,2)+length4s(ExS4eC(2:end)).*sum(gradScalMonAtVertC.*ExN4eC(2:end,:),2))/6;
  bdryI4s=2*length4s(ExS4eC(2:end)).*sum(gradScalMonAtMidEC.*ExN4eC(2:end,:),2)
      /3;
  auxiliary=[bdryI4n',bdryI4s',0; zeros(1,off4mC/2+1)];
  C(i, :)=auxiliary(:);
  auxiliary=auxiliary([2,1],:);
  C(i+1, :)=auxiliary(:);
end
C([7,8,11,12],:)=C([7,8,11,12],:)-2*a4e(elemNr).*repmat(C(1:2,:),[2,1])./(h4e(
    elemNr)^2);   % volume integral
```

Listing A.2: Berechnung der lokalen Matrix C

berechnet werden (s. Bemerkung 3.3 in [dBMR14b]), wie einfach nachgerechnet werden kann.

Mit S^* aus Gleichung (4.7) sind alle Matrizen vorhanden, um die lokale Steifigkeitsmatrix K_h^T mittels Gleichung (4.11) berechnen zu können.

Die Berechnung ist in

```
SstarCol{elemNr}=G\C; % S star
G(1:2,:)=0;   % GTilde
consistency=SstarCol{elemNr}'*G*SstarCol{elemNr}; % consistency part
S=D*SstarCol{elemNr};
stabilization=nu*(eye(nrDof)-S)'*(eye(nrDof)-S); % stabilization part
K_hLocal=nu*consistency + stabilization;   % local stiffness matrix for laplacian
```

implementiert.

Die Berechnung der Matrix B^T folgt Gleichung (4.12). Unter Ausnutzung der Integrationsformel A.2 für die Randintegrale ist die Berechnung von B^T durch

```
Blocal=zeros(nrDof, nrScalMon4pres);
Blocal(end-1:end,2:3)=a4e(elemNr)*eye(2)/h4e(elemNr);  % divergence DoF
bdryI4n=(length4s(ExS4eC(1:end-1)).*ExN4eC(1:end-1,:)+length4s(ExS4eC(2:end)).*
    ExN4eC(2:end,:))'/6;   % boundary integral for DoF wrt nodes
bdryI4s=(2*length4s(ExS4eC(2:end)).*ExN4eC(2:end,:)/3)';  % bdry int for
    midpoints
Blocal(1:off4sC,1)=bdryI4n(:);
Blocal(off4sC+1:off4mC,1)=bdryI4s(:);
```

realisiert.

Damit fehlt zum Aufstellen des Gleichungssystems nur noch die Berechnung der rechten Seite. Hier wird zwischen der herkömmlichen, der erweiterten und der druckrobusten Diskretisierung der rechten Seite unterschieden.

Die herkömmliche rechte Seite berechnet sich durch

$$\int_T P_0^T \boldsymbol{f} \cdot \varphi_j \,\mathrm{d}T = \int_T \boldsymbol{f} \cdot P_0^T \varphi_j \,\mathrm{d}T,$$

wobei $P_0^T \varphi_j$ bereits in den ersten beiden Zeilen von C berechnet wird. Die Berechnung des Integrals wird durch folgenden Quellcode ausgeführt.

```
259 intFScalMon=sum(integrate(c4nC,n4eDT,@(~,Gpts4p,~)rhs(Gpts4p(:,1)',Gpts4p(:,2)')
        ,2),1); % integral over current polygon of f
260 MaMaScalMon4fh=a4e(elemNr)*eye(2);    % mass matrix of scaled monomials for fh
271 rhsLocal(:,1)=sum(intFScalMon'.*C(1:2,:),1);  % rhs=P_{k-2}phi*int_TfdT
```

Für die erweiterte rechte Seite muss $P_2^T \varphi_j$ für alle Basisfunktionen φ_j vorhanden sein. Dafür wird $P_2^T \varphi_j = \sum_{\ell=1}^{12} t_\ell^{(j)} \boldsymbol{m}_\ell$ in Basisfunktionen zerlegt und dann

$$\underbrace{\begin{pmatrix} (\boldsymbol{m}_1,\boldsymbol{m}_1)_{L^2(T)} & \cdots & (\boldsymbol{m}_1,\boldsymbol{m}_{12})_{L^2(T)} \\ (\boldsymbol{m}_2,\boldsymbol{m}_1)_{L^2(T)} & \cdots & (\boldsymbol{m}_1,\boldsymbol{m}_{12})_{L^2(T)} \\ \vdots & \ddots & \vdots \\ (\boldsymbol{m}_{12},\boldsymbol{m}_1)_{L^2(T)} & \cdots & (\boldsymbol{m}_{12},\boldsymbol{m}_{12})_{L^2(T)} \end{pmatrix}}_{H^*} \underbrace{\begin{pmatrix} t_1^{(j)} \\ t_2^{(j)} \\ \vdots \\ t_{12}^{(j)} \end{pmatrix}}_{t^{(j)}} = \underbrace{\begin{pmatrix} (\varphi_j,\boldsymbol{m}_1)_{L^2(T)} \\ (\varphi_j,\boldsymbol{m}_2)_{L^2(T)} \\ \vdots \\ (\varphi_j,\boldsymbol{m}_{12})_{L^2(T)} \end{pmatrix}}_{r^{(j)}}$$

berechnet.

Zur Auswertung der rechten Seite durch Gleichung (6.1) wird dabei die Massematrix $H = (\int_T \boldsymbol{m}_j \cdot \boldsymbol{m}_\ell \,\mathrm{d}T)$, $j,\ell = 1,2,\ldots,20$ benötigt. Sie wird zusammen mit

$$\int_T \boldsymbol{f} \cdot \boldsymbol{m}_\ell \,\mathrm{d}T$$

in

```
262 H=permute(sum(integrate(c4nC,n4eDT,@(n4p,Gpts4p,~)integrand4MassMatScalMonom(n4p,
        Gpts4p,1),6),1),[2,3,1]);   % mass matrix of vectorvalued scaleld monomials
264 intFScalMon=sum(integrate(c4nC,n4eDT,@(n4p,Gpts4p,~)integrand4Rhs(n4p,Gpts4p,1)
        ,4),1);  % int_T f*ml dT
266 MaMaScalMon4fh=H(1:nrScalMon4velo,1:nrScalMon4velo);  % mass matrix of scaled
        monomials for fh
```

bestimmt, wobei die Integranden in Listing A.3 angegeben sind.

Damit kann die Berechnung von Gleichung (6.1) durchgeführt werden, wobei für die Zerlegung der skalierten Monome Lemma 3.5 genutzt wird. Mit

$$T^* := \begin{pmatrix} t^{(1)} & t^{(2)} & \ldots & t^{(\mathrm{nDof}^T)} \end{pmatrix} \qquad \text{und} \qquad R := \begin{pmatrix} r^{(1)} & r^{(2)} & \ldots & r^{(\mathrm{nDof}^T)} \end{pmatrix}$$

folgt dann

$$T^* = H^{*-1} R.$$

Dies wird in Listing A.4 durchgeführt.

