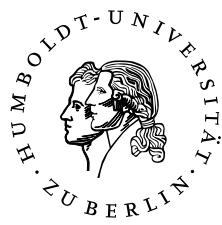


HUMBOLDT-UNIVERSITÄT ZU BERLIN
MATHEMATISCH-NATURWISSENSCHAFTLICHE FAKULTÄT
INSTITUT FÜR MATHEMATIK



Drei adaptive Finite-Elemente-Methoden in 3D für das Poisson-Problem

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Science (B. Sc.)

eingereicht von: Nando Farchmin

geboren am: 11.11.1990

geboren in: Berlin

Betreuer: Prof. Dr. rer. nat. Carsten Carstensen

eingereicht: Berlin, den 29. September 2014

Inhaltsverzeichnis

1 Einleitung	1
2 Grundlagen und Notationen	4
2.1 Notationen	4
2.2 Das Poisson-Problem	5
3 Finite-Elemente-Methoden	7
3.1 Die P_1 -konforme Finite-Elemente-Methode	7
3.2 Die CR-nicht-konforme Finite-Elemente-Methode	9
3.3 Die RT_0 -gemischte Finite-Elemente-Methode	11
3.4 Fehlerschätzer	17
4 Implementierung	20
4.1 Berechnung des P_1 -konformen Lösers	20
4.2 Berechnung des Crouzeix-Raviart Lösers	21
4.3 Berechnung des Raviart-Thomas Lösers	22
4.4 Berechnung der Fehlerschätzer	27
4.5 Berechnung der Marini-Darstellung	30
4.6 Realisierung	31
5 Numerische Beispiele	36
5.1 „Cube“ mit exakter Lösung	36
5.2 „L-Shape“ mit exakter Lösung	41
5.3 „Cube“ mit konstanter rechter Seite	46
5.4 „L-Shape“ mit konstanter rechter Seite	49
5.5 „Fichera-Corner“ mit konstanter rechter Seite	53
6 Auswertung	57
6.1 Sprunghaftes Verhalten der konformen Methode	57
6.2 Adaptivität	57
6.3 Eigenschaften der Fehlerschätzer	58
6.4 Marini-Identität	58
6.5 Fazit und Ausblick	58
Literatur	60
Anhang: Verzeichnisstruktur	61

Zusammenfassung

Diese Arbeit beschäftigt sich mit dem Poisson-Problem in drei Raumdimensionen und dessen Lösbarkeit mittels Finite-Elemente-Methoden. Dafür werden drei Finite-Elemente-Methoden, eine konforme, eine nicht-konforme und eine gemischte Methode, sowie ein zu jeder Methode passender Fehlerschätzer implementiert. Zusätzlich wird der Zusammenhang zwischen der nicht-konformen und der gemischten Methode untersucht und ausgewertet. Die erstellten Programme werden an fünf Beispielen getestet, wobei sich die in drei Raumdimensionen erwarteten Ergebnisse beobachten lassen.

1 Einleitung

Partielle Differentialgleichungen treten in Natur und Technik nahezu überall auf. Sei es, um die Konzentrationsverteilung in einem chemischen Gemisch zu bestimmen, die Wechselwirkung von elektrischen und magnetischen Feldern zu beschreiben oder eine Autokarosserie so zu gestalten, dass möglichst wenig störende Luftverwirbelungen entstehen. Allerdings sind nur für die wenigsten dieser Differentialgleichungen analytische Lösungen bekannt und diese auch nur auf einigen wenigen Gebieten. Daher ist es von großem Interesse, die Lösungen solcher Differentialgleichungen zumindest approximieren zu können.

Viele der auftretenden Differentialgleichungen können auf einen einfacheren Modellfall zurückgeführt werden. Anfang des 19. Jahrhunderts von Siméon Denis Poisson beschrieben, ist das Poisson-Problem eine elliptische partielle Differentialgleichung zweiter Ordnung, die heute ein breites Anwendungsgebiet in vielen Naturwissenschaften findet. In den meisten Anwendungsgebieten ist es von besonderem Interesse, sich die Poisson-Gleichung in drei Raumdimensionen anzuschauen, daher beschäftige ich mich im Rahmen dieser Arbeit mit diesem Spezialfall.

Sei $\Omega \subseteq \mathbb{R}^3$ ein beschränktes Lipschitz-Gebiet mit Rand $\partial\Omega$ und sei weiterhin eine rechte Seite $f: \Omega \rightarrow \mathbb{R}$ gegeben. Dann lautet das Poisson-Problem: Finde eine Funktion $u: \Omega \rightarrow \mathbb{R}$, die

$$-\Delta u = f \quad (1.1)$$

erfüllt. Die eindeutige Lösbarkeit erhalten wir nur durch die Vorgabe von Randdaten. Seien also $u_D: \Gamma_D \rightarrow \mathbb{R}$ die Dirichlet-Randdaten auf dem Dirichlet-Rand $\Gamma_D \subseteq \partial\Omega$ und $g: \Gamma_N \rightarrow \mathbb{R}$ die Neumann-Randdaten auf dem Neumann-Rand $\Gamma_N = \partial\Omega \setminus \Gamma_D$. Wir erhalten die notwendigen Bedingungen an die Randdaten durch

$$u = u_D \text{ auf } \Gamma_D, \quad (1.2)$$

$$\nabla u \cdot \nu = g \text{ auf } \Gamma_N, \quad (1.3)$$

wobei $\nu: \Gamma_N \rightarrow \mathbb{R}^3$ das äußere Normaleneinheitsvektorfeld auf dem Neumann-Rand ist. Die numerische Lösung der Poisson-Gleichungen wurde für zwei Raumdimensionen bereits für konforme und nicht-konforme [Bra07] sowie gemischte [BC05] Finite-Elemente-Methoden untersucht. Sowohl bei der konformen wie auch bei der nicht-konformen Methode werden die Gleichungen direkt diskretisiert, wobei die Lösungsräume so gewählt werden, dass Existenz und Eindeutigkeit der Lösung aus dem Rieszschen Darstellungssatz bzw. dem Lemma von Lax-Milgram folgen. Zur Veranschaulichung ist in Abbildung 1.2 die Lösung des Poisson-Problems mit konstanter rechter Seite und reinen Dirichlet-Randdaten auf dem Einheitswürfel dargestellt.

Bei der gemischten Finite-Elemente-Methode überführen wir das Poisson-Problem in ein System erster Ordnung. Dazu führen wir eine zusätzliche Variable $p: \Omega \rightarrow \mathbb{R}^3$ für den

Fluss ein, für die gilt

$$\begin{aligned} -\operatorname{div} p &= f, \\ p &= \nabla u. \end{aligned} \tag{1.4}$$

Aussagen über Existenz und Eindeutigkeit der diskreten Lösung erhält man, wie wir zeigen werden, über die Darstellung der Lösung $(p_{\text{RT}}, u_{\text{RT}})$ mittels der Lösung der nicht-konformen Methode und einem Eindeutigkeitsargument [BF91, §II.]. Diese Darstellung der Lösung der gemischten Methode geht auf Marini zurück [Mar85] und gibt uns damit zusätzlich eine Möglichkeit unsere Implementation der gemischten Methode zu überprüfen.

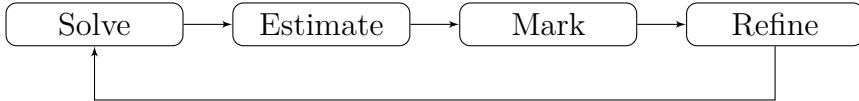


Abbildung 1.1: Der AFEM Zyklus

Um Rechenzeit und -kapazität zu minimieren, wollen wir adaptive Algorithmen benutzen. Da die exakte Lösung in den meisten Fällen nicht bekannt ist, ist es nicht möglich, den exakten Fehler der numerischen Lösung zu berechnen. Daher betrachten wir drei Fehlerschätzer, die analog zu bereits bekannten Schätzern aus dem zweidimensionalen Fall, die in [Bra07, BC05] behandelt werden, definiert sind. Wir erhalten durch deren Effizienz und Zuverlässigkeit eine sinnvolle Abschätzung für den exakten Fehler. Dies ermöglicht es uns eine Triangulierung lokal nur dort zu verfeinern, wo die Näherungslösung zu ungenau ist. Realisieren werden wir dies durch den in Abbildung 1.1 dargestellten AFEM-Algorithmus.

Unser Ziel ist es, numerische Beispielrechnungen durchzuführen, die uns die im dreidimensionalen Fall erwarteten Konvergenzeigenschaften liefern. Für die Rechnungen benutzen wir eine auf den dreidimensionalen Fall erweiterte Version von [Car09a], wobei wir für die Verfeinerung einer dreidimensionalen Triangulierung den in [Ste08] vorgestellten und von [GS11] implementierten Algorithmus verwenden. Wir werden drei verschiedene Lösungsalgorithmen implementieren, einen konformen, einen nicht-konformen nach Crouzeix und Raviart sowie einen gemischen nach Raviart und Thomas. Die Triangulierungen wollen wir sowohl uniform als auch adaptiv verfeinern, dafür implementieren wir zusätzlich je einen Fehlerschätzer für unsere Lösungsalgorithmen. Wir werden weiterhin eine Alternative zur Berechnung der gemischten Methode herausarbeiten und unsere Ergebnisse mit dieser Alternative überprüfen. Um die Konvergenz unserer Fehlerschätzer zu analysieren, werden wir den exakten Energiefehler berechnen, sofern die exakte Lösung des jeweiligen Problems bekannt ist.

Die Arbeit beginnt in Kapitel 2 mit einigen Notationen sowie den theoretischen Grundlagen zum Poisson-Problem im Allgemeinen und für drei Raumdimensionen im Speziellen. In Kapitel 3 werden wir das Problem entsprechend der gewählten Methode diskretisieren. Dabei gehen wir bei der gemischten Methode auf die sogenannte Lagrange-Multiplikator-Technik zur späteren Implementierung ein und stellen eine alternative Darstellung, die auf Marini zurück geht, vor. Außerdem betrachten wir für jede Methode eine Möglichkeit,

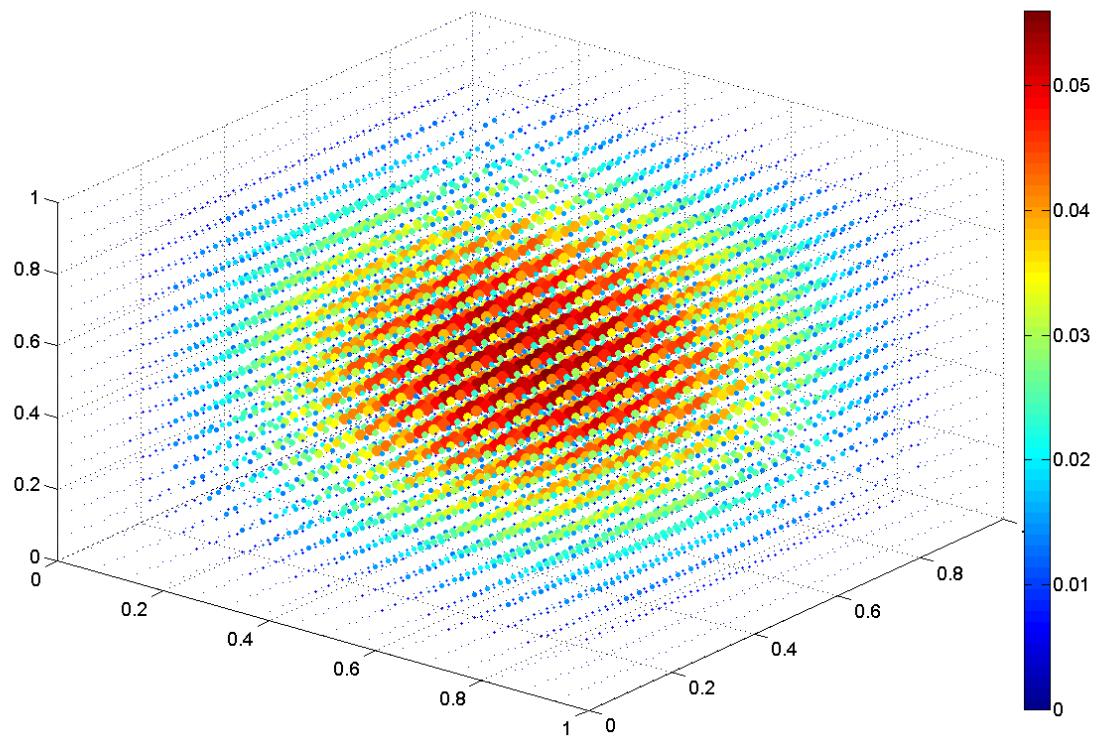


Abbildung 1.2: Näherungslösung des Poisson-Problems mit rechter Seite $f \equiv 1$ und reinen Dirchlet-Nullranddaten durch die P_1 -konforme Finite-Elemente-Methode mit 5735 Freiheitsgraden.

den entstandenen Fehler abzuschätzen. Die Berechnung der Steifigkeitsmatrizen und der rechten Seiten, der Marini-Darstellung sowie der Fehlerschätzer und der exakten Energiefehler werden wir in Kapitel 4 vornehmen. Die Realisierung dieser Programme wird dabei in Abschnitt 4.6 und in den Matlab-Funktionen dieser Arbeit beschrieben. In Kapitel 5 sind die Ergebnisse einiger Beispielrechnungen dargestellt. Dabei handelt es sich um verschiedene Standardbeispiele mit oder ohne bekannter exakter Lösung, wie unter anderem in Abbildung 1.2 dargestellt. Diskutiert werden diese Ergebnisse anschließend in Kapitel 6, in dem wir auch einen Ausblick für weitere Untersuchungen geben werden.

2 Grundlagen und Notationen

2.1 Notationen

In dieser Arbeit betrachten wir ausschließlich Lipschitz-Gebiete $\Omega \subseteq \mathbb{R}^3$ mit polygonalem Rand $\partial\Omega$. Sei \mathcal{T} eine reguläre Triangulierung aus abgeschlossenen dreidimensionalen Simplizes $T \in \mathcal{T}$ (siehe [Ste08]). Die Netzweite der Triangulierung wird im Allgemeinen mit h bezeichnet. Insbesondere sei h_T der Durchmesser eines Simplex, d.h. die Länge der längsten Kante des Simplex $T \in \mathcal{T}$ und h_F die Breite einer Fläche, d.h. die Länge der längsten Kante der Fläche $F \subseteq T$.

Den Schwerpunkt von $T \in \mathcal{T}$ bezeichnen wir mit $\text{mid}(T)$. Sei weiterhin \mathcal{N} die Menge der Knoten der Triangulierung, wobei $\mathcal{N}(\Omega)$ die Menge der inneren Knoten und $\mathcal{N}(\partial\Omega)$ die Menge der Randknoten bezeichnet. Analog definieren wir die Mengen \mathcal{E} , $\mathcal{E}(\Omega)$ und $\mathcal{E}(\partial\Omega)$ für die Kanten sowie \mathcal{F} , $\mathcal{F}(\Omega)$ und $\mathcal{F}(\partial\Omega)$ für die Flächen der Triangulierung. Die Menge der Knoten eines Simplex $T \in \mathcal{T}$ bezeichnen wir mit $\mathcal{N}(T)$ und analog definieren wir wieder $\mathcal{E}(T)$ sowie $\mathcal{F}(T)$ für die Kanten bzw. Flächen des Simplex. Mit ν_T und ν_F bezeichnen wir die Normaleneinheitsvektoren auf dem Rand von $T \in \mathcal{T}$ bzw. auf $F \in \mathcal{F}$. Zwei benachbarte Simplizes, d.h. zwei Simplizes, die sich eine gemeinsame Fläche $F \in \mathcal{F}(\Omega)$ teilen, bezeichnen wir mit T_+ und T_- entsprechend der Orientierung von ν_F . Dabei gilt stets $\nu_F \cdot \nu_{T_+} = 1$ und $\nu_F \cdot \nu_{T_-} = -1$. Wir bezeichnen weiterhin mit $a \cdot b$ das euklidische Skalarprodukt zwischen zwei Vektoren $a, b \in \mathbb{R}^n$. Der Raum der $(m \times n)$ Matrizen sei im folgenden mit $\mathbb{R}^{m \times n}$ bezeichnet. Der Sprung entlang einer inneren Fläche $F = T_+ \cap T_- \in \mathcal{F}(\Omega)$ sei $[\cdot]_F := \cdot|_{T_+} - \cdot|_{T_-}$.

Das Maß $|\cdot|$ steht je nach Argument für den Betrag einer Zahl, der euklidischen Länge eines Vektors, der Länge einer Kante, den Flächeninhalt einer Fläche oder das Volumen eines Simplex. Wir werden die für Sobolev-Räume übliche Notation $H^k(\Omega)$ und $H_0^k(\Omega)$ verwenden. Normen werden wir wie üblich mit den entsprechenden Indizes für den Raum abkürzen. Dabei bezeichnen wir gelegentlich die Energienorm einer Funktion f über das ganze Gebiet Ω mit $\|f\| = \|\nabla f\|_{L^2(\Omega)}$. Wir führen zusätzlich den Raum der stückweise H^1 -Funktionen

$$H^1(\mathcal{T}) := \left\{ v \in L^2(\Omega) \mid \forall T \in \mathcal{T}, v|_T \in H^1(T) \right\}$$

ein und werden diese im Falle von vektorwertigen Funktionenräumen entsprechend durch $H^k(\mathcal{T}; \mathbb{R}^m)$ kennzeichnen. Wir definieren weiterhin den stückweise Gradienten $\nabla_{\text{NC}}: H^1(\mathcal{T}) \rightarrow L^2(\Omega; \mathbb{R}^3)$ durch

$$(\nabla_{\text{NC}} v)|_T := \nabla v|_T.$$

Außerdem bezeichnen wir mit

$$\|f\|_{NC} = \left(\sum_{T \in \mathcal{T}} \int_T (\nabla f)^2 dx \right)^{1/2}$$

die nicht-konforme Energienorm einer Funktion $f \in H^1(\mathcal{T})$. Wir definieren den Raum

$$H(\text{div}, \Omega) := \left\{ u \in L^2(\Omega; \mathbb{R}^3) \mid \text{div } u \in L^2(\Omega) \text{ im schwachen Sinne} \right\},$$

und versehen ihn mit der Norm $\|u\|_{H(\text{div}, \Omega)} := \|u\|_{L^2(\Omega)} + \|\text{div } u\|_{L^2(\Omega)}$. Des weiteren benötigen wir den Raum der stückweise Polynome von Grad kleiner oder gleich $k \in \mathbb{N}_0$

$$P_k(\mathcal{T}; \mathbb{R}^m) := \left\{ p \in L^2(\Omega; \mathbb{R}^m) \mid \forall j = 1, \dots, m \forall T \in \mathcal{T}, \ p_j|_T \in P_k(T) \right\},$$

wobei wie üblich $P_k(\mathcal{T}) := P_k(\mathcal{T}; \mathbb{R})$ gilt. Weiterhin bezeichnen wir mit $f_{\mathcal{T}}$ die orthogonale L^2 -Projektion einer Funktion f auf $P_0(\mathcal{T})$.

2.2 Das Poisson-Problem

Die klassische Formulierung des Poisson-Problems haben wir bereits in der Einleitung in (1.1) betrachtet. Zur Vereinfachung gehen wir in diesem Abschnitt von reinen Dirichlet-Nullranddaten $u_D = 0$ aus. Wir erhalten die klassische Formulierung des Poisson-Problems in der folgenden Form.

Problem 1. (klassische Formulierung) Sei $\Omega \subseteq \mathbb{R}^3$ ein einfach zusammenhängendes, polygonal berandetes Lipschitz-Gebiet und sei $f \in L^2(\Omega)$. Finde $u \in C_0(\Omega) \cap C^2(\Omega)$, so dass

$$-\Delta u = f \text{ in } \Omega, \tag{2.1}$$

$$u = 0 \text{ auf } \partial\Omega. \tag{2.2}$$

Dabei sind die Regularitätsanforderungen an u jedoch sehr hoch. Bekanntermaßen überführt man daher das starke Problem in eine schwache Formulierung [Eva10], die weniger Regularität an u fordert. Dafür definieren wir uns die Bilinearform

$$a(u, v) := \int_{\Omega} \nabla u \cdot \nabla v dx$$

sowie das Funktional

$$F(v) := \int_{\Omega} fv dx$$

und erhalten durch eine partielle Integration

Problem 2. (schwache Formulierung) Sei $\Omega \subseteq \mathbb{R}^3$ ein einfach zusammenhängendes, polygonal berandetes Lipschitz-Gebiet und sei $f \in L^2(\Omega)$. Finde $u \in V := H_0^1(\Omega)$, so dass für alle $v \in V$ gilt

$$a(u, v) = F(v). \quad (2.3)$$

Die Existenz und Eindeutigkeit der Lösung u folgen in diesem Fall direkt aus dem Darstellungssatz von Riesz und der Definitheit von a . Um die gemischten Finite-Elemente-Methoden anwenden zu können, wird in der starken Formulierung des Poisson-Problems eine Hilfsvariable p für den Fluss ∇u eingeführt und man erhält ein System erster Ordnung.

Problem 3. (System erster Ordnung) Sei $\Omega \subseteq \mathbb{R}^3$ ein einfach zusammenhängendes, polygonal berandetes Lipschitz-Gebiet und sei $f \in L^2(\Omega)$. Finde $u \in H^1(\Omega)$ und $p \in H(\text{div}, \Omega)$, so dass

$$-\operatorname{div} p = f \text{ in } \Omega, \quad (2.4)$$

$$p - \nabla u = 0 \text{ in } \Omega, \quad (2.5)$$

$$u = 0 \text{ auf } \partial\Omega.$$

Auch hier überführt man das System wieder durch partielle Integration in eine schwache Form und erhält auf diese Weise das folgende Problem.

Problem 4. (schwache Formulierung des Systems erster Ordnung) Sei $\Omega \subseteq \mathbb{R}^3$ ein einfach zusammenhängendes, polygonal berandetes Lipschitz-Gebiet und sei $f \in L^2(\Omega)$. Finde $u \in L^2(\Omega)$ und $p \in H(\text{div}, \Omega)$, so dass für alle $v \in L^2(\Omega)$ und $q \in H(\text{div}, \Omega)$ gilt

$$\int_{\Omega} p \cdot q \, dx + \int_{\Omega} u \operatorname{div} q \, dx = 0 \quad (2.6)$$

$$\int_{\Omega} v \operatorname{div} p \, dx = - \int_{\Omega} f v \, dx. \quad (2.7)$$

Existenz und Eindeutigkeit der Lösung folgen aus der Theorie der Sattelpunktprobleme. Da der Nachweis sehr aufwändig ist, sei an dieser Stelle auf [Bra07] verwiesen.

3 Finite-Elemente-Methoden

3.1 Die P_1 -konforme Finite-Elemente-Methode

Die schwache Formulierung des Poisson-Problems (Problem 2) verlangt $u \in V = H_0^1(\Omega)$. Bei einer konformen Wahl des Diskretisierungsraumes nähern wir den unendlich-dimensionalen Raum V durch einen endlich-dimensionalen Teilraum $V_C \subseteq V$ an und suchen dann $u_C \in V_C$, so dass

$$\int_{\Omega} \nabla u_C \cdot \nabla v_C \, dx = \int_{\Omega} f v_C \, dx$$

für alle $v_C \in V_C$. Diese Methode wird auch Galerkin-Diskretisierung genannt. Wir wollen das Poisson-Problem mit Finite-Elemente-Methoden möglichst geringer Ordnung lösen und definieren uns dafür den Ansatzraum

$$V_C = V_C(\mathcal{T}) := P_1(\mathcal{T}) \cap C_0(\Omega).$$

Dieser Raum besteht aus stückweise Polynomen ersten Grades, die global stetig und Null auf dem Rand $\partial\Omega$ sind. Damit erhalten wir die folgende diskrete Problemstellung.

Problem 5. (diskrete P_1 -konforme Formulierung) Sei $\Omega \subseteq \mathbb{R}^3$ ein einfach zusammenhängendes, polygonal berandetes Lipschitz-Gebiet und sei \mathcal{T} eine reguläre Triangulierung von Ω . Sei weiterhin $f \in L^2(\Omega)$ gegeben. Finde $u_C \in V_C(\mathcal{T})$, so dass für alle $v_C \in V_C(\mathcal{T})$ gilt

$$a(u_C, v_C) = F(v_C). \quad (3.1)$$

Wir erhalten die Existenz und Eindeutigkeit einer Lösung u_C durch den Darstellungssatz von Riesz, da unser Ansatzraum $V_C(\mathcal{T})$ als Teilraum des Hilbertraums $V = H_0^1(\Omega)$ wieder ein Hilbertraum ist. Für die numerische Berechnung ist es wichtig, eine geeignete Basis für $V_C(\mathcal{T})$ zu finden. Um dies zu tun, definieren wir uns die nodalen Basisfunktionen wie folgt.

Definition 3.1. (nodale Basisfunktionen) Sei $\Omega \subseteq \mathbb{R}^3$ ein Lipschitz-Gebiet und \mathcal{T} eine reguläre Triangulierung von Ω . Zu einem Knoten $P \in \mathcal{N}$ definieren wir die *nodale Basisfunktion* $\varphi_P \in P_1(\mathcal{T}) \cap C(\Omega)$ durch die Knotenwerte

$$\varphi_P(x) := \begin{cases} 1 & , \text{ falls } x = P \\ 0 & , \text{ falls } x \in \mathcal{N} \setminus \{P\}, \end{cases}$$

gefolgt von einer linearen Interpolation dieser Werte auf jedem Simplex $T \in \mathcal{T}$, um $\varphi_P \in P_1(\mathcal{T})$ sicherzustellen.

Wir wollen nun zeigen, dass die Basisfunktionen φ_P tatsächlich eine Basis von $V_C(\mathcal{T})$ bilden.

Lemma 3.2. *Sei \mathcal{T} eine reguläre Triangulierung von $\Omega \subseteq \mathbb{R}^3$. Dann bildet*

$$\{\varphi_P \mid P \in \mathcal{N}(\Omega)\}$$

eine Basis von $V_C(\mathcal{T})$.

Beweis. Die lineare Unabhängigkeit der nodalen Basisfunktionen folgt direkt aus ihrer Konstruktion. Es bleibt zu zeigen, dass

$$\dim V_C(\mathcal{T}) = |\mathcal{N}(\Omega)|.$$

Zuerst stellen wir fest, dass $\dim P_1(T) = 4$ für ein $T \in \mathcal{T}$ und folglich $\dim P_1(\mathcal{T}) = 4|\mathcal{T}| = |\mathcal{N}| + 3|\mathcal{F}(\Omega)|$. Die letzte Gleichheit kommt zustande, da die Knoten aller Innenkanten genau zweimal gezählt werden. Aufgrund der an $V_C(\mathcal{T})$ zusätzlich gestellten Bedingung der globalen Stetigkeit müssen die Werte zweier benachbarter Simplizes in den Knoten der inneren Kanten gleich gewählt werden, wodurch wir

$$\dim (P_1(\mathcal{T}) \cap C(\Omega)) = 4|\mathcal{T}| - 3|\mathcal{F}(\Omega)| = |\mathcal{N}|$$

erhalten. Legt man nun noch die Randbedingungen fest, folgt $\dim V_C(\mathcal{T}) = |\mathcal{N}(\Omega)|$. \square

Da die nodalen Basisfunktionen der inneren Knoten tatsächlich eine Basis für unseren Ansatzraum $V_C(\mathcal{T})$ bilden, lässt sich die Lösung u_C von Problem 5 in dieser darstellen. Seien dafür die inneren Knoten mit $P_1, \dots, P_N \in \mathcal{N}(\Omega)$ bezeichnet. Wir schreiben verkürzt $\varphi_j = \varphi_{P_j}$ für $j = 1, \dots, N$ und erhalten für u_C die Basisdarstellung

$$u_C = u_1 \varphi_1 + \dots + u_N \varphi_N$$

mit unbekannten Koeffizienten u_1, \dots, u_N . Wir bemerken, dass es genügt (3.1) auf einer Basis von $V_C(\mathcal{T})$ zu testen und erhalten somit

$$F(\varphi_k) = a(u_C, \varphi_k) = \sum_{j=1}^N u_j a(\varphi_j, \varphi_k)$$

für alle $k = 1, \dots, N$. Wir definieren die Steifigkeitsmatrix $A \in \mathbb{R}^{N \times N}$ und die rechte Seite $b \in \mathbb{R}^N$ für $j, k = 1, \dots, N$ durch

$$\begin{aligned} b_k &:= F(\varphi_k) = \int_{\Omega} f \varphi_k \, dx, \\ A_{jk} &:= a(\varphi_j, \varphi_k) = \int_{\Omega} \nabla \varphi_j \cdot \nabla \varphi_k \, dx, \end{aligned}$$

und erhalten die Lösung u_C aus Problem 5 durch lösen des linearen Gleichungssystems

$$Au = b \quad (3.2)$$

für den Koeffizientenvektor $u = (u_1, \dots, u_N)^\top$.

3.2 Die CR-nicht-konforme Finite-Elemente-Methode

Eine weitere Möglichkeit Problem 2 zu diskretisieren geht auf Crouzeix und Raviart zurück und reduziert die Stetigkeitsanforderungen an die diskrete Lösung u_{NC} indem sie nur noch Stetigkeit in den Flächenmittelpunkten fordert. Da wir keine globale Stetigkeit von unserem Ansatzraum V_{NC} fordern, erhalten wir mit dem Sobolevschen Einbettungssatz, dass V_{NC} keine Teilmenge des eigentlichen Lösungsraumes V aus Problem 2 ist. Damit erhalten wir eine nicht-konforme Methode. Wir definieren uns zuerst einen geeigneten Ansatzraum für unsere Methode. Dafür bezeichnen wir für eine Triangulierung \mathcal{T} mit $\text{mid}(\mathcal{F}) := \{\text{mid}(F) \mid F \in \mathcal{F}\}$ die Menge aller Flächenmittelpunkte und definieren

$$\begin{aligned} \text{CR}^1(\mathcal{T}) &:= \left\{ v \in P_1(\mathcal{T}) \mid v \text{ stetig in } \text{mid}(\mathcal{F}) \right\}, \\ V_{NC}(\mathcal{T}) &:= \text{CR}_0^1(\mathcal{T}) := \left\{ v \in \text{CR}^1(\mathcal{T}) \mid v = 0 \text{ in } \text{mid}(\mathcal{F}) \cap \partial\Omega \right\}. \end{aligned}$$

Unser Ansatzraum besteht also aus stückweise Polynomen ersten Grades, die nur an den Flächenmittelpunkten stetig und an den Mittelpunkten der Randflächen Null sind. Damit erhalten wir die folgende Problemstellung.

Problem 6. (diskrete CR-nicht-konforme Formulierung) Sei $\Omega \subseteq \mathbb{R}^3$ ein einfach zusammenhängendes, polygonal berandetes Lipschitz-Gebiet und sei \mathcal{T} eine reguläre Triangulierung von Ω . Sei weiterhin $f \in L^2(\Omega)$ gegeben. Wir definieren die Bilinearform $a_{NC}: \text{CR}_0^1(\mathcal{T}) \times \text{CR}_0^1(\mathcal{T}) \rightarrow \mathbb{R}$ durch

$$a_{NC}(v_{CR}, w_{CR}) := \int_{\Omega} \nabla_{NC} v_{CR} \cdot \nabla_{NC} w_{CR} dx,$$

wobei wir mit ∇_{NC} den nicht-konformen Gradienten bezeichnen. Finde $u_{CR} \in V_{NC}(\mathcal{T})$, sodass für alle $v_{CR} \in V_{NC}(\mathcal{T})$ gilt

$$a_{NC}(u_{CR}, v_{CR}) = F(v_{CR}). \quad (3.3)$$

Die Existenz und Eindeutigkeit der Lösung u_{CR} folgen wieder aus dem Rieszschen Darstellungssatz. Wir bemerken an dieser Stelle nur, dass der Nachweis komplizierter als im konformen Fall ist, da erst eine diskrete Friedrichs-Ungleichung bewiesen werden muss. Da der Nachweis den Rahmen dieser Arbeit übersteigen würde, verzichten wir an dieser Stelle darauf. Analog zum konformen Fall wollen wir wieder eine Basis unseres Ansatzraumes bestimmen, um unser diskretes Problem zu lösen.

Definition 3.3. (nicht-konforme nodale Basisfunktionen) Sei \mathcal{T} eine reguläre Triangulierung des Lipschitz-Gebietes $\Omega \subseteq \mathbb{R}^3$. Wir definieren die *nicht-konforme nodale Basisfunktion* $\psi_F \in \text{CR}^1(\mathcal{T})$ für jede Fläche $F \in \mathcal{F}$ durch

$$\psi_F(\text{mid}(G)) := \begin{cases} 1 & , \text{ falls } G = F, \\ 0 & , \text{ falls } G \in \mathcal{F} \setminus F. \end{cases}$$

Lemma 3.4. Sei \mathcal{T} eine reguläre Triangulierung von $\Omega \subseteq \mathbb{R}^3$. Dann bildet

$$\{\psi_F \mid F \in \mathcal{F}(\Omega)\}$$

eine Basis von $V_{\text{NC}}(\mathcal{T}) = \text{CR}_0^1(\mathcal{T})$.

Beweis. Wir bemerken, dass $4|\mathcal{T}| - |\mathcal{F}(\Omega)| = |\mathcal{F}|$ und verweisen auf den Beweis von Lemma 3.2. \square

Wir halten zunächst eine wichtige Eigenschaft der nicht-konformen nodalen Basisfunktionen fest, die uns später von Nutzen sein wird.

Lemma 3.5. Sei $T = \text{conv}\{P_1, P_2, P_3, P_4\} \in \mathcal{T}$ für Eckpunkte $P_1 = P_5, P_2 = P_6, P_3 = P_7$ und P_4 . Wir bezeichnen mit $F_j = \text{conv}\{P_{j+1}, P_{j+2}, P_{j+3}\}$, $j = 1, \dots, 4$, die den Eckpunkten gegenüber liegenden Flächen. Dann gilt für $j = 1, \dots, 4$

$$\psi_{F_j}|_T = 1 - 3\varphi_{P_j}|_T.$$

Beweis. Sei $Q_j := \text{mid}(F_j)$ für $j = 1, \dots, 4$. Da die φ_{P_j} affine Funktionen auf T sind, gilt $\varphi_{P_j}|_T(Q_j) = 0$ sowie $\varphi_{P_j}|_T(Q_k) = 1/3$ für $k \neq j$. Wir erhalten also $1 - 3\varphi_{P_j}|_T(Q_j) = 1 = \psi_{F_j}|_T(Q_j)$ und $1 - 3\varphi_{P_j}|_T(Q_k) = 0 = \psi_{F_j}|_T(Q_k)$. \square

Wir können nun wieder unsere Lösung u_{NC} als Linearkombination der nicht-konformen nodalen Basisfunktionen darstellen. Wir nummerieren die Flächen der Triangulierung mit $T_1, \dots, T_M \in \mathcal{F}(\Omega)$ und erhalten

$$u_{\text{NC}} = u_1\psi_1 + \dots + u_M\psi_M$$

mit unbekannten Koeffizienten u_1, \dots, u_M und $\psi_j := \psi_{F_j}$. Wir testen (3.3) mit unseren nicht-konformen Basisfunktionen und müssen nun

$$F(\psi_k) = a_{\text{NC}}(u_{\text{NC}}, \psi_k) = \sum_{j=1}^M u_j a_{\text{NC}}(\psi_j, \psi_k)$$

für alle $k = 1, \dots, M$ lösen. Wir definieren die Steifigkeitsmatrix $A \in \mathbb{R}^{M \times M}$ und die rechte Seite $b \in \mathbb{R}^M$ durch

$$b_k := F(\psi_k) = \int_{\Omega} f \psi_k \, dx,$$

$$A_{jk} := a_{\text{NC}}(\psi_j, \psi_k) = \int_{\Omega} \nabla \psi_j \cdot \nabla \psi_k \, dx.$$

Dann erhalten wir die Lösung u_{NC} aus Problem 6 durch Lösen des linearen Gleichungssystems

$$Au = b \quad (3.4)$$

für den Koeffizientenvektor $u = (u_1, \dots, u_M)^{\top}$.

3.3 Die RT₀-gemischte Finite-Elemente-Methode

3.3.1 Diskretisierung

Bisher haben wir das Poisson-Problem als System zweiter Ordnung betrachtet und diskretisiert. Wie bereits in Kapitel 2.2 vorgestellt, können wir diese Gleichung zweiter Ordnung in zwei Gleichungen aufspalten und erhalten ein System erster Ordnung wie in Problem 3. Unser Ziel ist es, Problem 4 zu diskretisieren, indem wir eine auf Raviart und Thomas zurückgehende, $H(\text{div}, \Omega)$ -konforme, gemischte Finite-Elemente-Methode anwenden. Dazu definieren wir uns den Raviart-Thomas Finite-Elemente-Raum

$$\text{RT}_0(\mathcal{T}) := \left\{ q \in H^1(\mathcal{T}) \cap H(\text{div}, \Omega) \mid \begin{array}{l} \forall T \in \mathcal{T} \exists a, b, c, d \in \mathbb{R} \forall x \in T, \\ q(x) = (a, b, c) + dx \end{array} \right\}$$

als Ansatzraum für p . Wählen wir zusätzlich $P_0(\mathcal{T})$ als Ansatzraum für u , so erhalten wir das folgende diskrete Problem.

Problem 7. (diskrete Formulierung des Systems erster Ordnung) Sei $\Omega \subseteq \mathbb{R}^3$ ein einfach zusammenhängendes, polygonal berandetes Lipschitz-Gebiet und sei $f \in L^2(\Omega)$. Finde $u_{\text{RT}} \in P_0(\mathcal{T})$ und $p_{\text{RT}} \in \text{RT}_0(\mathcal{T})$, sodass für alle $v_{\text{RT}} \in P_0(\mathcal{T})$ und $q_{\text{RT}} \in \text{RT}_0(\mathcal{T})$ gilt

$$\int_{\Omega} p_{\text{RT}} \cdot q_{\text{RT}} \, dx + \int_{\Omega} u_{\text{RT}} \text{div } q_{\text{RT}} \, dx = 0 \quad (3.5)$$

$$\int_{\Omega} v_{\text{RT}} \text{div } p_{\text{RT}} \, dx = - \int_{\Omega} f v_{\text{RT}} \, dx. \quad (3.6)$$

Auf die Existenz und Eindeutigkeit der Lösung $(p_{\text{RT}}, u_{\text{RT}})$ werden wir zu einem späteren Zeitpunkt eingehen, da diese etwas mit der Beziehung zu der nicht-konformen Finite-Elemente-Methode nach Crouzeix und Raviart zu tun hat.

Definition 3.6. (globale Definition der flächenorientierten Basisfunktionen) Sei $F \in \mathcal{F}$ eine Fläche in \mathcal{T} . Dann existieren entweder zwei Elemente T_+ und T_- in \mathcal{T} , die sich die Fläche $F = \partial T_+ \cap \partial T_-$ teilen oder es existiert genau ein Element T_+ in \mathcal{T} mit $F \subset \partial T_+ \cap \partial \Omega$. Sei $T_\pm = \text{conv}\{F \cup P_\pm\}$, wobei P_\pm den gegenüberliegenden Punkt von F in T_\pm bezeichnet. Wir definieren

$$\Psi_F(x) := \begin{cases} \pm \frac{|F|}{3|T_\pm|}(x - P_\pm) & , \text{ falls } x \in T_\pm, \\ 0 & , \text{ sonst.} \end{cases}$$

Wir halten ein paar Eigenschaften der Ψ_F fest.

Lemma 3.7. Es gilt

- (a) $\Psi_F \cdot \nu_G = \begin{cases} 1 & , \text{ falls } G = F, \\ 0 & , \text{ falls } G \in \mathcal{F} \setminus \{F\}; \end{cases}$
- (b) $\Psi_F \in H(\text{div}, \Omega)$;
- (c) $\text{span}\{\Psi_F | F \in \mathcal{F}\}$ ist eine Basis von $\text{RT}_0(\mathcal{T})$;
- (d) $\text{div } \Psi_F = \begin{cases} \pm |F| / |T_\pm| & , \text{ auf } T_\pm, \\ 0 & , \text{ sonst.} \end{cases}$

Beweis. Der Beweis verläuft analog zu dem in [BC05, Lemma 4.1] für zwei Raumdimensionen gezeigten Beweis. \square

Da die Ψ_F eine Basis von $\text{RT}_0(\mathcal{T})$ bilden, können wir, zusammen mit der Basis $\Phi_T \in P_0(\mathcal{T})$ mit

$$\Phi_T(x) = \begin{cases} 1 & , \text{ falls } x \in T, \\ 0 & , \text{ sonst,} \end{cases}$$

unsere Lösung $(p_{\text{RT}}, u_{\text{RT}})$ in dieser Basis darstellen. Wir setzen $J = |\mathcal{F}|$ sowie $K = |\mathcal{T}|$ und erhalten mit der vereinfachten Darstellung $j = F_j$ und $k = T_k$ für $j = 1, \dots, J$, $k = 1, \dots, K$

$$p_{\text{RT}} = \sum_{j=1}^J p_j \Psi_j \text{ und } u_{\text{RT}} = \sum_{k=1}^K u_k \Phi_k$$

mit unbekannten Koeffizienten p_1, \dots, p_J und u_1, \dots, u_K . Setzen wir diese Darstellung von $(p_{\text{RT}}, u_{\text{RT}})$ in die Gleichungen (3.5) – (3.6) ein und testen nur auf den Basen Ψ_j bzw. Φ_k ,

erhalten wir das lineare Gleichungssystem

$$\begin{pmatrix} A & B \\ B^\top & 0 \end{pmatrix} \begin{pmatrix} p \\ u \end{pmatrix} = \begin{pmatrix} 0 \\ b \end{pmatrix} \quad (3.7)$$

für die Koeffizientenvektoren $p = (p_1, \dots, p_J)^\top$ und $u = (u_1, \dots, u_K)^\top$. Dabei sind $A \in \mathbb{R}^{J \times J}$, $B \in \mathbb{R}^{J \times K}$ und $b \in \mathbb{R}^K$ definiert durch

$$\begin{aligned} A_{j\ell} &:= \int_{\Omega} \Psi_j \cdot \Psi_\ell \, dx, \\ B_{jk} &:= \int_{\Omega} \Phi_k \operatorname{div} \Psi_j \, dx, \\ b_k &:= \int_{\Omega} f \Phi_k \, dx. \end{aligned}$$

3.3.2 RT₀-MFEM mit Lagrange-Multiplikatoren

Wir wollen uns nun eine weitere Möglichkeit der Diskretisierung von Problem 4 ansehen, die mit sogenannten Lagrange-Multiplikatoren arbeitet. Da wir diese Methode später auch implementieren wollen, werden wir in diesem Kapitel von allgemeinen Dirichlet sowie Neumann-Randdaten u_D bzw. g ausgehen. Daher müssen wir zunächst diese allgemeineren Randbedingungen in den uns bereits bekannten Ansatzräumen verarbeiten. Sei dafür

$$H_{0,N}(\operatorname{div}, \Omega) := \{q \in H(\operatorname{div}, \Omega) \mid q \cdot \nu = 0 \text{ auf } \Gamma_N\}.$$

Wir definieren unsere diskreten Räume durch

$$\begin{aligned} M(\mathcal{T}) &:= \operatorname{RT}_0(\mathcal{T}) \cap H_{0,N}(\operatorname{div}, \Omega), \\ L(\mathcal{T}) &:= P_0(\mathcal{T}) \end{aligned}$$

und führen zusätzlich die beiden Räume

$$\begin{aligned} \Lambda(\mathcal{T}) &:= P_0(\mathcal{F}(\Omega); \mathbb{R}^3) := \left\{ \lambda \in L^\infty(\cup \mathcal{F}) \mid \forall F \in \mathcal{F}, \lambda|_F := \lambda_F \in \mathbb{R}^3 \text{ und } \lambda_{\partial\Omega} \equiv 0 \right\}, \\ N(\mathcal{T}) &:= P_0(\mathcal{F}(\Gamma_N); \mathbb{R}^3) \end{aligned}$$

ein. Wir erhalten damit eine alternative Formulierung von Problem 7.