```
function integrand = integrand4MassMatScalMonom(n4p,Gpts4p,~)
indices_temp = setdiff(getIndices(3),indices4velo,'rows', 'stable');
temp_x = (Gpts4p(:,1)'-cent4e(elemNr,1))/h4e(elemNr);
temp_y = (Gpts4p(:,2)'-cent4e(elemNr,2))/h4e(elemNr);
scalMonAtGpts4p=[scalMonom(Gpts4p(:,1)',Gpts4p(:,2)'),temp_x.^indices_temp(:,1)
    '.*temp_y.^indices_temp(:,2)'];   % scalMon evaluated at Gpts4p
nrScalMon_temp=(k+2)*(k+3);   % number of vectorvalued scaMon for k+1
integrand=zeros(size(n4p,1),nrScalMon_temp,nrScalMon_temp);
for l=1:size(n4p,1)
    integrand(l,1:2:nrScalMon_temp,1:2:nrScalMon_temp)=scalMonAtGpts4p(l,:)'*
        scalMonAtGpts4p(l,:);
    integrand(l,2:2:nrScalMon_temp,2:2:nrScalMon_temp)=integrand(l,1:2:
        nrScalMon_temp,1:2:nrScalMon_temp);
end
end
function integrand = integrand4Rhs(n4p,Gpts4p,~)
scalMonAtGpts4p =  scalMonom(Gpts4p(:,1)',Gpts4p(:,2)');
rhsAtGpts4p = rhs(Gpts4p(:,1)', Gpts4p(:,2)');
integrand = zeros(size(n4p,1),12);
integrand(:,1:2:12) = rhsAtGpts4p(:,1).*scalMonAtGpts4p;
integrand(:,2:2:12) = rhsAtGpts4p(:,2).*scalMonAtGpts4p;
end
```

Listing A.3: Quellcode der Integranden zur Berechnung der erweiterten rechten Seite.

Als dritte mögliche rechte Seite steht noch die druckrobuste Diskretisierung zumindest auf Dreiecken zur Verfügung. Sie folgt im Wesentlichen der Beschreibung aus Abschnitt 6.2.2 und ist in Listing A.5 gezeigt.

Es verbleibt noch die Berechnung der Sicherstellung des Integralmittels für den Druck. Die dazu benötigte Berechnung $\int_T m_\ell \, \mathrm{d}T$, $\ell = 1, 2, 3$ ist bereits in der ersten Zeile von G durchgeführt.

Die lokalen Freiheitsgrade werden wie oben beschrieben durchnummeriert und in

```
off4sC=2*nrVertC;  % offset for edges
off4mC=off4sC+2*(k-1)*nrVertC; % offset for moments
globalDofA=reshape([2*n4e{elemNr}-1,off4s + 2*ExS4eC(2:end)-1,off4m+2*elemNr-1;
    2*n4e{elemNr},off4s + 2*ExS4eC(2:end), off4m+2*elemNr],4*nrVertC+2,1);
globalDofB=(elemNr-1)*3+[1,2,3];
```

berechnet. Sie werden dann genau wie die lokalen Matrizen gespeichert, woraus die globalen Matrizen analog zu AFEM über den `sparse`-Befehl aufgebaut werden.

Das globale Gleichungssystem aus Gleichung (4.13) ergibt sich dann durch

```
mat=[A,B,zeros(nrDof4Vel,1); B',sparse(nrDof4Pres,nrDof4Pres),E; sparse(1,
    nrDof4Vel),E',0];    % system matrix
```

Die Dirichletranddaten werden wie in AFEM durch Auswertung der Funktion $\boldsymbol{u}_D$ an den Randknoten gesetzt. Für die Kantenmittelpunkte wird dabei beachtet, dass das Integralmittel entlang der Seite erhalten bleibt. Dies geschieht durch den Code aus Listing A.6.

Durch

```
F=F-mat*x; % "delete" inhomogenious bdry values
x(freeDof)=mat(freeDof,freeDof)\F(freeDof);   % compute solutions
u = x(1:nrDof4Vel);    % velocity solution
p = x((nrDof4Vel+1):end-1);    % pressure solution
```

```
273 coeff4div=H(1:2:6,1:2:6)\Blocal'; % coefficient for divergence of basisfunctions
274 % Compute full L^2-Projection
275 R=zeros(nrScalMon4velo,nrDof);     % "rhs" for computation of L2-projection
276 R(1:2,:)=a4e(elemNr).*C(1:2,:);    % for l = 1,2
277 for i=3:2:nrScalMon4velo  % loop over all monomials
278     indexC=indices4velo((i+1)/2,:);   % current index
279     % gradient part for vector scal mon, underlying scalMon is in 1st component
280     indexG1=indexC+[1,0];     % index of gradient part of decomposition
281     [~,indexNrG1]=ismember(getIndices(3),indexG1,'rows');
282     indexNrG1=2*find(indexNrG1);  % number of indexG
283     R(i,:)=-h4e(elemNr)*H(indexNrG1,2:2:6)*coeff4div;     % divergence part
284     bdryI4Phi=h4e(elemNr)*permute(integrate(c4n,n4sC,@(n4p,Gpts4p,Gpts4ref)
            integrand4Bdry4L2(n4p,Gpts4p,Gpts4ref,indexG1),2*k+1),[1,3,2]);  % = h_T
            int_(part T) phi_j*n*m(alpha1+1,alpha2)ds
286     ExBdryI4Phi=[bdryI4Phi(end,:);bdryI4Phi];     % first side = last side
287     R(i,1:off4sC)=R(i,1:off4sC)+reshape((ExBdryI4Phi(1:end-1,5:6)+ExBdryI4Phi(2:
            end,1:2))',[1,off4sC]);   % boundary integral for vertices
289     R(i, off4sC+1:off4mC) = R(i, off4sC+1:off4mC)+ reshape((bdryI4Phi(:,3:4))
            ',[1,off4mC-off4sC]);  % boundary integral for midpoints
290     % Remaining part of decomposition
291     if indexC(1,2) > 0
292         coef4R(1:2:nrScalMon4velo) = H(1:2:12,i);     % coefficients
293         indexR = indexC + [1, -1];  % index
294         [~,indexNrR] = ismember(getIndices(3),indexR,'rows');
295         indexNrR = 2*find(indexNrR);     % number of index
296         coef4R(2:2:nrScalMon4velo) = -H(2:2:12,indexNrR);
297         R(i,:)=R(i,:)+indexC(1,2)*coef4R*SstarCol{elemNr}; % = alpha2* int_T Pi_k
                ^Nabla phi_j*m^orth*m(alpha1,alpha2-1) dT
298     end
300     % Analogously for vector scalMon, underlying scalMon is in 2nd component
324 Tstar = H(1:nrScalMon4velo,1:nrScalMon4velo)\R;
326 rhsLocal= (intFScalMon * Tstar)'; % Compute (P_k^0 f,phi_j)_{L^2(T)}
```

Listing A.4: Quellcode der Berechnung der erweiterten rechten Seite.

wird dann die Lösung an den inneren Freiheitsgraden berechnet und die Geschwindigkeits- beziehungsweise die Drucklösung extrahiert.

A.4. Darstellung der Lösung durch `plotVelocitySolution.m`

Für die Darstellung der errechneten Lösung stehen zwei Funktion zur Verfügung. Die Geschwindigkeit kann mittels `plotVelocitySolution.m` und der Druck mit Hilfe von `plotPressureSolution.m` visualisiert werden. Die erste Funktion wird hier kurz erläutert.

INPUT:

`c4n`, `n4e`, siehe `loadGeometryVEM.m` aus Abschnitt A.1
`uh`, siehe `StokesVemSolver.m` aus Abschnitt A.3
`titletext`, *string*, Text für den Titel des Bildes
`scale`, *boolean*, `true` für Skalieren der Geschwindigkeitspfeile zur Übersichtlichkeit, `false` für nicht, s. `quiver.m`
`xseed`, *vector*, x-Koordinaten für Ansatzpunkte für Stromlinien, s. `streamline.m`
`yseed`, *vector*, y-Koordinaten für Ansatzpunkte für Stromlinien, s. `streamline.m`

```
coef4RT1 = zeros(8,nrDof);  % coeff for RT1-basis functions
% RT0
% basis functions w.r.t vertices
bdryI4s=length4s(ExS4eC(2:end)).*ExN4eC(2:end,:)/6;  % D_1, D_2, D_3
coef4RT1(1:3,1:2:6)=diag(bdryI4s(:,1));   % basis fcts wrt 1st component
coef4RT1(1:3,2:2:6)=diag(bdryI4s(:,2));   % basis fcts wrt 2nd component
coef4RT1([3,1,2],1:2:6)=coef4RT1([3,1,2],1:2:6) + diag(bdryI4s([3,1,2],1));
coef4RT1([3,1,2],2:2:6)=coef4RT1([3,1,2],2:2:6) + diag(bdryI4s([3,1,2],2));
% basis functions w.r.t midpoints
coef4RT1(1:3,off4sC+(1:2:6))=diag(Blocal(7:2:12,1));  % 1st component
coef4RT1(1:3,off4sC+(2:2:6))=diag(Blocal(8:2:12,1));  % 2nd component
% RT1
% coefficients for other boundary RT1-functions
integrand=@(n4p,Gpts4p,Gpts4ref)integrand4Bdry4RT1(n4p,Gpts4p,Gpts4ref);
bdryI4Phi=integrate(c4n, n4sC, integrand, k+1);   % D_4, D_5, D_6
coef4RT1(4,1:4)=bdryI4Phi(1,[1:2, 5:6]);  % basis functions wrt 1st vertex
coef4RT1(5,3:6)=bdryI4Phi(2,[1:2,5:6]);   % basis functions wrt 2nd vertex
coef4RT1(6,[5:6,1:2])=bdryI4Phi(3,[1:2,5:6]); % basis functions wrt 3rd vertex
coef4RT1(4,7:8)=bdryI4Phi(1,3:4);   % basis functions wrt midpoints
coef4RT1(5,9:10)=bdryI4Phi(2,3:4);   % basis functions wrt midpoints
coef4RT1(6,11:12)=bdryI4Phi(3,3:4);   % basis functions wrt midpoints
% remaining coefficients, i.e., psi7, psi8
TP0phi=a4e(elemNr).*C(1:2,:);  % |T|P0\phij
D78_13=(cent4e(elemNr,:)-c4nC([3,1,2],:))'./2; % D78(phij), j=1:3
D78_46=(c4nC([3,1,2],:)-c4mid4sC([3,1,2],:))'; % D78(phij), j=4:6
D78_78=(c4mid4sC(1:2,:)-c4nC([3,1],:))';  % D78(phij), j=7:8
for j=1:14
    rs=(TP0phi(:,j)-sum(coef4RT1(1:3,j)'.*D78_13+coef4RT1(4:6,j)'.*D78_46,2));
    coef4RT1(7:8,j)=D78_78\rs;
end
integrand=@(n4p,Gpts4p,Gpts4ref) integrand4fxRT1(n4p,Gpts4p,Gpts4ref);
intFPsi=sum(integrate(c4n,n4e{elemNr},@(n4p,Gpts4p,Gpts4ref)integrand(n4p,Gpts4p,
    Gpts4ref),15),1);   % int_T f*psij dT, psij RT1 basis function
rhsLocal(:,1)=intFPsi*coef4RT1;   % int f*IRT(phi_j)
```

Listing A.5: Dargestellt ist der Quellcode zur Berechnung der druckrobusten rechten Seite. Dies ist nur für Dreiecke implementiert.

INPUT optional:

`OPTuExact`, *fcuntion handle*, exakte Lösung $\boldsymbol{u}$ wird auch dargestellt

OUTPUT:

`Bild`, *figure*, Darstellung von Vektorpfeilen auf einem uniformen Gitter der Geschwindigkeiten `uh` und gegebenenfalls `uExact` sowie Stromlinien für `uh`

Zur Darstellung der Geschwindigkeitslösung wird die von MATLAB bereit gestellte Funktion `quiver.m` genutzt. Sie zeichnet Vektorpfeile entsprechend ihrer Größe innerhalb des Gebiets.

Um die Übersichtlichkeit der Darstellung zu erhöhen, wird die Lösung auf einem uniformen Netz dargestellt. Dafür wird die ursprüngliche Lösung genommen und an den Knotenpunkten des uniformen Netzes interpoliert. Dies geschieht in dem folgenden Quellcode.