Problem 8. (modifizierte diskrete Formulierung des Systems erster Ordnung) Sei $\Omega \subseteq \mathbb{R}^3$ ein einfach zusammenhängendes, polygonal berandetes Lipschitz-Gebiet und sei $f \in L^2(\Omega)$. Finde $(u_{\operatorname{RT}}, p_{\operatorname{RT}}, \lambda_{\operatorname{RT}}, \ell_{\operatorname{RT}}) \in L(\mathcal{T}) \times M(\mathcal{T}) \times \Lambda(\mathcal{T}) \times N(\mathcal{T})$, sodass für alle

$(v_{\text{RT}}, q_{\text{RT}}, \mu_{\text{RT}}, m_{\text{RT}}) \in L(\mathcal{T}) \times M(\mathcal{T}) \times \Lambda(\mathcal{T}) \times N(\mathcal{T})$ gilt

$$\begin{aligned} \int_{\Gamma_D} u_d q_{\text{RT}} \cdot \nu \, ds &= \int_{\Omega} p_{\text{RT}} \cdot q_{\text{RT}} \, dx + \int_{\Omega} u_{\text{RT}} \operatorname{div} q_{\text{RT}} \, dx \\ &\quad - \sum_{F \in \cup \mathcal{F}} \int_F [q_{\text{RT}} \cdot \nu_F] \lambda_{\text{RT}} \, ds - \int_{\Gamma_N} (q_{\text{RT}} \cdot \nu) \ell_{\text{RT}} \, ds, \end{aligned} \quad (3.8)$$

$$-\int_{\Omega} v_{\text{RT}} f \, dx = \int_{\Omega} v_{\text{RT}} \operatorname{div} p_{\text{RT}} \, dx, \quad (3.9)$$

$$0 = - \sum_{F \in \cup \mathcal{F}} \int_F [p_{\text{RT}} \cdot \nu_F] \mu_{\text{RT}} \, ds, \quad (3.10)$$

$$-\int_{\Gamma_N} g m_{\text{RT}} \, ds = - \int_{\Gamma_N} [p_{\text{RT}} \cdot \nu] m_{\text{RT}} \, ds. \quad (3.11)$$

Wir werden in Kapitel 4 näher auf die Wahl einer Basis und damit auf die Darstellung von Problem 8 als lineares Gleichungssystem eingehen. Um hier schon einmal einen Einblick in die Struktur dieses Gleichungssystems zu geben, definieren wir die Blöcke B, C, D und E durch

$$\begin{aligned} B_{jk} &:= \left(\int_{\Omega} \Psi_j \cdot \Psi_k \, dx \right)_{jk}, \quad j, k = 1, \dots, 4|\mathcal{T}|, \\ C_{j\ell} &:= \left(\int_{\Omega} \Phi_{\ell} \operatorname{div} \Psi_j \, dx \right)_{j\ell}, \quad \ell = 1, \dots, |\mathcal{T}|, \\ D_{j\alpha} &:= \left(- \int_{F_{\alpha}} \Psi_j \cdot \nu_F \, ds \right)_{j\alpha}, \quad \forall F_{\alpha} \in \mathcal{F}(\Omega), \\ E_{j\beta} &:= \left(- \int_{F_{\beta}} \Psi_j \cdot \nu_F \, ds \right)_{j\beta}, \quad \forall F_{\beta} \in \mathcal{F}(\Gamma_N). \end{aligned}$$

mit Basisfunktionen Ψ_j und Φ_{ℓ} , die in Kapitel 4 eingeführt werden. Damit erhalten wir ein Gleichungssystem der Form

$$\begin{pmatrix} B & C & D & E \\ C^{\top} & 0 & 0 & 0 \\ D^{\top} & 0 & 0 & 0 \\ E^{\top} & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_p \\ x_u \\ x_{\lambda_M} \\ x_{\lambda_N} \end{pmatrix} = \begin{pmatrix} b_D \\ b_f \\ 0 \\ b_g \end{pmatrix} \quad (3.12)$$

mit $B \in \mathbb{R}^{4|\mathcal{T}| \times 4|\mathcal{T}|}$, $C \in \mathbb{R}^{4|\mathcal{T}| \times |\mathcal{T}|}$, $D \in \mathbb{R}^{4|\mathcal{T}| \times |\mathcal{F}(\Omega)|}$, $E \in \mathbb{R}^{4|\mathcal{T}| \times |\mathcal{F}(\Gamma_N)|}$ und $b_D \in \mathbb{R}^{4|\mathcal{T}|}$, $b_f \in \mathbb{R}^{|\mathcal{T}|}$, $b_g \in \mathbb{R}^{|\mathcal{F}(\Gamma_N)|}$.

3.3.3 Zusammenhang zwischen CR-NCFEM und RT₀-MFEM

Satz 3.8. Sei $q \in P_0(\mathcal{T}, \mathbb{R}^3)$ und $\hat{q}(x) := q|_T - f_T/3(x - \operatorname{mid}(T))$. Dann sind die folgenden Aussagen paarweise äquivalent.

- (a) $\int_{\Omega} f_{\mathcal{T}} \psi_F dx = \int_{\Omega} q \cdot \nabla_{NC} \psi_F dx$ für alle $F \in \mathcal{F}(\Omega)$;
- (b) $[\hat{q}]_F \cdot \nu_F = 0$ für alle $F \in \mathcal{F}(\Omega)$;
- (c) $\operatorname{div} \hat{q} + f_{\mathcal{T}} = 0$ im distributionellem Sinne.

Beweis. Der Beweis ist analog zu dem zweidimensionalen Fall. Daher verweisen wir an dieser Stelle nur auf [Mar85]. \square

Wir erhalten durch die Äquivalenz von Satz 3.8.(b) und Satz 3.8.(c), dass für alle $q \in H^1(\mathcal{T}; \mathbb{R}^3)$ die Aussagen $[q]_F \cdot \nu_F = 0$ für alle $F \in \mathcal{F}(\Omega)$ und $q \in H(\operatorname{div}, \Omega)$ äquivalent sind. Dieser Zusammenhang wurde erstmals von Marini in [Mar85] benutzt, um die diskrete Lösung der RT_0 -gemischten Finite-Elemente-Methode mittels der Lösung der CR-nicht-konformen Methode darzustellen. Wir betrachten im folgenden eine dreidimensionale Variante dieser Darstellung.

Satz 3.9. (*Marini-Identität*) Sei $u_{CR} \in CR_0^1(\mathcal{T})$ die Lösung von Problem 6 mit modifizierter rechter Seite $f_{\mathcal{T}} \in P_0(\mathcal{T})$. Dann ist die eindeutige Lösung $(p_{RT}, u_{RT}) \in RT_0(\mathcal{T}) \times P_0(\mathcal{T})$ von Problem 7 gegeben durch

$$p_{RT}(x) = \nabla_{NC} u_{CR} - f_{\mathcal{T}}/3(x - \operatorname{mid}(T)) \text{ für } x \in T \in \mathcal{T}, \quad (3.13)$$

$$u_{RT}|_T = \oint_T u_{CR} dx + s^2(T)f_{\mathcal{T}}/432 \text{ auf } T \in \mathcal{T}, \quad (3.14)$$

wobei $s^2(T) := |P_1 - P_2| + |P_1 - P_3| + |P_1 - P_4| + |P_2 - P_3| + |P_2 - P_4| + |P_3 - P_4|$ für die Eckpunkte P_1, P_2, P_3 und P_4 von T .

Bevor wir Satz 3.9 beweisen, beweisen wir das folgende Lemma.

Lemma 3.10. Für einen Simplex $T \in \mathcal{T}$ mit Schwerpunkt $\operatorname{mid}(T)$ gilt

$$\|x - \operatorname{mid}(T)\|_{L^2(T)}^2 = \frac{s^2(T)|T|}{48}. \quad (3.15)$$

Beweis. (von Lemma 3.10) Sei o.B.d.A. $\operatorname{mid}(T) = 0$ und sei

$$x = \lambda_1 P_1 + \lambda_2 P_2 + \lambda_3 P_3 + \lambda_4 P_4$$

mit baryzentrischen Koordinaten $\lambda_1, \dots, \lambda_4$ von T . Einsetzen in (3.15) liefert

$$\begin{aligned} \int_T |x|^2 dx &= \sum_{j,k=1}^4 P_j P_k \int_T \lambda_j \lambda_k dx \\ &= \sum_{j,k=1}^4 P_j P_k (1 + \delta_{jk}) |T| / 12, \end{aligned}$$

wobei wir bei der letzten Gleichheit verwendet haben, dass für die baryzentrischen Koordinaten

$$\int_T \lambda_j^a \lambda_k^b dx = 2|T| \frac{a!b!}{(a+b+2)!}$$

gilt. Wir bemerken, dass $4 \text{mid}(T) = P_1 + P_2 + P_3 + P_4$ und damit $0 = |P_1 + P_2 + P_3 + P_4|^2 = \sum_{j,k=1}^4 P_j P_k$ und erhalten

$$\int_T |x|^2 dx = \frac{|T|}{12} \sum_{j=1}^4 |P_j|^2.$$

Die gleiche Orthogonalität liefert uns

$$\begin{aligned} \sum_{j,k=1}^4 |P_j - P_k|^2 &= \sum_{j,k=1}^4 (|P_j|^2 + |P_k|^2 - 2P_j P_k) \\ &= \sum_{j,k=1}^4 (|P_j|^2 + |P_k|^2) \\ &= 8 \sum_{j=1}^4 |P_j|^2. \end{aligned}$$

Die linke Seite dieser Gleichung besteht aus den quadrierten Längen aller Kanten von T , was nichts anderes bedeutet als $\sum_{j,k=1}^4 |P_j - P_k|^2 = 2 s^2(T)$. Mit der vorherigen Abschätzung erhalten wir also

$$\int_T |x|^2 dx = \frac{|T|}{12} \sum_{j=1}^4 |P_j|^2 = \frac{|T|}{48} s^2(T).$$

□

Beweis. (von Satz 3.9) Satz 3.8 impliziert, dass p_{RT} aus (3.13) die Gleichung (3.6) erfüllt. Für (3.5) erhalten wir für alle $q_{\text{RT}} \in \text{RT}_0(\mathcal{T})$ mit

$$q_{\text{RT}}|_T := (a, b, c) + d/3 (x - \text{mid}(T))$$

für $x \in T \in \mathcal{T}$, dass

$$\begin{aligned}
\int_T p_{\text{RT}} \cdot q_{\text{RT}} + u_{\text{RT}} \operatorname{div} q_{\text{RT}} dx &= \int_T \nabla_{\text{NC}} u_{\text{CR}} \cdot q_{\text{RT}} - d f_T / 9 |x - \text{mid}(T)|^2 + d u_{\text{RT}} dx \\
&= \int_{\partial T} u_{\text{CR}} q_{\text{RT}} \cdot \nu_T ds - \int_T u_{\text{CR}} \operatorname{div} q_{\text{RT}} dx \\
&\quad + \int_T d u_{\text{RT}} - d f_T / 9 |x - \text{mid}(T)|^2 dx \\
&= \int_{\partial T} u_{\text{CR}} q_{\text{RT}} \cdot \nu_T ds + d \int_T u_{\text{RT}} - u_{\text{CR}} - s^2(T) f_T / 432 dx \\
&= \int_{\partial T} u_{\text{CR}} q_{\text{RT}} \cdot \nu_T ds,
\end{aligned}$$

indem wir Lemma 3.10 anwenden und partiell integrieren. Da $[q_{\text{RT}}]_F \cdot \nu_F = 0$ für $f \in \mathcal{F}(\Omega)$ und $\int_F [u_{\text{CR}}]_F ds = 0$ für $F \in \mathcal{F}(\partial\Omega)$, erhalten wir durch Summieren über alle $T \in \mathcal{T}$

$$\int_{\Omega} p_{\text{RT}} \cdot q_{\text{RT}} + u_{\text{RT}} \operatorname{div} q_{\text{RT}} dx = 0.$$

Damit erfüllen p_{RT} und u_{RT} die Gleichung (3.5). \square

Wir haben gezeigt, dass sich $(p_{\text{RT}}, u_{\text{RT}})$ durch die Lösung u_{CR} aus Problem 6 darstellen lässt und erhalten somit die Möglichkeit, p_{RT} und u_{RT} zu berechnen ohne das Gleichungssystem (3.7) bzw. (3.12) zu lösen. Weiterhin gibt uns Satz 3.9 zusammen mit einem Eindeutigkeitsargument die Existenz und Eindeutigkeit einer Lösung der RT_0 -gemischten Finite-Elemente-Methode wie in [Car09b, §3.3] beschrieben.

3.4 Fehlerschätzer

3.4.1 P_1 -konformer Fehlerschätzer

In [Car05] wird ein Fehlerschätzer für die konforme Diskretisierung des Poisson-Problems eingeführt, der auf Volumentermen und Normalensprüngen der diskreten Lösung über Flächen basiert. Wir betrachten zunächst den lokalen Fehlerschätzer auf einem Simplex $T \in \mathcal{T}$ gegeben durch

$$\eta^2(T) := h_T^2 \|f\|_{L^2(T)}^2 + h_T \sum_{F \in \mathcal{F}(T) \setminus \mathcal{F}(\Gamma_D)} \|[\nabla u_C]_F \cdot \nu_F\|_{L^2(F)}^2. \quad (3.16)$$

In dem wir die lokalen Beiträge aus (3.16) über alle Elemente $T \in \mathcal{T}$ aufsummieren, erhalten wir einen Fehlerschätzer für die gesamte Triangulierung

$$\eta(\mathcal{T}) := \left(\sum_{T \in \mathcal{T}} \eta^2(T) \right)^{1/2}. \quad (3.17)$$

Die Effizienz und Zuverlässigkeit dieses Fehlerschätzers werden ebenfalls in [Car05] bewiesen, weshalb wir an dieser Stelle auf deren Nachweis verzichten. Wir bemerken weiterhin, dass im Dreidimensionalen der folgende Zusammenhang zwischen den Durchmesser eines Simplex h_T , seinem Volumen und dem Flächeninhalt einer seiner Flächen besteht.

Bemerkung 3.11. Sei $T \in \mathcal{T}$ und $F \in \mathcal{F}(T)$. Dann ist $h_T \approx |T|^{1/3} \approx |F|^{1/2}$.

3.4.2 Crouzeix-Raviart Fehlerschätzer

Für den nicht-konformen Fall orientieren wir uns ebenfalls an dem in [Car05] beschriebenen. Dieser Fehlerschätzer basiert ebenfalls auf Volumentermen, berücksichtigt aber im Gegensatz zu dem konformen Schätzer aus (3.16) Tangential- anstatt Normalensprünge. Da die Tangentialvektoren einer Fläche im Dreidimensionalen eine Ebene bilden, lassen sich die Tangentialsprünge nicht so einfach berechnen. Um den Fehlerschätzer und seine Eigenschaften trotzdem verwenden zu können brauchen wir zunächst folgende Bemerkung.

Bemerkung 3.12. (Tangentialrichtung) Sei $F \in \mathcal{F}$ eine Fläche der Triangulierung \mathcal{T} . Sei ν_F der Normaleneinheitsvektor auf F und seien $\tau_{F,1}$ und $\tau_{F,2}$ zwei tangentiale Einheitsvektoren mit $\tau_{F,1} \cdot \tau_{F,2} = 0$. Dann gilt für alle $x \in F$

$$x = (x \cdot \nu_F)\nu_F + (x \cdot \tau_{F,1})\tau_{F,1} + (x \cdot \tau_{F,2})\tau_{F,2}.$$

Damit lässt sich der Tangentialsprung des Fehlerschätzers durch den Normalensprung ausdrücken, denn es ist

$$\begin{aligned} & \|[\nabla_{NC} u_{CR}]_F \cdot \tau_{F,1}\|_{L^2(F)}^2 + \|[\nabla_{NC} u_{CR}]_F \cdot \tau_{F,2}\|_{L^2(F)}^2 \\ &= \left\| \left([\nabla_{NC} u_{CR}]_F \cdot \tau_{F,1} \right) \tau_{F,1} + \left([\nabla_{NC} u_{CR}]_F \cdot \tau_{F,2} \right) \tau_{F,1} \right\|_{L^2(F)}^2 \\ &= \left\| [\nabla_{NC} u_{CR}]_F + \left([\nabla_{NC} u_{CR}]_F \cdot \nu_F \right) \nu_F \right\|_{L^2(F)}^2. \end{aligned}$$

Wir definieren also

$$\eta^2(T) := h_T^2 \|f\|_{L^2(T)}^2 + h_T \sum_{F \in \mathcal{F}(T) \setminus \mathcal{F}(\Gamma_N)} \left\| [\nabla_{NC} u_{CR}]_F - \left([\nabla_{NC} u_{CR}]_F \cdot \nu_F \right) \nu_F \right\|_{L^2(F)}^2 \quad (3.18)$$

als lokalen Beitrag auf jedem Element $T \in \mathcal{T}$ und summieren wieder über alle Elemente für den globalen Fehlerschätzer

$$\eta(\mathcal{T}) := \left(\sum_{T \in \mathcal{T}} \eta^2(T) \right)^{1/2}. \quad (3.19)$$

Für den Nachweis von Effizienz und Zuverlässigkeit des Fehlerschätzers verweisen wir wieder auf [Car05].

3.4.3 Raviart-Thomas Fehlerschätzer

Der Schätzer für die Raviart-Thomas Finite-Elemente-Methode ist ähnlich zu dem Fehlerschätzer in (3.18) aufgebaut. Er besteht ebenfalls aus Volumentermen und Termen für Tangentialsprünge, die wir wieder mit dem Argument aus Bemerkung 3.12 durch subtrahieren des Normalenanteils ausdrücken können. Da die RT_0 -MFEM eine gemischte Methode ist, erhalten wir durch die Lösung $(p_{\text{RT}}, u_{\text{RT}})$ mit p_{RT} bereits eine Approximation des exakten Flusses ∇u und müssen diese daher nicht zusätzlich, so wie in (3.18), bestimmen. Wir erhalten demnach

$$\eta^2(T) := h_T^2 \|f - f_T\|_{L^2(T)}^2 + h_T \sum_{F \in \mathcal{F}(T)} \left\| [p_{\text{RT}}]_F - ([p_{\text{RT}}]_F \cdot \nu_F) \nu_F \right\|_{L^2(F)}^2 \quad (3.20)$$

als lokalen Fehlerschätzer auf einem Element T unserer Triangulierung und summieren über alle T in \mathcal{T} für den globalen Fehlerschätzer

$$\eta(\mathcal{T}) := \left(\sum_{T \in \mathcal{T}} \eta^2(T) \right)^{1/2}. \quad (3.21)$$

Auch für diesen Fehlerschätzer wurden Effizienz und Zuverlässigkeit bereits in [Car05] bewiesen, weshalb wir an dieser Stelle darauf verzichten.

4 Implementierung

4.1 Berechnung des P₁-konformen Lözers

Wir berechnen zuerst die Steifigkeitsmatrix A aus (3.2). Dafür berechnen wir zunächst die lokale Steifigkeitsmatrix auf einem Element $T = \text{conv}\{P_1, P_2, P_3, P_4\}$.

Bemerkung 4.1. (Eigenschaften der nodalen Basisfunktionen) Sei $T = \text{conv}\{P_1, P_2, P_3, P_4\}$ ein Element aus \mathcal{T} . Dann gilt für die Gradienten der nodalen Basisfunktionen $\varphi_1, \dots, \varphi_4$ auf T

$$Q := \begin{pmatrix} \nabla \varphi_1 \\ \nabla \varphi_2 \\ \nabla \varphi_3 \\ \nabla \varphi_4 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ P_1 & P_2 & P_3 & P_4 \end{pmatrix}^{-1} \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \quad (4.1)$$

Als direkte Konsequenz aus Bemerkung 4.1 erhalten wir die lokale Steifigkeitsmatrix durch

$$A_C(T) = \left(\int_T \nabla \varphi_j \cdot \nabla \varphi_k \, dx \right)_{j,k=1,\dots,4} = |T| Q Q^\top.$$

Wir erhalten die globale Steifigkeitsmatrix A , indem wir die Einträge von $A_C(T)$ an die entsprechenden Stellen von A schreiben. Dabei müssen wir darauf achten, dass die lokale Nummerierung der Knoten in $A_C(T)$ durch die entsprechenden globalen Knotennummern ausgetauscht werden müssen. Dafür definieren wir uns die Indexmenge

$$I(T) := \{(\alpha, k) \in \{1, \dots, 4\} \times \{1, \dots, |\mathcal{N}|\} \mid P_\alpha = z_k\},$$

wobei die z_k die globalen Knotennummern darstellen. Damit können wir die globale Steifigkeitsmatrix durch

$$A = \sum_{T \in \mathcal{T}} \sum_{(\alpha, k) \in I(T)} \sum_{(\beta, \ell) \in I(T)} (A_C(T))_{\alpha, \beta} e_k \otimes e_\ell$$

berechnen, wobei wir mit $e_j \in \mathbb{R}^{|\mathcal{N}|}$ für $j = 1, \dots, |\mathcal{N}|$ die kanonischen Einheitsvektoren bezeichnen. Um die rechte Seite zu berechnen, bemerken wir, dass auf einem Simplex $T = \text{conv}\{P_1, P_2, P_3, P_4\}$ die Mittelpunktsregel

$$\int_T f(x) \varphi_j(x) \, dx \approx f(\text{mid}(T)) |T| / 4$$

gilt und erhalten

$$b_j = \int_\Omega f(x) \varphi_j(x) \, dx = \sum_{T \in \mathcal{T}(z_j)} \int_T f(x) \varphi_j(x) \, dx \approx \sum_{T \in \mathcal{T}(z_j)} f(\text{mid}(T)) |T| / 4,$$

wobei wir mit

$$\mathcal{T}(z_j) := \left\{ T \in \mathcal{T} \mid z_j \text{ ist Knoten von } T \right\}$$

die Menge aller Simplizes, die z_j als Knoten besitzen, bezeichnen. Die folgenden Zeilen Matlab-Code realisieren die elementweise Berechnung der lokalen Steifigkeitsmatrizen sowie der rechten Seite und setzen anschließend die globale Steifigkeitsmatrix aus den lokalen Beiträgen zusammen.

```

for elem = 1 : nrElems
    nodes = n4e(elem ,1:4);           % nodes of the element
    coords = c4n(nodes ,:);          % coordinates of the four nodes
    volume = volume4e(elem);        % volume of the element
    mid = mid4e(elem ,:);           % midpoint of the element
    grads = [1,1,1,1;coords'] \ [0,0,0;eye(3)];
    b(nodes) = b(nodes)+ volume/4 * f(mid)*[1;1;1;1];
    Alocal (:,:,elem) = volume * (grads * grads');
end

% assembly of the global stiffness matrix A
n4eT = n4e (:,1:4)';
I = [n4eT;n4eT;n4eT;n4eT];
J = [n4eT (:),n4eT (:),n4eT (:),n4eT (:)]';
A = sparse(I (:),J (:),Alocal (:));

```

4.2 Berechnung des Crouzeix-Raviart Lösers

Wir erinnern uns an die in Lemma 3.5 beschriebene Beziehung zwischen den nodalen Basisfunktionen und den nicht-konformen Basisfunktionen und erhalten für die lokale Steifigkeitsmatrix $A_{NC}(T)$

$$\begin{aligned}
 A_{NC}(T) &= \left(\int_T \nabla \psi_j \cdot \nabla \psi_k \, dx \right)_{j,k=1,\dots,4} \\
 &= \left(\int_T (-3\nabla \varphi_j) \cdot (-3\nabla \varphi_k) \, dx \right)_{j,k=1,\dots,4} \\
 &= 9 A_C(T).
 \end{aligned}$$

Die globale Steifigkeitsmatrix erhalten wir wieder durch geschicktes Aufsummieren der lokalen Steifigkeitsmatrizen. Auch hier müssen wir wieder die lokalen Flächen F_1, \dots, F_4 auf jedem Simplex T durch die globalen Flächennummern ersetzen. Wir definieren uns die Indexmenge

$$J(T) := \{(\alpha, k) \in \{1, \dots, 4\} \times \{1, \dots, |\mathcal{F}|\} \mid F_\alpha = y_k\},$$

wobei die y_k die globalen Flächennummern darstellen und erhalten

$$A = \sum_{T \in \mathcal{T}} \sum_{(\alpha, k) \in J(T)} \sum_{(\beta, \ell) \in J(T)} (A_{\text{NC}}(T))_{\alpha, \beta} e_k \otimes e_\ell.$$

Dabei bezeichnen wir mit $e_j \in \mathbb{R}^{|\mathcal{F}|}$ für $j = 1, \dots, |\mathcal{F}|$ die kannonischen Einheitsvektoren. Da auf einem Simplex $T \in \mathcal{T}$ stets die Mittelpunktsregel

$$\int_T f(x) \psi_j(x) dx \approx f(\text{mid}(T)) |T| / 4$$

gilt, lässt sich die rechte Seite b durch

$$\begin{aligned} b_j &= \int_{\Omega} f(x) \psi_j(x) dx \\ &= \int_{T_+} f(x) \psi_j(x) \Big|_{T_+} dx + \int_{T_-} f(x) \psi_j(x) \Big|_{T_-} dx \\ &\approx f(\text{mid}(T_+)) |T_+| / 4 + f(\text{mid}(T_-)) |T_-| / 4 \end{aligned}$$

berechnen, wobei wir wie üblich mit T_{\pm} die Nachbarelemente von F , d.h. $F = T_+ \cap T_-$, bezeichnen. Der Matlab-Code für die Berechnung der lokalen und globalen Beiträge lautet

```

for elem = 1 : nrElems
    nodes = n4e(elem, 1:4); % nodes of this element
    faces = f4e(elem, :); % faces of this element
    coords = c4n(nodes, :); % coordinates for the nodes
    volume = volume4e(elem); % volume of this element
    grads = [1, 1, 1, 1; coords'] \ [0, 0, 0; eye(3)]; % gradients for CR basis
    grads = -3*grads([4, 3, 2, 1], :); % reorder to fit DoF numbering
    Alocal(:, :, elem) = volume * grads * grads'; % local stiffness matrix
    mid = mid4e(elem, :); % midpoint of this element
    b(faces) = b(faces) + volume*f(mid)*ones(4, 1)/4; % right-hand side
end

% assembly of the global stiffness matrix A
f4eT = f4e';
I = [f4eT; f4eT; f4eT; f4eT];
J = [f4eT(:, 1), f4eT(:, 2), f4eT(:, 3), f4eT(:, 4)]';
A = sparse(I(:, 1), J(:, 1), Alocal(:, 1));

```

4.3 Berechnung des Raviart-Thomas L ösers

4.3.1 Blöcke B und C

Wir wollen zuerst die in (3.12) mit B und C bezeichneten Blöcke berechnen. Dafür definieren wir uns zunächst lokale Basisfunktionen auf jedem Element unserer Triangulierung.