```
433 uDbAtBdryNodes = u_Db(c4n(DbNodes,1)',c4n(DbNodes,2)');
436 integrand=@(n4p, Gpts4p, Gpts4ref) u_Db(Gpts4p(:,1)',Gpts4p(:,2)');
437 intMean4uDb=integrate(c4n, n4sDb, integrand, 15)./(repmat(length4s(s4Db),[1,2]));
438 uDbAtBdryMid=3/2*intMean4uDb - 1/4*(u_Db(c4n(n4sDb(:,1),1)',c4n(n4sDb(:,1),2)')+
        u_Db(c4n(n4sDb(:,2),1)',c4n(n4sDb(:,2),2)'));
440 uDbAtBdryNodes=reshape(uDbAtBdryNodes',2*length(uDbAtBdryNodes),1);
441 uDbAtBdryMid=reshape(uDbAtBdryMid',2*length(uDbAtBdryMid),1);
442 x(bdryDof)=[uDbAtBdryNodes; uDbAtBdryMid];   % set boundary values
```

Listing A.6: Gezeigt ist der Quellcode zum Setzen der inhomogenen Dirichletranddaten.

```
44 n4s = computeN4sVEM(n4e);  % nodes for sides
45 mid4s = computeMid4s(c4n, n4s);     % midpoint for sides
46 xTemp = [c4n(:,1); mid4s(:,1)]; % x coordinates
47 yTemp = [c4n(:,2); mid4s(:,2)]; % y coordinates
48 off4m = 2*size(xTemp,1);
49 ax = min(c4n(:,1));
50 bx = max(c4n(:,1));
51 ay = min(c4n(:,2));
52 by = max(c4n(:,2));
53 anz = 50;
54 [XI,YI] = meshgrid(ax:(bx-ax)/anz:bx,ay:(by-ay)/anz:by);
55 UI = griddata(xTemp,yTemp, uh(1:2:off4m),XI,YI);
56 VI = griddata(xTemp,yTemp, uh(2:2:off4m),XI,YI);
```

In Zeile 53 kann eingestellt werden, in wie viele Teile die beiden Koordinaten unterteilt werden sollen. Je größer die Zahl ist, desto feiner wird das Gitter und desto mehr Pfeile werden interpoliert.

Die so interpolierte Lösung wird jetzt mit Hilfe von `quiver.m` auf dem interpolierten Gitter dargestellt.

```
58 figure()
59 if scale == false
60     quiver(XI,YI, UI,VI, 0);  % plot u at interpolated gridpoints
61 else
62 quiver(XI,YI, UI,VI);  % plot u at at interpolated gridpoints
63 end
```

Falls die exakte Lösung `uExact` als Eingabeparameter vorhanden ist, wird jetzt mit ihr analog verfahren mit dem einzigen Unterschied, dass die Lösung an den Gitterpunkten durch direkte Auswertung errechnet wird.

Mit Hilfe der Unterfunktion `plotStreamlines(c4n,n4e,uh, xseed, yseed)` werden die Stromlinien dargestellt. Sie erhält als Eingabeparameter das Gitter, die diskrete Lösung und die Startpunkte der Stromlinien und interpoliert analog zu oben die Lösung auf einem uniformen Gitter. Dann stellt sie Stromlinien mit Hilfe der MATLAB-internen Funktion `streamlines` dar.

Die Funktion `plotPressureSolution(c4n, n4e, ph, titlePres, OPTpExact)` zur Darstellung des Drucks erwartet sehr ähnliche Eingaben. Es werden die Gitterinformationen `c4n` und `n4e` sowie die errechnete Drucklösung `ph` und ein möglicher Titel erwartet. Optional kann auch hier der exakte Druck $p(x, y)$ durch `OPTpExact` als Funktion von x, y mit dargestellt werden.

Zur Darstellung wird die MATLAB-interne Funktion `patch.m` genutzt.

Algorithm 2 avemStokes

1:	set problem data	▷ lines 33-43
2:	initialization	▷ lines 44-87
3:	**while** `true` **do**	
4:	solve problem	▷ lines 88-98
5:	estimate errors	▷ lines 99-119
6:	**if** number degrees of freedom $\geq$ minimal number degree of freedom **then**	
7:	**break while**	▷ line 120
8:	**end if**	
9:	mark elements	▷ lines 121-125
10:	refine elements	▷ lines 126-142
11:	**end while**	
12:	plot or save data	▷ lines 143-164

A.5. Die Hauptprogramme `avemStokes.m`, `vemVsP2b.m`

Die Hauptprogramme des Pakets sind `avemStokes.m` und `vemVsP2b.m`. Sie sind strukturell sehr ähnlich aufgebaut. Beide lösen adaptiv das Stokes-Problem (A.1a)-(A.1d) in dem Gebiet $\Omega \subset \mathbb{R}^2$. Das erstgenannte Skript, `avemStokes.m`, löst dabei das Problem adaptiv mit einer der in dieser Arbeit diskutierten vom Nutzer zu bestimmenden Virtuellen-Elemente-Methode. Im Gegenatz dazu löst das Skript `vemVsP2b.m` das Problem mit der herkömmlichen und der erweiterten Virtuellen-Elemente-Methode sowie der $\boldsymbol{\mathcal{P}}_2^b/\mathcal{P}$-Finiten-Elemente-Methode, was einen Vergleich ermöglicht. Für Gitter, die aus Polygonen bestehen, führt dabei `vemVsP2b.m` eine Delaunay-Triangulierung mit den Knoten durch, um ein Dreiecksgitter zu erhalten.

Im Folgenden wird nur `avemStokes.m` erläutert, da sich die beiden Skripte bis auf die oben gennanten Unterschiede gleichen. Das Skript hat den Aufbau aus Algorithmus 2.

Im ersten Schritt muss das Problem durch den Nutzer spezifiziert werden. Der Quellcode dafür ist in Listing A.7 dargestellt.

In Zeile 34 muss die Geometrie festgelegt werden. Das ist ein String, der auf eine Geometrie in `loadGeometryVEM.m` verweist (s. Abschnitt A.1).

Die ganze Zahl `lvl4Ref` aus Zeile 35 gibt die Anzahl an uniformen Verfeinerungen des Gebiets zu Beginnn der Rechnung an.

In der darauffolgenden Zeile wird festgelegt, ob als Verfeinerungsalgorithmus der Rot-Grün-Blau-Verfeinerungsalgorithmus aus AFEM [CGK$^+$10] oder der aus Kapitel 5.1.1 genutzt werden soll. Dabei steht `flag4AFEMRef = true` für den zuerst genannten Algorithmus und `false` für den zuletzt genannten.

Die Angabe in Zeile 37 ist ein String und verweist auf ein in `getProblemdata.m` definiertes Problem (s. Abschnitt A.1). Dort werden die rechte Seite und die Dirichletrandbedingungen festgelegt.

In der darauffolgenden Zeile kann die Viskosität $0 \leq \nu \in \mathbb{R}$ des betrachteten Fluids

in Form einer Gleitkommazahl eingestellt werden.

Die Zeilen 39-40 definieren in Form einer Gleitkommazahl die Abbruchbedingung für die AVEM-Schleife und den Volumenparameter $\theta \in (0, 1]$ aus Gleichung (5.1) für die adaptive Verfeinerung. Die Wahl $\theta = 1$ führt zu uniformen Verfeinerungen.

Ob die herkömmliche, die erweiterte oder die druckrobuste Virtuelle-Elemente-Methode aus dieser Arbeit genutzt wird, wird in Zeile 41 in Form einer ganzen Zahl bestimmt. Dabei können die Werte 1, 2 oder 3 für die jeweiligen Methoden gewählt werden. Im anderen Skript `vemVsP2b.m` entfällt diese Zeile, da dort die herkömmliche und die erweiterte Virtuelle-Elemente-Methode sowie die $\boldsymbol{\mathcal{P}}_2^b/\mathcal{P}$-Finite-Elemente-Methode benutzt werden.

Abschließend kann in Zeile 42 eingestellt werden, ob das ursprüngliche und das zuletzt erzeugte Gitter, die errechneten Lösungen und die Konvergenzgraphen graphisch dargestellt (`false`) oder in Form verschiedener Dateien gespeichert (`true`) werden sollen. Wird das Speichern gewählt, so muss dafür im Unterordner `experiments` ein Ordner mit dem selben Namen sein, das auch das Problem hat. Wird also beispielsweise `problem = 'p1'` gewählt, muss es einen Ordner `experiments/p1/` geben. In diesen Ordner werden dann die Dateien `geometry_problem_nu_minNrDof_theta_method_date_time_ToSave.dat` gespeichert. Dabei ist `ToSave` entweder `mesh0c4n`, `mesh0n4e`, `mesh1c4n`, `mesh1n4e`, `uh`, `ph` oder `Convergence` und spezifiziert entweder die Anfangszerlegung (`mesh0`), die feinste Zerlegung (`mesh1`), die Lösung (`uh`, `ph`) oder die Fehler (`Convergence`).

Im Schritt der Initialisierung werden die benötigten Vektoren erstellt, in welche die Lösungen und die Fehler geschrieben werden. Aus Gründen der Übersichtlichkeit wird dieser Teil des Quellcodes nicht explizit dargestellt.

Jetzt folgt der Hauptteil des Programms, die AVEM-Schleife. Der erste Schritt, das Lösen, wird durch

```
92 [uh,ph,nrDoF,meanDiam,SstarCol,stabilizationCol,intFScalMonCol,MaMaScalMon4fhCol
       ]=StokesVemSolver(c4n,n4e,n4sDb,n4sNb,nu,rhs,u_Db,method);
96 listNrDof = [listNrDof, nrDoF];
97 listMeanDiam = [listMeanDiam, meanDiam];
```

realisiert.

Dabei wird in den Zeile 92 der Löser `StokesVemSolver.m` aufgerufen. Dieses Programm führt die Virtuelle-Elemente-Methode zum Lösen des Stokes-Problems aus und gibt neben der Lösung `uh, ph`, der Anzahl der Freiheitsgrade `nrDoF` und den mittleren Durchmesser `meanDiam` noch weitere Informationen zur späteren Berechnung des Fehlers beziehungsweise des Fehlerschätzers zurück (s. Abschnitt A.3).

```
34 geometry = 'unitSasTab'; % geometry
35 lvl4Ref = 0;             % level for refinement to start
36 flag4AFEMRef = false;    % true=AFEM red-green-blue, false=Sutton
37 problem = 'p3';          % problem, see getProblemdata.m for predefined examples
38 nu = 1e-0;               % viscosity constant
39 minNrDoF = 1e4;          % minimal number degree of freedom
40 theta = 0.25;            % theta for bulk criterion
41 method = 2;              % 1=normal VEM, 2=enhanced VEM, 3=pressure robust VEM
42 flag4Save = false;       % flag for saving data into files
```

Listing A.7: Inputparamter zum Ausführen der Hauptskripte