Definition 4.2. (lokale Basisfunktionen) Sei $T \in \mathcal{T}$ mit Schwerpunkt $\text{mid}(T)$. Wir definieren für $x \in T$ die *lokalen Basisfunktionen* durch

$$\Psi_1 = (1, 0, 0), \quad \Psi_2 = (0, 1, 0), \quad \Psi_3 = (0, 0, 1), \quad \Psi_4 = (x - \text{mid}(T)).$$

Es wird sich zeigen, dass das Auswerten der Integrale in (3.8)–(3.11) mit diesen lokalen Basisfunktionen zu dem Gleichungssystem (3.12) führt. Wenn wir $p_{\text{RT}}|_T$ als Linearkombination der lokalen Basisfunktionen Ψ_1, \dots, Ψ_4 darstellen und lokal auf dieser Basis testen, erhalten wir die lokalen Steifigkeitsmatrizen B_T und C_T .

Definition 4.3. Für jedes Element $T \in \mathcal{T}$ sind die lokalen Steifigkeitsmatrizen $B_T \in \mathbb{R}^{4 \times 4}$ und $C_T \in \mathbb{R}^{4 \times 1}$ gegeben durch

$$\begin{aligned} (B_T)_{jk} &:= \int_T \Psi_j \cdot \Psi_k \, dx \quad \text{für } j, k = 1, \dots, 4, \\ (C_T)_j &:= \int_T \operatorname{div} \Psi_j \, dx \quad \text{für } j = 1, \dots, 4. \end{aligned}$$

Da $\Psi_j \cdot \Psi_k = 0$ für $j \neq k$, sind nur die Diagonaleinträge von B_T von Null verschieden. Wir erhalten

$$\begin{aligned} (B_T)_{11} &= (B_T)_{22} = (B_T)_{33} = \int_T 1 \, dx = |T|, \\ (B_T)_{44} &= \int_T |x - \text{mid}(T)|^2 \, dx = \|x - \text{mid}(T)\|_{L^2(T)}^2 = s^2(T) |T| / 48, \end{aligned}$$

wobei wir im letzten Schritt Lemma 3.10 benutzt haben. Die Einträge von C_T lassen sich ebenfalls direkt berechnen

$$\begin{aligned} (C_T)_j &= \int_T \operatorname{div} \Psi_j \, dx = 0 \quad \text{für } j = 1, 2, 3, \\ (C_T)_4 &= \int_T \operatorname{div} \Psi_4 \, dx = \int_T 3 \, dx = 3 |T|. \end{aligned}$$

Die Blöcke B und C entstehen durch direktes aneinandereihen der lokalen Steifigkeitsmatrizen

$$B = \begin{pmatrix} B_1 & 0 & \cdots & 0 \\ 0 & B_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & B_{|\mathcal{T}|} \end{pmatrix} \quad \text{und} \quad C = \begin{pmatrix} C_1 & 0 & \cdots & 0 \\ 0 & C_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & C_{|\mathcal{T}|} \end{pmatrix}.$$

Dabei legen wir die globale Nummerierung der Freiheitsgrade des Systems durch den Aufbau der Matrizen fest. Die vier lokalen Freiheitsgrade eines Elementes stehen dabei untereinander und sind elementweise aneinander gereiht. Wir erhalten die beiden Blöcke B und C durch die folgenden Zeilen Matlab-Code

```

for curElem = 1 : nrElems
    % Summing up the length (norm) of all sides of curElem
    s = sum(length4s(s4e(curElem,:)).^2);
    % assembling the blocks of global stiffness matrices B and C
    B( 4*curElem-[3,2,1,0],4*curElem-[3,2,1,0] ) = volume4e(curElem) * diag
        ([1,1,1,s/48]);
    C( 4*curElem-[3,2,1,0],curElem ) = [0;0;0;3*volume4e(curElem)];
end

```

4.3.2 Block D

Definition 4.4. Für jede Fläche $F \in \mathcal{F}(\Omega)$ ist der lokale Vektor $D^F \in \mathbb{R}^{8 \times 1}$ gegeben durch

$$D_k^F = - \int_F \Psi_k \cdot \nu_F \, ds \quad \text{für } k = 1, \dots, 8,$$

wobei Ψ_1, \dots, Ψ_4 die lokalen Basisfunktionen in $\text{RT}_0(T_+)$ und Ψ_5, \dots, Ψ_8 die lokalen Basisfunktionen in $\text{RT}_0(T_-)$ bezeichnen.

Wir berechnen nun die einzelnen Einträge von D^F . Sei dafür $\nu_F := (\nu_F(1), \nu_F(2), \nu_F(3))$ der Normaleneinheitsvektor auf $F \in \mathcal{F}(\Omega)$ und $P_F \in F$ ein beliebiger Punkt. Wir bemerken, dass $x - P_F$ tangential zu F und damit senkrecht auf ν_F ist. Damit ist $(x - \text{mid}(T)) \cdot \nu_F = (P_F - \text{mid}(T)) \cdot \nu_F$ und wir erhalten

$$\begin{aligned} D_k^F &= - \int_F \Psi_k \cdot \nu_F \, ds = -\nu_F(k) \int_F 1 \, ds = -\nu_F(k)|F| \quad \text{für } k = 1, 2, 3, \\ D_k^F &= - \int_F \Psi_k \cdot \nu_F \, ds = -\nu_F(k) \int_F -1 \, ds = \nu_F(k)|F| \quad \text{für } k = 5, 6, 7, \\ D_4^F &= - \int_F (x - \text{mid}(T_+)) \cdot \nu_F \, ds = -|F|(P_F - \text{mid}(T_+)) \cdot \nu_F \\ D_8^F &= - \int_F (x - \text{mid}(T_-)) \cdot \nu_F \, ds = |F|(P_F - \text{mid}(T_-)) \cdot \nu_F. \end{aligned}$$

Wir definieren $\rho_+ := (P_F - \text{mid}(T_+)) \cdot \nu_F$ und $\rho_- := (P_F - \text{mid}(T_-)) \cdot \nu_F$ und erhalten

$$D^F = |F| (-\nu_F, -\rho_+, \nu_F, \rho_-)^\top.$$

Die globale Matrix D ergibt sich aus den lokalen Beiträgen D^F mit der Bezeichnung $4j - [3, 2, 1, 0] := (4j - 3, 4j - 2, 4j - 1, 4j)$ für eine Elementnummer j der Triangulierung durch

$$D((4j_+ - [3, 2, 1, 0], 4j_- - [3, 2, 1, 0]), \ell) = D^{F_\ell} \quad \text{für } \ell = 1, \dots, |\mathcal{F}(\Omega)|,$$

wobei mit j_+ und j_- die globalen Nummern der Elemente T_+ und T_- bezeichnet werden. Realisiert wird die Berechnung des Blocks D durch

```

for curIFace = 1 : nrIFaces
    face = f4n{ n4iFaces( curIFace ,1) ,1} ( n4iFaces( curIFace ,2) ,n4iFaces(
        curIFace ,3));
    curNormal = normal4f( face ,:) ';
    rho1 = (c4n( n4iFaces( curIFace ,1) ,:)-mid4e( e4f( face ,1) ,:)) * curNormal;
    rho2 = (c4n( n4iFaces( curIFace ,1) ,:)-mid4e( e4f( face ,2) ,:)) * curNormal;
    % assembling the blocks of global stiffness matrix D
    D( [4*e4f( face ,1) -[3,2,1,0],4*e4f( face ,2) -[3,2,1,0]], curIFace ) ...
        = -area4f( face )*[ curNormal; rho1;-curNormal;-rho2 ];
end

```

4.3.3 Block E

Analog zur Berechnung von Block D definieren wir uns zuerst die lokalen Beiträge für die Konstruktion von Block E .

Definition 4.5. Für jede Fläche $F \in \mathcal{F}(\Gamma_N)$ ist der lokale Vektor $E^F \in \mathbb{R}^{4 \times 1}$ gegeben durch

$$E_k^F = \int_F \Psi_k \cdot \nu_F \, ds \quad \text{für } k = 1, \dots, 4.$$

Eine direkte Berechnung der Einträge von E^F auf der Fläche $F \in \partial T \cap \Gamma_N$ liefert

$$\begin{aligned} E_k^F &= \int_F \Psi_k \cdot \nu_F \, ds = \nu_F(k) \int_F 1 \, ds = \nu_F(k) |F| \quad \text{für } k = 1, 2, 3, \\ E_4^F &= \int_F (x - \text{mid}(T)) \cdot \nu_F \, ds = |F| (P_F - \text{mid}(T)) \cdot \nu_F. \end{aligned}$$

für einen beliebigen Punkt $P_F \in F$. Wir definieren $\rho := (P_F - \text{mid}(T)) \cdot \nu_F$ und erhalten somit $E^F = |F| (\nu_F, h)^\top$. Die globale Matrix E ergibt sich aus den lokalen Beiträgen E^F durch

$$E(4j - [3, 2, 1, 0], \ell) = E^{\ell} \quad \text{für } \ell = 1, \dots, |\mathcal{F}(\Gamma_N)|,$$

wobei mit j die globale Nummer von T bezeichnet wird. Die folgenden Zeilen Matlab-Code berechnen Block E

```

for curNbSide = 1 : nrNbFaces
    face = f4n{ n4fNb( curNbSide ,1) ,1} ( n4fNb( curNbSide ,2) ,n4fNb( curNbSide ,3)
        );
    curNormal = normal4f( face ,:) ';
    curElem = e4f( face ,1);
    rho = (c4n( n4fNb( curNbSide ,1) ,:)-mid4e( curElem ,:)) * curNormal;
    E(4*curElem -[3,2,1,0], curNbSide) = area4f( face )*[ curNormal; rho ];
end

```

4.3.4 Blöcke der rechten Seite b_D , b_f und b_g

Definition 4.6. Für jedes Element $T \in \mathcal{T}$ ist die rechte Seite $b_f^T \in \mathbb{R}$ unter Verwendung der Mittelpunktsregel gegeben durch

$$b_f^T := - \int_T f(x) dx \approx -f(\text{mid}(T)) |T| .$$

Die Dirichlet-Randdaten b_D erhalten wir wieder durch direkte Berechnung.

Definition 4.7. Für jede Dirichlet-Fläche $F \in \mathcal{F}(\partial T) \cap \mathcal{F}(\Gamma_D)$ ist die rechte Seite $b_D^F \in \mathbb{R}^4$ gegeben durch

$$(b_D^F)_k := \int_F u_D(x) \Psi_k \cdot \nu_F dx .$$

Setzen wir die Basisfunktionen ein, erhalten wir

$$\begin{aligned} (b_D^F)_k &= \nu_F(k) \int_F u_D(x) dx \approx \nu_F(k) u_D(\text{mid}(F)) |F| \quad \text{für } k = 1, 2, 3, \\ (b_D^F)_4 &= \rho \int_F u_D(x) dx \approx \rho u_D(\text{mid}(F)) |F| \end{aligned}$$

für $\rho = (P_F - \text{mid}(T)) \cdot \nu_F$ und $P_F \in F$ beliebig. Die globalen Dirichlet-Randdaten erhalten wir durch

$$b_D(4j - [3, 2, 1, 0]) = \sum_{F \in \mathcal{F}(\partial T_j) \cap \mathcal{F}(\Gamma_D)} b_D^F,$$

wobei j für die globale Nummer von T steht. Die Neumann-Randdaten bekommen wir durch einfaches Integrieren.

Definition 4.8. Für jede Neumann-Fläche $F \in \partial T \cap \Gamma_N$ erhalten wir die rechte Seite $b_g^F \in \mathbb{R}$ durch eine Mittelpunktsquadratur

$$b_g^F := \int_F g(x) dx \approx g(\text{mid}(F)) |F| .$$

Realisiert wird die Berechnung der rechten Seite durch

```

for curElem = 1 : nrElems
    b(4*nrElems+curElem) = -volume4e(curElem) * f(mid4e(curElem,:));
end

for curDbFace = 1 : nrDbFaces
    face = f4n{n4fdb(curDbFace,1),1}(n4fdb(curDbFace,2),n4fdb(curDbFace,3));
    curNormal = normal4f(face,:);
    curElement = e4f(face,1);
    rho = (c4n(n4fdb(curDbFace,1),:)-mid4e(curElement,:)) * curNormal;
    b(4*curElement-[3,2,1,0]) = b(4*curElement-[3,2,1,0]) ...

```

```

+ u4Db( mid4f( face ,:) ) *area4f( face )*[ curNormal ; rho ];
end

for curNbSide = 1 : nrNbFaces
    b(5*nrElems+nrIFaces+curNbSide) = area4f( face )*g( mid4f( face ,:) );
end

```

4.4 Berechnung der Fehlerschätzer

4.4.1 P_1 -konformer Schätzer

Wir wollen die einzelnen Summanden des konformen Fehlerschätzers berechnen. Sei also $u_C \in V_C(\mathcal{T})$ die konforme Lösung des Poisson-Problems mit Basisdarstellung

$$u_C = \sum_{j=1}^{|N|} u_j \varphi_j.$$

Durch Bemerkung 3.11 erhalten wir die Möglichkeit die Potenzen von h_T durch den Flächeninhalt der Flächen bzw. durch das Volumen der Elemente auszudrücken. Die L^2 -Norm der Funktion f können wir, wie wir es bereits in den Lösungsalgorithmen getan haben, durch

$$\|f\|_{L^2(T)}^2 = \int_T f(x)^2 dx \approx f(\text{mid}(T))^2 |T|$$

approximieren. Um den Sprungterm zu berechnen, betrachten wir die beiden an F angrenzenden Simplizes T_+ und T_- . Wir berechnen zunächst den Gradienten unserer Näherungslösung. Sei dafür $u_\pm := (u_{j_\pm}, u_{k_\pm}, u_{\ell_\pm}, u_{m_\pm})^\top$ mit globalen Nummern $j_\pm, k_\pm, \ell_\pm, m_\pm$ der Eckpunkte von T_\pm . Wir wählen Q_\pm wie in Bemerkung 4.1 für T_\pm und erhalten

$$\nabla u_C|_{T_\pm} = u_\pm^\top Q_\pm.$$

Wir können nun den Sprungterm berechnen mit

$$\begin{aligned} \|[\nabla u_C]_F \cdot \nu_F\|_{L^2(F)}^2 &= \int_F \left((\nabla u_C|_{T_+} - \nabla u_C|_{T_-}) \cdot \nu_F \right)^2 dx \\ &= ((u_+^\top Q_+ - u_-^\top Q_-) \cdot \nu_F)^2 |F|. \end{aligned}$$

Die folgenden Matlab-Zeilen berechnen die Sprünge über die inneren Kanten

```

TPlus4f = e4f(:,1);
TMinus4f = e4f(:,2);
% use elem 1 for calculation, overwrite later with correct value
TMinus4f(TMinus4f == 0) = 1;
valTPPlus = sum(gradU(TPlus4f,:).*normal4f,2);
valTMinus = sum(gradU(TMinus4f,:).*normal4f,2);

```

Da T_- für Randflächen nicht definiert ist, müssen wir die Dirichlet- und Neumann-Randdaten gesondert betrachten. Für eine Neumann-Randfläche $F \in \mathcal{F}(\partial T) \cap \mathcal{F}(\Gamma_N)$ approximieren wir den Sprung $[\nabla u_C \cdot \nu_F]_F = \nabla u_C|_T \cdot \nu_F - g(x)$ durch eine stückweise konstante Funktion für g , d.h. durch

$$\nabla u_C|_T \cdot \nu_F - g(\text{mid}(F)).$$

Da die Dirichlet-Flächen in unserem Fehlerschätzer nicht enthalten sind, setzen wir den Sprung über diese Flächen auf Null, d.h. für eine Fläche $F \in \mathcal{F}(\partial T) \cap \mathcal{F}(\Gamma_D)$ sei

$$[\nabla u_C \cdot \nu_F]_F = 0.$$

Die Berechnung der Randbedingungen wir durch

```

%% Neumann jumps
f4Nb = zeros(size(n4fNb,1),1);
for j=1:size(n4fNb,1)
    f4Nb(j) = f4n{n4fNb(j,1),1}{(n4fNb(j,2),n4fNb(j,3))};
end
valTMinus(f4Nb) = g(mid4f(f4Nb,:));
jump4f = valTPPlus - valTMinus;

%% Dirichlet jumps = 0
f4Db = zeros(size(n4fDb,1),1);
for j=1:size(n4fDb,1)
    f4Db(j) = f4n{n4fDb(j,1),1}{(n4fDb(j,2),n4fDb(j,3))};
end
jump4f(f4Db) = 0;

```

umgesetzt.

4.4.2 Crouzeix-Raviart Schätzer

Da die Terme für den nicht-konformen Fehlerschätzer fast dieselben wie für den konformen sind, müssen wir nur beachten, dass wir den nicht-konformen Gradienten unserer Näherungslösung korrekt berechnen und die Randbedingungen an unsere Methode anpassen. Sei dafür $u_{\text{CR}} \in V_{\text{NC}}(\mathcal{T})$ die nicht-konforme Lösung des Poisson-Problems mit Basisdarstellung

$$u_{\text{CR}} = \sum_{j=1}^{|\mathcal{F}|} u_j \psi_j.$$

Um den Gradienten zu bestimmen, erinnern wir uns an Lemma 3.5 und berechnen mit der Notation aus dem vorherigen Kapitel und $\hat{u}_\pm := (\hat{u}_{j\pm}, \hat{u}_{k\pm}, \hat{u}_{\ell\pm}, \hat{u}_{m\pm})^\top$ mit globalen Nummern $j_\pm, k_\pm, \ell_\pm, m_\pm$ der Flächenmittelpunkte von T_\pm

$$\nabla_{\text{NC}} u_{\text{CR}}|_{T_\pm} = -3 \hat{u}_\pm^\top Q_\pm,$$

was in den folgenden Zeilen umgesetzt wird

```

for elem = 1 : nrElems
    nodes = n4e(elem ,1:4) ;
    coords = c4n(nodes ,:) ;
    grads = [1 ,1 ,1 ,1; coords '] \ [0 ,0 ,0;(-3)*eye(3)] ;
    grads = grads([4 ,3 ,2 ,1],:) ;
    gradU(elem ,:) = x(f4e(elem ,:))' * grads ;
end
```

Da im Crouzeix-Raviart Schätzer keine Neumann-Randflächen enthalten sind, setzen wir dieses mal die Sprünge über die Neumann-Flächen Null. Für eine Dirichlet-Randfläche $F \in \mathcal{F}(\partial T) \cap \mathcal{F}(\Gamma_D)$ erhalten wir

$$[\nabla_{NC} u_{CR}]_F = \nabla_{NC} u_{CR}|_T - \nabla_{NC} u_D|_T \approx -3 \hat{u}_T^\top Q + 3 \hat{u}_D^\top Q,$$

wobei $\hat{u}_D = (u_D(\text{mid}(F_j)), u_D(\text{mid}(F_k)), u_D(\text{mid}(F_\ell)), u_D(\text{mid}(F_m)))^\top$. Wir können den Sprung in Tangentialrichtung durch die folgenden Matlab-Zeilen realisieren

```

%% inner jumps
TPlus4f = e4f(:,1) ;
TMinus4f = e4f(:,2) ;
%Use elem 1 for calculation, overwrite later with correct value
TMinus4f(TMinus4f == 0) = 1;

valTPPlus = gradU(TPlus4f ,:) - ...
    repmat(sum(gradU(TPlus4f ,:).*normal4f ,2) ,1,3).*normal4f;
valTMinus = gradU(TMinus4f ,:) - ...
    repmat(sum(gradU(TMinus4f ,:).*normal4f ,2) ,1,3).*normal4f;

%% Dirichlet jumps
f4Db = zeros(size(n4fDb ,1) ,1) ;
for j=1:size(n4fDb ,1)
    f4Db(j) = f4n{n4fDb(j ,1),1}(n4fDb(j ,2) ,n4fDb(j ,3)) ;
end

% gradU for DB (gradient on hole T_+)
gradU4Db = zeros(size(n4f ,1) ,3) ;
for k = 1 : size(f4Db ,1)
    face = f4Db(k) ;
    elem = e4f(face ,1) ;
    coord = c4n(n4e(elem ,1:4) ,:) ;
    grads = [1 ,1 ,1 ,1; coord '] \ [0 ,0 ,0;(-3)*eye(3)] ;
    grads = grads([4 ,3 ,2 ,1],:) ;
    gradU4Db(face ,:) = u4Db(mid4f(f4e(elem ,:))' * grads ;
end

valTMinus(f4Db ,:) = gradU4Db(f4Db ,:) - ...
    repmat(sum(gradU4Db(f4Db ,:).*normal4f(f4Db ,:),2) ,1,3).*normal4f(f4Db
        ,:) ;
jump4f = sum((valTPPlus - valTMinus).^2 ,2) ;
```

4.4.3 Raviart-Thomas Schätzer

Sei $(p_{\text{RT}}, u_{\text{RT}}) \in \text{RT}_0(\mathcal{T}) \times P_0(\mathcal{T})$ die Lösung der Raviart-Thomas Finite-Elemente-Methode. Wie wir in Abschnitt 3.3.3 gezeigt haben, lässt sich die Lösung durch die Lösung der Crouzeix-Raviart Methode darstellen. Damit erhalten wir die gleichen Anforderungen an die Dirichlet- und Neumann-Randdaten wie in Abschnitt 4.4.2. Um den Sprungterm zu berechnen, gehen wir ähnlich wie bei dem Crouzeix-Raviart Fehlerschätzer vor. Dabei müssen wir allerdings beachten, dass $p_{\text{RT}} \in \text{RT}_0(\mathcal{T})$ nicht auf jedem Element konstant ist und wir somit den Sprung durch eine Quadraturformel berechnen müssen. Es bleibt der Volumenterm des Fehlerschätzers zu berechnen. Wir erhalten

$$\|f - f_T\|_{L^2(T)}^2 = \int_T (f - f_T)^2 dx = \int_T \left(f - \frac{1}{|T|} \int_T f dx \right)^2 dx$$

und integrieren f auch hier wieder numerisch, was wir in den folgenden Zeilen Matlab-Code umsetzen

```
mean4e = integrate3d(c4n, n4e, @(parts, Gpts4p, Gpt4ref) f(Gpts4p), 6) ./  
volume4e;  
value4e = integrate3d(c4n, n4e, @(parts, Gpts4p, Gpt4ref) (f(Gpts4p) -  
mean4e).^2, 12);
```

Wir bemerken an dieser Stelle, dass die numerische Berechnung der Integrale in dieser Arbeit durch mehrmaliges anwenden von eindimensionalen Gauß-Quadraturformeln realisiert wird. Für nähere Informationen verweisen wir auf die Dokumentation in *integrate3D.m*.