```
102 if size(TD2u4Db(c4n(1,1),c4n(1,2),[0 0]),2) == 1
103     [eta4e,~,~,~,~]=computeEta4eVEM(c4n,n4e,n4sDb,nu,rhs,uh,ph,SstarCol,
            stabilizationCol,intFScalMonCol,MaMaScalMon4fhCol);
104 else
105     [eta4e,~,~,~,~]=computeEta4eVEM(c4n,n4e,n4sDb,nu,rhs,uh,ph,SstarCol,
            stabilizationCol,intFScalMonCol,MaMaScalMon4fhCol,TD2u4Db);
106 end
107 [errorVelo, errorPres] = ComputeErrors(c4n, n4e, uh, ph, SstarCol, dxuExact1,
    dyuExact1,dxuExact2, dyuExact2, pExact);
110 listEta = [listEta, sqrt(sum(eta4e))];
111 listErrVel = [listErrVel, errorVelo];
112 listErrPres = [listErrPres, errorPres];
```

Listing A.8: Der Schritt des Abschätzens in der AVEM-Schleife

In den Zeilen darauf werden die Anzahl der Freiheitsgrade und der mittlere Durchmesser gespeichert.

Der Schritt des Abschätzens wird in Listing A.8 durchgeführt.

In den Zeilen 102-106 wird durch `computeEta4eVEM` der Fehler auf jedem Element mittels des Fehlerschätzer aus Abschnitt 5.1.2 geschätzt. Dafür wird überprüft, ob `TD2u4Db` einen vektorwertigen Wert zurückgibt. Diese Funktion realisiert den zusätzlichen Term für das Abschätzen des Fehlers der inhomogenen Dirichletranddaten aus Gleichung (5.2) (s. Abschnitt A.1).

In Zeile 107 werden die Fehler

$$\|\mathrm{D}(\boldsymbol{u} - \Pi_k^{\nabla}\boldsymbol{u}_h)\|_{L^2(\Omega)} \qquad \text{und} \qquad \|p - p_h\|_{L^2(\Omega)}$$

zwischen der exakten Lösung $(\boldsymbol{u}, p)$ und der diskreten Lösung $(\boldsymbol{u}_h, p_h)$ errechnet. Ist die exakte Lösung nicht bekannt, dann berechnet die Zeile die Norm der Lösung, wenn für die exakte Lösung jeweils die Nullfunktion in `getProblemdata.m` angegeben ist (s. Abschnitt A.1). Abschließend werden diese Fehler gespeichert.

Die beiden letzten Schritte, das Markieren und das Verfeinern, sind durch

```
124 markedElem = markBulkVEM(eta4e, theta);
128 if flag4AFEMRef == true
129     s4e = computeS4eVEM(n4e);  % sides for elements
130     markedSidesAll = cell2mat(cellfun(@(cell) cell', s4e(markedElem), '
            UniformOutput', false)); % all sides of marked elements
131     markedSides = unique(markedSidesAll); % marked sides
132     n4s = computeN4sVEM(n4e);  % nodes for sides
133     markedNodes = n4s(markedSides,:);
134     [c4n,n4e,n4sDb,n4sNb,~,~,~]=refineRGB(c4n,cell2mat(n4e),n4sDb,n4sNb,
            markedNodes);
135     rowDist = ones(1,size(n4e,1));  % size of cell
136     n4e = mat2cell(n4e,rowDist);
137 else
138     [c4n, n4e, n4sDb, n4sNb] = refineSut(c4n,n4e,n4sDb,n4sNb, markedElem);
139 end
```

implementiert.

Das Markieren erfolgt in Zeile 124 durch `markBulkVEM`. Dafür wird das Dörfler-Markieren aus Abschnitt 5.1 genutzt.

In den Zeilen 128-138 wird das Verfeinern durchgeführt. Je nachdem, ob `flag4AFEMRef` wahr oder falsch ist, werden entweder die für die AFEM-Verfeinerung benötigten Informationen berechnet und dann rot-grün-blau verfeinert oder die Verfeinerungsstrategie aus dieser Arbeit verwendet. Da die Verfeinerungsroutine aus AFEM die Information `n4e` als Matrix und nicht als Cell-array erwartet, muss dafür zuerst `n4e` in eine Matrix und darauf für den Löser wieder in ein Cell-array umgewandelt werden.

Der Rest des Skripts dient der grafischen Darstellung beziehungsweise dem Speichern der Lösung und der Konvergenzraten abhängig davon, ob `flag4Save` falsch oder wahr ist. Auch dieser Teil des Quellcodes ist aus Übersichtlichkeit weggelassen.

Literaturverzeichnis

Die Literaturangaben sind alphabetisch nach den Nachnamen der Autoren sortiert. Bei mehreren Autoren wird nach dem ersten Autor sortiert.

[AAB+13] Ahmad, Bashir ; Alsaedi, Ahmed ; Brezzi, Franco ; Marini, Luisa D. ; Russo, Alessandro: Equivalent projectors for virtual element methods. In: *Computers & Mathematics with Applications* 66 (2013), Nr. 3, S. 376 – 391

[ACF99] Alberty, Jochen ; Carstensen, Carsten ; Funken, Stefan A.: Remarks around 50 lines of Matlab: short finite element implementation. In: *Numerical Algorithms* 20 (1999), Nr. 2-3, S. 117–137

[ALM17] Ahmed, Naveed ; Linke, Alexander ; Merdon, Christian: Towards Pressure-Robust Mixed Methods for the Incompressible Navier–Stokes Equations. In: *Computational Methods in Applied Mathematics* 18 (2017), 11

[ALM18] Ahmed, Naveed ; Linke, Alexander ; Merdon, Christian: On Really Locking-Free Mixed Finite Element Methods for the Transient Incompressible Stokes Equations. In: *SIAM Journal on Numerical Analysis* 56 (2018), Nr. 1, S. 185–209

[Bar16] Bartels, Sören: *Texts in applied mathematics*. Bd. 64: *Numerical approximation of partial differential equations*. Springer International Publishing Switzerland, 2016

[BBF13] Boffi, Daniele ; Brezzi, Franco ; Fortin, Michel: *Mixed finite element methods and applications*. Springer-Verlag Berlin Heidelberg, 2013 (Springer Series in Computational Mathematics 44)

[BCD04] Bartels, Sören ; Carstensen, Carsten ; Dolzmann, Georg: Inhomogeneous Dirichlet conditions in a priori and a posteriori finite element error analysis. In: *Numerische Mathematik* 99 (2004), Nr. 1, S. 1–24

[Bes06] Bestehorn, Michael: *Hydrodynamik und Strukturbildung Mit einer kurzen Einführung in die Kontinuumsmechanik*. Berlin, Heidelberg, 2006 (Springer-Lehrbuch)

[BF91] Brezzi, Franco ; Fortin, Michel: *Mixed and Hybrid Finite Element Methods edited by Franco Brezzi, Michel Fortin*. New York, NY, 1991 (Springer Series in Computational Mathematics, 15)

[Bre14] Brezzi, Franco: The Great Beauty of VEMs. In: *Proceedings of the International Congress of Mathematicians* 1 (2014), S. 217–235

[BS02] Brenner, Susanne C. ; Scott, Larkin R.: *Texts in Applied Mathematics*. Bd. 15: *The Mathematical Theory of Finite Element Methods*. 2. Auflage. Springer-Verlag New York, 2002

[CB18] Carstensen, Carsten ; Brenner, Susanne C.: Finite Element Methods. In: *Encyclopedia of Computational Mechanics Second Edition*, John Wiley and Sons, 2018, S. 1 – 47

[CFPP14] Carstensen, Carsten ; Feischl, Michael ; Page, Marcus ; Praetorius, Dirk: Axioms of adaptivity. In: *Comput. Math. Appl.* 67 (2014), Nr. 6, S. 1195–1253

[CGK+10] Carstensen, Carsten ; Gedicke, Joscha ; Kern, Leonard ; Neumann, Johannes ; Rabus, Hella ; Rozova, Maria: *AFEM Softwarepackage and documentation*. 2010. – rev. 577, unveröffentlicht, verfügbar online auf Nachfrage, Humboldt-Universität zu Berlin

[Clo04] Clough, Ray W.: Early history of the finite element method from the view point of a pioneer. In: *International Journal for Numerical Methods in Engineering* 60 (2004), Nr. 1, S. 283–287

[CP18] Carstensen, Carsten ; Puttkammer, Sophie: A low-order discontinuous Petrov-Galerkin method for the Stokes equations. In: *Numerische Mathematik* 140 (2018), September, Nr. 1, S. 1–34

[CR73] Crouzeix, Michel ; Raviart, Pierre-Arnaud: Conforming and nonconforming finite element methods for solving the stationary Stokes equations I. In: *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique* 7 (1973), Nr. R3, S. 33–75

[dBC+13] da Veiga, Lourenco B. ; Brezzi, Franco ; Cangiani, Andrea ; Manzini, Gianmarco ; Marini, Luisa D. ; Russo, Alessandro: Basic principles of Virtual Element Methods. In: *Mathematical Models and Methods in Applied Sciences* 23 (2013), Nr. 01, S. 199–214

[dBMR14a] da Veiga, Lourenco B. ; Brezzi, Franco ; Marini, Luisa D. ; Russo, Alessandro: H(div) and H(curl)-conforming VEM. In: *Numerische Mathematik* 133 (2014), 07

[dBMR14b] da Veiga, Lourenco B. ; Brezzi, Franco ; Marini, Luisa D. ; Russo, Alessandro: The Hitchhiker's Guide to the Virtual Element Method. In: *Mathematical Models and Methods in Applied Sciences* 24 (2014), 07

[dBMR14c] da Veiga, Lourenco B. ; Brezzi, Franco ; Marini, Luisa D. ; Russo, Alessandro: Mixed Virtual Element Methods for general second order elliptic

problems on polygonal meshes. In: *ESAIM: Mathematical Modelling and Numerical Analysis* 26 (2014), 12

[DGM13] Dreyer, Wolfgang ; Guhlke, Clemens ; Müller, Rüdiger: Overcoming the shortcomings of the Nernst-Planck model. In: *Physical chemistry chemical physics : PCCP* 15 (2013), 04

[dLR17] da Veiga, Lourenco B. ; Lovadina, Carlo ; Russo, Alessandro: Stability analysis for the virtual element method. In: *Mathematical Models and Methods in Applied Sciences* 27 (2017), Nr. 13, S. 2557–2594

[dLV17] da Veiga, Lourenco B. ; Lovadina, Carlo ; Vacca, Giuseppe: Divergence free virtual elements for the stokes problem on polygonal meshes. In: *ESAIM: M2AN* 51 (2017), Nr. 2, S. 509–535

[dLV18] da Veiga, Lourenco B. ; Lovadina, Carlo ; Vacca, Giuseppe: Virtual Elements for the Navier–Stokes Problem on Polygonal Meshes. In: *SIAM Journal on Numerical Analysis* 56 (2018), Nr. 3, S. 1210–1242

[DV19] Dassi, Franco ; Vacca, Giuseppe: Bricks for the mixed high-order virtual element method: Projectors and differential operators. In: *Applied Numerical Mathematics,* (2019). – im Druck

[Dö96] Dörfler, Willy: A Convergent Adaptive Algorithm for Poisson's Equation. In: *Siam Journal on Numerical Analysis - SIAM J NUMER ANAL* 33 (1996), 06

[Erv12] Ervin, Vincent: Computational bases for RTk and BDMk on triangles. In: *Computers & Mathematics with Applications* 64 (2012), 10, S. 2765–2774