4.5 Berechnung der Marini-Darstellung

Wir wollen nun noch zeigen, wie sich (3.13) und (3.14) aus der Lösung u_{CR} der Crouzeix-Raviart Finite-Elemente-Methode berechnen lassen. Für die Berechnung von $u_{\text{RT}}|_T$ bemerken wir, dass

$$\oint_T u_{\text{CR}} dx = u_{\text{CR}}(\text{mid}(T)) = \sum_{j=1}^4 u_{\text{CR}}(\text{mid}(F_j))/4$$

mit Flächen $F_1, \dots, F_4 \in \partial T$ und verweisen für die Berechnung des zweiten Summanden auf die Abschnitte 4.4.3 und 3.3.3. Um p_{RT} zu berechnen, erinnern wir uns an die lokale Darstellung von p_{RT} durch

$$p_{\text{RT}}(x) = \sum_{j=1}^4 p_j \Psi_j(x)$$

mit den lokalen Basisfunktionen Ψ_j aus Definition 4.2. Wir können jetzt die Koeffizienten p_1, \dots, p_4 direkt aus (3.13) durch

$$\begin{aligned} p_j &= \nabla_{\text{NC}} u_{\text{CR},j} \quad \text{für } j = 1, 2, 3, \\ p_4 &= -f_T/3, \end{aligned}$$

berechnen, wobei $\nabla_{\text{NC}} u_{\text{CR}} = (\nabla_{\text{NC}} u_{\text{CR},1}, \nabla_{\text{NC}} u_{\text{CR},2}, \nabla_{\text{NC}} u_{\text{CR},3})$. Die folgenden Zeilen Matlab-Code realisieren die Berechnung

```
%% compute integral means
mean4f = integrate3d(c4n, n4e, @(parts, Gpts4p, Gpt4ref) f(Gpts4p), 6)./
    volume4e;
mean4u_CR = sum(x(f4e), 2)/4;
s = sum(length4s(s4e).^2, 2);

%% compute p_RT with Marini-formula
p_RT = zeros(size(n4e, 1), 4);
for elem = 1 : size(n4e, 1)
    nodes = n4e(elem, 1:4); % nodes of this element
    faces = f4e(elem, :); % faces of this element
    coords = c4n(nodes, :); % coordinates for the nodes
    grads = [1, 1, 1, 1; coords'] \ [0, 0, 0; eye(3)]; % gradients for CR basis
    grads = -3*grads([4, 3, 2, 1], :); % reorder to fit DoF numbering
    p_RT(elem, 1:3) = sum(repmat(x(faces), 1, 3) .* grads, 1);
    p_RT(elem, 4) = -mean4f(elem)/3;
end

%% compute u_RT with Marini-formula
u_RT = mean4u_CR + s.*mean4f/(9*48);
```

4.6 Realisierung

Die im Rahmen dieser Arbeit implementierten Methoden sind als Erweiterung der AFEM-Software der Arbeitsgruppe „Numerische Analysis“ von Prof. Carstensen am Institut für Mathematik der Humboldt-Universität zu Berlin [Car09a] gedacht und sind folglich zur korrekten Funktionsweise auf andere Methoden aus diesem Paket angewiesen bzw. orientieren sich an bereits vorhandenen Funktionen. Die ausführbare Methode der Programme lautet

```
function afem3d(problem, method, minNrDoF, OPTtheta1, OPTtheta2,
    OPTcreatePlots)
```

Sie realisiert den AFEM-Algorithmus aus Abbildung 1.1. Über die Argumente der Funktion kann der Benutzer dem Programm mitteilen, welches Beispielproblem berechnet werden soll. Außerdem können eine minimale Anzahl von Freiheitsgraden als Abbruchkriterium sowie zwei optionale „bulk“ -Parameter θ_1 und θ_2 für die adaptive Rechnung

übergeben werden. Der letzte Parameter gibt dem Benutzer die Möglichkeit, die Ergebnisse graphisch darzustellen. Die möglichen Optionen sind in den folgenden Tabellen aufgelistet.

problem	Beschreibung
CubeExact	„Cube“ mit bekannter exakter Lösung
LshapeExact	„L-Shape“ mit bekannter exakter Lösung
Cube	„Cube“ mit rechter Seite $f = 1$ und Nullranddaten
Lshape	„L-Shape“ mit rechter Seite $f = 1$ und Nullranddaten
FicheraCorner	„Fichera-Corner“ mit rechter Seite $f = 1$ und Nullranddaten

method Methode

P1	P ₁ -konforme FEM
CR	CR-nicht-konforme FEM
RT0	RT ₀ -gemischte FEM

theta Verfeinerung

$\theta = 1$	uniform
$0 < \theta < 1$	adaptiv

Die Geometriedaten werden im Ablauf des Programmes aus den Dateien

Cube . mat
varLshape3D . mat
FicheraCorner . mat

ausgelesen. Das Programm ruft, je nach Eingabe, die entsprechende Funktion für die uniforme und bis zu zwei adaptiven Rechnungen aus der folgenden Liste auf.

afemP1CubeExactUniform .m	afemP1LshapeExactUniform .m
afemP1CubeExactAdaptive .m	afemP1LshapeExactAdaptive .m
afemCRCubeExactUniform .m	afemCRLshapeExactUniform .m
afemCRCubeExactAdaptive .m	afemCRLshapeExactAdaptive .m
afemRT0CubeExactUniform .m	afemRT0LshapeExactUniform .m
afemRT0CubeExactAdaptive .m	afemRT0LshapeExactAdaptive .m
afemP1CubeUniform .m	afemP1LshapeUniform .m
afemP1CubeAdaptive .m	afemP1LshapeAdaptive .m
afemCRCubeUniform .m	afemCRLshapeUniform .m
afemCRCubeAdaptive .m	afemCRLshapeAdaptive .m
afemRT0CubeUniform .m	afemRT0LshapeUniform .m
afemRT0CubeAdaptive .m	afemRT0LshapeAdaptive .m
afemP1FicheraCornerUniform .m	
afemP1FicheraCornerAdaptive .m	
afemCRFicheraCornerUniform .m	
afemCRFicheraCornerAdaptive .m	
afemRT0FicheraCornerUniform .m	
afemRT0FicheraCornerAdaptive .m	

Wir verwenden die Datenstrukturen aus Tabelle 4.1, die teilweise aus dem AFEM-Paket übernommen oder direkt für den dreidimensionalen Fall geschrieben wurden. Für nähere Informationen verweisen wir auf die AFEM-Dokumentation [CGK⁺10] sowie die den Programmen beiliegende Beschreibung.

Im Folgenden wollen wir die logische Struktur der implementierten Matlab-Funktionen erläutern. Die Programme starten mit der Übernahme eingegebener optionaler Parameter oder setzen diese auf sinnvolle Standardwerte, falls sie nicht extra übergeben wurden. Da der Rechenaufwand auf dreidimensionalen Geometrien sehr hoch ist, sind die Programme so geschrieben, dass sie alle Daten während eines AFEM-Zyklus abspeichert, um später auf diese Daten zugreifen zu können. Daher suchen die Programme, nachdem sie für den Programmablauf benötigte Konstanten definiert haben, nach dem letzten bereits berechneten Durchlauf des AFEM-Zyklus und laden diese Daten. Falls keine früheren Daten vorhanden sein sollten, laden die Programme die benötigten Daten der Geometrie. Anschließend wird der AFEM-Zyklus durchlaufen bis das Abbruchkriterium erfüllt ist.

4.6.1 Lösungsmethoden

Wir betrachten zunächst die drei Lösungsmethoden für die konforme Methode

```
function [x ,nrDoF ,A, b] = solveP1Poisson3D ( f ,g ,u4Db ,c4n ,n4e ,n4fDb ,n4fNb )
```

für die nicht-konforme Methode

```
function [x ,nrDof ,A, b] = solveCRPoisson3D ( f ,g ,u4Db ,c4n ,n4e ,n4fDb ,n4fNb )
```

und für die gemischte Methode

```
function [ p ,u ,nrDof] = solveRT0Poisson3D ( f ,g ,u4Db ,c4n ,n4e ,n4fDb ,n4fNb )
```

Wie die verschiedenen Steifigkeitsmatrizen und rechten Seiten berechnet werden, haben wir bereits in den Abschnitten 4.1 – 4.3 gesehen. Da die Lösungsmethoden, abgesehen von der Berechnung der Steifigkeitsmatrix, ähnlich aufgebaut sind, können wir sie hier alle gleichzeitig betrachten. Prinzipiell werden zuerst die benötigten Datenstrukturen berechnet (siehe Tabell 4.1) und anschließend wird Speicherplatz für die Blöcke der Steifigkeitsmatrix reserviert. Die lokalen Beiträge der Blöcke werden zunächst für jeden Simplex bzw. für jede Fläche an den entsprechenden Stellen aufaddiert und anschließend zu dem linearen Gleichungssystem zusammengebaut. Zum Schluss wird das Gleichungssystem auf den entsprechenden Freiheitsgraden gelöst. Wir bemerken an dieser Stelle nur, dass sich das Lösen des Gleichungssystems in *solveRT0Poisson3D.m* von den anderen beiden unterscheidet, da wir durch die Lagrange-Multiplikator-Technik nicht nur auf den Freiheitsgraden, sondern das ganze System lösen. Die Lösung der Raviart-Thomas Methode über die Marini-Darstellung berechnen wir in

```
function [ p_RT,u_RT] = solveRT0Marini3D ( f ,g ,u4Db ,c4n ,n4e ,n4fDb ,n4fNb )
```

Dabei berechnen wir zuerst die Crouzeix-Raviart Lösung mittels *solveCRPoisson3D.m* und berechnen daraus (p_{RT}, u_{RT}), wie in Abschnitt 4.5 beschrieben.

Datum	Beschreibung
<i>Geometrie</i>	
c4n	Koordinaten für die Knoten der Triangulierung (zeilenweise)
n4e	Knotennummern für die Simplizes der Triangulierung (zeilenweise)
n4s	Knotennummern für die Kanten der Triangulierung (zeilenweise)
n4f	Knotennummern für die Kanten der Flächen (zeilenweise)
s4e	Kantennummern für die Elemente der Triangulierung (zeilenweise)
s4n	Kantennummer für die gemeinsame Kante zweier Knoten (0, falls die Knoten keine gemeinsame Kante besitzen)
s4f	Kantennummern für die Flächen einer Triangulierung (zeilenweise)
f4e	Flächennummern für die Elemente einer Triangulierung (zeilenweise)
f4n	Flächennummer für die Fläche, die drei Knoten bilden (0, falls die Knoten keine gemeinsame Fläche bilden)
e4f	Nummern der angrenzenden Elemente einer Fläche (Reihenfolge entsprechend der Richtung des Normalenvektors der Fläche)
n4fDb	Knotennummern für die Randflächen der Triangulierung, auf denen Dirichlet-Daten gegeben sind (zeilenweise)
n4fNb	Knotennummern für die Randflächen der Triangulierung, auf denen Neumann-Daten gegeben sind (zeilenweise)
length4s	Länge der Kanten der Triangulierung
area4f	Flächeninhalt der Flächen der Triangulierung
volume4e	Volumen der Elemente der Triangulierung
normal4e	die vier äußeren Normaleneinheitsvektoren für die Elemente der Triangulierung
normal4f	Normaleneinheitsvektor für die Flächen der Triangulierung (als Zeilenvektor)
mid4s	Mittelpunkte der Kanten der Triangulierung
mid4f	Mittelpunkte der Flächen der Triangulierung
mid4e	Mittelpunkte der Elemente der Triangulierung
<i>Input Daten</i>	
f	rechte Seite aus dem zu lösenden Problem
u4Db	Dirichlet-Randdaten
g	Neumann-Randdaten
uExact	exakte Funktion (falls bekannt)
gradExact	Gradient der exakten Funktion (falls bekannt)

Tabelle 4.1: Datenstrukturen

4.6.2 Fehlerschätzer

Die Fehlerschätzer aus (3.16), (3.18) und (3.20) werden in

```
function eta4e = estimateP1EtaElements(f ,g ,u4Db ,x ,c4n ,n4e ,n4fDb ,n4fNb)
function eta4e = estimateCREtaElements(f ,g ,u4Db ,x ,c4n ,n4e ,n4fDb ,n4fNb)
function eta4e = estimateRT0EtaElements(f ,g ,u4Db ,p ,u ,c4n ,n4e ,n4fDb ,n4fNb)
```

implementiert. Die Berechnung ist auch hier wieder bei allen Fehlerschätzern ähnlich aufgebaut. Die Volumenterme werden entsprechend ihrer Beschreibung berechnet und die Sprungterme erhalten wir, je nach Fehlerschätzer, aus den Funktionen *estimateP1EtaFaces.m*, *estimateCREtaFaces.m* oder *estimateRT0EtaFaces.m*. Abschließend werden die Sprünge über die entsprechenden Flächen und der Volumenterm mit den jeweiligen h -Potenzen addiert.

Für die Berechnung der Sprungterme in den jeweiligen Programmen werden gegebenenfalls die Gradienten der Näherungslösung bzw. der Dirichlet-Randdaten berechnet. Die Sprünge sind direkt, wie in Abschnitt 4.4 beschrieben, implementiert. Wir benutzen die leicht abgeänderte Version *integrate3d.m* der Integrationsroutine der AFEM-Software, um Funktionen in *estimateRT0EtaElements.m* numerisch zu integrieren. Da wir lediglich Polynome geringer Ordnung und trigonometrische Funktionen integrieren, ist ein relativ geringer Integrationsgrad vollkommen ausreichend.

4.6.3 Exakter Fehler

Wir können zur Berechnung des exakten Fehlers $\|u - u_C\|$ für den konformen Fall die Funktion

```
function error4e = error4eP1Energy3D(c4n ,n4e ,gradExact ,uApprox)
```

und für den exakten Fehler im nicht-konformen Fall $\|u - u_{CR}\|_{NC}$ die Funktion

```
function error4e = error4eCREnergy3D(c4n ,n4e ,gradExact ,uApprox)
```

benutzen. Beide Funktionen berechnen den näherungsweisen Gradienten der Approximationslösung entsprechend der Methode und integrieren den Fehler mit *integrate3d.m*. Die Funktion

```
function error4e = error4eRT0Energy3D(c4n ,n4e ,gradExact ,p)
```

berechnet den Energiefehler $\|\nabla u - p_{RT}\|_{L^2(\Omega)}$ der Raviart-Thomas Finite-Elemente-Methode. Da unsere Approximation des Flusses p_{RT} elementweise in der lokalen Basis Ψ_1, \dots, Ψ_4 gegeben ist, müssen wir zuerst

$$p_{RT} = \sum_{j=1}^4 p_j \Psi_j$$

berechnen, bevor wir auch hier mit *integrate3d.m* den Fehler integrieren können.

5 Numerische Beispiele

Wir wollen jetzt die in den vorherigen Kapiteln beschriebenen Methoden an einigen Beispielen testen und untersuchen. Wir wählen uns dafür die folgenden fünf Probleme

1. „Cube“ mit bekannter exakter Lösung,
2. „L-Shape“ mit bekannter exakter Lösung,
3. „Cube“ mit konstanter rechter Seite,
4. „L-Shape“ mit konstanter rechter Seite,
5. „Fichera-Corner“ mit konstanter rechter Seite.

Wir werden jedes Beispiel mit der konformen, nicht-konformen und gemischten Finite-Elemente-Methode berechnen und dabei uniform und adaptiv verfeinern. Für die adaptive Verfeinerung werden wir den jeweils zur aktuellen Methode passenden Fehlerschätzer verwenden. Für die „bulk“-Parameter werden wir die Werte $\theta_1 = 0.5$ und $\theta_2 = 0.1$ wählen. Da wir durch die Verfeinerungsroutine eine Grenze in der Feinheit der Triangulierung vorgegeben bekommen, rechnen wir im konformen Fall mit bis zu 30 000 und im nicht-konformen sowie gemischten Fall mit bis zu 300 000 Freiheitsgraden, da eine Triangulierung mit 30 000 inneren Knoten etwa 300 000 innere Flächen besitzt.

Zur Kontrolle der Rechnungen werden wir, falls eine exakte Lösung bekannt ist, den exakten Energiefehler, wie er in Abschnitt 4.6.3 beschrieben ist, berechnen. Die Werte der Fehlerschätzer und der exakten Energiefehler werden gemeinsam für uniforme und adaptive Rechnungen in „convergence history plots“ dargestellt, um später Aussagen über die Güte der verwendeten Methoden und Fehlerschätzer treffen zu können.

Außerdem werden wir für einige Verfeinerungsstufen die Werte des Fehlerschätzers und (falls möglich) des Energiefehlers der RT₀-MFEM und der aus der CR-NCFEM berechneten Marini-Darstellung auflisten, um die Lösungen dieser beiden Methoden vergleichen zu können.

5.1 „Cube“ mit exakter Lösung

Sei $\Omega = (0, 1)^3$ der Einheitswürfel. Wir betrachten die rechte Seite

$$f(x, y, z) = 2 \left((x - x^2)(y - y^2) + (x - x^2)(z - z^2) + (y - y^2)(z - z^2) \right)$$

mit der exakten Lösung $u(x, y, z) = (x - x^2)(y - y^2)(z - z^2)$. Wir haben dieses Beispiel so gewählt, dass wir reine Dirichlet-Nullranddaten $u_D = 0$ erhalten. Die numerische Lösung der CR-NCFEM ist beispielhaft in Abbildung 5.2 für eine uniforme Verfeinerungsstrategie dargestellt. Unsere Ausgangstriangulierung besteht dabei aus fünf Elementen, wie in Abbildung 5.1 zu sehen ist.

Die Konvergenzraten (siehe Abbildungen 5.3 – 5.5) liegen bei allen drei Methoden, sowohl unter uniformer als auch unter adaptiver Verfeinerung bei $1/3$. Das insbesondere eine uniforme Verfeinerung die gleiche Konvergenzrate wie adaptive Verfeinerungen aufweist, sehen wir in einer nahezu gleichmäßigen Verfeinerung der Triangulierung auch bei adaptiven Rechnungen bestätigt. Wir bemerken außerdem, dass alle Methoden einen relativ kleinen vorasymptotischen Bereich haben, den sie ab ca. 500 Freiheitsgraden verlassen. Auffällig ist dabei ein sehr sprunghaftes Verhalten bei der konformen Methode, bei der die Werte des Fehlerschätzers anfänglich teilweise sogar größer werden. Es lässt sich beobachten, dass die Approximationsgüte aller drei Methoden ungefähr gleich ist. Dabei liegt der Fehlerschätzer in allen Fällen über dem exakten Fehler, wobei der Schätzer der gemischten Methode diesen am wenigsten überschätzt. Die in Tabelle 5.1 aufgelisteten Werte zeigen, dass die Lösungen der RT_0 -MFEM und der Marini-Darstellung relativ stark voneinander abweichen.