[Eva10] Evans, Lawrence C.: *Graduate studies in mathematics.* Bd. 19: *Partial differential equations.* 2nd edition. Providence, RI, 2010

[FGL+18] Fuhrmann, Jürgen ; Guhlke, Clemens ; Linke, Alexander ; Merdon, Christian ; Müller, Rüdiger: *Models and numerical methods for electrolyte flows.* 2018. – WIAS preprint No. 2525, unveröffentlicht, abrufbar unter `10.20347/WIAS.PREPRINT.2525`

[FLLB09] Fuhrmann, Jürgen ; Linke, Alexander ; Langmach, Hartmut ; Baltruschat, Helmut: Numerical calculation of the limiting current for a cylindrical thin layer flow cell. In: *Electrochimica Acta* 55 (2009), Nr. 2, S. 430 – 438

[GDN98] Griebel, Michael ; Dornseifer, Thomas ; Neunhoeffer, Tilman: *Numerical Simulation in Fluid Dynamics.* Society for Industrial and Applied Mathematics, 1998 (SIAM monographs on mathematical modeling and computation)

[GF05] Grote, Karl-Heinrich ; Feldhusen, Jörg: *Dubbel: Taschenbuch für den Maschinenbau*. Einundzwanzigste, neubearbeitete und erweiterte Auflage. Springer Berlin Heidelberg, 2005

[GR86] Girault, Vivette ; Raviart, Pierre-Arnaud: *Finite Element Methods for Navier-Stokes Equations Theory and Algorithms*. Berlin, Heidelberg, 1986 (Springer Series in Computational Mathematics, 5)

[GS17] Guzman, Johnny ; Scott, Larkin R.: The Scott-Vogelius finite elements revisited. In: *Mathematics of Computation* 88 (2017), 04

[HSV12] Hannukainen, Antti ; Stenberg, Rolf ; Vohralík, Martin: A unified framework for a posteriori error estimation for the Stokes problem. In: *Numerische Mathematik* 122 (2012), December, Nr. 4, S. 725–769

[JK18] J.Gander, Martin ; Kwok, Felix: *Numerical Analysis of Partial Differential Equations Using Maple and MATLAB*. Philadelphia, PA : Society for Industrial and Applied Mathematics, 2018

[JLM+17] John, Volker ; Linke, Alexander ; Merdon, Christian ; Neilan, Michael ; Rebholz, Leo G.: On the Divergence Constraint in Mixed Finite Element Methods for Incompressible Flows. In: *SIAM Review* 59 (2017), 01, S. 492–544

[Lan07] Langelaar, M: The Use of Convex Uniform Honeycomb Tessellations in Structural Topology Optimization. In: *Proceedings 7th World Congress on Structural and Multidisciplinary Optimization*, WCSMO, 2007, S. 2469–2478

[Lin14] Linke, Alexander: On the role of the Helmholtz decomposition in mixed methods for incompressible flows and a new variational crime. In: *Computer Methods in Applied Mechanics and Engineering* 268 (2014), 01, S. 782–800

[LLMS17] Lederer, Philip L. ; Linke, Alexander ; Merdon, Christian ; Schöberl, Joachim: Divergence-free Reconstruction Operators for Pressure-Robust Stokes Discretizations with Continuous Pressure Finite Elements. In: *SIAM Journal on Numerical Analysis* 55 (2017), Nr. 3, S. 1291–1314

[LM16] Linke, Alexander ; Merdon, Christian: Pressure-robustness and discrete Helmholtz projectors in mixed finite element methods for the incompressible Navier-Stokes equations. In: *Computer Methods in Applied Mechanics and Engineering* 311 (2016), 11, S. 304–326

[LMS19] Lederer, Philip L. ; Merdon, Christian ; Schöberl, Joachim: Refined a posteriori error estimation for classical and pressure-robust Stokes finite element methods. In: *Numerische Mathematik* 142 (2019), July, Nr. 3, S. 713–748

[Sto05] Stoer, Josef: *Numerische Mathematik 1.* Berlin, Heidelberg : Springer Berlin Heidelberg, 2005 (Springer-Lehrbuch)

[Sut17] Sutton, Oliver J.: *Virtual Element Methods*, University of Leicester, Diss., Juni 2017

[SV85] Scott, Larkin R. ; Vogelius, Michael: Norm estimates for a maximal right inverse of the divergence operator in spaces of piecewise polynomials. In: *ESAIM: M2AN* 19 (1985), Nr. 1, S. 111–143

[Vac17] Vacca, Giuseppe: An H^1-conforming Virtual Element for Darcy and Brinkman equations. In: *Mathematical Models and Methods in Applied Sciences* 28 (2017), 11, S. 159–194

[Ver89] Verfürth, Rüdiger: A posteriori error estimators for the Stokes equations. In: *Numerische Mathematik* 55 (1989), May, Nr. 3, S. 309–325

[Wei17] Weisser, Steffen: Residual based error estimate and quasi-interpolation on polygonal meshes for high order BEM-based FEM. In: *Computers & Mathematics with Applications* 73 (2017), Nr. 2, S. 187 – 202

Danksagung

An dieser Stelle möchte ich mich bei allen Personen bedanken, die mich während der Anfertigung der Masterarbeit unterstützt haben und die Arbeit in dieser Form möglich gemacht haben.

Zuerst gilt mein Dank Herrn Prof. Dr. Carstensen für das Ermöglichen dieser Masterarbeit. Dafür, mit Virtuellen-Elemente-Methoden in Kooperation mit dem Weierstraß-Institut arbeiten zu können und für die inhaltlichen Anregungen bezüglich der numerischen Experimente, bin ich sehr dankbar.

Ein besonders großer Dank gilt Herrn Dr. Merdon vom Weierstraß-Institut für meine Betreuung und seine in mich investierte Zeit. Vielen Dank für die etlichen Stunden an Fehlersuche im Programmcode, für die überaus erhellenden Diskusionen über die Virtuellen-Elemente-Methoden und die enorm vielen Erklärungen, ohne die ich nie so gut in dieses Thema gefunden hätte. Darüber hinaus möchte ich mich sehr aufrichtig für die inhaltliche Unterstützung beim Anfertigen der Beweise und dem Aufbau der Arbeit selbst, der Gestaltung der numerischen Experimente und die konstruktive Kritik bei der Erstellung der Arbeit bedanken. Ohne ihn wäre diese Arbeit wohl so nur schwer möglich gewesen.

Ein herzliches „Dankeschön“ steht auch meinen Freunden und Kommilitonen, Stephen und Joshua, zu. Zum Einen für die Ablenkung von der Arbeit und zum Anderen auch für die Korrekturvorschläge. Darüber hinaus gilt mein Dank meinem ehemaligen Übungsleiter und Kollegen, Franz, für die Einführungen in Git und VIm, die mir das Schreiben dieser Arbeit deutlich erleichtert haben.

Ebenfalls möchte ich die Gelegenheit nutzen, um mich bei meinen Eltern, Imke und Johannes, für alles zu bedanken, was sie mir ermöglicht haben. Für die Unterstützung bei meiner Studienwahl, meinen Studienwechseln und der Wahl des Studienorts bin ich zutiefst dankbar.

Mein größter Dank gilt meiner Verlobten, Klara. Ohne dich hätte ich vielleicht niemals Mathematik studiert und hätte diese Arbeit nie anfertigen können. Deine vielen Stunden an Korrekturarbeit und die unermüdlichen Hinweise zur inhaltlichen Gestaltung haben die Arbeit ein enormes Stück verbessert. Für deine bedingungslose Unterstützung, das Rückenfreihalten während der letzten Wochen der Arbeit, die Aufmunterungen in Phasen, wo es nicht so gut lief, die geistige Zerstreuung und dringend benötigten, teilweise verpflichtenden Ablenkungen bin ich dir von ganzem Herzen dankbar. Die Arbeit wäre ohne dich nie so geworden, wie sie jetzt ist.

Selbstständigkeitserklärung

Hiermit erkläre ich, Derk Frerichs, dass ich die vorliegende Arbeit selbstständig verfasst und noch nicht für andere Prüfungen eingereicht habe. Sämtliche Quellen, einschließlich Internetquellen, die unverändert oder abgewandelt wiedergegeben werden, insbesondere Quellen für Texte, Grafiken, Tabellen und Bilder, sind als solche kenntlich gemacht. Mir ist bekannt, dass bei Verstößen gegen diese Grundsätze ein Verfahren wegen Täuschungsversuchs bzw. Täuschung eingeleitet wird.

Berlin, den 20. August 2019 .....
(Unterschrift)