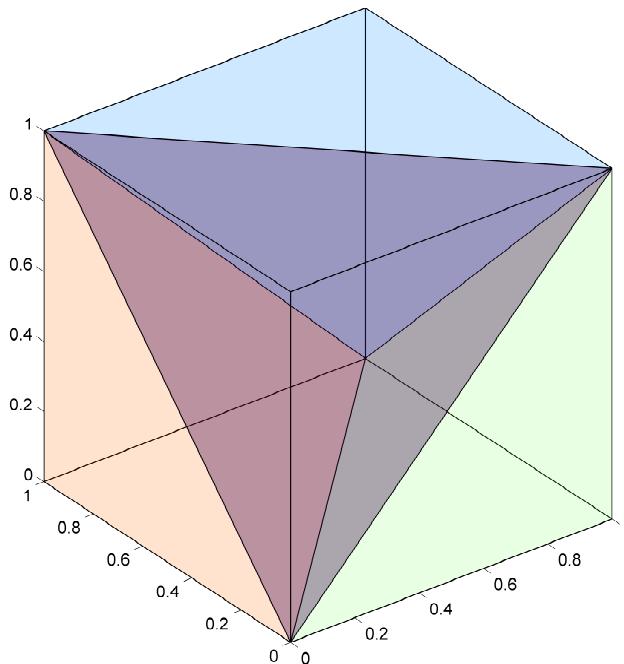


Abbildung 5.1: Anfangs-Triangulierung des „Cube“

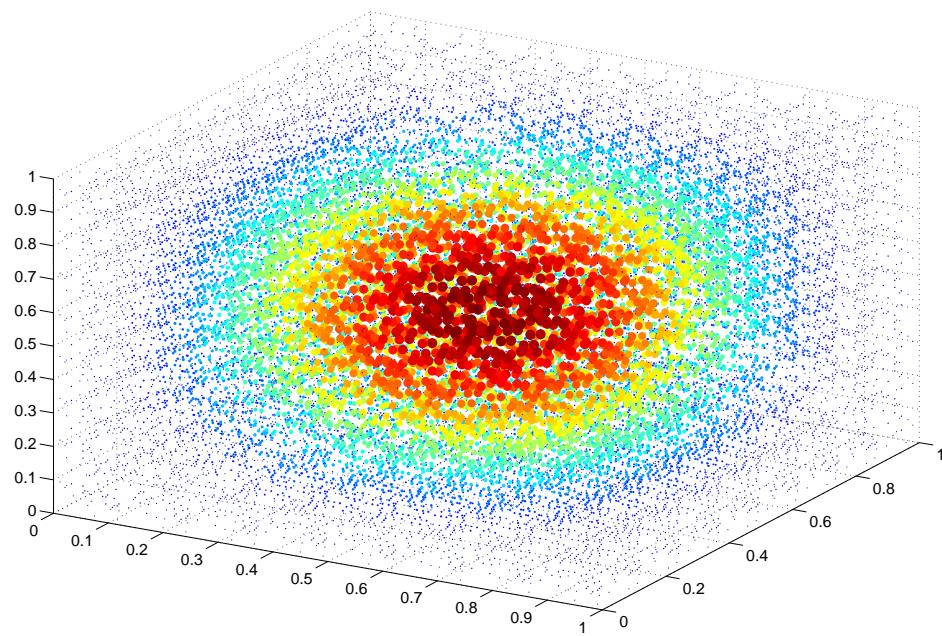


Abbildung 5.2: Uniforme Lösung der CR-NCFEM mit 19 520 Freiheitsgraden

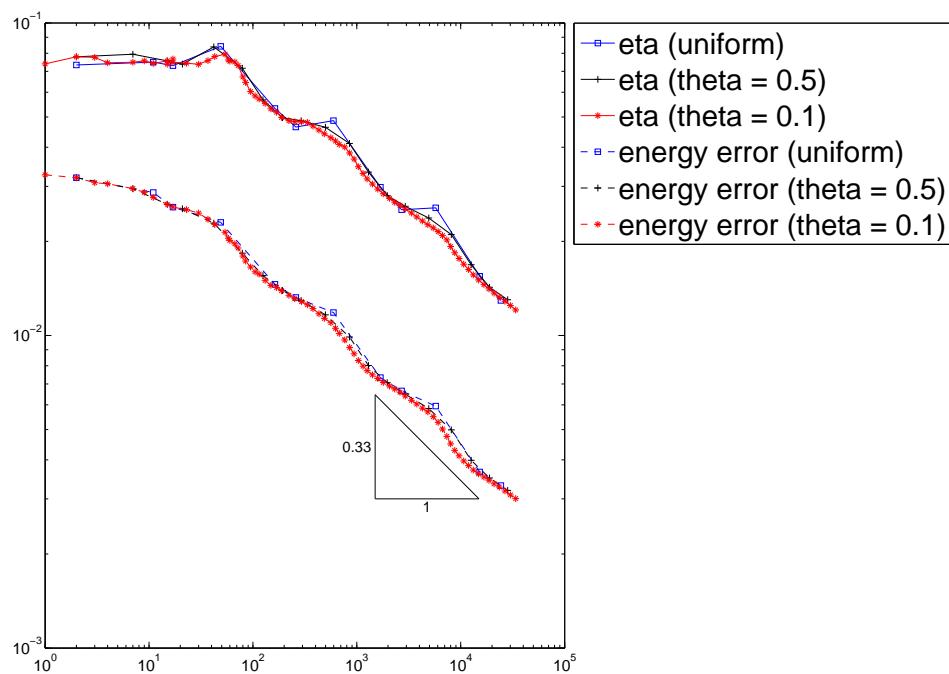


Abbildung 5.3: Konvergenzgraph der P₁-CFEM mit Fehlerschätzer und exaktem Energiefehler

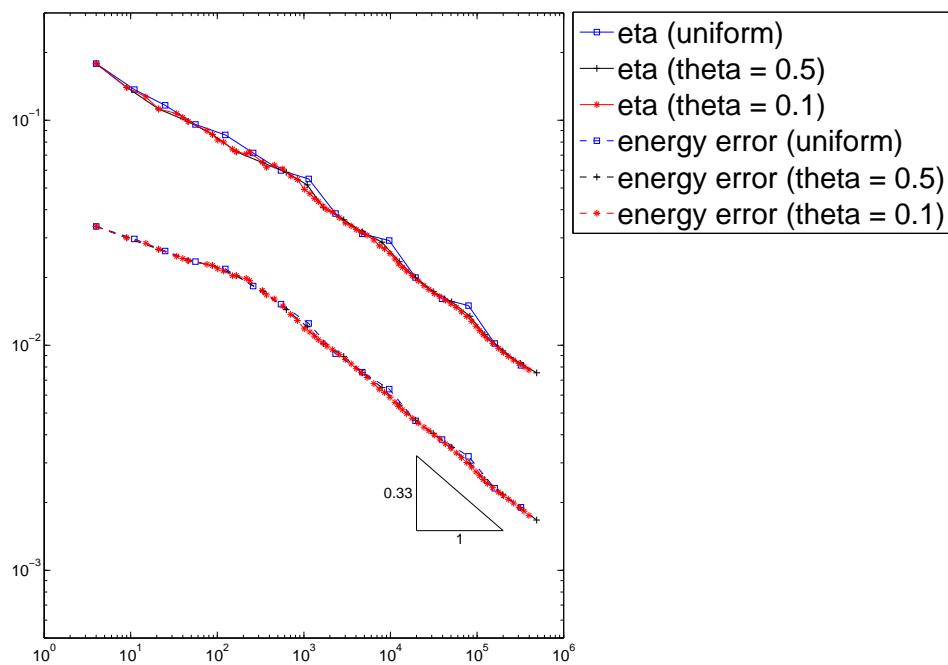


Abbildung 5.4: Konvergenzgraph der CR-NCFEM mit Fehlerschätzer und exaktem Energiefehler

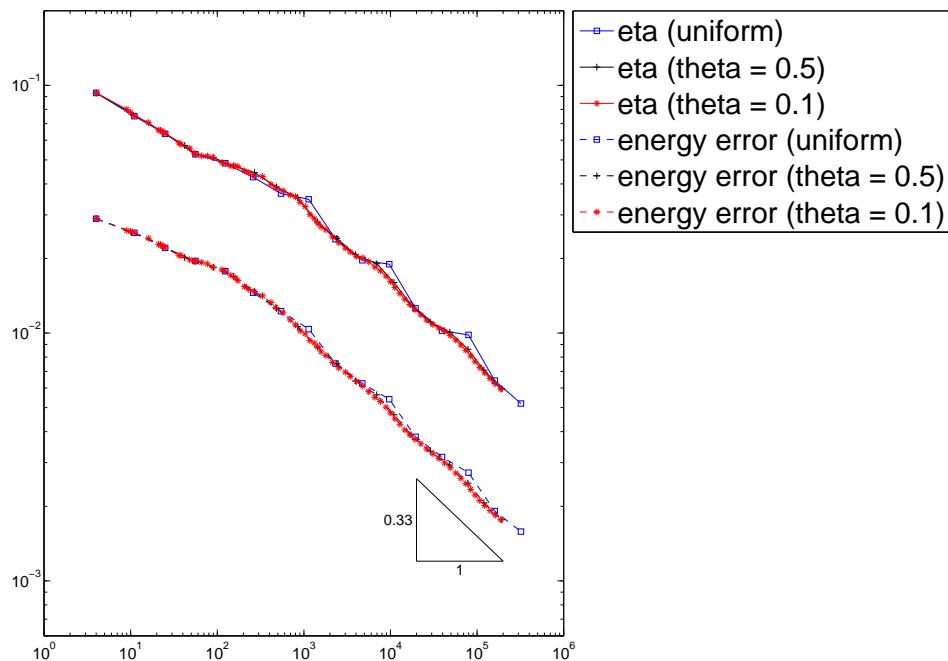


Abbildung 5.5: Konvergenzgraph der RT₀-MFEM mit Fehlerschätzer und exaktem Energiefehler

Level	eta4lvl	eta4lvlMarini	energy4lvl	energy4lvlMarini
<i>uniform</i>				
13	0.012592	0.0126	0.003813	0.0038151
14	0.010219	0.010224	0.0031594	0.0031607
15	0.0098334	0.0098353	0.0027337	0.0027343
16	0.0064329	0.0064339	0.0019133	0.0019136
17	0.0051953	0.0051958	0.0015807	0.0015809
$\theta = 0.5$				
18	0.011069	0.011074	0.0033364	0.0033379
19	0.010083	0.010087	0.0029342	0.0029353
20	0.0085963	0.0085987	0.0024812	0.0024819
21	0.0070905	0.0070918	0.0020652	0.0020655
22	0.0059458	0.0059467	0.001763	0.0017632
$\theta = 0.1$				
68	0.0072659	0.0072673	0.0021115	0.0021119
69	0.0069109	0.0069122	0.0020134	0.0020138
70	0.0065581	0.0065593	0.0019213	0.0019216
71	0.0062416	0.0062427	0.0018407	0.0018409
72	0.0059461	0.005947	0.0017638	0.0017641

Tabelle 5.1: Werte für Fehlerschätzer und Energiefehler der RT_0 -MFEM und der Marini-Darstellung für die jeweils fünf zuletzt berechneten Level unter uniformer und adaptiver Verfeinerung

5.2 „L-Shape“ mit exakter Lösung

Sei $\Omega = (0, 1)^3 \setminus ([0, 1] \times [-1, 0] \times [0, 1])$ das L-Gebiet wie in Abbildung 5.6 gezeigt. Wir betrachten die rechte Seite $f \equiv 0$ mit der exakten Lösung

$$u(r, \varphi, z) = r^{2/3} \sin(2/3 \varphi)$$

in dreidimensionalen Zylinderkoordinaten. Wir wählen die Dirichlet-Randdaten für dieses Beispiel exakt, d.h. $u_D(x) := u(x)$. Die Lösung der CR-NCFEM ist in Abbildung 5.8 dargestellt.

Die Konvergenzraten (siehe Abbildungen 5.9 – 5.11) des Fehlerschätzers und des exakten Fehlers liegen bei allen Methoden unter adaptiver Verfeinerung bei $1/3$. Lediglich bei uniformer Verfeinerung können wir eine leicht geringere Konvergenzrate von $1/5$ beobachten. Der vorasymptotische Bereich ist sehr klein und wird bei der konformen Methode ab ca. 10 und bei der nicht-konformen und gemischten Methode ab ca. 100 Freiheitsgraden verlassen. Der Fehlerschätzer liegt in allen Fällen wieder deutlich über dem exakten Fehler, wobei die Schätzer der nicht-konformen und der gemischten Methode fast die selben Werte liefern. Wir können weiterhin beobachten, dass der exakte Fehler ungefähr eine Zehnerpotenz höher liegt, als bei dem vorherigen Beispiel. Die Auflösung der Singularität und der einspringenden Kante spiegelt sich in dem adaptiv verfeinerten Netz der Triangulierung wieder, wie in Abbildung 5.7 zu sehen ist. Im Gegensatz zum vorherigen Beispiel lassen die in Tabelle 5.2 aufgelisteten Werte den Schluss zu, dass die RT₀-MFEM und die Lösung der Marini-Darstellung die gleichen Ergebnisse liefern.

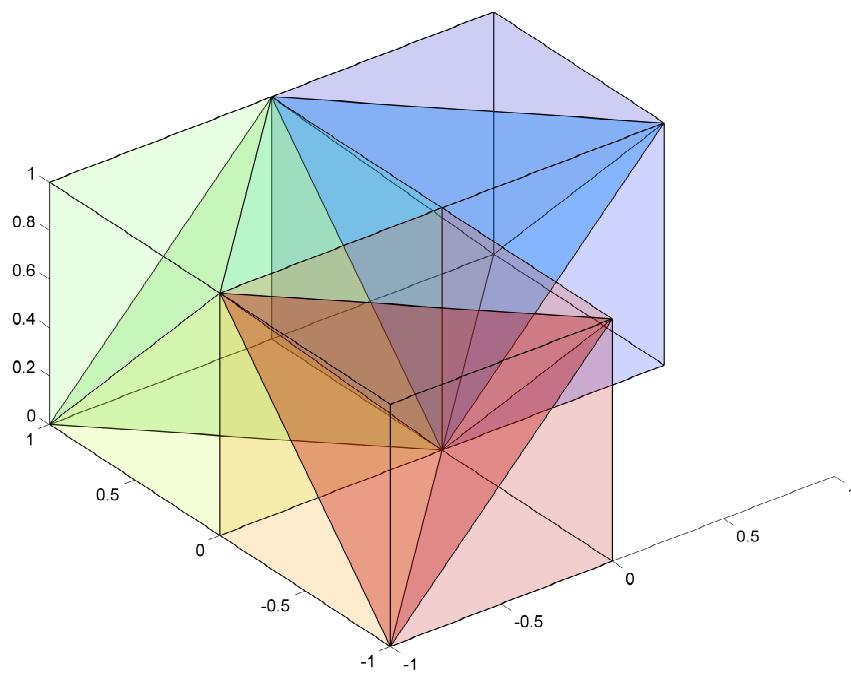


Abbildung 5.6: Anfangs-Triangulierung des „L-Shape“

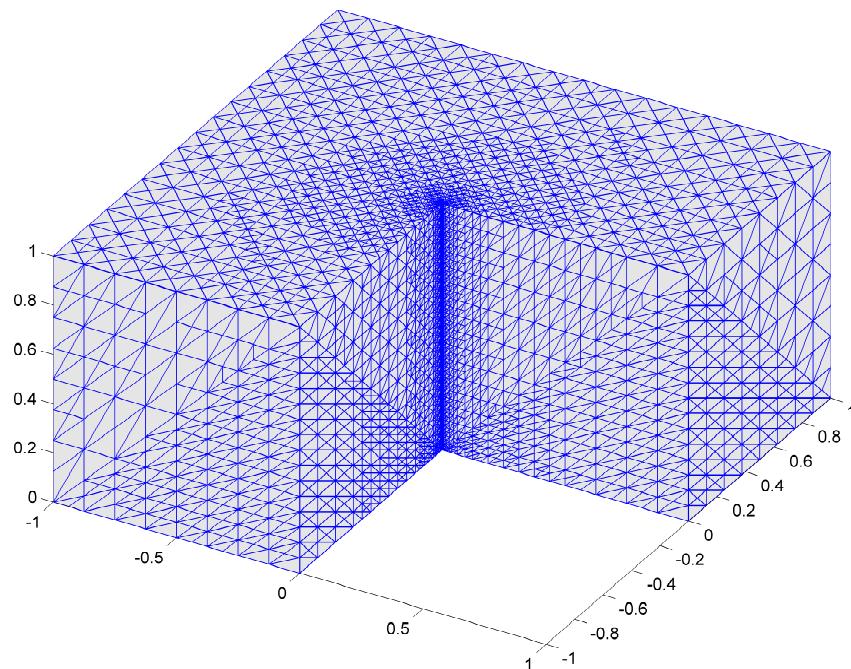


Abbildung 5.7: Oberfläche des adaptiv verfeinerten Netzes mit „bulk“-Parameter $\theta = 0.1$

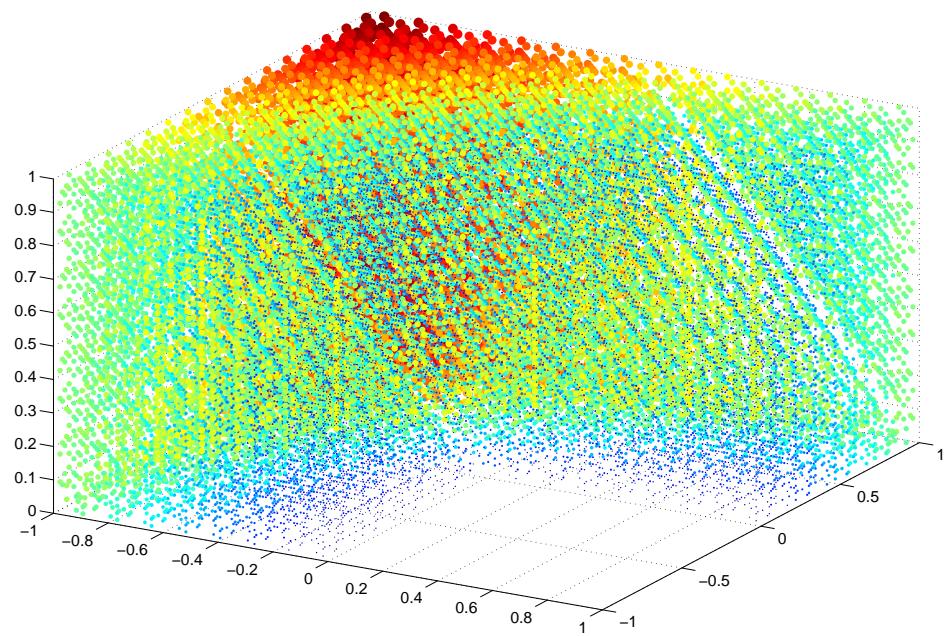


Abbildung 5.8: Uniforme Lösung der CR-NCFEM mit 29 376 Freiheitsgraden

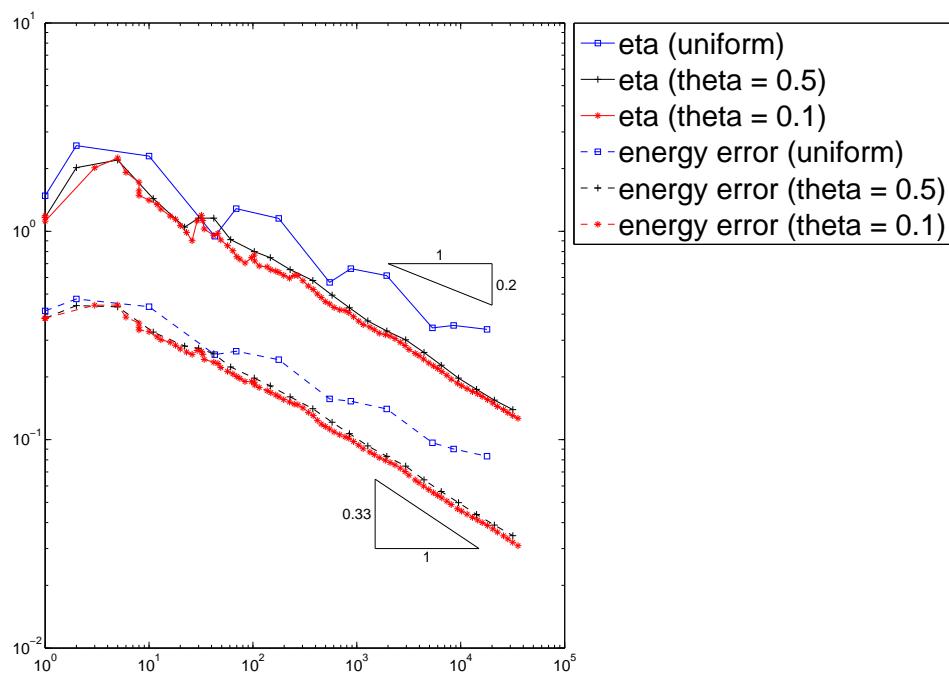


Abbildung 5.9: Konvergenzgraph der P₁-CFEM mit Fehlerschätzer und exaktem Energiefehler

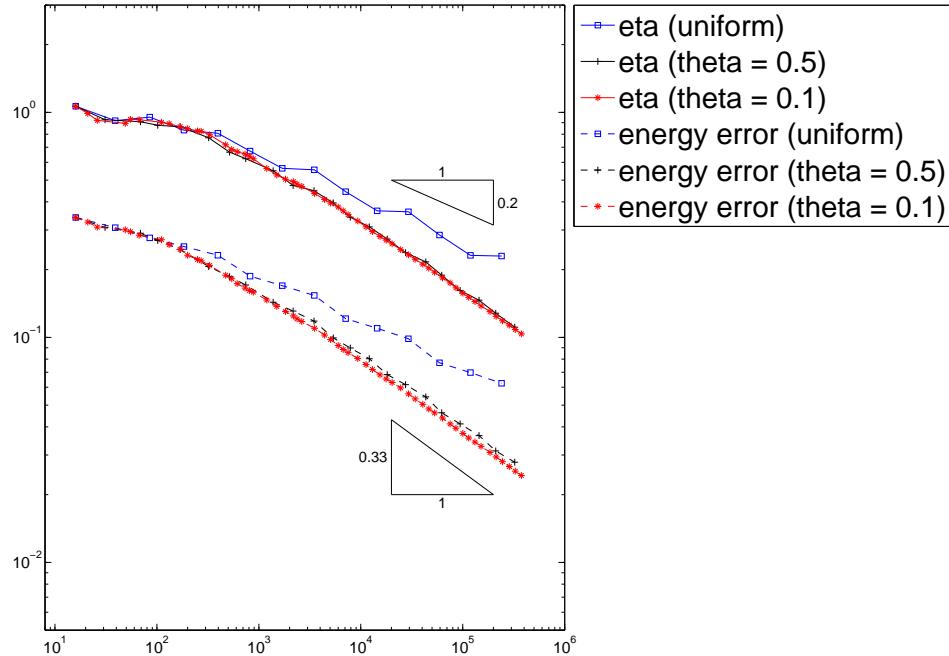


Abbildung 5.10: Konvergenzgraph der CR-NCFEM mit Fehlerschätzer und exaktem Energiefehler

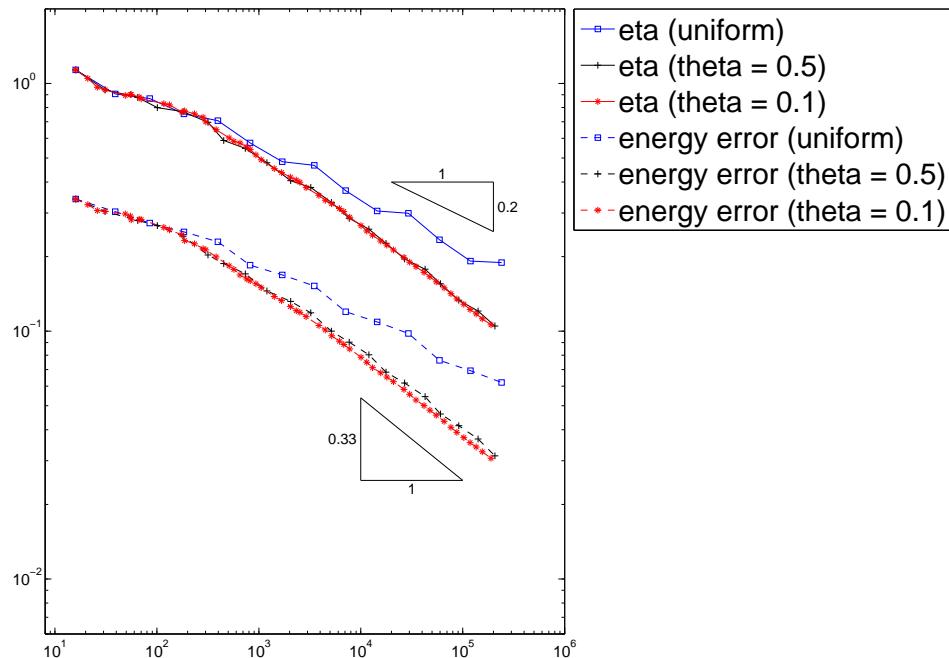


Abbildung 5.11: Konvergenzgraph der RT₀-MFEM mit Fehlerschätzer und exaktem Energiefehler

Level	eta4lvl	eta4lvlMarini	energy4lvl	energy4lvlMarini
<i>uniform</i>				
11	0.30549	0.30549	0.109	0.109
12	0.29921	0.29921	0.097869	0.097869
13	0.23383	0.23383	0.076297	0.076297
14	0.19175	0.19175	0.069307	0.069307
15	0.18909	0.18909	0.062045	0.062045
$\theta = 0.5$				
18	0.17793	0.17793	0.054436	0.054436
19	0.15533	0.15533	0.046348	0.046348
20	0.13339	0.13339	0.041537	0.041537
21	0.12051	0.12051	0.036716	0.036716
22	0.10492	0.10492	0.031396	0.031396
$\theta = 0.1$				
53	0.12848	0.12848	0.037183	0.037183
54	0.12247	0.12247	0.0355	0.0355
55	0.11743	0.11743	0.034092	0.034092
56	0.11227	0.11227	0.032615	0.032615
57	0.10609	0.10609	0.030687	0.030687

Tabelle 5.2: Werte für Fehlerschätzer und Energiefehler der RT_0 -MFEM und der Marini-Darstellung für die jeweils fünf zuletzt berechneten Level unter uniformer und adaptiver Verfeinerung

5.3 „Cube“ mit konstanter rechter Seite

Sei $\Omega = (0, 1)^3$ wieder der Einheitswürfel wie in Abbildung 5.1. Wir betrachten nun die rechte Seite $f \equiv 1$. Für den Dirichlet-Rand wählen wir $u_D = 0$ und der Neumann-Rand sei leer. Da wir keine exakte Lösung für dieses Problem kennen, können wir keinen exakten Fehler berechnen. Eine Approximation der Lösung durch die CR-NCFEM ist in Abbildung 5.12 dargestellt.

Wie man in den Abbildungen 5.13 – 5.15 erkennen kann, konvergiert der Fehlerschätzer ab etwa 1000 Freiheitsgraden mit der erwarteten Rate von $1/3$. Uniforme und adaptive Verfeinerung liefern nahezu identische Ergebnisse und wir erhalten wieder eine fast gleichmäßige Verfeinerung, wenn wir adaptiv rechnen. Wir bemerken, dass die Fehlerschätzer in allen drei Methoden ungefähr die gleiche Größenordnung besitzen. Es fällt auf, dass die konforme und die gemischte Methode anfangs eine wesentlich schlechtere Konvergenzrate als die insgesamt zu beobachtenden $1/3$ besitzen, diese dann allerdings sprunghaft größer wird. Die Werte in Tabelle 5.3 liefern wieder sowohl für die Lösung der RT_0 -MFEM als auch die Lösung der Marini-Darstellung die gleichen Ergebnisse.

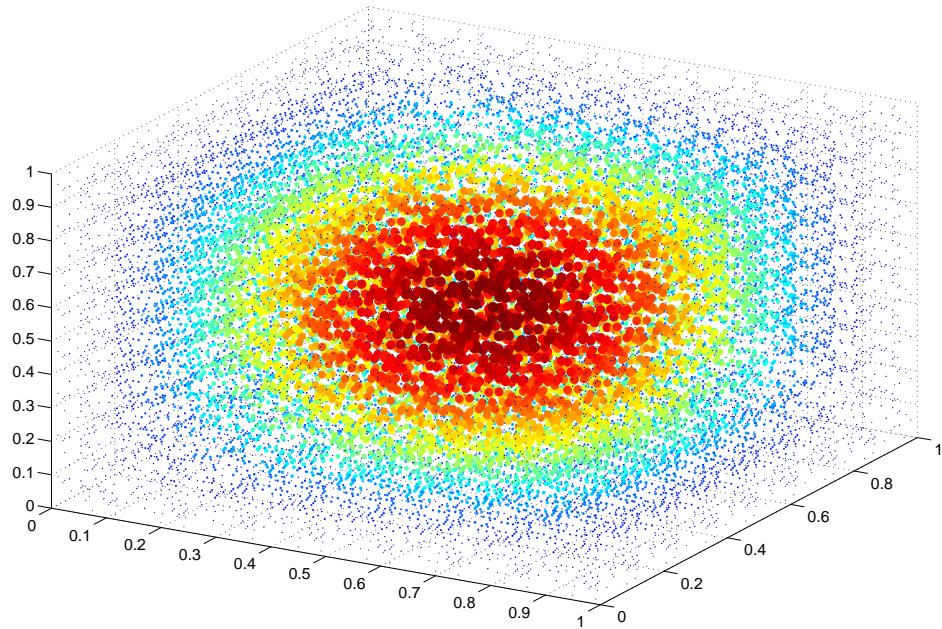


Abbildung 5.12: Uniforme Lösung der CR-NCFEM mit 19 520 Freiheitsgraden

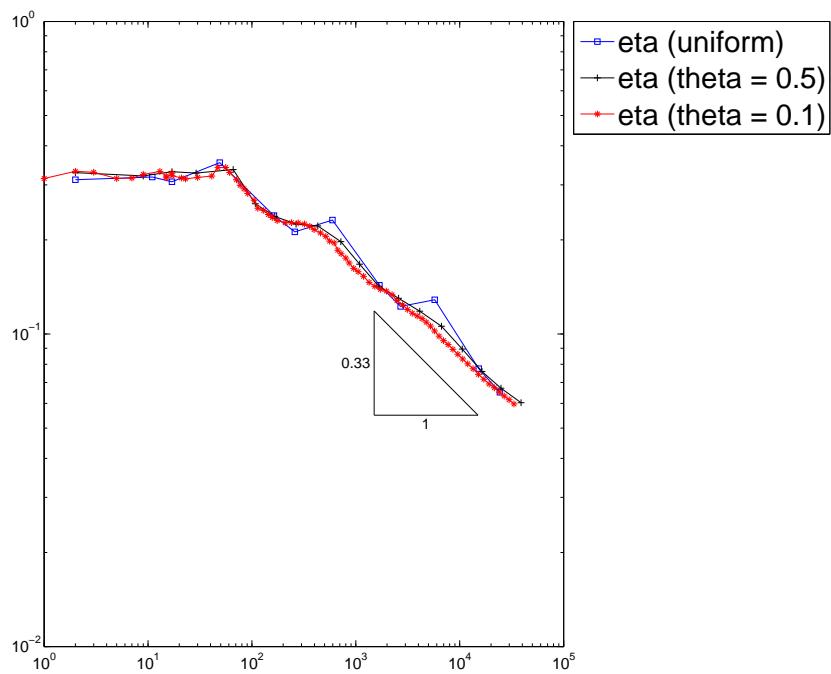


Abbildung 5.13: Konvergenzgraph der P₁-CFEM

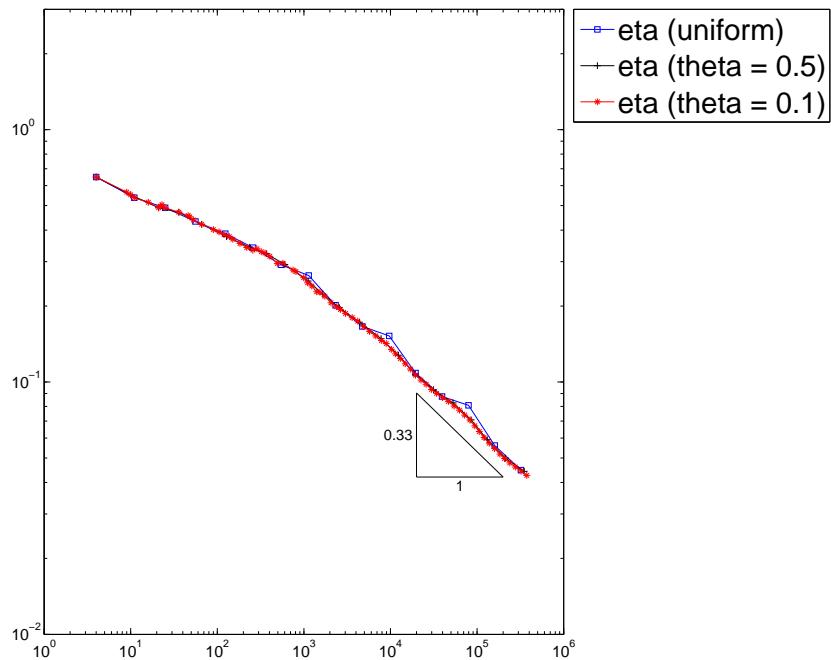


Abbildung 5.14: Konvergenzgraph der CR-NCFEM

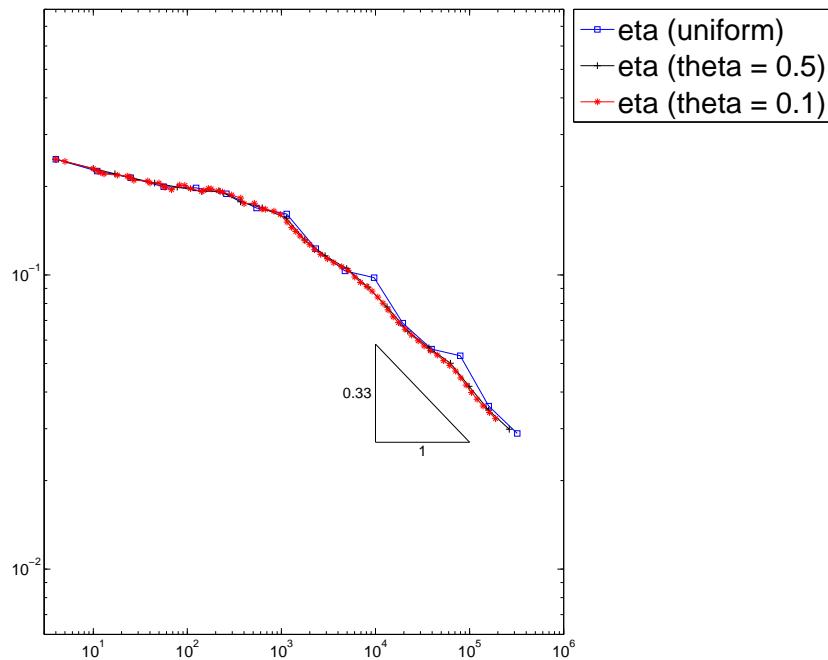


Abbildung 5.15: Konvergenzgraph der RT_0 -MFEM

Level	eta4lvl	eta4lvlMarini
<i>uniform</i>		
13	0.068573	0.068573
14	0.055974	0.055974
15	0.053101	0.053101
16	0.035822	0.035822
17	0.028922	0.028922
$\theta = 0.5$		
19	0.056486	0.056486
20	0.050099	0.050099
21	0.041796	0.041796
22	0.034682	0.034682
23	0.029876	0.029876
$\theta = 0.1$		
68	0.039882	0.039882
69	0.037783	0.037783
70	0.035857	0.035857
71	0.034069	0.034069
72	0.032472	0.032472

Tabelle 5.3: Werte für Fehlerschätzer und Energiefehler der RT_0 -MFEM und der Marini-Darstellung für die jeweils fünf zuletzt berechneten Level unter uniformer und adaptiver Verfeinerung

5.4 „L-Shape“ mit konstanter rechter Seite

Wir betrachten erneut das L-Gebiet $\Omega = (0, 1)^3 \setminus ([0, 1] \times [-1, 0] \times [0, 1])$ aus Abbildung 5.6. Für die rechte Seite wählen wir $f \equiv 1$. Die Dirichlet-Randdaten seien wie im vorherigen Beispiel $u_D = 0$ und der Neumann-Rand wieder leer. Auch für dieses Beispiel kennen wir keine exakte Lösung und können daher unsere Fehlerschätzer nicht mit dem Energiefehler vergleichen. Zur Veranschaulichung ist in Abbildung 5.17 die Approximationslösung der CR-NCFEM zu sehen.

Die „convergence history plots“ (Abbildungen 5.18 – 5.20) zeigen auch bei diesem Beispiel für eine adaptive Verfeinerungsstrategie langfristig die erwartete Konvergenzrate von $1/3$. Unter uniformer Verfeinerung weisen die Fehlerschätzer zunächst die gleiche Konvergenzrate wie unter adaptiver Verfeinerung auf. Es lässt sich jedoch beobachten, dass die Singularität an der einspringenden Kante durch die uniforme Verfeinerung nicht optimal aufgelöst werden kann, was zu einer geringfügig schlechteren Konvergenzrate bei der uniformen Verfeinerungsstrategie führt. Das der adaptive Algorithmus die Singularität an der einspringenden Kante erkennt, können wir in dem Oberflächennetz in Abbildung 5.16 erkennen. Es lässt sich beobachten, dass der vorasymptotische Bereich bei allen drei Methoden sehr groß ist und dass es auch unter adaptiver Verfeinerung sehr lange dauert, bis an der entsprechenden Kante verfeinert wird. Auch hier stellen wir wieder fest, dass die Werte der Fehlerschätzer in Tabelle 5.4 für beide Methoden übereinstimmen.

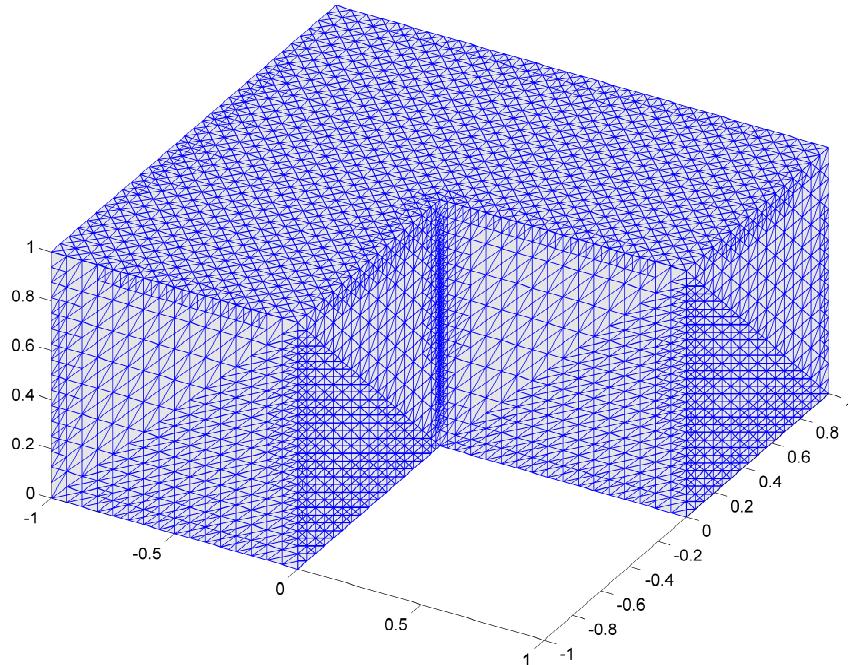


Abbildung 5.16: Oberfläche des adaptiv verfeinerten Netzes mit „bulk“-Parameter $\theta = 0.1$

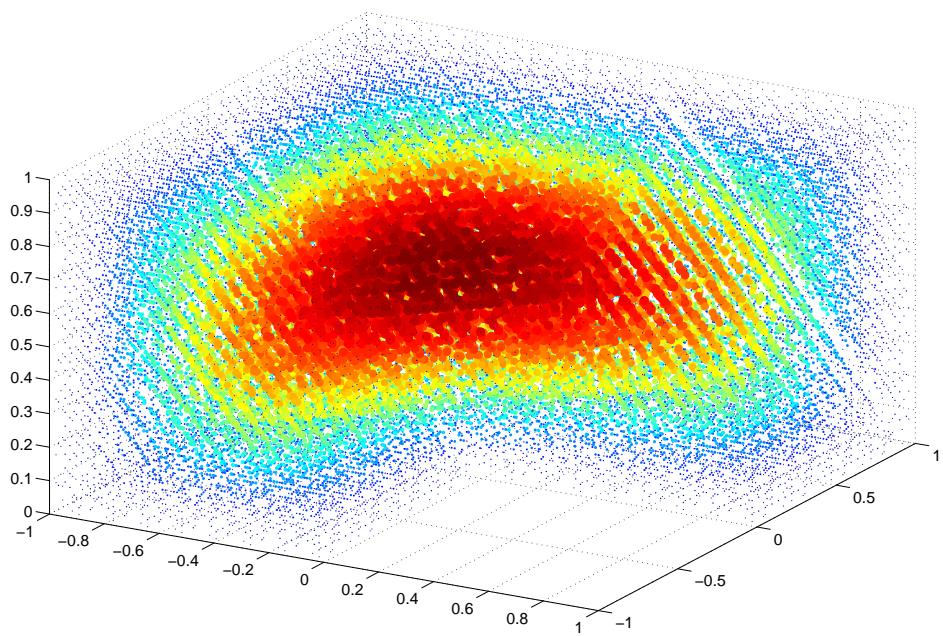


Abbildung 5.17: Uniforme Lösung der CR-NCFEM mit 29 376 Freiheitsgraden

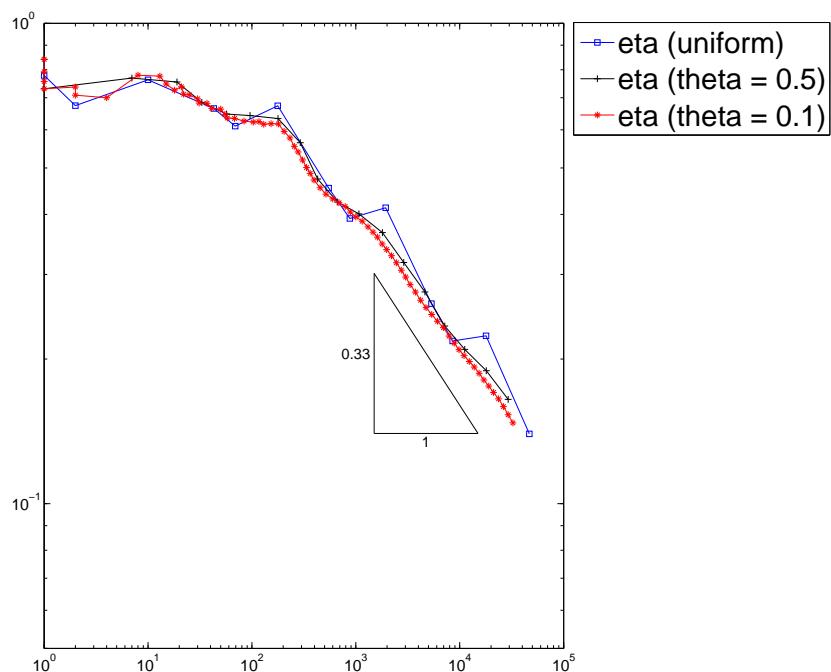


Abbildung 5.18: Konvergenzgraph der P₁-CFEM

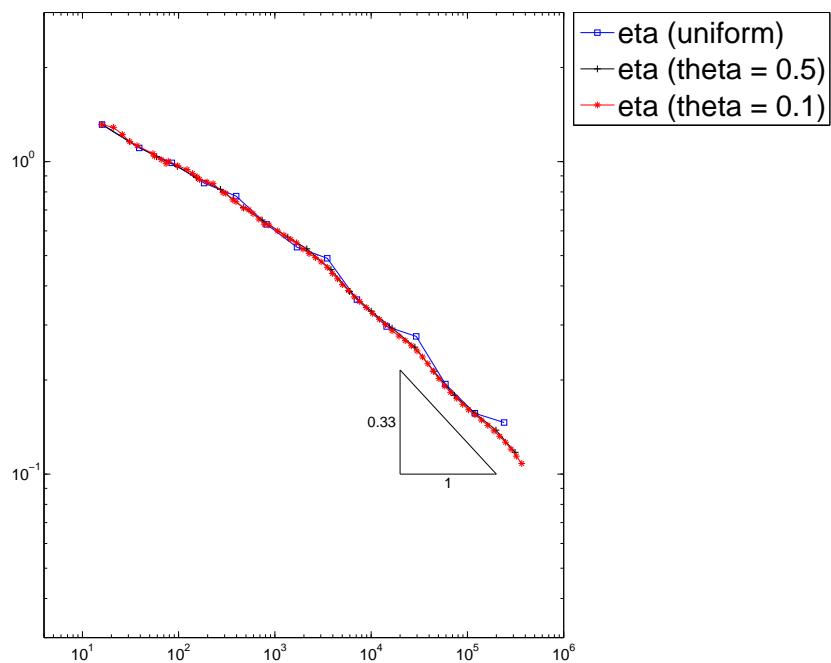


Abbildung 5.19: Konvergenzgraph der CR-NCFEM

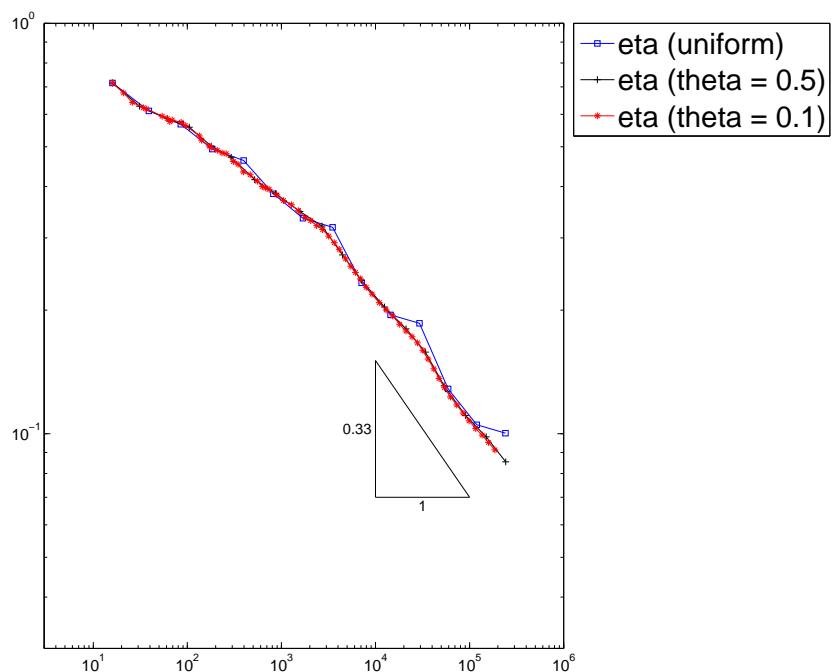


Abbildung 5.20: Konvergenzgraph der RT₀-MFEM

Level	eta4lvl	eta4lvlMarini
<i>uniform</i>		
11	0.1948	0.1948
12	0.18572	0.18572
13	0.12863	0.12863
14	0.10513	0.10513
15	0.10031	0.10031
$\theta = 0.5$		
16	0.15806	0.15806
17	0.13082	0.13082
18	0.11087	0.11087
19	0.098239	0.098239
20	0.085415	0.085415
$\theta = 0.1$		
60	0.10748	0.10748
61	0.10296	0.10296
62	0.099254	0.099254
63	0.09524	0.09524
64	0.09131	0.09131

Tabelle 5.4: Werte für Fehlerschätzer und Energiefehler der RT_0 -MFEM und der Marini-Darstellung für die jeweils fünf zuletzt berechneten Level unter uniformer und adaptiver Verfeinerung

5.5 „Fichera-Corner“ mit konstanter rechter Seite

Abschließend betrachten wir das Gebiet $\Omega = (-1, 1)^3 \setminus [0, 1]^3$. Dieses Gebiet wird auch als die Fichera-Corner bezeichnet. Wir wählen erneut $f \equiv 1$ mit reinen Dirichlet-Nullranddaten $u_D = 0$. Wie zuvor werden wir den Energiefehler nicht berechnen, da keine exakte Lösung u für dieses Problem bekannt ist. Eine Approximation der Lösung durch die CR-NCFEM ist in Abbildung 5.23 dargestellt.

Wie im vorherigen Beispiel erkennen wir zunächst keinen Unterschied zwischen einer uniformen und einer adaptiven Verfeinerungsstrategie. Erst sehr spät lässt sich erkennen, dass wir unter uniformer Verfeinerung eine etwas schlechtere Konvergenzrate als die im adaptiven Fall erreichten $1/3$ erhalten (Abbildungen 5.24 – 5.26). Wir bemerken jedoch, dass sich der Unterschied von uniformer und adaptiver Verfeinerung in den Konvergenzraten etwas eher einstellt, als bei dem vorherigen Beispiel. In Abbildung 5.22 lässt sich die Verfeinerung des adaptiven Algorithmus an der einspringenden Ecke nachvollziehen. Auch in diesem Beispiel ist der vorasymptotische Bereich sehr groß und insbesondere die konforme Methode weist eine deutlich schlechtere Konvergenzrate als $1/3$ im vorasymptotischen Bereich auf. Die beiden Lösungen der RT_0 -MFEM und der Marini-Darstellung liefern auch hier wieder für alle Verfeinerungsarten nahezu identische Werte für den Fehlerschätzer (siehe Tabelle 5.5).

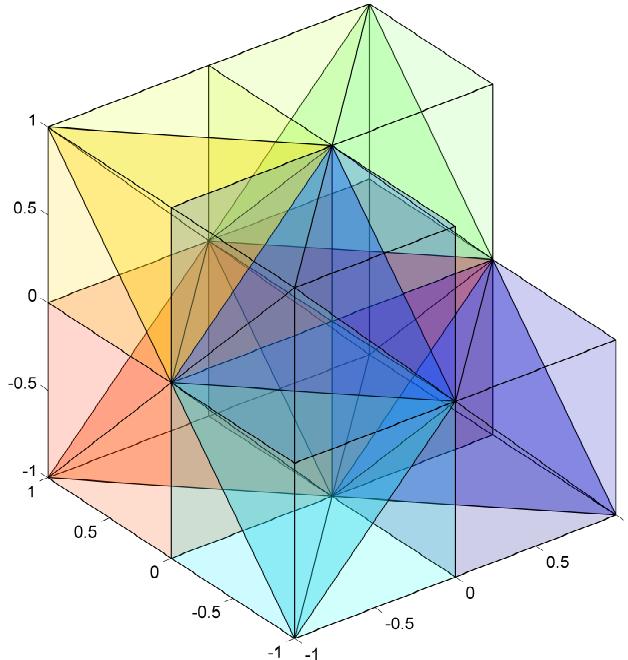


Abbildung 5.21: Anfangs-Triangulierung der „Fichera-Corner“

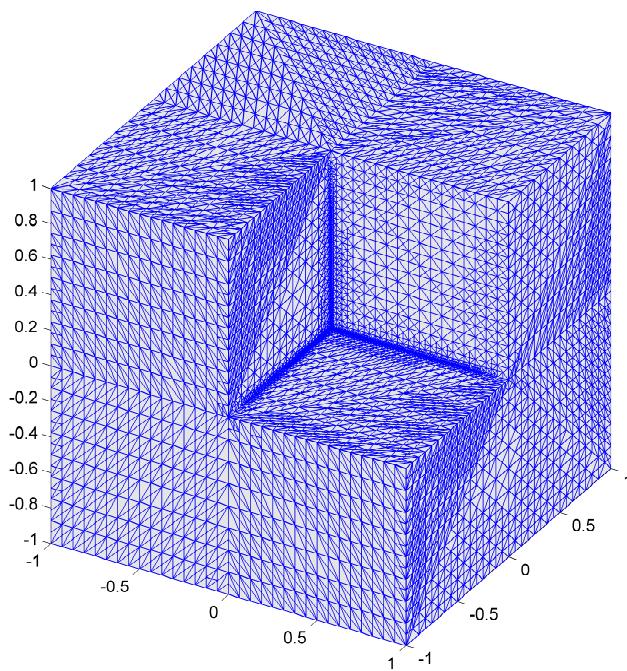


Abbildung 5.22: Oberfläche des adaptiv verfeinerten Netzes mit „bulk“-Parameter $\theta = 0.1$

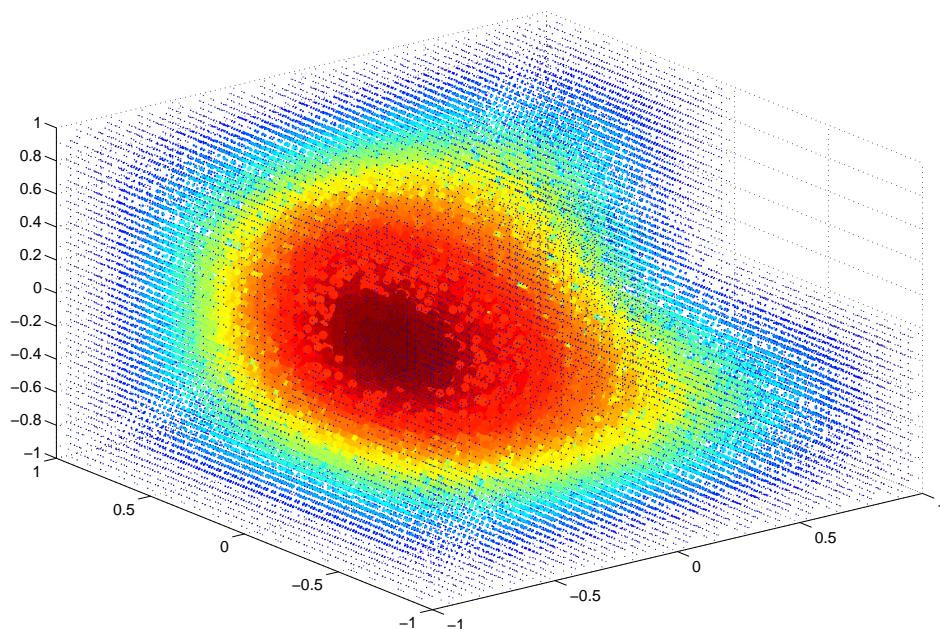


Abbildung 5.23: Uniforme Lösung der CR-NCFEM mit 69 568 Freiheitsgraden

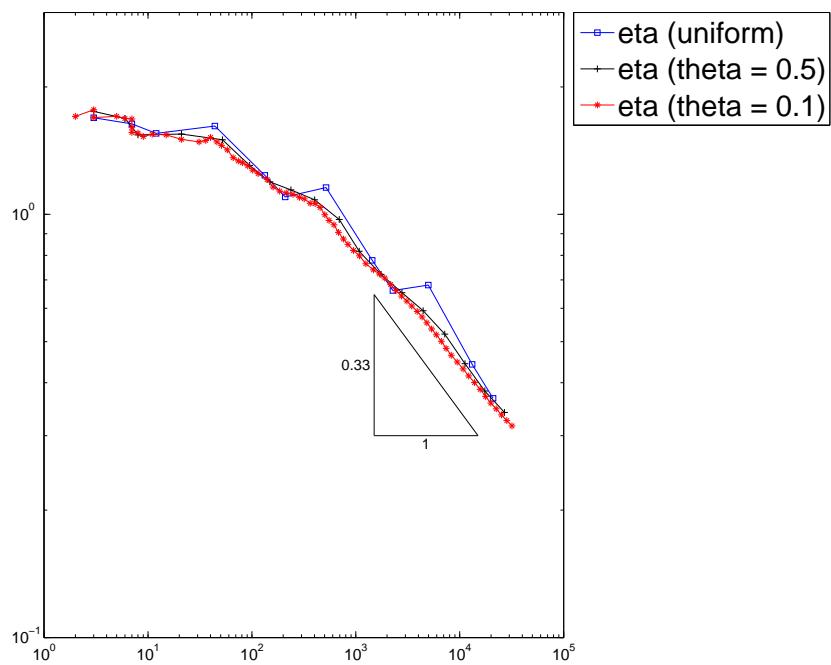


Abbildung 5.24: Konvergenzgraph der P₁-CFEM

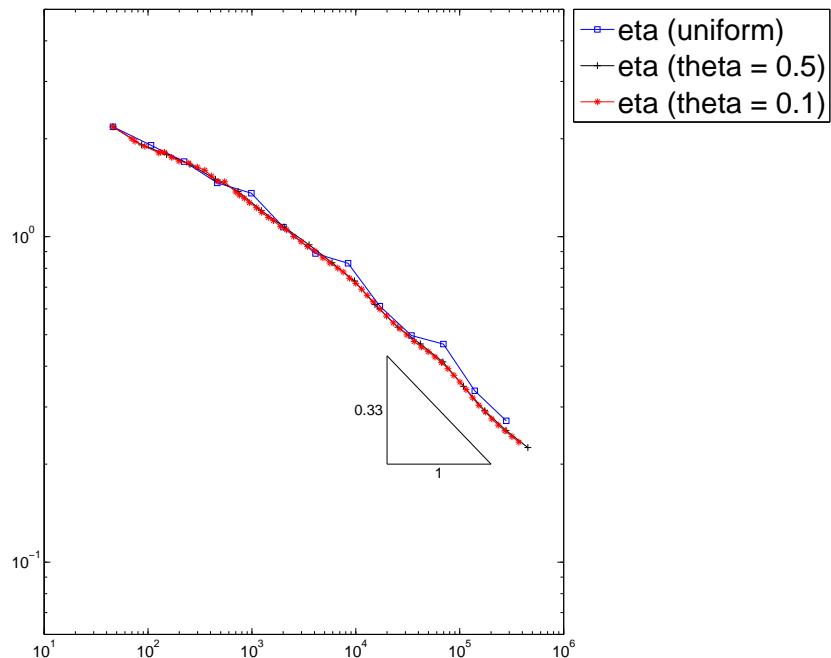


Abbildung 5.25: Konvergenzgraph der CR-NCFEM

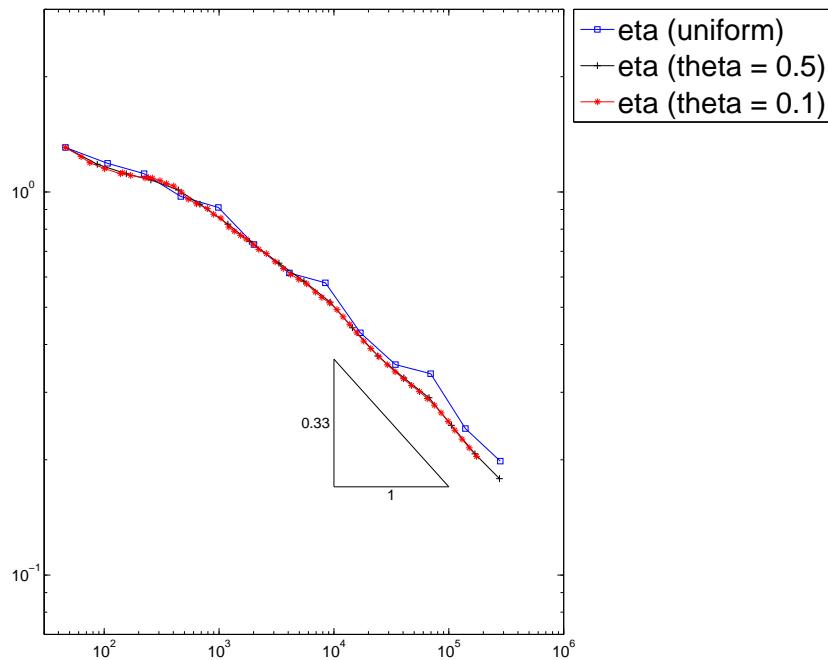


Abbildung 5.26: Konvergenzgraph der RT₀-MFEM

Level	eta4lvlRT0	eta4lvlMarini
<i>uniform</i>		
10	0.42888	0.42888
11	0.35428	0.35428
12	0.33547	0.33547
13	0.24119	0.24119
14	0.19827	0.19827
$\theta = 0.5$		
15	0.32699	0.32699
16	0.29026	0.29026
17	0.24607	0.24607
18	0.20731	0.20731
19	0.17845	0.17845
$\theta = 0.1$		
51	0.25185	0.25185
52	0.23892	0.23892
53	0.22636	0.22636
54	0.2148	0.2148
55	0.20399	0.20399

Tabelle 5.5: Werte für Fehlerschätzer und Energiefehler der RT₀-MFEM und der Marini-Darstellung für die jeweils fünf zuletzt berechneten Level unter uniformer und adaptiver Verfeinerung

6 Auswertung

Zunächst stellen wir fest, dass alle untersuchten Lösungsmethoden qualitativ unterschiedliche Lösungen für die gewählten Probleme liefern. Ein Beispiel dafür findet sich in den Konvergenzraten der Lösungen im vorasymptotischen Bereich. Während die konforme Methode anfänglich teilweise nur sehr schlecht konvergiert, erreicht die nicht-konforme Methode von Anfang an nahezu optimale Konvergenzraten.

6.1 Sprunghafte Verhalten der konformen Methode

Die Ursache des sprunghaften Verhaltens der konformen im Gegensatz zur nicht-konformen und gemischten Methode ist wahrscheinlich die in dieser Arbeit verwendete Verfeinerungsvorschrift für eine dreidimensionale Triangulierung. Man sieht leicht ein, dass die in [Ste08] vorgestellte Art der Verfeinerung nur sehr unregelmäßig neue innere Knoten erzeugt, was dazu führen kann, dass sich die Lösung anfangs nur sehr wenig verbessert obwohl die Triangulierung immer feiner wird. Zur Veranschaulichung verweisen wir an dieser Stelle auf die Geometrie des „Cube“ aus Abschnitt 5, der erst nach viermaliger uniformer Verfeinerung die ersten inneren Knoten aufweist. Eine alternative Erklärung wäre ein sehr langes vorasymptotisches Verhalten, was auch die Unregelmäßigkeiten der Konvergenz bei über 10^4 Freiheitsgraden in Abbildung 5.3 erklären würde.

6.2 Adaptivität

Die Ergebnisse der numerischen Beispielrechnungen zeigen, dass die adaptive Verfeinerungsstrategie immer optimale und damit im Vergleich zur uniformen oft bessere Konvergenzraten liefert. Ein markanter Unterschied bei der Wahl des „bulk“-Parameters fällt für die beiden in dieser Arbeit gewählten Werte $\theta = 0.5$ und $\theta = 0.1$ nicht auf. Dabei können die gleichen Konvergenzraten aus den Abschnitten 5.1 und 5.3 auf die Glattheit der Lösungen zurückgeführt werden. Dass die Fehlerschätzer in den letzten beiden Beispielen (Abschnitt 5.4 und 5.5) unter uniformer Verfeinerung anfangs nahezu die gleichen Konvergenzraten aufweisen wie unter adaptiver Verfeinerung, ist ein aus dem zweidimensionalen Fall gut bekanntes Phänomen. Auch dort gibt es Beispiele, wie z.B. das 2D-Lshape mit konstanter rechter Seite $f \equiv 1$ und reinen Dirichlet-Nullranddaten, bei denen die Fehlerschätzer unter uniformer Verfeinerung in einem sehr großen vorasymptotischen Bereich besser konvergieren als erwartet. An dieser Stelle bemerken wir, dass die optimale Konvergenzrate bezüglich der Freiheitsgrade einer Methode in den gewählten Beispielen stets bei $1/3$ liegt, da a priori Fehlerabschätzungen eine Konvergenz bzgl. h_T erwarten lassen und im \mathbb{R}^3 der Zusammenhang $nrDoF \approx h_T^3$ gilt (vgl. [Bra07]).

6.3 Eigenschaften der Fehlerschätzer

Wir können weiterhin beobachten, dass alle drei untersuchten Fehlerschätzer stets die gleiche Konvergenzordnung wie der exakte Fehler besitzen. Dies belegt die in der Theorie gezeigten Eigenschaften der Fehlerschätzer, wie beispielsweise deren Effizienz und Zuverlässigkeit.

6.4 Marini-Identität

Auch die Erkenntnisse aus Abschnitt 3.3.3 können wir durch unsere numerischen Beispiele bestätigen, denn sowohl die RT_0 -gemischte Methode als auch die CR-nicht-konforme Methode zusammen mit den Gleichungen (3.13) und (3.14) liefern uns, zumindest für den Fehlerschätzer sowie den Energiefehler die selben Ergebnisse (vgl. Tabelle 5.2 – 5.5). Das die Werte aus Tabelle 5.1 nicht übereinstimmen lässt sich damit erklären, dass die rechte Seite f in diesem Beispiel keine Funktion aus $P_0(\mathcal{T})$ ist und daher die Voraussetzungen für Satz 3.9 nicht erfüllt werden.

6.5 Fazit und Ausblick

Ziel dieser Arbeit war es, das Poisson-Problem im dreidimensionalen Raum zu untersuchen. Wir konnten drei der im Zweidimensionalen bereits viel untersuchten Methoden auch im dreidimensionalen Fall umsetzen und unsere theoretischen Untersuchungen zu diesen Methoden durch numerischen Beispielrechnungen bestätigen.

Ein möglicher nächster Schritt wäre die Untersuchung anderer im zweidimensionalen bekannter Lösungsmethoden, wie etwa der Least-Squares-Methode für den 3D-Fall. Außerdem bieten die in dieser Arbeit erzielten Ergebnisse einen Anreiz, sich mit anderen für die Praxis wichtigen Problemen, wie etwa der linearen Elastizität oder den Maxwell-Gleichungen in drei Raumdimensionen, zu beschäftigen.

Abschließend weisen wir darauf hin, dass die starke Verzögerung bei der Erreichung von optimalen Konvergenzraten bei der konformen Finite-Elemente-Methode problematisch ist und es zudem sein kann, dass das Netz bereits sehr fein sein muss, bevor überhaupt die Konvergenz der numerischen Lösung erkennbar wird. Dies lässt die Methode für die Praxis untauglich erscheinen. Ein Vorteil der konformen Methode ist der vergleichsweise geringe Rechenaufwand, da das zu lösende Gleichungssystem kleiner als bei den beiden anderen Methoden ist. Da die Fehler von allen drei Methoden in etwa die gleiche Größenordnung besitzen, ist die konforme Methode der nicht-konformen und gemischten Methode also vorzuziehen. Außerdem bemerken wir, dass die Marini-Darstellung der RT_0 -gemischten Finite-Elemente-Methode und die teilweise schlechteren Konvergenzraten im vorasymptotischen Bereich den höheren Rechenaufwand gegenüber der CR-nicht-konformen Methode (vgl. Größe der Gleichungssysteme) nicht rechtfertigen.

Danksagung

Ich möchte mich sehr bei Prof. Carstensen und Mira Schedensack für die hilfreiche und umfangreiche Betreuung dieser Arbeit und die erhellenden und anregenden Gespräche bedanken.

Literatur

- [BC05] BAHRIAWATI, C. ; CARSTENSEN, C.: *Three MATLAB implementations of the lowest-order Raviart-Thomas MFEM with a posteriori error control.* 2005
- [BF91] BREZZI, F. ; FORTIN, M.: *Mixed and Hybrid Finite Element Methods.* Springer-Verlag, 1991
- [Bra07] BRAESS, D.: *Finite elements - theory, fast solvers, and applications in solid mechanics.* Cambridge University Press, New York, 2007
- [Car05] CARSTENSEN, C.: *A unifying theory of a posteriori finite element error control.* Springer-Verlag, 2005
- [Car09a] CARSTENSEN, C.: *AFEM-Softwarepaket der Arbeitsgruppe 'Numerische Analysis'.* Institut für Mathematik der Humboldt-Universität zu Berlin, 2009
- [Car09b] CARSTENSEN, C.: *Yonsei Lectures at the WCU Department Computational Science and Engineering on Finite Element Method.* 2009. – lokal verfügbar
- [CGK⁺10] C. CARSTENSEN ; GEDICKE, J. ; KERN, L. ; NEUMANN, J. ; RABUS, H. ; ROZOVA, M.: *AFEM-Dokumentation.* 2010. – lokal verfügbar
- [Eva10] EVANS, L. C.: *Partial Differential Equations, Graduate Studies in Mathematics.* American Mathematical Society, Providence, 2010. – 2. Auflage
- [GS11] GALLISTL, D. ; SCHEDENSACK, M.: *Verfeinerungsroutine für 3D-Simplices.* Institut für Mathematik der Humboldt-Universität zu Berlin, 2011
- [Mar85] MARINI, L. D.: *An inexpensive method for the evaluation of the solution of lowest order Raviart-Thomas mixed method.* SIAM J. Numer. Anal., 22 (1985). – S. 493-496
- [Ste08] STEVENSON, R.: *The completion of locally refined simplicial partitions created by bisection.* Math. Comp., 77 (2008). – S. 227-241

Anhang: Verzeichnisstruktur

```
/  
  common  
    computeArea4f.m  
    computeE4f.m  
    computeF4e.m  
    computeF4n.m  
    computeLength4s.m  
    computeMid4e.m  
    computeMid4f.m  
    computeMid4s.m  
    computeN4f.m  
    computeN4s.m  
    computeNormal4e.m  
    computeNormal4f.m  
    computeS4e.m  
    computeS4f.m  
    computeS4n.m  
    computeTangent4s.m  
    computeVolume4e.m  
    PONormalJump.m  
    POTangentJump.m  
  estimate  
    estimateCREtaElements.m  
    estimateCREtaFaces.m  
    estimateP1EtaElements.m  
    estimateP1EtaFaces.m  
    estimateRTOEtaElements.m  
    estimateRTOEtaFaces.m  
  geometries  
    Cube  
      Cube.m  
      Cube.mat  
    FicheraCorner  
      FicheraCorner.m  
      FicheraCorner.mat  
    varLshape3D  
      varLshape3D.m  
      varLshape3D.mat  
  integrate  
    error4eCREnergy3D.m  
    error4eP1Energy3D.m
```

```
error4eRT0Energy3D.m
integrate3D.m
mark
markBulk3D.m
plot
plot3D.m
plotConvergence.m
plotError.m
plotSurface.m
scatterPlotCR.m
scatterPlotP1.m
scatterPlotRT0.m
problems
Cube
afemCRCubeAdaptive.m
afemCRCubeUniform.m
afemP1CubeAdaptive.m
afemP1CubeUniform.m
afemRT0CubeAdaptive.m
afemRT0CubeUniform.m
CubeExact
afemCRCubeExactAdaptive.m
afemCRCubeExactUniform.m
afemP1CubeExactAdaptive.m
afemP1CubeExactUniform.m
afemRT0CubeExactAdaptive.m
afemRT0CubeExactUniform.m
FicheraCorner
afemCRFicheraCornerAdaptive.m
afemCRFicheraCornerUniform.m
afemP1FicheraCornerAdaptive.m
afemP1FicheraCornerUniform.m
afemRT0FicheraCornerAdaptive.m
afemRT0FicheraCornerUniform.m
Lshape
afemCRLshapeAdaptive.m
afemCRLshapeUniform.m
afemP1LshapeAdaptive.m
afemP1LshapeUniform.m
afemRT0LshapeAdaptive.m
afemRT0LshapeUniform.m
LshapeExact
afemCRLshapeExactAdaptive.m
afemCRLshapeExactUniform.m
```

```
    └── afemP1LshapeExactAdaptive.m
        └── afemP1LshapeExactUniform.m
            └── afemRT0LshapeExactAdaptive.m
                └── afemRT0LshapeExactUniform.m
        refine
        └── refine3D.m
    solve
    └── solveCRPoisson3D.m
        └── solveP1Poisson3D.m
            └── solveRT0Marini3D.m
                └── solveRT0Poisson3D.m
    afem3d.m
    seePlots.m
```

Selbstständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe und ich zum ersten Mal eine Bachelorarbeit in diesem Studiengang einreiche.

Berlin, den 29. September 2014